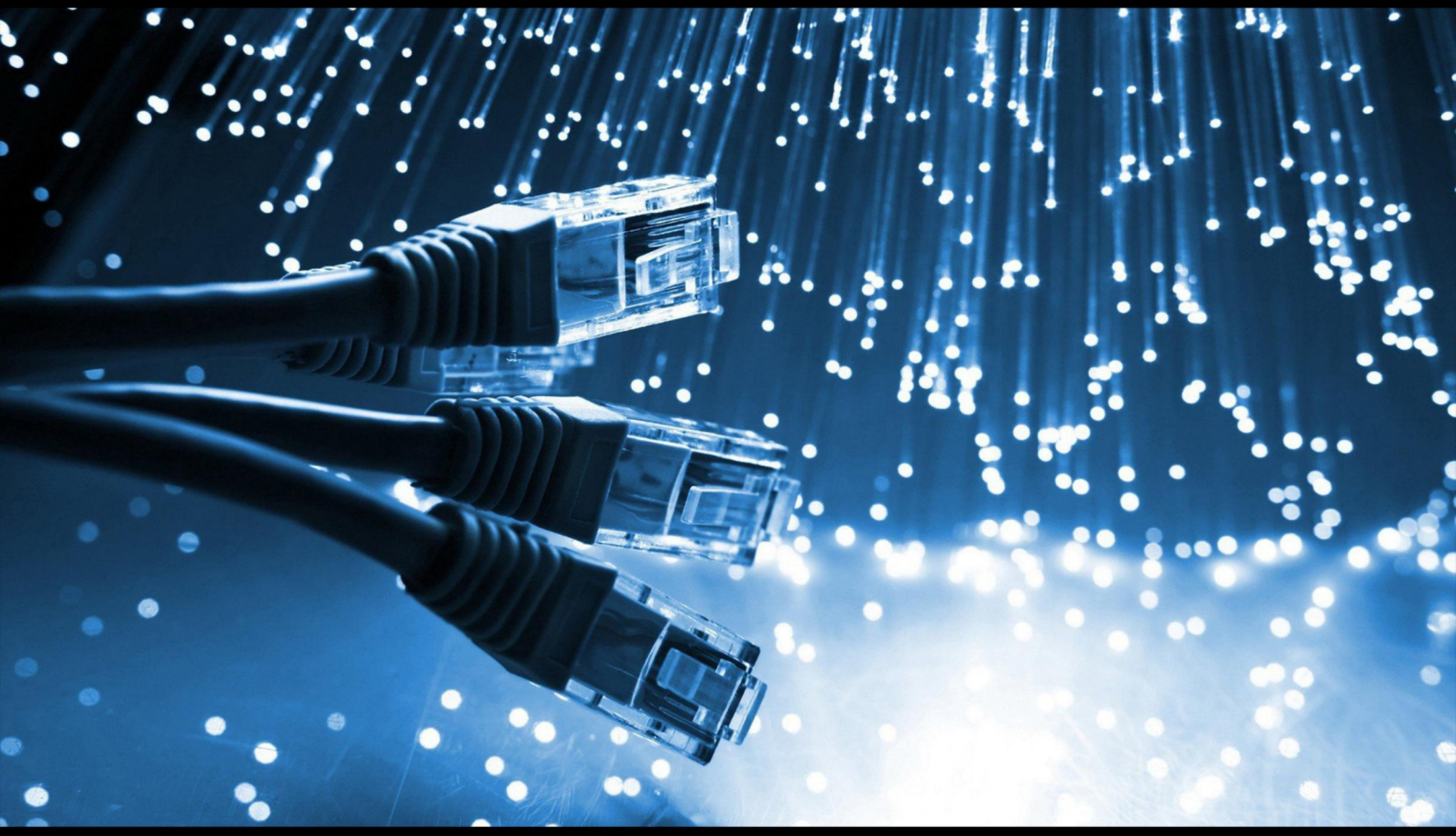
JCSI

Journal of Computer Sciences Institute

Volume 27/2023



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

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

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

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

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

Recenzenci numeru:

dr inż. Małgorzata Plechawska-Wójcik
dr hab. Paweł Kaczmarek, prof. uczelni
dr inż. Marek Miłosz, prof. uczelni
dr inż. Krzysztof Dziedzic
dr inż. Jakub Smołka
dr Paweł Powroźnik
dr Mariusz Dzieńkowski
dr inż. Maciej Pańczyk
dr inż. Tomasz Nowicki
dr inż. Grzegorz Kozieł

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

Projekt okładki:
Marta Zbańska

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

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

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

Reviewers:

Małgorzata Plechawska-Wójcik
Paweł Kaczmarek
Marek Miłosz
Krzysztof Dziedzic
Jakub Smołka
Paweł Powroźnik
Mariusz Dzieńkowski
Maciej Pańczyk
Tomasz Nowicki
Grzegorz Kozieł

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 27 (2023) 100–103
Received: 16 October 2022
Accepted: 26 October 2022

A comparative analysis of the measurement tools of fuel consumption in a passenger car

Analiza porównawcza narzędzi do pomiaru zużycia paliwa w samochodzie osobowym

Jakub Grzesiak, Karol Sawczuk*, Marcin Barszcz

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

Abstract
The article presents the results of practical application of the author's application for a mobile device that allows measuring fuel consumption in a passenger car. The application has two modules and works while driving the car. One of the modules retrieves data from the on-board computer using the OBD II protocol. The other module uses the phone's GPS and data entered by the user. Simultaneous data collecting from the modules allows them to be compared. Based on the results, it is concluded that the inaccuracy of GPS navigation in relation to OBD II for gasoline cars does not exceed 6%, and for those with diesel engines, 21%. The results of the work will make it easier for users to appropriately select a technique for testing the fuel consumption of a passenger car.
Keywords: OBD; GPS; Car; Fuel consumption

Streszczenie
W artykule zaprezentowano efekty praktycznego zastosowania autorskiej aplikacji na urządzenie mobilne pozwalającej na pomiar zużycia paliwa w samochodzie osobowym. Aplikacja posiada dwa moduły i pracuje w trakcie jazdy samochodem. Jeden z modułów pobiera dane z komputera pokładowego za pomocą protokołu OBD II. Drugi zaś, wykorzystuje GPS telefonu oraz dane wprowadzone przez użytkownika. Równoczesne pobieranie danych z modułów pozwala na ich zestawienie. Na podstawie otrzymanych wyników badań wnioskuje się, że niedokładność nawigacji GPS w stosunku do OBD II dla samochodów benzynowych nie przekracza 6%, a dla tych z silnikiem diesla, 21%. Rezultaty pracy ułatwią użytkownikom odpowiedni dobór techniki badania zużycia paliwa przez samochód osobowy.
Słowa kluczowe: OBD; GPS; Zużycia paliwa

*Corresponding author
Email address: **karol.sawczuk@pollub.edu.pl** (K. Sawczuk)

1. Wstęp

W obecnych czasach poruszanie się samochodem zajmuje znaczną część życia, co jednocześnie składa się na istotną część wydatków z budżetu kierowcy. Taki stan rzeczy prowadzi do skupienia uwagi na aspekty związane ze spalaniem bądź wydajnością samochodu, którym się poruszamy. Do dyspozycji jest wiele narzędzi, które mogą pomóc w ocenie ilości spalanego paliwa przez samochód. Nowe samochody posiadają wyświetlacze które informują o chwilowym i średnim zużyciu paliwa lub drodze jaką jeszcze samochód może przebyć do kolejnego tankowania. Stosując odpowiednie obliczenia i wzory istnieje możliwość wyznaczenia, z pewną dokładnością, ile paliwa zostało zużyte podczas pracy silnika [1]. Takie rozwiązanie jednak wymaga od użytkownika dodatkowej pracy, której mógłby uniknąć.

W celu rozwiązania tego problemu autorzy niniejszej pracy opracowali mobilną aplikację CarRide na smartfony, która za pomocą interfejsu On-Board Diagnostics (OBD II) pobiera na bieżąco informacje bezpośrednio z komputera pokładowego samochodu i wyznacza dzięki nim ilość spalonego paliwa. Według Statista, 83,72% populacji świata posiada smartfona co znacząco ułatwia dostępność do aplikacji [2]. Natomiast interfejs OBD II jest specjalistycznym urządzeniem diagnostycznym, i nie jest tak popularny jak nowoczesne telefony komórkowe co powoduje, że dostępność opracowanej aplikacji znacząco spada. Rozwiązaniem tego problemu było dodanie opcji estymacji spalania paliwa wykorzystując tylko narzędzia dostępne w samym telefonie.

Obecnie telefony wyposażone są w moduły Global Positioning System (GPS), które są w stanie monitorować miejsce oraz prędkość z jaką się poruszamy. Nie są to jednak wszystkie wystarczające informacje, które są potrzebne do określenia zużycia paliwa. Użytkownik musi wprowadzić do aplikacji wcześniej wyznaczone dane tj. spalanie paliwa samochodu przy danej prędkości [3]. Po wprowadzeniu funkcji estymowania spalonego paliwa na podstawie GPS, użytkownik nie jest zmuszony do zakupu interfejsu OBD II.

2. Analiza literatury

Parametry samochodów osobowych są monitorowane przez komputer pokładowy. Informacje te są niezbędne, żeby otrzymać jak najdokładniejsze wyniki podczas badania zużycia paliwa. Jeśli chcemy się posłużyć estymacją, należy określić zestaw niezbędnych parametrów i wyznaczyć ich wartości. Rimpas i in. wykazali w swojej pracy, że prędkość samochodu oraz jej zmiana, bezpośrednio wpływają na zużycie paliwa [3].

Jeśli komputer silnika nie posiada odpowiedniego zestawu sensorów, część danych trzeba przyjąć jako stałe. Gumus, Masi i in. w swoich artykułach badali silniki samochodów osobowych i wyznaczali ich sprawność objętościową [4, 5]. Zebrane dane posłużyły im następnie do wykonania obliczeń, których wynikiem jest ilość spalonego paliwa. Ribeiro i in. przeanalizowali informacje otrzymane z interfejsu OBD II do obliczania zużycia paliwa, tworząc usystematyzowany sposób obliczania zużycia paliwa na podstawie odczytów czujników OBD II dla wielu różnych pojazdów [1].

Nawigacja GPS nie wskazuje idealnie prędkości z jaką porusza się samochód. Największy błąd występuje wtedy, kiedy zmiany prędkości są bardzo małe. Ali i in. w swojej pracy zwracają uwagę na problem zalecając wykonywanie pomiarów przy prędkościach większych niż 10 km/h [6]. Aplikacja „CarRide" przewiduje wykorzystywanie nawigacji o małej dokładności, dlatego pomiary zapisywane są, gdy prędkość będzie większa niż 15 km/h.

Meseguer i in. przedstawili w swoim artykule wzory, które pozwalają na przekształcenie odczytów z komputera silnika na ilość spalonego paliwa na odcinku 100 kilometrów [7].

3. Metody badawcze

Badanie spalania paliwa samochodów osobowych zostało przeprowadzone na trzech różnych autach. Pierwszy samochód to Ford Focus MK2 z silnikiem benzynowym 1,6. Drugie auto to Ford Mondeo MK4 z silnik 2,0 diesel, ostatni zaś to Mercedes W204 z silnikiem benzynowym 1,8. Prowadzone badania uwzględniały trzy różne paliwa tj. benzynę, LPG, olej napędowy.

Aplikacja stworzona przez autorów pracy o nazwie "CarRide" gromadzi symultanicznie dane z samochodu, za pomocą interfejsu OBD II oraz modułu GPS telefonu w momencie, w którym porusza się z prędkością większą niż 15 km/h [6]. Zebrane informacje pozwalają na obliczenie ilości spalonego paliwa.

3.1. OBD II

Odczyt danych z komputera pokładowego samochodu w czasie rzeczywistym odbywa się za pomocą protokołu OBD II z wykorzystaniem zewnętrznego interfejsu bluetooth. Urządzenie wpinane jest w port OBD II w samochodzie, komunikacja z aplikacją odbywa się za pomocą standardu bluetooth.

Właściwy odczyt danych z auta opiera się o komunikację szeregową. Na początku połączenia dokonuje się konfiguracja parametrów komunikacji takich jak prędkość, częstotliwość, itp. Następnie za pomocą wysłania do modułu OBD II konkretnych kodów (PIDów), w odpowiedzi otrzymuje się żądaną wartość np. obroty silnika na minutę. Sam odczyt odbywa się dwa razy w ciągu sekundy, co przekłada się na duże zagęszczenie zbieranych danych.

Dla samochodów posiadających Mass Airflow Sensor (MAF Sensor), do obliczenia zużycia paliwa, stosuje się wzór [7]:

$$Zużyte\ paliwo = \frac{MAF \cdot 3600}{V \cdot AFR} \quad (1)$$

gdzie:
- MAF - ilość powietrza jaka dostaje się do silnika [g/s],
- V - prędkość auta [km/h],
- ρ - gęstość danego paliwa [g/l],
- AFR - stała, która określa idealny stosunek ilości powietrza do spalenia danej ilości paliwa (Tabela 1).

Tabela 1: Stałe wartości wykorzystywane do obliczeń

Rodzaj paliwa	Gęstość	AFR
Benzyna	710 g/l	14,7:1
Diesel	830 g/l	14,5:1

Jeśli samochód nie posiada MAF Sensora, przyjmuje się założenie, że wydajność silnika (VE) wynosi 80% [4, 5] i do obliczenia zużycia paliwa wykorzystuje się wzór [7]:

$$MAF = IMAP \cdot \frac{VE}{100} \cdot ED \cdot \frac{MM}{R} \quad (2)$$

gdzie:
- ED - pojemność silnika [l],
- MM - molekularna masa powietrza, która jest stała i wynosi 28,97 [g/mol],
- VE - wydajność silnika [%],
- R - stała gazowa, wynosi 8,314 [(J/K)/mol]

IMAP to syntetyczna zmienna, którą można wyznaczyć wzorem:

$$IMAP = \frac{RPM \cdot \frac{1}{120}}{ITEMP} \cdot MAP \quad (3)$$

gdzie:
- RPM - liczba obrotów silnika [obr./min.]
- ITEMP - temperatura powietrza dostającego się do silnika [K]
- MAP - ciśnienie w kolektorze dolotowym silnika [kPa]

RPM jest liczbą obrotów silnika na minutę, dzielona przez 120 ze względu na próbkowanie aplikacji z OBD II, wynoszące 2 Hz.

3.2. GPS

Aplikacja CarRide wykorzystuje moduł GPS telefonu, który zakresem funkcjonalności w pełni odpowiada realizacji badań.

Uruchomienie oraz praca modułu oparta jest na systemie Android oraz tego jak ta funkcjonalność została zaimplementowana przez producenta systemu. Obliczenia w przypadku modułu GPS opierają się o pobranie aktualnej prędkości dokładnie raz na sekundę [5].

Do prawidłowego działania modułu potrzebne są dane na temat zużycia paliwa w stosunku do prędkości.

Każdy z samochodów wyposażony był we wskaźnik informujący o ilości zużytego paliwa na trasie oraz tempomat. Aby wyznaczyć prawidłowe wartości, każdym samochodem został pokonany ten sam odcinek 5 kilometrów płaskiej drogi z konkretną prędkością przy najwyższym możliwym biegu skrzyni (Tabela 2). Wszystkie pomiary zostały wykonane tego samego dnia na suchej nawierzchni przy temperaturze 23°C.

Tabela 2: Zakres biegów w skrzyni przy wyznaczaniu zużycia paliwa

Prędkość	Ford Focus MK2	Mercedes W204	Ford Mondeo MK4
40 km/h	4 bieg	4 bieg	5 bieg
60 km/h	5 bieg	5 bieg	6 bieg
90 km/h	5 bieg	5 bieg	6 bieg

Użytkownik wpisuje wartości zużycia paliwa w oparciu o interfejs użytkownika samochodu dla kolejnych prędkości: 40 km/h, 60 km/h oraz 90 km/h (Tabela 3). Następnie za pomocą metody aproksymacji wielomianowej wyznaczana jest skala wartości zużycia paliwa przez samochód.

Tabela 3: Zużycie paliwa przez samochód odczytane z komputera pokładowego przy danej prędkości

Prędkość	Zużycie paliwa		
	Ford Focus MK2	Mercedes W204	Ford Mondeo MK4
40 km/h	6 l/100km	7,5 l/100km	6,8 l/100km
60 km/h	5,1 l/100km	7,1 l/100km	5,8 l/100km
90 km/h	6,3 l/100km	6,9 l/100km	6,5 l/100km

Na podstawie prędkości, przy jednosekundowym próbkowaniu, obliczana jest trasa pokonana w ostatniej sekundzie drogi oraz zużycie paliwa które jest wyznaczane z wielomianu wyznaczonego na podstawie aproksymacji.

4. Wyniki badań

Aplikacja rejestruje dane w chmurze z tych samych odcinków. W oparciu o zaproponowaną metodykę badań zebrane zostały miarodajne wyniki. W tabeli 4 przedstawiono przykładowe pomiary zużycia paliwa dla samochodu Ford Focus MK2.

Tabela 4: Przykładowe pomiary zużycia paliwa dla samochodu Ford Focus MK2

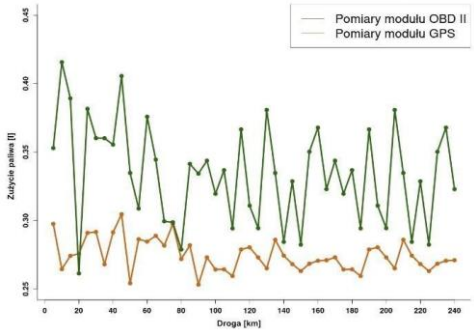
Liczba przejechanych kilometrów [km]	Ilość spalonego paliwa – moduł GPS [l/5km]	Ilość spalonego paliwa – moduł OBD II [l/5km]
[…]		
50	0,2542	0,3347
55	0,2863	0,3087
60	0,2846	0,3758
65	0,2888	0,3445
70	0,2817	0,2994
75	0,2974	0,2987
80	0,2719	0,2788
85	0,2819	0,3413
90	0,2531	0,3342
95	0,2730	0,3437
100	0,2642	0,3195
[…]		

W Tabeli 5 przedstawione są wyniki wyznaczenia średniej procentowej różnicy między pomiarem z interfejsu OBD II i estymacji wykorzystującej GPS.

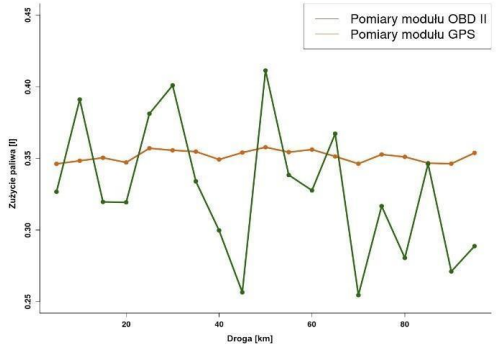
Tabela 5: Błąd przeciętny zużycia paliwa danego samochodu, przez aplikację

Model samochodu	Średni błąd pomiaru zużycia paliwa [%]
Ford Focus MK2	6,00
Mercedes W204	4,26
Ford Mondeo MK4	20,28

Rysunek 1 przedstawia wykres porównujący wyniki obu modułów aplikacji na trasie 240 kilometrów przejechanych samochodem Ford Focus MK2. Można zauważyć, że moduł GPS opierając się jedynie na prędkości i danych wprowadzonych przez użytkownika przedstawia niższe wartości zużycia paliwa w stosunku do pomiarów z OBD II. Dane zebrane z komputera silnika oscylują w zakresie od 0,26 do 0,42. Rozstęp w tym przypadku wynosi 0,16. Dla danych obliczanych na podstawie nawigacji rozstęp wynosi 0,05.



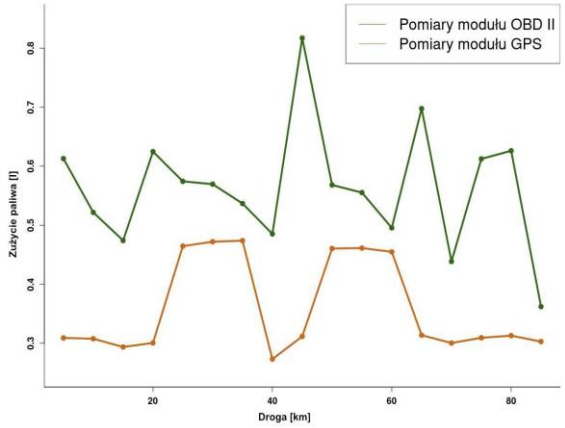
Rysunek 1: Wykres wartości zużycia paliwa przez samochód Ford Focus MK2 w pięciokilometrowych odstępach.



Rysunek 2: Wykres wartości zużycia paliwa dla samochodu Mercedes W204 w pięciokilometrowych odstępach.

Na Rysunku 2 można zauważyć znaczącą różnicę w pomiarze opierającym się na nawigacji dla Mercedesa W204, w stosunku do wyników otrzymanych dla samochodu Ford Focus MK2. Rozstęp zużycia paliwa obliczony na podstawie GPS wynosi zaledwie 0,01.

Ilustracja 3 przedstawia wyniki badania przeprowadzonego przy użyciu samochodu Ford Mondeo MK4. Technicznie łatwo zauważyć duży błąd pomiarowy. Wynika to z powodu konstrukcji silnika diesla, a jednocześnie zastosowanej metody obliczeń. Wzór użyty do obliczenia spalania paliwa wykorzystuje parametr MAF, co w przypadku nowoczesnych silników diesla okazuje się problematyczne. Charakter takiego silnika powoduje, że w niskich partiach obrotów silnika oraz przy stosunkowo niskich prędkościach jazdy uzyskujemy duży przepływ powietrza, a co za tym idzie aplikacja oblicza istotnie zawyżone wyniki w stosunku do pomiaru opartego o moduł GPS.



Rysunek 3: Wykres wartości zużycia paliwa przez samochód Ford Mondeo MK4 w pięciokilometrowych odstępach.

5. Wnioski

System estymacji oparty na nawigacji GPS, zaimplementowany w aplikacji CarRide, w części przypadków przedstawia wyniki zbliżone do spalania obliczanego na podstawie danych z interfejsu OBD II. Implementacja wykorzystuje tylko zmianę prędkości w czasie oraz dane użytkownika. Biorąc pod uwagę informacje z przeglądu literatury, można ją jeszcze rozszerzyć o dodatkowe dane wejściowe takie jak wzorce i modele jazdy lub specyfikację pojazdu, co mogłoby znacząco poprawić wyniki.

Przeprowadzona analiza rezultatów wyraźnie wskazuje na to, że estymacja GPS dużo bardziej zbliża się do wyników z systemu OBD II w przypadku samochodów (Ford Focus MK2, Mercedes W204) wyposażonych w sensor MAP. Różnica wyników opartych na modułach OBD II i GPS dla samochodów Ford Focus MK2 i Mercedes W204 różni się o dwa punkty procentowe, więc wyniki są bardzo zbliżone.

Błąd pomiaru w przypadku silnika z sensorem MAF (Ford Mondeo MK4) przekroczył 20%, w przedstawionym scenariuszu, a średnie spalanie paliwa podczas estymacji GPS wyniosło 7,2 l/100km, dla danych z interfejsu OBD II było to 11,2 l/100km. Różnica w spalaniu wynosząca 4 litry zupełnie przekreśla dla użytkownika taką funkcjonalność. Różnica w spalaniu na 100 kilometrów dla samochodu Mercedes W204 wynosi pół litra, estymacja za pomocą nawigacji GPS wskazała 7 litrów a dane z OBD II 6,5 litra. Wyniki dla samochodu Ford Focus MK2 różnią się o 1,2 litra, moduł oparty na nawigacji wskazał spalanie 5,4 litra na 100 kilometrów, wyniki z OBD II natomiast 6,6 litra. Są to wartości, które potencjalnie mogłyby zadowolić klienta, konieczne byłoby przeprowadzenie ankiety, w której użytkownicy mogliby wyrazić swoją opinię na temat oczekiwanej dokładności estymacji spalania.

Literatura

[1] V. Ribeiro, J. Rodrigues, A. Aguiar, Mining geographic data for fuel consumption estimation, Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), (2013) 124-129, https://doi.org/10.1109/ITSC.2013.6728221.

[2] Ilość subskrypcji smartphonów na świecie https://www.statista.com/statistics/330695/numberofsmartphone-users-worldwide/, [22.08.2022].

[3] D. Rimpas, A. Papadakis, M. Samarakou, OBD-II sensor diagnostics for monitoring vehicle operation and consumption, Energy Reports 6(3) (2020) 55-63, https://doi.org/10.1016/j.egyr.2019.10.018.

[4] M. Gumus, Effects of volumetric efficiency on the performance and emissions characteristics of a dual fueled (gasoline and LPG) spark ignition engine, Fuel Processing Technology 92(10) (2011) 1862-1867, https://doi.org/10.1016/j.fuproc.2011.05.001.

[5] M. Masi, P. Gobbato, Measure of the volumetric efficiency and evaporator device performance for a liquefied petroleum gas spark ignition engine, Energy Conversion and Management 60 (2012) 18-27, https://doi.org/10.1016/j.enconman.2011.11.030.

[6] M. Ali, M. D. Kamal, A. Tahir, S. Atif, Fuel Consumption Monitoring through COPERT Model—A Case Study for Urban Sustainability, Sustainability 13 (2021) 11614-11626, https://doi.org/10.3390/su132111614.

[7] J. E. Meseguer, C. T. Calafate, J. C. Cano, P. Manzoni, Assessing the impact of driving behavior on instantaneous fuel consumption, In 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), (2015) 443-448, https://doi.org/10.1109/CCNC.2015.7158016.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 104–111
Received: 17 October 2022
Accepted: 20 November 2022

Comparative Analysis of the Net 6 and NestJS Programming Frameworks in Terms of their Suitability for User Authentication and Authorization

Analiza porównawcza szkieletów programistycznych Net 6 i NestJS pod kątem ich przydatności do autentykacji i autoryzacji użytkowników

Przemysław Szymon Rodzik*

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

Abstract
The aim of the article was to perform a comparative analysis of the Net 6 and NestJS programming framework in terms of their suitability for user authentication and authorization. The functionalities and programming libraries offered by the researched technologies were reviewed. Applications were created in the tested skeletons. Application performance and load tests were carried out. The obtained test results showed that the application written in NestJS offered a shorter time to service the request and was able to handle a larger number of users compared to the application using Net 6. Net 6 offered a greater number of functionalities in the field of authentication and authorization, their implementation required less work from the developer compared to the NestJS backbone.

Keywords: comparative analysis; authentication; NestJS; Net

Streszczenie
Celem artykułu było przeprowadzenie analizy porównawczej szkieletu programistycznego Net 6 i NestJS pod kątem ich przydatności w autentykacji i autoryzacji użytkowników. Dokonano przeglądu funkcjonalności oraz bibliotek programistycznych oferowanych przez badane technologie. Utworzono aplikacje w badanych szkieletach. Przeprowadzono testy wydajnościowe oraz obciążeniowe aplikacji. Otrzymane wyniki testów pokazały, iż aplikacja napisana w NestJS oferowała krótszy czas obsłużenia żądania oraz była w stanie obsłużyć większą liczbę użytkowników w porównaniu do aplikacji wykorzystującej Net 6. Net 6 oferował większą liczbę funkcjonalności z dziedziny autentykacji i autoryzacji, ich implementacja wymagała mniej pracy od programisty w porównaniu do szkieletu NestJS.

Słowa kluczowe: analiza porównawcza; autentykacja; NestJS; Net

*Corresponding author
Email address: **przemyslaw.rodzik@pollub.edu.pl** (P. S. Rodzik)

1. Wstęp

Kluczowym elementem wielu systemów jest proces autentykacji i autoryzacji użytkowników korzystających z tego systemu. Proces autentykacji to sposób, w którym użytkownik dostarcza aplikacji dowodów, iż jest on w rzeczywistości osobą za którą się podaje poprzez przedstawienie ważnego poświadczenia [1]. Najczęściej za poświadczenie uważa się login i hasło. W przypadku dostarczenia poprawnych danych przez użytkownika jego tożsamość zostaje potwierdzona, a proces uwierzytelnienia kończy się powodzeniem. Autoryzacja jest to proces nadawania uprawnień do wykonywania określonych akcji w systemie określonym podmiotom [2]. Najczęściej do akcji tych zalicza się wykonywanie operacji na chronionych zasobach. Proces autoryzacji wymaga, by dany podmiot został uprzednio uwierzytelniony.

W artykule dokonano analizy porównawczej szkieletu programistycznego Net 6 [3] i NestJS [4] pod kątem ich przydatności w autentykacji i autoryzacji użytkowników. Analiza ta miała na celu dostarczyć programiście przydatnych informacji dotyczących zagadnień związanych z autentykacją i autoryzacją, występujących w omawianych szkieletach programistycznych, które mogłyby pomóc mu w podjęciu decyzji dotyczącej wyboru najlepszego szkieletu programistycznego implementującego badane zagadnienie. W tym celu porównano biblioteki programistyczne oraz funkcjonalności z dziedziny autentykacji i autoryzacji oferowane przez badane szkielety programistyczne. Utworzono aplikacje w każdym z badanych szkieletów oferujące identyczne funkcjonalności umożliwiające autentykację i nadanie uprawnień użytkownikowi. Wykonano testy wydajnościowe, w których mierzono czas wykonania zapytań w zależności od zastosowanych algorytmów do procesu uwierzytelniania i autoryzacji. Sprawdzono jak duży ruch sieciowy jest w stanie obsłużyć dana technologia.

1.1. Przegląd literatury

Powstało wiele pozycji poruszających temat procesu autentykacji i autoryzacji użytkowników. W artykule „Evaluation of password hashing schemes in open source web platforms” [5] autorzy zbadali ponad 10 szkieletów programistycznych pod kątem algorytmów, które są w nich domyślnie zaimplementowane w procesie uwierzytelniania. Wyniki pokazały, iż w większości technologii użyte algorytmy są przestarzałe, nie gwarantują bezpieczeństwa. Dodatkowo sporządzili listę wytycznych, zaproponowali alternatywne algorytmy w celu

zwiększenia poziomu bezpieczeństwa. W pracy „Implementation and Performance Analysis of PBKDF2, Bcrypt, Scrypt Algorithms" [6] utworzono aplikacje, w której przeprowadzono testy mierzące czas obsłużenia żądań logowania dla algorytmu Bcrypt [7], Scrypt [8] i PBKDF2 [9] w zależności od użytych parametrów wejściowych. Udowodniono, iż najszybszym rozwiązaniem jest algorytm PBKDF2. Niestety w badaniach nie uwzględnili jaki wpływ ma liczba zapytań wykonywanych jednocześnie na czas obsłużenia żądania. Branimir Pervan, Josip Knezovic i Katja Pericin [10] dokonali testów wydajnościowych opracowanego przez nich systemu rozproszonego do obliczania wartości funkcji skrótu algorytmu Bcrypt. Pokazali iż wydajność ich systemu, jak i jego efektywność energetyczna nie odbiega znacząco od porównywanych systemów. Aleksander Dikanski i Roland Steinegger [11] przeanalizowali szkielet Spring Security pod kątem jego wsparcia dla autentykacji i autoryzacji użytkowników. Ponadto dostarczyli czytelnikom wzorców do implementacji procesu uwierzytelnienia. Autorzy „Systematic Review of Authentication and Authorization Advancements for the Internet of Things" [12] przeanalizowali 1622 artykuły z dziedziny Internetu Rzeczy [13] w celu oceny aktualnego poziomu jakości implementacji rozwiązań z dziedziny autentykacji i autoryzacji. Wyniki ich badań pokazały, iż w większości systemów, moduły realizujące proces autentykacji, autoryzacji użytkowników działają w architekturze rozproszonej. Ponadto metoda autoryzacji użytkowników oparta na rolach [14] (ang. Role-based access control) traci na popularności na rzecz metody opierającej się na atrybutach [15] (ang. Attribute-based access control).

1.2. Cel badań

Celem badań było porównanie szkieletów programistycznych Net w wersji 6.0.101 i NestJS w wersji 8.0.0 pod kątem ich przydatności do autentykacji i autoryzacji użytkowników.

Hipotezy, których weryfikacji podjęto się w niniejszym artykule brzmiały następująco:

1. Technologia Net 6 oferuje większą liczbę funkcjonalności z zakresu autentykacji i autoryzacji niż technologia NestJS.
2. Średni czas obsłużenia żądania wysłanego do aplikacji napisanej przy wykorzystaniu szkieletu programistycznego NestJS jest krótszy niż w przypadku aplikacji korzystającej z technologii Net 6.
3. Aplikacja napisana w technologii NestJS w ciągu 10 sekund jest w stanie obsłużyć większą liczbę żądań wysyłanych przez użytkowników niż serwis napisany przy użyciu szkieletu programistycznego Net 6.

W celu weryfikacji powyższych hipotez utworzono dwie aplikacje prezentujące te same funkcjonalności, składające się z trzech mikro-usług. Każda z nich została napisana przy wykorzystaniu najnowszych wersji tychże technologii, tj. NestJS w wersji 8.0.0 i Net w wersji 6.0.101. Dokonano przeglądu funkcjonalności oraz rozwiązań z zakresu autentykacji i autoryzacji dostępnych w badanych szkieletach. Wykonano testy wydajnościowe oraz obciążeniowe badanych aplikacji w zależności od algorytmów użytych w procesie autentykacji. Otrzymane wyniki umożliwiły sformułowanie wniosków, weryfikację sformułowanych hipotez.

2. Metody badań

W celu porównania badanych technologii pod kątem ich przydatności do autentykacji i autoryzacji użytkowników:

- dokonano przeglądu dostępnych funkcjonalności oraz bibliotek programistycznych w badanych technologiach z zakresu uwierzytelniania i autoryzacji,
- utworzono aplikację testową w każdym z badanych szkieletów,
- przeprowadzono testy wydajnościowe mające na celu zebrać dane na temat czasu żądania wysłanego przez klienta. Testy te miały za zadanie porównać wydajność aplikacji w zależności od algorytmów użytych w procesie uwierzytelniania i autoryzacji,
- wykonano testy sprawdzające ile maksymalnie żądań wysłanych przez użytkowników jest w stanie obsłużyć serwis w zależności od sprzętu, na którym został uruchomiony.

2.1. Środowisko testowe

Testy przeprowadzono na laptopie marki Lenovo Legion V. Tabela 1 przedstawia parametry maszyny.

Tabela 1: Parametry komputera testowego

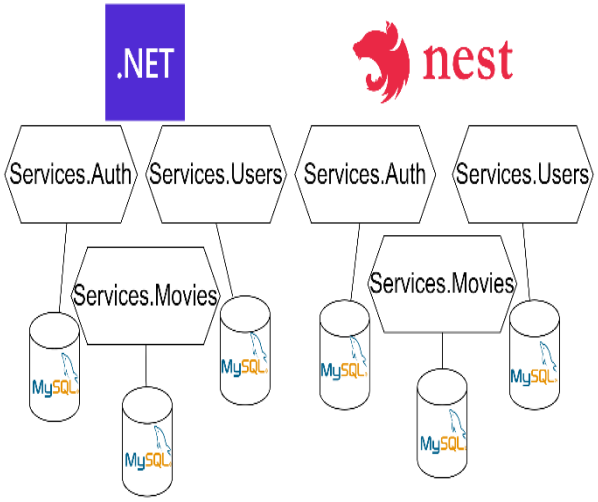
Parametr	Opis
Jednostka obliczeniowa	AMD Ryzen 7 4800H, 8 rdzeni, 16 wątków, 4,2Ghz
Karta graficzna	Nvidia Geforce RTX 2060 6GB
Pamięć RAM	32GB
Dysk	Crucial CT1000MX500SSD1
System operacyjny	Windows 10 Professional 21H1 19043.1586

W przypadku testów obciążeniowych wykorzystano dodatkowo maszynę wirtualną Vmware Workstation Player 16.0 [16] w celu zasymulowania działania aplikacji na mniej wydajnym sprzęcie. Zadbano by laptop był podłączony do sieci elektrycznej podczas przeprowadzania testów. Wszystkie inne procesy oprócz procesów systemu operacyjnego, które nie musiały pozostać uruchomione, zostały wyłączone.

2.2. Architektura aplikacji testowych

Aplikacje testowe zostały napisane przy wykorzystaniu architektury implementującej wzorzec architektury zorientowanej na usługi [17].

Na system składały się trzy mikro-usługi: uwierzytelnienia, użytkowników i filmów. Każda z nich posiadała własną bazę danych. Komunikowały się między sobą przy wykorzystaniu protokołu zdalnego wywoływania procedur (ang. Remote Procedure Call) firmy Google [18]. Wszystkie z nich wystawiały serwer http w celu udostępnienia swoich funkcjonalności użytkownikom. Rysunek 1 ilustruje schemat opisanej architektury dwóch systemów.



Rysunek 1: Architektura aplikacji testowych.

2.3. Implementacja aplikacji testowych

Aplikacje napisano wykorzystując szkielety programistyczne:

- Net w wersji 6.0.101,
- NestJS w wersji 8.0.0.

Zaprojektowano je w taki sposób by programista mógł zintegrować dane funkcjonalności do swojego systemu.

Szczególny nacisk postawiono na implementację serwisu uwierzytelniania. Tabela 2 przedstawia żądania możliwe do wysłania przez użytkownika do serwisu uwierzytelniania. Szczegółowo opisano tylko żądanie logowania, ponieważ tylko ono było wykorzystane w testach wydajnościowych.

Tabela 2: Żądania http możliwe do wysłania do serwisu uwierzytelnienia

Nazwa żądania	Ścieżka (względna)	Typ metody
Register	/register	POST
Login	/login	POST
Refresh	/refresh	POST
Logout	/logout	POST

Zadaniem żądania o nazwie „Login” było wygenerowanie tokenu JWT (ang. Json Web Token) [19], który umożliwiał użytkownikowi czasowy dostęp do serwisu. Klucz dostępu był generowany po uprzednim sprawdzeniu czy dostarczone dane logowania są prawidłowe. Po otrzymaniu danych logowania serwis uwierzytelniania w pierwszej kolejności wywoływał zdalnie procedurę, która zwracała dane o kliencie z serwisu użytkowników. Jeśli serwis uwierzytelniania nie otrzymał w odpowiedzi informacji o użytkowniku, zwracał kod błędu o numerze 401. W innym przypadku, serwis ten porównywał skrót hasła otrzymany w odpowiedzi od serwisu użytkowników ze skrótem obliczonym z hasła wprowadzonego przez klienta. Jeśli skróty się zgadzały, mikrousługa generowała token dostępu i token odświeżenia [20]. Następnie zwracała do użytkownika wiadomość zawierającą opisane powyżej tokeny.

Zadaniem serwisu użytkowników było zarządzanie znajdującymi się w bazie danych klientami aplikacji. Serwis ten odpowiadał na żądania wysyłane ze strony serwisu uwierzytelniania.

Serwis o nazwie „Services.Movies” został utworzony w celu wykonania testów odnoszących się do autoryzacji użytkownika w zależności od rodzaju użytego tokenu do walidacji. Implementacja serwisu filmów ograniczała się do udostępnienia punktu dostępu umożliwiającego pobranie informacji opisujących dany film. Skupiono się by dostęp do podanych treści był możliwy jedynie po pomyślnej walidacji tokenu JWT i w sytuacji, gdy użytkownik posiadał wymagane uprawnienie.

2.4. Przegląd funkcjonalności oraz bibliotek programistycznych do autentykacji i autoryzacji

Dokonano przeglądu bibliotek oferujących rozwiązania w zakresie autentykacji i autoryzacji, dostępnych w badanych szkieletach programistycznych. Sprawdzano dostępność i jakość bibliotek implementujących następujące funkcjonalności:

- generowanie i walidacja tokenów JWT (ang. Json Web Token),
- tworzenie skrótów kryptograficznych,
- tworzenie losowych ciągów bezpiecznych kryptograficznie,
- autoryzacje użytkowników dzięki rozpoznaniu ich twarzy bądź gestów,
- generowanie jednorazowych haseł.

Dla każdej funkcjonalności wybrano najlepszą bibliotekę. Wyboru najlepszej z nich dokonano biorąc pod uwagę parametry:

- średnią liczbę pobrań z oficjalnych repozytoriów pakietów omawianych szkieletów,
- liczbę nierozwiązanych błędów zgłaszanych przez użytkowników,
- siłę oferowanych algorytmów,
- obszerność dokumentacji,
- siłę domyślnej konfiguracji,
- stopień aktywności ze strony twórców oprogramowania, tj. liczbę przeprowadzonych modyfikacji biblioteki w okresie dwóch lat.

Porównano funkcjonalności z dziedziny autentykacji i autoryzacji oferowane przez badane technologie. Podczas analizy zwrócono szczególną uwagę czy implementacja danej funkcjonalności była możliwa bez konieczności dołączenia bibliotek firm trzecich oraz, w której z technologii implementacja danej funkcjonalności wymagała mniej pracy od programisty.

2.5. Testy wydajnościowe

Napisane aplikacje poddano testom wydajnościowym. Do tego celu wykorzystano narzędzie k6 [21]. Oprogramowanie to pozwala na pisanie skryptów w języku JavaScript opisujących przebieg testu, które są następnie interpretowane i wykonywane przez program. Dodatkowo w czasie trwania testu zapisywano informacje na temat zużycia procesora i pamięci RAM przy użyciu programu dołączonego z systemem Windows o nazwie Performance Monitor [22].

Dla każdej z aplikacji przeprowadzone zostały testy logowania w zależności od rodzaju algorytmu użytego do obliczenia funkcji skrótu [23] z dostarczonego hasła użytkownika. Każdy typ testu trwał 9 minut. Zbierano

informację o czasie obsłużenia pojedynczego żądania (czas liczony od chwili wysłania żądania do otrzymania odpowiedzi zwrotnej od serwisu). Badania były przeprowadzane dla 5, 10 i 15 użytkowników. Pojęcia określane jako „liczba wirtualnych użytkowników", bądź „liczba użytkowników", które mogą pojawić się w tekście, oznaczają liczbę jednocześnie wykonywanych zapytań do aplikacji do momentu zakończenia testu.

Kod źródłowy realizujący scenariusz logowania do sytemu ukazany na listingu 1 różnił się jedynie adresem bazowym żądań http w zależności od testowanej aplikacji. Skrypt był wykonywany w nieskończoność przez każdego z wirtualnych użytkowników do chwili zakończenia testu. Czas oczekiwania na wysłanie kolejnego żądania przez danego użytkownika wynosił 500ms. Zadbano by każdy z wirtualnych użytkowników odzwierciedlał innego użytkownika z bazy danych.

Listing 1: Kod źródłowy skryptu realizujący scenariusz zalogowania użytkownika do systemu

```
let body = JSON.stringify({
  username: `user${__VU}`,
  password: "VqhvQXxR",
});
const params = {
  headers: {
    "Content-Type": "application/json",
  },
  tags: {
    name: `login${__VU}`,
  },
};
const responseLogin = http.post(URL_LOGIN, body, params);
check(responseLogin, {
  "logged successfully": (r) =>
    r.status === 200 ? r.json().hasOwnProperty("accessToken") : false,
});
sleep(SLEEP_DURATION_IN_SEC);
```

Tabela 3 przedstawia algorytmy poddane testom wraz z bibliotekami użytymi do ich zaimplementowania. Biblioteki wybrano kierując się kryteriami wyboru przedstawionymi w rozdziale 2.4.

Tabela 3: Algorytmy użyte w testach logowania

Algorytm	Biblioteka dla NestJS (wersja)	Biblioteka dla Net 6 (wersja)
Bcrypt	bcrypt (5.0.1)	BCrypt.Net-Next (4.03)
Argon2id	argon2 (0.28.5)	Konscious.Security.Cryptography.Argon2 (1.21)

Testy logowania z algorytmem Bcrypt wykonano zmieniając odpowiednio wartość parametru „work factor" na 10, 11, 12, 13 i 14. Zmienna ta opisuje liczbę iteracji algorytmu mieszającego, które są wykonywane dla każdego hasła.

Algorytm Argon2id [24] posiada 3 główne parametry, których wartości można zmieniać, są to:

- memory cost,
- iterations,
- parallelism.

Testy przeprowadzano zmieniając wartości parametru „memory cost" na 16MB, 32MB, 64MB, 128MB, 256MB. Zmienna ta informuje nas o ilości pamięci RAM jaką algorytm może wykorzystać w celu obliczenia wartości skrótu. Pozostałe parametry ustawiono na minimalne wartości rekomendowane przez fundację OWASP [25].

Zadaniem testów obciążeniowych było sprawdzenie ile maksymalnie zapytań wysyłanych przez użytkowników jest w stanie obsłużyć serwis w zależności od sprzętu, na którym pracuje. Za maksymalną liczbę zapytań przyjęto wartość, którą zwiększenie spowoduje wystąpienie błędów w aplikacji, odrzucanie przez nią żądań. Test był przeprowadzony w ten sposób, iż w ciągu 10 sekund określona liczba użytkowników miała za zadanie zalogować się do systemu. Stopniowo zwiększano liczbę użytkowników do momentu kiedy serwis zaczął generować błędy, odrzucać żądania. Testy przeprowadzano na maszynie wirtualnej. Z puli 16 procesorów oraz 32GB pamięci RAM oferowanych przez maszynę hosta, zmieniano liczbę dostępnych procesorów (4, 6, 8), ilość pamięci RAM (8GB, 16GB) oraz algorytmy użyte w procesie autentykacji (Bcrypt, Argon2id). Oba algorytmy skonfigurowano przy wykorzystaniu minimalnych, rekomendowanych przez fundację OWASP parametrów. Dla algorytmu Bcrypt parametr „work factor" został ustawiony na wartość 10. W przypadku Argon2id parametry: parallelism, memory, time ustawiono odpowiednio na wartości: 1, 16MB, 2.

3. Wyniki badań

Przegląd rozwiązań do autentykacji i autoryzacji pokazał, iż więcej bibliotek ma do zaoferowania technologia Net 6 niż NestJS. Siła, bezpieczeństwo algorytmów oferowanych przez obydwie technologie była wystarczająca, lecz w przypadku Net 6 biblioteki były częściej aktualizowane. Rozwiązania napisane dla technologii NestJS posiadały większą liczbę błędów zgłaszanych przez użytkowników niż dla technologii Net 6. W przypadku obu technologii nie znaleziono biblioteki, która dostarczałaby implementacji wszystkich rekomendowanych przez fundacje OWASP algorytmów do tworzenia skrótów kryptograficznych. W celu zaimplementowania do swojego systemu algorytmu Bcrypt, Scrypt bądź Argon2 programista był zmuszony do dołączenia dodatkowych bibliotek. Wybierając bibliotekę należy brać pod uwagę siłę domyślnej konfiguracji oferowanych przez nią algorytmów. Domyślna konfiguracja algorytmów dostępnych w badanych bibliotekach dla Bcrypt, Scrypt, Argon2 oraz HMAC była tożsama z tą rekomendowaną przez fundację OWASP. Tabela 4 przedstawia listę rekomendowanych bibliotek możliwych do wykorzystania w przypadku konieczności zaimplementowania danej funkcjonalności.

Analizując funkcjonalności oferowane przez technologie stwierdzono, iż szkielet Net 6 posiada ich więcej oraz oferowana przez nie domyślna implementacja jest bardziej rozbudowana i dostarcza wielu metod, które przyspieszają pracę programisty. Tabela 5 przedstawia podsumowanie dostępnych funkcjonalności z zakresu

autentykacji i autoryzacji w każdym z omawianych szkieletów programistycznych.

Tabela 4: Zalecane biblioteki do wykorzystania w celu implementacji wybranych funkcjonalności

Opis funkcjonalności	Nazwa biblioteki (wersja)	
	Net 6	NestJS
Generowanie i walidacja tokenów JWT	jose-jwt (4.0.0)	jose (4.8.1)
Tworzenie skrótów kryptograficznych	Konscious.Security.Cryptography.Argon2 (1.21), BCrypt.Net-Next (4.03), System.Security.Cryptography.Algorithms (4.01), Scrypt.NET (1.3.0)	argon2 (0.28.5), bcrypt (5.0.1), hash.js (1.0.1), node-scrypt (6.0.3)
Tworzenie losowych ciągów bezpiecznych kryptograficznie	System.Security.Cryptography.RandomCryptoGenerator (2.16)	Crypto-random-string (5.0.0)
Autoryzacja – rozpoznawanie twarzy	FaceRecognitionDotNet (1.3.07)	node-facenet (0.10.3)
Generowanie jednorazowych haseł	Otp.NET (1.2.2)	Otplib (12.0.1)

Tabela 5: Spis funkcjonalności z zakresu autentykacji i autoryzacji dostępnych w technologii Net 6 i NestJS

Funkcjonalność	NestJS	Net 6
Autentykacja przy użyciu hasła	✓	✓
Autentykacja przy użyciu tokenu JWT	✓	✓
Autentykacja przy użyciu zewnętrznych serwisów	✓	✓
Polityka haseł	×	✓
Obliczanie wartości funkcji skrótu	×	✓
Blokowanie użytkowników	×	✓
Dwuetapowe logowanie	×	✓
Tworzenie sesji	✓	✓
Potwierdzenie adresu e-mail	×	✓
Autoryzacja oparta na rolach	✓	✓
Autoryzacja oparta na oświadczeniach	✓	✓
Ukrywanie elementów interfejsu graficznego	×	✓
Tworzenie tabel przechowujących informacje o użytkownikach, ich rolach, oświadczeniach	×	✓
Zwracanie informacji o przypisanej roli, oświadczeniach zalogowanego użytkownika	×	✓

Tabele 6 i 7 przedstawiają informacje na temat maksymalnej liczby użytkowników, wykonującej żądanie logowania, możliwej do obsłużenia przez aplikacje napisaną przy użyciu szkieletu Net 6 oraz NestJS w zależności od parametrów maszyny testowej odpowiednio przy autentykacji wykorzystującej algorytm Bcrypt oraz Argon2id. Po analizie wyników zaprezentowanych w tabelach 6 i 7 stwierdzono, iż aplikacja napisana w technologii NestJS obsłuży większą liczbę użytkowników niż aplikacja wykorzystująca szkielet Net 6. Zauważono, iż dla szkieletu Net autentykacja z wykorzystaniem algorytmu Bcrypt pozwala na obsłużenie większej liczby żądań niż z wykorzystaniem algorytmu Argon2id. Z kolei dla technologii NestJS wykorzystanie algorytmu Argon2id umożliwia obsłużenie większej liczby użytkowników.

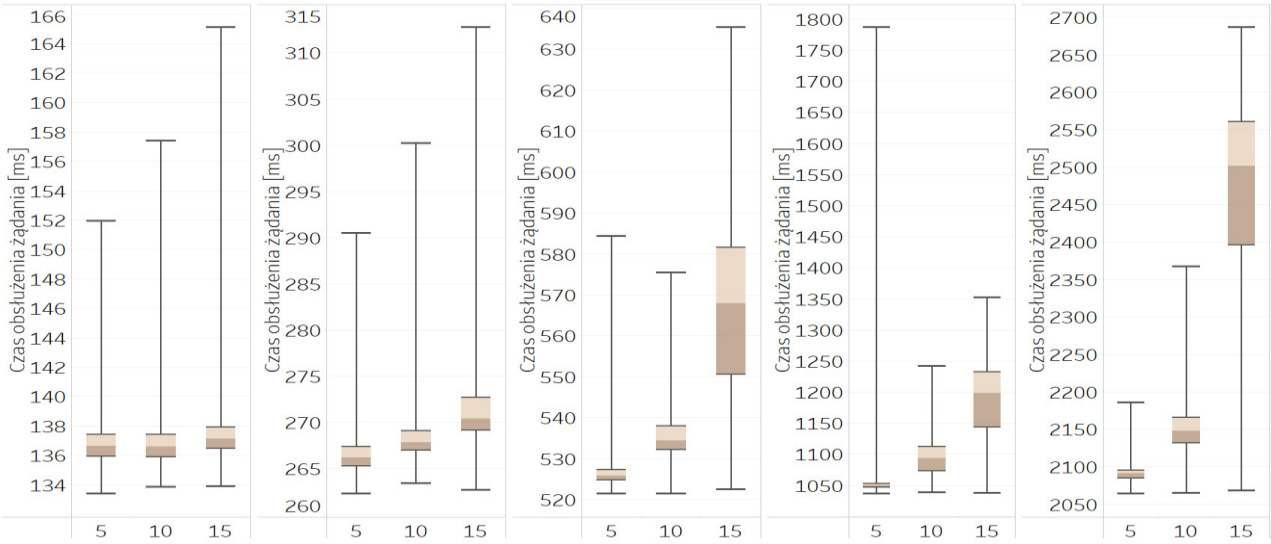
Tabela 6: Maksymalna liczba użytkowników wykonująca żądanie logowania możliwa do obsłużenia w danej technologii przy wykorzystaniu algorytmu Bcrypt do procesu autentykacji w zależności od ustawionych parametrów sprzętowych

Maksymalna liczba użytkowników		
Technologia / Parametry	Net 6	NestJS
4 procesory, 8GB RAM	1050	1300
4 procesory, 16GB RAM	1080	1320
6 procesorów, 8GB RAM	1600	1900
6 procesorów, 16GB RAM	1650	1920
8 procesorów, 8GB RAM	2020	2300
8 procesorów, 16GB RAM	2000	2260

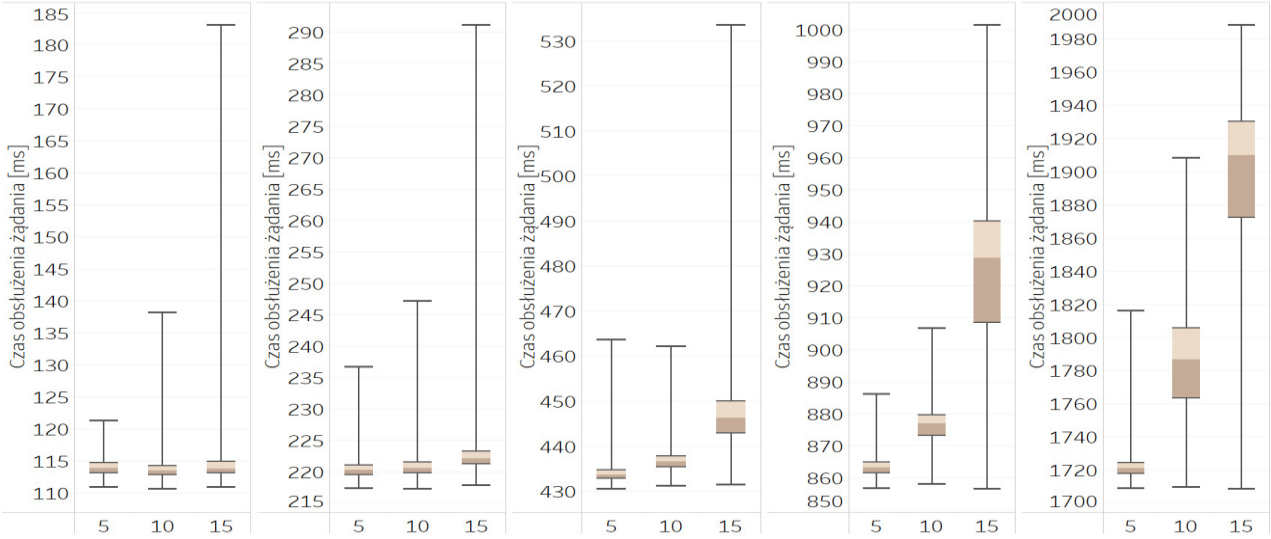
Tabela 7: Maksymalna liczba użytkowników wykonująca żądanie logowania możliwa do obsłużenia w danej technologii przy wykorzystaniu algorytmu Argon2id do procesu autentykacji w zależności od ustawionych parametrów sprzętowych

Maksymalna liczba użytkowników		
Technologia / Parametry	Net 6	NestJS
4 procesory, 8GB RAM	800	2200
4 procesory, 16GB RAM	830	2180
6 procesorów, 8GB RAM	1040	2970
6 procesorów, 16GB RAM	1000	3000
8 procesorów, 8GB RAM	1200	3180
8 procesorów, 16GB RAM	1220	3190

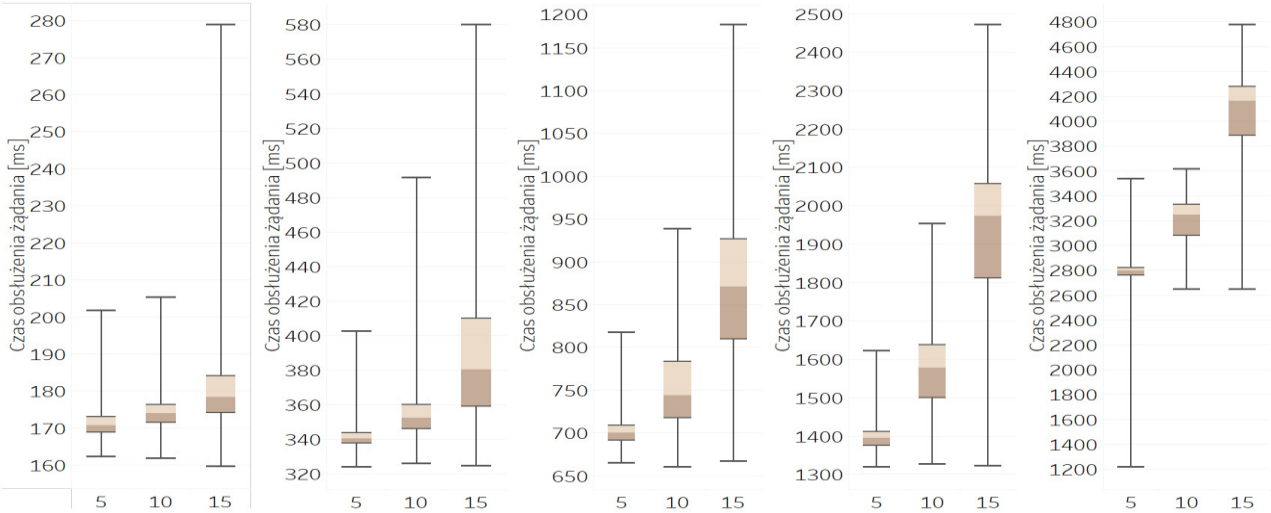
Analizując czasy logowania przedstawione na rysunkach 2-5 zauważono, iż niezależnie od użytego szkieletu programistycznego wraz ze wzrostem wartości współczynnika "work factor" w algorytmie Bcrypt jak i wraz ze zwiększeniem wartości parametru „memory” w algorytmie Argon2id czas obsłużenia żądania zwiększał się ok. dwukrotnie.



