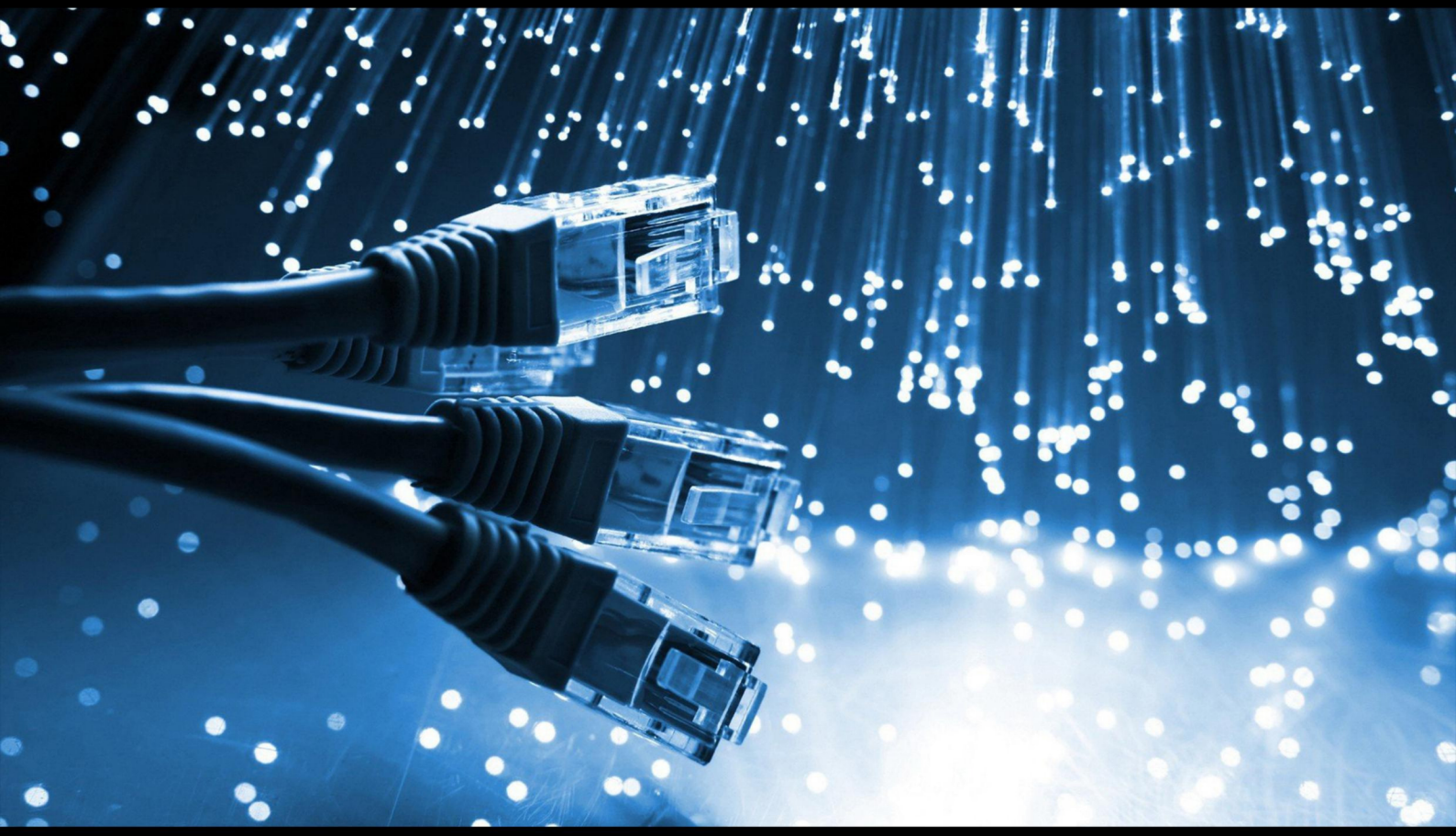
JCSI

Journal of Computer Sciences Institute

Volume 24/2022



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. Elżbieta Miłosz
dr inż. Małgorzata Plechawska-Wójcik
dr inż. Jakub Smołka
dr inż. Grzegorz Kozieł
dr inż. Krzysztof Dziedzic
dr inż. Maciej Pańczyk
dr inż. Sylwester Korga
dr inż. Kamil Żyła
dr Marcin Barszcz
dr Mariusz Dzieńkowski
dr inż. Marcin Badurowicz
dr Paweł Powroźnik
dr inż. Piotr Muryjas
dr Michał Dolecki

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:

Elżbieta Miłosz
Małgorzata Plechawska-Wójcik
Jakub Smołka
Grzegorz Kozieł
Krzysztof Dziedzic
Maciej Pańczyk
Sylwester Korga
Kamil Żyła
Marcin Barszcz
Mariusz Dzieńkowski
Marcin Badurowicz
Paweł Powroźnik
Piotr Muryjas
Michał Dolecki

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

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

1. **Szybkość uczenia czy dokładność predykcji? Analiza porównawcza szkieletów programistycznych do sztucznej inteligencji**
   Konrad Zdeb, Piotr Żukiewicz, Edyta Łukasik.....172-175
2. **Analiza porównawcza skuteczności narzędzi OWASP ZAP, Burp Suite, Nikto i Skipfish w testowaniu bezpieczeństwa aplikacji internetowych**
   Aleksandra Kondraciuk, Aleksandra Bartos, Beata Pańczyk.....176-180
3. **Analiza porównawcza funkcjonalności i jakości interfejsu wybranych aplikacji do zamawiania jedzenia**
   Maciej Gieroba, Marek Miłosz.....181-188
4. **Analiza porównawcza jakości zarejestrowanego dźwięku w funkcji różnych formatów zapisu**
   Andrzej Król, Tomasz Szymczyk.....189-194
5. **Porównanie najpopularniejszych systemów operacyjnych pod względem funkcjonalności**
   Jacek Lato, Marek Mucha, Tomasz Szymczyk.....195-202
6. **W jakim stopniu system cyfryzacji polskich placówek medycznych jest gotowy do pracy online?**
   Maciej Mikrut.....203-209
7. **Analiza funkcjonalności systemów komunikacji głosowej i wizyjnej**
   Aleksandra Piątkowska.....210-217
8. **Analiza porównawcza silników graficznych Cycles oraz Eevee na przykładzie renderowania modeli 3D artefaktów archeologicznych**
   Sebastian Dudek, Krzysztof Dziedzic.....218-223
9. **Analiza porównawcza szkieletów programistycznych React, Next oraz Gatsby do tworzenia aplikacji typu SPA**
   Adam Świątkowski, Karol Ścibior.....224-227
10. **Porównanie wydajności wybranych silników szachowych**
    Maciej Sojka.....228-235
11. **Porównanie ofert wybranych dostawców usług chmurowych z punktu widzenia realizacji projektów informatycznych opartych o otwarty kod**
    Jan Baran, Sławomir Przyłucki.....236-241
12. **Analiza porównawcza podejścia reaktywnego i imperatywnego w tworzeniu aplikacji internetowych w języku Java**
    Sebastian Iwanowski, Grzegorz Kozieł.....242-249
13. **Analiza wydajności relacyjnych baz danych MySQL, PostgreSQL oraz Oracle z zastosowaniem bibliotek Doctrine**
    Marcin Choina, Maria Skublewska-Paszkowska.....250-257
14. **Analiza porównawcza narzędzi dedykowanych zarządzaniu projektami**
    Piotr Pawłowski; Małgorzata Plechawska-Wójcik.....258-264
15. **Analiza wydajności szkieletów programistycznych Laravel oraz Yii2 opartych na wzorcu architektonicznym MVC oraz języku PHP**
    Konrad Sławomir Węgrzecki, Mariusz Dzieńkowski.....265-272
16. **Analiza porównawcza języków programowania Java oraz Dart pod kątem przydatności do tworzenia aplikacji mobilnych**
    Łukasz Kozłowski, Grzegorz Kozieł.....273-279

Contents

1. **Learning speed or prediction accuracy? Comparative analysis of program-ming frameworks for artificial intelligence**
   Konrad Zdeb, Piotr Żukiewicz, Edyta Łukasik.....172-175
2. **Comparative analysis of the effectiveness of OWASP ZAP, Burp Suite, Nikto and Skipfish in testing the security of web applications**
   Aleksandra Kondraciuk, Aleksandra Bartos, Beata Pańczyk.....176-180
3. **Comparative analysis of the functionality and quality of the interface of chosen applications for ordering food**
   Maciej Gieroba, Marek Miłosz.....181-188
4. **Comparative analysis of the quality of recorded sound in the function of different recording formats**
   Andrzej Król, Tomasz Szymczyk.....189-194
5. **Comparison of the most popular operating systems in terms of functionalities**
   Jacek Lato, Marek Mucha, Tomasz Szymczyk.....195-202
6. **Analysis of the medical personnel's views on keeping records in an electronic form**
   Maciej Mikrut.....203-209
7. **Analysis of the functionality of voice and video communication systems**
   Aleksandra Piątkowska.....210-217
8. **Comparative analysis of the Cycles and Eevee graphics engines on the example of rendering 3D models of archaeological artifacts**
   Sebastian Dudek, Krzysztof Dziedzic.....218-223
9. **Comparative analysis of React, Next and Gatsby programming frameworks for creating SPA applications**
   Adam Świątkowski, Karol Ścibior.....224-227
10. **Performance comparison between selected chess engines**
    Maciej Sojka.....228-235
11. **Comparison of the offer of selected cloud service providers from the point of view of implementing IT projects based on open code**
    Jan Baran, Sławomir Przyłucki.....236-241
12. **Comparative analysis of reactive and imperative approach in Java web application development**
    Sebastian Iwanowski, Grzegorz Kozieł.....242-249
13. **Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine libraries**
    Marcin Choina, Maria Skublewska-Paszkowska.....250-257
14. **A comparative analysis of tools dedicated to project management**
    Piotr Pawłowski; Małgorzata Plechawska-Wójcik.....258-264
15. **Performance analysis of Laravel and Yii2 frameworks based on the MVC architectural pattern and PHP language**
    Konrad Sławomir Węgrzecki, Mariusz Dzieńkowski.....265-272
16. **Comparative analysis of Java and Dart programming languages in terms of suitability for creating mobile applications**
    Łukasz Kozłowski, Grzegorz Kozieł.....273-279

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 172–175
Received: 21 April 2022
Accepted: 5 May 2022

Learning speed or prediction accuracy? Comparative analysis of programming frameworks for artificial intelligence

Szybkość uczenia czy dokładność predykcji? Analiza porównawcza szkieletów programistycznych do sztucznej inteligencji

Konrad Zdeb*, Piotr Żukiewicz, Edyta Łukasik

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

Abstract
The purpose of the article is to analyze frameworks for artificial intelligence applications. In particular, the effectiveness, time-consumption and resources requirement. Linear regression, random forests and k nearest neighbors models were created for each framework. The learning data is a dataset containing informations about diamonds and their prices. Each model was designed to learn diamonds' prices and then make a prediction depending on its specific characteristics such as cut, color, and volume. The learning data was divided into sets of different sizes to show changes in a model depending on the amount of training data. Out of the three machine learning frameworks tested, TensorFlow proved to be the most accurate and SciKit-Learn the fastest.

Keywords: artificial intelligence; data prediction; framework

Streszczenie
Celem artykułu jest analiza szkieletów aplikacji do sztucznej inteligencji. Zbadane zostały: skuteczność, czasochłonność oraz ilość potrzebnych zasobów. Dla każdego frameworka stworzono modele regresji liniowej, lasów losowych i k najbliższych sąsiadów. Dane uczące to zbiory danych zawierające informację o diamencie oraz jego cenie. Każdy model miał za zadanie nauczyć się cen diamentów, a następnie dokonać predykcji w zależności od ich konkretnych cech tj. szlif, kolor, objętość. Dane uczące zostały podzielone na zbiory o różnej wielkości dzięki czemu można było zaobserwować zmianę w modelu w zależności od liczby danych treningowych. Z trzech przebadanych szkieletów programistycznych do uczenia maszynowego TensorFlow wykazał się największą skutecznością, a SciKit-Learn najkrótszym czasem dokonywania predykcji.

Słowa kluczowe: sztuczna inteligencja; predykcja danych; szkielety programistyczne

*Corresponding author
Email address: konrad.zdeb@pollub.edu.pl (K. Zdeb)

1. Wstęp

Uczenie maszynowe to jedna z odnóg sztucznej inteligencji, która cieszy się coraz większym zainteresowaniem [1]. Odpowiednio przygotowane modele potrafią przetworzyć dużą liczbę danych i „nauczyć się" przewidywać wyniki tych pobranych później. Algorytmów wykorzystywanych do uczenia maszynowego jest jednak wiele i nie zawsze wiadomo, który spełni oczekiwania użytkownika. Według autorów publikacji z tej dziedziny, którzy starają się nauczyć czytelnika wykorzystywania sztucznej inteligencji do swoich celów, różne szkielety odznaczają się cechami deklasującymi konkurencję [2] [3]. Trudno jest jednak jednoznacznie stwierdzić, który framework jest najlepszy, a bardzo prawdopodobne jest to, że nie istnieje taki, który jest bezsprzecznie najlepszy i deklasuje konkurencję.

Jednym z najważniejszych parametrów cechujących sztuczną inteligencję jest jej skuteczność. Określa ona, w jakim stopniu można być pewnym, że przewidziany przez algorytm wynik jest bliski prawdzie. W artykule zostały przedstawione trzy badane szkielety mające zastosowanie w sztucznej inteligencji. Są to: SciKit-Learn [4], PyTorch [5] i TensorFlow [6]. Każdy z badanych w artykule algorytmów posiada swój własny sposób liczenia jego trafności. Kolejnym czynnikiem, pod względem którego porównywane są klasyfikatory, jest czasochłonność. W pracy rozróżniono jej dwa rodzaje: czasochłonność uczenia i predykcji. Czasochłonność uczenia określa, w jakim czasie algorytm jest w stanie przetworzyć otrzymane dane oraz wyciągnąć z nich wnioski. Z kolei czasochłonność predykcji oznacza czas potrzebny algorytmowi na przetworzenie danej testującej i wskazanie wyniku na podstawie zebranych wcześniej danych. Podczas wykonywania skomplikowanych obliczeń na ogromnych zestawach danych bardzo ważna jest również kwestia wydajności sprzętowej. Przy porównywaniu algorytmów zwrócono uwagę na potrzebne do działania aplikacji zasoby procesora i pamięci systemowej.

Celem artykułu jest przetestowanie modeli utworzonych na podstawie szkieletów programistycznych do uczenia maszynowego, porównanie i wytypowanie najlepszych pod względem skuteczności, czasochłonności i potrzebnych zasobów. Ma to pomóc w wyborze frameworka najbardziej odpowiadającego potrzebom programisty i postawione-go przed nim zadania.

Przez autorów badań postawione zostały dwie hipotezy badawcze następującej treści:

H1: Model charakteryzujący się największą skutecznością pobiera jednocześnie najwięcej zasobów procesora ze wszystkich porównywanych sztucznych inteligencji.

H2: Żaden z modeli nie przoduje nad innym w każdej z kategorii jednocześnie.

Po przeprowadzeniu badań i na podstawie analizy otrzymanych wyników autorzy pracy wyciągną wnioski o ich prawdziwości lub nie.

2. Metodyka badań

2.1. Stanowisko badawcze

W celu uzyskania jak najbardziej wiarygodnych wyników pomiarowych badanie zostało przeprowadzone w odizolowanym środowisku maszyny wirtualnej. Zostało ono skonfigurowane w następujący sposób:

- na komputerze osobistym zainstalowano oprogramowanie VirtualBox w wersji 6.1.28;
- na maszynie wirtualnej zainstalowano system Windows 11 PRO w wersji 64-bit;
- na wirtualnym systemie zainstalowano Pythona w wersji 3.10.0.

2.2. Przygotowanie danych

Dla każdego szkieletu aplikacji dane były przygotowywane w identyczny sposób. Kod przygotowujący dane został przedstawiony na Listingu 1.

Pierwsze dwie linie oznaczają pobranie zbioru danych z pliku “diamonds_ready.csv”. Następnie usuwana jest kolumna “Unnamed: 0”, gdyż tworzona jest wraz ze zbiorem, a zbiór permutowany jest w celu zmienienia kolejności danych za pomocą metody *shuffle*. Python zapewnia bardzo szybkie i efektywne działanie na zbiorach. Wywołanie metody *iloc* dla zbioru pozwala na elastyczne wybieranie danych. W tym konkretnym przypadku pobierane są rekordy od indeksu 0 do indeksu *records_count*, który przyjmuje wartości 10000, 30000 albo 50000, w zależności od wielkości testowanej próbki. Dzięki wcześniejszemu przestawieniu dane różnią się od tych przy wcześniejszej próbie. Tak przygotowany zbiór należy rozdzielić na dane szukane „y”, w tym przypadku jest to kolumna „price” zawierająca ceny diamentów oraz dane opisujące tę cenę „X” tj. liczba karatów, szlif, kolor itd.

Proces uczenia polega na wprowadzaniu do algorytmu danych „X”. Algorytm następnie zwraca przewidywaną cenę, która porównywana jest z prawidłową wartością zapisaną w zbiorze „y”. Po procesie uczenia należy sprawdzić skuteczność danego modelu na danych, których nie miał okazji przetworzyć czyli na zbiorze testowym. W przypadku opisywanych badań dane zostały rozdzielone w proporcji 80% danych treningowych oraz 20% danych testowych. W ten sposób, dzięki metodzie train_test_split, w prosty sposób można przygotować dane. Przyjmuje ona jako parametr zbiory „X” i „y”, wielkość zbioru sprawdzającego oraz losową wartość całkowitą, która jest wykorzystana przy permutacji elementów.

Każda kolumna w zbiorze opisuje pojedynczą cechę danego diamentu co skutkuje tym, że ich wartości posiadają różne rzędy wielkości. Aby zniwelować błąd, który może sugerować modelowi, że większe wartości są bardziej znaczące niż mniejsze, wykonywany jest proces normalizacji. Klasa, która została wykorzystana w tym kodzie to *StandardScaler*. Najpierw za pomocą metody *fit_transform* obiekt analizuje dane, na których będzie operował, a następnie przeprowadza proces normalizacji na podstawie średniej i odchylenia standardowego. Tak przetworzone dane są gotowe do wykorzystania przez modele.

Listing 1: Przygotowanie danych do nauki

```
diamonds = pd.read_csv(r"C:\Users\piotr\OneDrive\Dokumenty\Studia\Magister
diamonds.drop(['Unnamed: 0'], axis=1, inplace=True)

diamonds = shuffle(diamonds)

diamonds = diamonds.iloc[:records_count, :]

X = diamonds.drop(['price'], axis=1)
y = diamonds['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=óó)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test  = sc.transform(X_test)
```

2.3. Proces uczenia modelu

Algorytm przetwarza dane w postaci kroków bądź epok [7]. Każdy krok polega na ciągłym uczeniu i sprawdzaniu czy dana, którą przewidział algorytm jest tą, której szukano. Jeżeli wykryty zostanie błąd model nanosi na siebie poprawki. Po wprowadzonych zmianach program sprawdza czy dla kolejnej danej przewidziany wynik jest poprawny. Proces ten powtarza się ustawioną przez programistę liczbę razy. Im dłużej model się uczy tym lepszą skuteczność powinien osiągnąć.

Na Rysunku 1 przedstawiony został wyświetlany przez Tensorflow proces uczenia.

```
Epoch 1/20
1067/1067 [==============================] - 3s 2ms/step - loss: 1382.2325 - mae: 1382.2325 - val_loss: 517.8335 - val_mae: 517.8335
Epoch 2/20
1067/1067 [==============================] - 2s 2ms/step - loss: 484.5997 - mae: 484.5997 - val_loss: 459.0359 - val_mae: 459.0359
Epoch 3/20
1067/1067 [==============================] - 2s 2ms/step - loss: 438.0830 - mae: 438.0830 - val_loss: 430.9727 - val_mae: 430.9727
Epoch 4/20
1067/1067 [==============================] - 2s 2ms/step - loss: 414.2224 - mae: 414.2224 - val_loss: 418.9761 - val_mae: 418.9761
Epoch 5/20
1067/1067 [==============================] - 2s 2ms/step - loss: 401.5052 - mae: 401.5052 - val_loss: 402.7807 - val_mae: 402.7807
Epoch 6/20
1067/1067 [==============================] - 2s 2ms/step - loss: 391.2015 - mae: 391.2015 - val_loss: 393.2258 - val_mae: 393.2258
Epoch 7/20
1067/1067 [==============================] - 2s 2ms/step - loss: 383.5373 - mae: 383.5373 - val_loss: 400.3570 - val_mae: 400.3570
Epoch 8/20
1067/1067 [==============================] - 2s 2ms/step - loss: 377.5865 - mae: 377.5865 - val_loss: 380.2634 - val_mae: 380.2634
Epoch 9/20
1067/1067 [==============================] - 2s 2ms/step - loss: 370.3160 - mae: 370.3160 - val_loss: 381.7056 - val_mae: 381.7056
Epoch 10/20
1067/1067 [==============================] - 2s 2ms/step - loss: 366.0886 - mae: 366.0886 - val_loss: 376.7699 - val_mae: 376.7699
Epoch 11/20
1067/1067 [==============================] - 2s 2ms/step - loss: 364.3582 - mae: 364.3582 - val_loss: 385.8737 - val_mae: 385.8737
Epoch 12/20
1067/1067 [==============================] - 2s 2ms/step - loss: 360.8728 - mae: 360.8728 - val_loss: 379.8013 - val_mae: 379.8013
Epoch 13/20
1067/1067 [==============================] - 2s 2ms/step - loss: 358.3898 - mae: 358.3898 - val_loss: 363.5269 - val_mae: 363.5269
Epoch 14/20
1067/1067 [==============================] - 2s 2ms/step - loss: 357.0932 - mae: 357.0932 - val_loss: 366.7127 - val_mae: 366.7127
Epoch 15/20
1067/1067 [==============================] - 2s 2ms/step - loss: 352.9424 - mae: 352.9424 - val_loss: 370.2906 - val_mae: 370.2906
Epoch 16/20
1067/1067 [==============================] - 2s 2ms/step - loss: 351.8788 - mae: 351.8788 - val_loss: 364.5327 - val_mae: 364.5327
Epoch 17/20
1067/1067 [==============================] - 2s 2ms/step - loss: 352.8774 - mae: 352.8774 - val_loss: 366.8039 - val_mae: 366.8038
Epoch 18/20
1067/1067 [==============================] - 2s 2ms/step - loss: 349.5326 - mae: 349.5326 - val_loss: 358.6613 - val_mae: 358.6613
Epoch 19/20
1067/1067 [==============================] - 2s 2ms/step - loss: 346.8861 - mae: 346.8861 - val_loss: 367.3414 - val_mae: 367.3414
Epoch 20/20
1067/1067 [==============================] - 2s 2ms/step - loss: 346.2907 - mae: 346.2907 - val_loss: 364.5911 - val_mae: 364.5911
```

Rysunek 1: Przykładowy proces uczenia.

Ten model trenowany był przez 20 epok, a każda z nich zajęła mu od 2 do 3 sekund. Błąd oznaczany jest

przez „mae” (ang. mean absolute error), czyli średni błąd bezwzględny mówiący o tym, jakiego odchylenia od poszukiwanej wartości można się spodziewać. Im mniejszy jest dany błąd tym bardziej dokładny jest model. Przedstawiona na rysunku 1 sztuczna inteligencja zaczęła z bardzo wysokim błędem rzędu 1300, jednak kilka epok później zmniejszyła go do około 300. Zmieniając liczbę epok oraz liczbę rekordów w każdej epoce można usprawniać ten wynik, jednak w tym badaniu użyto najbardziej powszechnych metod i wartości.

2.4. Wyliczanie skuteczności

Skuteczność liczona jest za pomocą udostępnianych przez szkielety aplikacji metod, które przyjmowały jako dane wejściowe zbiór cen, które przewidział model oraz zbiór cen, które powinien przewidzieć (rzeczywistych) [8]. Dla SciKit-Learn i PyTorch wykorzystana została funkcja, która zwraca współczynnik determinacji R2, jako liczbę z zakresu od zera do jeden. Im bliżej jedynki jest dana skuteczność, tym dokładniejszy jest model. Dla TensorFlow wykorzystano funkcję „mean absolute percentage error”, która jest miarą dokładności przewidywania metody prognozowania w statystyce. Ten wariant metody zwraca wartość procentową.

2.5. Wyliczanie zużycia procesora i pamięci

Aby sprawdzić zużycie procesora i pamięci, przed i po procesie uczenia, pobierana jest informacja o aktualnym wykorzystaniu obu tych zasobów. Z uwagi na to, że w tle nie działa żaden inny mocno obciążający system proces, można przyjąć, że każde większe różnice spowodowane są przez badany model. Następnie obie te wartości są porównywane i ustala się stopień zużycia procesora i pamięci komputera.

3. Przebieg badania

Pierwszym krokiem w przeprowadzeniu badania jest trening aplikacji przygotowanymi wcześniej danymi. W tym celu każdy z klasyfikatorów poddano trzem sesjom treningowym; zawierającym 10000, 30000 oraz 50000 danych wejściowych. Algorytmem, który wykorzystano podczas uczenia, był algorytm regresji liniowej. Następnie klasyfikatory poddano próbie, której celem było ustalenie ich skuteczności. Każda aplikacja wykorzystywała własny sposób liczenia skuteczności, charakterystyczny dla użytego szkieletu programistycznego. Dodatkowo, zarówno podczas uczenia, jak i testowania, aplikacje zwracały czas potrzebny im do wykonania danych akcji oraz zużyte zasoby systemowe w postaci pamięci i mocy obliczeniowej procesora.

Model regresji liniowej zakłada, że dla zbioru danych zaobserwowanych $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^{n}$ istnieje liniowa relacja między zmienną zależną y_i a wektorem $p \times 1$ regresorów x_i. Przez uwzględnienie składnika losowego ε_i można zamodelować zależność postaci:

$$y_i = \beta_0 1 + \beta_1 \beta x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = x_i^T \beta + \varepsilon_i, \quad i = 1, \dots, n, \qquad (1)$$

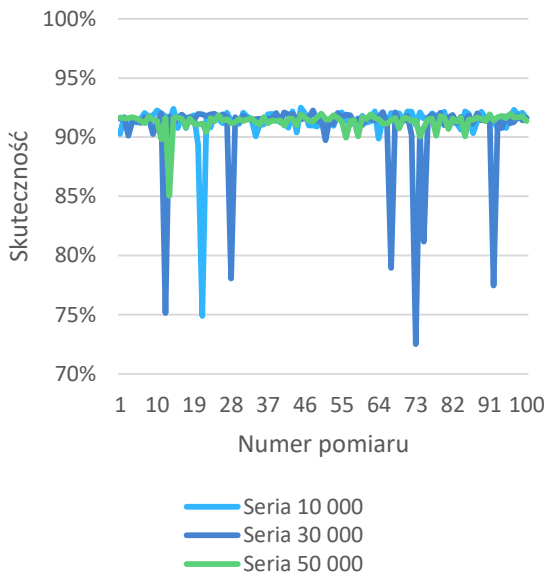
gdzie ^{T} oznacza transpozycję; $x_i^T \beta$, to iloczyn skalarny x_i i β.

Ponieważ algorytm regresji liniowej zawiera w sobie czynnik pseudolosowy, to każdorazowe pobieranie do badania losowej próbki z całości nie ma wpływu na wynik.

4. Wyniki

Podczas analizy statystycznej na danych wyjściowych dokonano działań mających na celu umożliwienie przedstawienia wyników w czytelny i zrozumiały dla czytelnika sposób oraz wyciągnięcie wniosków pozwalających na potwierdzenie bądź obalenie hipotez badawczych.

Badania wykazały, że skuteczność algorytmu SciKit-Learn wynosi średnio 91,09%. Średnia skuteczność algorytmu podczas badania klasyfikatorów wytrenowanych dla 10 000 danych wejściowych wynosi 91,30%, dla 30 000 – 90,67%, a dla 50 000 – 91,31% (Rysunek 2).

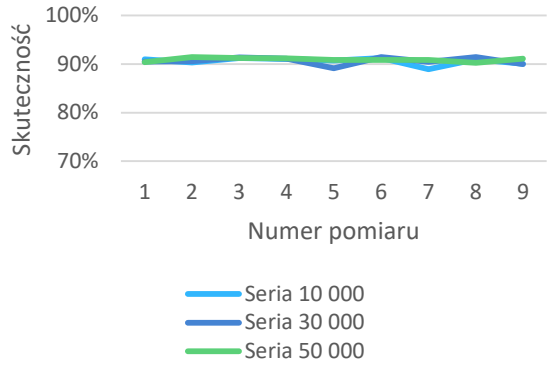


Rysunek 2: Skuteczność SciKit-Learn

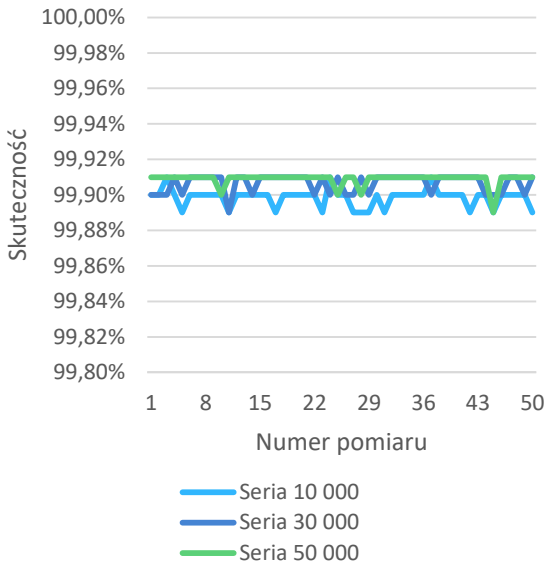
Badania wykazały, że skuteczność algorytmu PyTorch wynosi średnio 90,71%. Średnia skuteczność algorytmu podczas badania klasyfikatorów wytrenowanych dla 10 000 danych wejściowych wynosi 90,60%, dla 30 000 – 90,66%, a dla 50 000 – 90,88% (Rysunek 3).

Badania wykazały, że skuteczność algorytmu TensorFlow wynosi średnio 99,90%. Średnia skuteczność algorytmu podczas badania klasyfikatorów wytrenowanych dla 10 000 danych wejściowych wynosi 90,90% oraz 90,91% dla klasyfikatorów, których liczba danych wejściowych wynosiła 30 000 oraz 50 000 (Rysunek 4).

W Tabeli 1 zestawiono wszystkie wyniki dla poszczególnych używanych w badaniach szkieletów programistycznych.



Rysunek 3: Skuteczność PyTorch.



Rysunek 4: Skuteczność TensorFlow.

Tabela 1: Zestawienie wyników wszystkich algorytmów

	Średnia			
Algorytm	Skuteczność [%]	Czasochłonność [s]	Zużycie procesora [%]	Zużycie pamięci [%]
SciKit-Learn	91,09	0,04	14,64	-0,004
PyTorch	90,71	474,91	-24,22	-0,04
TensorFlow	99,90	68,55	-5,22	0,26

5. Wnioski

Badania wykazały, że największą skutecznością w dokonywaniu predykcji charakteryzuje się szkielet programistyczny TensorFlow o średniej trafności na poziomie 99,90%. Jego średni sumaryczny czas trenowania i dokonywania predykcji wynosi 68,55s. Jest to czas znacznie większy od algorytmu SciKit-Learn, ale ponad 6 krotnie mniejszy od algorytmu PyTorch. Plasuje to algorytm TensorFlow na drugim miejscu pod względem czasochłonności, co dowodzi nieprawdziwości hipotezy badawczej H1: Model charakteryzujący się największą skutecznością potrzebuje jednocześnie najwięcej czasu na wykonanie operacji ze wszystkich porównywanych sztucznych inteligencji.

Badania wykazały, że największą oszczędnością czasu charakteryzuje się szkielet programistyczny SciKit-Learn o średniej sumie czasów trenowania i dokonywania predykcji na poziomie 0,04s (Tabela 1). Jednoczesna dominacja szkieletu TensorFlow w skuteczności dokonywania predykcji dowodzi słuszności hipotezy badawczej H2: Nie istnieje model, który jest najlepszy w każdej kategorii.

W toku prowadzonych badań wyciągnięto wnioski, że poziom zużycia zasobów procesora i pamięci systemowej przez algorytmy jest znikomy. Pomimo upewnienia się, że podczas badania nie pracowały żadne zbędne programy, niezależne od użytkownika procesy systemowe powodowały znaczne wypaczenie pomiarów. Ujemne pomiary zużycia procesora i pamięci systemowej były prawdopodobnie spowodowane zamknięciem działającego w tle procesu systemowego, które zostało odczytane przez algorytm jako zysk, czyli ujemne zużycie.

Podsumowując, z trzech przebadanych szkieletów programistycznych do uczenia maszynowego, pod uwagę warto brać tylko dwa. Algorytm PyTorch wykazał się jednocześnie najmniejszą skutecznością jak i największą czasochłonnością. Użytkownicy oczekujący otrzymania rezultatów natychmiast lub ci, którzy potrzebują przeprocesować ogromną liczbę danych w rozsądnym czasie, powinni wybrać algorytm SciKit-Learn. Z kolei programiści, którym przede wszystkim zależy na bezbłędności dokonywanych szacunków, powinni zdecydować się na algorytm TensorFlow.

Literatura

[1] X. D. Zhang, A Matrix Algebra Approach to Artificial Intelligence, Springer, 2020.

[2] J. Brownlee, Deep learning for computer vision: image classification, object detection, and face recognition in python, Machine Learning Mastery, 2019.

[3] R. Garreta, G. Moncecchi, Learning scikit-learn: machine learning in python, PACKT Publishing Ltd., 2013.

[4] G. Hackeling, Mastering Machine Learning with scikit-learn, PACKT Publishing Ltd., 2017.

[5] E. Stevens, L. Antiga, T. Viehmann, Deep learning with PyTorch, Manning Publications, 2020.

[6] N. McClure, TensorFlow machine learning cookbook, PACKT Publishing Ltd., 2017.

[7] D. Sarkar, R. Bali, T. Sharma, Practical machine learning with Python. A Problem-Solvers Guide To Building Real-World Intelligent Systems, Apress, Berkeley, CA, 2018.

[8] S. Raschka, V. Mirjalili, Python Machine Learning: Machine Learning and Deep Learning with Python. Scikit-Learn, and TensorFlow, Second edition, PACKT Publishing Ltd., 2017.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 176–180
Received: 23 April 2022
Accepted: 5 May 2022

Comparative analysis of the effectiveness of OWASP ZAP, Burp Suite, Nikto and Skipfish in testing the security of web applications

Analiza porównawcza skuteczności narzędzi OWASP ZAP, Burp Suite, Nikto i Skipfish w testowaniu bezpieczeństwa aplikacji internetowych

Aleksandra Bartos, Aleksandra Kondraciuk*, Beata Pańczyk

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

Abstract
Application security is one of the key aspects necessary for its proper functioning. Security verification consists primarily in conducting regular penetration tests and checking the vulnerability of the application to various types of attacks. The recommended solution is to use tools dedicated to detecting security holes in applications. Choosing the right tool from among those available on the market can be difficult. This article presents a comparative analysis of the effectiveness of popular application security testing tools in terms of the number of detected vulnerabilities. The analysis was based on the obtained results of scanning two Internet applications containing a number of security vulnerabilities, used to learn ethical hacking.

Keywords: application security; penetration tests; testing tools

Streszczenie
Bezpieczeństwo aplikacji jest jednym z kluczowych aspektów, niezbędnych do jej prawidłowego funkcjonowania. Weryfikacja bezpieczeństwa polega przede wszystkim na przeprowadzaniu regularnych testów penetracyjnych i sprawdzaniu podatności aplikacji na różnego rodzaju ataki. Zalecanym rozwiązaniem jest skorzystanie z narzędzi dedykowanych do wykrywania luk bezpieczeństwa w aplikacjach. Wybór odpowiedniego narzędzia spośród dostępnych na rynku może okazać się trudny. W niniejszym artykule dokonano analizy porównawczej skuteczności popularnych narzędzi testowania bezpieczeństwa aplikacji pod kątem liczby wykrywanych zagrożeń. Analizę oparto o otrzymane wyniki skanowania dwóch aplikacji internetowych, zawierających szereg luk bezpieczeństwa, służących do nauki etycznego hackingu.

Słowa kluczowe: bezpieczeństwo aplikacji; testy penetracyjne; narzędzia do testowania

*Corresponding author
Email address: aleksandra.kondraciuk@pollub.edu.pl (A. Kondraciuk)

1. Wstęp

Testowanie aplikacji internetowych jest ważnym etapem w trakcie ich tworzenia jak i utrzymywania. Rodzaj testów jest zazwyczaj dobierany indywidualnie do potrzeb projektu. Jednymi z najbardziej popularnych typów są testy manualne i automatyczne. Jednak samo działanie aplikacji nie jest jedynym priorytetowym aspektem. Dlatego warto poświęcić więcej uwagi testom bezpieczeństwa.

Wraz z ciągłym rozwojem technologii rośnie liczba incydentów bezpieczeństwa. Stanowi to poważny problem, ponieważ może spowodować utratę wrażliwych danych z aplikacji i ich upowszechnienie. Użytkownik po tego typu zdarzeniu może całkowicie stracić zaufanie do aplikacji i przestać jej używać.

Testy penetracyjne [1] umożliwiają kontrolowane ataki na aplikację internetową, w celu wykrycia jej potencjalnych luk. Tego typu testy muszą być wykonywane cyklicznie z dużą dbałością o dokładność analizy, co może stać się monotonne. Dlatego warto zwrócić uwagę na dostępne rozwiązania do skanowania bezpieczeństwa aplikacji. Ważny jest odpowiedni wybór narzędzia, ponieważ każde z nich, mimo podobnych funkcjonalności oferuje inny zakres wykrywanych zagrożeń czy szybkości wykonywania skanu.

Celem artykułu jest przeprowadzanie testów penetracyjnych za pomocą narzędzi OWASP ZAP, Burp Suite, Nikto i Skipfish oraz ocena ich skuteczności w wykrywaniu zagrożeń bezpieczeństwa aplikacji.

Wybrane do badań narzędzia OWASP ZAP oraz Burp Suite są jednymi z najbardziej popularnych rozwiązań w dziedzinie testów bezpieczeństwa aplikacji internetowych [2]. Oferują zaawansowany interfejs użytkownika z rozbudowanymi funkcjonalnościami, które można dopasować do wymagań użytkownika i skanowanej aplikacji. Oba narzędzia pozwalają na wykrycie najważniejszych luk bezpieczeństwa. Do narzędzi reprezentujących nieco prostszą obsługę jak i mniejszy zakres funkcjonalności należą Nikto [3] oraz Skipfish [4]. Są to rozwiązania, których obsługa opiera się na komendach wpisywanych z poziomu linii poleceń danego systemu.

OWASP ZAP oferuje szeroką gamę funkcjonalności oraz jest powszechnie określany jako jeden z flagowych projektów organizacji OWASP, działającej na rzecz poprawy bezpieczeństwa oprogramowania [5], co pozwala na postawienie następującej tezy badawczej:

OWASP ZAP osiąga najlepsze wyniki spośród badanych narzędzi.

2. Przegląd literatury

Dokonanie wyboru narzędzi i aplikacji internetowych do przeprowadzenia badań wymagało zapoznania się z pozycjami literaturowymi opisującymi kwestie związane z bezpieczeństwem aplikacji internetowych oraz sposobami wykrywania zagrożeń. Zasady działania oraz scenariusze zastosowań testów penetracyjnych zostały dokładnie przedstawione w artykule [1]. Przeprowadzanie tego rodzaju testów wymaga pełnych uprawnień do zarządzania badaną aplikacją ze względu na możliwość wprowadzenia w niej poważnych uszkodzeń. Dlatego też konieczne było lokalne uruchomienie specjalnie do tego przeznaczonych aplikacji. Dostępne tego typu rozwiązania i ich dokumentacja została przedstawiona w pozycjach [6] oraz [7]. Opracowanie wyników skanowania wymaga wiedzy na temat badanego narzędzia oraz umiejętności porównywania otrzymanych wartości, dlatego też pomocna okazała się treść artykułu [3] zawierającego analizę wyników testów przeprowadzonych m.in. za pomocą narzędzi OWASP ZAP i Nikto. Autorzy artykułu dokonali porównania ww. narzędzi poprzez określanie czy dane narzędzie wykryło wszystkie z wyznaczonych 13 luk bezpieczeństwa. Otrzymane w ten sposób wyniki nie pozwoliły na wskazanie jednoznacznie skuteczniejszego narzędzia.

3. Narzędzia

Narzędzia, których skuteczność zostanie zbadana są popularnymi, darmowymi rozwiązaniami w dziedzinie testowania bezpieczeństwa aplikacji internetowych [2]. OWASP ZAP oraz Burp Suite to przykłady narzędzi z graficznym interfejsem użytkownika, które oferują zaawansowane ustawienia trybu skanowania aplikacji, natomiast narzędzia Nikto oraz Skipfish, uruchamiane są z poziomu linii poleceń systemu i nie oferują możliwości wyboru trybu ataku.

3.1. OWASP ZAP

OWASP ZAP (ang. Open-Source Web Application Security Scanner Zed Attack Proxy) to jeden z flagowych projektów organizacji OWASP, będący prostym w użyciu, darmowym skanerem źródłowym, służącym do wyszukiwania luk bezpieczeństwa w aplikacjach internetowych [5]. Wykryte luki narzędzie prezentuje wraz z opisem, liczbą i miejscem wystąpienia oraz propozycją rozwiązania. Dodatkowo generuje szczegółowy raport z przeprowadzonych testów.

3.2. Burp Suite

Burp Suite to produkt firmy PortSwigger służący do przeprowadzania zaawansowanych testów bezpieczeństwa aplikacji. Dostępne są trzy wydania narzędzia: Community, Enterprise i Professional. Rozbudowany skaner bezpieczeństwa pozwala na dopasowanie konfiguracji przeprowadzanych testów przez użytkownika. Dla każdej znalezionej luki zawarta jest informacja m.in. o liczbie oraz miejscu występowania, typie problemu, poziomie niebezpieczeństwa [8]. Po zakończonej weryfikacji użytkownik ma możliwość wygenerowania raportu z przeprowadzonego skanu.

3.3. Nikto

Narzędzie Nikto jest darmowym oprogramowaniem, uruchamianym z poziomu wiersza poleceń, służącym do automatycznego testowania podatności serwerów internetowych [3]. Narzędzie sprawdza występowanie w aplikacji szkodliwych plików, czy problemów z wersjami. Bada także możliwe luki spowodowane błędną konfiguracją zabezpieczeń, prowadzącą do nieautoryzowanego dostępu lub modyfikacji poufnych informacji. Nikto jest zaimplementowany w języku Perl, co umożliwia rozszerzanie narzędzia o dodatkowe funkcjonalności za pomocą wtyczek.

3.4. Skipfish

Skipfish jest narzędziem typu Open-Source do skanowania bezpieczeństwa aplikacji internetowych oraz serwerów. Uruchamiany jest za pomocą wiersza poleceń. Łatwość obsługi i szybkość działania sprawia, że jest przystępnym narzędziem nawet dla początkującego użytkownika. Umożliwia znalezienie luk bezpieczeństwa, tj. możliwość wstrzyknięcia komend systemowych, zapytań SQL, żądań HTTP. Po skończonej weryfikacji generowany jest raport z otrzymanymi wynikami, w którym znalezione błędy kategoryzowane są według stopnia ryzyka [4].

4. Badane aplikacje

Do weryfikacji skuteczności wytypowanych narzędzi zostały wybrane dwie aplikacje internetowe bWAPP [6] oraz Mutillidae [7]. Są one darmowymi projektami stworzonymi na potrzeby nauki testów bezpieczeństwa. Umożliwiają przeprowadzenie szerokiego zakresu testów pasywnych jak i aktywnych [1]. Tego typu testy mogą spowodować realne uszkodzenia w aplikacji, dlatego nie mogłyby zostać przeprowadzone na publicznie dostępnych stronach.

4.1. bWAPP

bWAPP (ang. a buggy Web APPlication) jest darmową aplikacją internetową zawierającą wiele luk bezpieczeństwa [9]. Możliwe jest jej pobranie do użytku lokalnego, jak i w wersji obrazu dysku wirtualnego do instalacji na maszynie wirtualnej. Rozwiązanie jest skierowane do początkującej jak i zaawansowanej grupy odbiorców zajmujących się przeprowadzaniem testów penetracyjnych.

4.2. Mutillidae

OWASP Mutillidae II to darmowa, otwarta, celowo podatna na ataki aplikacja internetowa, będąca dobrym środowiskiem badawczym dla osób wykonujących testy penetracyjne. Mutillidae można zainstalować w systemach Linux i Windows za pomocą LAMP, WAMP i XAMMP. Zagrożenia bezpieczeństwa występujące w aplikacji pochodzą z listy zagrożeń OWASP TOP 10, czyli standardowego dokumentu informacyjnego, prze-

znaczonego dla programistów i osób wykonujących testy bezpieczeństwa aplikacji internetowych. Lista reprezentuje szeroki zakres najważniejszych zagrożeń bezpieczeństwa aplikacji internetowych [4].

5. Kryteria badawcze i klasyfikacja zagrożeń

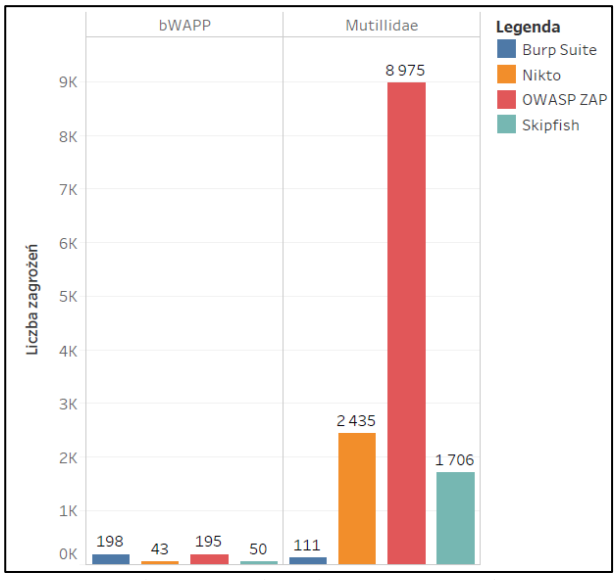
Lista powszechnych zagrożeń bezpieczeństwa aplikacji internetowych stanowi spis kryteriów badawczych, pod kątem których sprawdzano skuteczność badanych narzędzi. Są to luki bezpieczeństwa takie jak podatność na ataki XSS, SQL Injection, braki odpowiednich nagłówków, dostęp do wrażliwych danych, czy elementów struktury aplikacji. Klasyfikacja wykrytego zagrożenia pozwala na łatwy przekaz priorytetu znalezionej luki. Umożliwia to zaplanowanie odpowiednich akcji w celu eliminacji potencjalnych zagrożeń bezpieczeństwa opierając się na skali możliwych incydentów. Zagrożenia z reguły dzielą się na cztery kategorie: wysokie, średnie, niskie i informacyjne.

6. Metoda badań i środowisko badawcze

Eksperyment badawczy rozpoczęto od przygotowania środowisk badawczych oraz zaplanowania kolejności i sposobu przeprowadzania testów. Każde z narzędzi wykonało skan na dwóch uruchomionych lokalnie aplikacjach. Otrzymane wyniki zaprezentowano w tabelach, a następnie przedstawiono w formie wykresów i na ich podstawie dokonano analizy skuteczności badanych narzędzi.

7. Analiza wyników

Otrzymane wyniki badań zostały zestawione w formie wykresu oraz tabel. Wykres zaprezentowany na rysunku 1 przedstawia sumy wystąpień wszystkich wykrytych luk bezpieczeństwa przez poszczególne narzędzia w dwóch aplikacjach internetowych.



Rysunek 1: Wykres sumy wykrytych zagrożeń przez wybrane narzędzia podczas aktywnych skanów aplikacji bWAPP i Mutillidae.

Z wykresu wynika, że najlepiej pod względem liczby wykrytych zagrożeń wypadło narzędzie OWASP ZAP, ponieważ w aplikacji Mutillidae wykryło prawie 9 000 luk bezpieczeństwa, natomiast w aplikacji bWAPP 195, czyli jedynie 3 zagrożenia mniej niż narzędzie Burp Suite.

Podczas analizy wyników konieczne było jednak zweryfikowanie nie tylko sumy wykrytych zagrożeń, ale także ich rodzaju. W tym celu zestawiono wspólne zagrożenia bezpieczeństwa pod względem poziomu oraz liczby wystąpień.

Do przykładowych zagrożeń wykrytych przez każde z badanych narzędzi należą:

- **SQL Injection** - metoda ataku polegająca na wstrzyknięciu złośliwego zapytania do bazy danych, umożliwiająca manipulację jej zawartością.
- **XSS** (ang. Cross-Site Scripting) - atak polegający na wstrzykiwaniu skryptów, umożliwiających np. przejmowanie sesji użytkowników lub na przekierowywanie ich na złośliwe witryny.
- **Path Traversal** - obchodzenie ścieżki to technika ataku pozwalająca na uzyskanie dostępu do zawartości plików, katalogów i poleceń znajdujących się poza głównym katalogiem aplikacji poprzez manipulację adresem URL.
- **Directory Browsing** - możliwość podglądu zawartości katalogów. Pomaga to atakującemu w dostępie do ukrytych skryptów, plików źródłowych i poufnych informacji.
- **Absence of Anti-CSRF Tokens** - podatność aplikacji umożliwiająca zmuszenie użytkownika do wysłania żądania HTTP do miejsca docelowego bez jego wiedzy i przeprowadzenie ataku jako ofiara.
- **Cookie without Flags** - brak ustawienia odpowiednich flag bezpieczeństwa umożliwia manipulację zawartością pliku, przejęcie sesji lub przekierowanie ofiary do złośliwej aplikacji.
- **X-Content-Type-Options Header Missing** - nieodpowiednie ustawienie nagłówka X-Content-Type-Options może doprowadzić do nieprawidłowej interpretacji typu zawartości odpowiedzi z aplikacji.
- **Strict-Transport-Security Header** - brak nagłówka Stric-Transport-Security wymuszającego dostęp do aplikacji tylko za pośrednictwem protokołu HTTPS może powodować podatność witryny na ataki typu man-in-the-middle.

Podczas testów każde z narzędzi wykryło luki bezpieczeństwa, które nie są wspólnymi kryteriami skanowania pozostałych narzędzi. Ich analiza była konieczna do dokonania rzetelnej oceny skuteczności narzędzi.

Kluczowe zagrożenia wykryte przez OWASP ZAP:

- Missing Anti-clicjacking Header - odpowiedź nie obejmuje odpowiednich nagłówków zapewniających ochronę przed atakami typu „ClickJacking”.
- CSP: Wildcard Directive - dyrektywy tj. script-src, style-src, connects-src, form-action zezwalają na źródła wieloznaczne, nie są zdefiniowane lub są zdefiniowane zbyt szeroko.
- RFI (ang. Remote File Inclusion) - technika ataku używana do wykorzystywania mechanizmów „dynamicznego dołączania plików” w aplikacjach internetowych.

Kluczowe zagrożenia wykryte przez Burp Suite:

- XPath injection - metoda ataku polegająca na wstrzyknięciu prostych zapytań XPath (ang. XML Path Language) do dokumentów HTML i odczyt wrażliwych danych.
- LDAP injection - atak polegający na wstrzyknięciu danych przy pomocu protokołu LDAP (ang. Lightweight Directory Access Protocol) w celu uzyskania dostępu do poufnych danych.
- Server-side template injection - metoda ataku polegająca na wstrzyknięciu szablonu po stronie serwera w celu manipulowania działaniem aplikacji.

Kluczowe zagrożenia wykryte przez Nikto

- Component is in a non-updated version - przestarzałe komponenty używane przez aplikację są podatne na różnego rodzaju ataki.
- Allowed phpinfo() to be run, which gives detailed system information - funkcja phpinfo() wyświetla wrażliwe informacje m.in. o wersji PHP, serwerze, ścieżkach, lokalnych opcjach konfiguracyjnych, nagłówkach HTTP oraz licencji.
- HTTP TRACE method is active, suggesting the host is vulnerable to XST - atak XST (ang. Cross-Site Tracing) obejmuje użycie (XSS) oraz metod TRACE lub TRACK HTTP.

Kluczowe zagrożenia wykryte przez Skipfish

- Shell injection vector to metoda ataku polegająca na wstrzyknięciu dowolnych poleceń systemu operacyjnego za pomocą podatnych aplikacji.
- Incorrect caching directives - nieprawidłowo ustawione dyrektywy nagłówka HTTP Cache-Control. Odpowiedzialne są za kontrolowanie buforowania w aplikacjach i współdzielenia pamięci podręcznej.
- HTML form with no apparent XSRF protection - brak ochrony przed atakami typu XSRF (ang. Cross-site request forgery) w fotmularzach dostępnych w aplikacji. Podatność ma na celu wykonanie przez ofiarę niepożądanych akcji, które mogą doprowadzić do kradzieży danych.

W tabelach 2 i 3 przedstawiono wspólnie wykryte zagrożenia przez poszczególne narzędzia w aplikacjach Mutillidae oraz bWAPP. Znak „-” w komórkach oznacza, że dane zagrożenie nie jest wykrywane przez narzędzie. Natomiast wartość „0” oznacza, że narzędzie skanuje aplikację pod tym kątem, ale zagrożenie nie zostało wykryte.

Tabela 2: Liczba wystąpień poszczególnych zagrożeń wykrytych podczas skanowania aplikacji Mutillidae

Zagrożenie	Aplikacja Mutillidae							
	OWASP ZAP		Burp Suite		Nikto		Skipfish	
	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień
SQL Injection	wysoki	6	wysoki	8	wysoki	1	wysoki	96
Client-side SQL injection (DOM-based)	wysoki	13	wysoki	0	-	-	-	-
Cross-site scripting (DOM-based)	wysoki	3	wysoki	2	wysoki	0	wysoki	30
Cross Site Scripting (Persistent)	wysoki	3	wysoki	0	-	-	-	-
Cross Site Scripting (Reflected)	wysoki	100	wysoki	30	-	-	-	-
Path Traversal	wysoki	35	wysoki	3	-	-	-	-
Remote OS Command Injection	wysoki	3	wysoki	0	-	-	-	-
Vulnerable JS Library	średni	4	średni	0	-	-	-	-
Directory Browsing	średni	66	średni	0	średni	0	średni	398
Client-side HTTP parameter pollution	średni	76	niski	4	-	-	-	-
Absence of Anti-CSRF Tokens	niski	1570	niski	0	-	-	-	-
Cookie no HttpOnly Flag	niski	227	niski	1	niski	2	-	-
Cookie Without Secure Flag	niski	214	niski	0	niski	2	-	-
Cookie without SameSite Attribute	niski	5	niski	0	-	-	-	-
Loosely Scoped Cookie	niski	0	niski	0	-	-	-	-
Incomplete/No Cache-control Header Set	niski	1295	info	1	-	-	-	-
X-Content-Type-Options Header Missing	niski	2120	niski	0	-	-	-	-
X-Frame-Options Header Not Set	niski	0	info	2	niski	1	-	-
Strict-Transport-Security Header	niski	0	niski	1	niski	1	-	-
Retrieved x-powered-by header	niski	0	-	-	niski	1	-	-
Open redirection (DOM-based)	niski	0	niski	2	-	-	-	-
Cross-domain script include	niski	0	info	0	-	-	-	-
Secure Pages Iclude Mixed Content	niski	0	info	0	-	-	-	-
Charset Mismatch	info	139	info	0	-	-	niski	209
Private IP addresses disclosed	niski	30	info	2	-	-	-	-
File upload	info	0	info	1	-	-	-	-
Email addresses disclosed	info	0	info	1	-	-	-	-

Tabela 3: Liczba wystąpień poszczególnych zagrożeń wykrytych podczas skanowania aplikacji bWAPP

Zagrożenie	Aplikacja bWAPP							
	OWASP ZAP		Burp Suite		Nikto		Skipfish	
	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień
SQL Injection	wysoki	0	wysoki	6	wysoki	0	wysoki	0
Client-side SQL injection (DOM-based)	wysoki	0	wysoki	0	-	-	-	-
Cross-site scripting (DOM-based)	wysoki	0	wysoki	0	wysoki	1	wysoki	0

Cross Site Scripting (Persistent)	wysoki	0	wysoki	5	-	-	-	-
Cross Site Scripting (Reflected)	wysoki	0	wysoki	14	-	-	-	-
Path Traversal	wysoki	0	wysoki	0	-	-	-	-
Remote OS Command Injection	wysoki	0	wysoki	4	-	-	-	-
Vulnerable JS Library	średni	1	średni	1	-	-	-	-
Directory Browsing	średni	4	średni	0	średni	7	średni	16
Client-side HTTP parameter pollution	średni	7	niski	0	-	-	-	-
Absence of Anti-CSRF Tokens	niski	5	niski	9	-	-	-	-
Cookie no HttpOnly Flag	niski	5	niski	2	niski	7	-	-
Cookie Without Secure Flag	niski	0	niski	0	niski	7	-	-
Cookie without SameSite Attribute	niski	5	niski	0	-	-	-	-
Loosely Scoped Cookie	niski	0	niski	0	-	-	-	-
Incomplete/No Cache-control Header Set	niski	0	info	1	-	-	-	-
X-Content-Type-Options Header Missing	niski	86	niski	0	-	-	-	-
X-Frame-Options Header Not Set	niski	0	info	2	niski	1	-	-
Strict-Transport-Security Header	niski	0	niski	1	niski	1	-	-
Retrieved x-powered-by header	niski	0	-	-	niski	1	-	-
Open redirection (DOM-based)	niski	0	niski	0	-	-	-	-
Cross-domain script include	niski	0	info	0	-	-	-	-
Secure Pages Iclude Mixed Content	niski	0	info	0	-	-	-	-
Charset Mismatch	info	0	info	0	-	-	niski	4
Private IP addresses disclosed	niski	0	info	0	-	-	-	-
File upload	info	0	info	0	-	-	-	-
Email addresses disclosed	info	0	info	0	-	-	-	-

8. Wnioski

Badania wytypowanych narzędzi pozwoliły na ocenę ich skuteczności. Określono ją na podstawie liczby wykrytych luk bezpieczeństwa, ich rodzaju, liczby wystąpień oraz poziomu zagrożenia jakie stanowią.