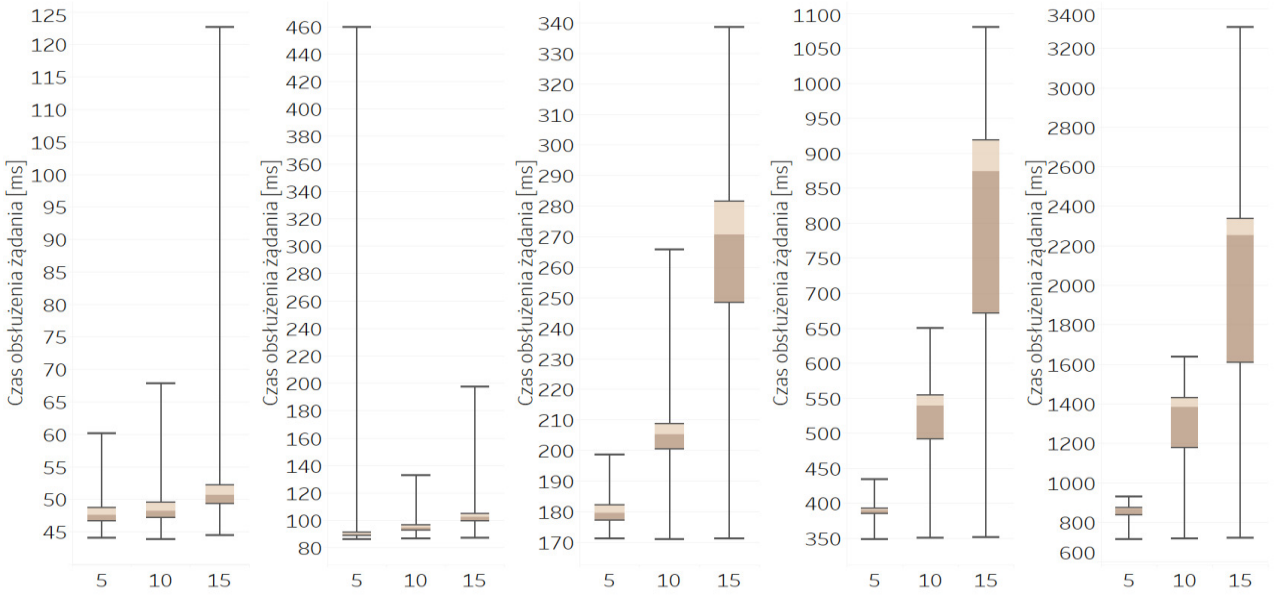
Rysunek 2: Wykresy czasów obsłużenia żądań logowania z aplikacji napisanej przy użyciu technologii Net 6 wykorzystującej algorytm Bcrypt w zależności od parametru „work factor" odpowiednio od lewej dla wartości równej 10, 11, 12, 13 i 14 dla 5, 10 i 15 użytkowników jednocześnie wykonujących żądania.



Rysunek 3: Wykresy czasów obsłużenia żądań logowania z aplikacji napisanej przy użyciu technologii NestJS wykorzystującej algorytm Bcrypt w zależności od parametru „work factor" odpowiednio od lewej dla wartości równej 10, 11, 12, 13 i 14 dla 5, 10 i 15 użytkowników jednocześnie wykonujących żądania.



Rysunek 4: Wykresy czasów obsłużenia żądań logowania z aplikacji napisanej przy użyciu technologii Net 6 wykorzystującej algorytm Argon2id w zależności od parametru „memory cost" wyrażonego w Megabajtach odpowiednio od lewej dla wartości równej 16, 32, 64, 128 i 256 dla 5, 10 i 15 użytkowników jednocześnie wykonujących żądania.



Rysunek 5: Wykresy czasów obsłużenia żądań logowania z aplikacji napisanej przy użyciu technologii NestJS wykorzystującej algorytm Argon2id w zależności od parametru „memory cost" wyrażonego w Megabajtach odpowiednio od lewej dla wartości równej 16, 32, 64, 128 i 256 dla 5, 10 i 15 użytkowników jednocześnie wykonujących żądania.

4. Wnioski

Dokonując przeglądu funkcjonalności oferowanych przez badane technologie stwierdzono, iż Net 6 ma ich do zaoferowania więcej. Domyślna implementacja była bardziej rozbudowana i dostarczała większej liczby metod przyspieszających pracę programisty. Implementacja wielu metod dostępnych w technologii NestJS była niewystarczająca, od programisty wymagało się samodzielnego zaimplementowania takich funkcjonalności jak generowanie skrótu kryptograficznego z hasła, bądź wprowadzenie polityki haseł. Do funkcjonalności, których zabrakło w NestJS, a były one dostępne w Net należało między innymi ukrywanie elementów interfejsu graficznego, tworzenie struktury bazy danych przechowujących informacje o użytkownikach i przyznanych im rolach. Na bazie uzyskanych wyników należało potwierdzić hipotezę badawczą mówiącą, iż technologia Net 6 oferuje większą liczbę funkcjonalności z zakresu autentykacji i autoryzacji użytkowników niż technologia NestJS. Analiza bibliotek programistycznych pokazała, iż w każdym z omawianych szkieletów było wiele rozwiązań, które cieszyły się popularnością wśród programistów, były na bieżąco aktualizowane, siła dostarczanych przez nie algorytmów jak i ich domyślna konfiguracja była wystarczająca by zapewnić wysoki poziom bezpieczeństwa w systemie. W każdym z badanych szkieletów znaleziono biblioteki implementujące algorytmy uznawane za bezpieczne i rekomendowane przez fundacje OWASP. Na bazie przeprowadzonych testów wydajnościowych stwierdzono, iż czas obsłużenia pojedynczego żądania był krótszy dla aplikacji napisanej w technologii NestJS niż w aplikacji wykorzystującej szkielet Net 6 niezależnie od algorytmów wykorzystanych do autentykacji i autoryzacji. Zauważono, iż niezależnie od użytego szkieletu programistycznego wraz ze wzrostem współczynnika "work factor" dla algorytmu Bcrypt jak i wraz ze zwiększeniem wartości parametru „memory" w algorytmie Argon2id czas obsłużenia żądania zwiększał się ok. dwukrotnie. Wraz ze wzrostem liczby użytkowników biorących udział w testach dane przyjmowały bardziej różniące się wartości i wydłużał się czas potrzebny na zakończenie pojedynczego żądania. Wyniki testów wydajnościowych pozwoliły potwierdzić hipotezę badawczą stanowiącą, iż średni czas obsłużenia żądania wysłanego do aplikacji napisanej przy wykorzystaniu szkieletu programistycznego NestJS jest krótszy niż w przypadku aplikacji wykorzystującej technologie Net 6. Ostatni z przeprowadzonych testów pokazał, iż serwis napisany w technologii NestJS był w stanie obsłużyć większą liczbę użytkowników niż ten sam serwis wykorzystujący szkielet Net 6. Zauważono, iż aplikacja napisana w Net 6, w której do procesu autentykacji wykorzystano algorytm Bcrypt, była w stanie obsłużyć większą liczbę użytkowników w porównaniu do konfiguracji z algorytmem Argon2id. Z kolei dla technologii NestJS wykorzystanie algorytmu Argon2id umożliwiło obsłużenie większej liczby użytkowników niż w przypadku konfiguracji z algorytmem Bcrypt. Należało potwierdzić hipotezę badawczą mówiącą, iż aplikacja napisana w technologii NestJS w ciągu 10 sekund jest w stanie obsłużyć większą liczbę żądań wysyłanych przez użytkowników niż serwis napisany przy użyciu szkieletu programistycznego Net 6.

Podsumowując, wyniki badań pokazały, iż nie można jednoznacznie stwierdzić, który ze szkieletów programistycznych jest bardziej przydatny do autoryzacji i autentykacji użytkowników. Biorąc pod uwagę szybkość obsłużenia pojedynczego żądania jak i maksymalną liczbę użytkowników, którą serwis może obsłużyć, godnym polecenia szkieletem będzie NestJS. Z kolei technologia Net 6 jest lepszym wyborem jeśli zależy nam na dużej liczbie oferowanych funkcjonalności oraz bibliotek programistycznych z dziedziny autentykacji i autoryzacji.

Literatura

[1] S. Tumin, S. Encheva, A Closer Look at Authentication and Authorization Mechanisms for Web-based Applications, Proceedings of the 5th WSEAS Congress on Applied Computing Conference, and Proceedings of the 1st International Conference on Biologically Inspired Computation (2012) 100-105.

[2] J. Lopez, R. Oppliger, G. Pernul, Authentication and authorization infrastructures (AAIs): a comparative survey, Computers & Security 23(7) (2004) 578-590, https://doi.org/10.1016/j.cose.2004.06.013.

[3] M. J. Price, C# 10 and .NET 6 - Modern Cross-Platform Development, Packt Publishing, 2021.

[4] G. Magolan, et.al., Nest.js: A Progressive Node.js Framework, Packt Publishing, 2022.

[5] C. Ntantogian, et.al., Evaluation of password hashing schemes in open source web platforms, Computers & Security 84 (2019) 206-224, https://doi.org/10.1016/j.cose.2019.03.011.

[6] L. Ertaul, et.al., Implementation and Performance Analysis of PBKDF2, Bcrypt, Scrypt Algorithms, Proceedings of the International Conference on Wireless Networks (ICWN) (2016) 66-72.

[7] N. Provos, D. Mazières, A Future-Adaptable Password Scheme, FREENIX Track: 1999 USENIX Annual Technical Conference Proceedings (1999) 81-92.

[8] C. Percival, S. Josefsson, The scrypt Password-Based Key Derivation Function, RFC 7914 (2016) 1-16, https://doi.org/10.17487/RFC7914.

[9] Ed. K. Moriarty, B. Kaliski, A. Rusch, PKCS #5: Password-Based Cryptography Specification Version 2.1, RFC 8018 (2017) 1-40, https://doi.org/10.17487/RFC8018.

[10] B. Pervan, J. Knezovic, K. Pericin, Distributed Password Hash Computation on Commodity Heterogeneous Programmable Platforms, 13th USENIX Workshop on Offensive Technologies (WOOT 19) (2019) 1-8.

[11] A. Dikanski, R. Steinegger, Identification and implementation of authentication and authorization patterns in the spring security framework, SECURWARE 2012 - 6th International Conference on Emerging Security Information, Systems and Technologies (2012) 14-20.

[12] M. Trnka, et.al., Systematic Review of Authentication and Authorization Advancements for the Internet of Things, Sensors 22(4) (2022) 1361-1385, https://doi.org/10.3390/s22041361.

[13] Internet rzeczy, definicja, https://en.wikipedia.org/wiki/Internet_of_things, [03.05.2022].

[14] Wikipedia - Role-based access control, https://en.wikipedia.org/wiki/Role-based_access_control, [22.09.2022].

[15] Wikipedia - Attribute-based access control, https://en.wikipedia.org/wiki/Attributebased_access_control, [19.09.2022].

[16] Oficjalna strona producenta maszyny wirtualnej Vmware, https://www.vmware.com/pl.html, [09.10.2022].

[17] S. Newman, Building Microservices, 2nd Edition, O'Reilly Media, 2021.

[18] K. Indrasiri, D. Kuruppu, gRPC: Up and Running, O'Reilly Media, 2020.

[19] Json Web Token, definicja, https://datatracker.ietf.org/doc/html/rfc7519, [03.05.2022].

[20] Token odświeżenia, definicja, https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/, [15.08.2022].

[21] Narzędzie do tworzenia testów wydajnościowych k6, https://k6.io, [03.05.2022].

[22] Informacje o programie Performance Monitor, https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc749154(v=ws.10), [03.05.2022].

[23] Definicja funkcji skrótu, https://en.wikipedia.org/wiki/Hash_function, [05.05.2022].

[24] A. Biryukov, et.al., Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications, 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (2016) 292-302, https://www.doi.org/10.1109/EuroSP.2016.31.

[25] Główna strona internetowa Open Web Application Security Project, https://owasp.org, [22.06.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 112–120
Received: 17 October 2022
Accepted: 26 October 2022

Analysis of the Spring Boot and Spring Cloud in developing Java cloud applications

Analiza zastosowania Spring Boot i Spring Cloud w tworzeniu aplikacji chmurowych w Javie

Mateusz Kozak*

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

Abstract
The paper is of a review character. It analyzes the possibilities of using the Spring Boot programming framework with the Spring Cloud extension to create cloud applications in Java. The article presents the concepts that have to be dealt with when implementing the application in cloud environments and the technologies that were used in the applications, along with the justification of the choice. Scalability tests were carried out on two applications - non-scalable with the use of Spring Boot and scalable with the use of Spring Boot and Spring Cloud. The research carried out in the article showed how to implement a scalable application using Spring Boot and Spring Cloud.

Keywords: analysis; Spring Boot; Spring Cloud; cloud applications

Streszczenie
Artykuł ma charakter przeglądowy. Przeanalizowano w nim możliwości zastosowania szkieletu programistycznego Spring Boot z rozszerzeniem Spring Cloud do tworzenia aplikacji chmurowych w Javie. W artykule przedstawiono pojęcia z jakimi trzeba się mierzyć wdrażając aplikację w środowiskach chmurowych oraz technologie jakie zastosowano w aplikacjach wraz z uzasadnieniem wyboru. Przeprowadzono badania pod kątem skalowalności na dwóch aplikacjach – nieskalowalnej z wykorzystaniem Spring Boot i skalowalnej z wykorzystaniem Spring Boot i Spring Cloud. Badania przeprowadzone w artykule wskazały sposób realizacji skalowalnej aplikacji z zastosowaniem Spring Boot i Spring Cloud.

Słowa kluczowe: analiza; Spring Boot; Spring Cloud; aplikacje chmurowe

*Corresponding author
Email address: **mateusz.kozak3@pollub.edu.pl** (M. Kozak)

1. Wstęp

Popularność mikroserwisów w ostatnich latach rośnie, mogą one rozwiązać wiele aktualnych wyzwań w sektorze IT, takich jak zwiększenie wydajności, skalowalność aplikacji i przeprowadzanie testów. W dzisiejszych czasach wiele firm wdrażając swoje aplikacji zdecydowało się na wykorzystanie architektury mikrousług. Najbardziej rozpoznawalne firmy to Coca-Cola, Spotify, Netflix i Amazon [1]. Powód dlaczego architektura mikrousług jest używana to przede wszystkim skalowalność, elastyczność, zdolność adaptacji (wykorzystywanie różnych technologii) oraz możliwości wdrażania bez ingerowania w pozostałe aplikacje [2].

Na rynku jest wiele technologii ułatwiających pracę z mikroserwisami. Najpopularniejsze z nich to Docker, Kubernetes oraz REST [3]. Java jest jednym z najpopularniejszych języków wykorzystywanych w mikroserwisach [4] ze względu na czytelność kodu oraz składnię adnotacji (ang. annotation syntax). Java posiada wiele narzędzi ułatwiających pracę z mikroserwisami, w tym Spring Boot [5].

Wdrażanie aplikacji opartych na mikroserwisach w chmurze udostępnia aplikacje na pewne problemy, takie jak skalowalność, wykrywanie usług czy zewnętrzna konfiguracja. Spring Cloud zapewnia zestaw narzędzi do szybkiego tworzenia aplikacji w chmurze oferując rozwiązania problemów występujących w środowiskach rozproszonych [6].

Celem artykułu jest analiza możliwości zastosowania Spring Boot i Spring Cloud w tworzeniu aplikacji chmurowych w Javie. W pracy omówiono pojęcia, występujące w środowiskach chmurowych. Przedstawiono zarys szablonowej aplikacji, którą można wdrożyć do platformy chmurowej, takiej jak Kubernetes. Przeprowadzono badania pod kątem skalowalności na dwóch aplikacjach, które demonstrują możliwości oferowane przez narzędzia Spring Boot i Spring Cloud.

2. Wprowadzenie

2.1. Mikroserwisy

Mikroserwis [7] (inaczej architektura mikrousług) umożliwia szybkie i niezawodne rozwijanie złożonych aplikacji. Wraz z rozwojem nieskalowalnych aplikacji trudniej jest wdrażać kolejne programy. Mikroserwisy oddzielają poszczególne fragmenty aplikacji. Pozwalają się skupić na konkretnym podzbiorze całej aplikacji, można wdrażać nowe funkcjonalności niezależnie od już istniejących. Takie rozwiązanie pozwala również na używanie różnych technologii.

Do najbardziej popularnych technologii tworzenia aplikacji z wykorzystaniem architektury mikrousług należą języki programowania Node.js, Java i Python [3], a także narzędzia API Fortress, Postman i RabbitMQ [4].

2.2. Spring Boot

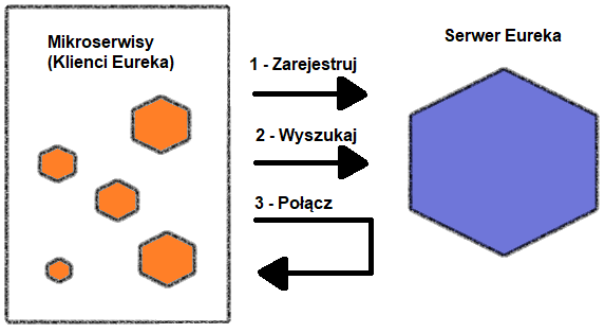
Spring Boot [8] jest rozszerzeniem szkieletu programistycznego Spring. Pozwala na szybkie opracowanie aplikacji i jest najpopularniejszym szkieletem języka Java. Dostarcza klasy i interfejsy potrzebne do opracowania aplikacji, które zapewniają bezpieczeństwo, integrację, łączenie z wieloma bazami, itp.

2.3. Spring Cloud

Budując serwisy w Spring wykorzystywane jest rozszerzenie Spring Cloud [9], które dostarcza wiele narzędzi i pomaga w pracy przy tworzeniu złożonych aplikacji w środowiskach systemów rozproszonych.

Service Discovery

Service Discovery [10] (w przypadku Spring Cloud jest to serwer Eureka) to proces automatycznego wykrywania urządzeń i usług w sieci. Mikrousługi są klientami serwera Eureka. Jest scentralizowanym miejscem, w którym wszyscy klienci rejestrują się na serwerze. Kiedy nastąpi rejestracja klienta, serwer zna dokładne informacje gdzie jest uruchomiona usługa, tj. host i port. Gdy mikroserwisy chcą komunikować się z innym serwisem muszą wyszukać na serwerze te informacje i w ten sposób mogą się ze sobą łączyć. Serwer kojarzy wszystkie aplikacje klienckie działające na każdym porcie i adresie IP (Rys. 1).



Rysunek 1: Rejestracja oraz komunikacja serwisów z serwerem Eureka [11].

Load Balancer

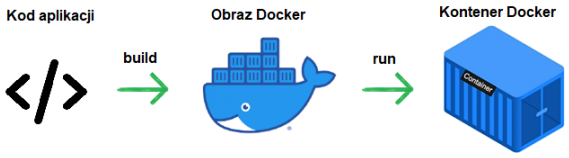
Klienci wysyłają żądania poprzez publiczny Internet. Load Balancer [12] służy jako główny punkt wejścia. Zwykle podczas tworzenia aplikacji klient nie komunikuje się bezpośrednio z usługami, ponieważ te serwisy powinny znajdować się w sieci prywatnej. Load Balancer jest odpowiedzialny za wykonanie przekierowania (ang. routing) do odpowiedniej mikrousługi w oparciu o ścieżkę.

Load Balancer jest konieczny podczas wdrażania w środowisku produkcyjnym. Klient nie powinien uzyskiwać dostępu do jednej instancji za pośrednictwem adresu IP. Im większy ruch jest w aplikacji tym trudniej sobie z tym poradzić.

2.4. Docker

Docker [13] jest platformą do budowania, uruchamiania i dostarczania (ang. shipping) aplikacji. Deweloperzy oprogramowania mogą z łatwością budować i opracowywać aplikacje uruchamiane w kontenerach. Kontener jest spakowaną instancją aplikacji. Lokalny rozwój (ang. local development) aplikacji jest taki sam w dowolnym stosie technologicznym (ang. environment). Docker jest wykorzystywany również w obiegu ciągłej integracji i ciągłego dostarczania (ang. continuous integration and continuous delivery/deployment - CI/CD) [14], tj. dostarczaniu aplikacji poprzez wprowadzanie automatyzacji na etapach opracowywania aplikacji.

Przy tworzeniu oprogramowania, kod aplikacji może być napisany w dowolnym języku z wykorzystaniem dowolnej technologii. Aplikacja jest budowana jako obrazy Docker (ang. Docker Images), z których może zostać uruchomiony kontener. Kontener jest na swój sposób szablonem do uruchamiania aplikacji. Z utworzonego obrazu można uruchomić wiele kontenerów, korzystających z różnych silników (Rys. 2).



Rysunek 2: Budowanie obrazu Docker oraz uruchomienie kontenera Docker [13].

2.5. Kubernetes

Kubernetes [15] jest otwarto-źródłowym narzędziem, napisanym w języku Go [16], pochodzącym od narzędzi Borg [17] i Omega [18]. Kuberneter jest zarządcą aplikacji, jego głównym zadaniem jest wdrażanie i zarządzanie aplikacjami (kontenerami).

3. Przegląd literatury

Większość istniejących pozycji opisuje wzorce, podejście oraz architekturę rozwiązań chmurowych. Badania dotyczą procesu tworzenia aplikacji. Istnieje niewiele badań i testów aplikacji chmurowych z wykorzystaniem architektury mikroserwisów.

W artykule [19] przedstawiono zastosowanie wstępnego zestawu wymagań, które mogą być przydatne przy wyborze wzorca architektury w celu wsparcia badań nad powtarzalnymi mikrousługami. Autor wskazuje na brak powtarzalnych badań dotyczących projektowania, rozwoju i oceny aplikacji mikrousług.

Autor książki [20] opisuje jak działa Spring Boot, zapoznaje z technologią Spring – przedstawia wzorce natywne dla chmury, programowanie reaktywne i aplikacje.

Autor książki [21] przedstawia koncepcje, architektury i scenariusze mikrousług pod względem technologicznym oraz pokazuje, jak je wdrażać za pomocą tech-

nologii, takich jak Docker, Java, Spring Boot i Spring Cloud.

Wydanie drugie pozycji [22] opisuje proces tworzenia aplikacji opartych na mikrousługach przy użyciu Java i Spring. Omawia proces tworzenia podstawowych usług, monitorowania aplikacji, opisuje sposoby refaktoryzacji za pomocą narzędzi Spring i wprowadza Spring Cloud Gateway do zarządzania API. Opisuje proces wdrażania aplikacje z użyciem technologii Spring Cloud z AWS i Kubernetes.

Autor książki [23] przedstawia jak za pomocą Spring Boot tworzyć aplikacje korporacyjne, internetowe i mikrousługowe oparte na Javie. W treści znajduje się między innymi to jak Spring Boot zwiększa produktywność dewelopera i jak działa autokonfiguracja Spring.

Badania w artykule [24] oceniają czy Spring promuje dobre praktyki definiowania architektury. Analiza wyników wykazała, że zdecydowana (70%) większość projektów łączy wszystkie funkcje definicji architektury Spring. Autorzy artykułu wskazali na niewystarczające dokumentacje dobrych praktyk oraz podsumowali zalecenia dotyczące pomocy programistom.

W artykule [25] przedstawiono podejście do opracowywania aplikacji z wykorzystaniem środowiska do wdrażania mikroserwisów skalowalnych. Zaletą korzystania z tego podejścia jest to, że systemy mogą być ciągle rozwijane.

Autorzy książki [26] opisują proces wdrażania systemów opartych na architekturze mikrousług. Wprowadzają w obszar projektowania mikrousług pozwalając zrozumieć jak rozwiązywać problemy napotkane podczas programowania.

Celem badań w artykule [27] jest opracowanie aplikacji internetowej zachowując architekturę mikroserwisów Spring Boot. Artykuł przedstawia proponowane rozwiązanie jako zaletę, ponieważ można dodać więcej mikrousług bez wpływu na inne.

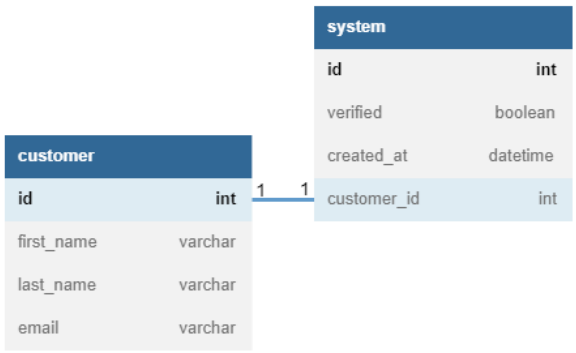
W niniejszej pracy podjęto tematykę tworzenia aplikacji chmurowych w Javie z wykorzystaniem narzędzi dostarczanych przez Spring Boot i Spring Cloud. Jak wskazują autorzy artykułu [24] sam Spring nie zawiera wielu dobrze opracowanych praktyk oraz dokumentacji. W pracy podjęto się zadania poszerzenia wiedzy na temat Spring Boot i Spring Cloud w tworzeniu aplikacji chmurowych, aby ułatwić zrozumienie związanych z tym pojęć i możliwości.

4. Aplikacje testowe

Do celów badawczych zostały opracowane dwie aplikacje. Z założenia są to proste aplikacje typu REST [28], które udostępniają dwa punkty końcowe (ang. endpoints). Pierwszy serwis *customer* udostępnia punkt końcowy *api/v1/customers*, gdzie można zarejestrować użytkownika poprzez wysłanie żądania metodą POST z danymi użytkownika. Drugi serwis *system* udostępnia punkt końcowy *api/v1/system/{customerId}*, który obsługuje żądanie przesłane metodą GET służący do zasymulowania weryfikacji użytkownika w systemie na podstawie podanego identyfikatora klienta. Na potrzeby badań serwis *system* zwraca pomyślną weryfikację użytkownika.

Aplikacje zostały wykorzystane w celu zbadania wydajności (liczby żądań na sekundę) oraz możliwości skalowalności aplikacji z wykorzystaniem narzędzi dostarczanych przez Spring Boot i Spring Cloud.

Diagram prezentuje strukturę bazy danych, wykorzystaną w aplikacjach i zawiera podstawowe informacje do zasymulowania rejestracji oraz weryfikacji użytkownika.



Rysunek 3: Diagram ERD.

W projekcie aplikacji testowych wykorzystano:

- Maven do zarządzania projektem;
- środowisko IntelliJ IDEA 2020.3 Ultimate Edition,
- SDK Java 14;
- bazę danych PostgreSQL wraz z graficznym interfejsem PGAdmin.

Spring wykorzystuje zewnętrzną konfigurację do definiowania właściwości aplikacji. Istnieje kilka sposobów konfiguracji, takich jak plik właściwości (application.properties), plik YAML (application.yml), zmienne środowiskowe lub argumenty wiersza poleceń. W projekcie zastosowano plik YAML ze względu na czytelną strukturę [29].

W aplikacjach zastosowano narzędzie Docker, w celu uruchomienia poszczególnych modułów jako oddzielne kontenery Docker. Cała konfiguracja usług znajduje się w jednym pliku o nazwie *docker-compose.yml*. Uruchomienie komendy *docker compose up -d* uruchomi aplikację jako izolowane środowisko, czyli kontenery [30].

Przygotowane zostały dwie aplikacje testowe:

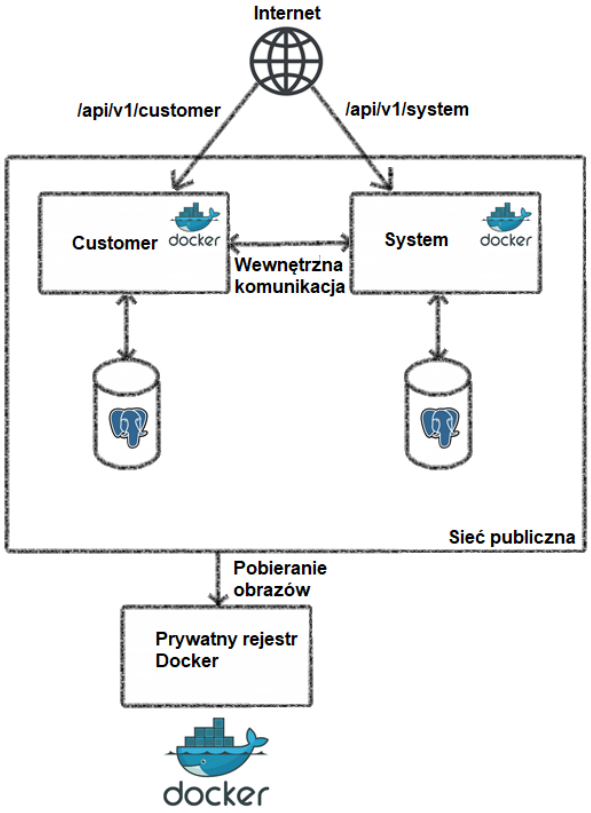
- aplikacja nieskalowana w Spring Boot;
- aplikacja skalowalna w Spring Boot i Spring Cloud.

Aplikacja nieskalowalna udostępnia jedynie dwa punkty końcowe służące do zasymulowania rejestracji użytkownika oraz weryfikacji w systemie. Zapisuje w bazie dane użytkownika oraz datę i rezultat weryfikacji. Aplikacja skalowalna pozwala na tworzenie wielu instancji serwisów oraz przygotowuje do wprowadzenia w środowisko chmurowe.

4.1. Aplikacja nieskalowalna

Pierwsza aplikacja jest podstawową aplikacją bazującą na architekturze mikroserwisów. Składa się z tylko jednej instancji każdego mikroserwisu. W projekcie znajduje się moduł nadrzędny oraz dwa moduły pod-

rzędne, czyli mikroserwisy. Schemat aplikacji nieskalowalnej przedstawiono na Rysunku 4.



Rysunek 4: Schemat aplikacji nieskalowalnej [31].

Aplikacja nieskalowalnej posiada podstawową funkcjonalność wymaganą do przeprowadzenia testów obciążeniowych. Udostępnia punkt końcowy, który umożliwia rejestrację użytkownika. Serwis *customer* komunikuje się z drugim serwisem *system* w celu weryfikacji użytkownika. Dane użytkownika oraz weryfikacji są zapisywane w bazie danych.

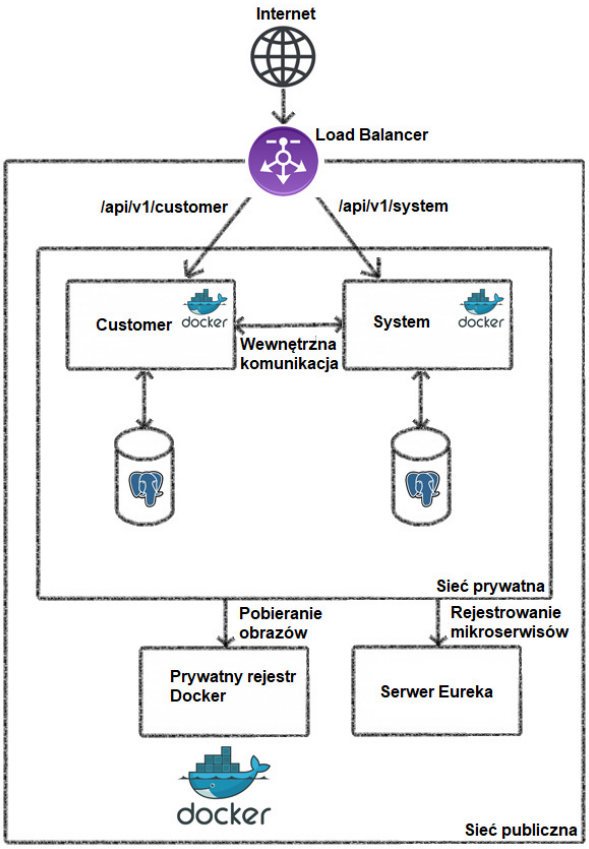
4.2. Aplikacja skalowalna

Druga, skalowalna aplikacja (Rys. 5) jest rozbudowaniem pierwszej, nieskalowalnej aplikacji. Zaimplementowano tutaj dodatkowo serwer Eureka, Spring Cloud OpenFeign oraz Spring Cloud Gateway.

Pakiet Spring Cloud Netflix dostarcza swój serwer Eureka. Panel serwera dostarcza informacje, takie jak zarejestrowani klienci Eureka, ich adres URL wraz z portem, a także liczba uruchomionych instancji [32].

Spring Cloud OpenFeign to deklaratywny klient REST dla aplikacji Spring Boot. Jest bardzo przydatnym narzędziem do tworzenia klientów usług internetowych za pomocą składni adnotacji. Zaletą OpenFeign jest prostota, do wywołania usługi wystarczy definicja interfejsu danego modułu [33].

Spring Cloud dostarcza swój własny Load Balancer – Spring Cloud Gateway. Umożliwia on opracowywanie mechanizmu przekierowań w aplikacjach opartych na architekturze mikrousług. Spring Cloud Gateway może działać zarówno jako Load Balancer jak i interfejs API [34].



Rysunek 5: Schemat aplikacji skalowalnej [31], [35].

Aplikacja skalowalna umożliwia przeprowadzenie testów obciążeniowych z wykorzystaniem wielu instancji serwisów. Posiada funkcjonalność opisaną w rozdziale 4.1. Konieczne jest, aby każda instancja była uruchomiona na innym, niezajętym porcie.

5. Metoda badań

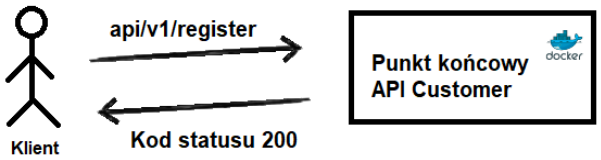
Aby pokazać możliwości skalowalności, dzięki narzędziom dostarczanym przez pakiet Spring Cloud, zostały przeprowadzone testy obciążeniowe na opracowanych aplikacjach. Każdy z eksperymentów został przeprowadzony w lokalnym środowisku. Skalowalność została zmierzona jako możliwość zwiększenia wydajności poprzez zwiększenie liczby instancji danego mikroserwisu.

W badaniach wzięto pod uwagę następujące czynniki:

- architekturę aplikacji (nieskalowalnej i skalowalna);
- liczbę instancji (1 dla aplikacji nieskalowalnej, od 1 do 5 dla poszczególnych serwisów aplikacji skalowalnej);
- liczbę zapytań (100, 1000, 10000).

5.1. Obiekt badań

Do celów badawczych zostały opracowane dwie aplikacje, które opisano w rozdziale 4. Aplikacje udostępniają punkt końcowy REST, rejestrujący nowego użytkownika w bazie danych (Rys. 6).



Rysunek 6: Schemat komunikacji użytkownika z serwisem customer [36].

Listing 1 przedstawia testowe dane rejestracji nowego użytkownika.

Listing 1: Parametry do rejestracji nowego użytkownika za pomocą serwisu customer

```
{
  "firstName": "firstName1",
  "lastName": "lastName1",
  "email": "email1@mail.com"
}
```

5.2. Środowisko badawcze

Eksperyment został przeprowadzony na lokalnym komputerze o następującej konfiguracji:

- procesor Intel Core i5-7300HQ CPU @ 2.50GHz;
- zainstalowana pamięć RAM 8,00 GB;
- typ systemu 64-bitowy system operacyjny, procesor x64;
- system operacyjny Windows 10 Home Edition.

5.3. Realizacja badań

Jako narzędzia badawcze zostały użyte programy popularny Apache JMeter oraz nowoczesny i konfigurowalny K6 [37].

Apache JMeter [38] to otwarto-źródłowy program Java, z prostym i intuicyjnym interfejsem graficznym (GUI). JMeter może przeprowadzać testy obciążenia i wydajności dla wielu różnych typów serwerów. Program jest elastyczny, pozwala na łatwą konfigurację parametrów testowych, takich jak liczba żądań i wątków.

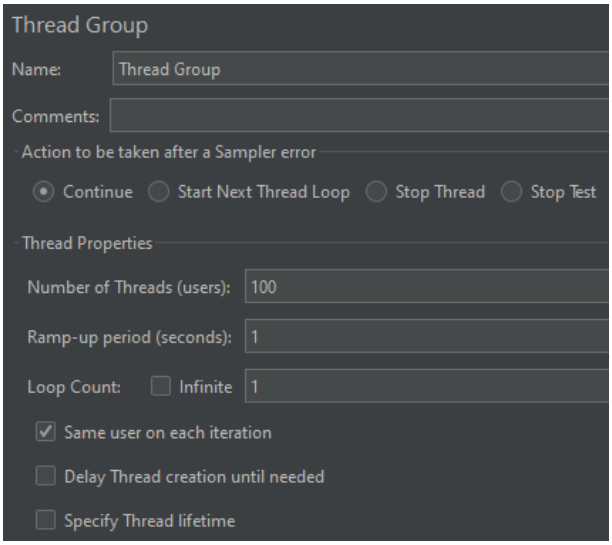
Na Rysunku 7 została przedstawiona konfiguracja programu, wykonująca konkretną liczbę zapytań (pole Number of Threads).

W oknie HTTP Request należy wprowadzić dane do zapytania REST, takie jak adres URL (pole Path), parametry żądania (pole Body Data) (Rys. 8).

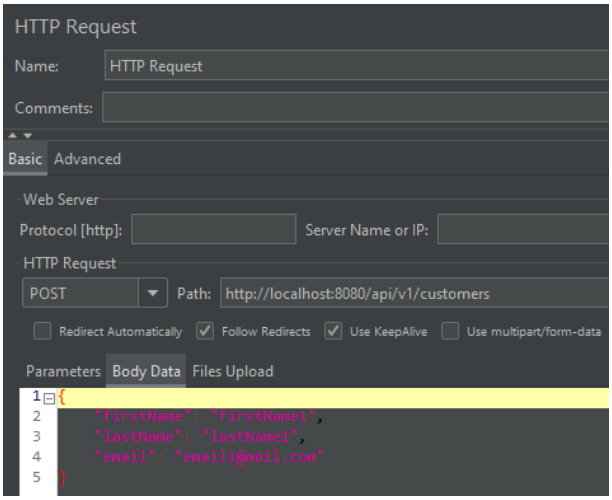
Po uruchomieniu zapytania w zakładkach podsumowania widoczne są różne parametry, takie jak czas wykonania danej liczby zapytań (Rys. 9).

K6 [39] jest również otwarto-źródłowym narzędziem testowania obciążenia (ang. load testing tool) do testowania wydajności API, mikroserwisów oraz stron internetowych. Program nie posiada GUI, może być używany jako wtyczka do środowiska Visual Studio Code. Wykorzystuje język JavaScript do tworzenia skryptów testów i posiada wiele możliwości konfiguracji.

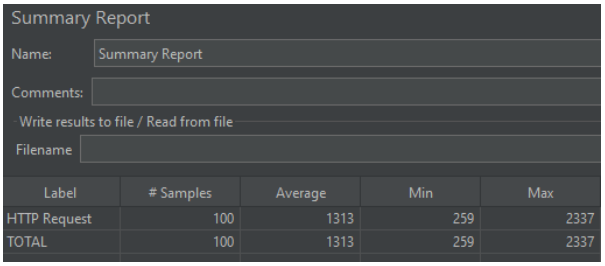
Na Listingu 2 przedstawiono skrypt JavaScript do przeprowadzenia testów obciążeniowych. Zmienna *options* odpowiada za konfigurację opcji testów, takich jak liczba wirtualnych maszyn – liczby zapytań (zmienna *vus*). Główna funkcja uruchamia testy i należy w niej zdefiniować metodę żądania, adres URL oraz parametry.



Rysunek 7: Konfiguracja grupy wątków programu JMeter.



Rysunek 8: Konfiguracja żądania HTTP w JMeter.



Rysunek 9: Przykładowy wynik uruchomienia testów w JMeter.

Listing 2: Skrypt JavaScript do przeprowadzenia testów obciążeniowych programem K6

```
// zaimportowanie modułu http do wykonywania zapytań API
import http from 'k6/http';
// konfiguracja testów
export const options = {
  // liczba maszyn wirtualnych (symyulujące użytkowników wykonujących zapytania)
  vus: 100,
  // czas trwania testów
  duration: '1s'
```

```
};
// główna funkcja uruchamiająca testy
export default function () {
  // parametry zapytania
  const payload = JSON.stringify({
    "firstName": "firstName1",
    "lastName": "lastName1",
    "email": "email1@mail.com"
  });
  // wykonanie testów
 http.post('http://localhost:8080/api/v1/customers'
, payload, {
    headers: {
      'Content-Type': 'application/json',
    },
  });
}
```

W trakcie badań skrypt uruchamiano w następujący sposób:

k6 run nazwa_pliku.js.

W konsoli wyświetlony zostanie wynik przeprowadzonych testów (Rys. 10).

```
data_received..................: 13 kB 5.4 kB/s
data_sent......................: 38 kB 17 kB/s
http_req_blocked...............: avg=18.92ms  min=0s       med=22ms     max=68.64ms
http_req_connecting............: avg=17.75ms  min=0s       med=21ms     max=66.83ms
http_req_duration..............: avg=973.63ms min=167.03ms med=1.12s    max=2.14s
  { expected_response:true }...: avg=973.63ms min=167.03ms med=1.12s    max=2.14s
http_req_failed................: 0.00% ✓ 0         ✗ 171
http_req_receiving.............: avg=193.83µs min=0s       med=0s       max=9.04ms
http_req_sending...............: avg=3.28ms   min=0s       med=999.6µs  max=24ms
http_req_tls_handshaking.......: avg=0s       min=0s       med=0s       max=0s
http_req_waiting...............: avg=970.15ms min=167.03ms med=1.12s    max=2.13s
http_reqs......................: 171   73.989316/s
iteration_duration.............: avg=993.04ms min=167.56ms med=1.12s    max=2.17s
iterations.....................: 171   73.989316/s
vus............................: 35    min=35      max=100
vus_max........................: 100   min=100     max=100
```

Rysunek 10 Przykładowy rezultat przeprowadzonego testu programem K6.

6. Wyniki badań

Wyniki przeprowadzonych testów zostały przedstawione w Tabelach 1-6 oraz na Rysunkach 11-16.

Wydajność aplikacji badano przy różnej licznie zapytań: 100, 1000 i 10000.

W tabelach i na wykresach zastosowano oznaczenia:

- AN – aplikacja nieskalowalna;
- AS – aplikacja skalowalna;
- AxB: A – liczba instancji serwisu *system*, B – liczba instancji serwisu *customer*.

Tabela 1: Wydajność aplikacji przy liczbie 100 zapytań

Aplikacja	Wydajność (liczba żądań na sekundę)					
	AN	AS 1x1	AS 1x2	AS 1x3	AS 1x4	AS 1x5
JMeter	42.7	75.5	60.1	44.1	94.5	79.1
K6	302.9	124.9	143.0	239.0	194.0	137.6

Tabela 2: Wydajność aplikacji przy liczbie 1000 zapytań

Aplikacja	Wydajność (liczba żądań na sekundę)					
	AN	AS 1x1	AS 1x2	AS 1x3	AS 1x4	AS 1x5
JMeter	100.6	91.4	99.4	106.8	137.4	137.4
K6	193.2	181.9	187.8	254.5	160.2	189.2

Tabela 3: Wydajność aplikacji przy liczbie 10000 zapytań

Aplikacja	Wydajność (liczba żądań na sekundę)					
	AN	AS 1x1	AS 1x2	AS 1x3	AS 1x4	AS 1x5
JMeter	84.5	83.6	158.5	266.2	201.3	153.3
K6	132.7	104.8	100.3	110.6	70.0	119.1

Tabela 4: Wydajność aplikacji przy liczbie 100 zapytań

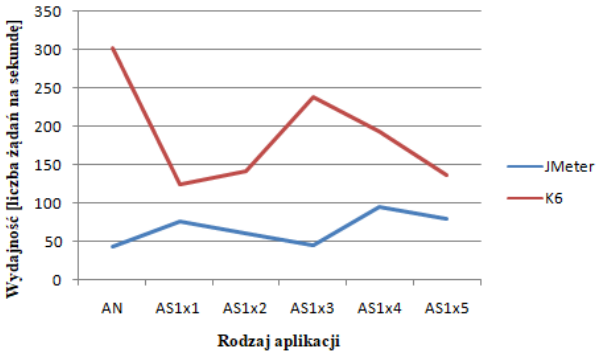
Aplikacja	Wydajność (liczba żądań na sekundę)					
	AN	AS 1x1	AS 2x2	AS 3x3	AS 4x4	AS 5x5
JMeter	42.7	75.5	30.0	98.7	72.0	98.8
K6	302.9	124.9	38.5	215.5	97.6	154.0

Tabela 5: Wydajność aplikacji przy liczbie 1000 zapytań

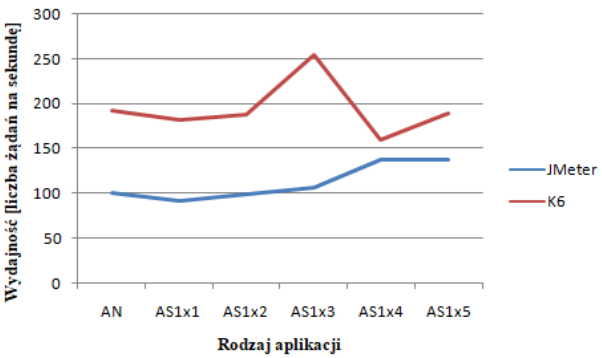
Aplikacja	Wydajność (liczba żądań na sekundę)					
	AN	AS 1x1	AS 2x2	AS 3x3	AS 4x4	AS 5x5
JMeter	100.6	91.4	100.2	136.6	147.2	115.8
K6	193.2	181.9	197.0	100.2	157.4	102.0

Tabela 6: Wydajność aplikacji przy liczbie 10000 zapytań

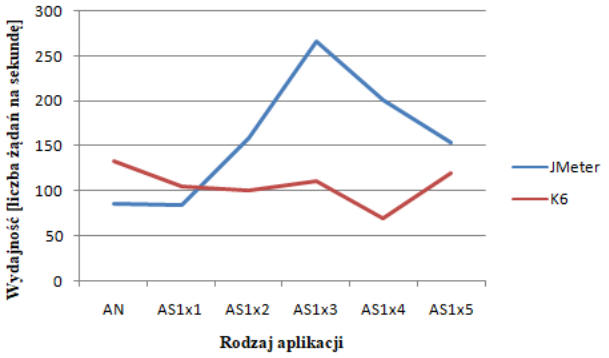
Aplikacja	Wydajność (liczba żądań na sekundę)					
	AN	AS 1x1	AS 2x2	AS 3x3	AS 4x4	AS 5x5
JMeter	84.5	83.6	101.9	146.8	95.0	84.0
K6	132.7	104.8	147.2	84.7	114.2	82.0



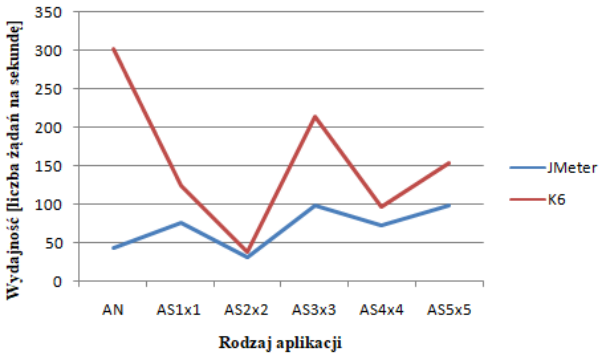
Rysunek 11: Wydajność aplikacji przy liczbie 100 zapytań.



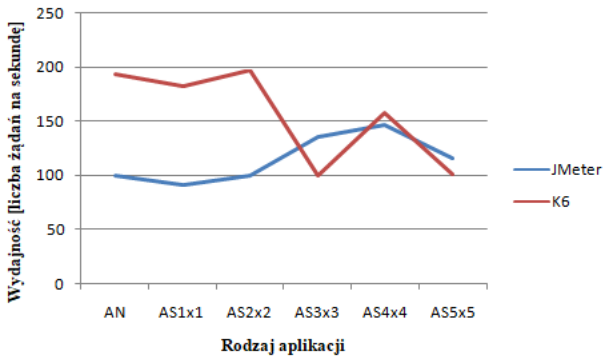
Rysunek 12: Wydajność aplikacji przy liczbie 1000 zapytań.



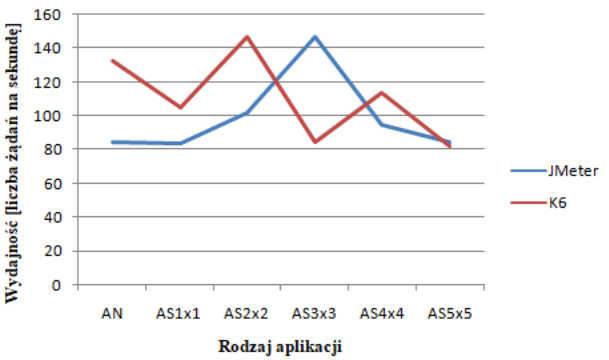
Rysunek 13: Wydajność aplikacji przy liczbie 10000 zapytań.



Rysunek 14: Wydajność aplikacji przy liczbie 100 zapytań.



Rysunek 15: Wydajność aplikacji przy liczbie 1000 zapytań.



Rysunek 16: Wydajność aplikacji przy liczbie 10000 zapytań.

7. Analiza wyników badań

Otrzymane wyniki pokazały, że Spring Cloud pozwala na skalowanie aplikacji, za pomocą narzędzi, takich jak Spring Cloud OpenFeign oraz Spring Cloud Gateway.

Warto zaznaczyć, że w pracy opracowane aplikacje były uruchomione w lokalnym środowisku, w którym nie zastosowano skalowania poziomowego (zwiększenia zasobów, tj. pamięci RAM lub CPU) lub pionowego (większej liczbie maszyn). Badania w lokalnym środowisku nie wykazały znaczącej poprawy wydajności ze względu na rodzaj skalowania (tj. zwiększanie liczby instancji poszczególnych mikroserwisów) oraz ograniczenia konfiguracji maszyny, tj. zwiększenia zasobów. Badania miały na celu jedynie pokazać efekty działających aplikacji z użyciem narzędzi dostarczanych przez Spring Cloud do tworzenia aplikacji, które mogą działać w chmurze.

Wybór narzędzi do przeprowadzenia testów wydajnościowych miał znaczenie. Zarówno narzędzie JMeter jak i K6 były dosyć łatwe w konfiguracji. K6 pozwolił na znacząco szybsze przeprowadzenie testów. Jest narzędziem lepiej zoptymalizowanym pod kątem wykonywania testów obciążeniowych. K6 pozwala również na szerszą konfigurację testów. JMeter posiada bardziej przyjazny i intuicyjny interfejs. K6 wymaga znajomości języka JavaScript oraz posługiwania się konsolą CLI.

Duże różnice w wynikach przy liczbie 100 (Tabele 1, 4 i Rysunki 11, 14) i 10000 (Tabele 3, 6 i Rysunki 13, 16) zapytań wynikały z zastosowanych aplikacji do przeprowadzania testów. K6 jest bardziej nowoczesnym narzędziem w porównaniu do JMeter, jednym z najlepszych dostępnych do tego typu testów [45].

Kolejnym powodem tak dużych różnic w wynikach była zbyt mała (100) lub zbyt duża liczba zapytań (1000). Przy liczbie 100 zapytań aplikacje szybko przetwarzały żądania, a przy liczbie 10000 zapytań platforma testowa (lokalny komputer) była zbyt obciążona.

Na różnice wpływ miało również lokalne środowisko, w którym nie zastosowano skalowania poziomowego lub pionowego. Rodzaj skalowania ma znaczenie w przypadku tego typu testów. W przypadku skalowania pionowego dodawane jest więcej zasobów (np. pamięci lub CPU), a w przypadku skalowania poziomowego ruch sieciowy jest rozprowadzany po większej liczbie maszyn.

Najbardziej optymalna okazała się liczba 1000 (Tabele 2, 5 i Rysunki 12, 15) zapytań. Wydajność aplikacji nieskalowanej i skalowalnej utrzymywała się na podobnym poziomie, ponieważ skalowanie aplikacji odbywało się poprzez zwiększanie liczby instancji poszczególnych serwisów. Nie zastosowano w badaniach skalowania poziomowego (zwiększenia zasobów, tj. pamięci RAM lub CPU) lub pionowego (większej liczbie maszyn) ze względu na ograniczenia spowodowane lokalnym środowiskiem.

Uruchomienie aplikacji w środowisku chmurowym, takim jak Kubernetes pozwala na skalowanie pionowe lub poziome. Skalowanie poziome oznacza to, że zwiększone obciążenie aplikacji zbyt dużych ruchem sieciowym zostanie rozprowadzone po większej liczbie

maszyn. Skalowanie pionowe polega na zwiększeniu zasobów (np. pamięci lub CPU) maszyn już uruchomionych. Pozwala to na przeprowadzenie nowych testów wydajnościowych w platformie chmurowej Kubernetes.

8. Wnioski i przyszłe kierunki badań

Narzędzia Spring Boot i rozszerzenie Spring Cloud zdecydowanie pomagają programistom w tworzeniu aplikacji bazujących na architekturze mikroserwisów w środowiskach chmurowych. W ostatnich latach można znaleźć więcej pozycji w literaturze, w których przedstawiona zostaje architektura mikroserwisów z użyciem Spring Boot, Spring Cloud. Również wzrasta liczba badań skupiających się na wydajności i skalowalności takich aplikacji.

W pracy przeanalizowano głównie możliwości jakie oferuje Spring Boot i Spring Cloud w tworzeniu aplikacji chmurowych. Opracowane aplikacje pokazały jakie korzyści daje wykorzystanie odpowiednich narzędzi takich jak Spring Cloud OpenFeign czy Spring Cloud Gateway. Aplikacje opracowane w pracy pozwoliły na przeprowadzenie testów obciążeniowych. Aplikacja nieskalowana skupia się jedynie na stworzeniu własnych mikroserwisów z użyciem Spring Boot. Aplikacja skalowalna z wykorzystaniem narzędzi dostarczanych przez Spring Boot i Spring Cloud pozwala na uruchomienie wielu instancji mikroserwisów. Lokalne środowisko nie daje możliwości pokazania prawdziwego skalowania aplikacji w środowisku produkcyjnym.