Potwierdzona została teza, że najskuteczniejszym narzędziem spośród grupy analizowanych, okazało się narzędzie OWASP ZAP, ze względu na dużą liczbę wykrytych zagrożeń wysokiego i średniego poziomu w każdej z aplikacji.

Nieco gorsze, ale znaczące wyniki osiągnęło narzędzie Burp Suite, ponieważ wykryło znaczną liczbę zagrożeń wysokiego poziomu w obu aplikacjach.

Narzędzia Nikto oraz Skipfish wykryły znacząco mniej zagrożeń niż OWASP ZAP i Burp Suite.

Poza wspólnie wykrytymi zagrożeniami narzędzia OWASP ZAP oraz Burp Suite wyłoniły także inne istotne podatności aplikacji. Luki wykryte przez Nikto i Skipfish nie stanowią przykładu poważnych zagrożeń.

Porównując otrzymane wyniki do wniosków z artykułu [3] można ocenić wybrany sposób badań skuteczności narzędzi za właściwy. Dzięki skupieniu się na pełnym zakresie otrzymanych wyników, a nie na ograniczonej liczbie kryteriów możliwe było wytypowanie najskuteczniejszego z wybranych narzędzi.

Podsumowując przeprowadzone badania narzędzie OWASP ZAP wydaje się być najlepszym wyborem, jednak należy pamiętać, że wybór narzędzia powinien być podyktowany potrzebami skanowanej aplikacji, jej wielkością i zachodzącymi procesami. Najlepszym rozwiązaniem zapewniającym bezpieczeństwo aplikacji jest jej systematyczne skanowanie za pomocą kilku narzędzi w odpowiedniej kolejności, a nie poleganie na wynikach tylko jednego z nich.

Literatura

[1] D.D. Bertoglio, A.F. Zorzo, Overview and open issues on penetration test, Journal of the Brazilian Computer Society 23(2) (2017) 1-2.

[2] Spis narzędzi służących do skanowania bezpieczeństwa polecanych przez OWASP, https://owasp.org/www-community/Vulnerability_Scanning_Tools, [02.2022].

[3] R. Devi, M. Kumar, Testing for Security Weakness of Web Applications using Ethical Hacking, 2020 4th International Conference on Trends in Electronics and Informatics 354 (2020) 358-360.

[4] D. Sagar, S. Kukreja, J. Brahma, S. Tyagi, P. Jain, Studying Open Source Vulnerability Scanners For Vulnerabilities In Web Applications, IIOABJ 9(2) (2018) 43-49.

[5] B. Mburano, W. Si, Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark, 2018 26th International Conference on Systems Engineering (2018) 1-2.

[6] Dokumentacja i kod źródłowy aplikacji bWAPP, https://sourceforge.net/projects/bwapp/files/bWAPP/, [03.2022].

[7] Dokumentacja i kod źródłowy aplikacji Mutillidae, https://github.com/webpwnized/mutillidae, [03.2022].

[8] M. El, E. McMahon, S. Samtani, M. Patton, H. Chen, Benchmarking vulnerability scanners: An experiment on SCADA devices and scientific instruments, IEEE International Conference on Intelligence and Security Informatics (ISI) (2017) 83-85.

[9] S. Tyagi, K. Kumar, Evaluation of Static Web Vulnerability Analysis Tools, 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (2018) 1-3.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 181–188
Received: 26 April 2022
Accepted: 19 June 2022

A comparative analysis of the functionality and quality of the interface of chosen applications for ordering food

Analiza porównawcza funkcjonalności i jakości interfejsu wybranych aplikacji do zamawiania jedzenia

Maciej Gieroba*, Marek Miłosz

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

Abstract

In recent years, growing popularity of food ordering mobile applications was noticeable. The COVID-19 pandemic has only accelerated this trend. In Poland, there are a large number of applications that allows to order meals from restaurants. In this article, three of the most popular ones have been selected - UberEats, Pyszne and Glovo. The aim of the research was to determine their functionality and the quality of their interfaces. The study used the method of experimenting with users who performed the most likely application usage scenarios, during which the time of their execution was measured, and then an survey using the SUS method was conducted. The obtained results allowed to determine that it is impossible to indicate the most effective interface in general. Depending on the task, the times of its execution differed. The results of the SUS survey showed that the UberEats app has the best interface quality.
Keywords: interface quality; user research; food ordering apps; SUS method

Streszczenie

W ostatnich latach można było zauważyć rosnącą popularność mobilnych aplikacji do zamawiania jedzenia. Pandemia wirusa COVID-19 tylko przyspieszyła ten trend. W Polsce istnieje duża liczba aplikacji pozwalających na zamawianie posiłków z restauracji. W niniejszym artykule wybrano trzy najpopularniejsze z nich – UberEats, Pyszne oraz Glovo. Celem badań było określenie ich funkcjonalności i jakości ich interfejsów. W badaniu zastosowano metodę eksperymentu z użytkownikami, którzy wykonywali najbardziej prawdopodobne scenariusze użycia aplikacji, w trakcie których mierzono czas ich wykonania a następnie przeprowadzono ankietę wykorzystującą metodę SUS. Uzyskane wyniki pozwoliły określić, że ogólnie nie można wskazać najbardziej efektywnego interfejsu. W zależności od zadania czasy jego wykonania różniły się. Wyniki ankiety metody SUS wykazały, że najlepszą jakość interfejsu posiada aplikacja UberEats.

Słowa kluczowe: jakość interfejsu; badania z udziałem użytkowników; aplikacje do zamawiania jedzenia; metoda SUS

*Corresponding author

Email address: maciej.gieroba@pollub.edu.pl

1. Introduction

The rapid development of the computer industry, especially mobile devices, makes many companies expand their activities by making their services available on the Internet [1]. This tendency also affected the catering industry [2]. The developing IT sector made the appearance of online stores allowing online shopping [3]. It is not surprising then those similar activities began to appear in other industries offering the purchase and delivery of goods, including the catering industry [4], of which telephone orders have long been a part. Unlike online stores, where most online stores have their own website or web application for shopping, the most popular online food ordering applications allow to choose to place orders at many restaurants. This is beneficial both for the restaurant, which does not have to have and maintain its own, independent application and provides advertising among users, and for the end customer who can use only one application offering access to many restaurants, and thus allows him a greater choice among prices and the type of food he want to order.

The scope of the research included examining the possibilities of application interfaces for food ordering in the form of a comparison, checking what a given interface enables and examining their efficiency by measuring the times of performed tasks as well as quality through the survey with the SUS (System Usability Scale) method among participants performing the tasks.

Before starting the research, the following hypotheses were made:

H1. Each of the interfaces of the tested applications allows to select the delivery time, type of receipt of the order, type of payment, and leave a tip for the courier.

H2. The quality of the food ordering app interfaces varies significantly.

H3. The quality of interfaces among users does not affect its performance.

In this article, the interfaces of three most popular applications for food ordering in Poland [5] [6]: UberEats, Pyszne and Glovo were compared. Testing the quality of the interface of a those applications was planned in two ways - through an experiment with users (i.e. measuring the time needed to perform specific tasks by

users) and through a survey according to SUS method [7], carried out among users after performing specific tasks. The research carried out in this way and the conclusions drawn on its basis allowed for the selection of the most effective and useful interface of the selected applications.

2. Literature overview

The role of the Internet in retail trade is becoming more and more important [1]. Information on the growing popularity of online shopping and importance of customer satisfaction was provided [1]. The work [1] examines existing problems, gathering information about customer satisfaction in selected online stores and provides them with suggestions on how to improve customer satisfaction and maintain their loyalty.

The article [2] reports that in the online food ordering market, many restaurants compete for orders placed by customers through online food ordering platforms. It has been found that the two main factors that guide restaurants are food quality and location. The results of the study [2] found that restaurant decisions regarding food quality are significantly influenced by customer behavior. The same study [2] showed that the location of the restaurant does not have a large impact on the customer due to the delivery that can be carried out by the food ordering platform.

The rapid growth of online customers and, consequently, of online stores was described in the work [3]. A study summarizing critical factors in the operation of online stores was conducted and the relationship between the popularity of the store and the marketing campaign was checked [3].

The rising popularity of mobile applications for food ordering was presented in the article [4]. This popularity is dependent on the customers age groups [4]. The study [4] analyzes also the perception of online food ordering services by customers. Article [5] provides information on the catering market in Poland in 2020 and data on online orders and how the Covid-19 pandemic affected this market. The presented research indicates the growing popularity of online food ordering in Poland and shows that the most popular mobile applications for food ordering are Pyszne, Glovo and UberEats. The same results are presented in the article [6].

The most commonly used standard questionnaire for assessing perceived usability of interfaces is SUS method [7]. Article [7] presents the history of SUS from its inception through recent research and future prospects, and says it is likely that SUS will continue to be a popular measure of perceived usability.

3. Chosen applications interfaces

Based on the analysis of the literature [5, 6], three most popular applications for food ordering in Poland were selected for the study: UberEats, Glovo and Pyszne.

3.1. UberEats

Figure 1 shows the UberEats mobile application interface. The colors of the interface are in light colors - the dominant color is white. The first view that the user encounters is the home page, which immediately contains a list of restaurants delivering to the user's current address, which is gained from the phone's shared location. This address can be changed by clicking on it and entering another address from the telephone keypad. At the top of the interface screen, the user can choose whether the order is to be delivered to the address indicated or picked up. When choosing a pickup, below the given or gained current address, instead of the list of restaurants, there is a map with marked restaurants, as shown in Figure 2. The list with restaurants and food categories is then transferred below the above-mentioned map. At the bottom of the interface, the user can switch between four main views - "Home", "Browse", "Orders" and "Account". The “Browse” view, contains product categories for dishes in the form of tiles. After clicking a given tile, the interface shows restaurants that fulfill orders for dishes from the selected category. Above these tiles there is a bar for searching for restaurants or specific dishes. Selecting the "Orders" view shows in the form of a list all orders placed by the user assigned to the account for which the user is logged in. The "Account" view allows to check the details and edit the user account and his functions.

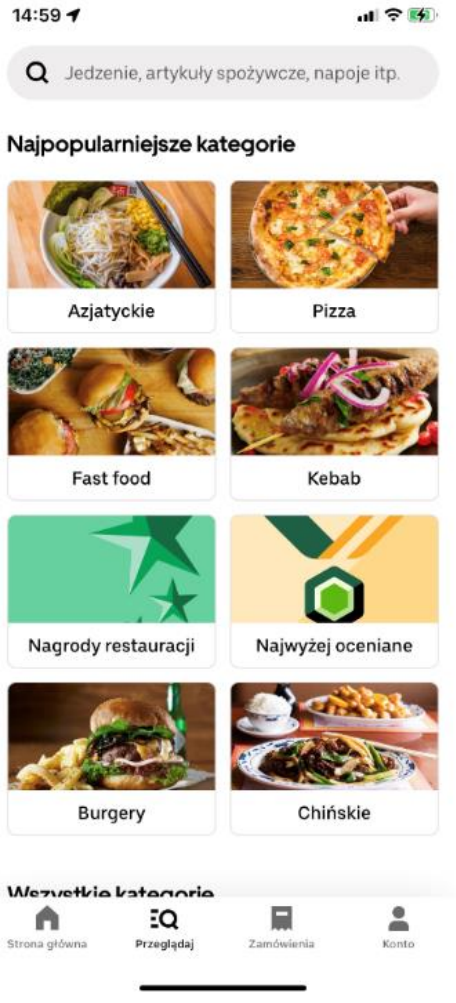


Figure 1: Home page of UberEats mobile application interface.

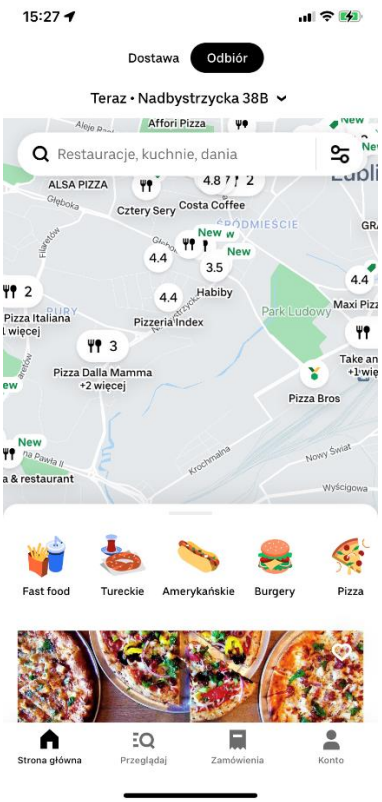


Figure 2: Map with restaurants in UberEats mobile application interface.

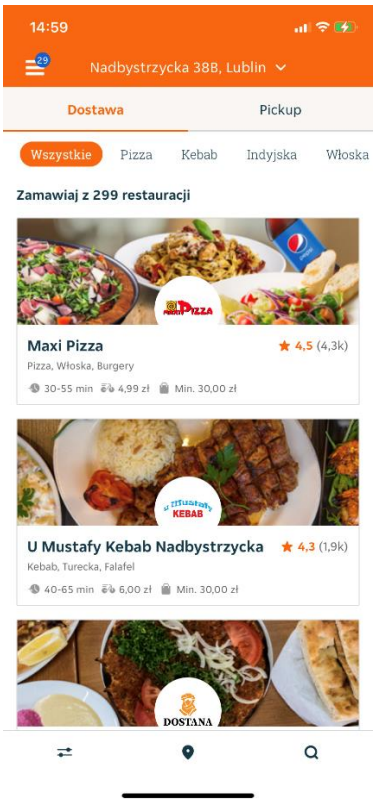


Figure 3: Main page of Pyszne mobile application interface.

3.2. Pyszne

Figure 3 shows the interface of the Pyszne mobile application. The interface colors are orange and white. The first view that appears to the user is the list of restaurants available for the current address which is at the top of the interface. After clicking on this address, the user can change it - by default, it is gained from the current mobile device location. Right under the address there are buttons for selecting delivery or picking up the order. At the bottom of the interface, there are three icons that allow user to: filter the displayed restaurants, show the map of the area with the marked restaurants available in the app and search for specific restaurants or dishes.

3.3. Glovo

Figure 4 shows the interface of the Glovo mobile app. The colors of the interface are dark. In the first view, after turning on the application, the user sees 8 round tiles that allow to choose one of the services offered by the Glovo service. At the top of the first view, the user can enter the delivery address - by default, it is taken from the current location. This article focuses on the interfaces for food ordering, and after selecting the "Food" tile, the user moves to the interface tested in this work. Figure 5 shows the interface for food ordering in the Glovo app. At the top of the interface is a search box for a restaurant or product. Below there are icons with captions showing product categories, after clicking which the user is presented with restaurants fulfilling orders that meet the requirements of the selected category. Below is a list of restaurants grouped into different collections, for example, as in Figure 5, the collections - "Best Nearby" and "Sales". In the upper right corner of the interface there is a switch that allows to choose the execution of the order - with delivery or pickup.

Figure 4: Main page of Glovo mobile application interface.

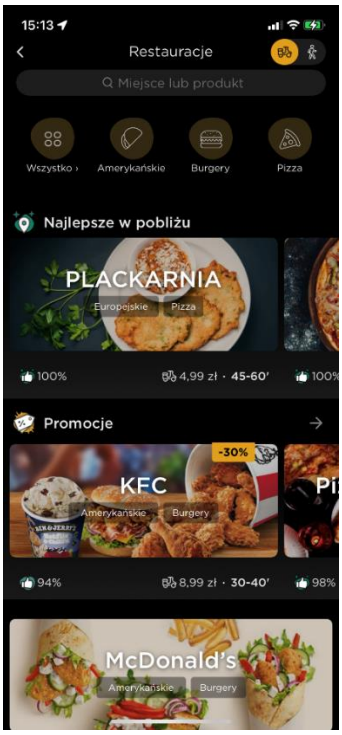


Figure 5: Restaurants list view of Glovo mobile application interface.

3.4. Joint overview

Online food ordering services via mobile applications, in addition to the obvious basic functionality, have additional possibilities, extending the main task of the service, enabling users to perform additional activities in relation to the details of the order, restaurant or encouraging the user to further orders through a system of loyalty programs. Testing the possibilities of the interfaces of the above-mentioned services were performed in mobile applications available on the iOS system, available in the AppStore. Table 1 presents the list of functionalities of selected mobile applications for food ordering.

The interface of each application allows user to choose the time of delivery, order pick-up type, online payment by card, payment on delivery, use of discount codes, tracking the supplier in real time, contact to the service staff after placing the order and tracking the status of the order. Each of the applications also allows for mobile payments in the BLIK system, however, in the Pyszne interface it is possible indirectly through the PayU internet payment operator. The Glovo application interface does not allow user to make payments with Apple Pay, unlike Pyszne and UberEats.

The tip for the courier is a phenomenon that can be realized directly upon received of the order, but each of the tested applications allows it to be sent via the Internet service. However, Glovo only allows this before placing the order - then the tip of customer choice is added to the cost of the order.

Other applications allow this after placing the order, allowing the user to decide depending on the time of delivery or whether the courier was pleasant.

Table 1: Summary of the interface capabilities of selected applications

Functionality	Pyszne	Glovo	UberEats
Delivery time selection	YES	YES	YES
Order pick-up type	YES	YES	YES
Card payment	YES	YES	YES
BLIK payment	YES	YES	YES
Apple Pay payment	YES	NO	YES
Cash on delivery	YES	YES	YES
Delivery without entering an address - "pin" on the map	NO	YES	PARTLY
Discount codes	YES	YES	YES
Gratuity to the courier	YES	PARTLY	YES
Loyalty program	YES	NO	PARTLY
Order personalization	YES	NO	YES
Message to the restaurant while ordering	PARTLY	PARTLY	PARTLY
Live provider tracking	YES	YES	YES
Contact the courier after placing the order	NO	YES	NO
Contact to the restaurant after placing the order	PARTLY	NO	YES
Contact for service after placing the order	YES	YES	YES
Order status tracking	YES	YES	YES

The loyalty program is not available in the Glovo interface. UberEats has such a program, however, only in selected restaurants and it cannot be combined with other eateries - the user can get a discount for a certain number of orders. Pyszne has the most developed loyalty program out of the three analyzed applications, also available in selected restaurants, however points obtained from orders from various restaurants sum up and can be exchanged for prizes.

Comparing the application capabilities, an examination was made of how the user can provide additional information regarding the order. Customization of the order is available at Pyszne and UberEats. There is no such functionality in the Glovo interface. It was also

checked whether it was possible to send a general message to the restaurant when placing the order. Pyszne and UberEats allow user to leave additional notes with each ordered meal and at the delivery address. Glovo allows to add information about user allergies. Neither of the interfaces allows general text information to be included in the order.

The contact after placing the order with the courier delivering the meal is only available in the Glovo app.

User can contact the restaurant executing the order directly from the UberEats interface. Pyszne allows to find phone number to restaurant, but not from the view of the order placed. Glovo does not have such capabilities.

4. Research methodology

4.1. Research environment

In order to start the research, it was necessary to prepare an appropriately configured research environment with locally installed applications that would allow the research to be carried out. The device used and the versions of the applications are presented in tables 2 and 3.

4.2. Study group

The study group on which the study will be performed will consist of 24 people aged 20-26 who use mobile devices on a daily basis. Each test person has never used the selected application. The study will not be extended to more than one application on each of the test persons, as the experiences gained from the previous application could distort the test result. As a result, according to the A/B method, there will be 8 people for each tested interface.

4.3. Research scenarios

A research group will be invited to perform the study and collect the results required to compare the performance and quality of interfaces of the selected applications.

Table 2: Research environment parameters

Name	Apple iPhone 11
Height	150.9mm
Width	75.7mm
Thickness	8.3mm
Weight	194g
Display	6.1-inch Multi-Touch LCD on the entire front surface of the device, made in IPS technology
Resolution	1792 x 828 pixels at 326 pixels per inch
Processor	A13 Bionic chip 6-core CPU with 2 performance cores and 4 energy-saving cores
Graphics processor	4-core GPU
RAM	4 GB
Storage	128 GB
Operating system	iOS 15.4.1

Table 3: Versions of the tested applications

App	Version
Pyszne	34.16.1
Glovo	7.38.0
UberEats	6.105.10003

People from the study group will carry out the study individually. In the study, a person from the study group will be asked to perform two tasks with the following content:

Task No. 1. “Please add a ”Hawaiian pizza“ or a replacement product from any restaurant to the cart."

Task No. 2. “Please find the price and show it to the person conducting the examination:

- A “Big Mac” product from a McDonald's restaurant
- The product “The Wołowino” from the restaurant “MOJO Kitchen & Friends”.”

The tasks were arranged so that each restaurant was available in each tested application. Task No. 1 was considered as completed if the cart included a pizza with the ingredient pineapple.

During the execution of the tasks, the time of task completion will be measured by the examiner who will constantly control whether the task has been performed correctly. If the test person reports that the task has been completed and the examiner finds that the task has not been completed, appropriate information will be provided to the test subject and person will be asked to continue the task - the measured time does not stop until the task is completed correctly. After performing the above-mentioned activities, the respondents will be asked to complete a questionnaire which will be performed in the survey used the SUS method.

5. SUS survey

SUS is a quick measurement of the usability of hardware, IT systems, websites and applications by means of a survey. SUS survey consists of 10 questions and a 5-point rating scale (based on the Likert scale) [7]:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

For questions 1, 3, 5, 7, 9, the following scores should be assigned [7]: strongly disagree = 0, rather

disagree = 1, have no opinion = 2, tend to agree = 3, and strongly agree = 4.

For questions 2, 4, 6, 8, 10 the score is as follows [7]: strongly disagree = 4, rather disagree = 3, have no opinion = 2, tend to agree = 1, and strongly agree = 0.

The points should be summed up and the obtained value multiplied by 2.5 [7].

6. Results

The research was conducted according to the methodology described in chapter 4. The results are presented in Tables 4-6 and Figures 6-8. Tables 4-6 show medians, averages and standard deviations of results. Box plots with a mustache were used to show the obtained results. The exceptions are the results showing the completion rate of the task by the study group presented in Table 7. On the basis of the collected results, the rho-Spearman correlation coefficient was also calculated between the times of the first and second tasks described in chapter 4.3

Table 4: Time of execution of the task No. 1

App	Median [s]	Average time [s]	Standard deviation [s]
Pyszne	23	24	8.95
Glovo	27	41.5	37.18
UberEats	74,5	74	29.61

Table 5: Time of execution of the task No. 2

App	Median [s]	Average time [s]	Standard deviation [s]
Pyszne	62	70	21.55
Glovo	62.5	64	20.40
UberEats	48.5	50	11.68

Table 6: The results of the SUS survey

App	Median [point]	Average [point]	Standard deviation [point]
Pyszne	72.5	67.5	16.35
Glovo	77.5	69.7	18.49
UberEats	86.25	78.125	17.26

The results of the SUS survey, which are presented in Figures 9, 10, 11, and Table 8.

Table 7: Task completion rate by the study group

App	Task No. 1 [%]	Task No. 2 [%]
Pyszne	100	89
Glovo	100	100
UberEats	100	100

Table 8: Rho-Spearman correlation coefficient between the times of the tasks and the results of the SUS survey and it's significance level

Task	rho-Spearman correlation	Significance level
Task No. 1	-0.014	0.951
Task No. 2	-0.317	0.14
Combined task No. 1 and task No. 2	-0.141	0.52

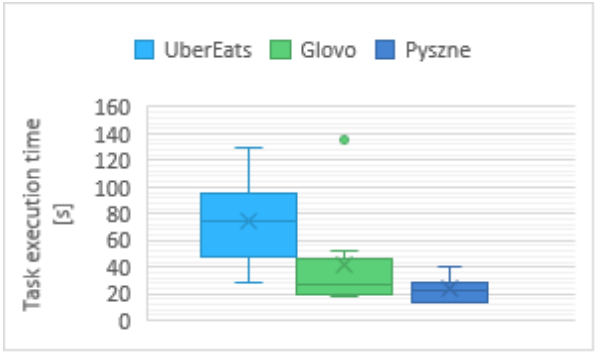


Figure 6: Summary of the research results for task No. 1. The mean values are shown with a X symbol, the horizontal line within the box represents the median, the top and bottom of the box is the 75th (Q3) and 25th (Q1) quartile respectively (interquartile range). The whiskers represent minimum and maximum values (excluding outliers), respectively, and the dots represent outliers.

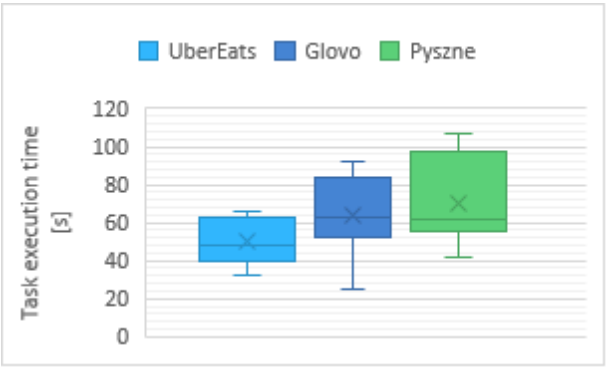


Figure 7: Summary of the research results for task No. 2. The mean values are shown with a X symbol, the horizontal line within the box represents the median, the top and bottom of the box is the 75th (Q3) and 25th (Q1) quartile respectively (interquartile range). The whiskers represent minimum and maximum values (excluding outliers), respectively, and the dots represent outliers.

7. Research discussion

In the research conducted with the use of three mobile applications for food ordering, it is not possible to clearly state which application interface is the most efficient. The selected tasks that the people from the study group were asked to do indicated different conclusions. Task No. 1 indicated that the most effective interface for finding one particular dish and adding it to the cart is in the Pyszne application with an average task completion of 24s.