Następnym krokiem w badaniach w świecie mikroserwisów powinno być wdrożenie aplikacji przedstawionej w rozdziale 4 do środowiska chmurowego. Istnieje wiele platform chmurowych pozwalających na uruchomienie aplikacji z wykorzystaniem Spring Cloud, takich jak Kubernetes.

Artykuł miał charakter przeglądowy, z ukierunkowaniem na wskazanie możliwości oferowanych przez Spring do tworzenia aplikacji chmurowych. Badania przeprowadzone w pracy pokazały jedynie możliwości skalowania. Wdrożenie aplikacji do środowiska chmurowego Kubernetes pozwoliłoby na skalowanie pionowe lub poziome, co otwiera to nowe możliwości do przeprowadzenia kolejnych badań.

Literatura

[1] Strona internetowa z informacjami na temat branż korzystających z architektury mikrousług, https://codeandpepper.com/companies-using-microservices, [1.08.2022].

[2] Strona internetowa z informacjami na temat powodów, dla których branże zwracają się ku mikroserwisom, https://annexbyte.com/blog/industries-turning-to-microservices, [1.08.2022].

[3] Strona internetowa z informacjami na temat technologii i języków do wyboru w budowaniu architektury mikroserwisów, https://www.mindinventory.com/blog/technologies-for-mircroservices-architecture, [1.08.2022].

[4] Strona internetowa z informacjami na temat popularności technologii i języków do wyboru w budowaniu architektury mikroserwisów, https://www.thirdrocktechkno.com/blog/top-languages-for-microservices-architecture, [1.08.2022].

[5] Strona internetowa z informacjami na temat narzędzi do budowania mikroserwisów, https://dzone.com/articles/30top-tools-for-building-microservices-on-all-leve, [1.08.2022].

[6] Strona internetowa z informacjami na temat Spring Cloud, https://spring.io/projects/spring-cloud, [1.08.2022].

[7] Strona internetowa z informacjami na temat mikroserwisów, https://aws.amazon.com/microservices, [1.08.2022].

[8] Strona internetowa z informacjami na temat Spring Boot, https://spring.io/projects/spring-boot, [1.08.2022].

[9] Strona internetowa z informacjami na temat Spring Cloud, https://spring.io/projects/spring-cloud, [1.08.2022].

[10] Strona internetowa z informacjami na temat Service Discovery, https://avinetworks.com/glossary/service-discovery, [1.08.2022].

[11] Strona internetowa z informacjami na temat mechanizmu serwera Eureka, https://medium.com/@ijayakantha/microservices-service-registration-and-discovery-with-netflix-eureka-9a2aa729da96, [1.08.2022].

[12] Strona internetowa z informacjami na temat Load Balancer, https://www.ovhcloud.com/pl/public-cloud/what-load-balancing, [1.08.2022].

[13] Strona internetowa z informacjami na temat Docker, https://docs.docker.com/get-started/overview, [1.08.2022].

[14] Strona internetowa z informacjami na temat mechanizmu CI/CD, https://docs.gitlab.com/ee/ci/introduction, [1.08.2022].

[15] Strona internetowa z informacjami na temat Kubernetes, https://www.mirantis.com/cloud-native-concepts/getting-started-with-kubernetes/what-is-kubernetes, [1.08.2022].

[16] Strona internetowa z informacjami na temat Kubernetes, https://www.mirantis.com/cloud-native-concepts/getting-started-with-kubernetes/what-is-kubernetes, [1.08.2022].

[17] Strona internetowa z informacjami na temat języku programowania Borg, https://memory-alpha.fandom.com/wiki/Borg_language, [1.08.2022].

[18] Strona internetowa z informacjami na temat języku programowania Omega, https://hackage.haskell.org/package/omega, [1.08.2022].

[19] C. M. Aderaldo, N. C. Mendonça, C. Pahl, P. Jamshidi, Benchmark Requirements for Microservices Architecture Research, 2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for

Architecture-Based Software Engineering (ECASE) (2017) 8-13.

[20] D. Rajput, Mastering Spring Boot 2.0: Build modern, cloud-native, and distributed systems using Spring Boot, Packt Publishing Ltd, 2018.

[21] E. Wolff, Microservices: flexible software architecture, Addison-Wesley Professional, 2016.

[22] J. Carnell, I. H. Sánchez, Spring microservices in action, Simon and Schuster, 2021.

[23] K. S. P. Reddy, Beginning Spring Boot 2: Applications and microservices with the Spring framework, Apress, 2017.

[24] Q. Perez, A. Le Borgne, C. Urtado, S. Vauttier, An Empirical Study about Software Architecture Configuration Practices with the Java Spring Framework, SEKE: Software Engineering and Knowledge Engineering (2019) 465-468.

[25] V. Saquicela, G. Campoverde, J. Avila, M. E. Fajardo, Building microservices for scalability and availability: Step by step, from beginning to end, International Conference on Software Process Improvement, Springer, Cham (2020) 169-184.

[26] S. Sharma, Mastering Microservices with Java - Third Edition: Build enterprise microservices with Spring Boot 2.0, Spring Cloud, and Angular, Packt Publishing Ltd, 2019.

[27] S. Hatma, J. D. Puji, T. Aris, Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot, Procedia Computer Science (2017) 124, 736-743.

[28] Strona internetowa z informacjami na temat REST API, https://www.redhat.com/en/topics/api/what-is-a-rest-api, [1.08.2022].

[29] Strona internetowa z informacjami na temat konfiguracji zewnętrznej Spring Boot, https://www.baeldung.com/spring-boot-yaml-vs-properties, [1.08.2022].

[30] Strona internetowa z informacjami na temat uruchamiania aplikacji Spring Boot w kontenerach Docker, https://www.baeldung.com/dockerizing-spring-boot-application, [1.08.2022].

[31] Strona internetowa z informacjami na temat tworzenia aplikacji monolitycznych oraz opartych o architekturę mikrousług, https://www.dineshonjava.com/microservices-with-spring-boot, [1.08.2022].

[32] Strona internetowa z informacjami na temat serwera Eureka w aplikacji Spring Boot, https://www.tutorialspoint.com/spring_boot/spring_boot_eureka_server.htm, [1.08.2022].

[33] Strona internetowa z informacjami na temat Spring Cloud OpenFeign, https://www.baeldung.com/spring-cloud-openfeign, [1.08.2022].

[34] Strona internetowa z informacjami na temat Load Balancer i interfejsu API, https://www.techtarget.com/searchitoperations/answer/Whats-the-role-of-an-application-load-balancer-vs-API-gateway, [1.08.2022].

[35] Strona internetowa z informacjami na tworzenia aplikacji opartych o architekturę mikrousług z wykorzystaniem narzędzi Spring Cloud, https://spring.io/microservices, [1.08.2022].

[36] Strona internetowa z informacjami na temat mechanizmu REST API, https://www.geeksforgeeks.org/rest-api-introduction, [1.08.2022].

[37] Strona internetowa z informacjami na temat narzędzi do przeprowadzania testów obciążeniowych, https://pflb.us/blog/best-load-testing-tools, [1.08.2022].

[38] Strona internetowa z informacjami na temat narzędzia JMeter, https://www.simplilearn.com/tutorials/jmeter-tutorial/jmeter-performance-testing, [1.08.2022].

[39] Strona internetowa z informacjami na temat narzędzia K6, https://www.geeksforgeeks.org/performance-testing-with-k6, [1.08.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 121–124
Received: 27 October 2022
Accepted: 29 November 2022

PaaS platform comparison based on users feedback

Porównanie platform typu PaaS na podstawie opinii użytkowników

Konrad Prządka*, Mateusz Saputa*, Jakub Smołka

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

Abstract
This article presents a description of the opinion study about SalesForce and ServiceNow platforms. These are modern PaaS (Platform as a Service) environments that are gaining more and more popularity. Their main common feature is the use of cloud technology in creating and maintaining internet applications. The scope of the study included such platform properties as: applicability, interface assessment, as well as the required level of knowledge of code development. For the purpose of comparing the platforms on each of them, the respondents were asked to create an identical business form. It covers different types of fields and how to validate them. The research group received full access to the test platforms. Training was conducted to introduce the respondents to a given system. The prerequisite for completing the survey was that they created a form based on a template on each of the platforms. The answers to the questions were in the form of a point scale, where the minimum value was 1 and the highest value was 5. Based on the results obtained, it was found that the SalesForce platform is a better choice, gaining more points compared to ServiceNow by about 15%.

Keywords: salesforce; servicenow; PaaS

Streszczenie
Niniejszy artykuł przedstawia opis badania opinii użytkowników na temat platform SalesForce i ServiceNow. Są to nowoczesne środowiska typu PaaS (Platform as a Service) uzyskujące coraz większą popularność. Ich główną cechą wspólną jest zastosowanie technologii chmurowej w tworzeniu i utrzymywaniu aplikacji internetowych. Zakres badania obejmował takie właściwości platform jak: stosowalność, ocena interfejsu, a także wymagany poziom znajomości programowania od twórców aplikacji. Na rzecz uzyskania miarodajnych ocen platform, ankietowani zostali poddani szkoleniu, które miało na celu wprowadzenie ich w opisywane środowiska. Następnym etapem badania była część praktyczna, gdzie uczestnicy otrzymali zadanie zbudowania identycznych formularzy biznesowych z wykorzystaniem obu systemów. Po wykonaniu zadania, następowało badanie ankietowe. Odpowiedzi udzielane na pytania miały formę skali punktowej, gdzie minimalną wartością było 1, a największą 5. Na podstawie uzyskanych wyników stwierdzono, iż platforma SalesForce jest korzystniejszym wyborem w zastosowaniach chmurowych, uzyskując około 15% lepszą ocenę respondentów w porównaniu z ServiceNow.

Słowa kluczowe: salesforce; servicenow; PaaS

*Corresponding author
Email address: konrad.przadka@pollub.edu.pl (K. Prządka), mateusz.saputa@pollub.edu.pl (M. Saputa)

1. Wstęp

Wraz z rozwojem rynku usług chmurowych coraz popularniejszy staje się model usługi (PaaS). Platforma jako usługa to typ środowiska kompletnego, pozwalającego na tworzenie i utrzymanie aplikacji internetowych. System PaaS pozwala na zbudowanie całej infrastruktury wymaganej przez klientów. Mogą się w niej zawierać: serwery, magazyny danych, a także oprogramowanie pośredniczące i narzędzia deweloperskie. W zależności od oferty dostawców, możliwe jest korzystanie z usług analizy biznesowej i systemów zarządzania bazami danych. Środowiska chmurowe wykorzystywane są przy tworzeniu innowacyjnych technologii, takich jak sztuczna inteligencja (AI), czatboty, blockchain, a także Internet rzeczy (IoT). Ponadto PaaS obejmuje także pakiety narzędzi do tworzenia aplikacji, w tym m.in. usługi cloud native, Kubernetes, Docker i silniki kontenerowe. Narzędzia te zaprojektowano do obsługi całego cyklu życia aplikacji internetowej: tworzenia, testowania, wdrażania, zarządzania i aktualizowania. Dostawcy najczęściej wymagają płatności zgodnie z rzeczywistym użyciem w zależności od wybranej oferty. Dostęp do zakupionych usług jest uzyskiwany za pośrednictwem połączenia internetowego. Technologia PaaS pozwala twórcom na wybór oprogramowania od jednego lub wielu dostawców usług w chmurze. Każdy z dostawców jest odpowiedzialny za zarządzanie bezpieczeństwem, systemami operacyjnymi, oprogramowaniem serwera i kopiami zapasowymi. Firma Salesforce jest jedną z największych korporacji zajmujących się technologiami chmurowymi. Została zainicjowana w 1999 przez byłego pracownika firmy Oracle - Marca Benioffa. Salesforce udostępnia wiele usług w tym przeznaczoną dla deweloperów Salesforce App Cloud, która jest zbiorem narzędzi programistycznych pozwalających na szybkie tworzenie aplikacji działających na platformie. Jednym z ważniejszych elementów Salesforce App Cloud jest platforma Force.com, która pozwala deweloperom na tworzenie witryn i aplikacji internetowych zintegrowanych

z główną platformą Salesforce. Platforma ServiceNow powstała w 2003 roku. Jest skierowana na zapewnianie wsparcia technicznego w zakresie zarządzania. Pozwala również na realizowanie aplikacji, które nie wymagają od ich twórców zaawansowanych umiejętności programistycznych.

2. Przegląd literatury

Przetwarzanie w chmurze to główny trend w bieżących badaniach do budowania skalowalnych i rozproszonych środowisk obliczeniowych [1]. W szczególności Cloud computing, jest to sposób na umożliwienie wygodnego dostępu sieciowego na żądanie do wspólnej puli konfigurowalnych zasobów obliczeniowych (np. sieci, serwerów, pamięci, aplikacji i usług), które można szybko udostępniać i zwalniać przy minimalnym wysiłku zarządzania lub interakcji z dostawcą usług. Można wyróżnić kilka warstw chmury obliczeniowej tj. infrastruktura, platforma i warstwy aplikacji, które dostarczają użytkownikom końcowym funkcjonalności określanych odpowiednio jako IaaS (Infrastructure as a Service), PaaS i SaaS (Software as a Service). Amazon Elastic Compute Cloud (Amazon EC2), Salesforce i Google App Engine to trzej najbardziej znani dostawcy platform chmurowych [2], ale w ciągu ostatnich miesięcy, oferta rozwinęła się bardzo szybko. Poza tym wiele kluczowych firm w branży IT oferuje również rozwiązania chmury prywatnej dla swoich centrów danych. W szczególności udostępniane są zasoby IaaS (tj. procesor, pamięć, pamięć masowa, sieć, system równoważenia obciążenia, zapora sieciowa i publiczny adres IP) z Amazon EC25 oraz InstaCompute z firmy Tata Communications [3].

Przetwarzanie w chmurze wspiera model tworzenia aplikacji, który łączy autorski kod z funkcjonalnością świadczoną przez istniejące, wyselekcjonowane serwisy internetowe. Kierowane na konsumentów aplikacje (w szczególności te przeznaczone na platformy mobilne), wchodzą w interakcje z wysoce skalowalnymi i niezawodnymi usługami sieciowymi „back-end", które są stale utrzymywane w dobrze połączonych, bezpiecznych centrach danych. Ponadto technologie informatyczne dla przedsiębiorstw (IT) koncentrują się na wdrażaniu zarówno sprzętu, jak infrastruktury oprogramowania do udostępniania ich zasobów [4, 5] cyfrowych

Interfejsy API stały się kluczowym elementem nowoczesnej gospodarki cyfrowej, mimo że są najdłużej żyjącymi i najdroższymi artefaktami oprogramowania. Opisany artykuł to pierwszy krok w kierunku rozwoju systemu do implementacji API governance poprzez wdrażanie najnowszych postępów w technologiach platformy chmurowej jako usługi [6]. Ten artykuł opisuje przegląd wstępnych badań nad narzędziami, które mają pomóc programistom i interesariuszom biznesowym zrozumieć kontrolę podobieństwa i zmian w ramach API governance.

W artykule [7] przedstawiono koncepcję rozwoju następnej generacji systemów PaaS [8]. W tym celu wdrożono prototyp EAGER (The Enforced API Governance Engine for REST). Korzystając z tego prototypu, empirycznie oceniona została jego skuteczność za pomocą losowo generowanych API. Przeanalizowano również prototyp za pomocą szeregu internetowych API z popularnych serwisów ecommerce oraz społecznościowych miejsc nawiązywania kontaktów. Wyniki eksperymentalne wskazują, że mechanizm jest wydajny i zapewnia dokładne wyniki w większości okoliczności. Przetestowano ważność podejścia poprzez porównanie wyników obliczonych przez formalne mechanizmy z tymi dostarczonymi przez niektórych programistów podczas ręcznej analizy kilku internetowych interfejsów API.

3. Cel badań

Celem pracy badań jest uzyskanie opinii użytkowników na temat platform SalesForce i ServiceNow na podstawie zdobytych doświadczeń podczas tworzenia identycznych formularzy w obu środowiskach. W tym celu postawiono hipotezę: jedna z badanych platform jest wygodniejsza do obsługi przez użytkownika.

4. Grupa badawcza

Grupa badawcza składała się z 15 osób w wieku od 23 do 24 lat. Wszyscy uczestnicy byli aktywnymi studentami Politechniki Lubelskiej na kierunku Informatyka. Respondenci posiadali wykształcenie techniczne na poziomie nie niższym niż stopień inżyniera. Warunkiem do wzięcia udziału w badaniu był brak doświadczenia z platformami SalesForce i ServiceNow. Zebrana w ten sposób grupa badawcza charakteryzowała się umiejętnościami pozwalającymi na posługiwanie się podstawowymi narzędziami przeglądarki i tworzenia prostego kodu. Ponadto uczestnicy potrafili szybko przyswoić wiedzę przekazaną im w szkoleniu prezentującym obie platformy oraz wykonać postawione przed nimi zadanie. Brak styczności uczestników badania z omawianymi platformami pozwolił na uzyskanie jedynie takich ocen, które były wynikiem doświadczeń zebranych podczas wykonywanego zadania.

5. Ankieta

Ankieta została przeprowadzona w formie elektronicznej z wykorzystaniem narzędzia Google Forms. Warunkiem udziału było ukończenie szkolenia oraz podjęcie próby wykonania zadania. Badanie ankietowe składało się z trzech sekcji. Pierwsza, zawierała pięć pytań i dotyczyła stosowalności. Uczestnicy oceniali w niej stopień trudności wykorzystania takich funkcjonalności jak:

- tworzenie pól - w tym celu wymagana była znajomość zagadnienia tworzenia sprecyzowanych pól, ich typów, długości oraz nazwy,
- tworzenie modeli obiektów - w tym celu wymagana była znajomość obiektów standardowych oraz zwyczajnych oraz umiejętności ich odróżnienia,
- wdrożenie - ocena łatwości poznania, przyswojenia oraz zapamiętania mechanizmów działania platform,

- tworzenie układu formularza - wykorzystanie dostępnych ustawień lub formatów uwzględniających układ pól według określonego schematu. Uwzględnienie styli arkusza, takich jak: czcionka, kolor oraz wielkość pola,
- walidacja danych - ustawienie odpowiedniej walidacji do danego pola oraz sprawdzenie jej działania.

Kolejne dwie sekcje zawierały po jednym zagadnieniu i odnosiły się odpowiednio do:

- graficznej reprezentacji oprogramowania sterującego - przejrzystości interfejsu, intuicyjności działania, kolorów i rozmieszczenia elementów,
- ilości kodu wykorzystanego przy wykonywaniu zadania.

W celu uzyskania opinii ankietowani odpowiedzieli na następujące pytania:

1) Jak Pan/Pani ocenia łatwość tworzenia pól formularza na platformie SalesForce/ServiceNow? (1pkt - 5pkt, im więcej tym lepiej).
2) Jak Pan/Pani ocenia łatwość tworzenia modeli obiektów formularza na platformie SalesForce/ServiceNow? (1pkt - 5pkt, im więcej tym lepiej).
3) Jak Pan/Pani ocenia łatwość wdrożenia na platformie SalesForce/ServiceNow? (1pkt - 5pkt, im więcej tym lepiej).
4) Jak Pan/Pani ocenia łatwość tworzenia układu formularza na platformie SalesForce/ServiceNow? (1pkt - 5pkt, im więcej tym lepiej).
5) Jak Pan/Pani ocenia wygląd interfejsu platformy SalesForce/ServiceNow? (1pkt - 5pkt, im więcej tym lepiej).
6) Jak Pan/Pani ocenia łatwość tworzenia walidacji pól formularza na platformie SalesForce/ServiceNow? (1pkt - 5pkt, im więcej tym lepiej).
7) Jak Pan/Pani ocenia ilość potrzebnego kodu do zbudowania formularza na platformie SalesForce/ServiceNow? (1pkt - 5pkt, im mniej kodu tym lepiej).

Uczestnicy mogli odpowiedzieć na pytanie poprzez wybór jednej z pięciu odpowiedzi. Każda z nich oznaczała inną liczbę punktów w skali od 1 do 5. Większa liczba punktów oznaczała lepszą ocenę danego zagadnienia w wybranej platformie. Każdy ankietowany oddawał łącznie 14 odpowiedzi, po siedem dla każdej platformy.

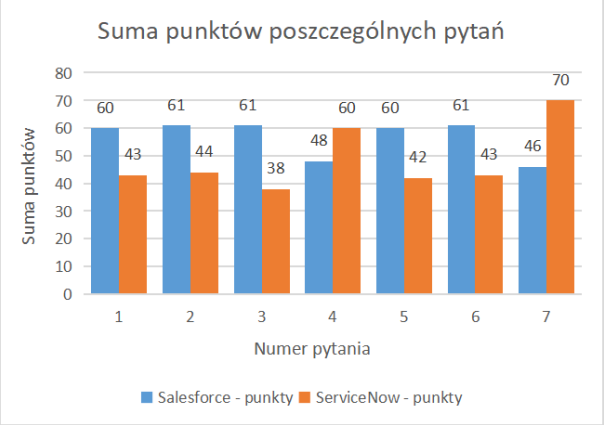
6. Wyniki

W ankiecie wzięło udział 15 osób. Każda z nich odpowiedziała poprawnie. W Tabeli 1 przedstawiono zestawienie pokazujące sumę wszystkich punktów w zależności od zagadnienia oraz rodzaju platformy. W sekcji stosowalności składającej się z pierwszych pięciu pytań, platforma SalesForce uzyskała więcej punktów niż platforma ServiceNow. Jedynym aspektem, w którym druga z platform uzyskała przewagę była funkcjonalność tworzenia układu formularza.

Tabela 1: Suma punktów z poszczególnych pytań

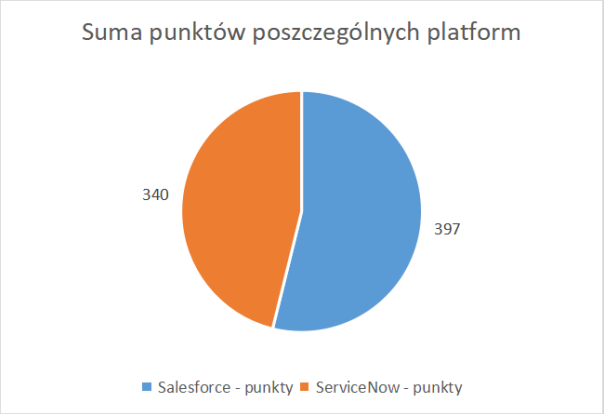
Nr. pytania	Salesforce punkty	ServiceNow punkty
1	60	43
2	61	44
3	61	38
4	48	60
5	60	42
6	61	43
7	46	70

Ankietowani lepiej ocenili również interfejs platformy SalesForce przyznając jej w sumie 61 punktów. ServiceNow uzyskał około 30% gorszy wynik w tej kategorii. W ostatnim pytaniu, odnoszącym się do ilości niezbędnego kodu do zbudowania formularza, platforma ServiceNow otrzymała maksymalną możliwą liczbę punktów (Rysunek 1).



Rysunek 1: Suma punktów z poszczególnych pytań.

Jednocześnie platforma SalesForce odnotowała w tym zagadnieniu 46 punktów. Jest to najsłabiej punktowana odpowiedź tego środowiska chmurowego. Na wykresie (Rysunek 2) przedstawiono zestawienie sumy wszystkich punktów zdobytych przez poszczególne platformy.



Rysunek 2: Suma punktów z poszczególnych platform.

Suma punktów uzyskanych we wszystkich pytaniach przez platformę ServiceNow jest równa 340. Wynik ten jest o około 15% gorszy niż ten uzyskany przez platformę SalesForce.

7. Wnioski

Doświadczenie zebrane podczas szkolenia oraz wykonane zadanie zbudowania formularzy na obu platformach, pozwoliło uczestnikom badania na wskazanie preferowanej platformy w zależności od poruszanego zagadnienia. Zebrane wyniki pozwalają stwierdzić, iż usługi oferowane przez dostawców ServiceNow, wymagają zdecydowanie mniej linii kodu od klientów niż w przypadku SalesForce, a także lepiej sprawdzają się przy budowie układu formularza. Jak wynika z uzyskanych rezultatów, respondenci uznali jednak, że rozwiązania drugiej platformy takie jak tworzenie pól, modeli obiektów oraz walidacja danych są bardziej przyjazne dla użytkownika. SalesForce został również wskazany jako platforma o atrakcyjniejszym interfejsie oraz pozwalająca na szybsze wdrożenie.

Platforma SalesForce uzyskała większą liczbę punktów. Oznacza to, że ankietowani uznali tę platformę jako preferowaną. Prawie we wszystkich pytaniach zdobyła znaczną przewagę, osiągając w sumie około 15% więcej punktów. Takie rezultaty mogą być konsekwencją charakteru grupy badawczej, która była złożona z osób zaznajomionych z programowaniem. W przypadku osób słabiej znających techniki tworzenia kodu, to właśnie ten aspekt badania mógłby zdecydować o przeciwnym wyniku.

Literatura

[1] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, L. Seinturier, A federated multi-cloud PaaS infrastructure, IEEE 5th International Conference on Cloud Computing (2012) 392-399.

[2] M. Sellami, S. Yangui, M. Mohamed, S. Tata, PaaS-independent provisioning and management of applications in the cloud, IEEE 6th International Conference on Cloud Computing (2013) 693-700.

[3] J. Hermann, A. David, A. Wagner, M. Ruskowski, Considering interdependencies for a dynamic generation of process chains for production as a service, Procedia Manufacturing 51 (2020) 1454-1461.

[4] F. Zalila, S. Challita, P. Merle, Model-driven cloud resource management with OCCIware, Future Generation Computer Systems 99 (2019) 260-277.

[5] P. Trakadas, et.al., Hybrid clouds for data-intensive, 5G-enabled IoT applications: An overview, key issues and relevant architecture, Sensors 19(16) (2019) 16.

[6] W. Huang, J. Li, Using Agent Solutions and Visualization Techniques to Manage Cloud-based Education System, 18th IEEE International Conference on Dependable (2020) 375-379.

[7] C. Krintz, et.al., Cloud platform support for API governance, 2nd IEEE International Conference on Cloud Engineering (2014) 615-618.

[8] C. Teixeira, et.al., The building blocks of a PaaS, Journal of Network and Systems Management 22(1) (2014) 75-99.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 125–131
Received: 27 October 2022
Accepted: 20 November 2022

Comparative analysis of the availability of cinema websites, taking into account the principles of universal design

Analiza porównawcza dostępności serwisów internetowych kin z uwzględnieniem zasad projektowania uniwersalnego

Hanna Boguta*, Maria Skublewska-Paszkowska

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

Abstract

The subject of this study is to conduct a comparative analysis of cinema websites, taking into account the principles of universal design. The universal designing is a philosophy which central assumption is to create products, including graphic interfaces so that they are accessible to as many users as possible. An accessibility is a term that describes the extent to which users of different ages, with varying degrees of physical or mental fitness, can use it. Therefore, it is essential to adapt the interface to the largest possible audience when designing the interface. In the paper the following hypothesis has been defined: "The application of universal design principles increases the accessibility of websites". The "Multikino" website has been selected for the research, which does not require the universal design, and the website created for the purpose of this study that meets these requirements. The research methods used in the study include eye-tracking, checking compliance with WCAG (ang. Web Content Accessibility Guidelines) guidelines using the WAVE (ang. Web Accessibility Evaluation Tool) tool, and conducting an interface assessment survey. The conducted research has shown that the stated hypothesis is true.

Keywords: accessibility; universal design; eye tracking

Streszczenie

Celem niniejszej pracy jest przeprowadzenie analizy porównawczej serwisów internetowych kin z uwzględnieniem zasad projektowania uniwersalnego. Projektowanie uniwersalne jest filozofią, której głównym założeniem jest tworzenie produktów, w tym interfejsów graficznych w taki sposób, aby były one dostępne dla jak największej liczby użytkowników. Dostępność jest pojęciem określającym w jakim stopniu użytkownicy w różnym wieku, o różnym stopniu sprawności fizycznej czy umysłowej są w stanie z niego korzystać. Dlatego ważne jest, aby projektując interfejs dostosować go do jak największej liczby odbiorców. W pracy postawiono hipotezę: „Zastosowanie zasad projektowania uniwersalnego zwiększa dostępność stron internetowych". Na potrzeby badań wybrano serwis internetowy „Multikino", który nie wspiera zasad projektowania uniwersalnego, oraz utworzono własny serwis internetowy, który spełnia te wymagania. Metody badawcze zastosowane w pracy to badania okulograficzne, sprawdzenie poziomu zgodności z wytycznymi WCAG (ang. Web Content Accessibility Guidelines) przy pomocy narzędzia WAVE (ang. Web Accessibility Evaluation Tool), oraz przeprowadzenie ankiety oceny interfejsu. Przeprowadzone badania wykazały, że postawiona hipotezę jest prawdziwa.

Słowa kluczowe: dostępność; projektowanie uniwersalne; okulografia

*Corresponding author

Email address: hanna.boguta@pollub.edu.pl (H. Boguta)

1. Wstęp

Rozrywka jest ważnym elementem codziennego życia, dlatego ważne jest, alby każdy miał do niej dostęp niezależnie od wieku oraz stanu zdrowia. Pojęcie dostępności określa czy i jak szerokie grono odbiorców usługi może z niej korzystać. Dostępność dotyczy również stron internetowych, między innymi takich jak uwzględnione w badaniu serwisy internetowe kin. Pojęcie projektowania uniwersalnego odnosi się do dostępności oraz funkcjonalności utworzonego środowiska. Głównym jego założeniem jest, aby różne dobra i usługi były dostępne w takim samym stopniu dla zwykłych użytkowników jak również tych z dysfunkcjami oraz niepełnosprawnościami.

Wytyczne WCAG [1] określają w jakim stopniu strony internetowe spełniają wymagania dostępności. W 2016 roku Parlament Europejski wydał dyrektywę o dostępności cyfrowej, na mocy której strony internetowe i aplikacje mobilne podmiotów publicznych muszą spełniać szereg wymagań określonych w standardzie WCAG 2.0 [2]. Głównymi zasadami są postrzegalność, funkcjonalność, zrozumiałość oraz solidność.

Dostępność stron internetowych można także badać przy pomocy metody okulograficznej. Metoda ta przy pomocy specjalistycznego sprzętu zwanego okulografem pozwala na śledzenie ruchów gałek ocznych. Okulograf wysyła światło podczerwone i zbiera światło podczerwone odbite od powierzchni gałek ocznych, na podstawie czego rejestrowane są tak zwane fiksacje i sakady. Fiksacje to punkty, w których wzrok pozostaje na chwilę skupiony, natomiast sakady to ścieżki pomiędzy punktami fiksacji. Na podstawie zebranych danych

w odpowiednim oprogramowaniu wytwarza się tak zwane ścieżki fiksacji oraz mapy cieplne, dzięki którym można zobaczyć jak badana osoba skanuje wzrokiem, na przykład po interfejsie graficznym aplikacji internetowej.

2. Cel i zakres pracy

Celem pracy było zbadanie czy zastosowanie zasad projektowania uniwersalnego zwiększa dostępność stron internetowych. W tym celu zestawiono ze sobą dwie strony internetowe, jedną spełniającą wymagania projektowania uniwersalnego oraz drugą, która tych wymagań nie spełnia. Badania dotyczyły przeprowadzenia analizy porównawczej stron internetowych przy pomocy wytycznych WCAG 2.0, badań okulograficznych oraz ankietowych. W pracy postawiono hipotezę, iż zastosowanie zasad projektowania uniwersalnego zwiększa dostępność stron internetowych.

Zakres pracy obejmował przegląd literatury dotyczący badań okulograficznych stron internetowych oraz ich dostępności. Kolejnymi etapami pracy było zaprojektowanie eksperymentu badawczego oraz wybór metod badawczych pozwalających na zweryfikowanie postawionej hipotezy, a następnie wykonanie analizy otrzymanych wyników.

3. Przegląd literatury

Autorzy w artykule [3] poruszają kwestie użyteczności wybranych witryn internetowych używając do badań analizy okulograficznej. Twierdzą oni, że priorytetem dla twórców stron internetowych powinna być ich prostota w użytkowaniu, aby ograniczyć czas potrzebny na znalezienie określonych informacji, co przyczyni się do zredukowania frustracji przy korzystaniu z nich oraz zwiększy satysfakcję z użytkowania. Do badań użyto 5 różnych stron internetowych banków, na których 22 uczestników rozwiązało 3 zadania. Następnie zostali oni poproszeni o swobodne przeglądanie stron przez minutę. Celem tych badań była identyfikacja elementów i struktur strony oraz sposobu ich zaprojektowania tak, aby zwiększyć ich użyteczność. Badano elementy takie jak style projektowania strony, layouty oraz formaty stron tego samego typu. Ponadto wśród badanych osób przeprowadzono ankietę, w której oceniali oni jak trudne były zadania, które musieli wykonywać. Autorzy na podstawie przeprowadzonych badań okulograficznych oraz danych ankietowych wysnuli wniosek, że góra, środek oraz prawa strona witryny przykuwają najwięcej uwagi. Wartości liczbowe powinny być przedstawiane w postaci tabel lub wykresów, aby zwiększyć przejrzystość strony.

W artykule [4] przebadano technologie okulograficzne i ich zastosowania w wielu dziedzinach. Technologia pozwala na śledzenie ruchu gałek ocznych. Wykorzystuje się ją do zbierania informacji, gdzie skierowany jest wzrok w celu dalszej ich analizy. Autorzy opisują zasadę działania okulografii, rodzaje okulografów oraz dziedziny w jakich wykorzystuje się je do przeprowadzania badań. Są to między innymi badania naukowe i akademickie, badania rynku, neurobiologia i psychologia, badania medyczne, badania użyteczności, badania opakowań, badania komputerów i gier komputerowych, czynniki ludzkie i symulacja, okulistyka. Dla każdej z tych dziedzin opisano sposób wykorzystania technologii okulograficznej oraz jej znaczenie. Autorzy zauważają, że technologia ta ma szerokie zastosowanie i można ją wykorzystać do śledzenia aktywności użytkowników smartfonów, ponieważ urządzenia te posiadają kamery o dużej rozdzielczości.

W artykule [5] przeprowadzono analizę dostępności 59 stron internetowych portugalskich uczelni wyższych. W badaniu uwzględniono zarówno uczelnie techniczne jak i uniwersytety. Do badania wykorzystano 3 główne narzędzia: AChecker, WAVE oraz aXe. Na podstawie otrzymanych wyników autorzy artykułu porównują, który rodzaj uczelni średnio osiąga lepszą dostępność w 4 różnych aspektach takich jak postrzegalność, operatywność, zrozumiałość oraz solidność. Na przykład uniwersytety uzyskały lepsze wyniki dotyczące kontrastu kolorów na stronie w porównaniu do uczelni technicznych. Ponadto autorzy sugerują w jaki sposób można poprawić dostępność badanych stron, na przykład poprzez zwiększenie kontrastu elementów strony, zmianę rodzaju czcionki czy dodanie tekstu alternatywnego.

Autorzy artykułu [6] przedstawiają badania dostępności stron internetowych uniwersytetów w Indiach. Przeprowadzili oni analizę stron głównych 302 indyjskich uniwersytetów z użyciem wytycznych WCAG 2.0 przy pomocy narzędzi WAVE oraz AChecker, które automatycznie sprawdzają czy badana strona spełnia te wytyczne. Po otrzymaniu wyników badane uniwersytety podzielono pod względem dostępności ich serwisów internetowych na trzy grupy: serwisy o niskiej dostępności, serwisy o średniej odstępności oraz serwisy o wysokiej dostępności. Dodatkowo utworzono tabele porównujące liczbę uniwersytetów pod względem grupy, do której je przypisano. Ponad 70% z badanych uniwersytetów zakwalifikowano do trzeciej grupy co oznacza, że znaczna większość z nich nie spełnia wymogów dostępności. Utworzono również wykresy porównujące liczbę błędów jakie uzyskały badane strony z podziałem na utworzone uprzednio grupy. Bazując na wynikach badania autorzy proponują rozwiązania w postaci uwzględnienia na stronach wytycznych WCAG 2.0, aby zwiększyć ich dostępność. Ponadto zauważają oni, że każda z badanych stron ma zarówno swoje mocne jak i słabe strony.

W pracy [7] autorzy opisują przeprowadzone badania porównawcze dostępności serwisów internetowych 348 uczelni wyższych w Ameryce Łacińskiej. Poruszają oni problematykę dostępności serwisów w odniesieniu do coraz większej popularności tych serwisów. Zauważają oni, że w dużym stopniu pierwsze wrażenie osób dotyczące instytucji opiera się na ich oficjalnych stronach internetowych. Dzieje się tak ponieważ, to na oficjalnych stronach w pierwszej kolejności wyszukiwane są informacje. Dlatego istotne jest zadbanie o to, aby strony te spełniały wymagania dostępności oraz były przejrzyste, intuicyjne i proste w obsłudze, także dla osób z niepełnosprawnościami. Badania zostały

przeprowadzone w oparciu o wytyczne WCAG 2.0 przy użyciu narzędzia WAVE. Badano zgodność stron internetowych z 12 wytycznymi opisanych według zasad P. O. U. R. (ang. Percivable, Operable, Understandable, Robust), które dotyczą postrzegalności, operatywności, zrozumiałości oraz solidności strony. Do badań użyto losową próbkę spośród wszystkich uczelni wyższych w Ameryce Łacińskiej. Na podstawie wyników badań autorzy utworzyli ranking uczelni w zależności od uzyskanej liczby błędów, ranking państw w oparciu o procentową liczbę błędów oraz mapę państw Ameryki Łacińskiej, na której zaznaczono wyniki uzyskane przez dane państwo. Z badań wynika, że żadna z badanych uczelni nie uzyskała akceptowalnego poziomu dostępności oficjalnej strony internetowej. Porównano również ogólny ranking uczelni z rankingiem dostępności stron internetowych, z czego wywnioskowano, że nie każda uczelnia umieszczona wysoko w rankingu posiada adekwatnie wysoki wynik w badaniu dostępności oficjalnej strony internetowej.

W pracy [8] opisano problematykę stron o tematyce e-commerce w odniesieniu do ich dużej ilości na rynku oraz świadczenia podobnych usług i produktów. Jakość strony internetowej oraz doświadczenia z użytkowania pełnią bardzo istotną rolę w odnoszeniu przez nią sukcesu. Metoda eQual jest metodą ewaluacji stron internetowych używającą 22 kryteria. Podzielone są one na grupy takie jak użyteczność, jakość informacji oraz jakość interakcji z serwisem. Autorzy w swoich poprzednich publikacjach proponowali metodę PEQUAL, która była rozszerzeniem dla metody eQual, biorąc również pod uwagę różne aspekty oraz preferencje modelowania z MCDA (ang. Multi-Criteria Decision Analysis) oraz dodano analizę okulograficzną do badań. Celem autorów było rozwinięcie wcześniej istniejącej metody MCDA przez dodanie połączonych wyników ankietowych oraz wzięcia pod uwagę postrzegalności danych na stronach. Zostało to zrealizowane poprzez dodanie 6 nowych kryteriów oceny. Badaniu poddano 10 najbardziej popularnych serwisów e-commerce. Na podstawie wyników badań stwierdzono, że badania ankietowe pozwalają badaczom dowiedzieć się o subiektywnej ocenie użytkowników. Analiza okulograficzna dostarcza mierzalnych danych oraz rozwinięta przez badania ankietowe może utworzyć na przykład rankingi jakości stron. Autorzy stwierdzają, że w badaniach warto łączyć ze sobą kilka metod badawczych, ponieważ umożliwia to szerszą analizę problematyki.

W pracy [9] przebadano 50 aplikacji internetowych pod kątem ich dostępności. Badanie skupiało się na średnich wartościach wynikowych ewaluacji z użyciem wytycznych WCAG 2.0. Wartościami wynikowymi są fail, pass i warning, które kolejno oznaczają, że dany element na stornie jest niezgodny z wytycznymi, zgodny z wytycznymi oraz posiada ostrzeżenie o błędach. Obliczono średnią liczbę elementów występujących na pojedynczej stronie danej aplikacji internetowej, która wyniosła 1010 elementów. Wynikiem badań są średnie wartości występujących na badanych stronach wartości wynikowych. Przedstawione zostały one w postaci liczbowej oraz procentowej. Średnio 28% elementów na stronie przeszło pomyślnie ewaluację, 12% nie przeszło ewaluacji, natomiast 70% otrzymało ostrzeżenia o błędach. Autorzy zauważają, że wyższa liczba elementów, które przeszły pomyślnie ewaluację, w porównaniu z elementami, które nie przeszły ewaluacji może wynikać z użycia nowoczesnych narzędzi, które podczas tworzenia stron internetowych automatycznie generują fragmenty kodu zgodne z wytycznymi WCAG 2.0. Autorzy we wnioskach stwierdzają, że obecnie strony internetowe są coraz bardziej rozbudowane oraz dynamiczne, co powoduje powstawanie coraz większej liczby błędów w kontekście dostępności, więc ważne jest aby korzystać z narzędzi do ewaluacji stron, które działają dynamicznie i powalają wykryć więcej błędów występujących na stronach internetowych.

Autorzy pracy [10] przedstawiają opracowany algorytm eMine, który na podstawie badań okulograficznych generuje najbardziej powszechną ścieżkę powstałą podczas wykonywania przez uczestników badania określonych zadań na stronach internetowych. W badaniu wzięło udział 40 uczestników i jako materiał badawczy użyto sześć stron internetowych. Początkowo sprawdzono, czy algorytm wykrywa takie ścieżki. Badane strony podzielono na sekcje i oznaczono je kolejnymi literami alfabetu, następnie uczestnicy wykonywali zadania dotyczące odnajdywania określonych informacji czy elementów na stronie. Na podstawie danych z okulografu utworzono sekwencje w jakich uczestnicy badania poruszali się po wydzielonych sekcjach. W kolejnym kroku użyto algorytmy eMine do wybrania spośród nich takich sekwencji, które były najbardziej powszechne. Badania następnie rozszerzono o uwzględnienie między innymi płci, znajomości danej strony oraz rodzaju wykonywanego zadania. Pozwoliło to wykazać, że te czynniki mogą mieć wpływ na wynik końcowy. Powstały algorytm może zostać wykorzystany podczas tworzenia stron internetowych w celu polepszenia doświadczeń użytkowników oraz zrozumienia jak korzystają ze stron internetowych.

4. Metoda badawcza

Głównym założeniem badań było przeprowadzenie analizy porównawczej dwóch serwisów internetowych kin, w celu zbadania czy zastosowanie zasad projektowania uniwersalnego zwiększa poziom dostępności strony internetowej. W tym celu wykorzystano metody badawcze takie jak: analiza WCAG 2.0, badania okulograficzne oraz ankieta oceny użytkowników. Następnie z uzyskanych danych okulograficznych obliczono średni czas w jakim użytkownik odnajdował na stronie dany element. Z danych ankietowych obliczono jak średnio w odczuciu badanych osób strony spełniają zasady projektowania uniwersalnego.

4.1. Obiekt badań

Na potrzeby badań wybrano serwis internetowy jednej z najpopularniejszych w Polsce sieci kin Multikino [11],

który nie spełniał wymagań projektowania uniwersalnego. Ponadto utworzono własną stronę internetową o podobnej tematyce, gdzie zastosowano zasady projektowania uniwersalnego, w celu przeprowadzenia analizy porównawczej obu stron internetowych.

4.2. Grupa badawcza

Grupę badawczą stanowiło 20 studentów studiów magisterskich. Byli to studenci kierunku Informatyka na Politechnice Lubelskiej.

4.3. Stanowisko badawcze

Stanowisko badawcze składało się ze laptopa oraz zewnętrznego okulografu Gazepoint GP3 HD [12], ustawionego tuż pod ekranem komputera. Na komputerze zainstalowano dedykowane oprogramowanie Gazepoint oraz iMotions [13] potrzebne do obsługi okulografu.

4.4. Przebieg badań

W pierwszej kolejności badane strony poddano weryfikacji dostępności z wykorzystaniem wytycznych WCAG 2.0. Użyto do tego narzędzia WAVE [14], które po podaniu adresu strony automatycznie sprawdza zgodność strony z wytycznymi oraz wyświetla wszelkie błędy występujące na stronie.

Kolejnym etapem badań było przeprowadzenie badań okulograficznych. Do badań wykorzystano okulograf Gazepoint GP3 HD podłączony do laptopa, na którym zainstalowano oprogramowanie Gazepoint Control oraz iMotions 9.2. Przed przystąpieniem do właściwych badań uczestnicy przechodzili proces kalibracji, który służył do skorygowania różnicy między rzeczywistymi punktami patrzenia, a tymi wskazywanymi przez urządzenie. Następnie badanym wyświetlano kolejno polecenia (Tabela 1) jakie elementy interfejsu mają odszukać na stronie, a następnie wyświetlano zrzuty ekranu aplikacji na których znajdują się dane elementy.

Trzecim krokiem badań było przeprowadzenie ankiety do oceny jakości interfejsu aplikacji. Przygotowano scenariusz badawczy, według którego badani zapoznawali się z wybranym serwisem internetowym kina oraz utworzoną na potrzeby badania aplikacją. Scenariusz badawczy wyglądał następująco:

1. Użytkownik wchodzi na stronę
2. Użytkownik wybiera opcję "Zaloguj się"
3. Użytkownik przechodzi do strony Rejestracji
4. Użytkownik tworzy konto.
5. Użytkownik loguje się na utworzone konto.
6. Użytkownik przechodzi na stronę z repertuarem
7. Użytkownik wyświetla opis pierwszego wyświetlającego się filmu.
8. Użytkownik wybiera godzinę seansu i przechodzi do kupna biletu.
9. Użytkownik rezerwuje bilet na wybrany.
10. Użytkownik wylogowuje się.

Następnie w ankiecie przygotowanej w oparciu o ankietę LUT (ang. Lublin University of Technology) oceniali oni w jakim stopniu (w ich subiektywnej ocenie) badane strony spełniały kryteria projektowania uniwersalnego. Wybrano 10 zagadnień najistotniejszych dla wykonywanych badań. W Tabeli 2 znajdują się pytania na jakie musieli odpowiedzieć badani.

Ocena powyższych zagadnień odbyła się według 5-stopniowej skali. W Tabeli 3 przedstawiono znaczenie każdego ze stopni skali.

Tabela 1: Polecenia dla badanych osób

Numer badania	Polecenie
1	Znajdź element, za pomocą którego przejdziesz do formularza rejestracji.
2	Znajdź przycisk, za pomocą którego zalogujesz się.
3	Znajdź pole, do którego należy wpisać adres e-mail.
4	Znajdź element, za pomocą którego wyświetlisz regulamin.
5	Znajdź element, za pomocą którego wyświetlisz opis filmu.
6	Znajdź element, za pomocą którego wyszukasz film.
7	Znajdź element, gdzie podana jest data premiery filmu.
8	Znajdź element, w którym widnieje informacja o godzinach seansów.
9	Znajdź elementy wskazujące na zarezerwowane już miejsca w kinie.
10	Znajdź element z informacją o godzinie, na którą rezerwujesz bilet.

Tabela 2: Zagadnienia oceniane w ankiecie LUT

Numer pytania	Treść pytania
1	Czy dostęp do wszystkich funkcji aplikacji jest łatwy i intuicyjny?
2	Czy struktura informacji jest przemyślana?
3	Czy struktura informacji jest zrozumiała dla użytkownika?
4	Czy layout jest czytelny?
5	Czy układ jest graficznie spójny?
6	6. Czy kontrast pomiędzy tekstem a tłem jest odpowiedni?
7	7. Czy dobór barw umożliwia skorzystanie z aplikacji przez osoby z zaburzeniami widzenia barw?
8	8. Czy zwykły użytkownik nie ma trudności z wprowadzeniem danych do formularza?
9	Czy formularze posiadają elementy walidujące wprowadzone dane?
10	Czy używane w aplikacji nazewnictwo jest zrozumiałe?

Tabela 3: Skala oceny zagadnień w ankiecie

Stopień	Znaczenie
1	Wystąpiły krytyczne problemy dotyczące użyteczności, uniemożliwiające korzystanie z aplikacji/strony bądź zniechęcające do korzystania z niej.
2	Napotkano poważne problemy dotyczące użyteczności mogące uniemożliwić większości użytkownikom realizacje zadań.
3	Wystąpiły drobne problemy związane z użytecznością, które pojedynczo nie stanowią utrudnienia dla większości użytkowników, jednak ich nagromadzenie może wpłynąć na jakość pracy użytkownika.
4	Zidentyfikowane pojedyncze drobne problemy związane z użytecznością mogące obniżyć jakość pracy z aplikacją
5	Nie stwierdzono problemów związanych z użytecznością ani mających wpływ na jakość pracy.

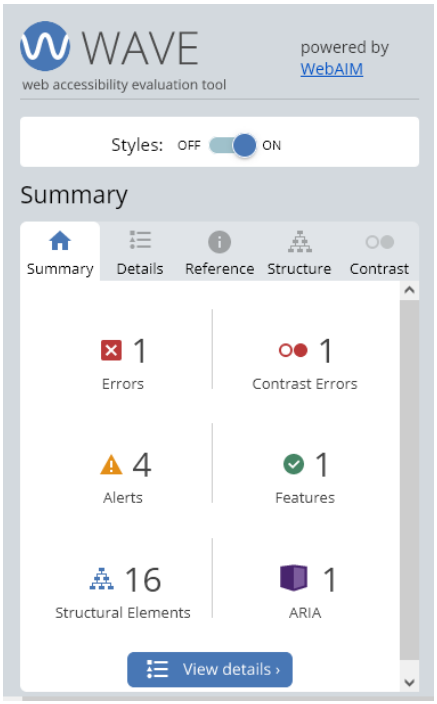
5. Wyniki

Po przeprowadzeniu badań dokonano analizy danych zebranych na poszczególnych etapach. Analiza WCAG

2.0 przy użyciu narzędzia WAVE, to proste sprawdzenie w jakim stopniu badane serwisy spełniają wymagania WCAG. Badania okulograficzne umożliwiły analizę ilościową na podstawie czasu wykonania zadań oraz liczby fiksacji. Ponadto umożliwiły analizę jakościową przy użyciu map cieplnych oraz ścieżek skanowania. Badania ankietowe pozwoliły sprawdzić, jak subiektywnie badane osoby oceniały serwisy.

5.1. Analiza WCAG 2.0

W wyniku analizy WCAG 2.0 przeprowadzonej w narzędziu WAVE dla utworzonej aplikacji otrzymano wynik przedstawiony na Rysunku 1, natomiast dla serwisu Multikino na Rysunku 2.

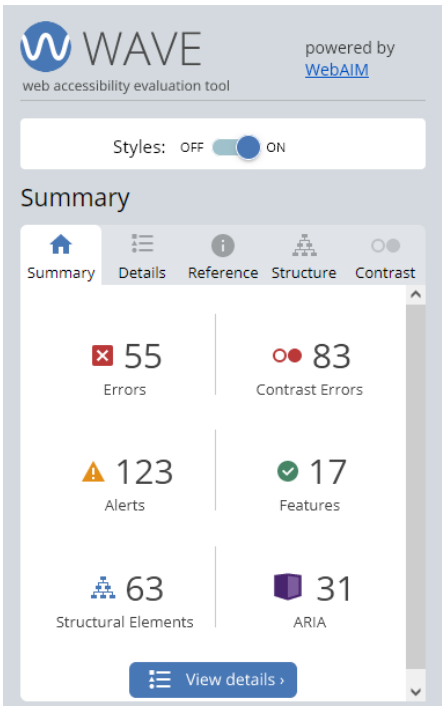


Rysunek 1: Wyniki analizy WCAG 2.0 dla aplikacji spełniającej wymagania.

Wyniki analizy wykazały znaczną różnicę pomiędzy badanymi serwisami. Utworzona aplikacja uzyskała pojedyncze błędy ogólne i kontrastu oraz 4 ostrzeżenia, natomiast aplikacja Multikino, 55 ogólnych błędów, 83 błędy kontrastu oraz 123 ostrzeżenia. Wynik taki może być spowodowany różnicą w ilości elementów graficznych na stronach.

5.2. Badania okulograficzne

Z badań okulograficznych uzyskano czasy w jakich badani wykonywali polecenia polegające na odnalezieniu danego elementu interfejsu na stronach oraz liczbę fiksacji od momentu wyświetlenia widoku do momentu odnalezienia szukanego elementu. Dla każdego elementu obliczono średni czas w jakim badani wykonywali polecenia oraz średnią liczbę fiksacji. Średnie czasy wykonania zadań przedstawiono w Tabeli 4, a średnią liczbę fiksacji w Tabeli 5.



Rysunek 2: Wyniki analizy WCAG 2.0 dla aplikacji niespełniającej wymagań.

Tabela 4: Średni czas wykonania zadania

Numer zadania	Aplikacja spełniająca wymagania (s)	Aplikacja nie spełniająca wymagań (s)
1	1,59	4,78
2	1,27	5,25
3	1,50	4,49
4	3,04	6,29
5	2,18	12,58
6	1,78	9,03
7	2,34	6,83
8	1,88	7,64
9	2,35	7,64
10	2,82	5,69

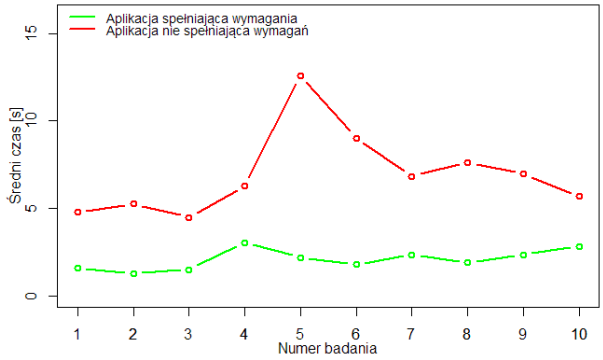
Tabela 5: Średnia liczba fiksacji dla zadań

Numer zadania	Aplikacja spełniająca wymagania	Aplikacja nie spełniająca wymagań
1	6,10	15,55
2	4,60	16,70
3	5,60	14,40
4	11,20	22,20
5	6,80	42,00
6	5,42	29,25
7	8,60	22,45
8	6,75	27,10
9	9,10	23,85
10	9,90	18,30

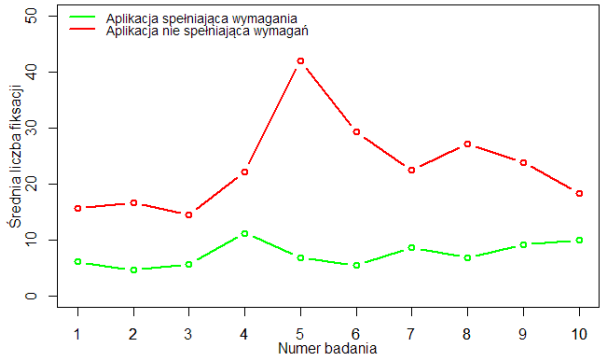
Dodatkowo dla lepszej wizualizacji wyników utworzono wykresy (Rysunek 3 i 4), które przedstawiają średni czas wykonania każdego z zadań oraz średnią liczbę fiksacji.

Na Rysunkach 5-8 przedstawiono przykładowe mapy cieplne oraz ścieżki fiksacji dla widoku logowania, obrazujące w jaki sposób badani wykonywali zadanie

polegające na odnalezieniu przycisku rejestracji. Mapa cieplna przedstawia, gdzie na badanym widoku przez najdłuższy czas badane osoby skupiały wzrok. Ścieżki fiksacji pokazują w jaki sposób jedna z badanych osób odnajdowała dany element na stronie.

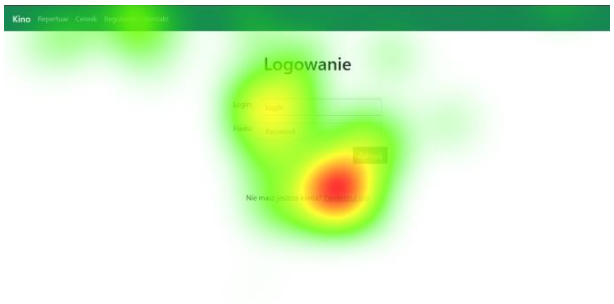


Rysunek 3: Średni czas wykonania zadania.