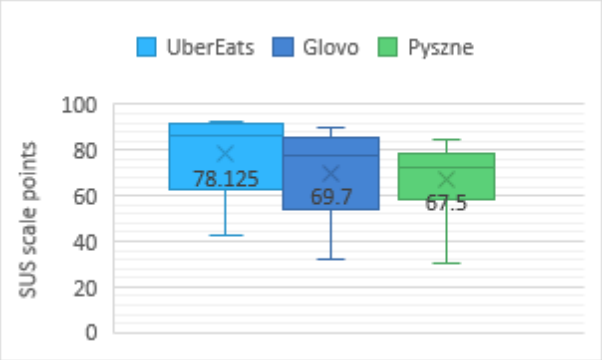


Figure 8: Summary of the results of the survey with the methodology of SUS (System Usability Scale). The mean values are shown with a X symbol, the horizontal line within the box represents the median, the top and bottom of the box is the 75th (Q3) and 25th (Q1) quartile respectively (interquartile range). The whiskers represent minimum and maximum values (excluding outliers), respectively, and the dots represent outliers.

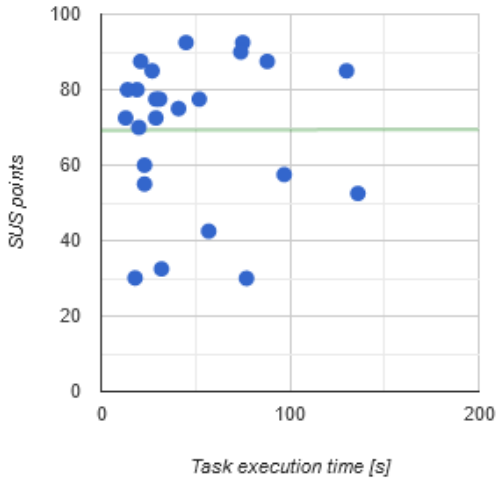


Figure 9: Rho-Spearman correlation coefficient between the execution times of task No. 1 and the results of the SUS survey.

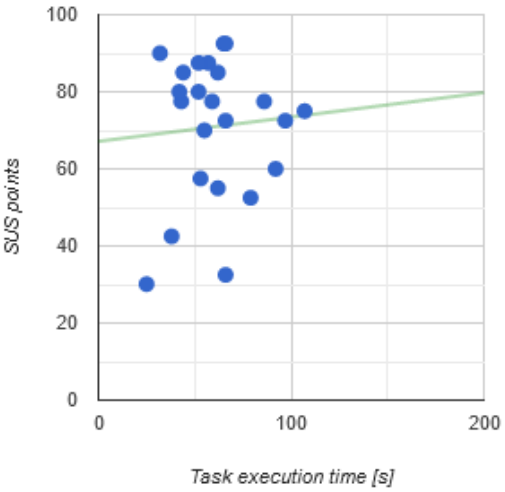


Figure 10: Rho-Spearman correlation coefficient between the execution times of task No. 2 and the results of the SUS survey.

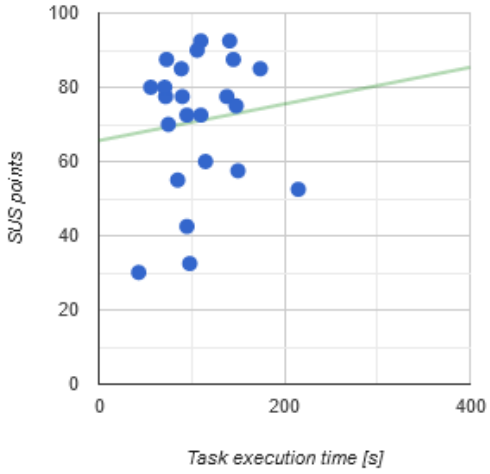


Figure 11: Rho-Spearman correlation coefficient between the combined execution times of task No. 1 and No. 2 and the results of the SUS survey.

The least effective interface in this task was the UberEats application interface with an average task completion time of 74 seconds, which is almost three times the value obtained for the Pyszne application interface. The times obtained in the Task No. 2 pointed to something completely different. In this task, the most effective interface turned out to be UberEats with the lowest average and median time needed to complete the task. Both Glovo and Pyszne had similar median times. Based on these studies, it can be concluded that the dependence of the effectiveness of the mobile application interface for food ordering is closely related to the type of action we want to achieve. Finding a specific meal from any restaurant is the most time-effective in Pyszne. However, if the user's goal is to find a specific dish from a specific restaurant, the most effective interface is the one in the UberEats application. During the research, the only repeated error in the implementation of tasks was searching for dishes or restaurants in the place where user should enter the address, in the interface of the Pyszne application, which is at the top of the interface. The place for searching for dishes or restaurants is located at the bottom of this interface and therefore it can be concluded that it is hardly visible.

The quality of the interfaces was tested using SUS surveys and it can be concluded from their results that the UberEats application interface is the interface of the highest quality for the user, while the interface of the Pyszne application has the lowest quality. The results of the SUS survey are similar for the Pyszne and Glovo interfaces, and the UberEats interface positively stands out from them. Based on the collected results, the hypothesis - "H2. The quality of the food ordering app interfaces varies significantly" can be rejected.

An important aspect of testing the quality of interfaces is also the degree of completion of the assigned tasks. Task No. 1 was completed by all people in each application. Tasks No. 2 could not be completed by one person working on the Pyszne UI, and the second task

completion rate for this app is approximately 89%. In the case the user did not complete the task in the Pyszne interface, he could not find a place to search for a dish and restaurant. Instead, he tried to enter values in the space intended for the delivery address.

Based on the analysis of the interface capabilities performed in chapter 3.4. it can be concluded that the posed hypothesis - "H1. Each of the interfaces of the tested applications allows to select the delivery time, type of receipt of the order, type of payment, and leave a tip for the courier " is confirmed.

In order to check the H3 hypothesis, correlation tests were performed using the rho-Pearson method. The obtained results, which are presented in Table 8 and in Figures 9, 10, and 11, show no correlation between the SUS survey results and the time of completing tasks for task No. 1 (correlation result equal to -0.014), but weak correlation for combined task times (correlation result equal to -0.317) and for task No. 2 (correlation result equal to -0.141). Based on these results, this hypothesis H3 is partially confirmed.

8. Conclusions

The adopted methodology, combining SUS and research with users, allowed to assess the quality of interfaces of the three most popular mobile applications for ordering food. As a result of research, it has been proven that the quality of food ordering application interfaces is not significantly different, as is their functionality.

The correlation hypothesis “H3 The quality of interfaces among users does not affect its performance" requires further research.

References

[1] Y. Qu, Research on online retail store customer satisfaction-A case study of HanDu, Journal of Chemical and Pharmaceutical Research 6(6) (2014) 2463-6.

[2] Z. He, G. Han, T. C. E. Cheng, B. Fan, J. Dong, Evolutionary food quality and location strategies for restaurants in competitive online-to-offline food ordering and delivery markets: An agent-based approach, International Journal of Production Economics 215 (2019) 61-72.

[3] C. Lin, T. Chien, H. Ma, The effects of popularity: An online store perspective, International Journal of Information Science and Management S(1) (2014) 1-11.

[4] A. M. Jacob, N. V. Sreedharan, K. Sreena, Consumer Perception of Online Food Delivery Apps in Kochi, International Journal of Innovative Technology and Exploring Engineering 8(7S2) (2019) 302-305.

[5] D. Czubatka, 34% of Poles order food with delivery more often (34 Proc. Polaków częściej zamawia jedzenie z dowozem), Rynek Gastronomiczny w Polsce – Raport (2020) 18-31.

[6] The most popular applications for ordering food in Poland (pol. Najpopularniejsze aplikacje do zamawiania jedzenia w Polsce), https://interaktywnie.com/biznes/newsy/biznes/zamawianie-jedzenia-przez-internet-oto-najpopularniejsze-dania-i-aplikacje-260931 , [20.04.2022].

[7] J. R. Lewis, The system usability scale: Past, present, and future, International Journal of Human-Computer Interaction 34(7) (2018) 577-90.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 189–194
Received: 3 May 2022
Accepted: 30 May 2022

Comparative analysis of the quality of recorded sound in the function of different recording formats

Analiza porównawcza jakości zarejestrowanego dźwięku w funkcji różnych formatów zapisu

Andrzej Król*, Tomasz Szymczyk

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

Abstract
In article, the quality of the following encoders was analyzed: mp3, AAC, wma and OGG Vorbis. An original graphic method was used to carry out the quantitative research. It consists in comparing the number of pixels (representing data) between the spectrogram of a wav file and the spectrograms of files compressed with different codecs and bit rates. It has been shown that the Ogg Vorbis encoder retains the most data from the uncompressed wav sample in all tested bit rates (128KBit/s, 160KBit/s, 320KBit/s).

Keywords: spectrogram; compression quality; comparative analysis

Streszczenie
W artykule przeanalizowano jakość koderów: mp3, AAC, wma i OGG Vorbis. Do przeprowadzenia badania ilościowego wykorzystano autorską metodę graficzną. Polega ona na porównaniu liczby pikseli (reprezentujących dane) pomiędzy spektrogramem pliku wav, a spektrogramami plików skompresowanych różnymi kodekami i przepływnościami. Wykazano, iż najwięcej danych z nieskompresowanej próbki wav zachowuje koder Ogg Vorbis we wszystkich badanych przepływnościach (128KBit/s, 160KBit/s, 320KBit/s).

Słowa kluczowe: spektrogram; jakość kompresji; analiza porównawcza

*Corresponding author
Email address: s96758@pollub.edu.pl (A. Król)

1. Wprowadzenie

Przez kilka ostatnich dziesięcioleci powstawały (i ciągle powstają) nowe rozwiązania w dziedzinie kompresji dźwięku. Przykładem może być koder AAC, który jest ciągle rozwijany i ulepszany głównie do celów przesyłania danych audio przez Internet lub do celów transmisji danych audio technologii cyfrowego radia naziemnego DAB/DAB+. Transmisja taka wymaga jak najlepszej jakości dźwięku i jednocześnie małej przepływności strumienia. Innym koderem ciągle rozwijanym jest kompresor Ogg Vorbis, który jest darmowy zarówno do celów komercyjnych jak i niekomercyjnych. W latach dziewięćdziesiątych ubiegłego stulecia ważne było, aby skompresowany plik był jak najmniejszy zachowując przy tym najwyższą jakość. Podyktowane to było małymi pojemnościami dysków w ówczesnych czasach. W późniejszych latach aż do teraz (2022 rok) dyski twarde są produkowane z coraz większą pojemnością sięgającą rzędu kilku terabajtów. W obecnych czasach kompresja dźwięku i obrazu jest bardziej potrzebna ze względu na przesył informacji (streamingowanie) przez Internet oraz ze względu, na rozwój jakości wideo oraz audio. Wraz ze wzrostem jakości (audio i wideo np. rozdzielczość FullHD, 4K) potrzebne jest więcej miejsca na dysku, więc zapotrzebowanie na coraz lepsze kodeki nie zmniejsza się.

Tematem artykułu jest analiza strumieni audio pod względem jakości. Do przeprowadzenia badania wykorzystano autorską metodę graficzną. Polega ona na porównaniu liczby pikseli (reprezentujących dane) pomiędzy spektrogramem pliku wav, a spektrogramami plików skompresowanych różnymi kodekami i przepływnościami.

2. Badania literaturowe

Badania jakości dźwięku, zarówno ilościowe jak i jakościowe były przeprowadzane przez różnych badaczy.

Grzegorz Brzuchalski w swojej rozprawie doktorskiej (Politechnika Warszawska 2015) opisał miary oceny jakości kodeków. Autor stwierdził, że „za najbardziej wiarygodny sposób oceniania jakości dźwięku uważa się przeprowadzanie testów odsłuchowych na odpowiednio dobranej grupie osób. Testy są normalizowane, a wyniki uśredniane” [1]. Testy takie polegają na odsłuchiwaniu skompresowanych próbek różnymi metodami oraz ocenie subiektywnej jakości przez osoby biorące udział w badaniu. Autor rozprawy wspomniał też o problemie odpowiedniego nagłośnienia i pomieszczenia do przeprowadzenia testów odsłuchowych. Inną grupą miar, które autor rozprawy opisał jest „stosunek szumu do progów słyszenia NMR (ang. Noise to Mask Ratio) jest jedną z podstawowych i najważniejszych miar w ocenie jakości dźwięku kompresowanego” [1]. Grzegorz Brzuchalski wspomniał też o miarach:

- mierze struktury harmonicznej błędu,
- miarach modulacji,
- miarach głośności zniekształceń,

- miarach prawdopodobieństwa wykrycia zniekształceń.

Innymi badaczami jakości dźwięku byli chińscy naukowcy (Luo, D., Luo, W., Yang, R., & Huang, J.), którzy do badania zastosowali „21 wymiarowy wektor cech do pomiaru artefaktów kwantyzacji w różnych współczynnikach kompresji" [2]. Porównali w ten sposób kodeki mp3, wma w różnych przepływnościach z kodekiem wav. Metoda ta polega na porównaniu pięciosekundowych skompresowanych próbek koderem mp3 i wma z analogiczną pięciosekundową próbką w formacie nieskompresowanym wav. Autorzy opisali to porównanie za pomocą liczb wyrażających procentową zgodność skompresowanej próbki z analogiczną próbką w formacie wav. Okazało się, że w badaniu tym próbki skompresowane kodekiem mp3 były bardziej wierne oryginałowi zapisanemu w formacie wav niż analogiczne próbki w formacie wma.

Testy odsłuchowe były wykonywane przez różnych badaczy. Jednym z nich był Gabriel Bouvigne, który przeprowadził test odsłuchowy koderów: mp3 i wma w różnych przepływnościach. Stwierdził on, że „przy bardzo niskich przepływnościach wydaje się oczywiste, że WMA 9 jest lepszy niż Lame 3.93. Jednak wraz ze wzrostem przepływności różnica się zmniejsza, a przy 128 kb/s są one dość podobne" [3].

„HE-AAC v2 – połączenie AAC, SBR i PS – to bez wątpienia najpotężniejszy kodek audio dostępny dzisiaj" [4]. W tym artykule został opisany test odsłuchowy dźwięku wielokanałowego. Najlepszy okazał się kodek AAC+.

Włoscy naukowcy (Bucci, G., Franciosi, F., & Valocchi, P.) opisali badanie jakości za pomocą modelu percepcyjnego jakości kodeka audio. „Ideą proponowanej techniki jest ważenie kodowanych zniekształceń poprzez filtrowanie ich za pomocą funkcji transferu 1/H(z), przed zastosowaniem standardowych pomiarów zniekształceń" [5].

Amerykańscy naukowcy (Kandadai, S., Hardin, J., & Creusere, C. D.) w artykule opisali wykorzystanie MSSIM w kontekście oceny jakości dźwięku. Zaprezentowali dwa różne sposoby zastosowania MSSIM do danych audio. Autorzy użyli siedmiu różnych sekwencji próbek monofonicznych w częstotliwości 44,1KHz. Do tych różnych sekwencji sampli wygenerowali różne rodzaje szumów [6].

H. i M. Portalscy przeprowadzili badanie jakości dźwięku. „Przedstawili analizy porównawcze dźwięku skompresowanego w formacie mp3 z oryginalnym. Podali oceny subiektywne różnic określone w oparciu o własne badania oraz metodę obiektywną bazującą na porównaniu poziomów sygnałów w podpasmach tercjowych spektrogramu" [7]. Porównali jakość formatu wav i mp3 metodą zarówno subiektywną jak i obiektywną. „Zdaniem autorów – skompresowane pliki mp3 dobrze nadają się do odtwarzania muzyki rozrywkowej, natomiast nie są zalecane dla muzyki klasycznej" [7].

Niemiecki naukowiec Karlheinz Branderburg porównał w swoim artykule koder mp3 z aac. Stwierdził, że „używając kodera o dobrej wydajności, MPEG-1 Layer 3 i MPEG-2 można kompresować muzykę przy zachowaniu jakości zbliżonej do CD lub CD" [8].

Naukowcy z uniwersytetu z Malezji (Gunawan, T. S., Aisyah, S., Rashiud, A., Katiwi, M.) zbadali wielokanałowe kodery AAC, AC3 i OGG. Badanie polegało na zmierzeniu czasu kompresji plików źródłowych różnymi koderami. „Z obserwacji porównania stratnych koderów stwierdzili, że najlepszym koderem dla czasu kodowania jest AC3. Ten koder zapewnia dobry czas kodowania przy wielokanałowym dźwięku 5.1 i 7.1." [9]. Natomiast koder Ogg Vorbis uzyskał najlepszy współczynnik kompresji.

Test odsłuchowy został przeprowadzony przez E.B Meyer i D. R Morgan w celu znalezienia różnicy pomiędzy jakością dźwięku z płyt CD, a jakością dźwięku z płyt SA-CD oraz płyt DVD. Test ten składał się z 554 prób, w których osoby biorące udział w badaniu odgadywały z jakiego źródła pochodził słyszany dźwięk. Poprawnych odpowiedzi było 276. Stwierdzono, że „praktycznie wszystkie płyty SACD i nagrania DVD - Audio brzmiały lepiej niż większość płyt CD - czasami znacznie lepiej" [10].

3. Opis badania

3.1. Stanowisko badawcze

Do przeprowadzenia badań wykorzystano komputer, którego specyfikację przedstawiono w tabeli (Tabela 1).

Tabela 1: Specyfikacja komputera wykorzystanego do badań

Komponent	Dokładne parametry
Płyta główna	ASRock Z97 Pro4
Karta dźwiękowa	Zintegrowana na płycie głównej
Procesor	Intel I5-4690K 3,5 GHz
Pamięć RAM	2x8GB DDR 3
Karta Graficzna	Gigabyte GeForce GTX 960 4 GB
Napęd optyczny	Nagrywarka Blu-Ray LG BH16NS40
Dysk twardy 1	SSD Samsung 970 EVO Plus 500 GB
Dysk twardy 2	SSD OCZ CT500BX100 500GB
System operacyjny	Microsoft Windows 10 Pro 64bit
Zasilacz	Be quiet Pure power 10 600W
Mysz, klawiatura	Logitech MK235
Monitor	LG M2380D
Słuchawki	Superlux HD681 EVO

Programy wykorzystane do badań przedstawiono w tabeli (Tabela 2).

Tabela 2: Programy zastosowane do badań

Program	Zastosowanie w badaniu
Sound Forge Audio Studio 12.6	Zgrywanie utworów z płyt CD do formatu wav, przygotowanie próbek – obcięcie wybranych

	fragmentów utworów.

Tabela 2 (c.d.): Programy zastosowane do badań

DiffImg v2.1.0	Wykonanie różnicy obrazów (zawierających spektrogramy próbek).
ImageJ 1.53e	Wykonanie różnicy obrazów (zawierających spektrogramy próbek).
Pazera Free Audio Extractor v2.9 64 bit	Kompresja próbek (wav) do formatów: mp3, ogg, aac, wma w różnych przepływnościach.
FFmpeg+sox	Generowanie obrazów spektrogramów z próbek.

3.2. Przygotowanie próbek i spektrogramów

Do badania wybrano 24 próbki. Próbki zostały przygotowane z fragmentów utworów z płyt CD (po 3 próbki z każdego z siedmiu utworów) oraz z próbek instrumentów pobranych ze specjalnej strony internetowej [11] zawierającej darmowe próbki dźwięków wysokiej jakości.

Próbki fragmentów utworów przygotowano za pomocą programu Sound Forge Audio Studio w wersji 12.6. Całe utwory najpierw zgrano z płyty CD na dysk, a następnie wybrano i wycięto fragmenty piosenek w celu przygotowania próbek źródłowych (nieskompresowanych). Utwory zgrano z oryginalnych płyt, które są w posiadaniu autora pracy. W celu zachowania praw autorskich, na potrzeby artykułu wykorzystane zostały 15 sekundowe próbki zamiast całych utworów. Autorzy uważają, że 15 sekundowe próbki są wystarczająco długie aby dać jednoznaczne wyniki, a jednocześnie taka długość próbek nie obciąża zbytnio stanowiska badawczego.

Tabela 3: Lista próbek instrumentów

Numer próbki	Nazwa pliku wav	Czas (s)	Opis
1.	Ensoniq-VFX-SD-Ride-Cymbal.wav	1,338	Cymbałki – wykonanie na sentyzatorze Ensoniq VFX-SD.
2.	Alesis-Fusion-Acoustic-Bass-C2.wav	2,897	Kontrabas dźwięk c2 – wykonane na sentyzatorze Alesis Fusion.
3.	Alesis-Fusion-Nylon-String-Guitar-C4.wav	2,208	Gitara akustyczna dźwięk c4 c2 – wykonane na sentyzatorze Alesis Fusion.

Do badań wykorzystano także próbki trzech instrumentów, które opisano w tabeli (Tabela 3).

Lista utworów wykorzystanych do badań:

1. Whitney Houston – I will always love you
2. Sandra- Midnight man
3. Belinda Carlisle – Shades of Michaelangelo
4. Marek Grechuta – Dni, których jeszcze nie znamy
5. Queen – Don't stop me now
6. Offspring – Have you ever
7. A-Ha – Take on me

Parametry wspólne dla wszystkich próbek źródłowych to:

1. Częstotliwość: 44.1 KHz.
2. Rozdzielczość: 16bit.
3. Liczba kanałów: 2 (stereo).
4. Format: PCM wav nieskompresowany.
5. Bitrate: 1411 Kbps.

Próbki fragmentów utworów przygotowano za pomocą programu Sound Forge Audio Studio w wersji 12.6. Całe utwory najpierw zgrano z płyty CD na dysk, a następnie wybrano i wycięto fragmenty piosenek w celu przygotowania próbek źródłowych (nieskompresowanych).

Tabela 4: Kodeki wraz przepływnościami

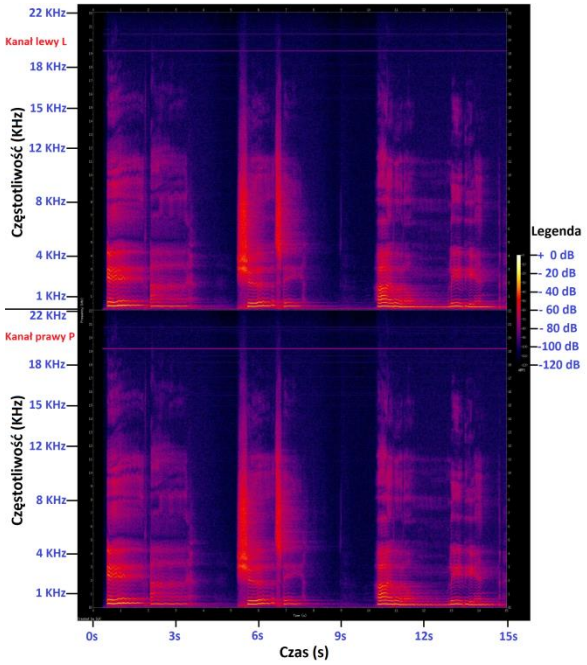
Numer próbki	Kodek	Przepływność (KBit/s)	Rodzaj przepływności
1.	Mp3	320	Stała
2.	Mp3	160	Stała
3.	Mp3	128	Stała
4.	Mp3	170-210	Zmienna
5.	Mp3	140-185	Zmienna
6.	Mp3	100-130	Zmienna
7.	Ogg Vorbis	320	Stała
8.	Ogg Vorbis	160	Stała
9.	Ogg Vorbis	128	Stała
10.	M4a (Aac)	320	Stała
11.	M4a (Aac)	160	Stała
12.	M4a (Aac)	128	Stała
13.	Wma	320	Stała
14.	Wma	160	Stała
15.	Wma	128	Stała

Następnym krokiem, który podjęto była kompresja próbek źródłowych do formatów: mp3, ogg vorbis, aac, wma. Kompresję przeprowadzono z różnymi wielkościami parametru bitrate (przepływność). Listę kodeków wraz z przepływnościami pokazano w tabeli (Tabela 4). Wykorzystano program Pazera Free Audio Extractor 64bit w wersji 2.9. Próbki skompresowane przygotowano z próbek źródłowych (wav).

Po skompresowaniu próbek źródłowych stworzono obrazy w formacie png zawierające spektrogramy próbek źródłowych (WAV) i skompresowanych (mp3, ogg vorbis, wma, aac). Spektrogramy przygotowano w identycznej skali za pomocą programu FFmpeg wraz programem Sox.

W celu przeprowadzenia badania przygotowano pliki png zawierające różnicę obrazów spektrogramów pomiędzy próbką źródłową nieskompresowaną wav, a próbkami skompresowanymi różnymi kodekami z różnymi przepływnościami.

Przykładowy spektrogram pliku WAV PCM nieskompresowanego zaprezentowano na rysunku (Rysunek1). Na osi x przedstawiono czas w sekundach, natomiast na osi y przedstawiono częstotliwości (KHz) w obu kanałach. Rysunek jest podzielony poziomo na dwie części. Górna część rysunku odpowiada za lewy kanał. Natomiast dolna część rysunku odpowiada za prawy kanał. Kolory na rysunku oznaczają moc sygnału na poszczególnych częstotliwościach w danym czasie. Amplitudę mocy sygnału przestawiono z prawej strony rysunku na legendzie.



Rysunek1: Przykładowy spektrogram z pliku Whitney Houston 1.wav.

3.3. Zasada działania autorskiej metody

Liczba pikseli spektrogramu jest liczbą pikseli, którymi narysowano spektrogram. W przypadku spektrogramu pliku wav tą liczbę oznaczono jako 100 % ze względu na bezstratny sposób kompresji pliku dźwiękowego wav. W przypadku spektrogramów plików skompresowanych stratnie, ta liczba będzie mniejsza. Liczba pikseli spektrogramu pliku skompresowanego, będąca najbardziej zbliżona do liczby pikseli spektrogramu pliku wav, będzie oznaczała najlepszą jakość kodeka (najmniejszy ubytek danych względem oryginału). Do obliczeń wykorzystano kody kolorów w formacie RGB. W celu znalezienia liczby pikseli spektrogramu należy wykonać następujące kroki:

1. Obliczyć liczbę wszystkich pikseli obrazu png (przemnożyć liczbę pikseli w poziomie przez liczbę pikseli w pionie).
2. Od otrzymanej liczby wszystkich pikseli należy odjąć:
   a) Liczbę czarnych pikseli tj. o wartości: (0,0,0).
   b) Liczbę szarych pikseli tj. o wartości: (127,127,127) oraz (191,191,191), które wykorzystano do narysowania osi oraz ich jednostek.
   c) Liczbę białych pikseli tj. o wartości: (255,255,255), które wykorzystano do opisu nazw osi, nazwy legendy oraz napisu „Created by SoX”. Liczba ta zawsze wynosi 561 pikseli.
   d) Liczbę pikseli, które przeznaczono na legendę (zawsze 5600 pikseli).
3. Otrzymana liczba jest „liczbą pikseli spektrogramu”.

Kolory są reprezentacją jednoznaczną harmonicznych dźwięku. Obrazy sprktrogramów stworzono z najwyższą możliwą rozdzielczością. Spektrogramy próbek wav i spektrogramy odpowiadających im plików audio skompresowanych są w tej samej rozdzielczości, co umożliwia przeprowadzenie badań.

Tabela 5: Analiza liczb pikseli spektrogramów próbek "Queen – Don’t stop me now" zapisanych w różnych formatach

Lp	Spektrogramu wraz z rodzajem kompresji (plik png)	Liczba pikseli spektrogramu	Procent zachowanych pikseli spektrogramu względem oryginału wav (%)
1.	wav	3219648	100
2.	m4a_128	2251199	69,92
3.	m4a_160	2348385	72,94
4.	m4a_320	2504022	77,77
5.	mp3_128	2359469	73,28
6.	mp3_160	2397147	74,45
7.	mp3_320	2790219	86,66
8.	mp3_V_100-130	2235228	69,42
9.	mp3_V_140-185	2358129	73,24
10.	mp3_V_170-210	2394366	74,37
11.	ogg_128	2510499	77,97
12.	ogg_160	2614504	81,20
13.	ogg_320	2879735	89,44
14.	wma_128	1907267	59,24
15.	wma_160	2033098	63,15
16.	wma_320	2333860	72,49

Tabela 5 zawiera obliczone liczby pikseli spektrogramów wraz z procentami zachowanych pikseli spektrogramów względem źródłowego i nieskompresowanego pliku wav dla jednej próbki. Im większa liczba (w %) , tym więcej zachowanych danych. Można wnioskować, że największa liczba (w %) oznacza najlepszą jakość skompresowanej próbki. Takie obliczenia wykonano dla wszystkich 24 próbek w formacie nieskompresowanym wav.

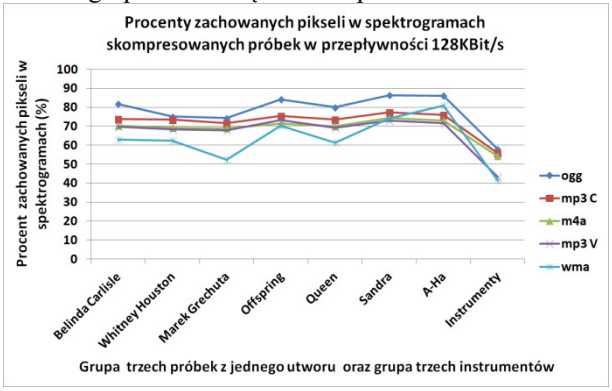
Następnym krokiem było policzenie średnich procentowych wartości (zachowanych pikseli względem oryginału wav) dla poszczególnych utworów, uwzględniając kodery i przepływności.

4. Wyniki badań

Otrzymane wyniki badań pogrupowano względem przepływności na trzy grupy:

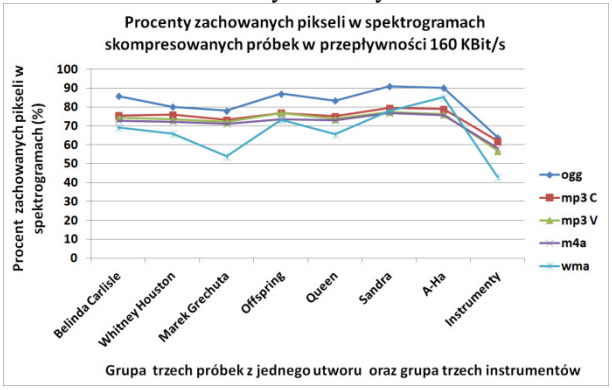
- pierwsza grupa - 128KBit/s,
- druga grupa - 160Kbit/s,
- trzecia grupa - 320KBit/s.

Na rysunku (Rysunek 2) przedstawiono wykres średnich procentów zachowanych pikseli ze spektrogramów grup próbek skompresowanych w przepływności 128 KBit/s względem ich odpowiedników w formacie wav nieskompresowanym. Pierwsze siedem grup składa się z trzech próbek wybranych z jednego utworu. Natomiast ostatnia grupa składa się z trzech próbek instrumentów.



Rysunek 2: Wykres średnich zachowanych pikseli w spektrogramach próbek skompresowanych (w przepływności 128 KBit/s) względem analogicznych spektrogramów w formacie wav.

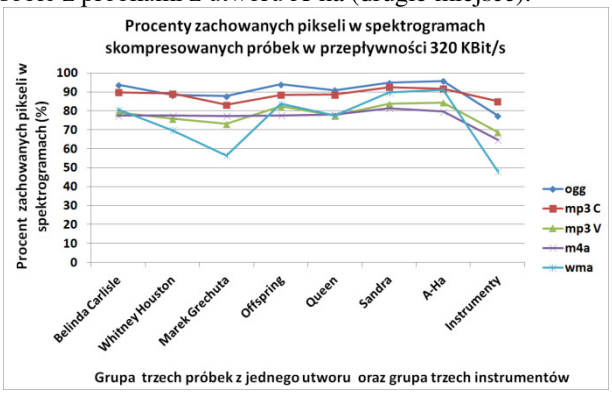
We wszystkich grupach najlepszym koderem okazał sie Ogg Vorbis. Natomiast najgorszym i najbardziej zmiennym okazał się koder wma, pomimo iż lepiej radził sobie z próbkami z utworu A-ha (drugie miejsce). Utwór ten ma charakter dynamiczny.



Rysunek 3: Wykres średnich zachowanych pikseli w spektrogramach próbek skompresowanych (w przepływności 160 KBit/s) względem analogicznych spektrogramów w formacie wav.

Na rysunku powyżej (Rysunek 3) przedstawiono wykres średnich procentów zachowanych pikseli ze spektrogramów grup próbek skompresowanych w przepływności 160 KBit/s względem ich odpowiedników w formacie wav nieskompresowanym. Pierwsze siedem grup składa się z trzech próbek wybranych z jednego utworu. Natomiast ostatnia grupa składa się z trzech próbek instrumentów. We wszystkich grupach najlepszym koderem okazał sie Ogg Vorbis. Natomiast najgorszym i najbardziej zmiennym okazał się koder wma, pomimo iż lepiej radził sobie z próbkami z utworu A-ha (drugie miejsce).

Na kolejnym rysunku (Rysunek 4) przedstawiono wykres średnich procentów zachowanych pikseli ze spektrogramów grup próbek skompresowanych w przepływności 320 KBit/s względem ich odpowiedników w formacie wav nieskompresowanym. Pierwsze siedem grup składa się z trzech próbek wybranych z jednego utworu. Natomiast ostatnia grupa składa się z trzech próbek instrumentów. W prawie wszystkich grupach najlepszym koderem okazał sie Ogg Vorbis. W grupie instrumentów najlepiej wypadł koder mp3 ze stałą przepływnością. Natomiast najgorszym i najbardziej zmiennym okazał się koder wma, pomimo iż lepiej radził sobie z próbkami z utworu A-ha (drugie miejsce).



Rysunek 4: Wykres średnich zachowanych pikseli w spektrogramach próbek skompresowanych (w przepływności 320 KBit/s) względem analogicznych spektrogramów w formacie wav.

Reasumując autorska metoda zastosowana w niniejszej pracy jednoznacznie wykazała, że najwięcej danych zostało zachowanych (w trzech badanych przepływnościach) po skompresowaniu próbek kodekiem Ogg Vorbis. Natomiast najgorzej poradził sobie koder wma, gdyż prawie wszystkie próbki skompresowane próbki tym koderem traciły najwięcej danych. Wszystkie badane kodery najgorzej radziły sobie z próbkami instrumentów, gdyż w tych skompresowanych samplach utracono najwięcej danych względem im odpowiadającej próbki nieskompresowanej w formacie wav.

5. Podsumowanie

W niniejszym artykule została zaprezentowana autorska metoda graficzna. Metoda ta wykazała, że najwięcej danych (względem pliku wav nieskompresowanego) zostało zachowanych po skompresowaniu próbek koderem Ogg Vorbis (w prawie wszystkich przypadkach). Jest to metoda mająca na celu uniezależnienie wyników od jakości odsłuchowych słuchawek lub głośników oraz od możliwości słuchowych badacza. Metoda ta wskazuje jaki procent danych zostało zachowanych w skompresowanym pliku dźwiękowym względem analogicznego oryginału nieskompresowanego. Można stwierdzić, że próbki zachowujące najwięcej danych z oryginału są najwierniejsze oryginałowi i przez to są najlepszej jakości.

Można też przypuszczać, iż wykorzystana autorska metoda graficzna nie jest idealna, gdyż nie ocenia w jakich pasmach częstotliwości utracono najwięcej danych, a w jakich najmniej. Metoda nie ocenia także, które dane są słyszalne (istotne), a które nie np. jeden koder może usuwać więcej nieistotnych danych, a drugi mniej. Każdy człowiek inaczej słyszy dźwięk, więc nie da się jednoznacznie stwierdzić, czy słyszalna jakość będzie najwyższa, gdyż metoda ta nie bierze pod uwagę

możliwości słuchowych badacza. Można jedynie przypuszczać, że skoro koder OGG Vorbis zachowuje najwięcej danych z oryginału nieskompresowanego, to znaczy, że pliki skompresowane tym koderem mają najlepszą jakość względem oryginału.

Literatura

[1] G. Brzuchalski, Optymalizacja algorytmów kwantyzacji w kodowaniu dźwięku, rozprawa doktorska, Politechnika Warszawska, 2015.

[2] D. Luo, W. Luo, R. Yang, J. Huang, Compression history identification for digital audio signal, 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, 3 (2012) 1733-1736, https://doi.org/10.1109/ICASSP.2012.6288233.

[3] G. Bouwigne, WMA9 versus MP3, http://www.mp3-tech.org/tests/wma9/index.html, [15.03.2022].

[4] S. Meltzer, G. Moser, MPEG-4 HE-AAC v2 – audio coding for today's digital media world, 1 (2006) 37-48.

[5] G. Bucci, F. Franciosi, P. Valocchi, The measurement of audio-codec sound quality, IEEE Instrumentation and Measurement Technology Conference and IMEKO Tec, 1 (1996) 622–627, https://doi.org/10.1109/IMTC.1996.507457.

[6] S. Kandadai, J. Hardin, C. D. Creusere, Audio quality assessment using the mean structural similarity measure, 2008 IEEE International Conference on Acoustics, Speech and Signal Processing (2008) 221–224, https://doi.org/10.1109/ICASSP.2008.4517586.

[7] H. Portalska, M. Portalski, Dźwięk w przekazie marketingowym – oryginał, czy mp3?, Marketing i rynek, 11 (2014) 555–564.

[8] K. Branderburg, MP3 and AAC explained, AES 17th International Conference on High Quality Audio Coding, Erlangen 1999.

[9] T. S. Gunawan, S. A. A. Rashid, M. Katiwi, Investigation of Various Algorithms on Multichannel Audio Compression, IEEE 4th International Conference on Smart Instrumentation, 11 (2017) 1–5, https://doi.org/10.1109/ICSIMA.2017.8311985.

[10] E. B. Meyer, D. R. Moran, Audibility of a CD-Standard A/D/A Loop Inserted into High-Resolution audio playback, Journal of the audio engineering society, 55 (2007) 775–779.

[11] Strona z próbkami instrumentów w formacie wav, https://freewavesamples.com/, [15.01.2022].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 195–202
Received: 4 May 2022
Accepted: 30 May 2022

Comparison of the most popular operating systems in terms of functionalities

Porównanie najpopularniejszych systemów operacyjnych pod względem funkcjonalności

Marek Mucha*, Jacek Lato*, Tomasz Szymczyk

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

Abstract
The main purpose of research is comparison of the following modern operating systems: Windows 10, Windows 11, MacOS Catalina and Linux Ubuntu 20.04 LTS. An analysis was made in terms of functionalities and time needed to perform basic activities. The systems were selected on the basis on performed popularity analysis, by using StatCounter statistic. To study each operating system it was necessary to create two test stands corresponding to the requirements of the systems. Conducted research were divided on two sections. In the first one, analysis of the possessed functionalities, assessment of the advancement and ease of using them was performed. In the second section, examination was carried out to compare the operating system in terms of the time of performing specific activities.

Keywords: operating systems; functionalities; comparative analysis

Streszczenie
Głównym celem artykułu jest porównanie następujących współczesnych systemów operacyjnych: Windows 10, Windows 11, MacOS Catalina oraz Linux Ubuntu 20.04 LTS. Dokonano analizy pod względem funkcjonalności i czasu potrzebnego na wykonanie podstawowych czynności. Systemy wybrano na podstawie przeprowadzonej analizy popularności, przy użyciu statystyk StatCounter. Do zbadania wszystkich systemów konieczne było przygotowanie dwóch stanowisk badawczych odpowiadających wymaganiom systemów. Przeprowadzone badania zostały podzielone na dwie części. W pierwszej, nastąpiła analiza posiadanych funkcjonalności oraz ocena zaawansowania i łatwości użycia. W drugiej części, zostało przeprowadzone badanie mające na celu porównanie systemów pod względem czasu wykonywania określonych czynności.

Słowa kluczowe: systemy operacyjne; funkcjonalności; analiza porównawcza

*Corresponding author
Email address: marek.mucha1@pollub.edu.pl (M. Mucha), jacek.lato@pollub.edu.pl (J. Lato)

1. Wstęp

Wraz z rozwojem technologicznym w wieku XX zaczęły powstawać pierwsze urządzenia powszechnie znane, jako komputer. Działania podczas II wojny światowej zmotywowały naukowców do poszukiwania szybszego sposobu wykonywania obliczeń [1, 2]. W ten sposób powstał komputer Atanasoff-Berry Computer. Do roku 1973 za jeden z pierwszych komputerów uznawano ENIAC [3], który został stworzony przez J.P. Eckerta i J.W. Mauchly'ego w 1946 roku. Wywarło to zauważalny wpływ na kolejne urządzenia takie jak np. UNIVAC, EDVAC [4]. Duża liczba rezystorów, kondensatorów, przełączników, przekaźników i innych elementów składowych skutkowała wielkimi gabarytami urządzenia oraz rekordowym na tamten czas taktowaniem 0.1 MHz. Do poprawnego działania niezbędna była czasochłonna obsługa operatorów, którzy musieli sami programować i przełączać styki, komutatory a jakiekolwiek błędy powodowały spore opóźnienia. Opisywane urządzenie służyło głównie do całkowania równań balistycznych i obliczania trajektorii pocisków morskich [5]. W późniejszych latach komunikacja na linii komputer - użytkownik została uproszczona przy pomocy systemów operacyjnych.

Celem niniejszego artykułu jest przeanalizowanie kilku wybranych współczesnych systemów operacyjnych dla stacji roboczych pod względem oferowanych funkcjonalności. W tym celu zostały wybrane dwa systemy z rodziny uniksowych: macOS Catalina oraz Linux - Ubuntu a także dwa systemy z rodziny Windows NT: Windows 10 i Windows 11. Przy wyborze analizowanych systemów przeprowadzono analizę rynku na podstawie danych zebranych z StatCounter [6]. Porównanie zostało wykonane dla systemów klasy desktop oraz z perspektywy zwykłego użytkownika, którego główne doświadczenia związane są ze środowiskiem Windows.

W artykule opisano dwie części badań: praktyczną oraz teoretyczną zawierająca analizę. Część praktyczna obejmuje instalacje systemów, przygotowanie wszystkich niezbędnych plików, poleceń, sprawdzenie wszystkich parametrów oraz posiadanych funkcji a w części teoretycznej są podsumowane wszystkie otrzymane wyniki na podstawie przeprowadzonych badań z części praktycznej.

W związku z przeprowadzaniem analiz i badań zostały postawione dwie hipotezy:

H1: System macOS potrzebuje najmniej czasu na wykonanie podstawowych czynności.
H2: System Windows 11 posiada największą liczbę funkcjonalności.

2. Materiały i metody

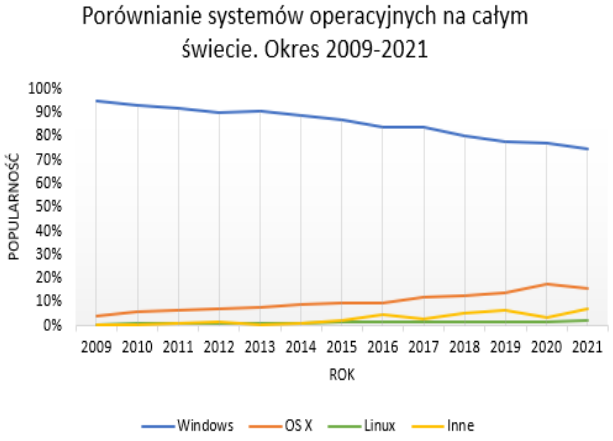
2.1. Analiza rynku pod względem popularności

Użytkownik komputera stacjonarnego ma duży wybór systemów operacyjnych, które znajdują się na rynku. Według statystyk [6] niezmiennie na czele zestawiania plasują się systemy z rodziny Windows. Na rysunku 1 przedstawiono wykres popularności pod względem całego świata. W latach 2009-2021 ten system był najczęściej wybierany i instalowany na komputerach stacjonarnych na świecie. W 2009 roku aż 94,73% użytkowników zdecydowało się na wybór Windowsa. Jednakże, na przełomie lat można zauważyć duże spadki popularności
i obecnie w 2021 popularność wynosi 74,77%.

Na drugim miejscu w zestawieniu popularności na świecie znajdują się systemy z rodziny OS X, czyli macOS. W 2009 roku popularność tego oprogramowania wynosiła 4,27%, ale co roku liczba użytkowników wzrasta i obecnie wynik wynosi 16,04%. Duża różnica popularności może wynikać z tego, że macOS jest systemem zamkniętym i można go zainstalować tylko i wyłącznie na urządzeniach dedykowanych. Co za tym idzie liczba urządzeń, na których można używać tego oprogramowania jest o wiele bardziej ogranicza niż w przypadku Windowsa czy Linuxa a cena urządzeń dedykowanych jest znacznie wyższa niż w przypadku innych o podobnych parametrach sprzętowych. Duża dysproporcja cenowa jest uwarunkowana tym, że firma Apple optymalizuje swoje oprogramowanie do danych podzespołów, co wpływa na to, że sprzęt przez kilka następnych lat nie traci tyle na wartości oraz na wydajności jak w przypadku innych producentów.

Na trzecim miejscu w tym zestawieniu została scharakteryzowana kategoria „inne". W skład tej grupy wchodzą takie systemy jak ChromeOS, Android, Playstation, nieznane itp. Wynik 6,99% w 2021 roku może wskazywać na to, że dużo użytkowników szuka alternatyw dla popularnych systemów oraz prawdopodobnie różnych modyfikowanych systemów. Wysoka pozycja w zestawieniu popularności wynika także z faktu, że systemy tych producentów są także często wykorzystywane w urządzeniach mobilnych takich jak telefony i tablety.

Na czwartej pozycji znajduje się Linux, który podobnie jak OS X, co roku zyskuje na popularności. Głównym znakiem rozpoznawalnym tego systemu jest otwarty kod źródłowy i fakt, że użytkownik może go modyfikować. W 2009 roku popularność wynosiła 0,69% a w 2021 roku jest to 2,20%. Wpływ na tak małą popularność może wynikać z małej ilości dodatkowego oprogramowania dla typowych użytkowników takie jak: gry, programy czy sterowniki oraz potrzeba zaawansowanej wiedzy do wykorzystania pełni możliwości systemu. W związku z tym systemy Linux są najprawdopodobniej częściej wykorzystywane jako systemy serwerowe a rzadziej jako stacje robocze.



Rysunek 1: Analiza porównawcza popularności systemów na świecie w latach 2009-2021 [6].

2.2. Opis systemów

Na podstawie analizy rynku autorzy do dalszego badania wybrali cztery różne systemy operacyjne:

Windows 10 - system operacyjny od Microsoft [7, 8]. Zaprezentowany został w lipcu 2015 roku. Microsoft zrobił ukłon w stronę użytkowników i wypuszczając go na rynek pozwolił na darmową aktualizację na tę wersję z sytemu Windows 7 oraz Windows 8.1. Podstawową zmianą w tym systemie była eliminacja niedoskonałości, które były związane ze źle odebranym interfejsem użytkownika stworzonym w systemie Windows 8.1.

Windows 11 – najnowsza wersja systemu od Microsoft [9], zaprezentowana 24 czerwca 2021 roku. Tak jak w przypadku wersji Windows 10 producent znowu poszedł w kierunku użytkowników i pozwala na darmową aktualizację do wersji 11 z wersji 10. Oficjalne wydanie zostało zaplanowane na 5 października 2021. Podstawowymi zmianami w odniesieniu do ostatniej wersji jest zmieniony interfejs użytkownika pod względem wizualnym.

MacOS Catalina - szesnaste wydanie systemu z rodziny macOS (wcześniej rodzina nosiła nazwę OS X) od firmy Apple [10, 11]. Zaprezentowany został 3 czerwca 2019 roku a wydany w październiku. Jest to system przystosowany do pracy na komputerach Macintosch, urządzeniach, które dzięki rozwiązaniom projektowych oraz doborze odpowiednich podzespołów (m. in. mikroprocesora) już od pierwszych wersji stały się niezastąpioną dedykowaną platformą sprzętową dla produktów Apple [12]. Zmiany wprowadzone w tej wersji systemu były poczynione by zapewnić lepszą spójność z pozostałymi urządzeniami firmy Apple – iPad i iPhone.

Ubuntu 20.04 LTS - Ubuntu jest systemem operacyjnym stworzonym i rozwijanym przez firmę Canonical [13-15]. Pierwsza wersja oparta o wersję Debian została wydana w 2004 roku i od tego czasu wychodzą coraz to nowsze wersje. Bardzo szybko dystrybucja spotkała się z pozytywny odbiorem stając się jednym z najpopularniejszych systemów z rodziny Linux na świe-

cie. System działa na zasadzie licencji GNU GPL (General Public Licence) [16]. Dodatkowo, systemy Linux wspierają szybki rozwój technologii internetowych [17].

2.3. Scenariusze badawcze

W związku z prowadzonymi badaniami zostały przygotowane dwa scenariusze badawcze mające na celu pokrycie całego obszaru badań.

2.3.1. Scenariusz badawczy – część pierwsza

1. Na podstawie przeanalizowanej literatury [18-21] oraz własnych doświadczeń wyszczególnienie sprawdzanych funkcjonalności.
2. Przygotowanie stanowiska badawczego dla systemu Windows 10, Windows 11, Linux oraz macOS.
3. Instalacja wybranego systemu operacyjnego.
4. Utworzenie i zdefiniowanie metodyki porównawczej kryterium oceny punktowej.
5. Wyliczenie sumarycznej wartości punktowej dla każdego systemu operacyjnego.

2.3.2. Scenariusz badawczy – część druga

1. Przygotowanie stanowiska badawczego dla systemu Windows 10, Windows 11, Linux oraz macOS.
2. Instalacja czystej wersji sytemu – czas mierzony stoperem od momentu pojawienia się ekranu powitalnego z wyborem języku instalacji do momentu pojawianie się ekranu pulpitu po instalacji. W pomiarze zawarte są interakcje użytkownika takie jak kliknięcia i wprowadzanie wymaganego tekstu (np. nazwa użytkownika i hasło). Cała dodatkowa konfiguracja, która nie była konieczna do czasu instalacji była pomijana. Dla każdego systemu proces instalacji wyglądał jednakowo.
3. Ingerencje użytkownika podczas instalacji – liczba kliknięć i wprowadzeń tekstu wymagana przez użytkownika w celu ukończenia podstawowej instalacji systemu operacyjnego.
4. Wyszukiwanie wymaganych aktualizacji – czas mierzony stoperem, który minął od ręcznego wyszukania aktualizacji do czasu rozpoczęcia ich pobierania.
5. Instalacja i konfiguracja wszystkich aktualizacji – czas mierzony stoperem od momentu potwierdzenia instalacji aktualizacji do momentu zakończenia instalacji i konfiguracji.
6. Liczba ponownych uruchomień - liczba restartów systemu wymaganych do pełnej konfiguracji nowo zainstalowanych aktualizacji.
7. Ilość zajmowanego miejsca przez system operacyjny przed aktualizacjami - ilość zajmowanego miejsca na dysku przez system operacyjny zaraz po instalacji, bez żadnych dodatkowych aktualizacji oraz sterowników.
8. Ilość zajmowanego miejsca przez system operacyjny po aktualizacjach – ilość zajmowanego miejsca na dysku przez system operacyjny po wykonaniu wszystkich zalecanych aktualizacji.
9. Uruchomienie systemu operacyjnego – czas mierzony stoperem od momentu kliknięcia przycisku włączającego komputer na stacji roboczej do czasu wyświetlenia pulpitu z dołączeniem logowania.
10. Wyłączenia systemu operacyjnego – czas mierzony stoperem od momentu kliknięcia “zamknij” do momentu wyłączenia się stacji roboczej.
11. Przypisanie adresu IP do interfejsu sieciowego – czas mierzony stoperem od momentu podpięcia kabla sieciowego do stacji roboczej do momentu uzyskania adresu IP. Dla każdego systemu uwzględniono taki sam warunek przypisania adresu sieciowego i jest nim zmiana ikony symbolizującej interfejs sieciowy.
12. Przenoszenie danych w obrębie tego samego dysku – czas mierzony stoperem od momentu rozpoczęcia przenoszenia danych z partycji A do czasu przeniesienia wszystkich danych na partycję B w obrębie tego samego dysku, na którym znajduje się system operacyjny.
13. Przenoszenia danych z dysku SSD na dysk komputera – czas mierzony stoperem od momentu rozpoczęcia przenoszenia danych z dysku zewnętrznego typu SSD poprzez kabel USB do momentu przeniesienia kompletu danych na dysk typu SSD zainstalowanego w komputerze. W danym punkcie obrano następujące kryteria:
    - Próba 1: 1 plik o rozmiarze 10 GB, przenoszony z użyciem graficznego interfejsu użytkownika
    - Próba 2: 10 plików o takim samym rozmiarze, gdzie suma wszystkich wynosi 10 GB, przenoszonych z użyciem graficznego interfejsu użytkownika
    - Próba 3: 100 plików o takim samym rozmiarze, gdzie suma wszystkich wynosi 10 GB, przenoszonych z użyciem graficznego interfejsu użytkownika
14. Sprawdzenie wykorzystania podstawowych podzespołów bez obciążenia - sprawdzenie wykorzystania podzespołów komputera przy pracy jałowej, przy czystym systemie z zainstalowanymi aktualizacjami bez uruchomionych dodatkowych programów. Pomiar przeprowadzono po 3 minutach od pełnego uruchomienia systemu.
15. Komunikacja z urządzeniem optycznym DVD po magistrali szeregowej - czas mierzony stoperem od momentu podłączenia napędu optycznego z płytą CD do portu USB do czasu aż w systemie pojawi się ikona symbolizująca podłączony nowy napęd z płytą.
16. Kompresja plików - czas mierzony stoperem od momentu rozpoczęcia kompresji do momentu pełnej kompresji danych. Dodatkowo zapisywane były rozmiary plików po kompresji. W danym punkcie obrano następujące kryteria:
    - Próba 1: 1 plik o rozmiarze 10 GB, kompresowany z użyciem graficznego interfejsu użytkownika.
    - Próba 2: 10 plików o takim samym rozmiarze, gdzie suma wszystkich wynosi 10 GB, kom-

presowany z użyciem graficznego interfejsu użytkownika.
- Próba 3: 100 plików o takim samym rozmiarze, gdzie suma wszystkich wynosi 10 GB, kompresowany z użyciem graficznego interfejsu użytkownika.

17. Czas dekompresji plików - czas mierzony stoperem od momentu rozpoczęcia dekompresji do momentu pełnej dekompresji danych. Dodatkowo zapisywane były rozmiary plików po dekompresji. W danym punkcie obrano następujące kryteria:
    - Próba 1: 1 plik o rozmiarze 10 GB, dekompresowany z użyciem graficznego interfejsu użytkownika.
    - Próba 2: 10 plików o takim samym rozmiarze, gdzie suma wszystkich wynosi 10 GB, dekompresowany z użyciem graficznego interfejsu użytkownika.
    - Próba 3: 100 plików o takim samym rozmiarze, gdzie suma wszystkich wynosi 10 GB, dekompresowany z użyciem graficznego interfejsu użytkownika.

2.4. Stanowiska badawcze

Stworzone zostały dwa stanowiska badawcze:
- maszyna referencyjna dla systemów Windows oraz Linux,
- maszyna referencyjna, dedykowana przez producenta dla systemu macOS.

Tabela 1: Specyfikacja pierwszego i drugiego stanowiska badawczego

Rodzaj systemu	Windows, Linux		macOS
Nazwa własna	---		Mac mini (Late 2012)
Procesor	Intel Core i5-2320		Intel Core i7-3615QM
Taktowanie procesora	3,00 GHz – 3,30 GHz		2,30 GHz – 3,30 GHz
Liczba rdzeni fizycznych	4		4
Liczba rdzeni logicznych	4		8
Rozmiar pamięci ram	16 GB		16 GB
Typ pamięci	DDR3		DDR3
Taktowanie pamięci	1333 MHz		1600 MHz
Karta graficzna	NVIDIA GeForce RTX 2060S		Intel HD Graphics 4000
Magistrala szeregowa	USB 2.0		USB 2.0
Rodzaj dysku	HDD	SSD	HDD
Pojemność dysku	250 GB	512 GB	1000 GB
Prędkość obrotowa	5400 obr/min	--	5400 obr/min
Prędkość przesyłu danych	300 MB/s	6 GB/s	300 MB/s

a) b)

Rysunek 2: Widok pierwszego oraz drugiego stanowiska badawczego.

a) Stanowisko badawcze przeznaczone dla systemu macOS.

b) Stanowisko badawcze przeznaczone dla systemów Windows 10, 11 oraz Linux Ubuntu 20.04 LTS.

2.5. Metodyka badawcza

Na podstawie prowadzanych badań zostały sporządzane dwie metodyki potrzebne do oceny punktowej funkcjonalności oraz oceny badań.