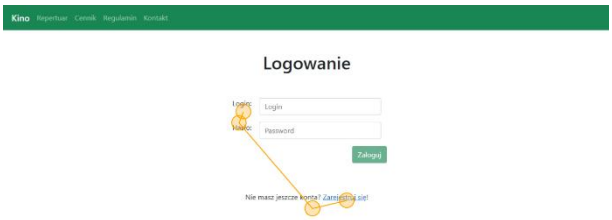
Rysunek 4: Średnia liczba fiksacji dla zadań.

Na utworzonych wykresach wyraźnie widać występującą między nimi korelację. Zależność tą najwyraźniej widać w przypadku badania numer 5. Spowodowane jest to faktem, iż większy czas potrzebny na odnalezienie elementu interfejsu to także większa liczba punktów, w których skupia się wzrok.

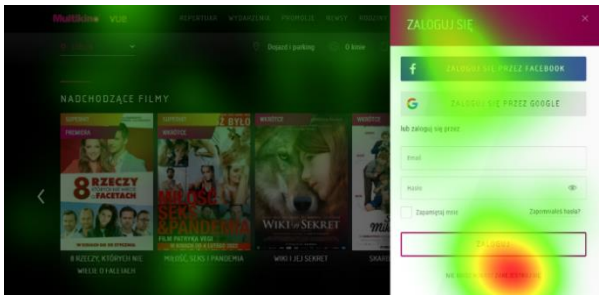


Rysunek 5: Mapa cieplna dla utworzonej aplikacji.

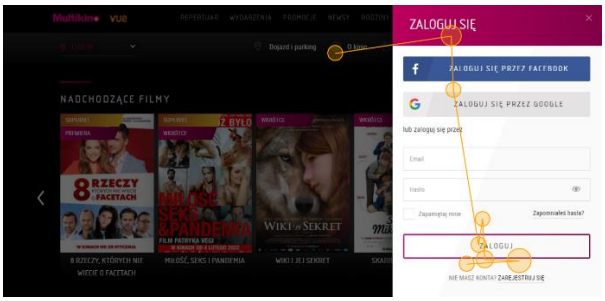
Zarówno na ścieżkach fiksacji oraz mapach cieplnych można zauważyć, iż w przypadku utworzonej aplikacji badanym osobom łatwiej było odnaleźć szukany element interfejsu. Na mapach cieplnych świadczy o tym różnica w rozproszeniu obszarów oznaczonych kolorem zielonym, natomiast na ścieżkach fiksacji różnica w ilości punktów skupienia wzroku (fiksacji).



Rysunek 6: Ścieżka fiksacji dla utworzonej aplikacji.



Rysunek 7: Mapa cieplna dla aplikacji Multikino.



Rysunek 8: Ścieżka fiksacji dla aplikacji Multikino.

5.3. Badania ankietowe

Po przeprowadzeniu badań ankietowych zebrano odpowiedzi badanych osób i obliczono jak średnio oceniali każdą z badanych stron. W Tabeli 6 przedstawiono oceny badanych oraz średnią ogólną ocenę dla każdej strony.

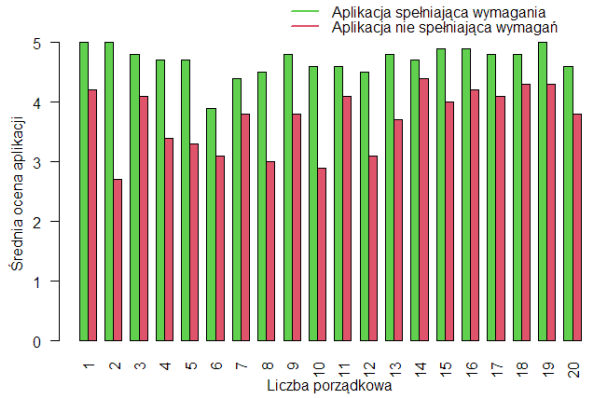
Tabela 6: Średnie oceny aplikacji z badań ankietowych

Liczba porządkowa	Aplikacja spełniająca wymagania	Aplikacja nie spełniająca wymagań
1	5,0	4,2
2	5,0	2,7
3	4,8	4,1
4	4,7	3,4
5	4,7	3,3
6	3,9	3,1
7	4,4	3,8
8	4,5	3,0
9	4,8	3,8
10	4,6	2,9
11	4,6	4,1
12	4,5	3,1
13	4,8	3,7
14	4,7	4,4
15	4,9	4,0
16	4,9	4,2

17	4,8	4,1
18	4,8	4,3
19	5,0	4,3
20	4,6	3,8
Średnia	**4,7**	**3,7**

Na podstawie otrzymanych wyników można stwierdzić, że badane osoby oceniali utworzoną aplikację o 1 punkt wyżej w 5-stopniowej skali, w porównaniu do serwisu Multikino. Oznacza to, że w ich subiektywnej ocenie utworzona aplikacja lepiej spełniała wymagania projektowania uniwersalnego.

Na Rysunku 9 przedstawiono również wykres przedstawiający jak badani oceniali badane aplikacje.



Rysunek 9: Średnie oceny aplikacji z badań ankietowych.

6. Wnioski

Na podstawie przeprowadzonych badań wykazano, że postawiona w pracy teza jest prawdziwa. Oznacza to, że zastosowanie zasad projektowania uniwersalnego zwiększa dostępność stron internetowych.

Za pomocą wytycznych WCAG 2.0 sprawdzono poziom dostępności badanych stron. Badania te wykazały, że aplikacja utworzona na potrzeby badań lepiej spełnia wytyczne, a tym samym jest bardziej dostępna. Badania okulograficzne wykazały, że w przypadku utworzonej aplikacji badane osoby znacznie szybciej odnajdowały dane elementy interfejsu w porównaniu z aplikacją Multikino. Ankieta oceny użytkowników również wykazała, iż w subiektywnej ocenie badanych, aplikacja Multikino zawierała znacznie więcej błędów, które miały wpływ na ogólne doświadczenia użytkowników podczas korzystania z aplikacji.

Dostępność aplikacji jest bardzo ważnym aspektem, który należy wziąć pod uwagę podczas jej tworzenia. Zastosowanie zasad projektowania uniwersalnego pozwala nie tylko na to, aby jak najszersze grono użytkowników mogło korzystać z aplikacji, ale również w przypadku serwisów internetowych kin niesie za sobą korzyści finansowe.

Literatura

[1] WCAG Overview, https://www.w3.org/WAI/standards-guidelines/wcag/, [5.10.2022].

[2] Web Content Accessibility guidelines, https://www.w3.org/TR/WCAG20/, [5.10.2022].

[3] M. Tichindelean, M. T. Tichindelean, I. Cetina, G. Orzan, A comparative eye tracking study of usability – Towards sustainable web design, Sustainability 13(18) (2021) 10415, https://doi.org/10.3390/su131810415.

[4] P. A. Punde, M. E. Jadhav, R. R. Manza, A study of eye tracking technology and its applications, International Conference on Intelligent Systems and Information Management 1 (2017) 86-90, https://www.doi.org/10.1109/ICISIM.2017.8122153.

[5] A. Isamail, K. S. Kuppusamy, S. Paiva, Accessibility analysis of higher education institution websites of Portugal, Universal Access in the Information 19 (2020) 685-700, https://doi.org/10.1007/s10209-019-00653-2.

[6] A. Ismail, K. S. Kuppusamy, Accessibility of Indian universities' homepages: An exploratory study, Journal of King Saud University - Computer and Information Sciences 30 (2018) 268-278, https://doi.org/10.1016/j.jksuci.2016.06.006.

[7] P. Acosta-Vargas, T. Acosta, S. Julan-Mora, Challenges to Assess Accessibility in Higher Education Websites: A Comparative Study of Latin America Universities, Institute of Electrical and Electronics Engineers 6 (2018) 36500-36508, https://doi.org/10.1109/ACCESS.2018.2848978.

[8] P. Ziemba, J. Wątróbski, A. Karczmarczyk, J. Jankowski, W. Wolski, Integrated Approach to e-Commerce Websites Evaluation with the Use of Surveys and Eye Tracking Based Experiments, Federated Conference on Computer Science and Information Systems 11 (2017) 1019-1030, http://dx.doi.org/10.15439/2017F320.

[9] N. Fernandes, D. Costa, C. Duarte, L. Carrico, Evaluating the Accessibility of Web Applications, Procedia Computer Science 14 (2012) 28-35, https://doi.org/10.1016/j.procs.2012.10.004.

[10] S. Eraslan, Y. Yesilada, Patterns in eyetrakcking scanpaths and the affecting factors, Journal of Web Engineering 14 (2015) 363-385, https://journals.riverpublishers.com/index.php/JWE/article/view/3841, [5.10.2022].

[11] Multikino, https://multikino.pl/, [5.10.2022].

[12] Okulograf Gazepoint GP3 HD, https://www.gazept.com/product/gp3hd/, [5.10.2022].

[13] Oprogramowanie iMotions, https://imotions.com/, [5.10.2022].

[14] Narzędzie WAVE, https://wave.webaim.org/, [5.10.2022].

[15] M. Miłosz, Ergonomia systemów informatycznych, Lublin, 2014, http://bc.pollub.pl/dlibra/docmetadata?id=8718&from=&dirids=1&ver_id=&lp=1&QI=#description, [5.10.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 132–137
Received: 13 November 2022
Accepted: 7 December 2022

Comparison of shallow and deep learning methods of ECG signals classification for arrhythmia detection

Porównanie metod płytkiego i głębokiego uczenia do klasyfikacji sygnałów EKG zastosowanych do wykrywania arytmii

Dodon Turianto Nugrahadi*, Rudy Herteno, Dwi Kartini, Muhammad Haekal, Mohammad Reza Faisal

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

Abstract

The research aims to compare the classification performance of arrhythmia classification from the ECG signal dataset from the Massachusetts Institute of Technology–Beth Israel Hospital (MIT-BIH) database. This study uses shallow learning methods: Support Vector Machine, Naïve Bayes, and Random Forest. 1D Convolutional Neural Network (1D CNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) are deep learning methods that are used for the study. The models are tested on a dataset with 140 samples that are grouped into four class labels. Each sample has 2160 features. Those models are tested for classification performance. This research shows Random Forest and 1D CNN have the best performance.

Keywords: ECG signals; arrhythmia classification; shallow learning; deep learning

Streszczenie

Badanie ma na celu porównanie wydajności klasyfikacji arytmii na podstawie zestawu danych sygnału EKG z bazy danych Massachusetts Institute of Technology – Beth Israel Hospital (MIT-BIH). W pracy zastosowano następujące metody: Support Vector Machine, Naïve Bayes i Random Forest. Ponadto wykorzystano następujące metody głębokiego uczenia: 1D Convolutional Neural Network (1D CNN), Long Short Term Memory (LSTM) oraz Gated Recurrent Unit (GRU). Modele zostały przetestowane na zbiorze danych zawierającym 140 próbek pogrupowanych w cztery etykiety klas. Każda próbka zawierała 2160 cech. Przeprowadzone testy wydajności klasyfikacji wskazały, że Random Forest i 1D CNN wykazują najwyższą wydajność.

Słowa kluczowe: sygnałów EKG; klasyfikacja arytmii; płytkie uczenie; głębokie uczenie

*Corresponding author

Email address: dodonturianto@ulm.ac.id (D. T. Nugrahadi)

1. Introduction

Nowadays, signal classification methods play an essential role in a wide variety of life. Examples of the implementation of this method are for the classification of music genres and moods [1], the classification of machine conditions [2], and the classification of signals in the health field [3], [4].

One of the signal classification cases in the health sector is the electrocardiogram (ECG) signal classification. ECG is a signal that describes the electrical activity carried out by the heart and is very important in diagnosing heart rhythm disturbances caused by changes in electrical impulses in the heart. This disorder is also known as arrhythmia [4]. One way to help detect heart rhythm disturbances early on automatically can be done with the help of deep learning. This paper compares deep learning methods of arrhythmia classification using ECG signals.

Currently, many studies have been carried out that utilize machine learning or artificial intelligence to recognize ECG patterns and then automatically predict heart condition categories. Machine learning techniques based on shallow learning for heart disease classification are used, such as Random Forest [5], Support Vector Machine (SVM) [6], K-Nearest Neighbors (KNN) [7], Naïve Bayes [8]. The deep learning technique used for the classification of cases of heart disease is Convolutional Neural Network (CNN) [6], Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) [4][6].

In research on the classification of heart disease, there are two treatments for ECG data. The first treatment using raw ECG data was done by [9]. In this study, the selected raw ECG data were classified using the shallow learning type classification method, namely SVM. The second treatment is to analyze the raw ECG signal by transforming the signal. One of the transformation techniques used is the wavelet transform [10]. The raw ECG signal usually consists of different frequency components and even noise, making it difficult for classification algorithms to learn to determine patterns. So the wavelet transform is used to eliminate these things. However, humans recognize arrhythmic patterns from visualization of the raw ECG signal. In addition, the classification of raw ECG signals can lighten the computer's work because it does not perform signal transformation calculations.

Based on this explanation, a comparative study was conducted on classifying raw ECG signals with shallow and deep learning algorithms. The goal is to determine

which shallow and deep learning algorithms can perform better classification. In addition, this study also aims to determine which classification performance is better between algorithms based on shallow learning and deep learning.

This report is divided into five section: 1) Introduction, 2) Dataset and method section explains the dataset and classification algorithms that are used, 3) Research implementation section explains the steps in implementation of this study, 4) Result, and the last section is 5) Conclusions.

2. Dataset and methods

2.1. Dataset

The raw ECG signal dataset used in the study is from research [9]. With details as can be seen in Tabel 1. The source of the research data came from the arrhythmia database of the Massachusetts Institute of Technology–Beth Israel Hospital (MIT-BIH). Six seconds of cardiac records resulted in 2160 data points.

Arrhythmias cause the heart to beat faster, slower, or irregularly. The condition causes symptoms such as feeling tired and pain in the chest. To detect arrhythmias, doctors use a heart record or electrocardiogram. An electrocardiogram (EKG) is a recording of heart activity obtained by attaching an electrode to the skin to capture the electric current generated by the heart. A series of heart activities recorded by the EKG can be used as an indicator of a heart rhythm disorder, which doctors or nurses can use to take appropriate action.

Table 1: Raw ECG dataset

No	Class Label	Number of features	Number of samples
1	Normal	2160	35
2	Atrial Fibrillation	2160	35
3	PVC Bigeminy	2160	35
4	Ventricular Tachycardia	2160	35
Total			140

A normal sample is regular heart rate ECG data, also known as normal sinus rhythm. A normal sample can be visualised in Figure 1, where the heart is beating in a regular sinus rhythm between 60 - 100 beats per minute (specifically 82 bpm).

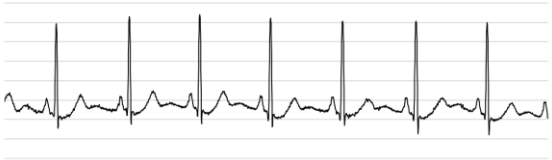


Figure 1: Normal.

Atrial Fibrillation is a type of arrhythmia. In this condition, a rapid heartbeat often causes poor blood flow. The heart's upper chambers (atria) beat out of coordination with the lower chambers (ventricles). These conditions may have no symptoms, but when they do they include palpitations, shortness of breath, and fatigue. Atrial Fibrillation visualization is shown in Figure 2.

Extra heartbeats known as premature ventricular contractions (PVCs) begin in one of the heart's two lower pumping chambers (ventricles). The regular heart rhythm is disrupted by this extra beat, which can sometimes make the chest feel like it is pounding or jumping up and down. This irregular heartbeat is also known as PVC Bigeminy, which is visualized in Figure 3.

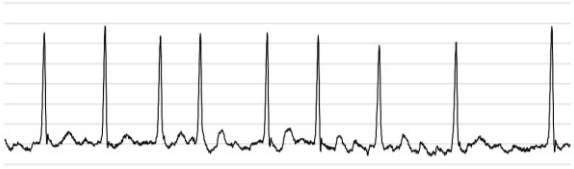


Figure 2: Atrial Fibrillation.

Ventricular Tachycardia is a condition in which the heart's lower chambers (ventricles) beat very fast. This irregular heartbeat ECG is shown in Figure 4.

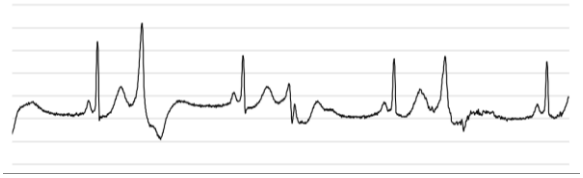


Figure 3: PVC Bigeminy.

This condition occurs because of a problem with the heart's electrical impulses. This condition can develop as a complication of a heart attack, or it can occur in people with certain conditions, such as valvular heart disease.

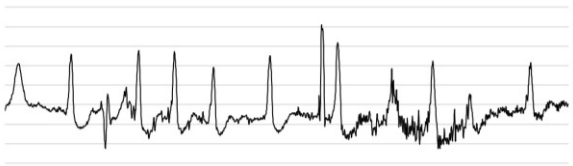


Figure 4: Ventricular Tachycardia.

2.2. Methods

There are two type learning that are used in this study: shallow learning and deep learning.

Shallow learning is a type of machine learning that learns from data with predetermined features extracted from features. In this study, shallow learning algorithms that are used are Support Vector Machine, Naïve Bayes, and Random Forest.

Vapnik introduced the Support Vector Machine (SVM) in 1992 as a machine learning method that works on the Structural Risk Minimization (SRM) principle, which aims to find a hyperplane to separate two classes in the input space. This method uses hypotheses as linear functions in feature space with high dimensions by implementing a learning bias derived from statistical learning theory. The level of accuracy in the model that the switching process will produce with

SVM depends on the kernel functions and parameters. [6].

The Naive Bayes algorithm is a classifier that uses the probabilistic and statistical methods proposed by the British scientist Thomas Bayes to predict future potentials based on past experiences. The main feature of Naïve Bayes is a solid assumption for the independence of each condition of an event. Calculations for comparison of new cases with old cases are indexed based on input parameters with mathematical calculations using probabilities that occur for cases considered similar (Bayesian model). The advantage of Naïve Bayes is that it does not require a lot of data to estimate the parameters needed for classification [8].

Random Forest has many decision trees, which can be hundreds or even thousands of trees, which predict individual classes. The Random Forest aims to form a single representative decision from multiple decision trees. The majority vote of the entire tree can be defined as the result of class predictions [5]. The structure of the Random Forest can be seen in Figure 5

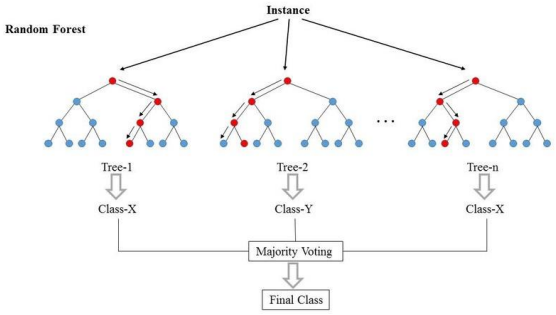


Figure 5: Structure of Random Forest [11].

In contrast to shallow learning, deep learning gets features from data from the feature extraction process, which is carried out automatically without human intervention. Deep learning algorithms automatically learn their features and weights, allowing deep learning to use the best features to get the best classification performance. Deep learning algorithms that are used in this study are 1D Convolutional Neural Network (1D CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

Wiesel and Hubel first carried out a Convolutional Neural Network (CNN) regarding the virtual cortex in the sense of sight. CNN is an architecture that can be trained and consists of several stages, inputs and outputs. CNN trains and tests each input data through a series of processes, namely the convolutional layer, followed by pooling to extract features from successive input data. The next step is the pooling process. In this process, the data is flattened and then entered into the fully connected-layer process to complete the classification process [12]. Usually, CNN is used to process input data in images as two-dimension data. CNN is also known as 2D CNN. Figure 6 shows an example of a 2D CNN architecture for 2D data.

In 2D CNN, the convolutional layer will produce two-dimensional data. CNN can also be used for one-dimensional data cases, for example, signals including ECG. For the case of one-dimensional data, CNN is known as 1D CNN. In 1D CNN, the convolutional layer will run in one dimension and produce one-dimensional data [14].

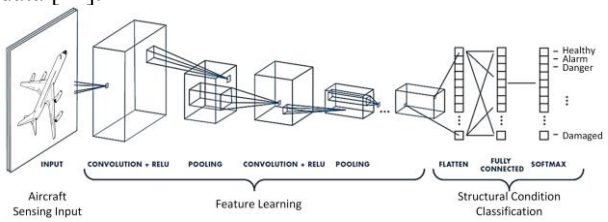


Figure 6: 2D CNN architecture [13].

Examples of 1D architecture for signal cases can be seen in Figure 7.

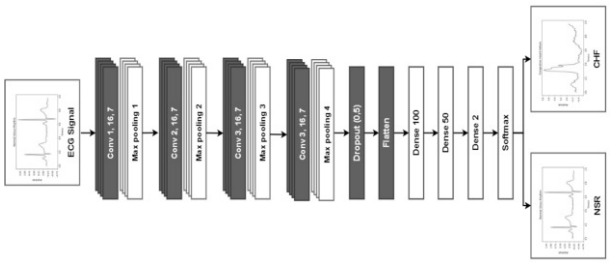


Figure 7: 1D CNN architecture [14].

Recurrent Neural Network (RNN) is an artificial neural network for processing sequential data such as speech recognition and language modelling. Long Short-Term Memory (LSTM) is a variation of RNN that was created to avoid the problem of long-term dependency on RNN [4]. LSTM can remember long-term information, and LSTM also has an iterative processing model like RNN. In the LSTM, there is a path that connects the old context to the new context or what is also known as the cell state, memory cell or memory path. The path makes the values in the old context easily linked to the new context if needed with minor modifications. The LSTM can delete or add information to memory lanes governed by the sigmoid function [6].

LSTM has four gates: forget gate, input gate, cell gate, and output gate. Forget gate is a gate that decides whether information should be removed or not from processing. The input Gate is a gate that decides to determine whether an input will be added to the cell gate memory. The cell gate is a gate that functions as a memory for a layer; this gate's ability is to remember long-term information. The output gate is a gate whose function is to decide what will be produced based on the input and cell gate.

Gated Recurrent Unit (GRU) is an architecture created in 2014 by Kyunghun Cho. GRU provides faster convergence, and the results can be compared with LSTM. GRU also has a smaller error value compared to LSTM. The purpose of the GRU is to make each recurrent unit so that it can capture every relationship (de-

pendency) in different time scales carried out adaptively. The GRU has a component that regulates the flow of information called the gate, and the GRU here has two gates, namely, the reset gate and the update gate [15].

3. Research implementation

The steps in the implementation of the research can be seen in Figure 8.

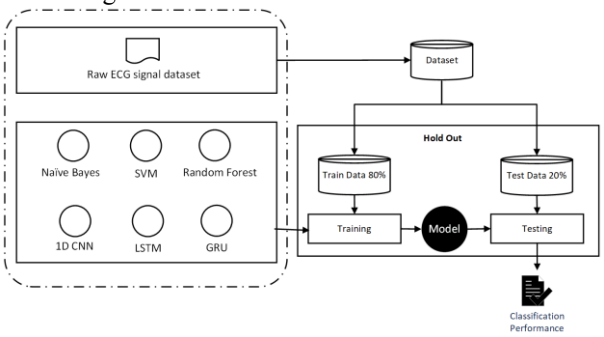


Figure 8: Steps of research implementation.

Based on the picture above, it can be seen that the raw ECG signal data is divided into 80% training data and 20% as test data. Furthermore, the training data will be used in the training process using the classification algorithm. The classification algorithms used in this study is as follows:

- Shallow learning-based classification algorithms are Naïve Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF).
- Classification algorithm based on deep learning, namely 1D Convolutional Neural Network (1D CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

From Figure 8, we conduct six experiments to make six classification models. Then each model will be tested using test data. So there are six performance classifications. The research questions raised in the Introduction section can be answered by comparing these performances.

Implementation of shallow learning algorithms using the Python programming language with the Scikit-Learn library. In Table 2 it can be seen the parameters used for each shallow learning algorithm.

Table 2: Shallow learning: algorithm's parameters

Algorithm	Parameters
NB	prior=none var_smoothing=1e-09
SVM	C=1.0 kernel=rbf degree=3 gamma=scale coef0=0.0
RF	n_estimators=100 criterion=gini max_depth=none min_sample_split=2 min_sample_leaf=1 mint_weight_fraction_leaf=0.0 max_features=sqrt bootstrap=True

The implementation of the deep learning's algorithm uses the Python programming language with the Keras and Tensorflow libraries. The architecture and parameters used in each algorithm can be seen in Table 3.

Table 3: Deep learning: algorithm's parameters

Algorithm	Parameters
1D CNN	1. Convolutional layers: Conv1D • Filter = 256 • Kernel size = 3 • Kernel = orthogonal • Bias = glorot_uniform • Regularizer = regularizers. L1L2(l1=1e-5, l2=1e-4) • activation = relu 2. BatchNormalization layers 3. Pooling layer: MaxPool1D 4. Convolutional layers: Conv1D • Filter = 128 • Kernel size = 3 • Kernel = orthogonal • Bias = glorot_uniform • Regularizer = regularizers. L1L2(l1=1e-5, l2=1e-4) • activation = relu 5. BatchNormalization layers 6. Pooling layer: MaxPool1D 7. Convolutional layers: Conv1D • Filter = 64 • Kernel size = 3 • Kernel = orthogonal • Bias = glorot_uniform • Regularizer = regularizers. L1L2(l1=1e-5, l2=1e-4) • activation = relu 8. BatchNormalization layers 9. Pooling layer: MaxPool1D 10. Convolutional layers: Conv1D • Filter = 64 • Kernel size = 3 • Kernel = orthogonal • Bias = glorot_uniform • Regularizer = regularizers. L1L2(l1=1e-5, l2=1e-4) • activation = relu 11. BatchNormalization layers 12. Pooling layer: MaxPool1D 13. Flatten layer 14. Dense layer • unit=64 • activation = relu 15. Dropout layer 16. Dense layer • unit=32

Algorithm	Parameters
	• activation = relu optimizer=Adam loss=categorical_crossentropy
GRU	1. CuDNNGRU layers • unit = 128 2. BatchNormalization layers 3. CuDNNGRU layers • unit = 64 4. BatchNormalization layers 5. CuDNNGRU layers • unit = 32 6. BatchNormalization layers 7. Dense layer • unit=32 • activation = relu 8. Dropout layer 9. Dense layer • unit=16 • activation = relu 10. Dense layer • unit=4 • activation = softmax optimizer=Adam loss=categorical_crossentropy
LSTM	1. CuDNNLSTM layers • unit = 128 2. BatchNormalization layers 3. CuDNNLSTM layers • unit = 64 4. BatchNormalization layers 5. CuDNNLSTM layers • unit = 32 6. BatchNormalization layers 7. Dense layer • unit=32 • activation = relu 8. Dropout layer 9. Dense layer • unit=16 • activation = relu 10. Dense layer • unit=4 • activation = softmax optimizer=Adam loss=categorical_crossentropy

4. Results

The results of the six experiments conducted in this study can be seen in Table 4. These results show that the highest accuracy for the shallow learning algorithm is 60%, namely by using the Random Forest algorithm. As for the deep learning algorithm, the 1D CNN algorithm works better than other deep learning algorithms with a classification performance of 60%.

Table 4: Classification performance of each model

Learning Type	Methods	Accuracy (%)
Shallow Learning	SVM	50
	NB	35
	RF	60
Deep Learning	1D CNN	60
	LSTM	45
	GRU	30

The classification performance comparison chart can be seen in Figure 9. As explained in section 2.1, Datasets, samples are high dimension data with 2160 features. The Introduction section also mentions that raw ECG data may contain noise and other different frequency components that can interfere with the classification algorithm for pattern recognition. Those problems can cause the low performance of classifications such as SVM, NB, LSTM and GRU.

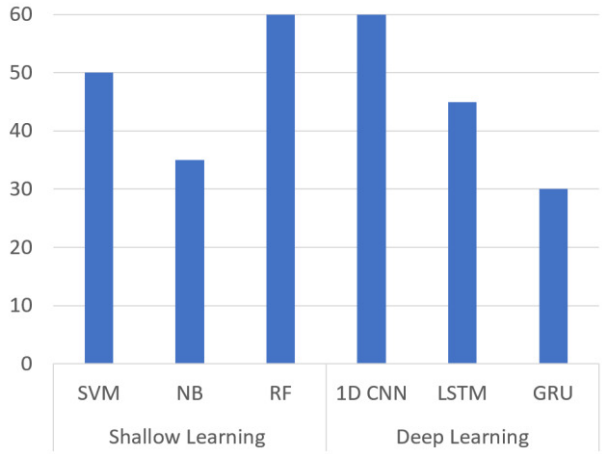


Figure 9: Comparison of classification performance.

Although the classification performance of RF and 1D CNN cannot be categorized as good, these two algorithms can work better than other algorithms. It is because of the characteristics of RF and 1D CNN.

The characteristic of RF is to create many decision trees that take features randomly. With these characteristics, RF can select the features that can provide the best classification performance. Therefore, the RF performance is better than other shallow learning classification algorithms.

The 1D CNN algorithm has a convolutional layer that functions as a filter which becomes the parameter that is updated in the learning process, and an optimum value is obtained that can provide the best classification performance. This is why the performance of the 1D CNN algorithm is better than the other deep learning algorithms used in this study.

5. Conclusions

From the results of this study, it can be concluded that the shallow learning type classification algorithm that can provide the best classification performance on raw ECG signals is Random Forest. In comparison, the 1D CNN algorithm is the best for deep learning classifica-

tion algorithm types. Both of them produce the same classification performance, which is 60%, so the capabilities of the shallow and deep learning classification algorithms work equally well.

The following research will focus on optimizing the 1D CNN algorithm to get the classification performance of raw ECG signals to predict arrhythmias better. The way we are going to do this is to try out different 1D CNN architectures. In addition, hyperparameter tuning and other fine-tuning will be carried out to produce a better classification model.

6. Acknowledgements

In this research, the computer system's computation time was provided by the Data Science Lab of the Computer Science Department, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University. This work was supported by the Program Dosen Wajib Meneliti (PDWM) grant from PNBP Lambung Mangkurat University.

References

[1] A. Shakya, B. Gurung, M. S. Thapa, M. Rai, B. Joshi, Music classification based on genre and mood, International Conference On Computational Intelligence, Communications, and Business Analytics (2017) 168-183, https://doi.org/10.1007/978-981-10-6430-2_14.

[2] R. F. R. Junior, F. A. D. Almeida, G. F. Gomes, Fault classification in three-phase motors based on vibration signal analysis and artificial neural networks, Neural Computing and Applications 32(18) (2020) 15171-15189, https://doi.org/10.1007/s00521-020-04868-w.

[3] M. K. Delimayanti, B. Purnama, N. G. Nguyen, M. R. Faisal, K. R. Mahmudah, F. Indriani, M. Kubo, K. Satou, Classification of brainwaves for sleep stages by high-dimensional FFT features from EEG signals, Applied Sciences 10(5) (2020) 1797-1809, https://doi.org/10.3390/app10051797.

[4] Z. Ebrahimi, M. Loni, M. Daneshtalab, A. Gharehbaghi, A review on deep learning methods for ECG arrhythmia classification, Expert Systems with Applications X 7 (2020), https://doi.org/10.1016/j.eswax.2020.100033.

[5] M. Kropf, D. Hayn, G. Schreier, ECG classification based on time and frequency domain features using random forests. In 2017 Computing in Cardiology (CinC) (2017) 1-4, https://doi.org/10.22489/CinC.2017.168-168.

[6] O. M. A. Ali; S. W. Kareem, A. S. Mohammed, Evaluation of electrocardiogram signals classification using CNN, SVM, and LSTM algorithm: A review, In 2022 8th International Engineering Conference on Sustainable Technology and Development (IEC) (2022) 185-191, https://doi.org/10.1109/IEC54822.2022.9807511.

[7] C. Venkatesan, P. Karthigaikumar, R. J. M. T. Varatharajan, A novel LMS algorithm for ECG signal preprocessing and KNN classifier based abnormality detection, Multimedia Tools and Applications 77(8) (2018) 10365-10374, https://doi.org/10.1007/s11042-018-5762-6.

[8] B. Krithiga, P. Sabari, I. Jayasri, I. Anjali, Early detection of coronary heart disease by using naive bayes algorithm, In Journal of Physics: Conference Series Vol. 1717 No. 1 IOP Publishing (2021) 012-040, http://doi.org/10.1088/1742-6596/1717/1/012040.

[9] R. A. Cahya, C. Dewi, B. Rahayudi, Arrhythmia classification from electrocardiogram results using support vector machine with feature selection using genetic algorithms, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (2018) 1170 – 1178.

[10] T. Wang, C. Lu, Y. Sun, M. Yang, C. Liu, C. Ou, Automatic ECG classification using continuous wavelet transform and convolutional neural network, Entropy 23(1) (2021) 119-132, https://doi.org/10.3390/e23010119.

[11] S. I. Dimitriadis, D. Liparas, How random is the random forest? Random forest algorithm on the service of structural imaging biomarkers for Alzheimer's disease, Neural Regeneration Research 13(6) (2018) 962-970, https://doi.org/10.4103%2F1673-5374.233433.

[12] M. Dovbnych, M. P. Wójcik, A comparison of conventional and deep learning methods of image classification, Journal of Computer Sciences Institute (2021) 303-308, https://doi.org/10.35784/jcsi.2727.

[13] I. Tabian, H. Fu, Z. S. Khodaei, A convolutional neural network for impact detection and characterization of complex composite structures, Sensors, 19(22) (2019) 4933-4958, https://doi.org/10.3390/s19224933.

[14] D. Li, J. Zhang, Q. Zhang, X. Wei, Classification of ECG signals based on 1D convolution neural network, 19th International Conference on e-Health Networking Applications and Services (Healthcom) (2017) 1-6, https://doi.org/10.1109/HealthCom.2017.8210784.

[15] H. M. Lynn, S. B. Pan, P. Kim, A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks, IEEE Access 7 (2019) 145395-145405, https://doi.org/10.1109/ACCESS.2019.2939947.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 138–144
Received: 22 November 2022
Accepted: 18 December 2022

Comparative analysis of VPN protocols

Analiza porównawcza protokołów sieci VPN

Jerzy Antoniuk* , Malgorzata Plechawska-Wójcik

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

Abstract

The aim of the study was to check the performance of the set-up internet connection using the three VPN protocols: Wireguard, OpenVPN and L2TP / IPSec. The Docker applications were used for the tests, on which the VPN server configuration file was launched. The containers were running on the Amazon server. The tests were performed on a laptop and a virtual machine with Windows 10 Pro. The virtual machine has been run in the Microsoft Azure cloud. The next step was to launch three docker containers and start performance tests using three tools: ping command, Speedtest-cli and Iperf3. The result of the research is the analysis of the measurement results and drawing conclusions.

Keywords: computer network; cloud computing; virtual private network

Streszczenie

Celem badania było sprawdzenie wydajności zestawionego połączenia internetowego za pomocą trzech protokołów VPN: Wireguard, OpenVPN oraz L2TP/IPSec. Do badań wykorzystano aplikacje Docker, na której uruchomiono plik konfiguracyjny serwerów VPN. Kontenery były uruchomione na serwerze Amazon. Testy były wykonywane na laptopie oraz maszynę wirtualnej z systemem Windows 10 Pro. Maszyna wirtualna została uruchomiona w chmurze obliczeniowej Microsoft Azure. Kolejnym krokiem było uruchomienie trzech kontenerów dockerowych i rozpoczęcie testów wydajnościowych za pomocą trzech narzędzi: polecenie ping, program Speedtest-cli oraz Iperf3. Wynikiem badań jest analiza wyników pomiarów oraz wyciagnięcie wniosków.

Słowa kluczowe: sieci komputerowe; chmura obliczeniowa; wirtualna sieć prywatna

*Corresponding author

Email address: antoniukjurek@gmail.com (J. Antoniuk)

1. Wstęp

Sieci VPN coraz częściej towarzyszą ludziom w codziennych czynnościach. Szczególnie jest to widoczne od 2020, gdzie duża cześć sektora gospodarki rozpoczęła prace zdalną. W główniej mierze praca zdalna dotyczy branży IT, gdzie tworzone jest oprogramowanie dla firm i instytucji. W takim przypadku sieć VPN jest wykorzystywana do pracy zdalnej w domu. Coraz częściej osoby prywatne korzystają z prywatnej sieci wirtualnej do przeglądania multimediów, niedostępnych z powodu blokady geograficznej.

Celem artykułu naukowego było przeprowadzenie analizy wydajnościowej trzech wybranych protokół sieci VPN.

Teza, która została postawiona przez autora to „Protokół WireGuard jest wydajny w zastosowaniach konsumenckich".

W artykule postawiono następujące szczegółowe hipotezy badawcze:

1. Protokół WireGuard jest wydajniejszy niż starsze protokoły VPN.
2. Protokół WireGuard powoduje mniejsze obciążenie sieci komputerowej.
3. Protokół WireGuard zapewnia stabilne połączenie pomiędzy hostami.

Zakres artykułu obejmuje porównanie wydajności łącza z wykorzystaniem trzech protokołów VPN OpenVPN, Wireguard, L2TP/IPSec. W kolejnych krokach wykonano testy wydajnościowe i analizę otrzymanych wyników.

2. Przegląd literatury

W literaturze naukowej można spotkać wiele badań na temat wydajności sieci VPN. Analizę taką wykonuje się, aby sprawdzić czy jakość usług oferowanych przez wybrany serwer VPN będzie zadowalająca w określonych sytuacjach np. wykonywania zdalnych spotkań dla pracowników firmy. Badania takie również wykonuje się w przypadku, gdy firma będzie musiała wybrać optymalne rozwiązanie, które musi być niezawodnie i wydajnie. Jednak w przytoczonych poniżej badaniach naukowych można zobaczyć, że badane są również inne aspekty wpływające na wydajność połączenia np. opóźnienie pakietów, utrata wysyłanych ramek danych itp.

Najczęściej, analiza wydajności sieci VPN jest wykonywana w celu sprawdzenie maksymalnej przepustowości sieci komputerowej podczas korzystania z wybranego protokołu VPN [1, 2]. Takie analizy są zwykle wykonywane, aby można było stwierdzić jaki protokół VPN będzie optymalny np. do łączenia sieci komputerowych dwóch oddziałów firmy oddalonych o 1000 km. Jednak niektóre badania obejmują szerszy zakres kryteriów niż tylko maksymalna osiągana prędkość transmisji danych [2, 3, 4].

W artykule [5], mierzono wydajność protokołów VPN w dwóch wersji protokołów IP, w którym przedstawiono metody pomiaru przepustowości łącza z wykorzystaniem protokołu OpenVPN oraz WireGu-

ard. W tym przypadku brano pod uwagę trzy kryteria eksperymentu. Pierwszym kryterium było łącze 100Mbps przy każdej wykonywanej próbie. Kolejnym kryterium był czas każdego testu, który wynosił 10 sekund. Ostatnim warunkiem, który miał być spełniony był stopień utraty mniejszy poniżej 1%. Do tej próby wykorzystano dwa komputery, jeden pracujący pod systemem Windows 10 Pro i drugi serwer posiadający z systemem Ubuntu 19.04 Serwer. Pomiary wykonywano za pomocą programu Iperf [1, 6, 7]. Po wykonaniu eksperymentów okazało się, że tym wypadku bardziej wydajnym protokołem był Wireguard, co związane było przede wszystkim z wielkością bufora na serwerze.

Inne podejście do badań w zakresie wydajności protokołów VPN zaprezentowano w pracy [2]. Autorzy analizowali różnice wynikające z wysyłania przez sieć komputerową zróżnicowanych pakietów danych z zakresu od 1000 do 9000 bajtów. Analizę przeprowadzono z wykorzystaniem programu do pomiaru przepustowości Iperf3, podobnie jak w pracy [1]. Tak jak w pierwszej próbie wykorzystano dwa komputery połączone ze sobą. W tym eksperymencie mierzono również opóźnienie przy transmisji danych pakietów za pomocą popularnego narzędzia Iperf3. Po wykonaniu symulacji okazało się, że wielkość pakietów ma wpływ na opóźnienie transmisji sygnałów.

Ostatnim przedstawionym eksperymentem [3] była analiza wydajności sieci VPN w chmurze obliczeniowej. Jest to ważne, ponieważ coraz więcej usług dużych firm korzysta z chmury publicznej. W tym przypadku posłużono się programem komputerowym do wykonywania symulacji OPNET Simulator [4]. Badano trzy warianty połączenia dwunastu komputerów z dwoma serwerami VPN. Pierwszy wariant zakładał połączenie pomiędzy klientami a serwerem pocztowym bez zapory sieciowej i połączenia przez VPN. W kolejnych wariantach brano pod uwagę połączenie przez firewall lub skorzystanie z połączenia tunelowanego oraz włączonego filtrowania ruchu sieciowego. Po wykonaniu eksperymentów który brały wyżej wymienione kryteria oceny eksperymentów, okazało się, że włączenie firewall i VPN zmniejsza wydajność sieci i zwiększa opóźnienia przesyłanych pakietów [10].

Najczęstszym spotkanym rozwiązaniem analizy wydajności połączenia VPN jest połączenie dwóch komputerów ze sobą za pomocą sieci przewodowej. W tym wypadku jedna maszyna pełni rolę serwera usługi VPN a drugie urządzenie - klienta usługi. Najczęściej wtedy wykorzystuje się narzędzie Iperf, które pozwala na pomiar uzyskanych przepustowości zestawionego połączenia. Dobrym przykładem takiej analizy będzie porównanie wydajności protokołu OpenVPN [10, 13]. W tym przypadku połączono dwa komputery stacjonarny z przełącznikiem, Oba hosty miały dostęp do Internetu. W celu przedstawienia rzetelnych i powtarzalnych wyników ustawiono przepustowość łącza na 10Mbps. Komputery były połączone do tej samej podsieci. W tym eksperymencie skupiono się na sprawdzeniu wydajności pakietów przy protokole TCP oraz UDP [10]. Okazało się, że wydajność TCP jest trochę niższa, z powodu enkapsulacji pakietów. Również zauważono, że proces enkapsulacji i dekapsulacji ma wpływ na opóźnienie wysyłania i odbierania pakietów [7, 1].

Bardzo podobnym doświadczeniem [14], które zostało przeprowadzone w sieci dwóch uniwersytetów w Australii było porównanie wydajności łącza oraz opóźnienia sieci. W tym doświadczeniu skorzystano z protokołu IPSec. Aby było możliwe wykonanie tego przedsięwzięcia połączono ze sobą dwie bramki VPN połączone z Internetem na terenie tych uniwersytetów. Klienci VPN byli połączeni z lokalną siecią VPN za pomocą sieci bezprzewodowej. Zestawiono także połączenie pomiędzy typu Multiple Wireless VPN polegające na zestawieniu wielu połączeń jednocześnie [10, 14]. Następnie zrobiono pomiary prędkości i opóźnienia korzystając z oprogramowania Ipferf. Podczas tego eksperymentu sprawdzano prędkość transmisji podczas korzystania z jednego połączenia VPN jak i wielu. Również sprawdzono jakie opóźnienia generuje korzystanie z połączenia tunelowego. Przy okazji sprawdzono, ile procent wysłanych pakietów zostało utraconych podczas transmisji [3, 14]. Okazało się, że korzystanie z połączenia VPN miało duży wpływ na jakość wysyłanych paczek danych, co miało bezpośrednie przełożenie na prędkość i opóźnienia pakietów [9, 7].

3. Eksperyment badawczy

Do eksperymentu badawczego zostało użyte wbudowane do systemu Ubuntu oprogramowanie, ping. Do wykonywania testów przepustowości skorzystano z portali internetowych Speedtest-cli oraz dedykowanego oprogramowania Iperf3.

Podejmowane kroki do wykonania eksperymentu:

- Dziesięciokrotne wykonanie pomiarów prędkości połączenia internetowego za pomocą strony Speedtest-cli przed połączeniem VPN.
- Wykonanie 10 pomiarów za pomocą oprogramowania polecenia ping wbudowanych w system Windows 10.
- Wykonanie 10 pomiarów za pomocą testów jednokierunkowych i dwukierunkowych za pomocą oprogramowania Iperf z domyślnymi wartościami pakietów (segmentacja pakietów 100B, 150B i 200B) i różnymi wartościami pakietów.
- Połączenie się z serwerem VPN zlokalizowanym w chmurze Amazon za pomocą protokołu OpenVPN.
- Ponowne wykonanie pomiarów prędkości łącza za pomocą wcześniej wymienionych stron internetowych.
- Powtórzenie testów wykonywanych za pomocą oprogramowania Iperf z takimi samymi ustawieniami.
- Zmiana protokołu połączeniowego na L2TP/IPSec i ponowne wykonanie pomiarów poprzez przeglądarkę internetową i program Iperf z wcześniej wymienionymi założeniami.
- Zmiana protokołu połączeniowego na WireGuard i 10 krotne wykonanie pomiarów poprzez przeglądarkę firmy Google i program Iperf z wyżej wymienionymi założeniami.

4. Narzędzia badawcze

4.1. Narzędzia

Amazon Web Service

Amazon Web Services jest platformą usług przetwarzania w chmurze oferowaną przez firmę Amazon. Oferuje ona ponad 200 różnorodnych usług dla klientów na całym świecie. Firma udostępnia dla swoich klientów elementy takie jak: bazy danych, pamięć masowa, dedykowane oprogramowanie do uczenia maszynowego itp. Poziom usług oferowany przez firmę Amazon jest dosyć szeroki zaczynając od maszyn wirtualnych i kończąc na dedykowanym oprogramowaniu jako usłudze (ang. Software as a service*).* Dzięki temu można wykorzystać oferowane usługi przez firmę Amazon do różnorodnych zastosowań m.in. utrzymywania stron internetowych opartych na np. Wordpresie. W chmurze obliczeniowej AWS (ang. *Amazon Web Services*) można uruchomić każdy typ aplikacji m.in. aplikacje internetowa oparta na języku Java.

Z niektórych oferowanych usług Amazon Web Service można korzystać za darmo przez 12 miesięcy w ramach oferty Free Tier. Założeniem usługi Free Tier jest zdobywanie doświadczenia z korzystania z usług Amazon Web Services. Oferta zawiera następujące elementy:

- 750 godzin korzystania z maszyn wirtualnych z systemem Microsoft Windows oraz Linux,
- 750 godzin działania modułu równoważenia obciążenia,
- 5GB przestani dyskowej na dane,
- 20GB na bazę danych włącznie z kopiami zapasowymi,
- 2 miesiące działania maszyn wirtualnych odpowiedzialnych za uczenie maszynowe,
- 30 dni sieciowej ochrony antywirusowej.

Platforma Docker

Docker jest to platforma służąca do wdrażania, automatyzacji aplikacji uruchamianych w postaci kontenerów. W ten sposób można uruchamiać wiele aplikacji w postaci odizolowanych od siebie kontenerów. Głównym założeniem tej platformy jest eliminowanie problemów związanych z kompatybilnością aplikacji w środowisku produkcyjnym.

Platforma Docker działa w kontenerze Docker. W tym przypadku każdy obraz maszyny wirtualnej działa niezależnie od systemu operacyjnego gospodarza. Natomiast kontenery korzystają z jednego wspólnego jądra tego systemu operacyjnego działający na platformie hosta. Każdy kontener posiada własny system plików oraz zmienne środowiskowe, który są samowystarczalne do ich działania. Główną różnicą pomiędzy konteneryzacją a tradycyjną wirtualizacji jest wielkość zużywanej pamięci masowej. W tym wypadku kontener z reguły posiada rozmiar około rzędu kilkuset megabitów. W przypadku maszyn wirtualnych rozmiar obrazu jest rzędu kilku lub kilkunastu gigabitów. W ten sposób można uruchomić wiele kontenerów, które jednocześnie nie zużywają dodatkowych zasobów komputera np. pamięci operacyjnej. Zaletą kontenerów jest izolowanie każdej instancji pomiędzy sobą oraz systemem operacyjnym gościa. Pomaga to w zachowaniu wyższego bezpieczeństwa kontenerów. Drugą ważną cechą jaka wyróżnia konteneryzację w porównaniu do klasycznej wirtualizacji jest wydajność. Jest to związane z tym, że kontenery działają w ramach jednego jądra systemu operacyjnego na platformie Docker. Jedną z ważnych zalet architektury Dockera jest to, że kontenery po pobraniu obrazów są natychmiast uruchamiane.

Podsystem WSL

Podsystem Windows dla systemu Linux (WSL) (Podsystem Windows dla systemu Linux, 2017) jest to funkcja wprowadzona w rocznicowej aktualizacji systemu operacyjnego Windows 10 firmy Microsoft. WSL zapewnia pierwszą prawdziwie natywną obsługę aplikacji Linux w systemie operacyjnym Windows poprzez implementację ładowania i wykonywania aplikacji i bibliotek ELF. Pozwala ona na uruchamianie natywnych plików ELF, który zapewnia użytkownikom systemu Windows duży i zróżnicowany zestaw istniejących aplikacji linuksowych, takich jak serwery WWW, e-mail, FTP i SSH, a także pełny zestaw aplikacji dla użytkowników końcowych. Wraz z zapewnieniem prostej metody przenoszenia istniejących aplikacji z systemu Linux na Windows, firma Microsoft zobowiązała się również do długoterminowego wsparcia platformy WSL.

4.2. Sprzęt używany do badań

Specyfikacja sprzętu użytego do eksperymentu została przedstawiona w Tabelach 1—3, gdzie Tabela 1 opisuje serwer VPN, a Tabele 2 i 3 odpowiednio pierwszy i drugi komputer kliencki.

Tabela 1: Specyfikacja serwera VPN

Nazwa	Wartość
Procesor	Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
Pamięć RAM	1GB DDR4
Dysk SSD	8GB
Karta sieciowa	Intel NUC 82371SB PIIX3 ISA [Natoma/Triton II]
System operacyjny	Ubuntu 20.04

Tabela 2: Specyfikacja techniczna komputera klienta mobilnego

Nazwa	Wartość
Procesor	Intel i5-8250u
Pamięć RAM	16GB DDR4
Dysk SSD	256GB
Karta sieciowa bezprzewodowa	Intel® Dual Band Wireless-AC 8265
System operacyjny	Windows 10 Pro 21H1

Tabela 3: Specyfikacja techniczna komputera klienta mobilnego

Nazwa	Wartość
Procesor	Intel Xeon Platinium 8168
Pamięć RAM	4GB DDR4
Dysk SSD	128GB
System operacyjny	Windows 10 Pro 20H2

Specyfikacja urządzenia dostępowego LTE Zyxel LTE4506-M606:

- modem kategorii 6;
- obsługa pasma LTE-FDD 1/3/7/8/20 (800/900/1800/2100/2600 MHz);
- obsługa pasma LTE-TDD 38/40 (2300/2600 MHz);
- obsługa DC-HSPA+/HSPA/UMTS/EDGE/GPRS/GSM;
- prędkość pobierania LTE do 300 Mb/s i prędkość wysyłania do 50 Mb/s;
- obsługa do 32 urządzeń bezprzewodowych;
- WLAN: 802.11a/b/g/n/ac, dwa pasma (2,4 GHz + 5 GHz);
- jedno złącze micro-USB do ładowania i jeden port Gigabit LAN.

Laptop został podłączony do sieci Wi-Fi rozgłaszającej sygnał na częstotliwości 5GHz na kanale 100. Do routera mobilnego połączone było tylko jedno urządzenie. Podczas wykonywana pomiarów skorzystano z aplikacji Speedtest-cli, Iperf3 oraz polecenia ping, który został wbudowany w jądra systemu Ubuntu. System Ubuntu 20.04 został uruchomiony pod systemem Windows 10 Pro za pomocą podsystemu WSL (ang. *Windows Subsystem for Linux*).

W przypadku pozostałych badań połączenie było realizowane do instancji serwera EC2 zlokalizowanego w chmurze Amazon. Fizycznie serwer był zlokalizowany w Europie a dokładniej w mieście Frankfurt. Klient stacjonarny, na którym wykonywano testy wydajnościowe serwerów VPN znajdował się w Centralnej części Stanów Zjednoczonych. Jednak w tym przypadku skorzystano z chmury firmy Microsoft. Było to podyktowane dostępnością systemu Windows 10 Pro z graficznym interfejsem użytkownika.

Pierwszy etapem gromadzenia wyników było zweryfikowanie działania łącza mobilnego LTE pod katem uzyskiwania zakładanych prędkości łącza. Jest to związane z problemem niestabilnej prędkości łącza mobilnego wynikającego z charakterystyki działania sieci komórkowej. W celu zniwelowania zmian prędkości, które mogły wpływać na wyniki, założoną prędkość łącza od klienta ustawiono na 50Mbit/s a klienta na 10Mbit/s. W przypadku łącza stacjonarnego wykonano dziesięciokrotnie test prędkości łącza programem Speedtest-cli z serwerem Orange Polska w Warszawie. W tym przypadku średnia prędkość Internetu wynosi 3100Mbit do klienta na 525 Mbit/s od klienta. Podczas przeprowadzania eksperymentu badawczego sprawdzano następujące parametry:

- upload – jest to przepustowość łącza, gdy użytkownik wysyła dane na serwer znajdujący się w sieci Internet;
- download - jest to prędkość łącza, gdy klient pobiera plik na swój komputer;
- opóźnienie – jest to czas jaki upływa pomiędzy wysłaniem pakietu przez serwer i dotarciem do odbiorcy;
- jitter – jest to zmiana opóźnienia przesyłania pakietu; parametr pozwala na sprawdzenie czy połączenie internetowe jest stabilne;
- packet lost – jest to procent pakietu danych, który został utracony podczas przesyłania ramki do hosta.

Po weryfikacji nastąpiła właściwa cześć przebiegu eksperymentu badawczego. Rozpoczęto pomiary prędkości łącza za pomocą narzędzia Speedtest-cli.

Drugim etapem przeprowadzania eksperymentu było przeprowadzenie testów za pomocą narzędzia Iperf3. W tej części pomiary były wykonywane dziesięciokrotne, gdzie sprawdzano wydajność łącza VPN pomiędzy dwoma hostami pomiędzy nadawcą i odbiorcą. Podczas testów założono wartości maksymalnej segmentacji pakietów na kolejno: 1000, 1500 i 2000 bitów. Za każdym razem powtarzano pomiar dla każdego badanego protokołu, aby wykluczyć wpływ zewnętrznych czynników.

Ostatnim krokiem było sprawdzenie połączenia pomiędzy serwerem VPN a klientem za pomocą programu ping.

W wierszu poleceń systemu Windows 10 mierzone opóźnienia pakietów transmitowane z komputera klienckiego do serwera, gdzie brano pod uwagę następujące parametry:

- minimalne opóźnienie,
- maksymalne opóźnienie,
- średnie opóźnienie,
- odchylenie standardowe opóźnienia,
- procentowa utrata pakietów,
- całkowity czas transmisji.

5. Wyniki badań

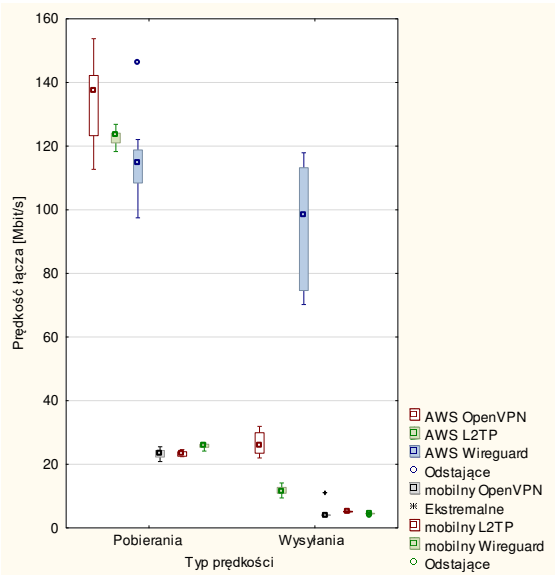
Badania zostały przeprowadzone za pomocą dwóch komputerów klienckich oraz maszyny wirtualnej znajdującej się w chmurze Amazon. Eksperyment polegał na przeprowadzeniu 10 testów za pomocą trzech programów: Speedtest-cli, Iperf3 oraz polecenia ping w wierszu poleceń systemu Ubuntu. System Ubuntu został uruchomiony na platformie WSL w systemie Windows 10 Pro. W tabeli 4 przedstawiono wyniki przepustowości połączenia pomiędzy serwerem a klientem, wykonane za pomocą programu Iperf3. Podczas przeprowadzania testów wydajności podzielono pomiary na dwie grupy określone przez: maksymalną segmentację pakietów o wartościach: 1000, 1500, 2000 bitów i drugą grupę określającą typ urządzenia klienckiego. Podczas wykonywania testów wydajnościowych zaobserwowano, że prędkość połączenia pomiędzy hostami w protokole Wireguard jest niższa o około 50% w stosunku do połączenia klienta stacjonarnego w pozostałych protokołach VPN. W pozostałych przypadkach

różnice pomiędzy prędkościami łącza internetowego wahają się w granicach 20% w zależności od protokołu VPN.

Tabela 4: Prędkości transmisji połączenia klienta z serwerem VPN za pomocą narzędzia Iperf3

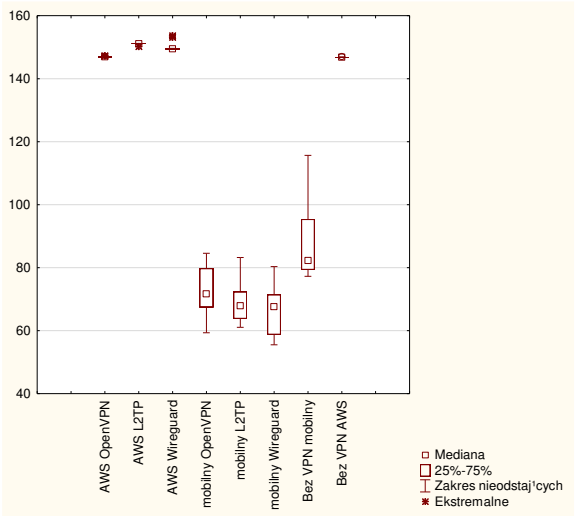
Nazwa	Nazwa protokołu	Urządzenie klienckie					
		stacjonarne			mobilne		
		Segmentacja pakietów					
		1000	1500	2000	1000	1500	2000
sender transfer (Mbits/sec)	Bez VPN	**10,84**	**10,84**	**11,05**	**12,56**	**12,53**	**12,56**
sender bitrate (Mbits/sec)		**9,07**	**9,07**	**9,27**	**10,55**	**10,53**	**10,52**
reciver transfer (Mbits/sec)		**10,78**	**10,78**	**10,96**	**12,53**	**12,52**	**12,52**
reciver bitrate (Mbits/sec)		**9,03**	**9,03**	**9,20**	**10,54**	**10,53**	**10,53**
sender transfer (Mbits/sec)	OpenVPN	**10,91**	**11,20**	**11,28**	**12,55**	**12,50**	**12,70**
sender bitrate (Mbits/sec)		**9,12**	**9,41**	**9,46**	**10,52**	**10,49**	**10,65**
reciver transfer (Mbits/sec)		**10,82**	**11,12**	**11,21**	**12,53**	**12,48**	**12,69**
reciver bitrate (Mbits/sec)		**9,05**	**9,34**	**9,41**	**10,52**	**10,48**	**10,65**
sender transfer (Mbits/sec)	L2TP/IPSec	**10,89**	**11,18**	**11,22**	**12,48**	**12,62**	**12,68**
sender bitrate (Mbits/sec)		**9,29**	**9,40**	**9,43**	**10,49**	**10,59**	**10,63**
reciver transfer (Mbits/sec)		**10,64**	**11,11**	**11,14**	**12,48**	**12,60**	**12,67**
reciver bitrate (Mbits/sec)		**9,05**	**9,34**	**9,36**	**10,49**	**10,60**	**10,62**
sender transfer (Mbits/sec)	Wireguard	**11,23**	**11,48**	**11,40**	**5,12**	**5,16**	**5,27**
sender bitrate (Mbits/sec)		**9,41**	**9,62**	**9,54**	**4,30**	**4,33**	**4,42**
reciver transfer (Mbits/sec)		**11,19**	**11,44**	**11,35**	**5,11**	**5,12**	**5,22**
* reciver bitrate (Mbits/sec)		**9,38**	**9,59**	**9,51**	**4,29**	**4,29**	**4,37**

Na rysunku 1 przedstawiono prędkość połączenia internetowego połączonego za pomocą sieci VPN. Wykres podzieloną na dwa grupy prędkości połączenia internetowego: prędkość pobierania i wysyłania. Na wykresie wyróżnia się prędkość wysyłania klienta stacjonarnego w protokole Wireguard, która waha się w granicach 110Mbit/s. Jednak warto zauważyć, że w pozostałych przypadkach prędkość wysyłania jest średnio 4-krotnie mniejsza niż w pozostałych pomiarach. W przypadku prędkości pobierania, różnice pomiędzy protokołami VPN pomiędzy jednym typem prędkości są niewielkie.



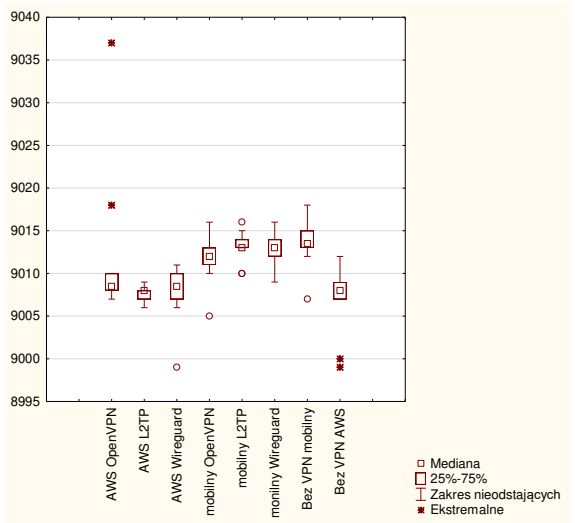
Rysunek 1: Wykres prędkości łącza za pomocą protokołów VPN.

Na rysunku (Rysunku 2) zaprezentowano wykres wartości opóźnienia sygnału zbadane za pomocą polecenia ping w systemie WSL. Jedyną zasadniczą różnicą zaobserwowaną na wykresie to wyższe opóźnienia w pomiarach wykonywanych za pomocą maszyny wirtualnej w chmurze Azure niż w przypadku pomiarów wykonywanych na komputerze mobilnym. Na laptopie opóźnienia transmisji osiągały wartości w zakresie od 1 do 20 ms a na stacjonarnym około 5 ms. W pozostałych przypadkach przedział wartości opóźnienia waha się w granicach 5ms.



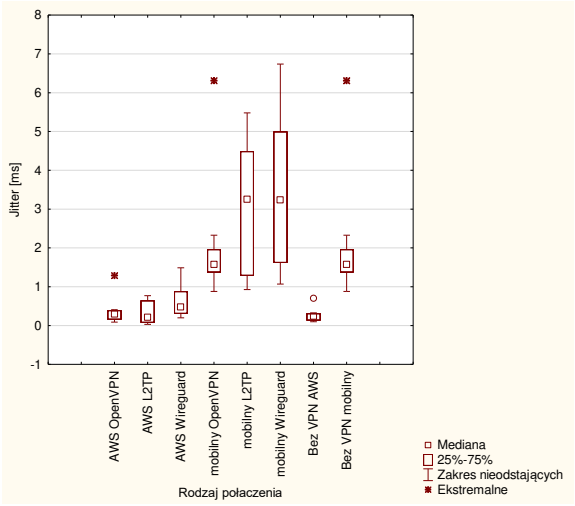
Rysunek 2: Wykres opóźnienia pakietów połączenia pomiędzy klientem i serwerem VPN za pomocą programu ping.

Na rysunku 3 zaprezentowano całkowity czas transmisji pakietu w komunikacji pomiędzy hostami. W tym przypadku pomiary wykonano za pomocą polecenia ping w systemie WSL. Generalnie całkowity czas transmisji jest niższy w przypadku klienta stacjonarnego. Jednak różnice pomiędzy całkowitym czasem transmisji pomiędzy komputerem stacjonarnym a laptopem mieszczą się w granicach 10ms.



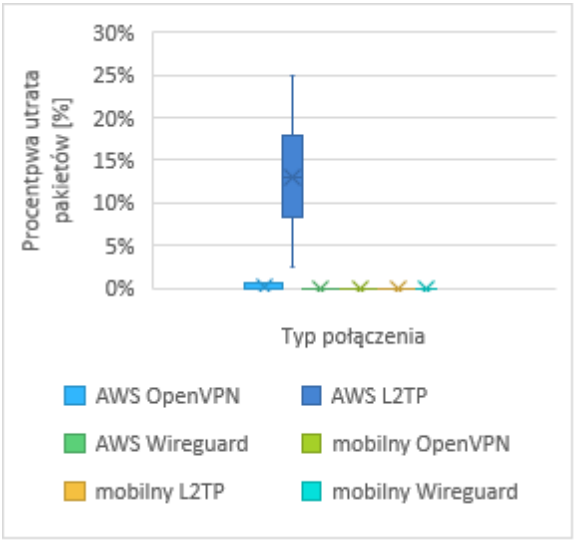
Rysunek 3: Wykres całkowitego czasu transmisji pomiędzy serwerem VPN a klientem w programie ping.

Na rysunku 4 przedstawiono wykres pudełkowy wariancji opóźnienia transmisji pakietów (ang. *Jitter*) w programie Iperf3. W tym przypadku można zauważyć duży zakres zmienności opóźnienia transmisji pakietów w przypadku protokołu OpenVPN oraz Wireguard. W tym przypadku przedział zmienności jittera wynosi 3ms. W przypadku pozostałych pomiarów zmienność opóźnienia transmisji pakietów mieści się w przedziale od 0 do 1,5 milisekund.



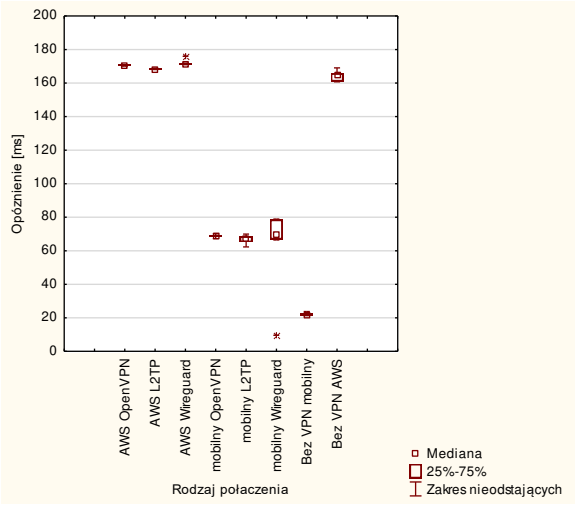
Rysunek 4: Wykres zmiany opóźnień pakietów połączenia (ang. Jitter) pomiędzy klientem i serwerem VPN w programie Iperf3.

Na rysunku 5 zaprezentowano wykres procentowej utraty pakietów pomiarów uzyskany z pomiarów wykonanych za pomocą narzędzia Speedtest-cli. W przypadku protokołu L2TP w komputerze stacjonarnym odnotowano stratę pakietów na poziomie 12,93%. Drugim protokołem, w którym zauważono problem z transmisją danych był OpenVPN. Odnotowano utratę pakietów dla komputera stacjonarnego na poziomie 0,29%. Dla pozostałych protokołów VPN nie wychwycono utraty pakietów.



Rysunek 5: Wykres procentowego utraty pakietów pomiędzy klientem i serwerem VPN w programie Speedtest-cli.

Na ostatnim rysunku 6 pokazano wyniki opóźnień transmisji pakietów danych zmierzonych za pomocą programu Speedtest-cli. W tym przypadku można zauważyć, że opóźnienia bez VPN jest niższe niż w pozostałych scenariuszach eksperymentu badawczego. W przypadku pomiarów wykonanych za pomocą protokołu Wireguard średnia wartość opóźnienia dla komputera mobilnego jest wyższa niż w innych protokołach VPN. Zakres opóźnienia dla protokołu Wireguard wynosi w granicach 15-20ms w przypadku pomiarów na komputerze mobilnym. W pozostałych przypadkach zakres waha się w przedziale 0-5ms.



Rysunek 6: Wykres zmiany opóźnień pakietów połączenia pomiędzy klientem i serwerem VPN w programie Speedtest-cli.

6. Wnioski

Przeprowadzone badania pozwoliły sprawdzić działanie sieci VPN w odniesieniu do wybranych protokołów VPN. Pierwszym spostrzeżeniem, które nasunęło się podczas testów wydajności było to, że prędkość wysyłania dla protokołu Wireguard, która oscylowała w zakresie 100Mbit-140Mbit/s. Warto zauważyć, że w pozostałych przypadkach prędkość połączenia do serwera oscylowała w granicach 20Mbit/s. Kolejnym elementem, który został zauważalny podczas korzystania z serwera VPN na domyślnych ustawieniach było to, że prędkość zestawionego połączenia VPN w protokole Wireguard. Przepustowość była mierzona za pomocą programu Iprerf3 i była ona mniejsza średnio o 50% w stosunku do klienta stacjonarnego. Podczas realizacji eksperymentu nie stwierdzono, czy było to spowodowane użytymi algorytmami do szyfrowania danych. Mogło to być związane konfiguracją stanowiska badawczego w protokole Wireguard. Sytuacja ta nie występowała w pozostałych protokołach VPN. Ważnym aspektem sieci komputerowej jest dostarczanie wszystkich pakietów danych. Jednak podczas mierzenia wydajności połączenia za pomocą programu Speedtest-cli odnotowano utratę pakietów w dwóch protokołach VPN: OpenVPN i L2TP.W pierwszym odnotowano średnio 13% a w drugim: 0,29%. Ostatni aspekt, który wpływa na wydajność łącza internetowego to zakres zmiany opóźnienia sygnału, które wahają się w granicach 10-

20ms dla klienta mobilnego. W pozostałych przypadkach przedział zmienności jitter wynosi około 5ms. Aby uzupełnić omówione analizy zostały także wykonane statystyczne testy nieparametryczne badające różnice pomiędzy poszczególnymi rodzajami połączeń. W badaniach użyto testu Kruskala-Wallisa z poziomem istotności ustawionym na wartość 0,05. W przypadku danych opóźnienia sygnału nie zauważono znaczących różnic pomiędzy wartościami opóźnienia dla poszczególnych grup. Zauważono istotne różnice pomiędzy następującymi grupami: prędkość pobierania połączenia, prędkość wysyłania, opóźnienie sygnału. Podczas wykonywania testów nieparametrycznych, nie stwierdzono istotnych różnic pomiędzy danymi zmienności opóźnienia sygnału i całkowitym czasem transmisji. Po przeprowadzonych eksperymentu badawczego można stwierdzić, że protokół Wireguard jest wydajny tylko w wybranych zastosowaniach. Jednak w celu weryfikacji wyników, należy ponownie wykonać eksperyment badawczy który pozwoli stwierdzić zależność pomiędzy odległością pomiędzy punktami końcowymi połączenia a prędkością łącza internetowego. Również należy wziąć pod uwagę konfiguracje serwerów VPN, które mogły wpływać na wydajność zestawionego połączenia sieciowego.

Literatura

[1] B. Hoekstra, D. Musulin, J. J. Keijser, Comparing TCP performance of tunneled and non-tunneled traffic using OpenVPN, Universiteit Van Amsterdam, System & Network Engineering, Amsterdam (2011) 2010-2011.

[2] C. A. Shue, M. Gupta, S. A. Myers, Ipsec: Performance analysis and enhancements, IEEE International Conference on Communications (2007) 1527-1532.

[3] S. Y. Ammen., S. W. Nourildean, Firewall and VPN investigation on cloud computing performance, International Journal of Computer Science and Engineering Survey 5(2) (2014) 15.

[4] M. Aamir, M. Zaidi, H. Mansoor, Performance Analysis of DiffServ based Quality of Service in a Multimedia Wired Network and VPN effect using OPNET, arXiv preprint rXiv:1206.5469 (2012).

[5] M. Sabbagh, A. Anbarje, Evaluation of WireGuard and OpenVPN VPN solutions, 2020.

[6] P. Dymora, M. Mazurek, T. Pilecki, Performance Analysis of VPN Remote Access Tunnels, Annales Universitatis Mariae Curie-Sklodowska sectio AI–Informatica 14 (3) (2014) 53-64.

[7] S. Ullah, J. Choi, H. Oh, IPsec for high speed network links: Performance analysis and enhancements, Future Generation Computer Systems 107 (2020) 112-125.

[8] M. Pudelko, P. Emmerich, S. Gallenmüller, G. Carle, Performance Analysis of VPN Gateways, IFIP Networking Conference (Networking) (2020) 325-333.

[9] D. Taneja, S. S. Tyagi, Factors Impacting the Performance of Data Transferred Via VPN, International Journal of Innovative Technology and Exploring Engineering 8 (2019) 2962-2966.

[10] I. Coonjah, P. Clarel Catherine, K. M. S. Soyjaudah, Experimental performance comparison between TCP vs UDP tunnel using OpenVPN, 2015, International Conference on Computing Communication and Security IEEE (2015) 1-5.

[11] M. Sakib, J. Singh. Simulation Based Performance Analysis of IPSec VPN over IPv6 Networks, International Journal of Electronics and Information Engineering 12(2) (2020) 92-104.

[12] S. Mackey, I, Mihov, Alex Nosenko, F. Vega, Y. Cheng, A Performance Comparison of WireGuard and OpenVPN, Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy (2020) 162-164.

[13] E. Danilova, Comparing the performance of different VPN technologies with TLS/SSL, 2015.

[14] K. S. Munasinghe, VPN over a wireless infrastructure: evaluation and performance analysis. Diss, University of Western Sydney, 2005.

[15] I. Kotuliak, P. Rybár, P. Trúchly, Performance comparison of IPsec and TLS based VPN technologies, 9th International Conference on Emerging eLearning Technologies and Applications IEEE (2011) 217-221.

[16] J. Ronan, S. Davy, J. Rossebo, An Analysis of IPsec Deployment Performance in High and Low Power Devices, (2004).

[17] R. Munadi, D. D. Sanjoyo, D. Perdana, F. Adjie, Performance analysis of tunnel broker through open virtual private network, Telkomnika 17(3) (2019) 1185-1192.

[18] S. Padhiar, Sneha, P. Verma, A Survey on Performance Evaluation of VPN, International Journal of Engineering Development and Research 3(4) (2015) 516-519.

[19] R. Malik, R. Syal, Performance analysis of ip security vpn, International Journal of Computer Applications 8.4 (2010) 0975.

[20] N. Lewis, Memory forensics and the windows subsystem for linux, Digital Investigation 26 (2018) S3-S11.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 145–153
Received: 25 November 2022
Accepted: 7 December 2022

A comparison of word embedding-based extraction feature techniques and deep learning models of natural disaster messages classification

Porównanie technik wyodrębniania cech opartych na osadzeniu słów oraz modeli głębokiego uczenia w klasyfikacji wiadomości o klęskach żywiołowych

Irwan Budiman, Mohammad Reza Faisal*, Friska Abadi, Dodon Turianto Nugrahadi, Muhammad Haekal

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

Abstract

The research aims to compare the classification performance of natural disaster messages classification from Twitter. The research experiment covers the analysis of three-word embedding-based extraction feature techniques and five different models of deep learning. The word embedding techniques that are used in this experiment are Word2Vec, fastText, and Glove. The experiment uses five deep learning models, namely three models of different dimensions of Convolutional Neural Network (1D CNN, 2D CNN, 3D CNN), Long Short-Term Memory Network (LSTM), and Bidirectional Encoder Representations for Transformer (BERT). The models are tested on four natural disaster messages datasets: earthquakes, floods, forest fires, and hurricanes. Those models are tested for classification performance.

Keywords: Twitter; Natural disaster; CNN; LSTM; BERT

Streszczenie

Badanie ma na celu porównanie skuteczności klasyfikacji wiadomości o klęskach żywiołowych z Twittera. Eksperyment badawczy obejmuje analizę technik ekstrakcji cech opartych na osadzeniu trzech słów oraz pięciu różnych modeli głębokiego uczenia. Techniki osadzania słów używane w tym eksperymencie to Word2Vec, fastText i Glove. Eksperyment wykorzystuje pięć modeli głębokiego uczenia, a mianowicie trzy modele o różnych wymiarach konwolucyjnej sieci neuronowej (1D CNN, 2D CNN, 3D CNN), oraz dwie sieci: Long Short-Term Memory Network (LSTM) oraz Bidirectional Encoder Representations for Transformer (BERT). Modele zostały przetestowane na czterech zestawach danych dotyczących klęsk żywiołowych, a mianowicie trzęsień ziemi, powodzi, pożarów lasów i huraganów. Modele te przetestowano pod kątem wydajności klasyfikacji.

Słowa kluczowe: twitter; klęska żywiołowa; CNN; LSTM; BERT

*Corresponding author

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

1. Introduction

The existence of social media currently plays an essential role in assisting in every activity in the disaster management cycle. In the pre-disaster stage, social media can be used as an early warning before a disaster occurs [1]. At the scene when the natural disaster occurred, eyewitnesses shared information about the situation at that time. It can be used by volunteers or the government to deal with the impact of disasters. Whereas in the post-disaster stage, social media users share messages containing information on relief that has been carried out or information on locations that have not received assistance [2].

Natural disaster messages on social media are categorized into three: *eyewitness* and *non-eyewitness*, and *don't-know* [3]. Messages of the *eyewitness* category are natural disaster messages posted by eyewitnesses at the location when the disaster occurred. Messages in the *non-eyewitness* category are messages about natural disasters uploaded by users who are not eyewitnesses. In contrast, a message in *don't-know* category is a message in which there are words related to natural disasters, but the meaning is not about natural disasters.

Utilization of social media messages related to natural disasters for natural disaster management can be maximized with the help of artificial intelligence. Artificial intelligence can help find natural disaster messages faster [4]. The artificial intelligence system will classify social media messages into the three categories that are mentioned above.

The word embedding-based feature extraction technique is formed by the concatenation of word vectors into 1-dimensional data (1D) [5], [6]. Sentance vectors can be formed by arranging word vectors into a matrix (2D) [7]. Three 2D data created by each word embedding technique such as Wod2vec, Glove and fastText can be combined into 3 layers to form 3-dimensional data (3D) [8]. The output of the feature extraction process is structured data.

The deep learning method that can process multi-dimensional structured data is the Convolutional Neural Network (CNN) [9]–[11]. For text classification with 1D CNN with feature extraction technique based on word2vec [5]. The application of the 2D CNN technique

to classify forest fire messages produces a good accuracy of 81.97%[7]. This study used three-word embedding techniques to create 2D data, namely word2vec, fastText and Glove. The application of text data classification with 3D CNN is made by combining 2D data based on word embedding consisting of three layers based on word embedding techniques word2vec [12], Glove [13] and fastText [8].

Another deep learning technique that is commonly used to classify text is the Long Short-Term Memory Network (LSTM). For the case of sentiment analysis on the IMDB dataset [14], the classification performance obtained using the LSTM model works better than the 1D CNN model. In this study, 1D concatenated data were used from the Word2Vec vector and the 1D CNN model. Meanwhile, the LSTM model uses input data from the tokenised form, which is then converted by an embedding layer based on Word2Vec. For the case of classifying natural disaster messages using deep learning models, the performance of the LSTM model is better than CNN [15]. The CNN model used in this study uses 2D data from the Glove vector and the 2D CNN model. And the LSTM model uses input data from the tokenized form, which is then converted by a Glove-based embedding layer.

Bidirectional Encoder Representations for Transformer (BERT) [16] is a recently popular deep learning method. BERT has achieved state-of-the-art results in a broad range of NLP tasks because of its ability to understand words more thoroughly [17]. BERT can provide a richer linguistic structure because linguistic knowledge is stored in hidden states and on attention maps [18].

The explanation above has provided knowledge of the methods used to carry out feature extraction and classification in the case of sentiment analysis and classification of natural disaster messages. However, it is necessary to carry out a comprehensive comparative study of these methods to obtain knowledge of the technique that can provide the best classification performance in the case of natural disaster messages. This research is done to answer following questions:

1. What are the classification performance of 1D CNN, 2D CNN, and 3D CNN models using the three-word embedding techniques in the feature extraction process?
2. What is the classification performance of the LSTM model using the three-word embedding techniques in the feature extraction process?
3. What is the classification performance of the BERT model?

Existing research generally only uses a word embedding technique for feature extraction. This research also combines feature extraction results based on Word2Vec, Glove, and fastText for processing with 1D CNN, 2D CNN, 3D CNN, and LSTM models. The aim is to determine whether combining those techniques can improve the natural disaster message classification performance.

This report is divided into five sections: 1) Introduction, 2) Dataset and method section explaining the dataset and classification algorithms that are used, 3) Research implementation section explaining the steps in the implementation of this study, 4) Result, and the last section is 5) Conclusions.

2. Dataset

The natural disaster message dataset used in this research comes from research [3]. Details about this dataset can be seen in Table 1.

Table 1: The natural disaster dataset

Dataset	Class Label	#Messages
Earthquakes	eyewitness	1600
	dont-know	200
	non-eyewitness	200
Floods	eyewitness	627
	dont-know	822
	non-eyewitness	551
Forest fires	eyewitness	189
	dont-know	432
	non-eyewitness	1379
Hurricanes	eyewitness	465
	dont-know	336
	non-eyewitness	1199

Each dataset has three categories or class labels: *eyewitness*, *dont-know*, and *non-eyewitness*. Examples of natural disaster messages from each class label can be seen in Table 2.

Table 2: Samples of natural disaster messages

No	Message	Class Label
1	We're a family pulled from a flood	eyewitness
2	I'm ready for these earthquake memes	dont-know
3	Houston streets flood again, dampening July 4th celebrations https://t.co/3I1aOZkNBx https://t.co/4i5kHBcRjo	non-eyewitness

The first message is from natural disasters such as floods and earthquakes uploaded by eyewitnesses when the disaster occurred. The secon message is message that contain the words earthquake and flood, but the meaning is not about natural disasters. While the last message is about natural disasters from the news that are re-shared by Twitter users.

3. Research implementation

The steps in the implementation of the research can be seen in Figure 1.

3.1. Text Normalization & Word Padding

Four natural disaster message datasets were normalized with steps commonly performed in text classification cases: removing double spaces, punctuation marks, numbers and non-alphanumeric characters [19]. The four text data that have been normalized are used as input for the BERT method to create a classification model.

Clean text data is then counted as the number of words in each message. After that, each message in each dataset is equated with the word padding based on the mean value.

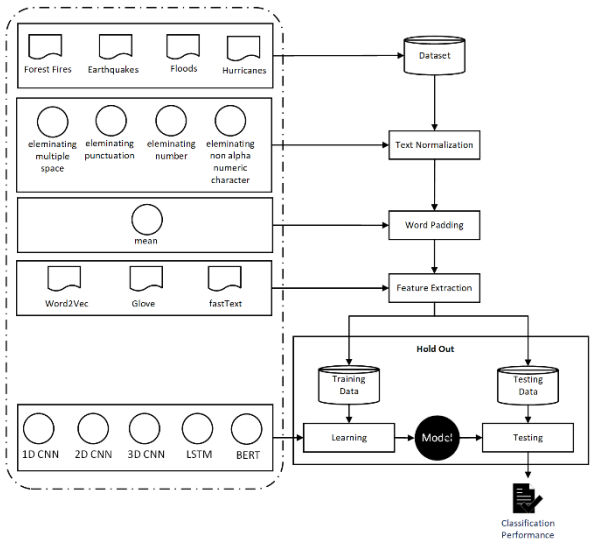


Figure 1: Steps of research implementation.

3.2. Feature Extraction

The next stage is feature extraction. There are three groups of techniques based on the dimensions of the output data, namely, one dimension (1D), two dimensions (2D), and three dimensions (3D). Each of these techniques will use three popular word embedding techniques used in classification studies, namely Word2ve [12], fastText [20], Glove [13], [21]. Data structures with different dimensions are created using the three-word embedding models. The formation of 1D data is explained as follows.

v_i =

V1	V2	V2	…	V98	V99	V100

If v_i is the vector of a word, and n is the number of words in a sentence, then the 1D data is the sentence vector v_s which is formed by combining all word vectors (Formula 1). The value of n corresponds to the number of words searched for with a statistically based word padding technique, namely the mean.

$$v_s = \bigcup_{i=1}^{n} v_i \quad (1)$$

Formula 2 shows how to create 1D data by combining the three-word embedding techniques.

$$v_{all} = v_{s\ word2vec} \cup v_{s\ fastText} \cup v_{s\ Glove} \quad (2)$$

The feature extraction results in this way produce structured data with 1D dimensions, as seen in Table 3. From these results, 16 structured data were obtained, which were used as input to create a classification model using the 1D CNN method.

Table 3: 1D structured data

Input dataset	Word embedding method	Dimension of 1D structured data
Earthquakes	Single word embedding technique	900
	Union of 3 word embedding techniques (All)	2700
Floods	Single word embedding technique	1600
	Union of 3 word embedding techniques (All)	4800
Forest fires	Single word embedding technique	1200
	Union of 3 word embedding techniques (All)	3600
Hurricanes	Single word embedding technique	1300
	Union of 3 word embedding techniques (All)	3900

The formation of 2D data with each word embedding technique is explained as follows. If v_1, v_2, to v_n re word vectors resulting from a number of word embedding techniques m and n is formed by creating a two-dimensional matrix $m \times n$ as shown in Figure 2. Where m is 100.

v_1 =	V1	V2	V2	…	V98	V99	V100
v_2 =	V1	V2	V2	…	V98	V99	V100
… …	…	…	…	…	…	…	…
v_n =	V1	V2	V2	…	V98	V99	V100

Figure 2: 2D Data.

This research also proposes the formation of 2D data from a combination of three-word embedding techniques. The combined 2D data can be seen in Figure 3. The 2D data is a two-dimensional matrix $m \times N$, where m is 100 and N is $3 \times n$.

word2vec	V1	V2	V2	…	V98	V99	V100
	V1	V2	V2	…	V98	V99	V100
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
fastText	V1	V2	V2	…	V98	V99	V100
	V1	V2	V2	…	V98	V99	V100
	V1	V2	V2	…	V98	V99	V100
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
Glove	V1	V2	V2	…	V98	V99	V100
	V1	V2	V2	…	V98	V99	V100
	V1	V2	V2	…	V98	V99	V100
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
	V1	V2	V2	…	V98	V99	V100

Figure 3: 2D data combined with three word embedding techniques.

The feature extraction results in this way produce structured data with 2D dimensions, as seen in Table 4. From these results, 16 structured data are obtained, which are used as input to create a classification model using the 2D CNN and LSTM methods.

Table 4: 2D structured data

Input Dataset	Word Embedding Method	Dimension of 2D structured data
Earthquakes	Single word embedding technique	9 x 100
	Union of 3 word embedding techniques (All)	27 x 100
Floods	Single word embedding technique	16 x 100

Input Dataset	Word Embedding Method	Dimension of 2D structured data
	Union of 3 word embedding techniques (All)	48 x 100
Forest fires	Single word embedding technique	12 x 100
	Union of 3 word embedding techniques (All)	36 x 100
Hurricanes	Single word embedding technique	13 x 100
	Union of 3 word embedding techniques (All)	39 x 100

To create 3D data by combining 2D data from three different word embedding techniques. There are two ways of generating 3D data. The way of forming the first 3D data can be seen in Figure 4. 3D data type 1 is generated in 3 layers. The first layer is 2D data from the word2vec technique, followed by 2D data from the fastText and Glove techniques.

If 3D data type 1 is represented as a matrix, the dimensions are $m \times n \times z$. Where m is 100, n is the number of words in a sentence, and z is 3. The method of forming 3D data type 1 follows the method of forming 3D image data consisting of 3 RGB color channels.

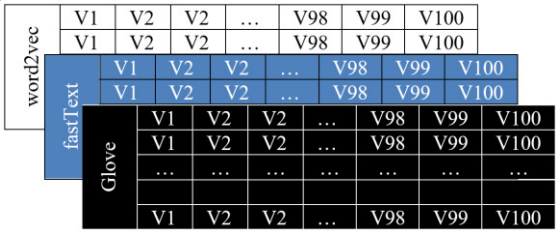


Figure 4: Data 3D type 1.

The way to generate 3D data type 2 is to follow the formation of 3D video. Video are a set of frames or images. For this study, a frame is formed by three vectors of a word from three-word embedding techniques. Then the next layer represents the second word, and so on. So that this data will have n layers. If 3D data type 2 is defined as a matrix, the dimensions are $m \times z \times n$. Where m is 100, z is the number of word embedding techniques, namely 3. n is the number of words in the sentence.

The feature extraction results in this way produce structured data with 3D dimensions, which can be seen in Table 5.

Table 5: 3D structured data

Input Dataset	Word Embedding Method	Dimension of 3D structured data
Earthquakes	Type 1	100 x 9 x 3
	Type 2	100 x 3 x 9
Floods	Type 1	100 x 16 x 3
	Type 2	100 x 3 x 16
Forest fires	Type 1	100 x 12 x 3
	Type 2	100 x 3 x 12
Hurricanes	Type 1	100 x 13 x 3
	Type 2	100 x 3 x 13

3.3. Classification

Three deep learning methods are used in this research: CNN, LSTM, and BERT.

Convolutional Neural Network (CNN) is an artificial neural network used initially in image recognition and processing [22]. The input and output of each stage are in the form of an array called a feature map. The output of each stage is a feature map of the processing results from all input locations. Each stage consists of three layers: the convolutional layer, the activation layer, and the pooling layer [22]. The convolutional layer is the first layer that receives direct input data to the architecture. The convolutional layer performs the convolution operation on the previous layer's output. The purpose of convolution on data is to extract features from the input data. Pooling layer is reducing the size of the matrix by using a pooling operation. Two types of pooling are often used: average pooling and max pooling. The activation function is a node that is added at the end of the output of each neural network. The activation function, also known as the transfer function, is used to determine the neural network output. In the CNN architecture, the activation function lies in the final computation of the feature map output or after the convolution or pooling calculation process to produce a feature pattern. Several kinds of activation functions that are often used in research include the sigmoid, tanh, Rectified Liniear Unit (ReLU), Leaky ReLU (LReLU) dan Parametric ReLU [23].

The result of these three layers is a feature map in the form of a multi-dimensional array. The feature map is then processed by the flatten operation to become a vector. Then the vector is processed by the fully connected layer. The fully connected layer is where all the previous layer's activated neurons are connected to the neurons in the next layer [24]. Each activation of the prior layer needs to be converted into one-dimensional data before it can be linked to all neurons. The Fully-Connected layer is usually used in the Multi Layer Perceptron method to process data so that it can be classified.

Long Short-Term Memory is a variant of Recurrent Neural Network (RNN). LSTM can conduct training and overcome vanishing gradient problems which are difficult for RNNs [25]. LSTM was created with the aim of overcoming the hidden layer problem. LSTM can learn long patterns from sequential data because it prevents vanishing gradient situations. However, LSTM still has the same principle as RNN and what differentiates it from RNN is the cell content. RNN is simple because with cells that only contain a layer of neurons with the tanh activation function. LSTM becomes more complex because the cell's contents are more than one layer of neurons. There is a layer of neurons called gates [26].

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained contextual word representation model based on MLM (Masked Language Model), using two-way Transformers [18]. The BERT model architecture is a multi-layer bidirectional transformer encoder-decoder structure. Transformers follow this overall architecture using self-attention and point-wise stacked, fully connected encoders and decoders. There are two steps in the performance of the BERT framework, namely pre-training and fine-tuning [18]. BERT pre-training does not use the traditional left-to-right or right-to-left method but uses Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) for pre-training data. MLM fills in the blanks, where the model uses the context word around the mask token to predict what word should be, while NSP is the prediction of the next sentence with the two models given. After pre-training data, BERT will perform fine-tuning where fine-tuning is initialized with previously trained parameters, and all fine-tuning parameters use labeled data from downstream tasks.

Model building implementation uses the Python programming language with the Keras and Tensorflow libraries. In this study, three CNN models were made, namely 1D CNN, 2D CNN and 3D CNN. Table 6 shows the parameters used in each model in this study.

Table 6: Parameters of CNN models

Algorithm	Parameters
1D CNN	Input Shape layer • Output = max_word
	Word embedding layer • embedding word, embedding vector, embedding matrix, max word • Output = max_word, 100
	4 convolutional layers = Conv_1, Conv_2, Conv_3, Conv_4 Convolutional 1D • Kernel size = n; n = {1, 2, 3, 5} • Activation=Tanh • Kernel initializer = orthogonal • Kernel regularizer=L1,L2 • Bias initializer = glorot • Output = max_word, 128 Max pooling 1D • Pool size = max_word-n; n = {1, 2, 3, 5} • Padding = valid • Output = max_word, 128; 1, 128 Dropout • P=0.25 • Output = 1, 128
	Concatenate • Input = Conv_1, Conv_2, Conv_3, Conv_4 • Output = 4, 128
	Flatten_1 • Input = flatten • Output = 512
	Dropout_1 • P = 0.35 • Output = 512
	Dense_1 • Units = 3 • Activation = softmax • Output = 3
2D CNN	Input Shape layer • Output = max_word
	Word embedding layer • embedding word, embedding vector, embedding matrix, max word • Output = max_word, 100
	4 convolutional layers = Conv_1, Conv_2, Conv_3, Conv_4 Conv_1 Convolutional 2D • Kernel size= n, embedding vector; n = {1, 2, 3, 5} • Activation=Tanh • Kernel initializer = orthogonal • Kernel regularizer=L1,L2 • Bias Initializer = glorot • Output = max_word, 1, 128 Max pooling 2D • Pool size = max_word-n, 1; n = {1, 2, 3, 5} • Padding = valid • Strindes = 1, 1 • Output = max_word, 1, 128; 1, 1, 128 Dropout • P=0.25 • Output = 1, 1, 128
	Concatenate • Input = Conv_1, Conv_2, Conv_3, Conv_4 • Output = 4, 1, 128
	Flatten_1 • Input = flatten • Output = 512
	Dropout_1 • P = 0.35 • Output = 512
	Dense_1 • Units = 3 • Activation = softmax • Output = 3
3D CNN Type 1	Input Shape layer • Output = max_word
	4-word embedding layers = Word embedding 1, Word embedding 2, Word embedding 3, Word embedding 4 Word embedding • embedding word, embedding vector, embedding matrix, max word • Output = max_word, 100
	Concatenate • Input = Word_embedding_1, Word_embedding_2, Word_embedding_3 • Output = max_word, 300
	4 convolutional layers = Conv_1, Conv_2, Conv_3, Conv_4 Conv_1 Convolutional 3D • Kernal size = **n, embedding vector, 3; n = {1, 2, 3, 5}** • Activation=Tanh • Kernel initializer = orthogonal • Kernel Regularizer=L1,L2 • Bias Initializer = glorot • Output = max_word, 1, 128 Max pooling 3D • Pool size = max_word-n, 1, 1; n = {1, 2, 3, 5}

Algorithm	Parameters
	• Padding = valid • Strindes = 1, 1 • Output = max_word, 1, 1, 1, 128; 1, 1, 1, 128 Dropout • P=0.25 • Output = 1, 1, 1, 128
	Concatenate • Input = Conv_1, Conv_2, Conv_3, Conv_4 Output = 4, 2, 1, 128
	Flatten_1 • Input = flatten • Output = 512
	Dropout_1 • P = 0.35 • Output = 512
	Dense_1 • Units = 3 • Activation = softmax • Output = 3
3D CNN Type 2	Input Shape layer Output = max_word
	4-word embedding layers = Word embedding 1, Word embedding 2, Word embedding 3, Word embedding 4 Word embedding • embedding word, embedding vector, embedding matrix, max word • Output = max_word, 100
	4 convolutional layers = Conv_1, Conv_2, Conv_3, Conv_4 Conv_1 Convolutional 3D • Input = **n, embedding vector, 1; n = {1, 2, 3, 5}** • Activation=Tanh • Kernel initializer = orthogonal • Kernel Regularizer=L1,L2 • Bias Initializer = glorot • Output = max_word, 1, 128 Max pooling 3D • Pool size = (max_word*3)-n, 1, 1; n = {1, 2, 3, 5} • Padding = valid • Strindes = 1, 1 • Output = max_word, 1, 1, 1, 128; 1, 1, 1, 128 Dropout • P=0.25 • Output = 1, 1, 1, 128
	Concatenate • Input = Conv_1, Conv_2, Conv_3, Conv_4 • Output = 4, 2, 1, 128
	Flatten_1 • Input = flatten • Output = 512
	Dropout_1 • P = 0.35 • Output = 512
	Dense_1 • Units = 3 • Activation = softmax • Output = 3

Meanwhile, the LSTM and BERT models were built using the parameters shown in Table 7.

Table 7: parameters of LSTM and BERT model.

Algorithm	Parameters
LSTM	Input Shape layer • Output = max_word
	Word embedding • embedding word, embedding vector, embedding matrix, max word • output = max_word, 100
	Bidirectional • unit = 256 • output = 256
	Dense 1 • unit = 64 • activation = Relu • Kernel Regularizer = L1, l2 • Bias = Glorot_uniform • Output = 64
	Dense 2 • unit = 32 • activation = Relu • Kernel Regularizer = L1, l2 • Bias = Glorot_uniform • Output = 32
	Dropout 1 • P = 0.35
	Dense 3 • Unit 3 • activation = softmax • Output = 3
BERT	Input_ids • Input = max_word • Output = max_word
	Attention_mask • Input = max_word • Output = max_word
	Tf_Bert_model • Input = input_ids, attention_mask • Output = TFBaseModelOutput
	LSTM • Unit = 32 • Output = 32
	Batch normalization • Output = 32
	Dense 1 • unit = 16 • activation = Relu • Kernel Regularizer = L1, l2 • Bias = Glorot_uniform • Output = 16
	Batch normalization • Output = 16
	Dense 2 • unit = 8 • activation = Relu • Kernel Regularizer = L1, l2 • Bias = Glorot_uniform • Output = 8
	Dropout 1 • P = 0.3
	Dense 3 • Unit 3 • activation = softmax • Output = 3

Table 1 shows the difference in the number of samples from each class so that it is known that this is a case of imbalanced data classification. So that the classification performance used in this research is F1 Score and ROC AUC.

Results

This research consists of 15 experiments for each dataset. So that the total number of experiments carried out is 60 experiments. Figure 5 shows a comparison of the classification performance of the earthquake dataset. The 1D CNN model produced the highest classification performance with the fastText word embedding technique, and the lowest was the BERT model.

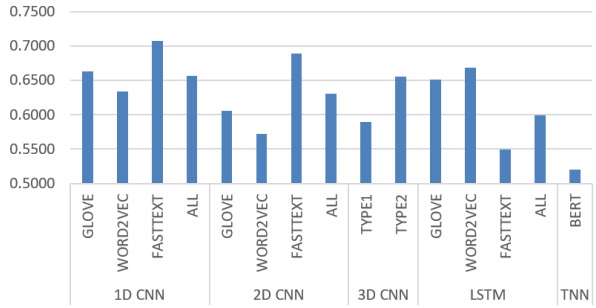


Figure 5: Comparison of performance on earthquake messages classification.

Figure 6 shows a comparison of the classification performance of flood datasets. The CNN type 1 3D model produced the highest classification performance and the lowest was the performance of the LSTM model with the fastText word embedding technique.

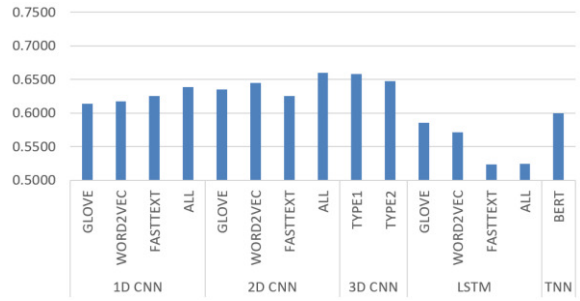


Figure 6: Comparison of performance on flood messages classification.

Figure 7 shows a comparison of the classification performance of forest fire datasets. The highest classification performance is produced by the 2D CNN model with the word embedding Word2Vec technique, and the lowest is the performance of the 1D CNN model with the combination of the three-word embedding techniques.

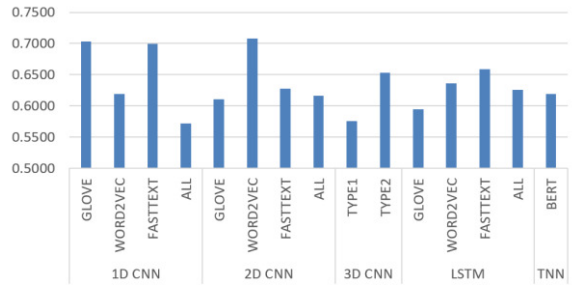


Figure 7: Comparison of performance on forest fire messages classification.

Figure 8 shows a comparison of the classification performance of hurricane datasets. The 3D CNN type 2 model produced the highest classification performance, and the lowest was the performance of the LSTM model with the word embedding Glove technique. The effect of increasing performance with feature extraction combining the three-word embedding techniques works well when used on 1D data with the 1D CNN classification method. This technique also improves the performance of the LSTM classification model.

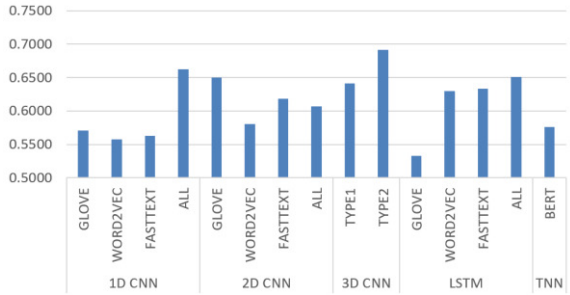


Figure 8: Comparison of performance on hurricane messages classification.

Figure 9 shows the average F1 score based on the classification method. This figure shows that the 3D CNN type 2 method as a feature extraction method proposed in this study, provides the highest average performance compared to other classification methods. In this research, BERT, a state-of-the-art classification method, cannot perform well in the case of natural disaster message classification.

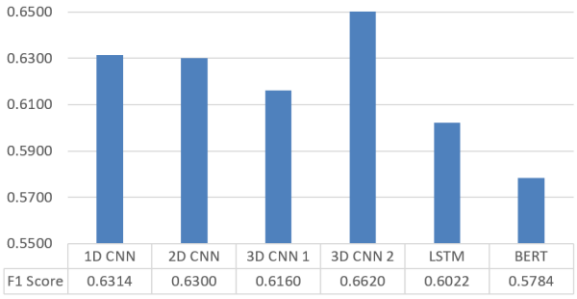


Figure 9. Average F1 Score by classification methods.

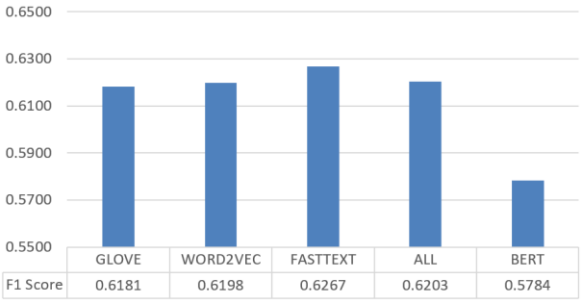


Figure 10: Average F1 score by word embedding methods.

Figure 10 shows the average F1 score based on the word embedding method. The highest classification performance is produced by the fastText-based feature extraction method. However, combining the three word

embedding methods, the proposed feature extraction method can work better than the Glove and Word2Vec methods.

Further analysis of the performance of the 3D CNN type 2 model is to look at the predictive performance of each class in the dataset. Figure 11 shows the average AUC of each class using this model. This figure shows that the predictive performance of the eyewitness class is below the prediction performance of the other two classes.

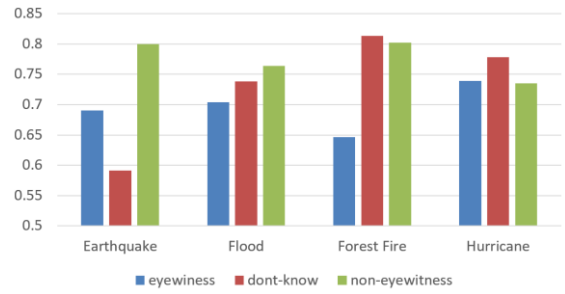


Figure 11: Average AUC of each class by 3D CNN Type 2.

The main reason for the low average performance of this prediction is because the number of messages in the eyewitness category is less than the messages in other categories, except for the earthquakes dataset. Another cause is that eyewitness category messages generally contain concise messages, namely 2-3 words, so the structured data that is formed includes a value of 0. Those reasons can impact the model training process and decrease classification performance [27].

Analysis of the prediction performance of the classes was also carried out by comparing the performance provided by the model formed by fastText-based feature extraction with the incorporation of three word embedding techniques. The results of the performance comparison can be seen in Figure 12. In this result it can be seen the decrease in the prediction performance of class eyewitness from combining the three word embedding techniques. Figure 12 also shows that the predictive performance of the eyewitness class is below the prediction performance of the other two classes. The reason of this issue is same as the explanation in previous paragraph.

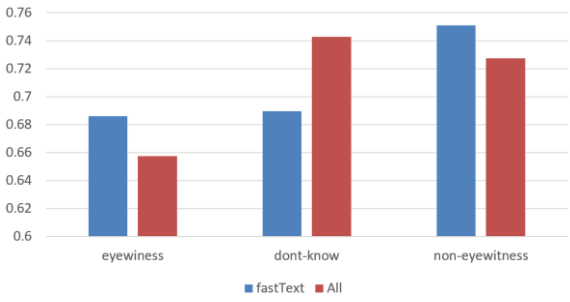


Figure 12: Performance comparison of classes prediction.

Conclusions

From the results of this study, it can be concluded that the formation of 3D data type 2 and 3D CNN models as the proposed method can provide a better average performance in the four cases of classification of natural disaster messages. In comparison, the proposed method combining three-word embedding techniques can improve classification performance in the 1D CNN and LSTM classification models.

However, the average performance for predicting messages from eyewitnesses is still lower than the predictive performance of other message categories. It is because the number of eyewitness category messages is less than the other categories, resulting in cases of unbalanced class classification, which decreases the performance of the minority class classification.

Future research will focus on solving unbalanced data classification cases by balancing the data before creating a classification model. This step is expected to improve the prediction performance for eyewitness category messages.

Acknowledgements

In this research, the computer system's computation time was provided by the Data Science Lab of the Computer Science Department, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University. This work was supported by the Program Dosen Wajib Meneliti (PDWM) grant from PNBP Lambung Mangkurat University.

References

[1] D. Wu, Y. Cui, Disaster early warning and damage assessment analysis using social media data and geo-location information, Decision Support Systems 111 (2018) 48-59, https://doi.org/10.1016/j.dss.2018.04.005.

[2] K. M Rodriguez, S. K. Ofori, L. C. Bayliss, J. S. Schwind, K. Diallo, M. Liu, J. Yin, G. Chowell, I. C. H. Fung, Social media use in emergency response to natural disasters: a systematic review with a public health perspective, Disaster Medicine and Public Health Preparedness, 14(1) (2020) 139-149, https://doi.org/10.1017/dmp.2020.3.

[3] K. Zahra, M. Imran, F. O. Ostermann, Automatic identification of eyewitness messages on twitter during disasters, Information Processing & Management, 57(1) (2020) 102-107, https://doi.org/10.1016/j.ipm.2019.102107.

[4] A. Devaraj, D. Murthy, A. Dontula, Machine learning methods for identifying social media-based requests for urgent help during hurricanes, International Journal of Disaster Risk Reduction 51 (2020) 101757, https://doi.org/10.1016/j.ijdrr.2020.101757.

[5] B. Jang, M. Kim, G. Harerimana, S. Kang, J. W. Kim, Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism, Applied Sciences 10(17) (2020) 5841, https://doi.org/10.3390/app10175841.

[6] M. R. Faisal, R. A. Nugroho, R. Ramadhani, F. Abadi, R. Herteno, T. H. Saragih, Natural Disaster on Twitter: Role of Feature Extraction Method of Word2Vec and Lexicon Based for Determining Direct Eyewitness, Trends in Sciences 18(23) (2021) 680-680, https://doi.org/10.48048/tis.2021.680.

[7] R. Rinaldi, M. R. Faisal, M. I. Mazdadi, R. A. Nugroho, F. Abadi, Eye witness message identification on forest fires disaster using convolutional neural network, Journal of Data Science and Software Engineering 2(02) (2021) 100-108.

[8] J. O. Luna, D. Ari, Word Embeddings and Deep Learning for Spanish Twitter Sentiment Analysis, Communications in Computer and Information Science, 898 (2019) 19-31, https://doi.org/10.1007/978-3-030-11680-4_4.

[9] D. Li, J. Zhang, Q .Zhang, X. Wei, Classification of ECG signals based on 1D convolution neural network, IEEE 19th International Conference on e-Health Networking Applications and Services (Healthcom) (2017) 1-6, https://doi.org/10.1109/HealthCom.2017.8210784.

[10] H.M. Rai, K. Chatterjee, 2D MRI image analysis and brain tumor detection using deep learning CNN model LeU-Net, Multimedia Tools and Applications 80 (2021) 36111–36141, https://doi.org/10.1007/s11042-021-11504-9.

[11] B. Khagi, G. R. Kwon, 3D CNN design for the classification of Alzheimer's disease using brain MRI and PET, IEEE Access 8 (2020) 217830-217847, https://doi.org/10.1109/ACCESS.2020.3040486.

[12] Y. Goldberg, O. Levy, word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, arXiv:1402.3722 (2014).

[13] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (2014) 1532-1543.

[14] U. D. Gandhi, P. M. Kumar, G. C. Babu, G. Karthick, Sentiment Analysis on Twitter Data by Using Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM), Wireless Personal Communications (2021) 1-10, https://doi.org/10.1007/s11277-021-08580-3.

[15] A. Bhoi, S. P. Pujari, R. C. Balabantaray, A deep learning-based social media text analysis framework for disaster resource management, Social Network Analysis and Mining 10(78) (2020) 1-14, https://doi.org/10.1007/s13278-020-00692-1.

[16] J. Devlin, M. W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv:1810.04805 (2018).

[17] S. G. Carvajal, E. C. G. Merchán, Comparing BERT against traditional machine learning text classification. arXiv:2005.13012 (2020).

[18] W. Maharani, Sentiment analysis during Jakarta flood for emergency responses and situational awareness in disaster management using BERT, 8th International Conference on Information and Communication Technology (ICoICT) (2020) 1-5, https://doi.org/10.1109/ICoICT49345.2020.9166407.

[19] M. K. Delimayanti, R. Sari, M. Laya, M. R. Faisal, R. F. Naryanto, The effect of pre-processing on the classification of twitter's flood disaster messages using support vector machine algorithm, 3rd International Conference on Applied Engineering (ICAE) (2020) 1-6, https://doi.org/10.1109/ICAE50557.2020.9350387.

[20] S. Khomsah, R. D. Ramadhani, S. Wijaya, The Accuracy Comparison Between Word2Vec and FastText On Sentiment Analysis of Hotel Reviews, Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi) 6(3) (2022) 352-358, https://doi.org/10.29207/resti.v6i3.3711.

[21] F. Anistya, E. B. Setiawan, Hate Speech Detection on Twitter in Indonesia with Feature Expansion Using GloVe, Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi) 5(6) (2021) 1044-1051, https://doi.org/10.29207/resti.v5i6.3521.

[22] R. Chauhan, K. K. Ghanshala, R. C. Joshi, Convolutional neural network (CNN) for image detection and recognition, 1st International Conference on Secure Cyber Computing and Communication (ICSCCC) (2018) 278-282, https://doi.org/10.1109/ICSCCC.2018.8703316.

[23] A. K. Dubey, V. Jain, Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions, Applications of Computing, Automation and Wireless Systems in Electrical Engineering (2019) 873-880 https://doi.org/10.1007/978-981-13-6772-4_76.

[24] S. S. Basha, S. R. Dubey, V. Pulabaigari, S. Mukherjee, Impact of fully connected layers on performance of convolutional neural networks for image classification, Neurocomputing 378 (2020) 112-119, https://doi.org/10.1016/j.neucom.2019.10.008.