2.5.1. Metodyka oceny punktowej posiadanych funkcjonalności

Analiza systemu pod względem posiadanych funkcjonalności według wyznaczonego kryterium oceniania:

Jeśli dany system posiada daną funkcjonalność – otrzymuje 1 punkt, w przeciwnym wypadku otrzymuje 0 punktów.

Na podstawie stopnia zaawansowania i łatwości użycia zostaje przydzielona ocena z zakresu 0-5 punktów, w odniesieniu do pozostałych przypadków oraz uwzględniając wykonywanie operacji przez przeciętnego użytkownika systemu operacyjnego, gdzie:

0 punktów – brak danej funkcjonalności

1 punkt – funkcjonalność trudna do odszukania oraz konfiguracji, wymagająca odnalezienia dodatkowych informacji np. w instrukcjach lub poradnikach, oferująca najmniejszy zakres cech dodatkowych.

2 punkty – funkcjonalność trudna do wyszukania, oraz konfiguracji, niewymagająca dodatkowych pomocy zewnętrznych, oferująca małą liczbę dodatkowych opcji.

3 punkty – funkcjonalność o przeciętnej łatwości wyszukania, posiadająca wystarczający zakres opcji oraz niepowodująca trudności konfiguracyjnych.

4 punkty – funkcjonalność łatwa do wyszukania, posiadająca uproszczoną konfigurację oraz oferująca średni zakres dodatkowych opcji.

5 punktów – funkcjonalność bardzo łatwa do wyszukania, posiadający ergonomiczny interfejs, przyjazdy dla użytkownika. Oferuje duży zakres dodatkowych opcji.

2.5.2. Metodyka oceny badań

Systemy zostaną przeanalizowane pod względem czasu potrzebnego na wykonanie podstawowych czynności niezbędnych dla zwykłego użytkownika w odniesieniu do pracy przy użyciu graficznego interfejsu użytkownika. Dodatkowo, wszystkie systemy zostaną porównane pod względem podstawnych cech: wykorzystanie podzespołów, liczba niezbędnych ingerencji użytkownika

przy wykonaniu określonej czynności, ilość zajmowanego miejsca na dysku przed i po aktualizacji. Wszystkie systemy zostaną zainstalowane i sprawdzone na dysku twardym HDD. Wszystkie wyniki zostaną uśrednione w celu określenia miary tendencji centralnej oraz miary położenia rozkładu.

$$\overline{x} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{n} \quad (1)$$

gdzie x_i jest wartością przyjętą przez i-tą obserwację, n jest liczebnością badanej zbiorowości.

Następnie zostanie wyliczone odchylenie standardowe próbki w celu określenia jak bardzo poszczególne wyniki odbiegają od siebie. Pozwoli to na określenie, który system jest w stanie wykonywać poszczególne operacje najstabilniej(najmniejsze wartości odchylenia). Jest to istotny aspekt dla codziennego używania systemu, ponieważ mogą pojawić się duże odchylenia, co będzie zauważane przez użytkownika gołym okiem.

$$\sigma = \frac{1}{V_i}\sqrt{\frac{\varepsilon(x - \bar{x})^2}{(n-1)}} \quad (2)$$

gdzie x jest średnią wartością próbki, n jest wielkością próbki.

3. Wyniki badań

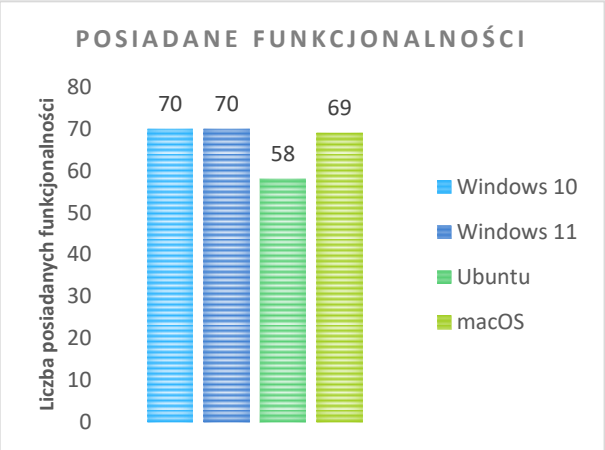
3.1. Opracowane wyniki analizy funkcjonalności systemów

Wszystkie cztery systemy tj. Windows 10, Windows 11, macOS Catalina oraz Ubuntu 20.04 LTS zostały sprawdzone pod względem posiadanych funkcjonalności. W tabeli 2 zostały zaprezentowane wartości punktowe w zależności od systemu operacyjnego i kategorii funkcjonalności.

Tabela 2: Wartości punktowe analizy funkcjonalności systemów

System operacyjny	Windows 10	Windows 11	Ubuntu	macOS
Zabezpieczenia				
Liczba punktów	60	60	36	67
Aktualizacje				
Liczba punktów	20	20	11	20
Produktywność				
Liczba punktów	48	48	31	51
Komunikacja				
Liczba punktów	21	24	2	25
Sterowanie				
Liczba punktów	12	12	6	15
Dostępność				
Liczba punktów	19	20	15	25
Zgodność				
Liczba punktów	35	28	25	22
Wbudowane aplikacje				
Liczba punktów	35	36	42	46
Wygląd/Interfejs				
Liczba punktów	20	24	20	22
Funkcjonalności z instalowania i konfiguracji systemu				
Liczba punktów	22	29	40	20

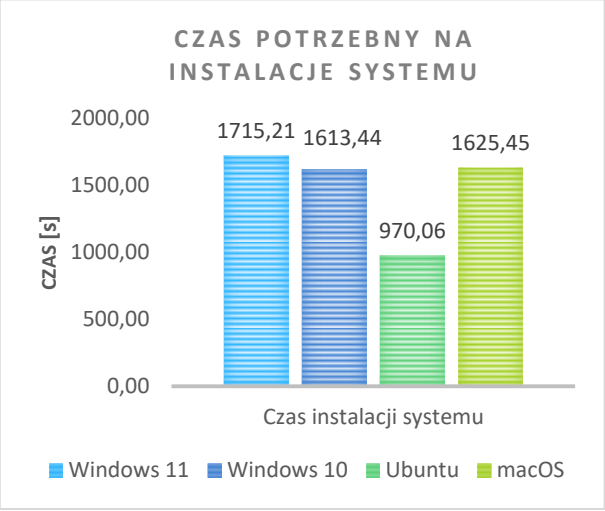
Na Rysunku 3 przedstawiono zestawienie posiadanych funkcjonalności w zależności od systemu operacyjnego. Łączna liczba wszystkich weryfikowanych cech wynosi 84.



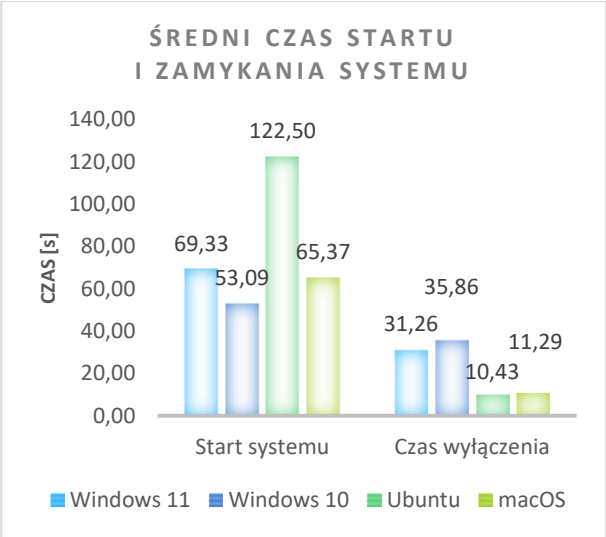
Rysunek 3: Liczba posiadanych funkcjonalności.

3.2. Opracowane wyniki badań dla systemów z dyskiem HDD

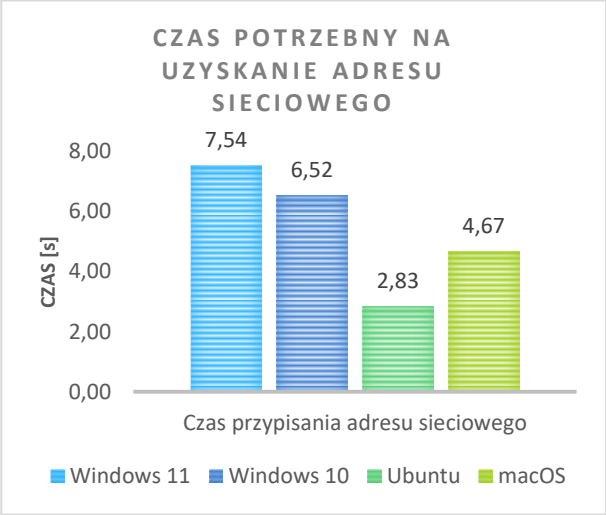
Na rysunkach od 4 do 14 przedstawiono porównanie wyników dla wszystkich badanych systemów z dyskiem HDD.



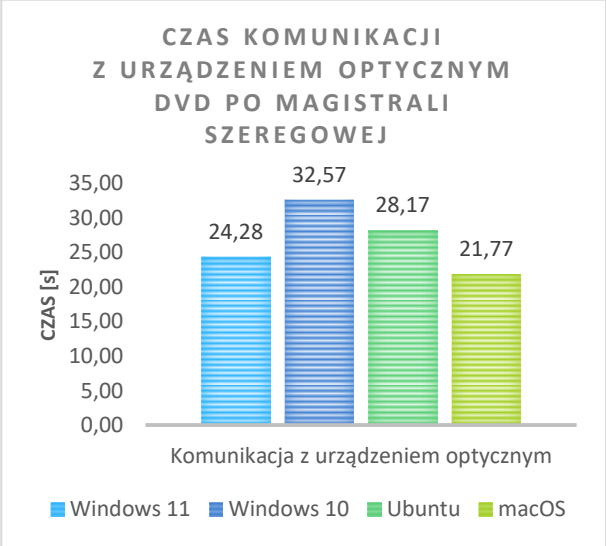
Rysunek 4: Analiza porównawcza czasu instalacji systemu.



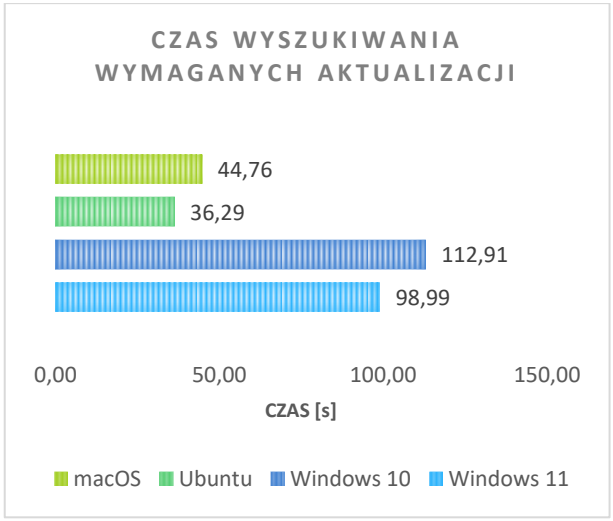
Rysunek 5: Analiza porównawcza czasu uruchomienia i zatrzymania systemu.



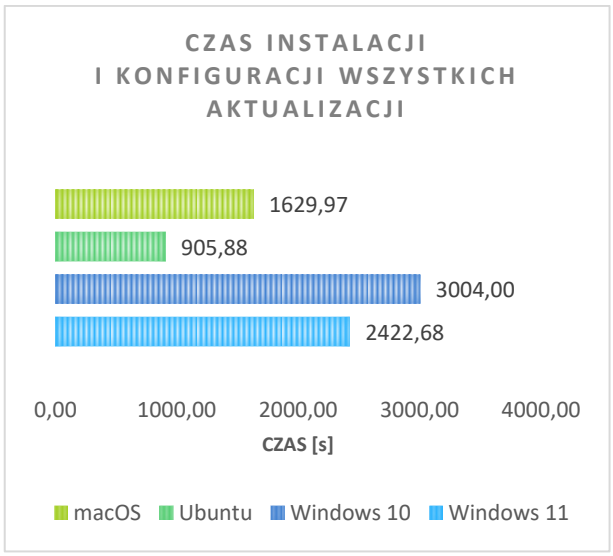
Rysunek 6: Analiza porównawcza czasu potrzebnego na uzyskanie adresu sieciowego.



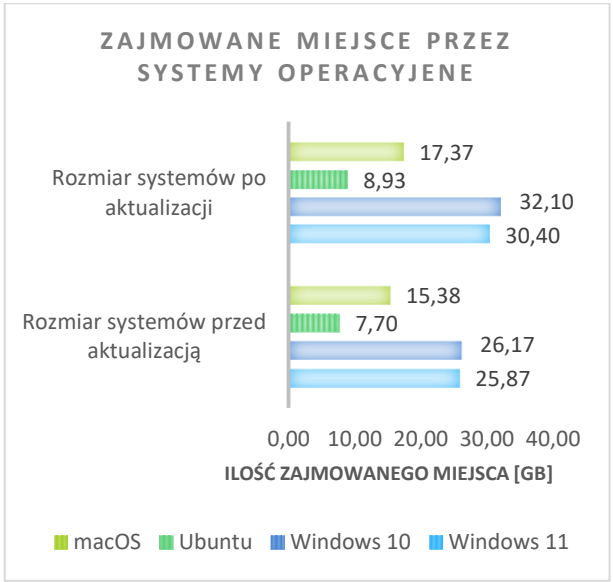
Rysunek 7: Analiza porównawcza czasu komunikacji z urządzeniem optycznym DVD.



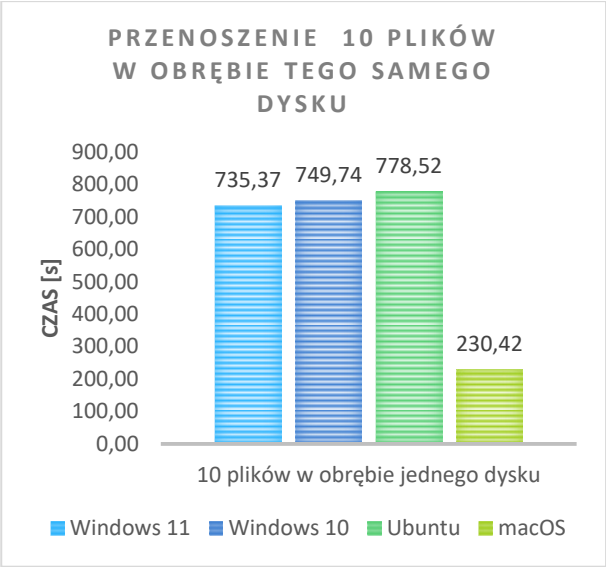
Rysunek 8: Analiza porównawcza czasu wyszukiwania aktualizacji.



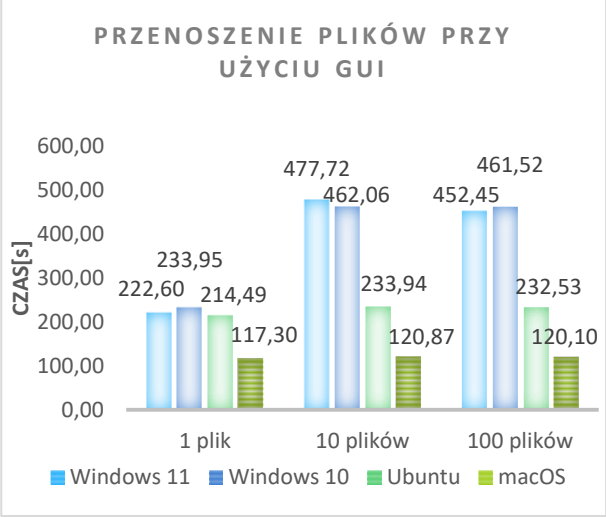
Rysunek 9: Analiza porównawcza czasu instalacji i konfiguracji aktualizacji.



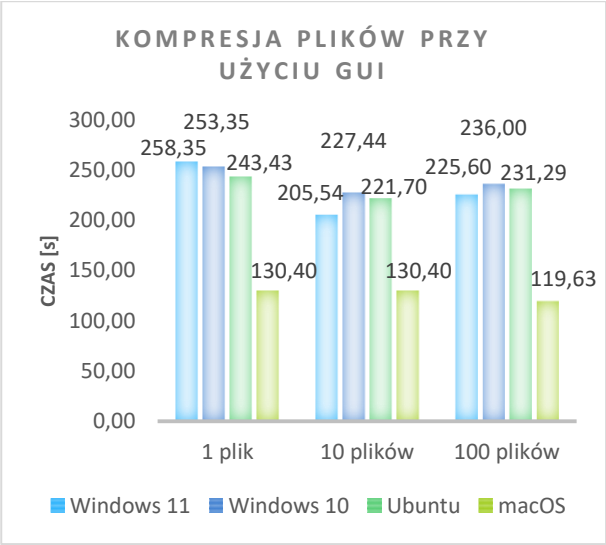
Rysunek 10: Analiza porównawcza rozmiaru systemu po instalacji oraz aktualizacji.



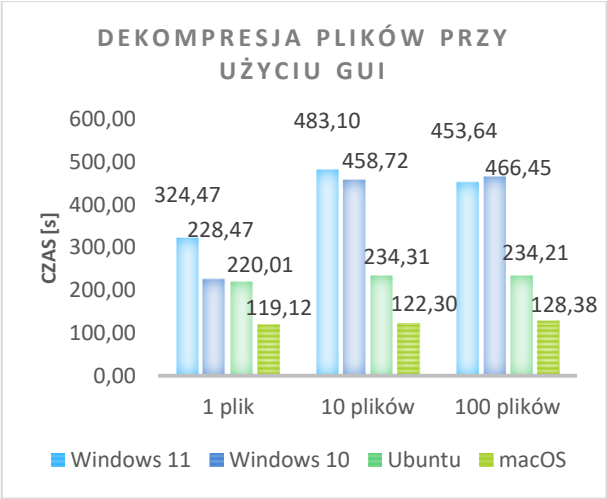
Rysunek 11: Analiza porównawcza czasu przenoszenia plików płaskich w obrębie jednego dysku.



Rysunek 12: Analiza porównawcza czasu przenoszenia plików.



Rysunek 13: Analiza porównawcza czasu kompresji plików.



Rysunek 14: Analiza porównawcza czasu dekompresji plików.

4. Wnioski

Na podstawie przeprowadzonej analizy można stwierdzić, że Windows 11 i Windows 10 posiada największą liczbę funkcjonalności. W związku z tym postawiona hipoteza „System Windows 11 posiada największą liczbę funkcjonalności" została odrzucona. Taka sama liczba funkcjonalności w systemach z rodziny Windows NT najprawdopodobniej wynika z faktu, że firma Microsoft bardzo często wypuszcza aktualizacje systemu, które pozwalają korzystać użytkownikom starszej wersji systemu z nowych funkcjonalności. Zważając na ten fakt, porównanie tych dwóch systemów jest ciężkie w ocenie, ponieważ granica pomiędzy systemami w pewnym czasie się zaciera.

W przypadku systemu macOS można zauważyć, że aplikacje i funkcjonalności są lepiej dopracowane. Wskazuje na to fakt wyższych ocen w kategoriach: zabezpieczenia, komunikacja, produktywność, komunikacja, sterowanie, dostępność oraz wbudowane aplikacje. System Ubuntu 20.04 LTS osiągnął najsłabszy wynik punktowy w kontekście posiadanych funkcjonalności. Pomimo tego, że miał nawet więcej wbudowanych aplikacji niż systemy z rodziny Windows NT, to ich jakość znacząco odbiegała od standardów oferowanych w pozostałych systemach. Tak samo prezentuje się kwestia ocen funkcjonalności.

W drugiej części prac badawczych wszystkie systemy zostały przeanalizowane pod względem czasu wykonywania podstawowych czynności systemowych. Na siedem pomiarów z ogólnego porównania najlepiej zaprezentował się Ubuntu 20.04 i pięciokrotnie osiągnął najkrótszy czas. MacOS Catalina okazał się najszybszy przy teście komunikacji po magistrali szeregowej z urządzeniem optycznym a Windows 10 potrzebował najmniejszą liczbę czasu na uruchomienie systemu. Windows 11 prezentował bardzo porównywalne wyniki do innych systemów, ale z żadnych z ww. czynności nie osiągnął najlepszego rezultatu. W dalszej części, badania zostały oparte na operacjach związanych z plikami. System macOS Catalina potrzebował najmniejszą liczba czasu przy każdej z tych czynności oraz odchylenie standardowe próbki było bardzo niskie, co świadczy, że

system wykonywał każdą próbę w podobnych czasie. Windows 10 i Windows 11 we wszystkich sprawdzanych czynnościach potrzebowały podobną liczba czasu i różnica pomiędzy systemami często oscylowała między 10%.

Odnosząc się do postawionej hipotezy „System macOS potrzebuje najmniejszą liczbę czasu na wykonanie podstawowych czynności" podczas badań została potwierdzona. Analizując wyniki okazało się, że system macOS Catalina na 26 testów różnych funkcjonalności dwudziestokrotnie potrzebował najmniej czasu na wykonanie czynności.

Literatura

[1] P. N. Edwards, The closed world: Systems discourse, military strategy and post WWII American historical consciousness, AI & Society 2 (1988) 245–255.

[2] G. O'Regan, The First Digital Computers, A Brief History of Computing, Springer (2021) 53-70.

[3] Historia związana z stworzeniem pierwszego komputera – ENIAC, http://www.komputer.cuprum.pl/historia/22-eniac.html, [06.11.2021].

[4] J. W. Cortada, The ENIAC's influence on business computing, 1940s-1950s, IEEE Annals of the History of Computing 28 (2006) 26-28.

[5] G. O'Regan, EDVAC and ENIAC Computers, The Innovation in Computing Companion, Springer (2018) 113-117.

[6] Porównanie systemów pod względem popularności, https://gs.statcounter.com/os-market-share/desktop/worldwide, [05.11.2021].

[7] K. Dziedzic, Najważniejsze funkcje Windows 10 w pigułce, Ringier Axel Springer Polska, 2020.

[8] A. Szeląg, Windows 10 PL. Optymalizacja i zaawansowane zarządzanie systemem, Helion 2015.

[9] Dokumentacja Windows, https://docs.microsoft.com/en-us/windows/, [05.11.2021].

[10] Oficjalna dokumentacja do systemu macOS, https://developer.apple.com/documentation/macos-release-notes/macos-catalina-10_15-release-notes, [05.11.2021].

[11] A. Jopek, J. Klopp, P. Marczyński, macOS Catalina. Proste poradniki, Proste Poradniki 2021.

[12] F. Guterl, Design case history: Apple's Macintosh: A small team of little-known designers, challenged to produce a low-cost, exceptionally easy-to-use personal computer, turns out a technical milestone, IEEE Spectrum 21 (1984) 34-43.

[13] Oficjalna dokumentacja do systemu Ubuntu, https://help.ubuntu.com/, [05.11.2021].

[14] A. Hundson, P. Hudson, Ubuntu LTS. Księga eksperta, Helion 2010.

[15] Przewodnik po Ubuntu, 20.04 LTS, https://www.tomekmarszal.pl/dokumenty/Przewodnik%20Ubuntu%2020.04%20LTS%20Focal%20Fosa.pdf, [05.11.2021].

[16] Założenia licencji GNU GPL, https://www.gnu.org/licenses/gpl-3.0.html, [05.11.2021].

[17] W. K. Sedano, M. Salman, Auditing Linux Operating System with Center for Internet Security (CIS) Standard, International Conference on Information Technology, ICIT 2021 – Proceedings, IEEE (2021) 466-471.

[18] Artykuł naukowy o porównaniu Windowsa i Linuxa, https://www.computereconomics.com/article.cfm?id=1128, [05.11.2021].

[19] C. K. Bourne, Linux Tools, Application Administrators Handbook, Morgan Kaufmann 2013.

[20] K. Kowalczyk, Porównanie kosztów rozwiązań serwerowych na przykładzie systemów Windows i Linux, Prace Naukowe Akademii Ekonomicznej we Wrocławiu, Nowoczesne technologie informacyjne w zarządzaniu nr 986 (2003) 408-415.

[21] Porównanie Windowsa, Linuxa, macOS, https://antyweb.pl/windows-macos-oraz-linux-pelne-wad-czyli-idealny-system-nie-istnieje, [05.11.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 203–209
Received: 4 May 2022
Accepted: 31 May 2022

To what extent is the digitization system of Polish medical institutions ready to work online?

W jakim stopniu system cyfryzacji polskich placówek medycznych jest gotowy do pracy online?

Maciej Mikrut

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

Abstract

The aim of the article was to analyze ambulatory health care in Poland in order to show how much infor-mation is processed daily by the Polish health service. To show these values the breakthrough year 2020 was used, when the COVID- 19 pandemic occurred and the healthcare sector had to increase the level of its quality and work efficiency. Electronic medical records, which are used in the work of medical institutions as they are an integral factor in the digitalization of Polish healthcare, were presented, as well as their advantages and disadvantages. A survey on the security of electronic medical records has been conducted among medical personnel through a survey questionnaire. The results showed what the users are most afraid of when working with EDM and how they evaluate the security of electronic medical records. Digitization of the Polish healthcare system is a developing, but also a complex topic, which makes it an area for discussion on how it is going to proceed and what challenges it faces.
Keywords: electronic medical records; outpatient healthcare; electronic patient record

Streszczenie

Celem artykułu była analiza ambulatoryjnej opieki zdrowotnej w Polsce w celu przedstawienia jak dużą ilość informacji przetwarza codziennie polska służba zdrowia. Aby ukazać te wartości został wykorzystany przełomowy rok 2020, kiedy to nastąpiła pandemia COVID – 19 i sektor służby zdrowia musiał zwiększyć poziom swojej jakości i efektywności pracy. Przedstawiona została elektroniczna dokumentacja medyczna, która jest wykorzystywana w pracy w placówkach medycznych, ponieważ stanowi nieodłączny czynnik cyfryzacji polskiej służby zdrowia, a także zaprezentowane zostały jej zalety oraz wady. Przeprowadzone zostały badania dotyczące bezpieczeństwa elektronicznej dokumentacji medycznej wśród personelu medycznego dzięki utworzonemu kwestionariuszowi ankietowemu. Uzyskane rezultaty pokazały czego najbardziej obawiają się użytkownicy podczas pracy z EDM oraz jak oceniają bezpieczeństwo dokumentacji medycznej w formie elektronicznej. Cyfryzacja polskiej służby zdrowia jest tematem rozwojowym, ale również skomplikowanym, co powoduje, że jest obszarem do dyskusji odnośnie sposobu w jaki przebiega i wyzwań przed nią stojących.
Słowa kluczowe: elektroniczna dokumentacja medyczna; ambulatoryjna opieka zdrowotna; elektroniczny rekord pacjenta

*Corresponding author

Email address: maciekmikrut@wp.pl (M. Mikrut)

1. Wstęp

Rozwój technologii, który towarzyszy nam każdego dnia nie omija sektora ochrony zdrowia. Postęp, który się w nim pojawił, to już nie tylko sfera teoretyczna i projektowa. To proces trwający w Polsce od kilka lat. W ostatnim czasie – od 2020 roku – znacząco zwiększył się popyt na usługi elektroniczne. Bezpośrednią przyczyną był wybuch pandemii COVID – 19. Placówki medyczne zostały zmuszone do znacznego ograniczenia bezpośredniego kontaktu z pacjentami. Tak, aby zapobiec w jak największym stopniu rozprzestrzenianiu się choroby. Jednym ze skutków tej sytuacji było zwiększenie natężenia teleporad oraz udzielania pacjentom pomocy przy użyciu internetu i opcji, którymi on dysponuje. Elektroniczna dokumentacja medyczna posiada ogromne możliwości, a twórcy systemów medycznych cały czas udoskonalają ich użytkowanie. Celem jest poprawa jakości i eliminacja problemów, które pojawiają się w trakcie ich użytkowania.[1]

Elektroniczną dokumentacją medyczną nazywamy dokumenty, które są generowane w formie cyfrowej. Podstawowym aktem prawnym regulującym to rozwiązanie jest ustawa z dnia 28 kwietnia 2011 roku o systemie informacyjnym w ochronie zdrowia. W art. 2 pkt. 6 zaznacza, że elektroniczny dokument medyczny jest dokumentem sporządzonym w formie elektronicznej z kwalifikowanym podpisem elektronicznym, zaufanym lub osobistym bądź podpisanym dokumentem metodą potwierdzającą źródło i integralność danych dostępnych w informacji oraz udostępnionym przez ZUS: recepty, określone w przepisach wydanych na podstawie art. 13a, skierowanie określone w przepisach wydanych na podstawie art. 59aa ust. 2 ustawy z dnia 27 sierpnia 2004 r. o świadczeniach opieki zdrowotnej finansowych ze środków publicznych.[2] Jednocześnie na podstawie

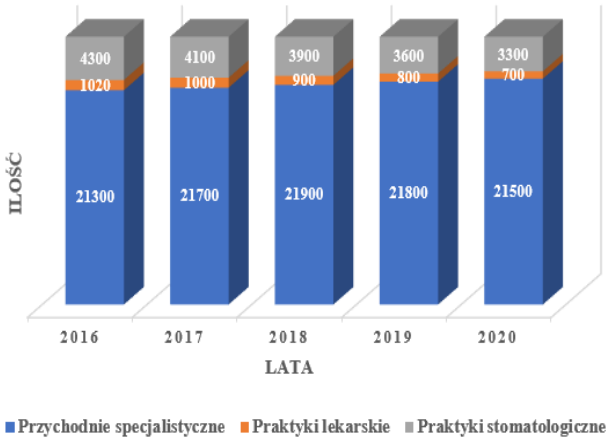
art. 13a zostało opublikowane rozporządzenie Ministra Zdrowia z dnia 8 maja 2018 r. w sprawie rodzaju elektronicznej dokumentacji medycznej. Wykazano, że elektroniczna dokumentacja medyczna (nie wliczając e – recept oraz e – skierowań) to: informacje diagnostyczne o chorobach, urazach i innych problemach zdrowotnych, wynikach przeprowadzonych badań, informacje o diagnozie, leczeniu, rokowaniu, lekach wydawanych na receptę, środkach spożywczych i wyrobach medycznych specjalnego przeznaczenia żywieniowego, okresie ich stosowania i zażywania oraz o umówionych wizytach kontrolnych, wyniki pacjenta z laboratorium wraz z opisem oraz karta informacyjna z leczenia szpitalnego.[3]

Elektroniczna dokumentacja medyczna jest dziedziną, którą cechuje ciągły i dynamiczny rozwój Placówki medyczne opierają swoją pracę o wybrany system. Oprogramowanie w dziedzinie medycyny powinno być jak najbardziej kompletne, ponieważ codziennie placówki medyczne przyjmują setki pacjentów i każda informacja o stanie zdrowia pacjenta musi być precyzyjnie zapisana.[4]

Jednakże, inwestowanie w cyfryzację napotyka też na problemy wynikające z pandemii, a wsparcie NFZ ograniczone jest do placówek POZ oraz biorących udział w pilotażach, a perspektywa wykorzystania w tym celu środków unijnych jest zaledwie mgliście zarysowana. Zgodnie z zarządzeniem NFZ o dofinansowane na cyfryzację (w tym serwery, oprogramowanie) mogą się ubiegać tylko przychodnie podstawowej opieki zdrowotnej. A z listów nadsyłanych do NRL, jak też z opinii, jakie płyną z okręgowych izb lekarskich wynika, że prowadzący podmioty i praktyki mają świadomość nadchodzącej „nowej cyfrowej ery" w prowadzeniu dokumentacji, jednak na pierwszym miejscu jest obecnie przetrwanie ich placówek w obliczu pandemii.