[25] S. Bodapati, H. Bandarupally, R.N. Shaw, A. Ghosh, Comparison and Analysis of RNN-LSTMs and CNNs for Social Reviews Classification, Advances in Applications of Data-Driven Computing (2021) 49-59, https://doi.org/10.1007/978-981-33-6919-1_4.

[26] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Computation 31(7) (2019) 1235-1270, https://doi.org/10.1162/neco_a_01199.

[27] J. Bissmark, O. Wärnling, The Sparse Data Problem Within Classification Algorithms : The Effect of Sparse Data on the Naïve Bayes Algorithm (Dissertation), (2017). http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-209227.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 154–161
Received: 25 March 2023
Accepted: 17 April 2023

Comparative analysis of frameworks and automation tools in terms of functionality and performance on the Salesforce CRM Platform

Analiza porównawcza szkieletów programistycznych i narzędzi automatyzujących pod względem funkcjonalności i wydajności na platformie Salesforce

Damian Sebastian Ciechan*

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

Abstract

Article describes comparative analysis of both code and low-code automation tools together with frameworks used for developing graphical user interfaces that are available on the Salesforce Platform. The research is being carried out due to lack of such comparison in the available literature and due to popularity of the Salesforce CRM. Four automation tools were put together: code-based *Apex Triggers* and three *point-and-click* tools: *Workflow Rules, Process Builder, Flow Builder*. In each of the frameworks (*Visualforce, Aura Components, Lightning Web Components*) an application module was developed and example logic was implemented in each of the automation tools. DML operations *insert, update, delete* were compared in terms of performance and each technology was analyzed in terms of provided functionalities and limitations. It was concluded that the most efficient automation tool is *Flow Builder* and the *Lightning Web Components* framework is the best choice for developing graphical user interfaces.

Keywords: Salesforce; performance; low-code tools; frameworks

Streszczenie

Artykuł opisuje analizę porównawczą narzędzi automatyzujących (niskokodowych i programistycznych) oraz szkieletów do budowania interfejsu graficznego użytkownika dostarczanych wraz ze środowiskiem Salesforce. Badania zostały przeprowadzone ze względu na brak takowych w dostępnej literaturze i ze względu na popularność systemu Salesforce. W zestawieniu porównano cztery narzędzia automatyzujące: oparte na bazie kodu *Apex Triggers* i trzy narzędzia pozwalające na budowanie logiki metodą wskaż i kliknij: *Workflow Rules, Process Builder, Flow Builder*. W każdym ze szkieletów (*Visualforce, Aura Components, Lightning Web Components*) wytworzone zostały trzy analogiczne moduły aplikacji I zaimplementowano logikę w każdym z narzędzi automatyzujących. Operacje DML tworzenia, aktualizowania i usuwania rekordów porównano pod względem wydajnościowym, a każdą technologię przeanalizowano pod względem udostępnianych funkcjonalności i ograniczeń na platformie. Z przeprowadzonych badań wywnioskowano, że najwydajniejszym narzędziem jest *Flow Builder*, a szkielet *Lightning Web Components* jest lepszym wyborem do tworzenia interfejsu graficznego niż jego konkurenci.

Słowa kluczowe: Salesforce; wydajność; narzędzia niskokodowe; szkielety programistyczne

*Corresponding author

Email address: damian.ciechan@pollub.edu.pl (D. S. Ciechan)

1. Introduction

Low-Code Software Development is a new, emerging application development technique that combines minimal amount of source code with graphical user interfaces to reduce development time [1]. In recent years, increasing number of organisations have chosen to use Low-Code Development Platforms (*LCDP*). In many cases, low-code application are developed by so-called *Citizen Developers* [2], i.e. company employees who do not have deep technical or programming knowledge. According to the Gartner report, by 2024, around 65% of large enterprises will be using Low-Code Development Platforms to some degree, and the market is expected to be worth more than $31 billion [3-4].

The popularity of *LCDP* noticeably correlates with the popularity of Customer Relationship Management (CRM) Systems. Providers such as Microsoft or Salesforce provide solutions for both low-code application development and the CRM software themselves. Integrated tools allow employees to customize and extend functionalities in implemented CRM system to support new business requirements. Despite its many advantages, the development of such system can bring new challenges as the business grows. The main one is the size of the data to be processed – out-of-the-box modules and tools have pre-defined limits on how many records they can process simultaneously. Flexibility also has its limits – despite providing ready to use connectors to integrate with external systems, in many cases integration may require deeper technical knowledge and programming skills to ensure everything works flawlessly.

Considering above observations, the comparative analysis was conducted to determine the most performant automation tool and the best framework for front-

end development in terms of functionalities. Evaluation criteria consist of used CPU time on DML operations, whole transaction time, heap memory and network usage. Due to its high market share and popularity, the research was focused on Salesforce products.

2. Salesforce Platform

According to International Data Corporation raport named *Worldwide Semiannual Software Tracker*, Salesforce ranked first as worldwide CRM System provider. This is the ninth consecutive year on the podium with a 23.8% market share [5]. Salesforce provides its products in the *Software as a Service (SaaS)* delivery model, meaning all of the system functionalities are accessible by users from the web browser. This way, *SaaS* minimizes the need to maintain advanced IT infrastructure and all information and data within the CRM System is stored on the provider's servers and disk space. This reduces costs in terms of maintenance and ensures system availability level at >99%, as all updates are carried out remotely, usually during the lowest load hours.

Out-of-the-box, Salesforce provides ready to use environment with functionalities such as:

- cloud applications (Sales Cloud, Marketing Cloud, Service Cloud) all within one environment,
- predefined objects (tables in database nomenclature),
- low-code automation tools (*Workflow Rules, Process Builder, Flow Builder*),
- prioprietary, object oriented programming language *Apex* with database language *SOQL (Salesforce Object Query Language),*
- dedicated front-end frameworks (*Visualforce, Aura Components, Lightning Web Components*),
- Integrated Development Environment – *Visual Studio Code* extension with *sfdx* command line interface,
- REST and SOAP API access to environment.

The platform's architecture is based on *multitenancy* and *metadata*. *Metadata-driven* design means that when creating new field or object, Salesforce internally registers those changes as data (records) in its database table. No data definition operation is executed (i.e. ALTER TABLE) which could block reading and writing data for the duration of processing a potentially lengthy operation. Thanks to *metadata-driven* design, multiple independent environments (*tenants*) can make changes to their instances simultaneously. Although the metadata is physically stored in the same, shared database with identical structure, each tenant has isolated access only to its metadata. Due to its cloud-based architecture, Salesforce enforces limits on each tenant which cannot be exceeded and are taken into consideration in the research:

- CPU time usage per transaction (10 seconds in synchronous context),
- total heap size (6 MB),
- total number of records processed per transaction (10000),
- data storage (10 GB for most licenses, 5 MB in *Developer Edition* license used in research).

2.1. Low-Code automation tools

Workflow Rules is the oldest and most limited tool. In response to insert or update events, it can only perform an update of a field in a given record, send an email to the users associated with the record, create a Task record or send a record SOAP message. Multiple actions can be performed in a single *Workflow*, but the order in which they are performed cannot be modified. Figure 1 shows an example view of defined *Workflow Rule* which sends an email alert.

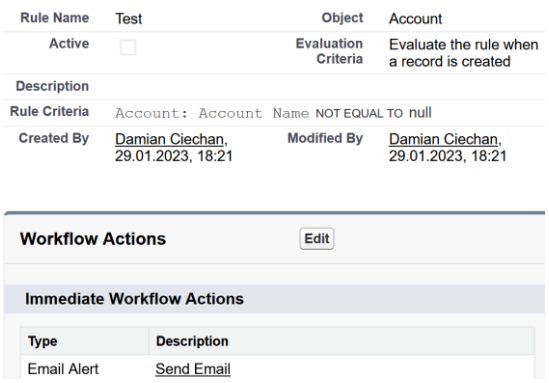


Figure 1: Example of *Workflow Rule.*

Process Builder and *Flow Builder* internally have the same architecture, but the former is better suited for simple tasks. *Process Builder* allows the construction of conditional sets of actions performed one after another (*if – else if - … - else*), while *Flow Builder* allows the branching of the performed operations and their arrangement on the GUI in any manner. Both tools also offer much greater capabilities in terms of available actions to perform compared to *Workflow Rules* – they can update fields on related records, create record of any object, send notifications, execute code from *Apex* classes. Additionally, using *Screen* elements in Flow *Builder* a component can be created that can be embedded into an application view and allow for user interaction. Figure 2 shows automation which was created in *Process Builder* tool, and figure 3 shows the same automation previewed as a *Flow*.

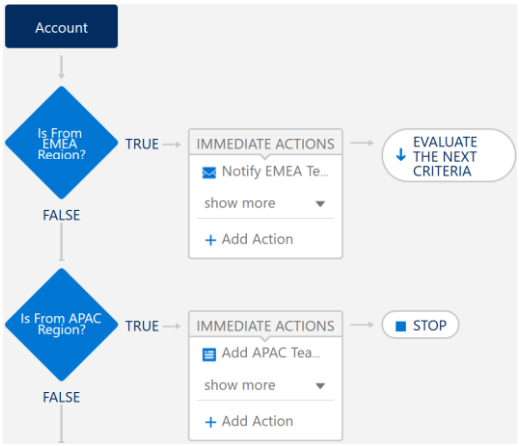


Figure 2: Example *Process Builder* view.

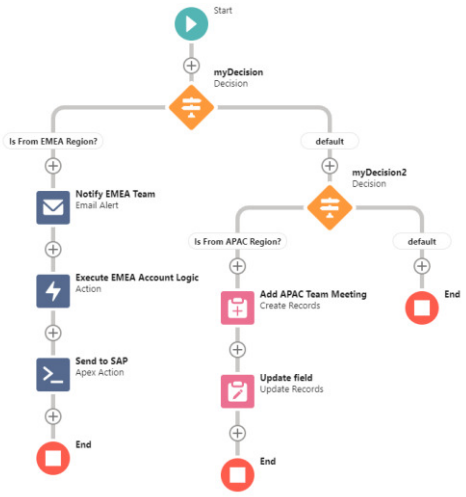


Figure 3: *Process Builder* previewed as a *Flow*.

Table 1 summarizes and compares the capabilities of each tool.

Table 1: Comparison of low-code tools capabilities

Operation	Workflow Rules	Process Builder	Flow Builder
Record creation	Only Task	✓	✓
Record's fields update	Only context record	Context, child and parent record	Any record in the system
Sending email message	Only to users related to the record	Only to users related to the record	Any user in the system
Sending SOAP message	✓		
Sending to approval process		✓	✓
Sending system notification		✓	✓
Can be reused?		✓	✓
Chatter post creation		✓	✓
Apex class invocation		✓	✓
DML listening	Insert, update	Insert, update	Insert, update, delete
Database queries			✓
Deleting records			✓
Logic branching			✓
Versioning		✓	✓
Executing on a regular time interval			✓
Can be used as front-end component?			✓
Debug mode			✓

2.2. Front-end frameworks

There are three available frameworks for developing graphical user interfaces on the Salesforce Platform: *Lightning Web Components, Aura Components, Visualforce*.

The most modern one, *Lightning Web Components*, is built upon standardized W3C Web Components with additional elements required to integrate with Salesforce. As *LWC (Lightning Web Components)* code is run natively by browser's engine and is open source [6], this framework can be used to build any web application (unrelated to Salesforce). The newest version of HTML and Javascript is used to build components, so the learning curve for developers with experience in other front-end frameworks is not very high. Salesforce also provides *sfdx-lwc-jest* CLI add-on to create and run unit tests for the components.

Unlike *LWC*, *Aura Components* is a Salesforce-specific framework; it is not possible to use it to develop applications outside CRM. *Aura* does not support the latest *ECMAScript (European Computer Manufacturers Association Script)* specification – Javascript code must comply with ES5 standards, although some ES6 features are available (e.g. promises). Thus, the entry-level is higher, as the code syntax used in some cases is platform-specific. *Aura Components* is tightly coupled to the Salesforce Platform – it uses its own component model and engine to render the views, resulting (in theory) in lower performance than *LWC*. *Aura Components* also allows unit tests to be written for components, but this requires more configuration – there is a need to manually install *Lightning Testing Service* package on the environment and configure it properly.

The oldest of Salesforce frameworks, *Visualforce*, can be compared to the *JavaServer Pages* technology. It allows pages to be developer using server-side Apex code and platform-specific HTML-like markup language. All business logic is placed in an Apex class associated with the page, called controller. When modifications are made to the page, the platform server compiles the markup language into a set of instructions that can be interpreted by *Visualforce* engine, which returns ready to use HTML document. Unlike previous frameworks, generating the view is done on the server side and consumes resources (time and memory) of the environment instance's processor, consequently offering lower performance. The use of Javascript (ES5 version) is very limited and amounts to placing logic between *<script>* tags – there is no separate file where actions can be delegated. *Visualforce* pages only work within Salesforce, as they are heavily dependent on the platform's server-side language. However, this framework offers a functionality that is absent in other frameworks – native PDF document generation. However, due to limitations in the ability to include CSS styles in such documents (supported only CSS 2.1 version), the preferred approach is to use external Javascript libraries or plugins installed directly on the environment.

3. Literature review

The available literature related to Salesforce is dominated by presentations of various custom applications developed using *Visualforce* and *Aura Components* framework. At the time of the research, no article containing information about *Lightning Web Components* or *Flow Builder* automation tool was available.

The most recent publications [7-8] presents applications for monitoring statistics about Covid-19 disease. In [7] Thanduparakkal et al. using *Aura* framework have developed dashboard named *COVID-19 Tracker* visualizing new cases and number of deaths due to coronavirus. The source of their data was open source REST API *covid19api*. In the article [8] Sharma et al. used fully no-code *Salesforce Einstein Analytics* tool in order to create reports, dashboard and data mining related to the pandemic. As *Einstein Analytics* is powered by artificial intelligence, the publication also shows how the said tool can predict data based on found patterns.

Poniszewska-Maranda et al. [9] presented the *Top 16* Manager application implemented for the Polish Snooker and Billards Association for the management of tournaments in pool games. *Visualforce* framework was used to help the main referee in smoothly managing the tournament by:

- entering match results into the system,
- automatic calculation of players' score,
- automatic generation of competition ladder,
- automatic assignments of referees and players to individual tables,
- email notifications of upcoming matches to players and all tournament participants.

Free *Developer Edition* license was used, which allows up to two users to use the system. From the point of view of Top 16 tournament, that is more than enough as only one person needs access to the system at a given time.

One may question the usefulness of the solution presented by Gupta et al. [10]. Authors have developed a *Visualforce* application for booking metro tickets in the city of Nagpur, India. They mentioned that registration is needed to use the application, but they did not include the information on whether the *Visualforce* site is made public for guests using *Public Site* or *Experience Cloud*. If authorization to internal Salesforce would be required, such solution would be too expensive to implement on wider scale.

In the available literature it is also possible to find articles related directly to the performance of the Salesforce Platform. Miącz [11] in his work analyzed the loading performance of pages created with *Visualforce* framework and the out-of-the-box list views. The main comparison criteria were average number of network requests, file download size, page response and loading times measured with *Chrome Developer Tools*. In the study, the best performant tool was concluded to be standard list view, although the presented results did not differ significantly from each other. The study was also conducted only on 4 and 100 records – in order to obtain a more accurate comparison, the number of records can be increased to 2500 (the limit of data storage in free Salesforce environment edition) or repeat the measured activities multiple time in the system.

The authors of the [12] article focused on the analysis of asynchronous data processing using *Batch, Future, Queueable, Schedulable* and synchronous *Apex Trigger* methods. Total transaction time and the number of records processed per second were chosen as the main comparison criteria. DML insert, update, delete operations were performed on 10000, 50000 and 100000 records. The results obtained by the authors clearly identified *Queuable* as the fastest method for processing data regardless of the number of records (with 885 records created per second in a batch of 100000 records), but they concluded, that *Batch* remains the preferred method if there is a requirement to more closely monitor and manage the amount and order of input data. *Queueable* does not have the transaction splitting mechanism that *Batch* has. Table 2 summarizes the results obtained in article [12] for insert operation – *Apex Trigger* does not include results for more than 10000 records, since synchronous limit is equal to 10000.

Table 2: Number of processed records during insert operation [12]

Method	Number of records		
	10000	50000	100000
	Performance (records/s)		
Batch	653	612	635
Future	399	359	292
Queueable	884	934	885
Schedulable	440	369	396
Trigger	420	-	-

Dan Appleman and Robert Watson at the *Dreamforce 2016* conference [13] performed a detailed analysis of the platform's CPU time usage during the execution of various operations. The authors focused on examining how the certain operations in the Apex language and how the various low-code tools affect the platform's CPU time consumption during transactions. Among other things, they concluded that a static assignment is 30 times faster (~0.58 microseconds vs. ~18 microseconds) than a dynamic assignment and one run of the most efficient *for* loop construction is more than five times faster than the slowest construction (Fig. 4).

```
1  for (Integer i : array) {
2      calculateExecutionTime(); //4 microseconds
3  }
4  for (Integer i = 0; i < array.size(); i++) {
5      calculateExecutionTime(); //2.5 microseconds
6  }
7  Integer s = array.size();
8  for (Integer i = 0; i < s; i++) {
9      calculateExecutionTime(); //0.75 microseconds
10 }
```

Figure 4: Comparison of *for* loop constructions.

In the context of automation tools, *Apex Trigger, Process Builder* and *Workflow Rules* were compared against each other during insert operation on 200 records. *Process Builder* performance proved to be the worst (almost 3 times slower than *Workflow Rules*). Well-designed code proved to be the most performant choice (table 3 presents results obtained by authors of [13]).

Table 3: CPU time consumption for 200 records insert [13]

Automation Tool	CPU time consuption per record (ms)
No automation	1.1
Apex Trigger	2.2
Workflow Rule	2.8
Process Builder	8.2

Above study [13] is particularly relevant for the research carried out in this work. Based on this, one can presume about the low performance of the *Process Builder* tool and it describes good practices that will be used during the development of the individual application modules. The current state of the literature does not include analysis of the *Flow Builder* and whether the choice of framework affects the execution time of server operations. This research will be extended to include the latest tools and the comparison criteria will be expanded.

4. Research method

For the benchmarking, three application modules were created using *Visualforce, Aura Components* and *Lightning Web Components* frameworks. In each of the frameworks, a page was developed that met the same functional requirements – display list of records, buttons to perform insert, update, delete and select list controlling the number of records in the operation (200, 2500). 2500 is the limit of data storage on the *Developer Edition* license. Then, automation logic in each tool (*Apex Trigger, Workflow Rules, Process Builder, Flow Builder*) was implemented which performs the same actions (update Boolean, date, datetime, number and text field values). The Salesforce instance parameters are shown in Table 4. Frankfurt and Paris instance location means that each transaction is replicated in both locations to minimize errors, increase service availability and avoid single points of failure in the Salesforce infrastructure.

Table 4: Salesforce instance parameters

Parameter	Value
License	Developer Edition
Instance	EU46
Location	Frankfurt, Paris
System version	Spring ’23 Patch 9.2

Hardware parameters used to conduct the research are shown in Table 5.

Table 5: Hardware parameters

Parameter	Value
Processor	Intel Core i7-8650U
RAM	24 GB
Storage	512 GB SSD
Graphics	Intel UHD Graphics 620
Operating System	Windows 11 Pro, 22H2
Web Browser	Google Chrome 110

Execution time of each individual task was used as the main comparison criterion. The execution time of DML operation, the time of the entire transaction and the amount of heap memory used were also measured. *Chrome Developer Tools* was used to compare the frameworks – count, time and size of network requests were registered. A newly created object containing only standard set of Salesforce fields and 5 custom fields (*CheckboxField*, *DateField*, *DatetimeField*, *NumberField*, *TextField*) was used. No *Validation Rules, Sharing Rules, Scoping Rules, Restriction Rules* and *Record Types* were defined on this object. The tests were performed by going through the steps from the following scenario:

1. Insert *n* records to the database.
2. Display table of records.
3. Update *n* records in the database.
4. Display updated table of records.
5. Delete *n* records.
6. Display updated of records (empty table).
7. Download measured parameters.
8. Rollback to initial database state.
9. Repeat 1-8 steps for *n:* 200, 2500.

Scenario was executed for each framework and enabled automation combination and repeated 30 times.

5. Results

5.1. Operations processing time

Figures 5-7 show average CPU DML processing time for 200 records grouped by automation tool. Those results are independent from the source of the operation – in this case front-end framework.

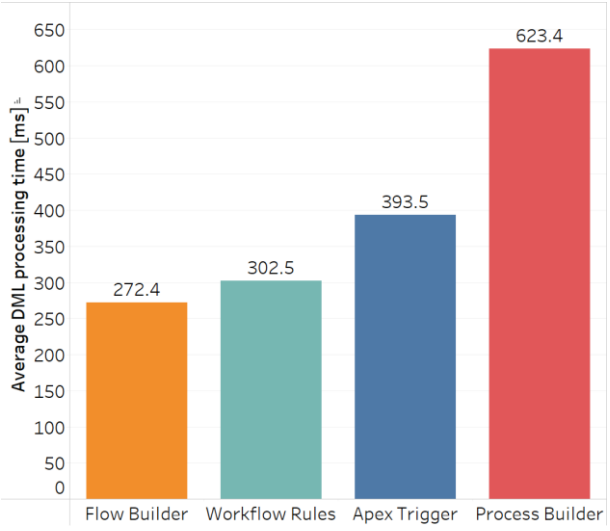


Figure 5: Average DML insert processing time for 200 records.

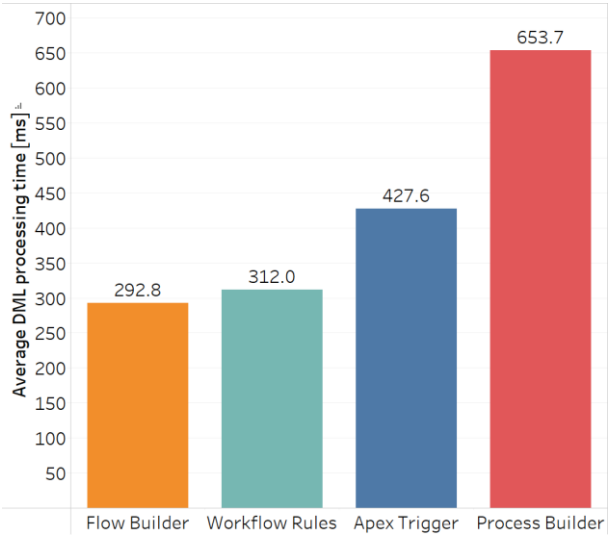


Figure 6: Average DML update processing time for 200 records.

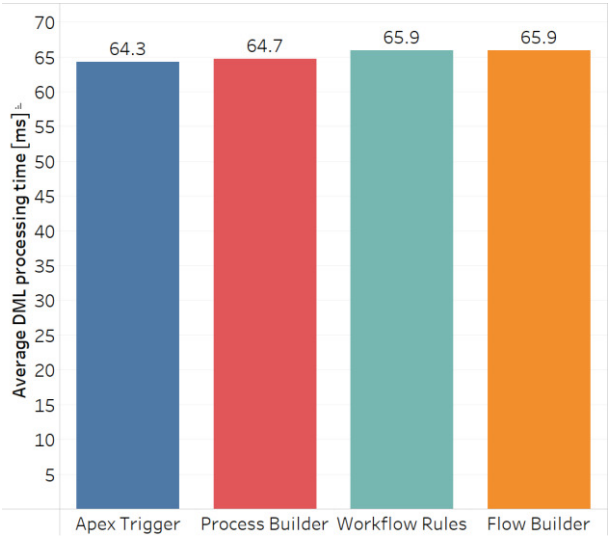


Figure 7: Average DML delete processing time for 200 records.

Similar results were obtained for 2500 records processing with the *Process Builder* having the most influence and greatly extending processing time. Figures 8 to 11 shows average processing time per record for each automation tool.

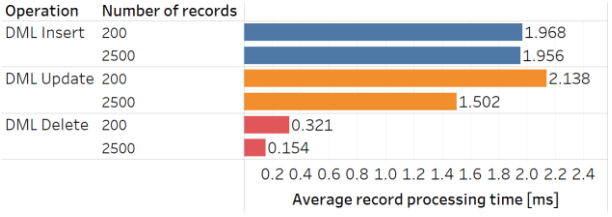


Figure 8: Average record processing time for *Apex Trigger*.

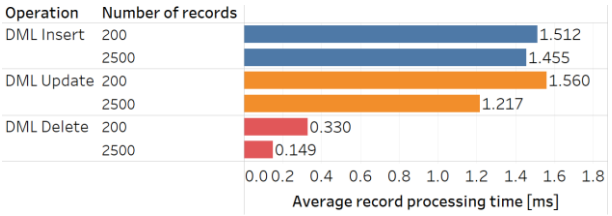


Figure 9: Average record processing time for *Workflow Rules*.

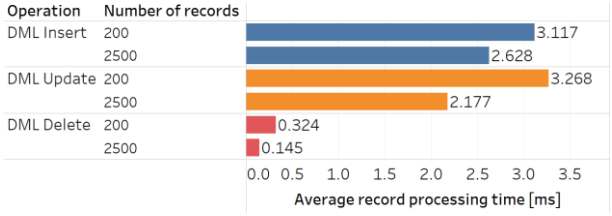


Figure 10: Average record processing time for *Process Builder*.

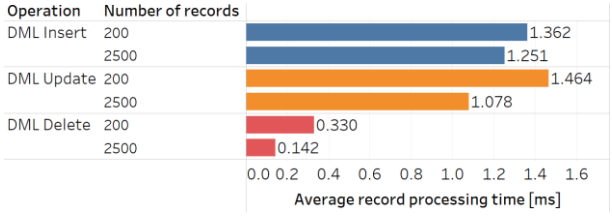


Figure 11: Average record processing time for *Flow Builder*.

Table 6 summarizes average time for whole transaction grouped by DML operation type, number of records and framework.

Table 6: Average transaction time by operation and framework

		Framework		
Operation	Number of records	LWC (ms)	Aura (ms)	Visualforce (ms)
insert	200	1130	1137	2045
	2500	13158	12957	16431
update	200	1148	1200	2258
	2500	13325	13851	17037
delete	200	860	784	962
	2500	5621	5765	5873

5.2. Heap memory consumption

Heap memory consumption depended not on automation but the source of the operation (framework), number of records and type of the operation. Figures 12 to 14 show heap memory consumption for each operation grouped by framework and number of records.

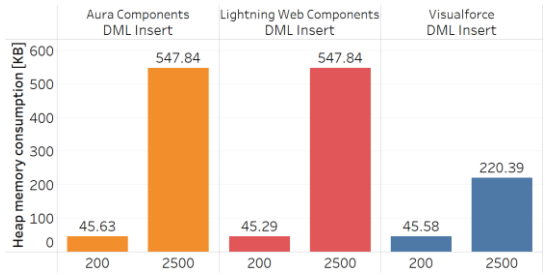


Figure 12: Heap memory consumption for insert operation.

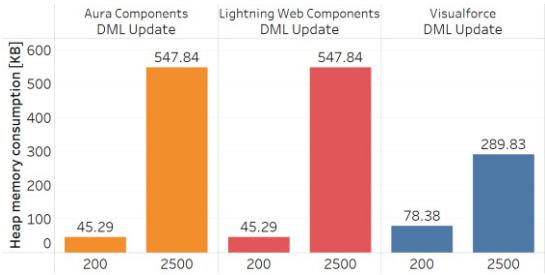


Figure 13: Heap memory consumption for update operation.

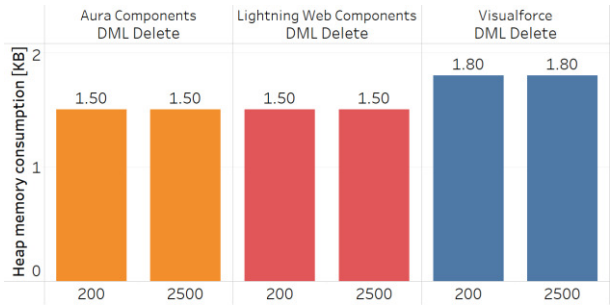


Figure 14: Heap memory consumption for delete operation.

5.3. Network requests

Using *Chrome Developer Tools*, time, count and size of the network requests were measured for each operation. Figure 15 shows average network request size grouped by framework, operation type and number of records.

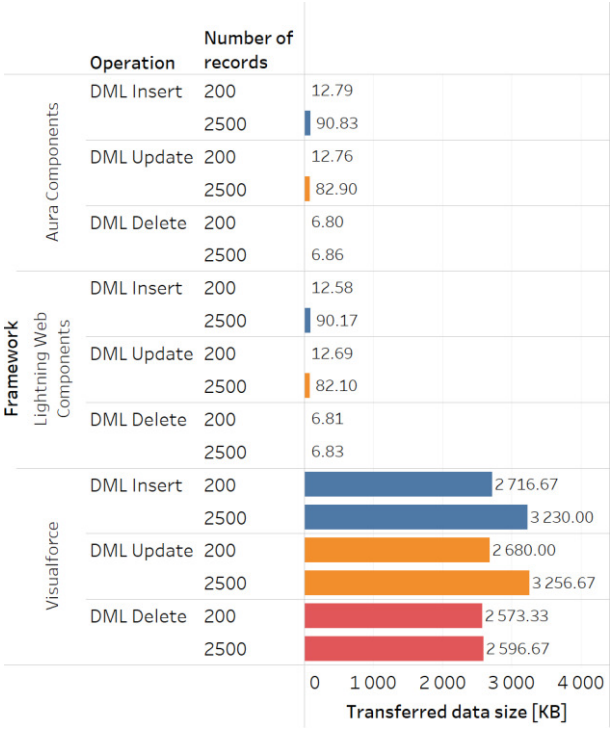


Figure 15: Average network request size by framework.

Table 7 summarizes the number of requests sent in total when executing the scenario 30 times. This value did not vary by chosen automation tool.

Table 7: Overall number of requests sent

		Framework		
Operation	Number of records	LWC	Aura	Visualforce
insert	200	33	33	810
	2500	33	33	810
update	200	33	33	810
	2500	33	33	810
delete	200	33	33	810
	2500	33	33	810

6. Results analysis

The type of enabled automation did not affect the time of the delete operation. The deletion time for 200 records oscillated at ~62 ms, while for 2500 records it was around 350 ms (figures 16 and 17). The difference between the minimum and maximum value for 2500 records is less than 250 ms, i.e. only 2% of the CPU time limit.

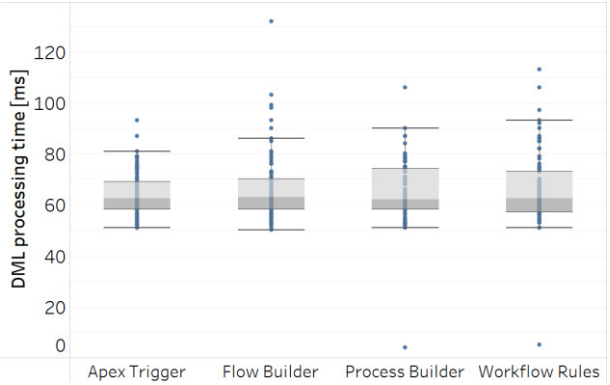


Figure 16: Deletion time for 200 records

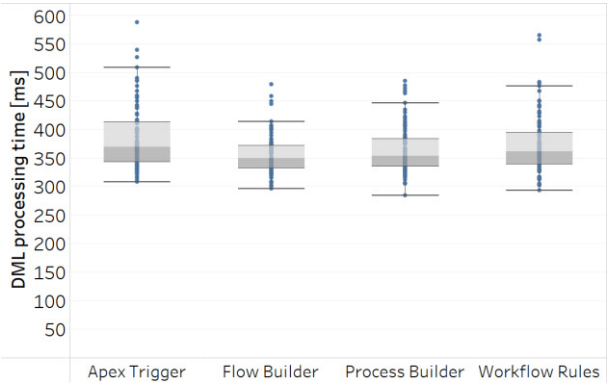


Figure 17: Deletion time for 2500 records.

Process Builder has the most negative impact on record creation and update. Despite its simplified graphical user interface, both insert and update operation were 2 times slower than the fastest *Flow Builder* tool. Having platform's limits in consideration, *Process Builder* uses up to 65% of available CPU time during bulk (2500) operations, leaving limited amount of time for the other operations.

Apex Trigger code ranked penultimate of the four analyzed automation tools. *Workflow Rules* proved to process records ~30% times faster compared to the code. The most performant solution was found to be *Flow Builder* – regardless of the number of records being processed and regardless of the operation type, it performed those operations the fastest.

Correlation can be observed between the front-end framework and the use of server and network resources. *Visualforce* used on average almost twice as much server-side time when creating and updating 200 records compared to the Javascript-based frameworks. When creating or updating 2500 records, the time increased by ~3 seconds. For number of network requests, *Aura* and *Lightning Web Components* achieved similar re-

sults, with a total of 33 requests per DML operation sending minimal amounts of information (a maximum of 90.83 kB, and a minimum of 6.8 kB). *Visualforce*, due to its server-side rendering technology, uses much larger amount of network and server resources – for 200 records, it sent around 2.5 – 2.7 MB of data and for 2500 records – 3.2 MB. Total number of requests added up to 810 during execution of the whole scenario.

Every framework during operation on 200 records achieved similar results in context of heap memory usage. Only for insert and update operations *Visualforce* achieved lower memory usage of 47% (update) and 60% (insert) than the *Aura* and *Lightning Web Components*.

7. Conclusions

In case of automation tools, both in terms of functionalities and the performance of the operations performed, *Flow Builder* turned out to be unquestionable choice. *Workflow Rules* has many limitations and *Process Builder* is highly unoptimized for record processing. *Flow Builder* is a tool tailored for development by citizen developers and can be supported with actions provided by *Apex* developers. Obtained results partly overlap with those presented by Appleman [13], where *Process Builder* was also found to be the slowest tool, although the author obtained worse results for single record processing.

For creating graphical user interfaces, *Lightning Web Component* is the preferred framework.

Visualforce is an outdated tool, offering the lowest performance. Despite using less heap memory, it consumes incomparably more client network resources, which is a higher priority criterion in this analysis.

There are no noticeable differences in the performance results obtained for *Aura* and *Lightning Web Components*, but *LWC* is better suited to modern application development standards than *Aura Components*. Within the Salesforce Platform, both frameworks offer the same capabilities, but *LWC's* open source nature, support for the latest *ECMAScript* specification and architecture based on native *Web Components* make it more suitable choice in the long run – the entry threshold should not be high for developers with experience in another front-end technology.

The results presented in this paper suggest a path for related research in the future. For a more thorough analysis, it would be useful to narrow scope of the benchmarking tests (e.g. comparing only *Flow Builder* with *Apex* code) but implement more complex actions. The *Lightning Web Component* framework allows application to be developed outside of the Salesforce platform, enabling comparative analysis against another popular framework: Angular or React.

References

[1] R. Waszkowski, Low-code platform for automating business processes in manufacturing, IFAC-PapersOnLine 52(10) (2019) 376–381, https://doi.org/10.1016/j.ifacol.2019.10.060.

[2] N. Carroll, L. Móráin, D. Garrett, A. Jamnadass, The importance of citizen development for digital transformation, Cutter IT Journal 34(3) (2021) 5–9.

[3] J. Wong, M. Driver, P. Vincent, Low-code development technologies evaluation guide, Gartner, 2019.

[4] Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023, https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023, [06.03.2023].

[5] Salesforce ranked #1 CRM Provider for Ninth Consecutive Year, https://www.salesforce.com/news/stories/salesforce-ranked-1-crm-provider-for-ninth-consecutive-year/, [21.01.2023].

[6] Lightning Web Components, https://lwc.dev/, [07.02.2023].

[7] H. Thanduparakkal, P. Shahad, C. G. Raji, Using Salesforce to Build Real Time Covid 19 Tracker with Cloud Computing Technology, Proceedings of the International Conference on Applied Artificial Intelligence and Computing, ICAAIC, Salem, India (2022) 942–948, https://doi.org/10.1109/ICAAIC53929.2022.9792802.

[8] V. Sharma, S. Saraswat, S. Verma, P. Banga, D. Gupta, Cost-Effective Data Mining Application Covid-19 Analyzer, Proceedings of the 5th International Conference on Information Systems and Computer Networks, ISCON, Mathura, India (2021) 1–5, https://doi.org/10.1109/ISCON52037.2021.9702328.

[9] A. Poniszewska-Maranda, R. Matusiak, N. Kryvinska, Use of Salesforce platform for building real-time service systems in cloud, Proceedings of the IEEE International Conference on Services Computing, SCC, Honolulu, HI, USA (2017) 491–494, https://doi.org/10.1109/SCC.2017.72.

[10] R. Gupta, S. Verma, K. Janjua, Custom application development in cloud environment: Using salesforce, Proceedings of the 4th International Conference on Computing Sciences, ICCS, Jalandhar, India (2018) 23–27, https://doi.org/10.1109/ICCS.2018.00010.

[11] D. R. Miącz, Analiza wydajności metod tworzenia aplikacji w technologii Salesforce, Journal of Computer Sciences Institute 10 (2019) 24–27, https://doi.org/10.35784/jcsi.189.

[12] W. Marańda, A. Poniszewska-Marańda, M. Szymczyńska, Data Processing in Cloud Computing Model on the Example of Salesforce Cloud, Information 13(2) (2022) 85, https://doi.org/10.3390/info13020085.

[13] D. Appleman, R. Watson, The Dark Art Of CPU Benchmarking, https://www.salesforce.com/video/296515/, [16.02.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 162–170
Received: 2 April 2023
Accepted: 9 May 2023

Influence of video content type on the usefulness of reinforcement learning algorithms in DASH systems

Wpływ typu treści wideo na przydatności algorytmów uczenia ze wzmocnieniem w systemach DASH

Przemysław Grzegorz Markiewicz*, Sławomir Wojciech Przyłucki

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

Abstract

The article presents the result of research on DASH (Dynamic Adaptive Streaming over HTTP) systems. In the proposed solution, the adaptive algorithm is based on the RL (Reinforcement Learning) paradigm. The Pensieve algorithm was chosen as the basis for the tests. This algorithm is widely discussed in the scientific literature and therefore the study and analysis of its properties is useful in a wide range of solutions using DASH. The main contribution of the presented test results to the development of knowledge on video streaming services consists in the analysis of the impact of the characteristics of video materials on the effectiveness of the adaptation process implemented by the developed RL model. The presented results show that this influence should not be omitted in any in-depth analyses of the characteristics of DASH systems.

Keywords: DASH; reinforcement learning; video streaming; QoE

Streszczenie

Artykuł przedstawia wynik badań nad systemami adaptacyjnego strumieniowania DASH (ang. Dynamic Adaptive Streaming over HTTP). W zaproponowanym rozwiązaniu algorytm adaptacyjny oparty jest na paradygmacie uczenia ze wzmocnieniem RL (ang. Reinforcement Learning). Jako podstawę do przeprowadzonych testów wybrany został algorytm Pensieve. Algorytm ten jest szeroko omawiany w literaturze naukowej i dlatego badanie i analiza jego własności jest przydatna w szerokiej gamie rozwiązań wykorzystujących DASH. Główny wkład zaprezentowanych wyników testów w rozwój wiedzy nad usługami strumieniowej transmisji wideo polega na analizie wpływu cech charakterystycznych materiałów wideo na efektywność procesu adaptacji realizowanego przez opracowany model RL. Przedstawione wyniki świadczą o tym, że wpływ zmienności treści wideo nie powinien być pomijany w jakichkolwiek pogłębionych analizach cech systemów DASH.

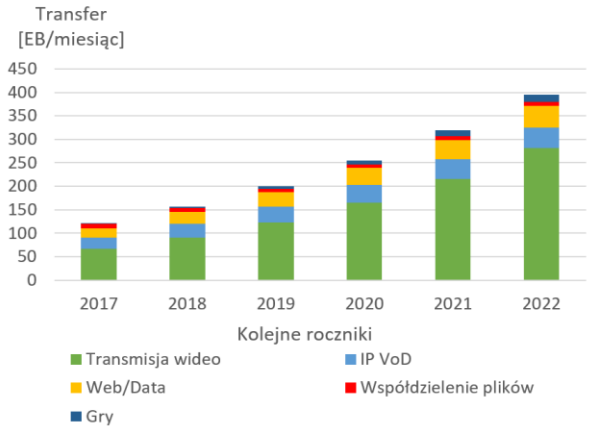
Słowa kluczowe: DASH; uczenie ze wzmocnieniem; transmisja wideo; QoE

*Corresponding author

Email address: **przemyslaw.markiewicz@pollub.edu.pl** (P. G. Markiewicz)

1. Wstęp

Od wielu lat ruch sieciowy [1] na całym świecie jest zdominowany przez platformy transmitujące wideo. Większość z nich wywodzi się z branży rozrywkowej a wśród nich są tacy potentaci jak Netflix, Apple TV, Youtube, Twitch i wiele innych. Dostarczenie dobrej jakości wideo zachowując płynność odtwarzania jest jednym z najbardziej krytycznych problemów technicznych. Popyt i oczekiwania na wysoką jakość wideo wciąż rosną, napędzane przez coraz większe rozdzielczości interfejsów urządzeń mobilnych, czy innego rodzaju odbiorników wideo. Geneza badanego zagadnienia jest złożona, stanowi konsolidację kilku problemów. Dostawcy muszą zapewnić wysoką dostępność usługi, a klienci dostosowują jakość wideo do bieżących parametrów łącza. Ze względu na losowy charakter ruchu sieciowego, możliwość gwarantowania wymaganych wartości przepustowości nie zawsze jest proste. Kolejnym aspektem jest poziom satysfakcji odbiorcy oglądającego transmisję. Użytkownik z założenia ma sprzeczne oczekiwania co do odtwarzanego wideo. Z jednej strony oczekuje on dostępu do treści wideo w możliwie najwyższe jakości wideo, a z drugiej oczekuje zachowania dużej płynności transmisji. Jeżeli dostępna przepustowość wykorzystywanego łącza nie pozwala by w danym momencie odtworzyć wideo najwyższej jakości, może wystąpić konieczność buforowania strumienia danych. W takim przypadku, aplikacja klienta zatrzymuje odtwarzanie wideo i czeka aż bufor zapełni się danymi, po czym wznawia odtwarzanie. Buforowanie wstrzymuje odtwarzanie, więc zakłóca płynność transmisji. Te zjawiska ilościowo opisuje współczynnik QoE (ang. Quality of Experience) [2]. Im wyższa wartość tego współczynnika tym wyższa satysfakcja z odbieranej treści, co oznacza jak najmniej zdarzeń buforowania i wysoką płynność transmisji. By spełnić wymienione wyżej oczekiwania użytkowników, powstały dedykowane rozwiązania transmisji strumieniowej wideo, takie jak DASH (ang. Dynamic Adaptive Streaming over HTTP), czy HLS (ang. HTTP Live Streaming) [3]. Najnowsze implementacje klienta DASH wykorzystują uczenie ze wzmocnieniem. W tej pracy badany jest wpływ parametrów strumieniowanej treści wideo na metrykę QoE w tego typu rozwiązaniach oraz analiza w jakim kierunku należałoby poszerzyć analizę.



Rysunek 1: Estymacja transferu danych w EB/miesiąc w podziale na różne typy danych/aplikacje [1].

1.1. Adaptacyjne strumieniowanie wideo

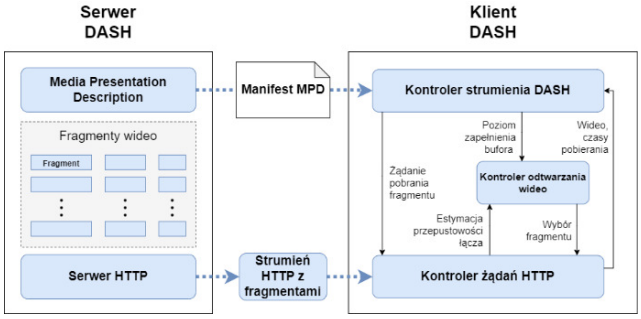
Elementy składowe systemu opartego o ten standard i ich wzajemne relacje są przedstawione na rysunku 2. Standard DASH, w najbardziej podstawowym założeniu co do sposobu komunikacji, jest w pełni oparty o protokół HTTP. W podstawowych założeniach funkcjonalnych system DASH umożliwia dynamiczne dostosowanie wyświetlanych treści adekwatnie do możliwości medialnych urządzenia, wydajności łącza oraz preferencji użytkownika. Klient DASH, próbując odtworzyć wideo, wysyła pierwsze żądania do serwera DASH. DASH w standardzie wymaga od serwera dostarczenia dla klienta definicji metadanych wideo niezbędnych do przeprowadzenia transmisji wideo. Metadane odnoszące się do treści i formatu wideo są umieszczone w pliku MPD (ang. Media Presentation Description) [4]. Jest to plik zawierający tzw. Manifest w formacie XML (ang. Extensible Markup Language). W pliku MPD [5] opisane są informacje o kodowaniu fragmentów wideo, ich rozmiarze, dostępnych jakościach. Dla każdego wideo odtwarzanego w systemie DASH jest wymagana definicja treści wideo zawarta w MPD. Odpowiedzialność za śledzenie stanu transmisji i za decyzję jaki fragment wideo pobrać jako następny oraz jakiej jakości ma być ten fragment, spoczywa na kliencie. Klient odpowiedzialny za dobór odpowiedniej jakości wykorzystuje zaimplementowany algorytm adaptacji ABR (ang. Adaptive Bitrate Streaming). Algorytm ten to zestaw instrukcji, w oparciu o które dobierana jest jakość kolejnego fragmentu wideo w zależności od parametrów łącza czy innych danych wejściowych. Istnieją czynniki zewnętrzne, które mogą zakłócić transmisję. Najbardziej istotne to [6]:

- niedostateczna wartość dostępnej przepustowości łącza internetowego,
- zbyt długi czas odpowiedzi serwera – opóźnienie.

Dostępna przepływność wpływa na to ile czasu zajmie pobranie konkretnego fragmentu wideo. Czas odpowiedzi serwera ma szczególnie istotne znaczenie w przypadku transmisji na żywo. Najważniejszym założeniem algorytmów ABR jest zapewnienie jak najlepszej jakości i płynności transmisji w ramach dostępnych zasobów sieciowych i sprzętowych urządzenia klienta. Zdarzenia negatywnie wpływające na odbiór transmitowanej treści to [2]:

- buforowanie – zatrzymanie transmisji spowodowane tym, że klient nie dysponuje w danym momencie fragmentem wideo w buforze i jest w trakcie jego pobierania,
- częste przełączanie jakości na przestrzeni kolejnych odtwarzanych fragmentów wideo w transmisji.

Zakłócenia są wypadkową parametrów sieciowych transmisji jak i sterowania doborem fragmentów wideo przez algorytm ABR.



Rysunek 2: Diagram działania systemu DASH [34].

1.2. Kierunki badań nad systemami DASH

Pierwsze badania [7] adaptacji strumienia w standardzie DASH oparły się na obserwacji stanu zapełnienia bufora BBA (ang. Buffer Based Approach). Zarządzanie bufora tak by zminimalizować zdarzenia buforowania transmisji wideo zapewnia płynność odtwarzania przez czas całej transmisji. Priorytetem algorytmu było początkowe zapełnienie bufora fragmentami niższej jakości, tak by zapewnić ciągłość w przypadku spadku przepustowości łącza i stopniowe zwiększanie jakości pobieranych kolejnych fragmentów treści wideo. W ten sposób możliwe było amortyzowanie wszelkich wariacji w przepustowości łącza. Jednakże kolejne badania wykazały, że omawiany algorytm jest rozwiązaniem suboptymalnym, ponieważ jego responsywność na dostępną wysoką przepustowość łącza była bardzo opóźniona, co uniemożliwiało poprawienie jakości. Alternatywne rozwiązania jak RB [8] (ang. Rate Based) wykorzystujące obserwację dostępnej przepływności łącza również nie dały zadowalających rezultatów, przyczyniły się do dużej fluktuacji w transmisji wideo co znacznie pogorszyły płynność. Kolejnym krokiem w rozwoju było utworzenie hybrydowych rozwiązań jak [8] MPC (ang. Model Predictive Control) wykorzystujące podejście kontrolowania strumieniem w oparciu o predykcję dwóch wymienionych wcześniej zmiennych (przepływność łącza, stan zapełnienia bufora). Dodatkowo w dalszych badaniach nad rozwojem wprowadzono pojęcie metryki QoE, która umożliwiła obiektywną ocenę algorytmów ABR w oparciu o zmienne, które mają rzeczywiste odzwierciedlenie w odczuciach odbiorcy końcowego strumieniowych usług wideo. MPC realizuje proces predykcji, w którym szacuje się wybór najbardziej opłacalnej przepływności fragmentu w perspektywie metryki QoE. Umożliwiło to uzyskanie o wiele lepszych rezultatów w stosunku do klasycznych

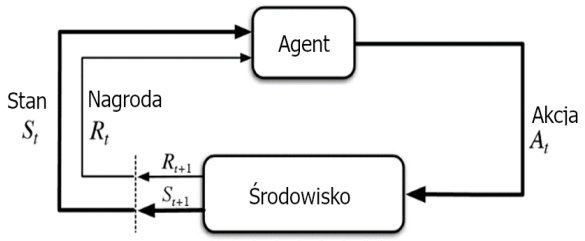
algorytmów (BBA, RB). Jednocześnie metoda oceny metryką QoE stała się podwaliną w opracowaniu kolejnych rozwiązań. Inne rozwiązania jak BOLA-E, czy DYNAMIC [9] również wykorzystują tą metodę.

2. Uczenie ze wzmocnieniem

Uczenie ze wzmocnieniem jest jedną z 3 podstawowych technik uczenia. Większość środowisk uczenia ze wzmocnieniem jest zdefiniowana w postaci procesu decyzyjnego Markov'a MDP (ang. Markov Decision Process), który jest rozbudowaną koncepcją równania Bellmana. Podstawowymi elementami w stochastycznym modelu MDP są [10]:

- środowisko (ang. Environment) – jest to odizolowana skończona przestrzeń w ramach, której agent operuje,
- agent (ang. Agent) – jest to algorytm odpowiedzialny za podejmowanie działań w celu zrealizowania narzuconego zadania/wytycznych,
- cel – są to wytyczne/wytyczna, która mają być finalnie zrealizowane przez agenta,
- stan (ang. State) – agent może przyjąć określony stan po podjęciu określonej akcji, są to etapy przejściowe na drodze do realizacji celu,
- akcja (ang. Action) – jest to działanie/zestaw działań, które może podjąć agent w celu przejścia z jednego stanu do kolejnego. Gama możliwych działań jest narzucona przez dostępne stopnie swobody w środowisku,
- nagroda (ang. Reward) – agent za podjęcie akcji i przejście do kolejnego stanu otrzymuje odpowiednią nagrodę.

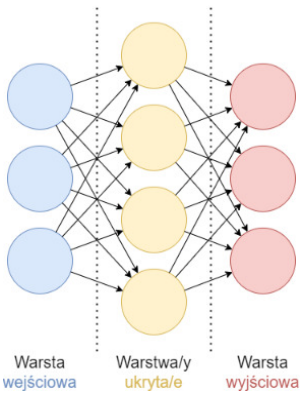
Agent w środowisku, podejmując akcje przechodząc z jednego stanu do kolejnego, otrzymuje nagrodę. Nagroda jest przydzielana adekwatnie do tego czy uzyskany stan przez agenta przybliża go do końcowego celu, czy oddala. W dużym stopniu to w jaki sposób jest przydzielona nagroda jest silnie uzależnione od warunków środowiska jak i implementacji agenta. Każda akcja podjęta przez agenta, która wiąże się z przejściem z pierwotnego stanu do kolejnego wybranego stanu, opisana może być z wykorzystaniem pojęcia prawdopodobieństwa. Każdy stan prowadzący coraz bliżej do zrealizowania celu jest wyżej nagradzany. W ten sposób agent wie jaki jest priorytet w realizacji celu. Tą metodą można znaleźć najlepszą sekwencję akcji (politykę). Znalezienie optymalnej polityki w środowisku o skończonej ilości stanów jest kluczowym celem.



Rysunek 3: Interakcja elementów składowych w modelu uczenia ze wzmocnieniem [11], gdzie t stanowi krok w ciągłej iteracji zmiany stanów.

2.1. Algorytm Pensieve

Algorytm Pensieve został zaproponowany w pracy [12]. Jego fundamentem jest metoda A3C (ang. Asynchronous Advantage Actor-Critic). Jest to asynchroniczny wariant metody A2C (ang. Advantage Actor-Critic) [13, 14]. Podstawowa metoda AC (ang. Actor-Critic) obejmuje założenia co do formy samego agenta i metody maksymalizacji wynagrodzenia. W procesie AC funkcja agenta podejmujące akcje są podzielone na dwie sieci neuronowe. Sieć neuronowa jest to zagadnienie silnie powiązane z głębokim uczeniem [15]. W dużym uogólnieniu sztuczne sieci neuronowe to systemy, które mogą „nauczyć się" rozpoznawać wzorce i relacje między danymi. Nazywa się to siecią neuronową, ponieważ jest inspirowana strukturą i funkcją ludzkiego mózgu. Proste sieci są to modele składające się typowo z 3 warstw powiązanych węzłów/neuronów. Pierwsza warstwa jest wejściowa, a ostatnia pełni rolę warstwy wyjściowej. Poszczególne węzły sieci (sztuczne neurony) operują na wartościach skalarnych i każdy z węzłów ma przypisaną wagę, odpowiedzialną za jego względną istotność w procesie przetwarzania danych w sieci [10]. Agent zawiera sieć neuronową aktora i sieć neuronową krytyka. Sieć neuronowa aktora jest odpowiedzialna za przyjmowanie parametrów wejściowych z środowiska i podejmowanie akcji. Każde podjęte działanie aktora jest oceniane przez sieć krytyka, a jego ocena przekazana jest dalej do aktora jako informacja zwrotna. Proces uczenia nie następuje na końcu każdej iteracji jak przy polityce gradientowej, a już w trakcie danego przebiegu. Dzięki temu również możliwe jest wykorzystanie tej techniki w środowiskach ciągłych i wymaga ona mniejszej liczby iteracji w celu wytrenowania polityki. W przypadku gradientu polityk mamy również do czynienia z dużą wariancją w rezultatach generowanych polityk, co jest również powiązane z ilością iteracji potrzebnych do osiągnięcia konwergencji. Natomiast metoda AC stanowi rodzinę algorytmów, stąd A2C funkcjonuje z tymi samymi założeniami, a różnica wynika w funkcji estymacji sieci krytyka. Tak samo wariant A3C jest rozbudowanym A2C, gdzie środowiska agenta są powielone w procesie uczenia i operują na wspólnym zestawie parametrów [14].



Rysunek 4: Poglądowa struktura głębokiej sieci neuronowej [11].

2.2. Uczenie ze wzmocnieniem w systemach DASH

Obecnie uczenie wzmocnieniem jest dominującym zagadnieniem i kierunkiem badań w celu usprawnienia adaptacji strumienia wideo. Przykładem takich rozwiązań jest system DeepLive [16] do generowania algorytmu ABR wykorzystujący metodę optymalizacji wartości metryki QoE. Jednakże DeepLive różni się od MPC nie tylko samą implementacją, a dodatkowo przeznaczeniem rozwiązania w systemach DASH i inaczej sformułowaną definicją QoE. DeepLive jest zaprojektowany z myślą o platformach z transmisjami na żywo. Oznacza to, że sposób fragmentacji wideo i priorytety odtwarzania transmisji różnią się np. od standardowego wideo na platformie Netflix. Dodatkowym usprawnieniem jest mechanizm omijania klatek. Oznacza to, że jeżeli czas odpowiedzi przekroczy narzucony, dopuszczalny limit to pomijane są klatki wideo by zredukować opóźnienie i pobierane są kolejne. W implementacji Deeplive wykorzystany jest algorytm Double-DQN, czyli złożenie dwóch głębokich sieci neuronowych. Wprowadzono też usprawnienia takie jak użycie algorytmu typu RB na początku wyświetlania transmisji, czy w momencie przewinięcia wideo do określonego miejsca. Okazało się one korzystniejsze ponieważ algorytm Pensieve przyjmował domyślne ustawienia, które okazywały się nieadekwatne do bieżącego stanu systemu. Opracowane rozwiązanie zostało zestawione w testach z innymi algorytmami z grup BBA, RB oraz referencyjnym algorytmem w dash.js DYNAMIC i podobnym rozwiązaniem Pensieve. Z testów zrealizowanych dla różnych symulowanych warunków sieciowych wynika, że Deeplive pod każdym względem zyskuje po około 16% w stosunku do najlepszego ówcześnie rozwiązania jakim był Pensieve.