2. Analiza ambulatoryjnej opieki zdrowotnej w roku 2020

Dzięki danym zbieranym i prezentowanym przez Główny Urząd Statystyczny można przedstawić zarys opieki ambulatoryjnej na przestrzeni lat 2016 – 2020. Rysunek nr 1 pokazuje, że świadczenia te skupiają się głównie w przychodniach, a – co symptomatyczne – mniej usług udzielanych jest przez praktyki lekarskie.

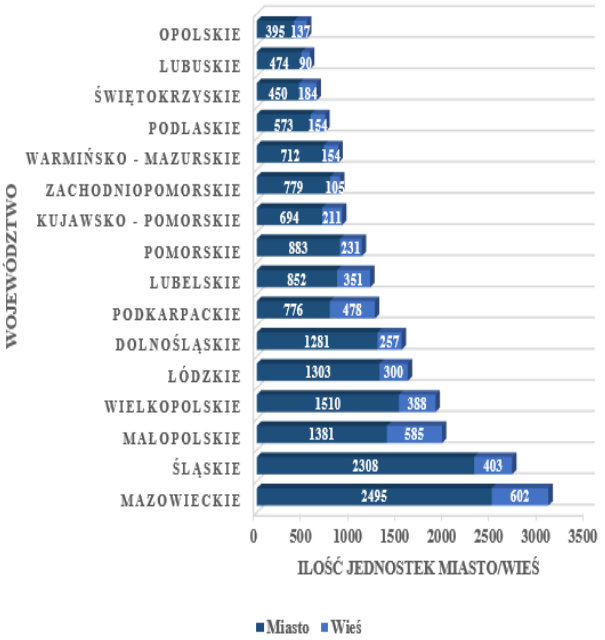


Rysunek 1: Liczba przychodni i praktyk w latach 2016 – 2020 (źródło danych GUS).

Liczba przychodni wzrosła w roku 2020 o 200 placówek w porównaniu z rokiem 2016. Z tą uwagą, że na przestrzeni lat ich liczba wynosiła więcej niż w roku 2020. Spowodowane jest to rocznym limitem ustalonym arbitralnie przez NFZ. Wielu pacjentów zamiast trafić do lekarza w październiku czy listopadzie, czeka na wizytę dłużej. To m.in. w ten sposób brak pieniędzy w systemie wydłuża kolejki do specjalistów. To skutkuje również zamykaniem przychodni specjalistycznych.

Liczba praktyk lekarskich zmniejszyła się o około 300 w porównaniu do roku 2016. Przychodnie, w których pracuje kilku lub nawet kilkudziesięciu lekarzy lepiej zabezpieczają usługi medyczne dla pacjentów chociażby w okresach urlopowych lub innych nieplanowanych, niespodziewanych niedyspozycjach personelu medycznego.

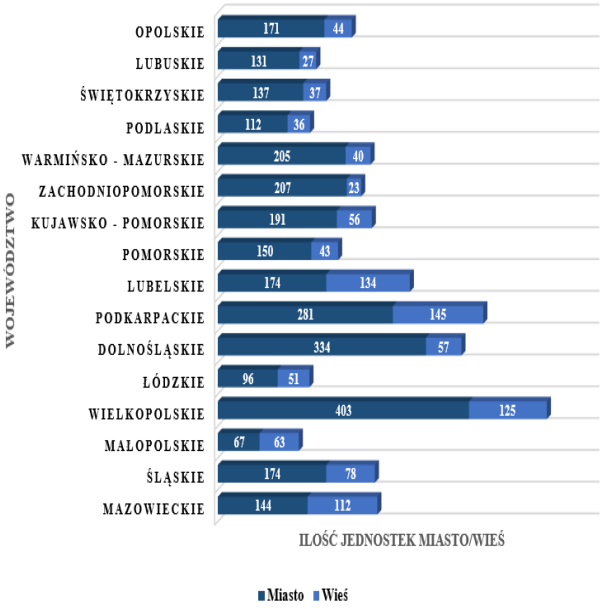
Na poniższym rysunku nr 2 liczba przychodni specjalistycznych i praktyk lekarskich według województw potwierdza, że potrzeby medyczne realizowane są przede wszystkim w największych skupiskach ludności (aglomeracje miejskie), Dotyczy to nie tylko liczby przyjmowanych pacjentów, ale także różnorodności i dostępności usług związanych ze ochroną zdrowia.



Rysunek 2: Przychodnie według charakteru miejscowości i województw w 2020 r. (źródło danych GUS).

W małych miejscowościach mieszkańcy są pozbawieni opieki lekarza rodzinnego. Jest to wynikiem deficytu osób, które mogłoby się podjąć praktyki. Drugą przyczyną jest to, że medycy -przede wszystkim młodzi – decydują się na pracę w dużych miastach. Sytuacja ta nie ulega poprawie, mimo złagodzenia wymaga formalnych co do kwalifikacji, jakie musi posiadać lekarz pracujący na wsi (Rys. 3). O brakach kadry medycznej w Polsce powszechnie wiadomo. Natomiast mniej nagłośniony jest problem w kontekście lekarzy rodzinnych. Rozgłos medialny zyskują przede wszystkim

sytuacje dotyczące zamykania oddziałów w szpitalach z powodu braków kadrowych. Ubywanie praktyk lekarskich to zjawisko, które nie jest zauważalne z perspektywy mieszkańców dużych miast. W mniejszych ośrodkach – przede wszystkim na wsiach - coraz częściej dochodzi jednak do takich sytuacji.

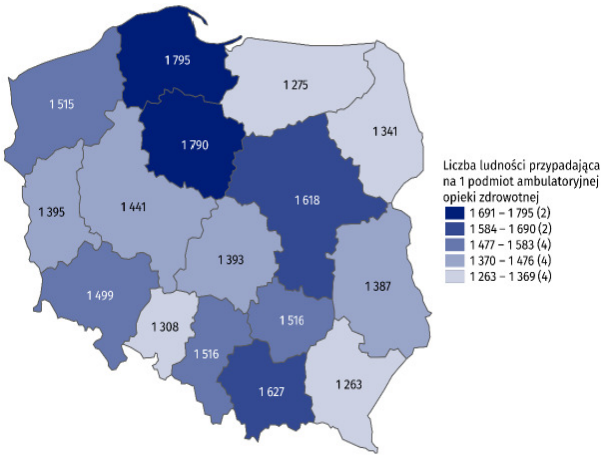


Rysunek 3: Praktyki lekarskie i stomatologiczne realizujące świadczenia finansowane ze środków publicznych według charakteru miejscowości i województw w 2020 r. (źródło danych GUS).

Rysunek nr 4 przedstawia liczbę ludności przypadającą na jedną placówkę medyczną. W Polsce 2020 r. - na podmiot ambulatoryjnej opieki zdrowotnej przypadało średnio 1498 osób (o 37 więcej niż przed rokiem). Najwięcej ludności na jedną przychodnię i praktykę odnotowano w województwie pomorskim (1795), najmniej w województwie podkarpackim – 1263.

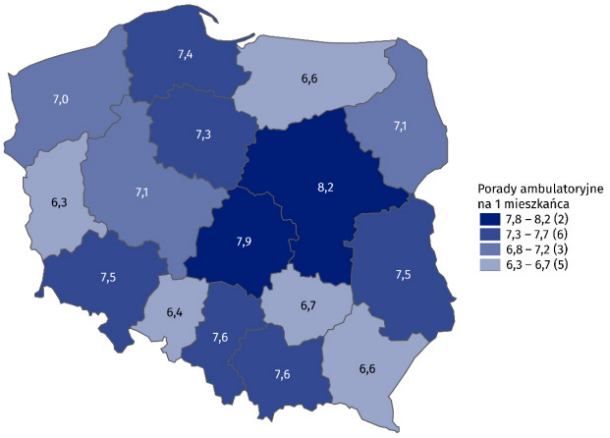
W 2020 r. w przychodniach i praktykach udzielono 283,1 mln porad w ambulatoryjnej opiece zdrowotnej – 256,6 mln lekarskich (spadek o prawie 12% w skali roku) z czego 36% stanowiły teleporady. Wysoki procent udzielania ich udzielania w 2020 r. jest skutkiem wybuchu epidemii COVID – 19. Aby zapobiec rozprzestrzenianiu się wirusa placówki medyczne znacznie ograniczyły kontakt fizyczny z pacjentem i zdecydowały się na udzielanie teleporad w sytuacjach nie stanowiących zagrożenia życia. Liczba porad stomatologicznych wyniosła 26,5 mln (spadek o 22,9% w porównaniu z rokiem poprzednim). Spośród porad lekarskich 156,2 mln stanowiły porady udzielone w podstawowej opiece zdrowotnej2 (spadek o 10,1% w stosunku do 2019 r.), a 100,5 mln – porady w opiece specjalistycznej (spadek o 14,6% w skali roku).

Liczba ludności przypadająca na jeden podmiot (rysunek nr 4) pokazuje największe obciążenie województw najbardziej uprzemysłowionych. Spadek o 10,6% udzielanych porad w POZ i spadek o 14,6% w AOS świadczy o popycie tych usług w społeczeństwie. Szczególnie spadek usług w ramach AOS potwierdza duże zapotrzebowanie na specjalistykę. Może więc niepokoić zapowiedź zmniejszenia liczby tych świadczeń w kolejnym kontraktowaniu przez NFZ.



Rysunek 4: Liczba ludności przypadająca na 1 podmiot ambulatoryjnej opieki zdrowotnej według województw (stan na 31 grudnia 2020 r. - źródło danych GUS).

Na rysunku nr 5 zaprezentowano przeciętną liczbę porad ambulatoryjnych w przeliczeniu na 1 mieszkańca w 2020 r., która wyniosła 7,4. Najwyższą wartość tego wskaźnika odnotowano w województwie mazowieckim (8,2), a najniższą w województwie lubuskim (6,3).[5]



Rysunek 5: Porady ambulatoryjne na 1 mieszkańca według województw w 2020 r (źródło danych GUS).

3. Zalety i wady elektronicznej dokumentacji medycznej

Zaprezentowane powyżej wartości pokazują z jak dużą ilością informacji muszą mierzyć się placówki medyczne. Na wprowadzeniu elektronicznej dokumentacji medycznej zyskują pacjenci i personel jednostki, który dzięki szybkiemu generowaniu raportów, przejrzystemu wglądowi w podsumowanie stanu i historię choroby pacjenta posiada większy komfort pracy. Dodatkowo poprawiają go tablety umożliwiające lekarzom wgląd do dokumentacji chorego bezpośrednio przy jego łóżku. Zamiana tradycyjnej dokumentacji na cyfrową likwidu-

je również konieczność utrzymywania znaczącej powierzchni magazynowej potrzebnej dawniej do przechowywania dokumentów archiwalnych.[6]

Rezygnacja z prowadzenia dokumentacji w formie papierowej przyczyni się do zwiększenia ochrony danych z powodu ograniczeń dostępu. Oznacza to, że wszystkie dane będzie posiadał lekarz, do którego indywidualnie przypisany jest dany pacjent. Zrezygnowanie z prowadzenia dokumentacji w formie papierowej zmniejszy prawdopodobieństwo dostępu do wrażliwych danych pacjenta przez osoby postronne lub nieupoważnione. Pomimo to, prawidłowym podejściem byłoby określenie już na samym początku, który lekarz bądź pracownik personelu w placówce medycznej będzie miał wgląd do danych pacjenta. Dzięki temu osoby nieupoważnione do informacji bałyby się konsekwencji z powodu nieprzestrzegania tajemnicy lekarskiej.[7]

Tylko wyznaczeni użytkownicy medyczni mogą w dowolnej chwili i na każdym ze stanowisk w szpitalu uzyskać informacje na temat danego pacjenta, Wszystkie zdarzenia z procesu leczenia zostają zapisane w systemie. Wdrożenie elektronicznej dokumentacji medycznej znacząco zmniejsza ryzyko powstania błędu informacyjnego. Skraca czas obsługi pacjenta a co za tym idzie postawienia diagnozy. Zapobiega to zjawisku zwiększonej zachorowalności z powodu braku podjętych działań.

Z drugiej strony elektroniczna dokumentacja medyczna uzależniona jest od programowania. To co, zapisane na papierze i schowane w odpowiedniej kartotece już tam pozostanie. Elektroniczna dokumentacja medyczna jest uzależniona od pracy sprzętu, łącza internetowego i samego oprogramowania. Wystarczy drobny problem (np. brak połączenia z siecią), by lekarze stracili dostęp do danych.

Istnieje również ryzyko zwielokrotnienia opuszczonych terminów. Zapisywanie się na wizytę przez internet może spowodować, że pacjenci będą się czuć mniej zobowiązani do pojawienia się w gabinecie lekarskim. Ten sam problem może się pojawić w przypadku tradycyjnego zapisu na wizytę, jednak przy wersji elektronicznej może być występować częściej.

Koszt zakupu profesjonalnego oprogramowania medycznego na początku może się okazać dla przychodni niespodziewanym kosztem, którego nie musiała ponosić, prowadząc dokumentację tradycyjną.

3.1. Analiza bezpieczeństwa elektronicznej dokumentacji medycznej w opinii użytkowników

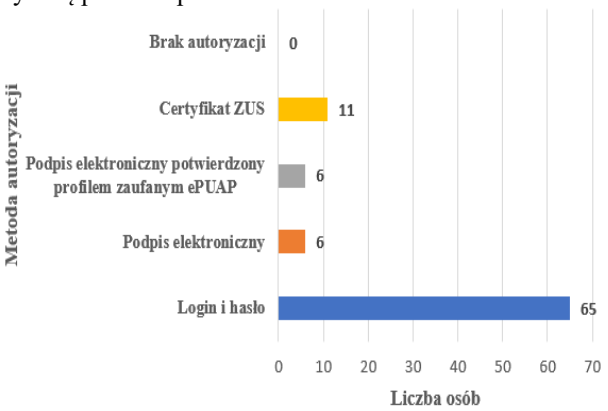
Badanie analizy bezpieczeństwa oraz użyteczności elektronicznej dokumentacji medycznej było elektronicznym badaniem ankietowym. Kwestionariusz miał na celu ocenę modułu EDM w takich aspektach jak jakość, użyteczność oraz bezpieczeństwo. Ankieta została przeprowadzona w formie elektronicznej za pomocą Google Forms.

Grupę badaną stanowiły osoby w przedziale wiekowym 18 - 25 lat, 26 - 40 lat, 41 - 59 lat, 60 - 69 lat. Były one związane z medycyną tj. lekarze, pielęgniarki oraz personel medyczny, którzy na co dzień w swoim zawodzie są użytkownikami elektronicznej dokumentacji medycznej. Dzięki swojemu doświadczeniu mogli w sposób rzetelny ocenić omawiany moduł EDM.

Elektroniczna dokumentacja medyczna służy do gromadzenia i przechowywania danych na temat stanu zdrowia pacjentów. Przez wzgląd na zawarte w niej dane osobowe - w tym wrażliwe - osoba odpowiedzialna za system EDM musi zadbać o jej szczególną ochronę.

Celem prowadzenia elektronicznej dokumentacji medycznej jest zagwarantowanie wymiany informacji pomiędzy podmiotami realizującymi zadania w ramach systemu ochrony zdrowia. Ponadto korzystanie z EDM pozwala na cyfryzację archiwum kartotek, jak i integrację dokumentów tworzonych w różnych systemach służby zdrowia.[8] Aby zadbać o bezpieczeństwo danych osobowych pacjentów użytkownicy elektronicznej dokumentacji medycznej posługują się podpisem elektronicznym. Takie rozwiązanie pozwala na weryfikację podpisu elektronicznego i umożliwia integralność dokumentu. Poza tym znakują datą i czasem dokument, do którego wprowadzili jakiekolwiek zmiany. Każdy użytkownik systemu ma spersonalizowany login i hasło do własnego konta w systemie elektronicznej dokumentacji medycznej, które powinno składać się przynajmniej z 8 znaków i zawierać małe oraz wielkie litery, cyfry lub znaki specjalne. Aby dodatkowo zabezpieczyć system EDM przed osobami nieupoważnionymi, zmiana hasła powinna następować nie rzadziej niż co 30 dni. Dostęp do danych pacjentów jest zatem możliwy tylko po wprowadzeniu identyfikatora i dokonaniu uwierzytelnienia. Następuje ono za pośrednictwem certyfikatu pochodzącego z kwalifikowanego centrum certyfikacji.

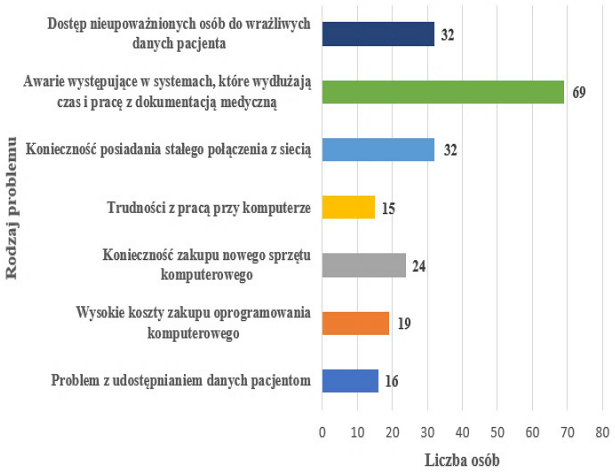
Rysunek nr 6 przedstawia metody autoryzacji przed rozpoczęciem pracy z elektroniczną dokumentacją medyczną przez respondentów:



Rysunek 6: Metody autoryzacji stosowane przez respondentów.

Było to pytanie wielokrotnego wyboru. Większość badanych odpowiedziało, że metodą zabezpieczającą przed osobami nieupoważnionymi w systemie jest podanie własnego loginu i hasła przed rozpoczęciem pracy z EDM. Niektórzy z ankietowanych w swojej pracy używają podpisu elektronicznego, podpisu elektronicznego z profilem zaufanym ePUAP lub certyfikatu ZUS. Powyższe metody autoryzacji zwiększają bezpieczeń-

stwo danych zarejestrowanych w cyfrowej dokumentacji medycznej i każdy członek personelu medycznego powinien rozważyć uwierzytelnianie dwuskładnikowe do konta, aby jak najlepiej chronić dane wrażliwe pacjenta (np. podpis elektroniczny potwierdzony profilem zaufanym).



Rysunek 7: Obawy użytkowników związane w pracy z modułem EDM.

Jednym z pytań wielokrotnego wyboru udostępnionym ankietowanym dotyczyło ich obaw związanych w pracy z elektroniczną dokumentacją medyczną. Zgodnie z wykresem zaprezentowanym na rysunku 7 największym problemem według respondentów są awarie systemu w trakcie pracy. Następnym co do częstości wyborem, jest obawa związana z dostępem nieupoważnionych osób do danych wrażliwych pacjenta oraz konieczność stałego połączenia z siecią. Badana grupa jak problem wskazała także wysokie koszty zakupu nowych komputerów jak i oprogramowania, które miałoby być wdrożone w placówce. Najmniejsze wyzwanie dla ankietowanych stanowiłyby ewentualne trudności w pracy z nowym rozwiązaniem oraz problem z udostępnieniem danych pacjentom.

Rysunek 8: Występowanie awarii podczas pracy z elektroniczną dokumentacją medyczną.

Potwierdzeniem obaw o awarię systemów jest powyższy rysunek nr 8 na którym można dostrzec, że prawie wszyscy respondenci mają problem z oprogramowaniem obsługującym moduł EDM. 69 osób wyraża swoje niezadowolenie odnośnie awarii w systemach. Jest to niepokojący aspekt w kontekście jakości i efektywności pracy oraz tego, że każda informacja o pacjencie powinna być precyzyjnie i szybko odnotowana.

Rysunek 9: Opinia bezpieczeństwa dokumentacji elektronicznej względem papierowej.

Pomimo obaw o awarie w systemach podczas pracy z elektroniczną dokumentacją medyczną respondenci uważają, że przechowywanie dokumentacji w formie elektronicznej jest wciąż bezpieczniejsze niż w formie papierowej (Rys. 9). Ponad połowa badanej grupy (56 osób) jest przekonana, że prowadzenie dokumentacji pacjenta w formie elektronicznej zapobiegnie ewentualnemu zagubieniu ważnych dokumentów z danymi pacjenta

4. Porównanie elektronicznej dokumentacji medycznej z elektronicznym rekordem pacjenta

Tabela 1 przedstawia porównanie modułu elektronicznej dokumentacji medycznej oraz elektronicznego rekordu pacjenta.[9]

Tabela 1: Porównanie Elektronicznej Dokumentacji Medycznej i Elektronicznego Rekordu Pacjenta

Cecha	EDM	ERP
Tworzenie rekordów pacjenta i ich na aktualizacja	X	X
Śledzenie historii choroby pacjenta	X	X
Dane pochodzące z laboratoriów	-	X
Wyniki badań obrazowych	-	X
Dane demograficzne	-	X
Przesyłanie danych pomiędzy placówkami	-	X
Integracja z systemami innych dostawców	-	X

Elektroniczna dokumentacja medyczna jest cyfrową wersją tradycyjnego papierowego dokumentu o ograniczonej funkcjonalności. Z kolei cyfrowy rekord pacjenta zapewnia dostęp nie tylko do placówki, w której utworzono część dokumentu, ale także do dowolnego miejsca, w którym jest to konieczne (niezależnie od tego, czy jest to placówka prywatna, instytucja publiczna, ambulatoryjna czy szpitalna). Jest rozwiązaniem znacznie ułatwiającym pracę lekarzy oraz personelowi pielęgniarskiemu. Dając im lepsze możliwości diagnostyczne. Lekarz na podstawie historii choroby pacjenta, dostępnej w systemie ERP może szybciej i efektywniej pomóc osobie, w przypadku której brakuje ważnych informacji diagnostycznych. Należy pamiętać, że istnieje również możliwość konsultacji i oceny wyników badań z innym lekarzem, niezależnie od aktualnej lokalizacji.

Pomimo widocznych różnic i nasuwających się wniosków - po porównaniu EDM i ERP - można zaobserwować, że drugie rozwiązanie jest w stanie zaoferować o wiele więcej. Przy wyborze systemu do cyfrowej dokumentacji medycznej trzeba wziąć pod uwagę aktualne potrzeby. Z względów ekonomicznych EDM jest lepszy, ponieważ koszty są niższe Dlatego też praca jest oparta na elektronicznej dokumentacji medycznej w Polsce. Jednakże, nie można zapominać o tym, że wymienione moduły przyniosły korzystne zmiany w sektorze ochrony zdrowia. Umożliwiają jeszcze większą efektywność pracy placówki medycznej.[10]

5. Wnioski

Podsumowując cyfryzację polskiej służby oraz pracę z elektroniczną dokumentacją medyczną, które były tematem tego artykułu – dzięki danym udostępnianym przez Główny Urząd Statystyczny było możliwe przedstawienie ambulatoryjnej opieki zdrowotnej w roku 2020. Był on przełomowym w przyspieszeniu cyfryzacji polskiej służby zdrowia ze względu na wybuch pandemii COVID – 19.

Analiza ambulatoryjnej opieki zdrowotnej pokazała jak na przestrzeni lat 2016 – 2020 zmieniała się ilość przychodni specjalistycznych, praktyk lekarskich jak i stomatologicznych. Ukazało to, że ich liczba podczas pandemii oraz arbitralnie ustalonym limitem przyjęć przez NFZ powiększyła się o 200 placówek. Oba rodzaje praktyk znacznie zmalały, ponieważ młodzi lekarze wolą wyjechać z niewielkich ośrodków miejskich do dużych miast, aby móc rozwijać swoje umiejętności. Potwierdziło się, że potrzeby medyczne realizowane są głównie w największych aglomeracjach miejskich. Jest to podyktowane nie tylko liczbą stałych mieszkańców, ale również usług udzielanych pacjentom czasowo przebywającym w mieście (np. ze względu na pracę) i potrzebują zabezpieczenia medycznego.

Statystyki udostępnione przez GUS, zawarte na mapie Polski z podziałem na województwa pokazują również, że liczba ludności przypadająca na jeden podmiot ambulatoryjnej opieki zdrowotnej waha się w granicach 1200 – 1300 osób. Natomiast w najbardziej zaludnionych aglomeracjach dochodzi on do 1800. Podobnie jest w przypadku porad ambulatoryjnych przypadających na jednego mieszkańca według województw. W najbardziej zaludnionych miejscach wynosi ponad 8, z kolei w mniej zaludnionych województwach wartości wahają się w przedziale 6 – 7 porad.