W innych publikacjach [17] badane jest również zaganienie rywalizacji o dostęp do zasobów. Tradycyjnie główną odpowiedzialność za optymalizację metryki QoE w systemie DASH ponosi klient. Poszerzyło to założenia co do optymalizacji transmisji dla wielu klientów współdzielących to samo łącze internetowe. Rozwiązania klientów korzystających z uczenia ze wzmocnieniem np. DeepLive, czy Pensieve, w swojej ograniczają się do samolubnej alokacji łącza w celu maksymalizacji metryki QoE. W przypadku wielu klientów wykorzystujących powyższe rozwiązania może to spowodować, że klienci będę rywalizować co negatywnie wpłynie na jakość odbieranych transmisji (na wartości metryki QoE każdego klienta) i spowoduje niezadowolenie użytkowników dostrzegających brak płynności w wyświetlanej treści. Rozwiązanie FBAC [12] korzysta z pomocy serwera sterując przepływnością łącza dla klientów. Analogicznie jak na potrzeby klienta w politykach ABR wygenerowanych metodami uczenia ze wzmocnieniem korzysta się z mechanizmów optymalizacji metryki QoE, tak i dla serwera opracowana została nowa metryka optymalizacji QoE. Natomiast do realizacji kontroli przepływności łącz klientów wykorzystany został algorytm generowany metodą DRL (ang. Deep Reinforcement Learning). Nowo zaproponowana metryka QoE jest nazwana metryką uczciwości, czyli pomaga ustalić na ile sprawiedliwie jest przydzielany dostęp do przepływności łącza w kontekście maksymalizacji wartości metryk QoE uzyskiwanych u poszczególnych klientów. FBAC korzysta z mechanizmu kontrolowania przeciążeń. W FBAC również wykorzystana jest metoda uczenia ze wzmocnieniem, ale w tym wypadku autorzy zdecydowali się na zastosowanie metody opartej na niezależnej proksymalnej polityce optymalizacji IPPO (ang. Independent Proximal Policy Optimization). Przedstawione rozwiązanie systemowe zostało zweryfikowane w środowisku testowym. Środowisko to obejmowało dedykowany serwer i wielu klientów wykorzystujących algorytm MPC, dla których serwer przeliczał swoją globalną wartość metryki w oparciu o wartości metryk QoE uzyskane od klientów. Mechanizm sterujący przeciążeniami został oparty o mechanizm QUIC (ang. Quick UDP Internet Connections) zaimplementowany w serwerze DASH. Zrealizowano serię testów zestawiając ze sobą różne treści wideo w różnych jakościach dostosowane do standardu DASH. Scenariusze testowe obejmowały wiele klientów z różnymi implementacjami polityki ABR: MPC, Pensieve, BOLA, FBAC z serwerem Nginx i protokołem QUIC. Wyniki tych testów wykazały, że metryka uczciwości jak i średnia wartość metryk QoE klientów w przypadku FBAC była wyższa o 9-13% w stosunku do pozostałych polityk ABR. To oznacza, że odgórne sterowanie łączem, tak by zoptymalizować metrykę uczciwości, pozytywnie wpłynęło na wartości metryki QoE po stronie klientów.

3. Metodyka badań

Opracowane środowisko testowe, do weryfikacji własności algorytmów ABR generowanych w oparciu o uczenie ze wzmocnieniem, składa się z dwóch podstawowych modułów. Pierwszy z nich to moduł odpowiedzialny za proces przygotowania modelu, natomiast drugi pozwala na przeprowadzenie symulacji wykorzystujących wytrenowany model i akwizycję danych pomiarowych. Korzystając z doświadczeń innych zespołów badawczych [12, 18] zostało zaprojektowane oraz wdrożone środowisko testowe o opisanej poniżej strukturze.

3.1. Środowisko testowe

Algorytm ABR został wygenerowany metodą uczenia ze wzmocnieniem zaktualizowanym systemem Pensieve. Platforma testowa observing-rl-agents-netai2019 [18] posłużyła do ewaluacji modelu w testowym środowisku symulacyjnym oraz do wizualizacji zgormadzonych rezultatów. System Pensieve wymaga zbiorów danych wejściowych o określonym szczegółowo formacie w celu wytrenowania modelu. Jednym z podstawowych składników tego zbioru są zapisy parametrów transmisji w sieciach komputerowych, które są wykorzystywane w procesie uczenia agenta jako informacja o dostępnej w danej chwili przepustowości łącza. Kolejnym składnikiem zbioru danych wejściowych są fragmenty wideo. Każdy z tych fragmentów reprezentuje określoną przepływność/jakość fragmentu w danym

przedziale czasu. Drugi moduł realizuje dwa etapy badań. Pierwszy z nich to ewaluacja wytrenowanego modelu w środowisku symulacyjnym wykorzystującym zmodyfikowane zapisy transmisji w sieci komputerowej [19] oraz generowanych sztucznie zapisy transmisji na łączach o stałej przepustowości. Drugim etapem jest analiza zebranych rezultatów ewaluacji modelu z użyciem narzędzia lime [20]. Jednym z podstawowych celów zastosowania tego narzędzia jest sprawdzenie i ocena wpływu poszczególnych cech wejściowych na jakość odpowiedzi generowanych przez badany model. Ocena stanowi wartość metryki QoE [8]:

$$QoE_1^K = \sum_{k=1}^{K} q(R_k) - \lambda \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)| - \mu \sum_{k=1}^{K} \left(\frac{d_k(R_k)}{c_k} - B_k \right) - \mu_s T_s \quad (1)$$

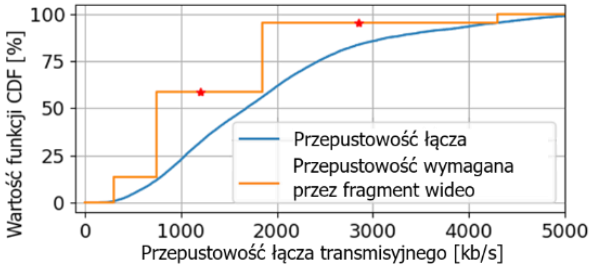
gdzie:

- $\sum_{k=1}^{K} q(R_k)$ – średnia wartość jakości fragmentu wideo w perspektywie całego wideo,
- $\sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)|$ – obserwowalny zakres zmian w jakości między kolejnymi fragmentami wideo,
- $\sum_{k=1}^{K} \left(\frac{d_k(R_k)}{c_k} - B_k \right)$ – jest to łączny czas jaki użytkownik spędził w oczekiwaniu na załadowanie kolejnych fragmentów wideo. Buforowanie zachodzi wtedy, gdy czas pobierania fragmentu wideo zajmuje dłużej niż czas pobieranego fragmentu wideo,
- T_s – początkowe opóźnienie,
- λ, μ, μ_s – to współczynnik istotności (wagi) dla odpowiednio: średniej wariancji jakości fragmentów wideo, łącznego czas buforowania całego wideo i początkowego opóźnienia.

3.2. Zbiory danych i ich charakterystyka

Zbiór wszystkich zapisów parametrów transmisji został podzielony na zbiór testowy i zbiór treningowy. Zbiór testowy jest skomponowany z udostępnionego zbioru HSDPA [19] i generowanych sztucznie danych o transmisji na łączu o stałej przepustowości. Dane generowane sztucznie reprezentują warunki transmisji dla różnych (ale o stałej wartości) przepływności łącza, od 300 kb/s do 4700 kb/s. Ich uwzględnienie w zbiorze danych służy odpowiedzi na pytanie na ile wytrenowany model jest skłonny do dobierania lepszej jakości fragmentów kosztem większej częstotliwości zmian jakości wideo, czy też zapewnienia jak najpłynniejszy obraz i unika pobierania fragmentów wyższej jakości. Zbiór treningowy jest kompozycją różnych źródeł danych, w tym zapisów z HSDPA oraz danych z bazy FCC [21]. Ten zbiór pokrywa się w pełni ze zbiorem danych, które został wykorzystany na potrzeby procesu uczenia modelu Pensieve.

Oba powyższe zbiory danych zostały wykorzystane do analizy własności opracowanego modelu w testowym środowisku symulacyjnym (moduł drugi) i pozwoliły na obserwację zachowania tego modelu w systemie transmisji strumieniowej wideo opartym o zasady DASH. Testy zostały wykonane z wykorzystaniem łącza sieciowego o zestawie parametrów, które były wykorzystywane w procesie uczenia jak i całkowicie nowych (nie wykorzystywanych w procesie uczenia). Dzięki temu uzyskane rezultaty testów są bardziej obiektywne z punktu widzenia jego wykorzystania w



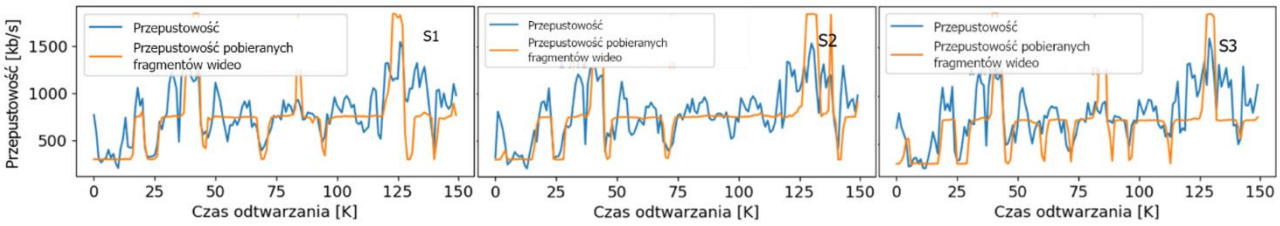
Rysunek 5: Wartość funkcji CDF dla przepustowości w fragmentach wideo. Materiał wideo: „Red Bull Playstreets".

rzeczywistym systemie sieciowych usług strumieniowych. Innymi słowami, takie podejście stawia na jak największą zbieżność symulacji z oczekiwanymi warunkami rzeczywistymi a nie na testowanie warunków krytycznych (specyficznych dla danej technologii transmisji danych i topologii infrastruktury sieciowej). Treści wideo zostały przygotowane w postaci list rozmiarów kolejnych fragmentów wideo w bajtach. Zostały sporządzone oddzielnie dla każdej dostępnej przepływności wraz z plikiem manifestu. Przepływności wideo w użytych zbiorach wideo dobrano tak, by były jak najbliższe przepływnościom zdefiniowanym w zbiorze treningowym oraz zostały one dostosowane do zaprezentowanego wcześniej formatu danych: 300, 750, 1200, 1850, 2850, 4300 Kb/s.

4. Wyniki testów

Na potrzeby zbadania zagadnienia użyteczności uczenia ze wzmocnieniem w generowaniu algorytmów ABR opracowano poszczególne etapy wraz z scenariuszami testowymi:

1. Wytrenowanie modelu.
2. Ewaluacja modelu w emulowanym środowisku dla utworzonego zbioru parametrów łącza transmisyjnego:
   a. scenariusz 1 – dalej nazywany S1, emulacja odtwarzania 10 minut wideo „Big Buck Bunny" [22, 23]– film animowany z dużą ilością statycznych ujęć,
   b. scenariusz 2 – dalej nazywany S2, emulacja odtwarzania 10 minut wideo „Red Bull Playstreets" [22] – film sportowy z dużą ilością dynamicznych ujęć,
   c. scenariusz 3 – dalej nazywany S3, emulacja odtwarzania 10 minut wideo „Valkaama" [22] – film obyczajowy z różnym zakresem ujęć.
3. Analiza statystyczna zebranych pomiarów.



Rysunek 7: Zależność pomiędzy procesem decyzyjnym, a zmianami dostępnej przepustowości łącza transmisyjnego.

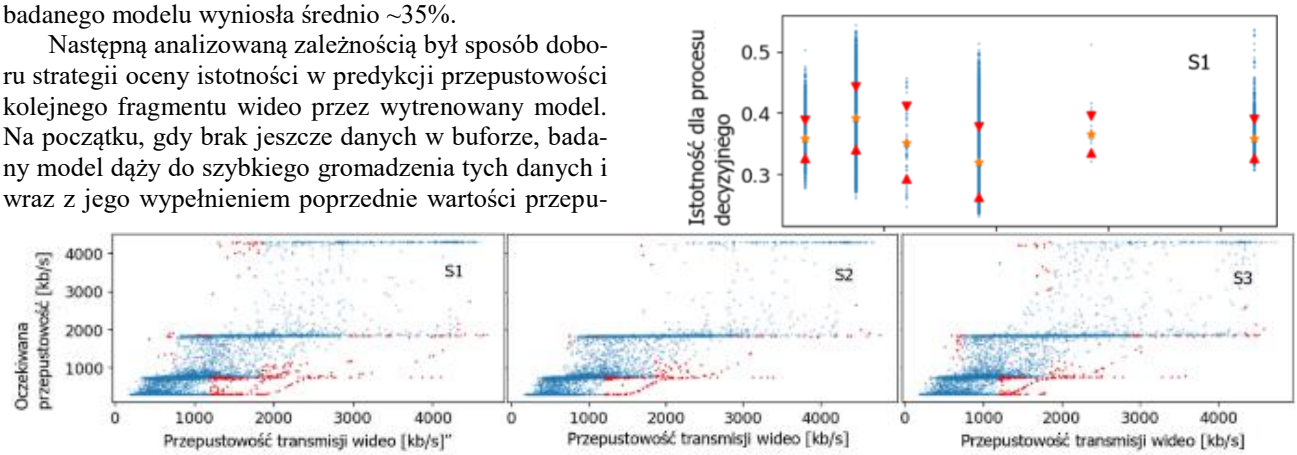
4. Wizualizacja kluczowych metryk na potrzeby interpretacji wyników.

Dla każdego ze scenariuszy emulację środowiska przeprowadzono dla tych samych warunków łącza internetowego. Analiza statystyczna obejmuje badanie zależności między danymi wejściowymi oraz tym jaki mają wpływ na metrykę QoE.

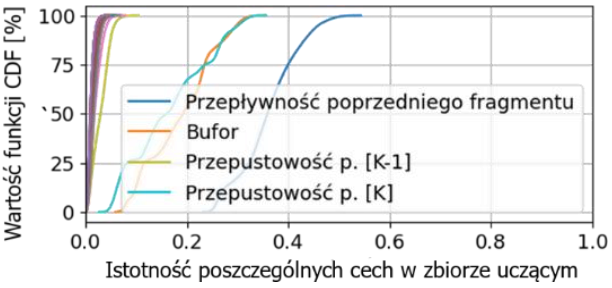
Pierwszą badaną cechą jest rozkład wybieranych przepustowości. Dla wszystkich scenariuszy badawczych S1, S2, S3 uzyskano bardzo podobne rezultaty. Na przykładzie rysunku 5 można zauważyć, że mimo dostępności przepustowości 1200 kb/s, 2850 kb/s badany model unika wyboru fragmentów o tych przepustowościach, co oznacza, że ich wybór jest suboptymalny. Można to potwierdzić na podstawie obserwacji danych na rysunku 7. Przedstawia on wynik analizy istotności przepustowości ostatniego fragmentu i dostępnej przepustowości w trakcie wyboru przepustowości kolejnego fragmentu. Dla przepustowości 1200 kb/s i 2850 kb/s widoczne jest znacznie mniejsze zagęszczenie lub całkowity brak istotnych relacji, natomiast dla pozostałych przepustowości widać zdecydowanie większą istotność relacji. Jednocześnie należy podkreślić, że we wszystkich przeprowadzonych testach zauważono że poprzednia przepustowość fragmentu wideo jest jednym z kluczowych elementów w doborze kolejnego fragmentu tego wideo.

Widoczna jest tendencja, że dla niższych przepustowości (<=2850 kb/s) oczekiwana jest coraz wyższa przepustowość kolejnego fragmentu i ma istotny wpływ na wartości metryki QoE. Natomiast dla wyższych przepustowości (>2850 kb/s) słabnie i oczekiwana przepustowość jest taka sama lub niższa. Ilustruje to rysunek 6. Generalny wniosek ze wszystkich zrealizowanych testów jest taki, że istotność wartości przepływności poprzedniego fragmentu wideo w procesie decyzyjnym badanego modelu wyniosła średnio ~35%.

Następną analizowaną zależnością był sposób doboru strategii oceny istotności w predykcji przepustowości kolejnego fragmentu wideo przez wytrenowany model. Na początku, gdy brak jeszcze danych w buforze, badany model dąży do szybkiego gromadzenia tych danych i wraz z jego wypełnieniem poprzednie wartości przepustowości fragmentów stają się coraz mniej istotne. Krytyczny moment następuje po przekroczeniu zapełnienia objętości bufora treścią wideo o czasie odtwarzania powyżej 20 sekund. Wtedy, ponownie wartość przepustowości poprzednio pobranego fragmentu staje się znaczący w predykcji. Należy jednak zaznaczyć, że ta zaobserwowana istotność jest mniej zależna od zapełnienia bufora i maleje wraz z rosnącą ilością buforowanych danych. Istotna część testów została poświęcona zagadnieniu wpływu zmian w dostępnej przepustowości łącza transmisyjnego na dobór przepływności kolejnych fragmentów wideo. Wyniki przedstawione na rysunku 7 dowodzą, że w trakcie testów zaobserwowano zbliżone działanie badanego modelu dla wszystkich trzech scenariuszy. W przypadkach gdy treść wideo wymagała łącza o dużych wartościach przepływności, model podejmował decyzje maksymalizujące wykorzystanie łącza. Następnie, po osiągnięciu stanu wypełnienia łącza, następowała decyzja o dość gwałtownym zmniejszeniu wymagań odnośnie przepustowości dla kolejnego fragmentu. Ta strategia pozwalała na zachowanie stałych wartości metryki QoE dzięki danym zgromadzonym w buforze. Zaobserwowano też, że przez większość czasu transmisji wideo średnia wartość przepustowości fragmentów wideo wynosiła 750 kb/s. Wartość ta wynika z faktu, że przepustowość łącza w zakresie 0-750 kb/s spełniała zapotrzebowanie dla większości pobieranych fragmentów. Dodatkowo, na podstawie rysunku 7, można stwierdzić, że w przypadku filmu w scenariuszu 2, wykorzystywana była największa przepustowość łącza przy jednocześnie najmniejszej liczbie zmian przepustowości pomiędzy kolejnymi fragmentami wideo. Natomiast, dla scenariusza 3 transmisja wideo miała najmniejsze zapotrzebowanie na dostępną przepustowość łącza ale charakteryzowała się największą fluktuacją przepustowości pomiędzy poszczególnymi, po-



Rysunek 8: Relacja pomiędzy przepustowością pobieranych fragmentów wideo a oczekiwaną wartością przepustowości transmisji.



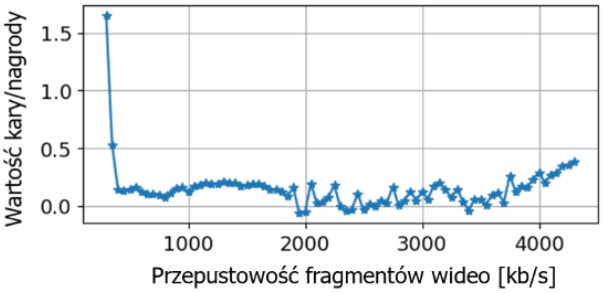
Rysunek 9: Wartość funkcji CDF dla istotności dla średniej wartości metryki QoE w relacji do poszczególnych cech zbioru uczącego. Symbol K oznacza k-tą decyzję modelu. Materiał wideo: „Big Buck Bunny".

bieranymi fragmentami.

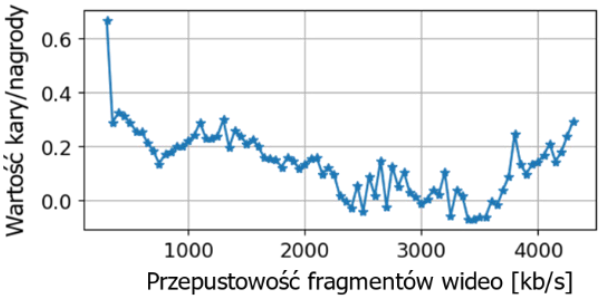
Omówione wyżej zależności zostały dodatkowo przeanalizowane pod kątem relacji pomiędzy przepustowością pobieranych fragmentów a oczekiwaną wartością przepustowości transmisji. Relacje te dla wszystkich trzech scenariuszy badawczych przedstawia rysunek 8. Na tym rysunku, kolorem niebieskim oznaczono relacje optymalne dla chwilowej wartości dostępnej przepustowości łącza a kolorem czerwonym, oznaczano przeciwną sytuację. Dla wszystkich scenariuszy, w zdecydowanej większości przypadków, model dokonywał doboru optymalnej przepustowości. Jednakże widoczne są też różnice między scenariuszami, co wynika z cech charakterystycznych transmitowanej treści wideo. Dla scenariuszy S1 i S2 w zakresie mniejszych, oczekiwanej przepływności można zaobserwować większe zagęszczenie sub-optymalnych wyborów. Jak wspomniano wyżej, może to wynikać z charakteru wideo ponieważ scenariusz 1 oraz 2 obejmują odpowiednio, kategorie filmów animowanych i sportowych. Na podstawie wyników analizy, które zostały zebrane na rysunku 9, można stwierdzi, że kluczowymi elementami w ocenie wyboru przepływności kolejnego fragmentu wideo są:

- przepływność poprzedniego fragmentu wideo,
- dostępna przepustowość łącza,
- stan zapełnienia bufora.

Ten wniosek dodatkowo potwierdza przydatność definicji metryki QoE [12], w której powyższe parametry są głównymi składnikami. Należy się zastanowić czy parametr – stan zapełnienia bufora nie jest zbyt dużym uproszczeniem dla rzeczywistej zmienności treści wideo. Opracowane rozwiązania monitorują jedynie jego stan zapełnienia a nie z jaką dynamiką zmienia się jego objętość. Zakładając, że różnica w przyrostach objętości



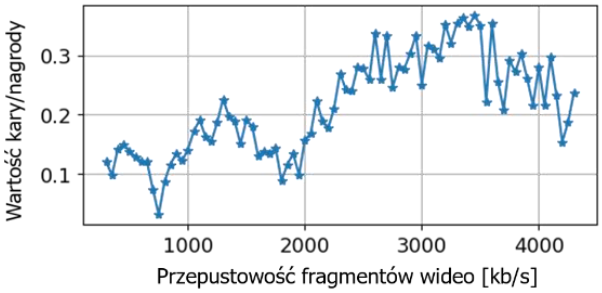
Rysunek 10: Relacja pomiędzy wartościami kary/nagrody związanej z wartością metryki QoE a przepustowością fragmentów wideo. Materiał wideo: „Valkaama".



Rysunek 11: Relacja pomiędzy wartościami kary/nagrody związanej z wartością metryki QoE a przepustowością fragmentów wideo. Materiał wideo: „Big Buck Bunny".

może być zależna od dynamiki wideo, świadczy to o pominięciu tej zależności w dotychczasowych rozwiązaniach, jak i braku jej uwzględnienia w badanym modelu. Stanowi to istotny wniosek prze definiowaniu kolejnych etapów rozwoju tego modelu.

Ostatnim analizowanym aspektem działania wytrenowanego modelu była jego zdolność do adaptacji w sytuacji sterowania transmisją zróżnicowanych materiałów wideo. Wyniki dla tej części badań są przedstawione na rysunkach 10, 11 oraz 12. Można zaobserwować, że wytrenowany model, którego proces uczenia oparty był jedynie na pojedynczym zbiorze danych wideo, próbuje dostosować się do nowych warunków środowiska (do materiałów o zdecydowanie innych cechach, np. dynamiki zmian przepustowości). Dla tych samych parametrów łącza sieciowego, ale dla różnych kategorii wideo otrzymano znacząco różne wyniki testów. Można zatem wysunąć wniosek, że trenowanie modelu na tylko pojedynczym zbiorze wideo jest istotnym uproszcze-



Rysunek 12: Relacja pomiędzy wartościami kary/nagrody związanej z wartością metryki QoE a przepustowością fragmentów wideo. Materiał wideo: „Red Bull Playstreets".

niem. Stanowi to cenną informację, że w przypadku przyszłych działań zmierzających do udoskonalenia modelu należałoby poszerzyć proces uczenia o treści wideo jak najbardziej zróżnicowanych parametrach (o różne kategorie wideo). Przedstawione wyżej relacje prowadzą do jeszcze jednej, istotnej obserwacji. Generalizacja znajomości własności środowiska jaką posiadł badany model na podstawie zbioru uczącego o niewystarczająco zróżnicowanych parametrach wideo prowadzi do zachowawczych decyzji odnośnie doboru przepływności kolejnego fragmentu wideo.

5. Podsumowanie

Idea realizacji algorytmów ABR z wykorzystaniem technik jak uczenie ze wzmocnieniem może być trakto-

wana jako istotna alternatywa dla rozwiązań klasycznych. W trakcie wykonanych testów potwierdzono, żetego typu rozwiązania mogą być w prosty sposób implementowane w klientach DASH, takich jak chociażby referencyjny klient dash.js. Algorytmy z rodziny Pensieve, w swojej pierwotnej postaci, opierają się jedynie o dane o przepustowości łącza, rozmiary pobieranych fragmentów wideo oraz informacje o dostępnych poziomach jakości. Zaprezentowane wyniki badań zależności wpływu różnych kategorii treści wideo odbieranej przez klienta DASH na jakościowe parametry wchodzące w skład metryki QoE wykazano ich silne, wzajemne relacje. W praktyce oznacza to, że w zależności na jakim zestawie danych wideo model został wytrenowany, będzie w konsekwencji inaczej radził sobie dla treści wideo innej kategorii. Na podstawie uzyskanych rezultatów zauważono, że są to subtelne różnice w wartościach metryki QoE pomiędzy takimi kategoriami wideo jak film animowany, akcji i sport. Jako przyszły, niezwykle istotny kierunek badań zaproponowanego modelu, należy wskazać testy wzajemnego wpływ kilku klientów DASH uruchomionych w tym samym segmencie sieci. W takim przypadku scenariusze testowe należałoby rozbudować o dodatkowy element pozwalający na ewaluację wpływu wzajemnej konkurencji o tą samą pulę dostępnej przepustowości.

Literatura

[1] Cisco Visual Networking Index: Forecast and Methodology 2016-2021, High Efficiency Video Coding (HEVC) Algorithms and Architectures (2017).

[2] K. u. R. Laghari, O. Issa, F. Speranza, T. H. Falk, Quality-of-Experience perception for video streaming services: Preliminary subjective and objective results, Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference, Hollywood, CA, USA, (2012) 1-9.

[3] A. Reuban, MPEG-DASH Enhanced Multimedia Streaming, International Journal of Advanced Research in Computer Science and Software Engineering, 4 (2014) 848-851.

[4] D. You, S. -H. Kim, D. H. Kim, ATSC 3.0 ROUTE/DASH Signaling for Immersive Media: New Perspectives and Examples, in IEEE Access, 9 (2021) 164503-164509, https://dx.doi.org/10.1109/ACCESS.2021.3133626.

[5] I. Sodagar, The MPEG-DASH Standard for Multimedia Streaming Over the Internet, IEEE MultiMedia, 18 (2011) 62-67, https://doi.org/10.1109/MMUL.2011.71.

[6] O. Izima, R. de Fréin, A. Malik, A Survey of Machine Learning Techniques for Video Quality Prediction from Quality of Delivery Metrics, Electronics, 10 (2021) 2851, https://dx.doi.org/10.3390/electronics10222851.

[7] T-Y. Huang, R. Johari, N. McKeown, M. Trunnell, W. Mark, A buffer-based approach to rate adaptation: Evidence from a Large Video Streaming Service, SIGCOMM Computer Communication Review, New York, NY, USA, 44 (2014) 187-198, https://dx.doi.org/10.1145/2619239.2626296.

[8] X. Yin, A. Jindal, V. Sekar, B. Sinopoli, A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP, SIGCOMM Computer Communication Review, New York, NY, USA, 45 (2015) 325-338, https://doi.org/10.1145/2829988.2787486.

[9] K. Spiteri, R. Sitaraman, D. Sparacio, From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player, ACM Transactions on Multimedia Computing, Communications, and Applications, New York, NY, USA, 15 (2019) 1-29, https://doi.org/10.1145/3336497.

[10] M. Otterlo, M. Wiering, Reinforcement Learning and Markov Decision Processes, Reinforcement Learning: State of the Art, (2012) 3-42, https://doi.org/10.1145/2829988.2787486.

[11] S. Bhatt, Reinforcement Learning 101, Learn the essentials of Reinforcement Learning!, https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292, [03.11.2022].

[12] H. Mao, R. Netravali, M. Alizadeh, Neural Adaptive Video Streaming with Pensieve, Association for Computing Machinery, Los Angeles, CA, USA, (2017) 197–210, https://doi.org/10.1145/3098822.3098843.

[13] I. Grondman, L. Busoniu, G. Lopes, R. Babuska, A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients, IEEE Transactions on Systems, Man and Cybernetics Part B-Cybernetics, 42 (2012) 1291-1307, https://doi.org/10.1109/TSMCC.2012.2218595

[14] H. -C. Jang, Y. -C. Huang, H. -A. Chiu, A Study on the Effectiveness of A2C and A3C Reinforcement Learning in Parking Space Search in Urban Areas Problem, International Conference on Information and Communication Technology Convergence, Jeju, South Korea, (2020) 567-571, https://doi.org/10.1109/ICTC49870.2020.9289269.

[15] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, 61 (2015) 85-117, https://doi.org/10.1016/j.neunet.2014.09.003.

[16] Z. Tian, L. Zhao, L. Nie, P. Chen, S. Chen, Deeplive: QoE Optimization for Live Video Streaming through Deep Reinforcement Learning, IEEE 25th International Conference on Parallel and Distributed Systems, Tianjin, China, (2019) 827-831, https://doi.org/10.1109/ICPADS47876.2019.00122.

[17] Y. Liu, D. Wei, C. Zhang, W. Li, Distributed Bandwidth Allocation Strategy for QoE Fairness of Multiple Video Streams in Bottleneck Links, Future Internet, 14 (2022) 152, https://doi.org/10.3390/fi14050152.

[18] A. Dethise, M. Canini, S. Kandula, Cracking Open the Black Box: What Observations Can Tell Us About Reinforcement Learning Agents, Association for Computing Machinery, New York, NY, USA, (2019) 29-36, https://doi.org/10.1145/3341216.3342210.

[19] H. Riiser, P. Vigmostad, C. Griwodz, P. Halvorsen, Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications, Association for Computing Machinery, New York, NY, USA, (2013) 114-118, https://doi.org/10.1145/2483977.2483991.

[20] M. Ribeiro, S. Singh, C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier, Association for Computational Linguistics: Demonstrations, San Diego, California, (2016) 97-101, http://dx.doi.org/10.18653/v1/N16-3020.

[21] Raw Data - Measuring Broadband America 2016, Federal Communications Commission, https://www.fcc.gov/reports-research/reports/measuring-broadband-america/, [03.11.2022].

[22] ITEC - Dynamic Adaptive Streaming over HTTP, https://dash.itec.aau.at/contact/, [03.11.2022].

[23] C. Müller, S. Lederer, C. Timmerer, H. Hellwagner, Dynamic Adaptive Streaming over HTTP/2.0, IEEE International Conference on Multimedia and Expo, (2013) 1-6, http://dx.doi.org/10.1109/ICME.2013.6607498.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 171–177
Received: 13 April 2023
Accepted: 15 May 2023

Comparative analysis of data reading performance from the Salesforce platform using GraphQL, REST and SOAP interfaces

Analiza porównawcza wydajności odczytu danych z platformy Salesforce przy wykorzystaniu interfejsów GraphQL, REST oraz SOAP

Ryszard Roman Rogalski*

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

Abstract
The article describes a comparative analysis of data reading from tables in the Salesforce environment using three different application programming interfaces. The popularity of the Salesforce platform and the release of the GraphQL interface on it on October 5, 2022 were an inspiration to perform the research. Based on the literature reviewed, there was no such study for the Salesforce platform. The performance of reading data from the Salesforce platform was investigated using an automation script. For four tables containing a different number of rows, 8 types of queries were repeatedly executed using each of the three interfaces. It was found that depending on the number of rows, either REST API or SOAP API should be considered. In all cases, the lowest performance was observed while using GraphQL API.

Keywords: Salesforce; performance; GraphQL; API

Streszczenie
Artykuł opisuje analizę porównawczą odczytu danych z tabel w środowisku Salesforce przy pomocy trzech różnych interfejsów programowania aplikacji. Popularność platformy Salesforce oraz udostępnienie na niej interfejsu GraphQL dnia 5 października 2022 roku były inspiracją do przeprowadzenia badań. Na podstawie przeanalizowanej literatury stwierdzono brak takiego badania dla platformy Salesforce. Zbadano wydajność odczytu danych z platformy Salesforce przy pomocy skryptu automatyzującego. Dla czterech tabel zawierających inną liczbę wierszy wielokrotnie wykonano 8 rodzajów zapytań przy pomocy każdego z trzech interfejsów. Stwierdzono, że zależnie od liczby wierszy należy rozważyć użycie REST API lub SOAP API. We wszystkich przypadkach najniższą wydajność zaobserwowano podczas zastosowania GraphQL API.

Słowa kluczowe: Salesforce; wydajność; GraphQL; API

*Corresponding author
Email address: ryszard.rogalski@pollub.edu.pl (R. Rogalski)

1. Wstęp

Branża informatyczna wykorzystuje tradycyjne relacyjne bazy danych od około 40 lat. Jednak w ostatnich latach nastąpiła znaczna przemiana branży IT w zakresie aplikacji komercyjnych. Samodzielne aplikacje zostały zastąpione rozwiązaniami serwerowymi. Niższe opłaty, elastyczność oraz model naliczania kosztu usługi dopiero po jej wykonaniu to główne przyczyny, które spowodowały, że obliczenia rozproszone stały się rzeczywistością [1].

Database-as-a-Service (DBaaS) jest usługą, która zapewnia bazę danych lub dedykowaną instancję do zarządzania danymi w chmurze, bez ponoszenia kosztów czasowych związanych z zarządzaniem infrastrukturą, udostępnianiem sprzętu, konfiguracją silnika, aktualizacjami i kopiami zapasowymi. Baza danych jest hostowana w zdalnym centrum danych i może być współdzielona przez użytkowników w przejrzysty sposób. Usługa taka obejmuje na przykład tworzenie tabel lub ładowanie i dostęp do danych w tabelach. Aby zaoferować swobodny dostęp do bazy danych, tego typu rozwiązania często dostarczają interfejsy programistyczne i wykonują operacje bazodanowe za ich pośrednictwem [2].

Platforma Salesforce dostarcza wiele interfejsów programistycznych do komunikacji z bazą danych [3], lecz zdecydowano się dokonać analizy wydajności trzech interfejsów, które są powszechnie używane w innych systemach. Dokonano analizy tej platformy ze względu na jej udział w rynku i rosnącą popularność [4]. W badaniu przeprowadzono testy wydajności zapytań odczytujących dane ze względu na to, że to jedyna dostępna obecnie operacja bazodanowa udostępniona w interfejsie GraphQL (*Graph Query Language*) na platformie Salesforce [5].

2. Interfejs programowania aplikacji GraphQL

GraphQL to propozycja alternatywnego języka zapytań i silnika wykonywania dla interfejsów programowania aplikacji internetowych, który ma na celu rozwiązanie problemów z dostępem do danych i wersjonowaniem interfejsów [6]. Jest to język przeznaczony dla interfejsów wykorzystywanych głównie na stronach internetowych. Zaproponowany przez firmę Facebook w 2016 roku, stanowi alternatywę dla interfejsów takich

jak REST lub SOAP [7]. Oficjalna specyfikacja określa GraphQL jako język zapytań i silnik wykonawczy, który służy do opisu możliwości i wymagań modelu danych dla aplikacji klient-serwer [8].

3. Przegląd literatury

Uwzględniona literatura przedstawia analizę porównawczą trzech interfejsów programowania aplikacji: SOAP, REST oraz GraphQL. Nie przedstawia ona jednak analizy tych interfejsów na platformie Salesforce. Autorzy artykułu [7] przeprowadzili badanie przy pomocy trzech języków programowania: C#, Java oraz PHP. W pracy niestety nie sprecyzowano rodzajów wykonywanych zapytań. Autorzy dla 50 wirtualnych użytkowników wykonujących zapytania jednocześnie uzyskali wyniki przedstawione w Tabeli 1.

Tabela 1: Średni czas odpowiedzi dla zapytania wykonanego przy użyciu danego języka oraz interfejsu [7]

Język	Interfejs	Średni czas odpowiedzi (s)
C#	SOAP	74000
Java	SOAP	127600
PHP	SOAP	107800
C#	REST	56800
Java	REST	107400
PHP	REST	102800
C#	GraphQL	32200
Java	GraphQL	54000
PHP	GraphQL	48800

Jednak wyniki świadczyły, że niezależnie od wykorzystanego języka programistycznego to interfejs GraphQL okazał się najszybszy.

Autorzy kolejnej pracy [9] podkreślają, że dyskusje na temat najbardziej optymalnego rozwiązania dotyczącego tworzenia usług sieciowych wystawiających API nie są rozstrzygnięte. Autorzy przeprowadzili test, w którym wykonano 500 zapytań w czasie 5 sekund, uzyskując średni czas wykonania jednego zapytania na poziomie 0,01 sekundy. Wyniki pokazały, że w przypadku użycia GraphQL, wraz ze wzrostem liczby kolejnych zapytań czas odpowiedzi znacznie się wydłużał.

1. REST okazał się wydajniejszy w przypadku pobierania wszystkich zasobów z bazy.
2. GraphQL był efektywniejszy w przypadku pobierania wyszczególnionych danych.

Erlandsson oraz Remes [10], również przeprowadzili porównanie wydajności tych trzech interfejsów. Postawili oni dwie hipotezy. Pierwsza zakładała, że interfejs GraphQL, będzie wydajniejszy tylko, gdy liczba wierszy w tabeli będzie dostatecznie duża oraz wybierane będą konkretne wiersze. Wyniki świadczyły o tym, że GraphQL wypada najgorzej, niezależnie od liczby wierszy w tabeli lub liczby wybieranych wierszy. Druga hipoteza zakładała, że to interfejs SOAP będzie osiągał najgorsze wyniki, niezależnie od przypadku testowego. Okazało się ponownie, że to GraphQL był najmniej wydajny podczas prowadzonych testów.

Każda praca przedstawia zupełnie inne wyniki, co świadczy o tym, że zależnie od środowiska, na którym prowadzone są badania, można uzyskać odmienne rezultaty. Podczas analizy literatury nie znaleziono testów przeprowadzonych w środowisku Salesforce.

4. Metoda badawcza

4.1. Przygotowanie danych i wybór środowiska

Przed wykonaniem testów wydajnościowych przygotowano tabele w bazie danych. Utworzono cztery tabele zawierające identyczny model danych. Dla każdej z tabel zostało utworzone 8 kolumn o następujących typach:

1. Text (pole indeksowane, unikalne).
2. Number.
3. Currency.
4. Date.
5. Date/Time.
6. Checkbox
7. Picklist.
8. Multi-Select Picklist.

Dla każdej z czterech tabel utworzono następującą liczbę wierszy: 100, 1000, 10 000 oraz 500 000. Wartości kolumn w wierszach zostały wygenerowane pseudolosowo. Listing 1 przedstawia kod napisany w języku *Apex* odpowiedzialny za przypisywanie wartości kolumnom.

Listing 1: Kod do generowania danych psuedolosowych

```
/* pseudolosowa wartość True/False*/
Boolean__c = Math.random() < 0.5,
/* pseudolosowa data*/
Date__c = Date.newInstance(
              (Integer) Math.floor
               (Math.random() * 53) + 1970,
               (Integer) Math.floor
               (Math.random() * 12) + 1,
              (Integer) Math.floor
               (Math.random() * 31) + 1
),
/* pseudolosowa wartość Currency*/
Currency__c = (Integer)  Math.floor(
       Math.random() * 100000) + 1 + 0.5,
/* pseudolosowa wartość Number*/
Number__c = (Integer) Math.floor(
       Math.random() * 99999) + 1
```

Ze względu na ograniczenia platformy Salesforce, kod który generował 500 000 wierszy dla ostatniej tabeli został umieszczony w klasie implementującej interfejs *Batchable* (Listing 2), umożliwiający operowanie na większej liczbie rekordów.

Listing 2: Implementacja interfejsu *Batchable* w Salesforce

```
public without sharing class
    BatchHelp implements Database.Batchable<SObject>{
/* klasa implementująca interfejs Batchable (Salesforce) */

public void execute(Database.BatchableContext info,
       List<Account> scope){
    /*metoda wykonawcza w interfejsie */

    String query = 'SELECT COUNT() FROM Table_500000__c';

    Integer records = Database.countQuery(query);
    /*sprawdzenie aktualnej liczby wierszy w tabeli */
}
```

Utworzona została również piąta tabela, w której utworzono kolumny zawierające referencje do poprzednich tabel. Utworzono wiersze w tej tabeli zachowując integralność referencyjną (Listing 3).

Listing 3: Tworzenie wierszy zachowując integralność referencyjną

```
List<Relationship_Table__c> relation =
        new List<Relationship_Table__c>();
/*zapewnienie integralności referencyjnej w nowych wierszach*/
for(Table_500000__c x : insertsDML){
    relation.add(new Relationship_Table__c(
            Table_500000__c = x.Id
    ));
insert relation;
/*wykonanie operacji DML dodania wierszy do tabeli*/
```

Podczas badania wykorzystano odpowiednią wersję platformy Salesforce, umożliwiającą przechowywanie takiej ilości danych. Przykładowo wersja deweloperska oferuje tylko 5 megabajtów pamięci do przechowywania danych. W Tabeli 2 przedstawiono parametry użytej instancji Salesforce.

Tabela 2: Parametry wykorzystanej instancji

Parametr	Wartość
Edycja	Enterprise
Instancja	EU46
Lokalizacja	Frankfurt, Niemcy
Wersja	Spring 23

4.2. Przebieg badania

Badanie zostało przeprowadzone przy pomocy skryptu automatyzującego. Skrypt został napisany w języku *Python* (wersja 3.9.13). Podczas zliczania rezultatów upewniono się, że jedyną wartością mierzoną jest czas odpowiedzi serwera na żądanie oraz czas przetworzenia zapytania POST, a przypisania zmiennych oraz wszelkie inne instrukcje zostały wyniesione poza obszar kodu, w którym dokonywano pomiaru (Listing 4). Zdecydowano się na symulację komunikacji serwera zewnętrznego z platformą Salesforce i wysyłano jedno żądanie jednocześnie.

Listing 4: Pomiar wykonania zapytania przy pomocy protokołu HTTP

```
start_time = time.perf_counter()
#rozpoczecie pomiaru czasu
response = requests.post(instance_url,
                         headers=headers,
                         json=json_variable)
#wykonanie zapytania HTTP POST
end_time = time.perf_counter()
#zakonczenie pomiaru czasu
```

Konfiguracja sprzętowa komputera, na którym został uruchomiony skrypt została przedstawiona w Tabeli 3.

Tabela 3: Parametry sprzętowe komputera

Parametr	Wartość
Pamięć RAM	16 GB
Procesor	Apple M1 (8 rdzeni)
Dysk	SSD 256 GB
Częstotliwość zegara	2064 - 3220 MHz
Architektura	ARM
System operacyjny	macOS Ventura 13.2.1

W ramach eksperymentu zrealizowano następujące scenariusze badawcze:

1. Zmierzenie czasu odpowiedzi serwera na żądanie wierszy z wyszczególnieniem wszystkich kolumn.
2. Zmierzenie czasu odpowiedzi serwera na żądanie wierszy z wyszczególnieniem tylko identyfikatora wiersza.
3. Zmierzenie czasu odpowiedzi serwera na żądanie konkretnego wiersza opisanego przy pomocy kolumny nieindeksowanej.
4. Zmierzenie czasu odpowiedzi serwera na żądanie konkretnego wiersza opisanego przy pomocy kolumny indeksowanej.
5. Zmierzenie czasu odpowiedzi serwera na żądanie wierszy z zastosowaniem sortowania według kolumny nieindeksowanej.
6. Zmierzenie czasu odpowiedzi serwera na żądanie wierszy z zastosowaniem sortowania według kolumny indeksowanej.
7. Zmierzenie czasu odpowiedzi serwera na żądanie wierszy z wartością w kolumnie typu *Currency* większą niż zadana.
8. Zmierzenie czasu odpowiedzi serwera na żądanie wierszy z zastosowaniem podzapytania w klauzuli *WHERE*.

W scenariuszach opisujących wybranie wierszy, dla tabel o liczbie wierszy większej niż 200 wykonano serie zapytań, zwracających po 200 wierszy każde. Odpowiedzi interfejsów na platformie Salesforce w takich przypadkach zwracają adresy dla których powinno wykonane zostać kolejne zapytanie zwracające kolejne 200 wierszy (Listing 5). W ten sam sposób działa zarówno SOAP, REST oraz GraphQL API (na platformie Salesforce). Ze względu na ograniczenia występujące na platformie Salesforce, dla tabel zawierających 10 000 oraz 500 000 wierszy wybrano ich jedynie 4000. Dla pozostałych tabel wybrano wszystkie wiersze. Ograniczenie do 4000 wierszy wynika ze sposobu implementacji interfejsu GraphQL na platformie Salesforce, czyli limitem maksymalnej wartości (2000) opcjonalnych klauzul *LIMIT* oraz *OFFSET* [10]. Operacje wysłania opisanych żądań HTTP powtórzono 19 razy dla trzech interfejsów oraz czterech rozmiarów tabel, zatem przeprowadzono 228 prób.

Listing 5: Uzyskanie adresu kursora ze zwróconej odpowiedzi

```
while hasNextPage:
    if response.status_code == 200:
        #sprawdzenie statusu odpowiedzi serwera

        data = response.json()['data']['uiapi']
        ['query'][TABLE_NAME]
        #transformacja zwroconego tekstu do postaci JSON

        totalCount = data['totalCount']
        pageInfo = data['pageInfo']
        cursor = data['pageInfo']['endCursor']
        #uzyksanie informacji o zwroconym kursorze,

        hasNextPage = pageInfo['hasNextPage']
```

W Tabeli 4 przedstawiono zapytania przekazywane w żądaniu podczas realizacji scenariuszy badawczych.

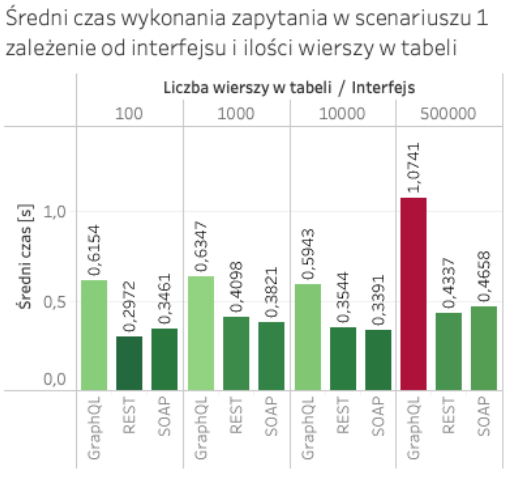
Zapytania w tabeli przedstawione napisane są języku SOQL (*Salesforce Object Query Language*). SOQL jest podobny do instrukcji SELECT w powszechnie używanym języku SQL, ale został zaprojektowany specjalnie dla danych Salesforce.

Tabela 4: Zapytania przekazywane w żądaniu w danym scenariuszu badawczym

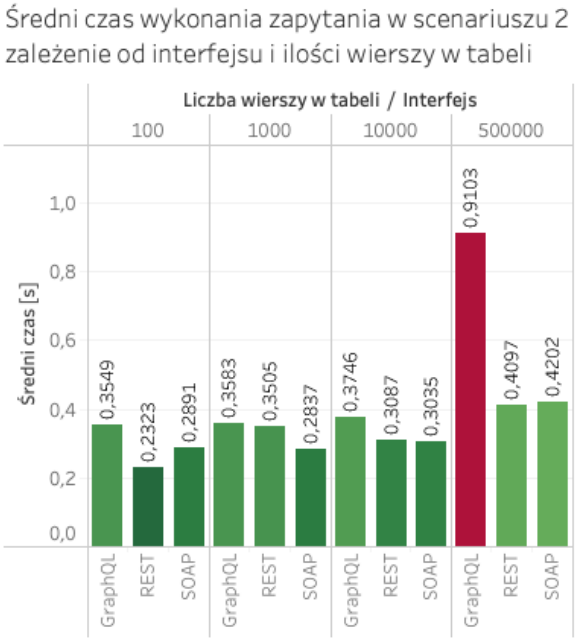
Scenariusz badawczy	Zapytanie SOQL
1	SELECT Id, Boolean__c, Currency__c, Date__c, Date_Time__c, Index__c, Multi_Pick_List__c, Number__c, Picklist__c FROM Table__c
2	SELECT Id FROM Table__c
3	SELECT Id FROM Table__c WHERE Currency__c = 0.2022
4	SELECT Id FROM Table__c WHERE Index__c = '1'
5	SELECT Id, Boolean__c, Currency__c, Date__c, Date_Time__c, Index__c, Multi_Pick_List__c, Number__c, Picklist__c FROM Table__c ORDER By Currency__c ASC
6	SELECT Id, Boolean__c, Currency__c, Date__c, Date_Time__c, Index__c, Multi_Pick_List__c, Number__c, Picklist__c FROM Table__c ORDER By Index__c ASC
7	SELECT Id, Boolean__c, Currency__c, Date__c, Date_Time__c, Index__c, Multi_Pick_List__c, Number__c, Picklist__c FROM Table__c WHERE Currency__c > 10
8	SELECT Id, Boolean__c, Currency__c, Date__c, Date_Time__c, Index__c, Multi_Pick_List__c, Number__c, Picklist__c FROM Table__c WHERE Id IN (SELECT Table__c FROM Relationship_Table__c)

5. Wyniki

Rysunki 1–2 przedstawiają średni czas odpowiedzi serwera zależenie od użytego interfejsu i liczby wierszy w odpytywanej tabeli.

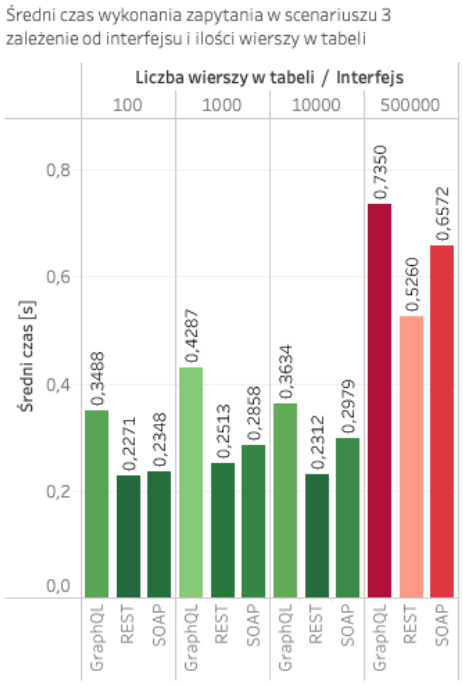


Rysunek 1: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 1.

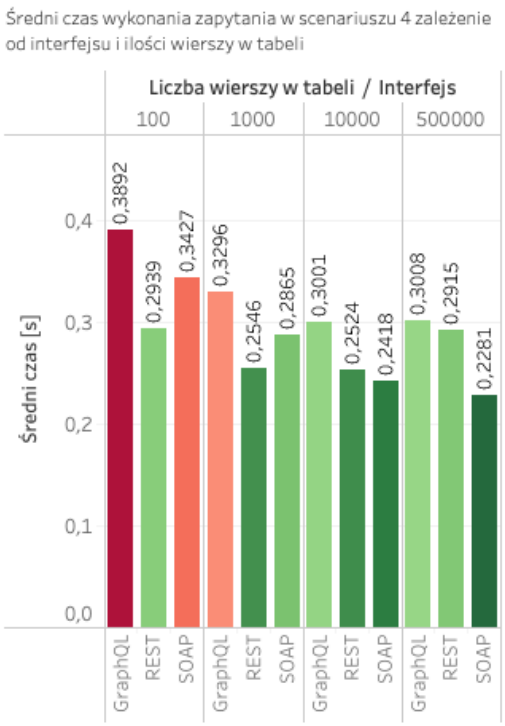


Rysunek 2: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 2.

Otrzymane rezultaty świadczą o tym, że zależnie od liczby wierszy najwydajniejsze są interfejsy REST oraz SOAP, a najmniej wydajny GraphQL, natomiast wraz ze wzrostem liczby wierszy w tabelach interfejs GraphQL staje się coraz wolniejszy w porównaniu do pozostałych interfejsów. Rysunki 3–4 przedstawiają średni czas odpowiedzi serwera zależenie od użytego interfejsu i liczby wierszy w odpytywanej tabeli, lecz dla jednego wybieranego wiersza. Wiersz wybierano przy pomocy kolumny indeksowanej oraz kolumny nieindeksowanej.

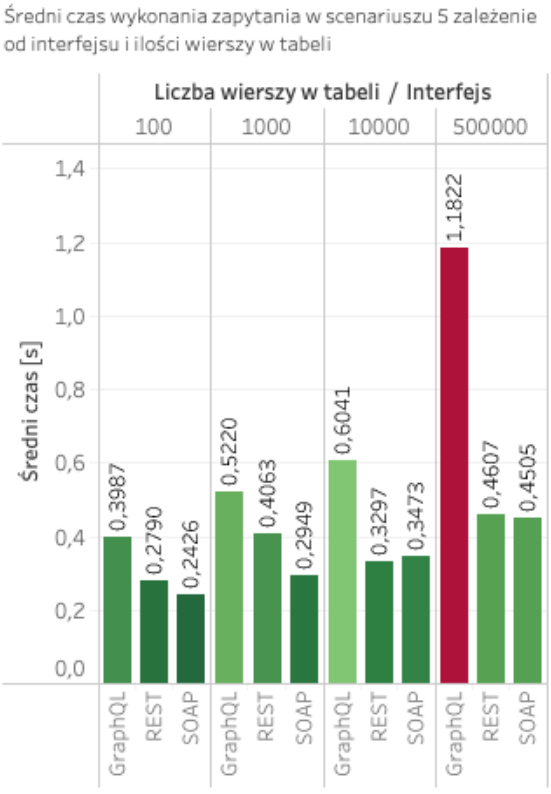


Rysunek 3: Średni czas odpowiedzi serwera w scenariuszu numer 3.

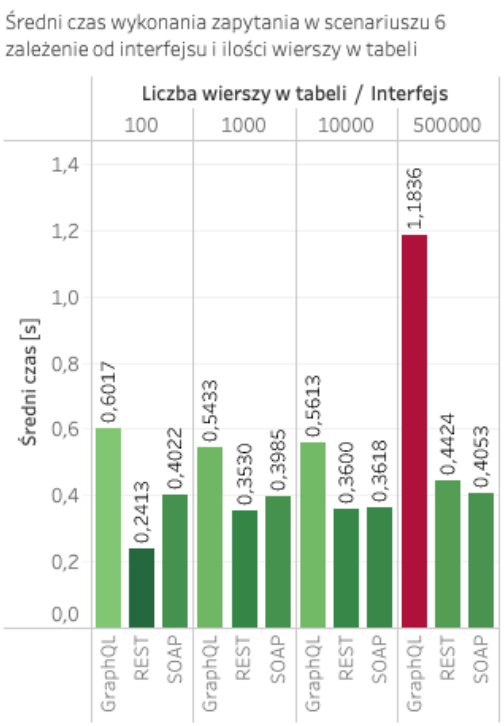


Rysunek 4: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 4.

Rysunki 5–6 przedstawiają średni czas odpowiedzi serwera zależenie od użytego interfejsu i liczby wierszy w odpytywanej tabeli. Zapytania w tym wypadku zostały posortowane przy pomocy kolumny indeksowanej oraz kolumny nieindeksowanej.

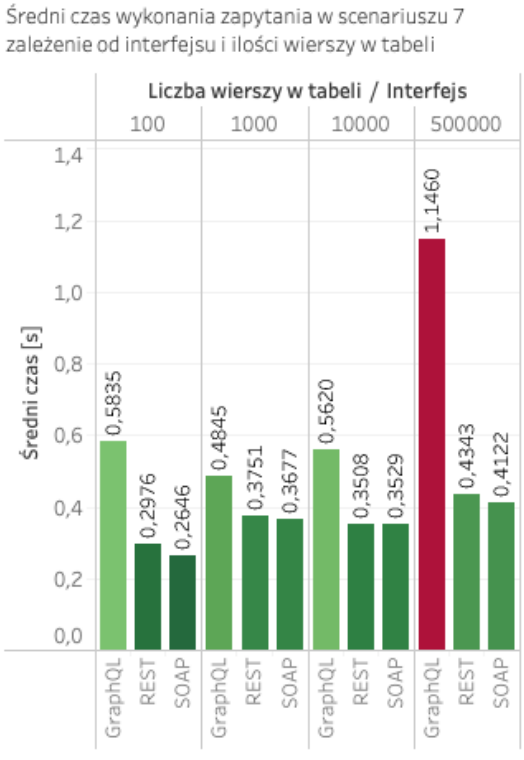


Rysunek 5: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 5.

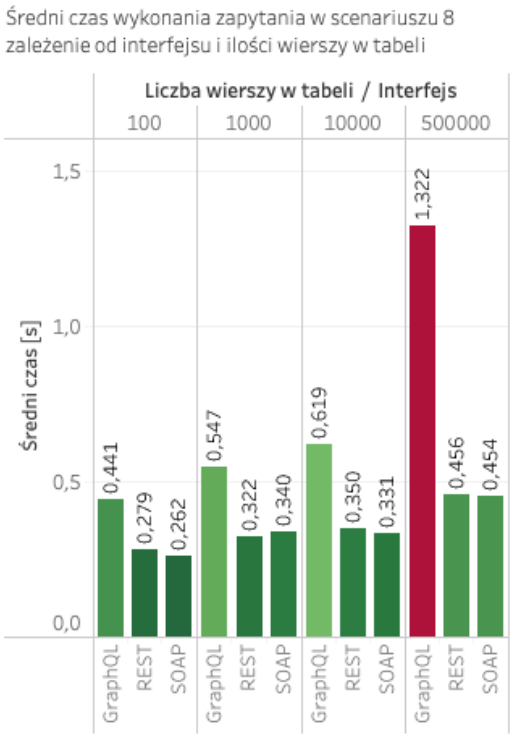


Rysunek 6: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 6.

Rysunki 7–8 przedstawiają średni czas odpowiedzi serwera zależenie od użytego interfejsu i liczby wierszy w odpytywanej tabeli, lecz w zapytaniu zostało zastosowanie filtrowanie. Zastosowano filtrowanie po polu typu *Currency* oraz filtrowanie z użyciem przy podzapytania w klauzuli *WHERE*.

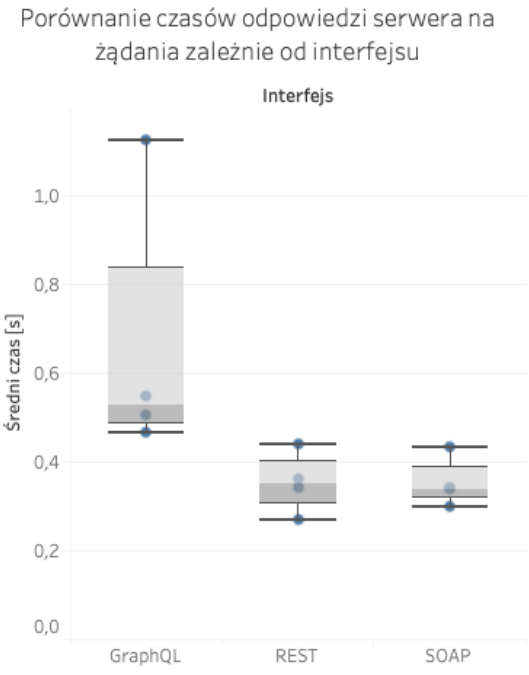


Rysunek 7: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 7.



Rysunek 8: Średni czas odpowiedzi serwera w scenariuszu badawczym numer 8.

Rysunek 9 przedstawia porównanie czasów odpowiedzi serwera we wszystkich przeprowadzonych eksperymentach.



Rysunek 9: Porównanie średnich czasów odpowiedzi serwera we wszystkich scenariuszach badawczych.

W Tabeli 5 przedstawiono szczegółowe wyniki dotyczące zależności interfejsu oraz liczby wierszy w tabeli. Podsumowano wszystkie przeprowadzone scenariusze badawcze oraz przedstawiono średni czas odpowiedzi jak i odchylenie standardowe.

Tabela 5: Szczegółowe wyniki wszystkich przeprowadzonych scenariuszy

Interfejs	Liczba wierszy w tabeli	Średni czas odpowiedzi (s)	Odchylenie standardowe czasu odpowiedzi (s)
GraphQL	100	0,47	0,52
	1000	0,51	0,39
	10000	0,55	0,42
	500000	1,13	0,46
REST	100	0,27	0,16
	1000	0,36	0,38
	10000	0,34	0,37
	500000	0,44	0,56
SOAP	100	0,30	0,31
	1000	0,34	0,32
	10000	0,34	0,36
	500000	0,43	0,59

6. Analiza wyników i wnioski

Różnica pomiędzy średnimi wynikami uzyskanymi przy pomocy interfejsów REST oraz SOAP jest na tyle znikoma (biorąc pod uwagę czynniki takie jak obciążenie serwera oraz obciążenie sieci), że nie stwierdza się różnicy w wydajności działania dla większości przypadków. Wyjątkiem jest sytuacja, gdy liczba rekordów w tabeli jest istotnie niewielka (poniżej tysiąca) - wtedy średnio o 11% szybsze jest REST API.

Wyniki świadczą o tym, że niezależnie od wybranego scenariusza badawczego, a więc liczby oraz typu kolumn, filtrowania czy liczby wierszy sprecyzowanych w zapytaniu najmniej wydajne jest GraphQL API. Należy również zwrócić uwagę, że zwiększenie liczby wierszy w odpytywanej tabeli drastycznie wpływa na czas wykonywania zapytania dla interfejsu GraphQL. Pomiędzy liczbą wierszy 10 000, a 500 000 dla interfejsów REST oraz SOAP, zauważono średnią różnice ok. 29%, a dla GraphQL wynosi ona już 105%. Uzyskano średnie odchylenie standardowe interfejsu GraphQL większe o 17% niż dla interfejsów REST oraz SOAP, co oznacza że czas uzyskiwania odpowiedzi od bazy danych jest mniej przewidywalny podczas użycia GraphQL. Najstabilniejszym interfejsem programowania aplikacji jest REST, a wykorzystanie SOAP wygenerowało średnio o 7% większe odchylenie standardowe.

Na podstawie uzyskanych wyników można jednoznacznie stwierdzić, że z pośród trzech interfejsów programowania aplikacji udostępnionych na platformie Salesforce to GraphQL jest najmniej wydajny. Są to wyniki zupełnie przeciwne do tych uzyskanych przez autorów badania przeprowadzonego w roku 2020 [7]. Z drugiej strony wyniki pokrywają się jednoznacznie z wynikami jakie otrzymali Erlandsson oraz Remes [10].

Podsumowując, można stwierdzić, że wybór interfejsu, który zostanie użyty podczas wytwarzania oprogramowania powinien być determinowany środowiskiem, które dane interfejsy udostępnia. Nie można jednoznacznie stwierdzić, który interfejs jest wydajniejszy bez określenia platformy, bądź środowiska.

Literatura

[1] J. H. Bhatti, B. B. Rad, Databases in Cloud Computing: A literature review, International Journal of Information Technology and Computer Science 9(4) (2017) 9–17, https://doi.org/10.5815/ijitcs.2017.04.02.

[2] L. A. B. Silva, C. Costa, J. L. Oliveira, A common API for delivering services over multi-vendor cloud resources, Journal of Systems and Software 86(9) (2013) 2309–2317, https://doi.org/10.1016/j.jss.2013.04.037.

[3] Dokumentacja interfejsów programowania aplikacji w Salesforce, https://developer.salesforce.com/docs/apis , [04.04.2023].

[4] Ranking systemów CRM dla roku 2023, https://www.pcmag.com/picks/the-best-crm-software , [04.04.2023].

[5] Dokumentacja Salesforce dotycząca GraphQL, https://developer.salesforce.com/docs/platform/graphql/references/graphql?meta=Summary , [04.04.2023].

[6] A. Quiña-Mera, P. Fernandez, J. M. García, A. Ruiz-Cortés, GraphQL: A Systematic Map-ping Study, ACM Computing Surveys 55(10) (2023) 1–35, https://doi.org/10.1145/3561818.

[7] J. Sayago Heredia, E. Flores-García, A. R. Solano, Comparative analysis between standards oriented to web services: SOAP, REST and GRAPHQL, Proceedings of the Applied Technologies: First International Conference, ICAT 2019, Quito, Ecuador (2019) 286–300, https://doi.org/10.1007/978-3-030-42517-3_22.

[8] Oficjalna dokumentacja GraphQL, https://spec.graphql.org/June2018/ , [04.04.2023].

[9] P. Margański, B. Pańczyk, Analiza porównawcza technologii REST i GraphQL, Journal of Computer Sciences Institute 19 (2021) 89–94, https://doi.org/10.35784/jcsi.2473.

[10] P. Erlandsson, J. Remes, Performance Comparison between GraphQL, REST & SOAP, University of Skovde, Dissertation, https://www.diva-portal.org/smash/record.jsf?pid=diva2:1449837, 2020, [04.04.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 178–185
Received: 3 May 2023
Accepted: 15 May 2023

The analysis of Java ORM frameworks performance in terms of analytical data processing

Analiza efektywności analitycznego przetwarzania danych w języku Java z wykorzystaniem wybranych narzędzi ORM

Justyna Baran*, Piotr Muryjas

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

Abstract
The aim of this paper is to evaluate Java ORM frameworks in terms of analytical data processing. The analysis includes following technologies: Hibernate, Apache Cayenne, EclipseLink and DataNucleus. Article contains characteristics and importance of ORM technologies, as well as the research of related literature. The main study has been performed with the use of implemented Java applications that enabled to run and measure the execution time of analytical queries with various levels of complexity. The analysis of the obtained results enabled to define Hibernate as the most efficient technology for analytical data processing. Additionally the limitations of examined tools and the significant differences between them have been identified and presented.

Keywords: Java; ORM; analytical data processing; performance

Streszczenie
Celem artykułu jest ocena efektywności analitycznego przetwarzania danych w języku Java z użyciem technologii ORM takich jak Hibernate, Apache Cayenne, EclipseLink oraz DataNucleus. Na wstępie przedstawiono charakterystykę i znaczenie narzędzi ORM oraz dokonano przeglądu literatury przedmiotu. Badania zostały zrealizowane przy użyciu aplikacji zaimplementowanych w języku Java i polegały na zmierzeniu czasu wykonania zapytań analitycznych o różnym stopniu złożoności. Uzyskane wyniki badań pozwalają stwierdzić, iż Hibernate jest najbardziej efektywną technologią stosowaną do analitycznego przetwarzania danych. Dodatkowo, na ich podstawie dokonano identyfikacji ograniczeń zastosowania wybranych technologii ORM oraz wskazano istotne różnice występujące między poszczególnymi narzędziami.

Słowa kluczowe: Java; ORM; analityczne przetwarzanie danych; efektywność

*Corresponding author
Email address: **s99171@pollub.edu.pl** (J. Baran)

1. Wstęp

Dane są nieodłączną częścią większości istniejących aplikacji, które gromadzą je oraz umożliwiają ich przetwarzanie. Oprócz podstawowych operacji takich jak odczyt, zapis, modyfikowanie czy usuwanie rekordów, zebrane dane mogą także być analizowane. Analityczne przetwarzanie danych w dzisiejszych czasach pełni istotną rolę w rozwoju organizacji na wielu płaszczyznach. Dostarcza ono cennych informacji, które umożliwiają wsparcie i optymalizację procesów planowania czy podejmowania decyzji. Aby zapewnić wysoką jakość i skuteczność tych działań, niezbędne jest przeprowadzenie efektywnej analizy danych, która często wiąże się z koniecznością wykonywania złożonych zapytań na dużych zbiorach danych. Z tego względu istotna jest kwestia wydajności tego procesu.

Wśród istniejących paradygmatów programowania pod względem popularności wyróżnia się programowanie obiektowe. W procesie komunikacji między relacyjnymi bazami danych, a obiektowymi językami programowania wsparcie stanowią technologie mapowania obiektowo-relacyjnego, czyli ORM.

Jednym z najpopularniejszych języków obiektowych jest Java, która stanowi zaawansowane, wieloplatformowe rozwiązanie i umożliwia efektywną implementację wydajnego oprogramowania biznesowego. Udostępnia ona liczne dodatkowe biblioteki, a także technologie służące do mapowania obiektowo-relacyjnego. Wybór odpowiedniego narzędzia ORM może zależeć od konkretnych potrzeb i przeznaczenia oraz powinien zostać dostosowany do indywidualnych wymagań rozwiązania oferowanego użytkownikowi końcowemu.

Głównym celem niniejszego artykułu jest ocena wybranych narzędzi ORM z punktu widzenia efektywności ich użycia do analitycznego przetwarzania danych. W dalszej jego części przedstawione zostaną szczegółowe wyniki badań dla czterech technologii przeznaczonych dla języka Java. Wybrane do badań narzędzia to: Hibernate, Apache Cayenne, EclipseLink oraz DataNucleus. Wyniki badań pozwoliły dokonać oceny szybkości wykonywania zapytań analitycznych, uruchamianych w poszczególnych środowiskach, a także zaobserwować różnice występujące pomiędzy nimi.

2. Przegląd literatury

Niniejszy rozdział zawiera analizę literatury obejmującej tematykę technologii ORM dostępnych dla języka Java.

S.N. Bhatii i in. dokonali analizy wydajności narzędzi Hibernate, Ebean oraz Oracle TopLink [1]. Porównanie zostało przeprowadzone na podstawie czasu wykonania wybranych zapytań SQL. Środowisko badawcze przygotowano z wykorzystaniem bazy danych MySQL, zawierającej dwie tabele powiązane relacją jeden do wielu, w których znajdowało się odpowiednio 5200 i 40 rekordów. Podczas badań wykonano instrukcje INSERT, DELETE, UPDATE oraz SELECT z różnymi operatorami. Spośród badanych narzędzi, technologia Ebean okazała się najbardziej wydajna, jedynie w przypadku operacji INSERT lepszy rezultat uzyskała technologia Hibernate. Dla wszystkich przeprowadzonych scenariuszy narzędzie TopLink uzyskało najgorsze wyniki.

Poszerzone badania w tym obszarze przeprowadzili N. Dhingra i in. W ich publikacji [2] dokonano porównania efektywności technologii opartych na standardzie JPA, tj. Hibernate, EclipseLink, OpenJPA oraz DataNucleus. W ramach przeprowadzonych badań zweryfikowano wpływ różnych zapytań na wykorzystanie procesora i dysku oraz zarządzanie wątkami i pamięcią przez maszynę wirtualną Javy (JVM). W badaniach wykorzystano pięć zapytań SQL zawierających: złączenie wewnętrzne, stronicowanie, zliczanie rekordów, funkcje agregujące oraz zapytanie sparametryzowane. Z punktu widzenia w/w kryteriów najlepsze rezultaty uzyskały technologie Hibernate oraz EclipseLink, a narzędzie OpenJPA okazało się najmniej wydajne. Technologia DataNucleus uzyskała akceptowalne wyniki, ale rozwiązanie to nie jest powszechnie stosowane ze względu na duży stopień skomplikowania.

Podobne badania zostały przeprowadzone przez B.Pllana, a ich celem szczegółowym było porównanie wydajności trzech najpopularniejszych technologii ORM implementujących specyfikację JPA, tj. EclipseLink, Hibernate oraz OpenJPA [3]. Autor dokonał oceny efektywności wykonywania operacji DML (INSERT, UPDATE, DELETE) oraz prostych zapytań SELECT. Dla zapytań DML najbardziej efektywną okazała się technologia OpenJPA, na drugim miejscu uplasował się EclipseLink, a na ostatnim Hibernate. Z kolei w przypadku zapytań SELECT narzędzie Hibernate uzyskało najlepsze wyniki, natomiast technologia OpenJPA okazała się najmniej efektywna. Wyniki badań dodatkowo pokazały, iż testowane technologie charakteryzują się różną wydajnością w zależności od rodzaju przeprowadzanych operacji.

Ocenę wydajności technologii Hibernate, EclipseLink, Apache OpenJPA oraz DataNucleus przeprowadzili również M.Połeć i in. [4]. W ich badaniach wykorzystano aplikację webową połączoną z bazą danych MySQL, zawierającą 14 tabel z relacjami jeden do jeden oraz jeden do wielu. Wykorzystane przez nich scenariusze badawcze obejmowały odczytywanie, tworzenie, aktualizowanie oraz usuwanie danych dla różnej liczby rekordów w poszczególnych tabelach. Uzyskane wyniki badań wskazują, iż najwyższą wydajnością charakteryzuje się narzędzie Apache OpenJPA. Jednak pod względem konfiguracji oraz dostępności dokumentacji technologia Hibernate posiada znaczącą przewagę.

W uzupełnieniu warto także przedstawić wnioski z badań przeprowadzonych przez M. Żuchnik i in., dotyczących oceny wydajność komunikacji z bazą danych przy pomocy interfejsu JDBC oraz wybranych technologii ORM [5]. Badaniom zostały poddane narzędzia JDBC, jOOQ, MyBatis oraz Hibernate. Jako kryteria ich oceny przyjęto czas wykonania zapytań dla operacji odczytu, zapisu, aktualizacji i usuwania rekordów, zużycie pamięci RAM, wykorzystanie procesora, popularność narzędzia, podatność na ataki SQL Injection, wsparcie dla różnych dialektów, weryfikację składni oraz konieczność znajomości języka SQL. Dla każdego z rozpatrywanych kryteriów przypisano odpowiednią wagę. Najwyższą z nich otrzymała szybkość wykonywania zapytań oraz odporność na ataki SQL Injection, natomiast za najmniej istotne uznano znajomość SQL oraz weryfikację składni. Uwzględniając zdefiniowane wagi, najbardziej efektywne okazało się rozwiązanie JDBC, na drugim miejscu znalazła się technologia Hibernate, a następnie MyBatis i jOOQ.

Przeprowadzona analiza literatury przedmiotu wskazuje na brak publikacji dotyczących oceny narzędzi ORM z punktu widzenia analitycznego przetwarzania danych. Z tego powodu niniejszy artykuł i zaprezentowane w nim wyniki badań mogą mieć znaczenie i stanowić pomoc w optymalnym wyborze przeznaczonej dla języka Java technologii ORM wykorzystywanej w analitycznym przetwarzaniu danych.

3. Charakterystyka technologii ORM

Obsługę współpracy obiektowych technologii programistycznych z relacyjną bazą danych zapewnia mapowanie obiektowo-relacyjne. Z jego pomocą dane w tabelach relacyjnych oraz obiekty w aplikacji są wzajemnie konwertowane [6]. Ze względu na liczne różnice między modelem relacyjnym i obiektowym, narzędzia ORM stanowią duże usprawnienie, tworząc dodatkową warstwę abstrakcji, która ułatwia synchronizację tabel bazodanowych i obiektów [7].

Spośród technologii ORM dostępnych dla języka Java, w niniejszym artykule zaprezentowano cztery narzędzia o otwartym kodzie źródłowym, które są stabilnymi i rozbudowanymi rozwiązaniami. Oferują one współpracę z licznymi systemami zarządzania bazami danych, a oprócz podstawowych operacji, dostarczają także dodatkowe funkcjonalności wpływające na poprawę wydajności zarządzania danymi. W związku z tym powinny one umożliwić efektywną realizację złożonych zapytań, niezbędnych podczas analitycznego przetwarzania danych.

3.1. Hibernate

Hibernate jest technologią powszechnie stosowaną dla projektów implementowanych w języku Java. Zapewnia ona możliwość korzystania z implementacji JPA (Java Persistence API), a oprócz wbudowanego w tym standardzie języka JPQL (Java Persistence Query Language) dostarcza także jego własną, rozszerzoną wersję –

HQL (Hibernate Query Language). Wśród zaawansowanych funkcjonalności oferowanych przez Hibernate można wymienić kontrolę współbieżnego dostępu do danych, leniwe ładowanie, liczne strategie pobierania danych oraz ich walidację [8].

3.2. EclipseLink

Technologia EclipseLink powstała w oparciu o narzędzie TopLink stworzone przez firmę Oracle. Implementuje ona standard JPA, dostarczając zaawansowaną obsługę komunikacji z relacyjnymi bazami danych. Nie ogranicza się tylko do funkcjonalności dostępnych w standardzie, ale dostarcza także dodatkowych komponentów, które między innymi pozwalają wykorzystać pamięć podręczną i zaawansowane mapowanie oraz zapewniają prostą współpracę z formatami XML i JSON [9].

3.3. DataNucleus

DataNucleus to technologia oferująca wsparcie podczas pracy z różnymi źródłami danych. Oprócz relacyjnych, nierelacyjnych i grafowych baz danych, zapewnia ona współpracę z arkuszami kalkulacyjnymi oraz plikami w formacie OpenDocument. Jedną z dostarczonych metod komunikacji z bazą danych jest implementacja standardu JPA. Inne funkcjonalności tej technologii obejmują także wykorzystanie interfejsów JDO i Jakarta [10].

3.4. Apache Cayenne

W przeciwieństwie do uprzednio przedstawionych rozwiązań, technologia Apache Cayenne nie implementuje standardu JPA, ale wykorzystuje moduł EJBQLQuery obsługujący ciągi znaków zgodnych ze składnią języka JPQL. W celu komunikacji z relacyjną bazą danych udostępnia także interfejs dostarczający gotowy zestaw zapytań wraz z możliwością definiowania własnych wyrażeń. Wbudowane dodatkowe narzędzie z graficznym interfejsem użytkownika – Cayenne Modeler, ułatwia mapowanie obiektów i umożliwia bezpośrednie generowanie kodu Java na podstawie schematu bazy danych [11].

4. Środowisko badawcze

Dla każdej z wybranych technologii zaimplementowana została aplikacja w języku Java, która umożliwia połączenie z bazą danych oraz realizację zaplanowanych scenariuszy badawczych. Do badań wybrane zostały następujące wersje technologii: Hibernate 6.1.7, Apache Cayenne 4.0, EclipseLink 2.7.8 oraz DataNucleus 6.0.0. Aplikacje zostały przygotowane przy pomocy środowiska programistycznego IntelliJ Idea Community Edition 2022.2.2, z wykorzystaniem OpenJDK w wersji 11 i szkieletu programistycznego Spring Boot 3.0.5. Zastosowanie narzędzia Apache Maven usprawniło proces budowania projektu oraz dołączania zależności.

Do przeprowadzenia badań wykorzystany został system zarządzania bazą danych Microsoft SQL Server 2019 Express oraz zintegrowane środowisko do zarządzania jego komponentami, tj. SQL Server Management Studio v18.12.1. Wybranym źródłem danych jest udostępniona przez Microsoft przykładowa relacyjna baza danych AdventureWorks2019, która reprezentuje fikcyjną firmę produkującą rowery i akcesoria rowerowe. W bazie zgromadzone są między innymi dane na temat produktów, pracowników, klientów oraz zamówień [12]. W badaniach uwzględnione zostały tabele o następującej liczbie rekordów: FactInternetSales – 100000, DimDate – 3652, DimProduct – 606, DimProductSubcategory – 37, DimProductCategory – 4, DimCustomer – 18484, DimSalesTerritory – 11.

Badania zostały przeprowadzone z wykorzystaniem tego samego sprzętu komputerowego, dane techniczne przedstawia (Tabela 1).

Tabela 1: Specyfikacja środowiska testowego

Nazwa	Wartość
Laptop	ASUS VivoBook 15 R520UA
Procesor	Intel Core i3-8130U CPU 2.20GHz
System operacyjny	Windows 10 64 bit
Pamięć RAM	8GB (DDR4, 2400MHz)
Dysk	SSD M.2 256 GB

5. Metodyka badawcza

W celu przygotowania aplikacji dla każdej z badanych technologii niezbędne było przeprowadzenie konfiguracji środowiska i połączenia z bazą danych. W kolejnym etapie zdefiniowano klasy reprezentujące poszczególne encje, a następnie zaimplementowane zostały zapytania odpowiadające zdefiniowanym scenariuszom badawczym. Podczas implementacji zapytań wykorzystano udostępnione przez technologie ORM języki: HQL dla technologii Hibernate, JPQL w przypadku EclipseLink i DataNucleus oraz EJBQL dla Apache Cayenne. Aby umożliwić zebranie wyników badań wydajnościowych, dodana została funkcjonalność umożliwiająca pomiar czasu, w jakim wykonywane jest każde zapytanie (w milisekundach). Po uruchomieniu aplikacji wykonanie zapytania było możliwe za pomocą wywołania odpowiedniej komendy z poziomu wiersza poleceń.

W celu zwiększenia dokładności wyników badań, każde zapytanie dla każdej testowanej technologii zostało wykonane 10 razy, a ostateczny poddany porównaniu wynik stanowił średnią wartość z zebranych pomiarów cząstkowych. Przed każdą próbą niezbędne było wyczyszczenie pamięci podręcznej systemu bazy danych przy pomocy polecenia DBCC DROPCLEANBUFFERS. Przeprowadzanie eksperymentu odbyło się bez połączenia internetowego, a oprócz systemu operacyjnego uruchomione zostały tylko niezbędne usługi, tj. stworzona aplikacja oraz serwer baz danych. Pozwoliło to zmniejszyć obciążenie systemu operacyjnego i zminimalizować liczbę działających w tle procesów, które mogłyby korzystać z zasobów procesora w trakcie dokonywania pomiarów.

Oprócz czasu wykonania poszczególnych zapytań porównana została także częstotliwość występowania wzmianek na temat badanych technologii w literaturze na podstawie liczby wyników w wyszukiwarce Google Scholar, która pozwala przeszukiwać bazy danych za-

wierające publikacje naukowe. Uwzględniono wszystkie dostępne dane wyszukiwania na dzień 15.04.2023. Oceniono także wielkość społeczności korzystającej z wybranych rozwiązań na podstawie liczby pytań zadanych w serwisie Stack Overflow dotyczących poszczególnych technologii. Dane zostały zebrane w dniu 15.04.2023 z uwzględnieniem wszystkich dostępnych zapytań w serwisie.

6. Scenariusze badawcze

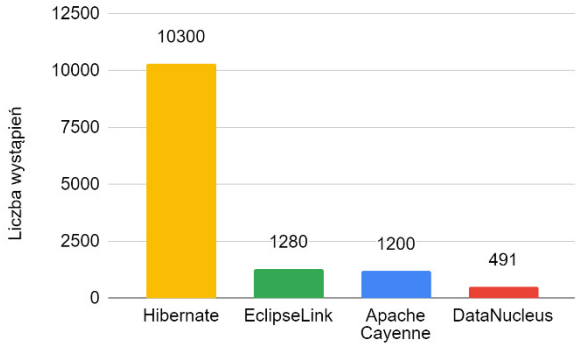
Na potrzeby oceny efektywności analitycznego przetwarzania danych z wykorzystaniem wybranych technologii ORM przygotowane zostały następujące scenariusze badawcze:

1. Wyznaczenie całkowitej wartości sprzedaży dla poszczególnych produktów (S1).
2. Wyznaczenie całkowitej wartości poszczególnych zamówień oraz liczby pozycji dla każdego zamówienia w poszczególnych miesiącach kolejnych lat (S2).
3. Wyznaczenie całkowitej wartości zamówień, liczby zamówień oraz maksymalnej i minimalnej ceny produktów w obrębie danego zamówienia w poszczególnych miesiącach kolejnych lat (S3).
4. Wyznaczenie całkowitej wartości sprzedaży dla poszczególnych podkategorii i kategorii produktów wraz z podsumowaniem całkowitej wartości sprzedaży dla każdej podkategorii oraz łączną wartością sprzedaży dla wszystkich produktów (S4).
5. Wyznaczenie produktów, które nie pojawiły się w żadnych zamówieniach (S5).
6. Wyznaczenie 5 krajów o największej całkowitej wartości sprzedaży (S6).
7. Wyznaczenie liczby klientów, którzy złożyli zamówienia w grudniu poszczególnych lat (S7).
8. Wyznaczenie liczby klientów oraz całkowitej wartości zamówień złożonych między określonymi datami w Australii i Stanach Zjednoczonych (S8).
9. Wyznaczenie liczby klientów, którzy w grudniu poszczególnych lat w Stanach Zjednoczonych złożyli zamówienie zawierające produkt o nazwie HL Mountain Tire (S9).
10. Wyznaczenie produktów sprzedanych w 2018 roku, których całkowita wartość sprzedaży wyniosła więcej niż 10000 (S10).
11. Wyznaczenie danych 3 klientów, których średnia wartość zamówienia jest największa (S11).
12. Wyznaczenie procentowego udziału wartości sprzedaży z poszczególnych krajów w całkowitej wartości sprzedaży (S12).
13. Wyznaczenie procentowego udziału wartości miesięcznej sprzedaży w sprzedaży rocznej dla poszczególnych regionów (S13).
14. Wyznaczenie krajów w poszczególnych regionach, dla których całkowita wartość sprzedaży jest większa od średniej wartości sprzedaży krajowej w danym regionie (S14).
15. Wskazanie dla każdego miesiąca kolejnych lat produktu, którego wartość sprzedaży jest największa (S15).
16. Wyznaczenie liczby klientów i całkowitej wartości sprzedaży w kolejnych latach dla poszczególnych rodzajów rowerów (S16).
17. Wyznaczenie danych klientów, których łączna wartość zamówień jest większa niż całkowita wartość sprzedaży dla kraju Canada w grudniu 2018 roku (S17).
18. Wyznaczenie produktów, na które zostały złożone zamówienia zarówno w regionie Southwest, jak i Australia (S18).
19. Wyznaczenie produktu, który został sprzedany w największej liczbie sztuk w poszczególnych latach dla poszczególnych krajów (S19).

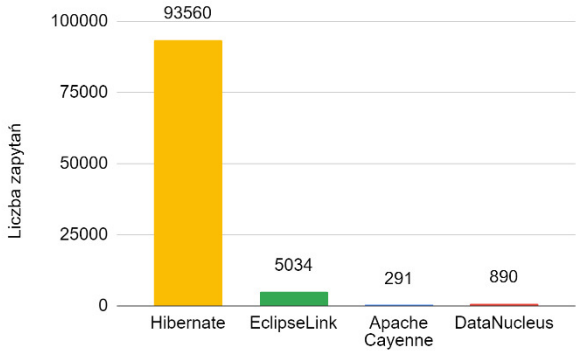
7. Wyniki badań

W niniejszej części artykułu przedstawiono wyniki przeprowadzonych badań dla uprzednio opisanych scenariuszy.

Rysunek 1 ilustruje liczbę wystąpień fraz Hibernate ORM, EclipseLink, Apache Cayenne oraz DataNucleus w wyszukiwarce Google Scholar. Technologia Hibernate pod względem liczby wystąpień w literaturze posiada znaczącą przewagę w porównaniu do pozostałych narzędzi.



Rysunek 1: Częstotliwość występowania wzmianek w literaturze.

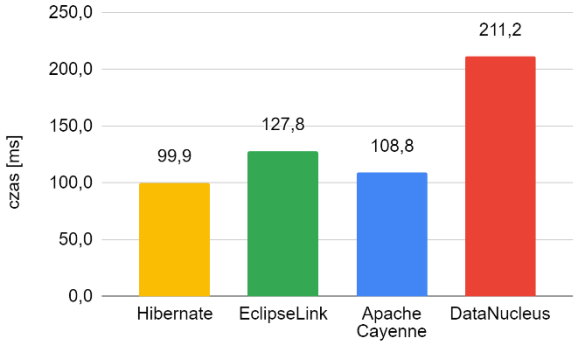


Rysunek 2: Liczba zapytań w serwisie Stack Overflow.

Największa liczba zapytań w serwisie Stack Overflow dotyczy technologii Hibernate, następnie EclipseLink i DataNucleus, a najmniej pytań pojawiło się dla Apache Cayenne, co zostało zobrazowane na Rysunku 2.

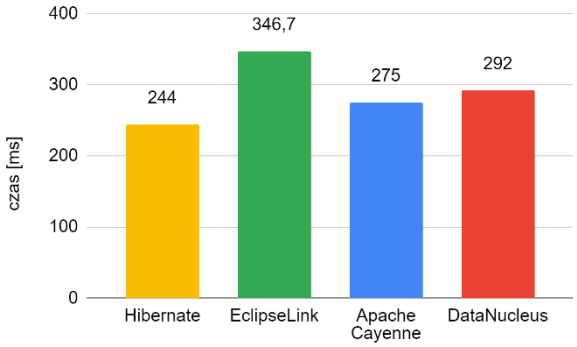
Wyniki badań dla scenariusza S1 przedstawiono na Rysunku 3. Najkrótszy średni czas wykonania zapytania uzyskano w technologii Hibernate. Narzędzie Apache

Cayenne umożliwiło realizację scenariusza w nieznacznie dłuższym czasie (9%). Trzecia w kolejności znalazła się technologia EclipseLink (28% dłuższy czas), natomiast DataNucleus jest narzędziem, dla którego średni czas wykonania scenariusza S1 okazał się ponad dwukrotnie dłuższy w porównaniu do Hibernate.



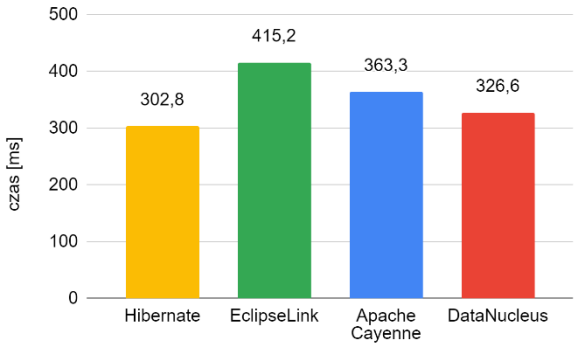
Rysunek 3: Średni czas wykonania scenariusza S1.

Rysunek 4 przedstawia wyniki badań dla scenariusza S2. Wykonanie zapytania w najkrótszym czasie było możliwe z wykorzystaniem technologii Hibernate. Kolejne miejsca w rankingu efektywności zajęły technologie Apache Cayenne oraz DataNucleus. Scenariusz S2 był realizowany najdłużej w technologii EclipseLink.



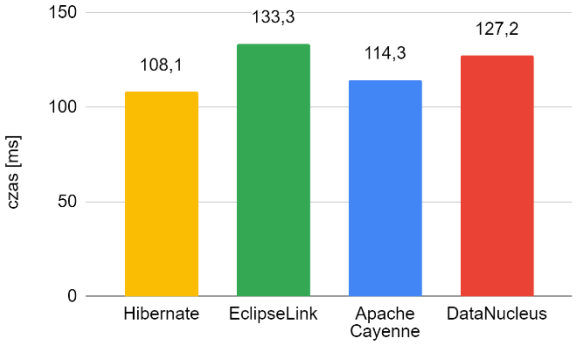
Rysunek 4: Średni czas wykonania scenariusza S2.

Rysunek 5 obrazuje średni czas realizacji scenariusza S3 dla wszystkich środowisk. Najlepsza z punktu widzenia efektywności okazała się technologia Hibernate, następnie DataNucleus, Apache Cayenne i ostatecznie EclipseLink.



Rysunek 5: Średni czas wykonania scenariusza S3.

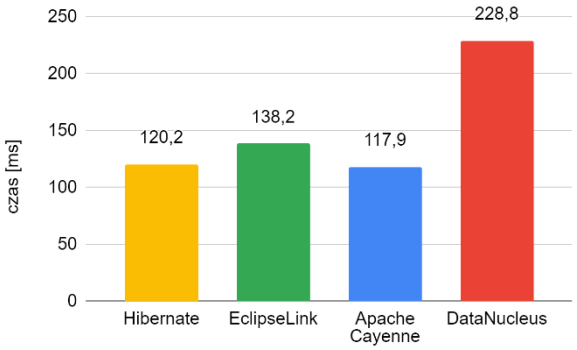
Scenariusz S4 został zrealizowany z wykorzystaniem natywnego zapytania SQL, ponieważ we wszystkich omawianych technologiach ORM udostępnione obiektowe sposoby definiowania zapytań nie pozwalają na zastosowanie klauzuli ROLLUP. Wyniki tego eksperymentu przedstawiono na Rysunku 6.



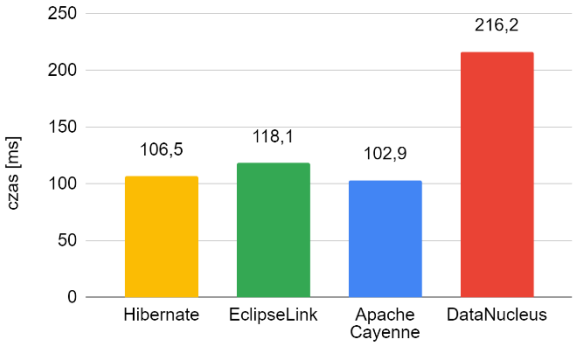
Rysunek 6: Średni czas wykonania scenariusza S4.

W przypadku natywnego zapytania SQL ponownie technologia Hibernate zapewniła najkrótszy czas jego wykonania, jakkolwiek jej przewaga nad pozostałymi nie jest już tak znacząca jak w dotychczas zaprezentowanych scenariuszach.

Rysunki 7 i 8 prezentują wyniki realizacji scenariuszy S5 i S6. W tych przypadkach najkrótszy średni czas wykonania zapytania zapewniła technologia Apache Cayenne. Nieznacznie ustępują jej narzędzia Hibernate oraz EclipseLink. Najgorszy wynik uzyskała technologia DataNucleus.

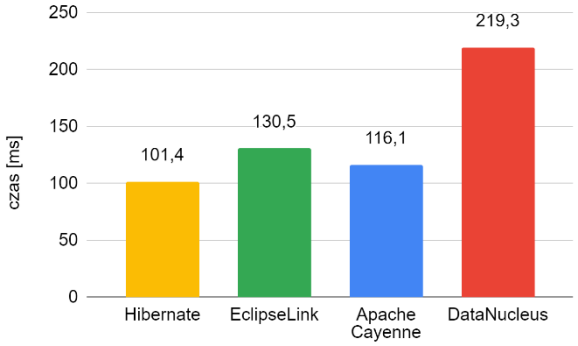


Rysunek 7: Średni czas wykonania scenariusza S5.

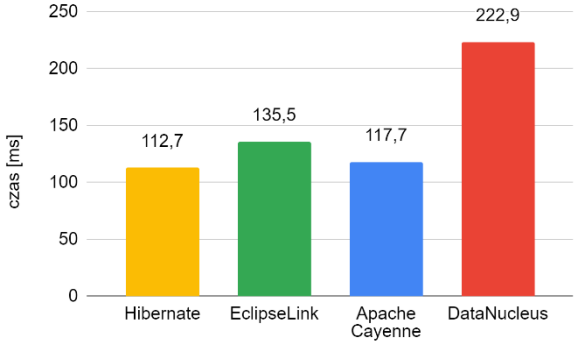


Rysunek 8: Średni czas wykonania scenariusza S6.

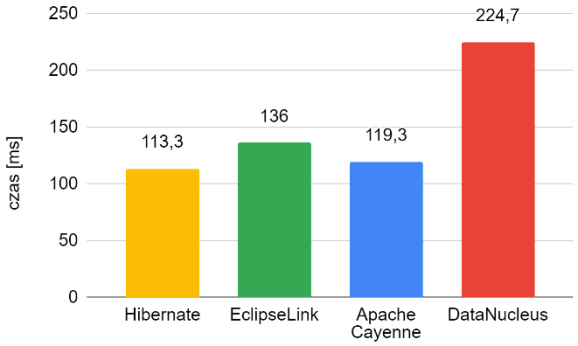
Wizualizacje Rysunek 9 do Rysunek 15 prezentują wyniki badań dla scenariuszy S7, S8, S9, S10, S16, S17 i S18. We wszystkich wymienionych przypadkach najbardziej wydajna okazała się technologia Hibernate. Bardzo dobre wyniki analitycznego przetwarzania danych uzyskano również przy użyciu technologii Apache Cayenne. Na trzeciej pozycji w tym rankingu znalazła się technologia EclipseLink. Natomiast przetwarzanie danych w narzędziu DataNucleus wymagało najdłuższego czasu, niemal dwukrotnie dłuższego niż w technologii Hibernate.



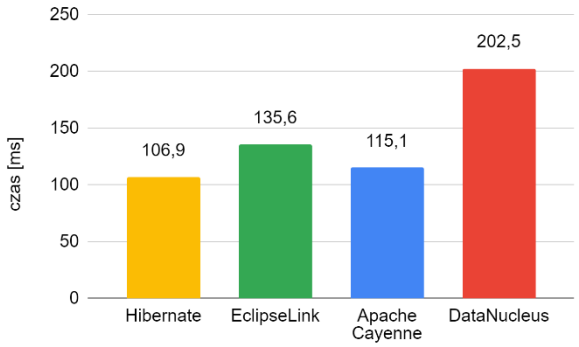
Rysunek 9: Średni czas wykonania scenariusza S7.



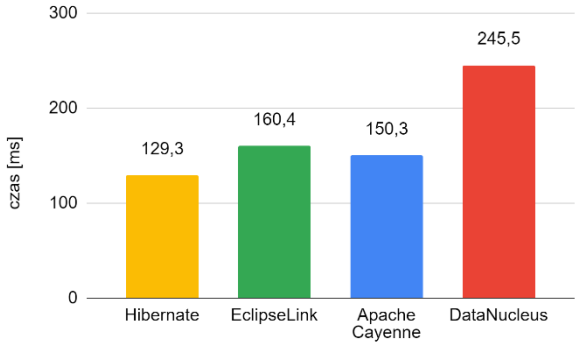
Rysunek 10: Średni czas wykonania scenariusza S8.



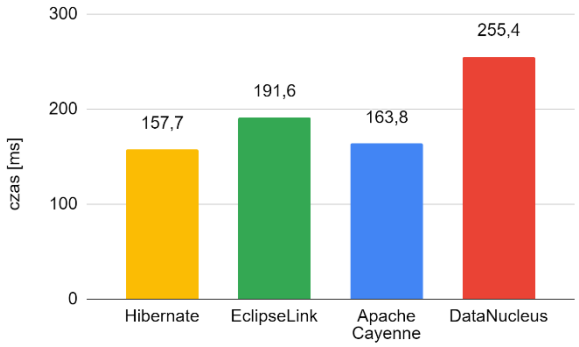
Rysunek 11: Średni czas wykonania scenariusza S9.



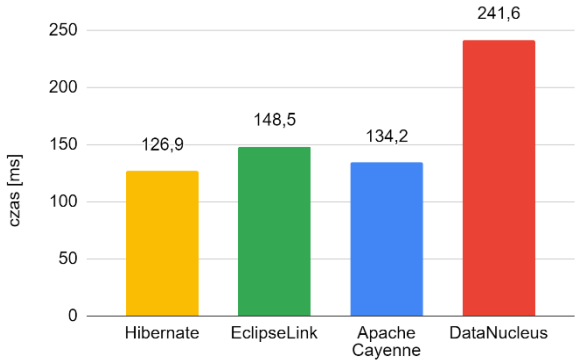
Rysunek 12: Średni czas wykonania scenariusza S10.



Rysunek 13: Średni czas wykonania scenariusza S16.



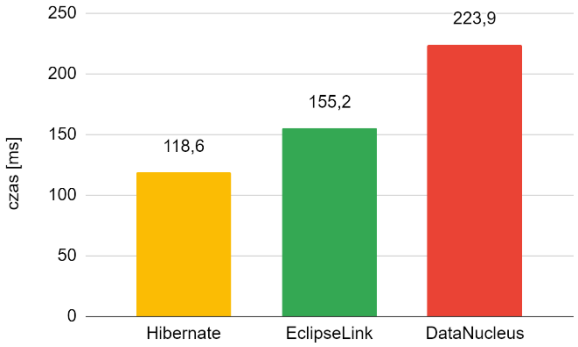
Rysunek 14: Średni czas wykonania scenariusza S17.



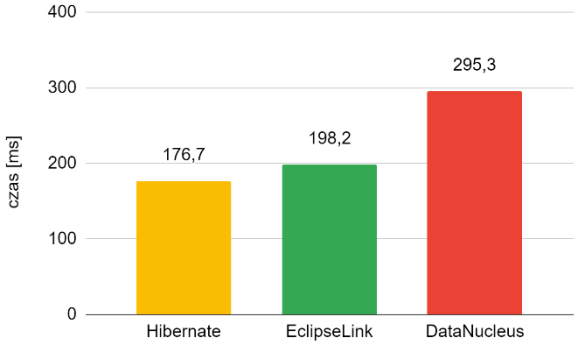
Rysunek 15: Średni czas wykonania scenariusza S18.

Na Rysunkach 16 i 17 zostały zaprezentowane wyniki realizacji scenariuszy S12 i S13 dla wszystkich omawianych technologii z wyjątkiem Apache Cayenne. W przypadku wykorzystanej wersji tego narzędzia

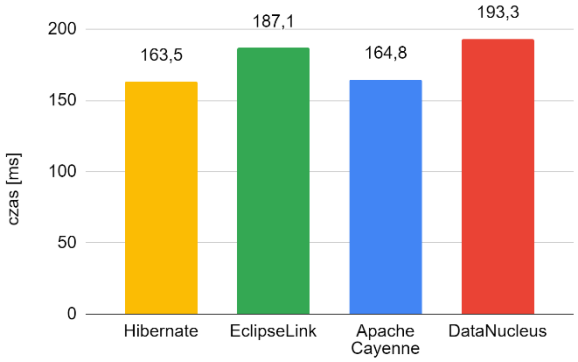
i języka EJBQL nie jest możliwe zastosowanie operacji dzielenia w klauzuli SELECT. Obydwa scenariusze zostały wykonane najszybciej z użyciem technologii Hibernate, a następnie EclipseLink i DataNucleus.



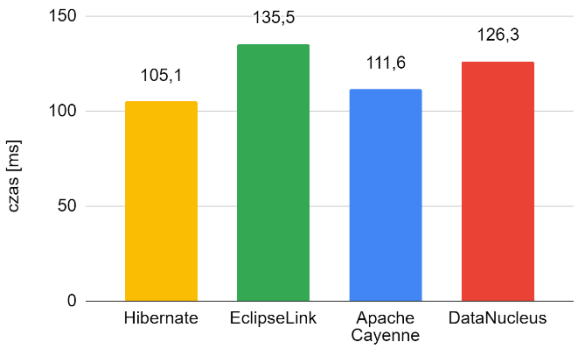
Rysunek 16: Średni czas wykonania scenariusza S12.



Rysunek 17: Średni czas wykonania scenariusza S13.



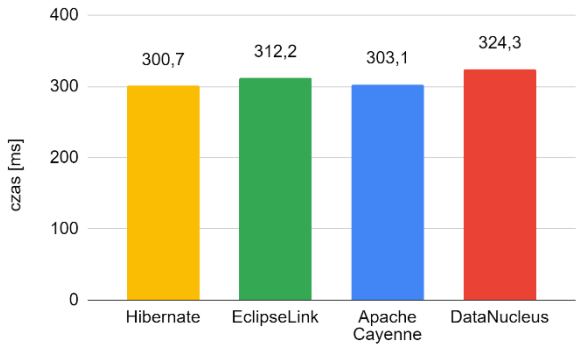
Rysunek 18: Średni czas wykonania scenariusza S11.



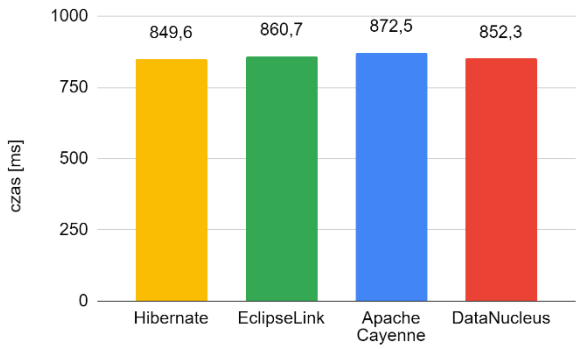
Rysunek 19: Średni czas wykonania scenariusza S14.

Scenariusze S11 i S14 wymagają zastosowania podzapytania w klauzuli FROM. Wykorzystane do implementacji zapytań języki JPQL i EJBQL nie obsługują tego typu operacji. Z tego powodu scenariusze te zostały zrealizowane z wykorzystaniem zapytania SQL, a wyniki tych badań przedstawiono na Rysunkach 18 i 19. Technologia Hibernate umożliwiła realizację wspomnianych scenariuszy w języku HQL z wykorzystaniem mapowania obiektowo-relacyjnego. W tym przypadku średni czas wyniósł 170,2 ms dla scenariusza S11 i 107,9 ms dla scenariusza S14.

W celu wykonania scenariuszy S15 i S19 konieczne jest zastosowanie klauzuli TOP w podzapytaniu, co okazało się niemożliwe w językach JPQL i EJBQL. Zapytania zostały zdefiniowane z zastosowaniem języka SQL, a średni czas ich wykonania zaprezentowano na Rysunkach 20 i 21.



Rysunek 20: Średni czas wykonania scenariusza S15.



Rysunek 21: Średni czas wykonania scenariusza S19.

Technologia Hibernate pozwala również na zastosowanie mapowania obiektowo-relacyjnego do realizacji tych scenariuszy. Średni czas wykonania scenariusza S15 wyniósł 357,4 ms, a scenariusza S19 - 858,5 ms.

8. Wnioski

Przeprowadzone badania oraz analiza ich wyników pozwoliły ocenić efektywność analitycznego przetwarzania danych z wykorzystaniem wybranych technologii ORM przeznaczonych dla języka Java.

Na podstawie wyników badań wykonanych z użyciem scenariuszy S4, S11, S12, S13, S14, S15 i S19 można zaobserwować pewne ograniczenia technologii ORM, które w udostępnianych zorientowanych obiektowo metodach nie zapewniają wszystkich elementów

składni języka SQL występujących w jego standardzie. Rozwiązaniem tego problemu jest wykonanie operacji przetwarzania danych z użyciem natywnych zapytań SQL. Jednak takie podejście eliminuje możliwość korzystania z zalet technologii ORM, takich jak bezpośrednie operowanie na obiektach, wygodna konwersja między obiektami i tabelami w bazie danych oraz przenośność rozwiązania na inne systemy zarządzania bazą danych.

Dla większości scenariuszy badawczych (17 na 19) najkrótszy średni czas wykonania zapytań uzyskano w technologii Hibernate. Tylko w dwóch przypadkach jej wydajność była nieznacznie niższa w porównaniu do najlepszego wyniku. Narzędzie Hibernate umożliwiło także realizację największej liczby scenariuszy badawczych (18 na 19) przy pomocy zorientowanego obiektowo języka HQL. Świadczy to o wysokim stopniu zaawansowania technologii i możliwości realizacji nawet bardzo złożonych zapytań. Technologia ta sprawdziła się również najlepiej w przypadku scenariuszy wykonanych przy pomocy języka SQL. Warto również zauważyć, że w przypadku porównania czasu wykonania zapytań przy użyciu języka HQL, średni czas był niewiele dłuższy (średnio o ok. 7%) od uzyskanego podczas korzystania z języka SQL. Fakt ten stanowi dowód na wysoką wydajność mapowania obiektowo-relacyjnego zaimplementowanego w tej technologii. Jego użycie nie powoduje dużego spadku efektywności przetwarzania danych zdefiniowanego w natywnym języku SQL. W porównaniu do pozostałych technologii, Hibernate charakteryzuje się największą społecznością korzystającą z technologii i największą liczbą wystąpień wzmianek w literaturze przedmiotu. Posiada również rozbudowaną i aktualną dokumentację, a proces konfiguracji czy budowania zapytań jest najszybszy i najmniej skomplikowany. Wymienione powyżej zalety w istotny sposób wpływają na wybór tej technologii ORM przez większość twórców aplikacji webowych, zapewniając w ten sposób skrócenie czasu ich produkcji oraz efektywne ich wykorzystanie w docelowym środowisku klienta końcowego.

Technologia Apache Cayenne w 13 na 19 przypadków zapewniła uzyskanie zbliżonych średnich czasów realizacji scenariuszy do tych, jakie zmierzono w środowisku Hibernate. Z tego powodu może ona stanowić alternatywę dla rozwiązań bazujących na implementacji standardu JPA. Jednak znacznymi wadami tego rozwiązania są ograniczenia związane z definiowaniem złożonych zapytań, niewielka liczba materiałów dotyczących danej technologii oraz bardzo mała społeczność. Również korzystanie z tego narzędzia, w porównaniu do technologii Hibernate, nie było intuicyjne, wymagało umiejętności konfiguracji mapowania obiektów i relacji w języku xml, a czas definiowania zapytań był znacznie dłuższy.

Dla większości scenariuszy badawczych (14 na 19) technologia DataNucleus okazała się zdecydowanie najmniej wydajna. Niewielka społeczność oraz niska dostępność dodatkowych materiałów dotyczących tej technologii negatywnie wpływa na niską skłonność wyboru tego narzędzia do analitycznego przetwarzania danych.

Z kolei technologia EclipseLink nie wyróżniła się znacząco pod względem wydajnej realizacji zdefiniowanych scenariuszy. Zaletą tego rozwiązania jest rozbudowana, poparta wieloma przykładami dokumentacja oraz większa społeczność w porównaniu do Apache Cayenne i DataNucleus, dzięki czemu podczas korzystania z EclipseLink zapewnione jest większe wsparcie.

Na podstawie otrzymanych wyników badań własnych i powyższych wniosków można stwierdzić, że technologia Hibernate charakteryzuje się największą wydajnością i jest najbardziej efektywnym rozwiązaniem ORM w przypadku analitycznego przetwarzania danych.

Literatura

[1] S. N. Bhatti, Z. H. Abro, F. Rufabro, Performance evaluation of java based object relational mapping tools, Mehran University Research Journal of Engineering and Technology 32(2) (2013) 159-166.

[2] N. Dhingra, E. Abdelmoghith, H. T. Mouftah, Performance Evaluation of JPA Based ORM Techniques, 2nd International Conference on Computer Science Networks and Information Technology (2016) 15-23.

[3] B. Pllana, Performance Analysis of Java Persistence API Providers, UBT International Conference (2018) 100-107.

[4] M. Połeć, J. Pitera, G. Kozieł, Comparing the Performance of the Object-Relational Mapping Programming Frameworks Available in Java, Journal of Computer Sciences Institute 22 (2022) 59-65.

[5] M. Żuchnik, P. Kopniak, Comparative analysis of connection performance with databases via JDBC interface and ORM programming frameworks, Journal of Computer Sciences Institute 21 (2021) 309-315.

[6] Z. Rosiek, Mapowanie obiektowo-relacyjne ORM-czy tylko dobra idea, Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki 4(4) (2010) 99-112.

[7] R. G. Sfirlogea, A Decision Support Model for using an Object-Relational Mapping Tool in the Data Management Component of a Software Platform, University of Utrecht Master's thesis, 2015.

[8] Dokumentacja techniczna technologii Hibernate, https://hibernate.org/orm, [10.04.2023].

[9] Dokumentacja techniczna technologii EclipseLink, https://wiki.eclipse.org/EclipseLink, [10.04.2023].

[10] Dokumentacja techniczna technologii DataNucleus, https://www.datanucleus.org, [10.04.2023].

[11] Dokumentacja techniczna technologii Apache Cayenne, https://cayenne.apache.org, [10.04.2023].

[12] M. Mitri, Teaching Tip: Active Learning via a Sample Database: The Case of Microsoft's Adventure Works, Journal of Information Systems Education 26(3) (2015) 177-186.