Przedstawione statystki miały na celu ukazać jak bardzo placówki medyczne są obciążone w codziennej pracy. Zwłaszcza w największych aglomeracjach oraz naprowadzić na poruszenie kolejnej kwestii jaką jest cyfryzacja polskiej służby zdrowia za pośrednictwem elektronicznej dokumentacji medycznej, która ma być rozwiązaniem dla placówek, zwiększając jakość ich pracy. Zostały omówione zarówno zalety jak i wady EDM. Faktem jest, że elektroniczna dokumentacja medyczna zwiększa komfort pracy personelu medycznego oraz likwiduje konieczność utrzymywania znaczącej powierzchni magazynowej potrzebnej do przechowywania dokumentów archiwalnych. Wadą jest to, że wystąpienie jednej usterki w systemie (np. awaria sieci) skutkuje utraceniem dostępu do danych przez lekarza. Zostało to ukazane również za pośrednictwem pytania ankietowego dotyczącego awarii systemu, w którym 92% wyraziło swoje niezadowolenie z powtarzającego się zjawiska. Wprowadzenie EDM wiąże się również z kosztami, na które nie każda placówka może sobie pozwolić. Swoje obawy ukierunkowali również na m.in. konieczność stałego połączenia z siecią a także zakupu nowego sprzętu komputerowego.

Bezpieczeństwo danych wrażliwych pacjenta powinno być priorytetem dla każdego lekarza. W związku z powyższym grupa badanych została zapytana o metody autoryzacji przed rozpoczęciem pracy z elektroniczną dokumentacją medyczną. Większość osób wskazała na podstawową metodę jaką jest podanie ustalonego loginu i hasła (65 osób). Jednakże kilkunastu ankietowanych stosuje uwierzytelnianie dwuskładnikowe poprzez potwierdzenie swojej tożsamości. Za pośrednictwem podpisu elektronicznego zatwierdzonego profilem zaufanym ePUAP lub certyfikatem ZUS. Taka metoda powoduje, że dane wrażliwe pacjenta są jeszcze bardziej zabezpieczone przed osobami nieupoważnionymi do nich.

Pomimo zwróceniu uwagi na częste awarie systemu, użytkownicy wciąż uważają, że prowadzenie dokumentacji w formie elektronicznej jest bezpieczniejsze niż forma papierowa i zapobiegnie utraty danych pacjenta.

Ostatnim aspektem było porównanie elektronicznej dokumentacji medycznej i elektronicznego rekordu pacjenta. Faktem jest, że ERP oferuje o wiele więcej niż EDM. Wiąże się to jednak z wyższym kosztem. Dlatego też większość placówek medycznych opiera swoją pracę o elektroniczną dokumentację medyczną.

Powyższe badania pokazały dużą ilość informacji, która jest przetwarzana w służbie zdrowia, a także ogromne oczekiwania i zapotrzebowanie na udogodnienia technologiczne w celu ułatwienia pracy personelowi medycznemu. Zostały zaprezentowane zalety i wady elektronicznej dokumentacji medycznej jako narzędzia

usprawniającego jakość funkcjonowania placówki medycznej. Należy pamiętać, że jest to technologia, która ulega ciągłemu rozwojowi w aspekcie informatycznym.

W jakim stopniu system cyfryzacji polskich placówek medycznych jest gotowy do pracy online? Jest to pytanie, na które nie można odpowiedzieć precyzyjnie, ponieważ od 1 lipca 2021 r. usługodawcy z dziedziny medycyny mają obowiązek za pośrednictwem Systemu Informacji Medycznej (SIP) zapewnić możliwość wymiany danych zawartych w elektronicznej dokumentacji medycznej określonej w rozporządzeniu ministra zdrowia z 8 maja 2018 r. niezależnie od stopnia cyfryzacji placówki na danym etapie.

Z uwagi na fakt jak duża ilość danych jest przetwarzana w największych ośrodkach miejskich w Polsce, Ministerstwo Zdrowia powinno właśnie tam położyć nacisk na wsparcie merytoryczne placówek. Dzięki temu zarządzanie nimi i wykonywanie codziennych obowiązków będzie zgodne z przepisami prawa. Na wsiach i małych miastach musi pojawić się większe wsparcie finansowe, aby móc zapewnić stałą opiekę lekarską pacjentom oraz zachęcić młodych medyków do pozostania w regionach Polski, gdzie zapotrzebowanie na stale funkcjonującą placówkę medyczną jest największe. Temat jest rozwojowy, a sam artykuł stanowi jedynie przyczynek do dyskusji na temat cyfryzacji ochrony zdrowia, sposobu w jaki przebiega i wyzwań przed nim stojących.

Literatura

[1] Elektroniczna Dokumentacja Medyczna z perspektywy placówki i pacjenta, https://www.medfile.pl/blog/czym-jest-edm , [22.02.2022].

[2] M. Kamiński, Powierzanie przetwarzania osobowych danych medycznych a perspektywy prowadzenia dokumentacji medycznej w postacie elektronicznej, Wrocław, 2007.

[3] Klimas K., Prawo pacjenta do dostępu do elektronicznej dokumentacji medycznej, Studenckie Prace Prawnicze, Administratywistyczne i Ekonomiczne: Człowiek, przedsiębiorstwo, gospodarka światowa. Wyzwania srebrnej gospodarki, Tom 26, (2018) 95-103, https://doi.org/10.19195/1733-5779.26.6.

[4] D. Szymczyk, A. Horoch, Implementacja elektronicznej dokumentacji medycznej. Część 1 – wpływ na efektywność pracy personelu medycznego, Medycyna Ogólna i Nauki o Zdrowiu, Tom 19, Nr 3, (2013) 319–323.

[5] Ambulatoryjna opieka zdrowotna w roku 2020, https://stat.gov.pl/obszary-tematyczne/zdrowie/zdrowie/ambulatoryjna-opieka-zdrowotna-w-2020-roku,13,5.html, [22.05.2022].

[6] D. Szymczyk, A. Horoch, Implementacja elektronicznej dokumentacji medycznej. Część 2 – korzyści dla uczestników ochrony zdrowia, Medycyna Ogólna i Nauki o Zdrowiu, Tom 19, Nr 3, (2013) 324–330.

[7] J. Pacian., A. Pacian., T. Kulik, H. Skórzyńska, P. Kaczor, Czy prowadzenie elektronicznej dokumentacji może zapewnić bardziej skuteczną ochronę tajemnicy lekarskiej?, Hygeia Public Health 46(4) (2011) 418-422.

[8] D. Szymczyk, A. Horoch, Implementacja elektronicznej dokumentacji medycznej. Część 3 – szkolenie personelu medycznego w zakresie technicznego i prawnego użytkowania danych sensytywnych., Medycyna Ogólna i Nauki o Zdrowiu, Tom 19, Nr 3, (2013) 331–336.

[9] Porównanie Elektronicznej Dokumentacji Medycznej i Elektronicznego Rekordu Pacjenta, czyli cyfryzacja służby zdrowia, https://koltowski.com/2015/04/12/jaka-jest-roznica-miedzy-ehr-a-emr-czyli-cyfryzacja-sluzby-zdrowia/ , [04.03.2022].

[10] Dokumentacja w formie elektronicznej. System EDM i ERP: https://synappsehealth.com/en/articles/i/electronic-medical-documentation-emr-and-ehr-systems/, [03.03.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 210–217
Received: 5 May 2022
Accepted: 23 May 2022

Analysis of the functionality of voice and video communication systems

Analiza funkcjonalności systemów komunikacji głosowej i wizyjnej

Aleksandra Piątkowska

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

Abstract

The article contains a comparison of the functionality and qualitative verification of voice and video communication systems, through tests of voice message shift in time, at different data speed, tests of system use cases and the study of the popularity of applications in various environments and age groups, thanks to an electronic survey . The systems examined were: Microsoft Teams, Zoom, Signal, WhatsApp, Cisco Webex Meetings. The following hypothesis was made: Voice and video communication systems are very popular systems among users, and their functional capabilities are similar.

Keywords: system functionality; voice and video communication systems; quality check

Streszczenie

Artykuł zawiera porównanie funkcjonalności i sprawdzenie jakościowe systemów komunikacji głosowej i wizyjnej, przez testy przesunięcia komunikatu głosowego w czasie o różnej prędkości transmisji danych, testy przypadków użycia systemów oraz badanie popularności aplikacji w różnych środowiskach i grupach wiekowych, dzięki elektronicznemu badaniu ankietowemu. Systemy poddane analizie to: Microsoft Teams, Zoom, Signal, WhatsApp, Cisco Webex Meetings. Postawiono następującą hipotezę: Systemy komunikacji głosowej i wizyjnej są popularnymi systemami wśród użytkowników i posiadają podobne możliwości funkcjonalne.

Słowa kluczowe: funkcjonalność systemu; systemy komunikacji głosowej i wizyjnej; ocena jakości

Corresponding author

Email address: **aleksandra.piatkowska@onet.com.pl,** (A. Piątkowska)

1. Wstęp

Systemy komunikacji głosowej i wizyjnej to aplikacje, dzięki którym użytkownicy mogą komunikować się ze sobą na odległość poprzez pisanie, dzwonienie, przeprowadzanie różnego rodzaju wideokonferencji. Od lat notuje się wzrost popytu na technologie do komunikacji na odległość. Rok 2021, w którym wybuchła pandemia COVID-19 pokazał, że społeczeństwo w szkole i pracy musi posiłkować się zdalnymi systemami komunikacji głosowej i wizyjnej. Dorośli i dzieci zostali w pewnym sensie zmuszeni do nauki obsługi tych systemów i korzystania z nich. Trudno sobie wyobrazić, by młodzież nie uczęszczała na zajęcia, a na czas pandemii nauka została wstrzymana, dlatego właśnie aplikacje do komunikacji okazały się świetnym rozwiązaniem. Na szczęście dają one ogromne możliwości, a od czasu pandemii twórcy aplikacji stworzyli wiele nowych funkcjonalności, udoskonalając ich użytkowanie.

Wyzwanie postawione w celu realizacji artykułu to zbadanie jakości i użyteczności systemów komunikacji głosowej i wizyjnej, podsumowanie ich funkcjonalności oraz zbadanie opóźnień czasowych komunikatu głosowego na różnej prędkości transmisji danych oraz uzasadnienie hipotezy: Systemy komunikacji głosowej i wizyjnej są popularnymi systemami wśród użytkowników i posiadają podobne możliwości funkcjonalne.

2. Systemy komunikacji głosowej i wizyjnej

Systemy to usługi chmurowe, posiadające różnego rodzaju narzędzia oraz usługi do współpracy w zespole. Definicja takich systemów, z jaką najczęściej się spotykamy to: *„Program komputerowy pozwalający na przesyłanie natychmiastowych komunikatów (komunikacja natychmiastowa – ang. instant messaging) pomiędzy dwoma lub większą liczbą komputerów, poprzez sieć komputerową, zazwyczaj Internet."*[1]

Systemy komunikacji wizyjnej i głosowej to aplikacje, których na rynku w ciągu ostatnich lat pojawiło się bardzo dużo. Podstawą zwiększenia ich liczby było ogromne zapotrzebowanie na wszelkiego rodzaje usługi zdalnej komunikacji oraz przekazywania informacji na odległość.

Wśród komunikatorów występują np.: Skype, Microsoft Teams, Facebook Messenger, Viber, TeamSpeak, Discord, LINE, WhatsApp, Signal, Telegram, Zoom, Cisco Webex, Theema, Wire oraz wiele innych. W artykule zbadano część popularnych systemów, których używa się w szkołach, na uczelniach, w korporacjach jak: Microsoft Teams, Zoom, Whatsapp oraz te mniej znane użytkownikom jak: Signal, Cisco Webex, by zbadać występujące między nimi różnice.

Na temat niektórych systemów i ich wykorzystania np. w szkołach można znaleźć wiele artykułów. Jednym z nich jest *WhatsApp goes to school: Mobile instant messaging between teachers and students* [2]. Zbadano w nim aplikację WhatsApp, stwierdzając iż nie spraw-

dza się ona w szkole. Niektórzy uczniowie nie posiadali smartfonów lub odpowiedniego systemu, by zainstalować aplikację, nauczyciele z kolei irytowali się natłokiem wiadomości od uczniów (wielu uczniów spodziewało się całodobowej dostępności nauczycieli). W artykule autor jako mocną stronę aplikacji określa niski koszt systemu, dostępność i natychmiastowość interakcji. Wskazuje jednak, iż jest to aplikacja do użytku prywatnego.

Systemy poddano analizie i przeprowadzono już badania, ale wciąż istnieje wiele obszarów i funkcjonalności, które nie zostały zbadane. Podobne badanie części systemów Zoom, Google Meet, Microsoft Teams, WebEx Teams i GoToMeeting zostało wykonane w artykule *Updated Comparative Analysis on Video Conferencing Platforms- Zoom, Google Meet, Microsoft Teams, WebEx Teams and GoToMeetings* [3]. Zbadano jednak tylko część funkcjonalności. Autor wyciągnął ważne wnioski, iż w każdej aplikacji pojawiają się wady, natomiast największym priorytetem powinny być funkcjonalności bezpieczeństwa aplikacji.

Następny artykuł badający możliwości systemów takich jak: Zoom, Skype, Microsoft Teams i WhatsApp to *Evaluating videoconferencing systems for the quality of the educational experience* [4]. Systemy zbadano pod kątem nauki online i ich użyteczności w nauczaniu na odległość.

3. Analiza porównawcza systemów

Do obróbki wyników użyto statystyki matematycznej. W przypadku ankiety liczono np. procentową liczbę osób, znających dany system i nieznających danego systemu itd. Analiza ankiety zawiera również graficzną prezentację danych statystycznych, czyli wykresy słupkowe, kołowe, pierścieniowe i kolumnowe, w celu zwizualizowania wyników ankiety.

Analiza funkcjonalności zawiera wyliczenia procentowe pokazujące, który system posiada największy procent funkcjonalności, a który najmniejszy oraz wyliczenie różnic w dostępności funkcjonalności między systemami.

W skład analizy badania danych osobowych pobieranych w aplikacjach, wchodzą również wykresy słupkowe, aby w odpowiedni sposób zwizualizować proporcje między pobieraniem różnego rodzaju danych.

Przeanalizowano również czas dotarcia komunikatu głosowego na rożnej prędkości łącza do odbiorcy. Z 10 pomiarów obliczono średnie opóźnienie dotarcia komunikatu dla każdego systemu i dla różnych prędkości. Następnie przedstawiono dane na wykresach i obliczono różnice procentowe między przesunięciami w aplikacjach.

3.1. Analiza porównawcza wymaganych danych osobowych w systemach

Każdy system na etapie instalacji oraz wstępnej konfiguracji prosi o podanie danych osobowych. Na potrzeby artykułu pokazano jakie dane zbierają badane systemy.

Na temat niektórych systemów i ich bezpieczeństwa powstały już artykuły naukowe. Jeden z ciekawszych dotyczy WhatsApp. System posiada zróżnicowane opinie, jeśli chodzi o bezpieczeństwo aplikacji. Artykuł *Insecure Whatsapp Chat History, Data Storage and Proposed Security* podkreśla, że system jest dobrze zabezpieczony z sieci, ale przypomina też, że w przeszłości wiadomości w pamięci lokalnej nie były skutecznie zabezpieczone [5]. Aplikacja posiadała nieodpowiedni algorytm, co powodowało lukę. WhatsApp poprawił swoje działanie, natomiast specjaliści od bezpieczeństwa IT wciąż o tym pamiętają.

Aby poprawnie sprawdzić, jakie dane są przetwarzane w aplikacji, przygotowano tabelę 1. z wykorzystaniem danych osobowych, których wymaga się przy instalacji (1 – „wymagane”, 0 – „niewymagane”). Na etapie instalacji systemów weryfikowano poniższe informacje: adres email, imię, nazwisko, datę urodzenia, kraj, adres, listę kontaktów, lokalizację, pesel i numer telefonu.

Tabela 1: Dane przetwarzane we wszystkich badanych systemach (opracowanie własne)

Dane	MS Teams	Zoom	Signal	WhatsApp	Cisco Webex
Adres e-mail	**1**	**1**	0	0	**1**
Imię	**1**	**1**	**1**	**1**	**1**
Nazwisko	**1**	**1**	0	0	**1**
Data urodzenia	**1**	**1**	0	0	0
Kraj	**1**	0	0	**1**	0
Adres	0	0	0	0	0
Lista kontaktów	0	0	**1**	**1**	0
Lokalizacja	0	0	0	0	0
Pesel	0	0	0	0	0
Numer telefonu	0	0	**1**	**1**	0
Wynik	**5**	**4**	**3**	**4**	**3**

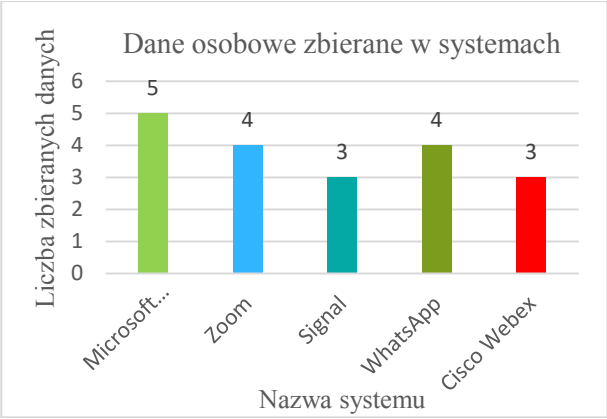
Po przeanalizowaniu powyższych danych widać wyraźnie, iż systemy otrzymały różne wyniki. Najniższy wynik (3 na 10) otrzymali Signal i Cisco Webex, co świadczy o tym, że systemy te zbierają najmniej danych osobowych i danych wrażliwych. Zoom i WhatsApp uzyskali wynik 4 na 10. Microsoft Teams natomiast wymaga od nowego użytkownika podania aż 5 danych osobowych.

Na Rysunku 1 pokazano dane osobowe zbierane przez systemy.

Na Rysunku 1 można zobaczyć, jak prezentują się wyniki. Wyraźnie widać dużą różnicę między systemami, które zbierają najmniej danych jak Signal, czy Cisco Webex, a Microsoft Teams, który zbiera ich najwięcej (Rysunek 1).

Warto zauważyć też, jakie to dane. Informacją, której wymagała każda z aplikacji było imię, natomiast dane niewymagane w żadnej aplikacji to adres, lokalizacja i pesel. Są to najważniejsze dane i nie powinny być zbierane przez aplikacje. Osoba niepożądana poznawszy lokalizację/adres użytkownika aplikacji mogłaby bez problemu go odnaleźć. Pesel jako indywidu-

alny identyfikator także nie powinien dostać się w niepowołane ręce.



Rysunek 1: Wizualizacja danych osobowych zbieranych w systemach (opracowanie własne).

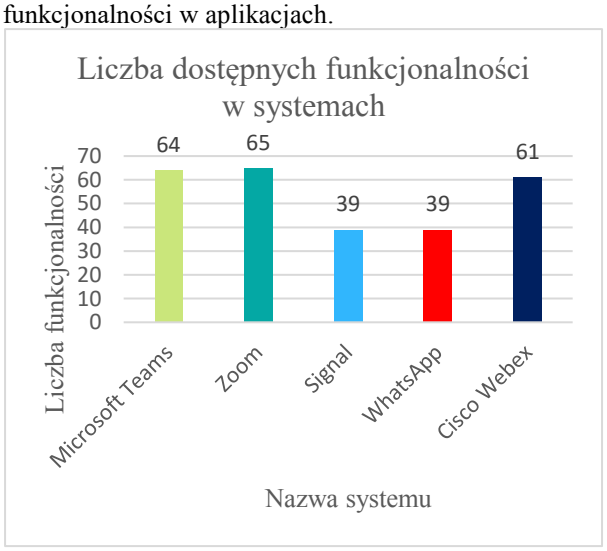
Signal, który zbiera najmniej danych wymaga podania imienia, listy kontaktów i numeru telefonu, a Cisco Webex – imienia, nazwiska i adresu e-mail, co sprawia, że łatwiej zidentyfikować tożsamość osoby w Cisco Webex. W powyższym badaniu najlepiej wypada właśnie Signal, mimo tej samej liczby zbieranych danych. Microsoft Teams zbiera takie dane jak: adres e-mail, imię, nazwisko, data urodzenia, kraj. Jest to bardzo duża ilość danych wymaganych przy instalacji systemu, które ułatwiają identyfikację użytkownika.

3.2. Analiza porównawcza ilości funkcjonalności

Celem tego badania było zebranie informacji na temat funkcjonalności występujących w badanych systemach. Aby poprawnie przeprowadzić badanie konieczna była instalacja wszystkich systemów i testowanie aplikacji poprzez ich użycie.

Każdy system zbadano pod kątem 68 funkcjonalności. Funkcjonalności jakie zostały zbadane to: Szyfrowanie połączeń, kompatybilność z Outlook, kompatybilność z MS Office, kompatybilność z kalendarzem, dodawanie emotikon, wysyłanie gifów, usuwanie wysłanych wiadomości, zapisywanie obrazów wysłanych na czacie, edytowanie wysłanych wiadomości, wysyłanie zdjęć/obrazów, przechowywanie historii wiadomości, wysyłanie plików multimedialnych, zmiana czcionki, dodawanie kontaktów, możliwość blokowania kontaktów, możliwość dodania własnego tła do wideokonferencji, zmiana tła już istniejącego w programie w wideokonferencji, przypomnienie o nadchodzącym spotkaniu, planowanie spotkań cyklicznych, funkcja asystenta planowania spotkania, wyciszenie/wyłączenie mikrofonu, wyłączenie kamerki, edytowanie spotkań, usuwanie spotkań, zapraszanie członków do kanału, edycja kanałów, usuwanie uczestnika z kanału, efekty specjalne, szyfrowany czat, znikanie wiadomości po odczytaniu, darmowa wersja, prowadzenie wideokonferencji, prowadzenie rozmów głosowych, wbudowany czat, możliwość tworzenia grup czatowych, tworzenie pokoi i grupowania osób podczas wideokonferencji, przydzielanie ról prowadzącego spotkanie, usuwanie uczestnika ze spotkania, sprawdzenie stanu obecności na spotkaniu, udostępnianie dźwięku z komputera, zmiana zdjęcia profilowego, ręczna zmiana statusu, wyświetlanie wszystkich plików wysłanych na czacie, możliwość wirtualnej poczekalni, możliwość blokowania spotkań, możliwość tworzenia kanałów, możliwość interakcji w rozmowach (podnoszenie ręki, brawo itd.), statystyki w wideokonferencji, możliwość przeprowadzenia ankiet, możliwość przeprowadzenia testu/egzaminu, limit liczby osób wideokonferencji (wersja darmowa), przesyłanie załączników zip, przesyłanie załączników docx, przesyłanie załączników csv, przesyłanie załączników pdf, przesyłanie załączników exe, możliwość nagrywania wideokonferencji, możliwość nagrywania rozmowy głosowej, możliwość robienia screenu czatu, możliwość reakcji na wiadomości, możliwość pobierania nagrania w formie video, możliwość dołączenia do grupy za pomocą linku, możliwość udostępniania spotkania za pomocą linku, możliwość zmiany języka, możliwość tablicy, dodawanie notatek udostępnionym ekranie lub tablicy, współdzielenie ekranu, możliwość przekazywania zdalnej kontroli, możliwość wyciszenia członka konferencji.

Niestety żaden z systemów nie posiada wszystkich oczekiwanych funkcjonalności, natomiast najwięcej z nich ma Zoom – 65 na 68. Następne miejsce zajmuje Microsoft Teams z 64 funkcjonalnościami. Cisco Webex posiada 61 funkcjonalności, a Signal i WhatsApp posiadają taką samą liczbę funkcjonalności – 39 na 68. Poniżej znajduje się Rysunek 2, który pokazuje liczbę funkcjonalności w aplikacjach.



Rysunek 2: Wizualizacja dostępnych funkcjonalności w systemach (opracowanie własne).

Powyższy Rysunek 2 pokazuje, że między Zoom, Microsoft Teams, Cisco Webex nie występuje duża różnica w liczbie dostępnych funkcjonalności. Oczywiście wyraźnie widać, że Zoom jest systemem przodującym, ale w porównaniu z Microsoft Teams jest to różnica zaledwie w jednej funkcjonalności. Znaczna różnica występuje za to między systemem Signal i WhatsApp. Porównując je z innymi systemami wyglądają dosyć niekorzystnie. Wykres precyzyjnie pokazuje dysproporcję w możliwościach funkcjonalnych.

3.3. Analiza porównawcza prędkości łącza i odbioru komunikatu głosowego w systemach

Badanie porównania prędkości łącza i odbioru w systemach było zadaniem bardzo pracochłonnym i wymagającym skupienia. Jego głównym celem było określenie prędkości transmisji danych i przepustowości łącza internetowego (w megabitach na sekundę), gdy systemy będą mieć opóźnienia oraz jakie będą to wartości czasowe.

Wyjątkowo ważna w badaniu była weryfikacja wszystkich systemów na różnych łączach i określenie, który system wypada najlepiej na tle innych przy konkretnej prędkości łącza, a który najgorzej oraz czy transmisja danych ma duży wpływ na działanie aplikacji.

Badanie polegało na zsynchronizowaniu zegarka autora pracy oraz odbiorcy komunikatu, następnie na sprawdzeniu rodzaju łącza i jego prędkości na stanowiskach badawczych oraz przeprowadzeniu wideorozmowy w konkretnym systemie. Osoba wypowiadająca ciąg słów obserwowała godzinę kiedy zaczęła mówić, zapamiętywała ją, a następnie zapisywała godzinę po skończonym pomiarze, a osoba odbierająca komunikat sprawdzała godzinę kiedy komunikat dotarł, a następnie po pomiarze zapisywała godzinę, kiedy usłyszała ciąg słów. Na podstawie tego można było łatwo obliczyć czas dotarcia komunikatu i przesunięcie w czasie. Na tym samym łączu badano w kolejności wszystkie systemy, a następnie zmieniano łącze na inne i ponawiano analizę. Badania wykonano na różnych prędkościach łączy począwszy od najmniejszego do największego i na każdym z systemów osobno.

Należy uwzględnić błąd pomiarowy wynoszący 0,5 s, który przypada na czas reakcji oczytania godziny przez odbiorcę komunikatu z zegarka. Następnie po odczytaniu godziny usłyszenia komunikatu odbiorca zapisywał czas. Zapisywanie czasu odbywało się już po komunikacie głosowym i sprawdzeniu godziny na zegarku, by zaoszczędzić błędów pomiarowych związanych z czasem zapisu. Autor biorąc pod uwagę możliwość błędu wykonał, aż 10 pomiarów i wyciągnął średnią z zaokrągleniem do 1 s. W połączeniu brały udział 2 osoby i było ono z włączonym video. Podczas badania sprawdzane były opóźnienia sieci, natomiast nie utrudniały wideokonferencji, ponieważ nie przekraczały 30 ms.

Stanowisko badawczego 1 to laptop ASUS TUF Gaming A15 z systemem operacyjnym Microsoft Windows 11 Home, pamięcią RAM 16 GB, dyskiem 512 GB, procesorem AMD Ryzen 5 i typem systemu 64 bitowym. Stanowisko badawcze 2 to laptop Lenovo ThinkPad T460p z systemem operacyjnym Microsoft Windows 10 Pro, pamięcią RAM 32 GB, dyskiem 500 GB, procesorem Intel Core i7 i typem systemu 64 bitowym.

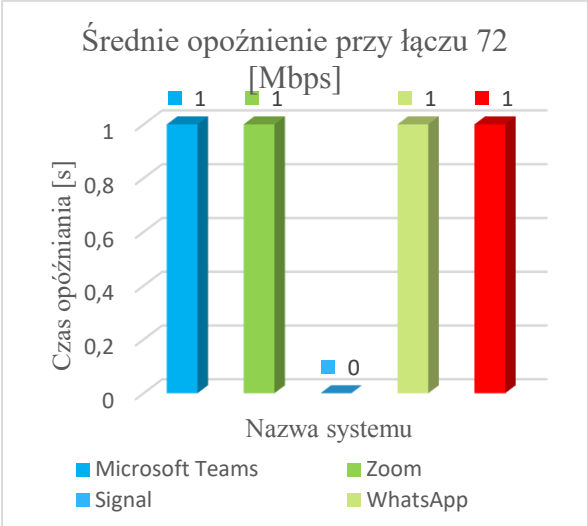
Podobne badania przeprowadzano na niektórych systemach, o których mowa w artykule *Performance Comparison of WhatsApp versus Skype on Smart Phones* [6], a dotyczą one Skype i WhatsApp. Wyniki w artykule wskazywały na bardzo podobną jakość połączeń VoIP. Przy znacznie wolniejszej transmisji danych wygrywał WhatsApp, który w badaniach w tej pracy nie wypadł najlepiej. Poniżej jest pokazana tabela 2. obliczonego już po badaniach średniego opóźnienia w systemach na łączu 72 Mbps.

Tabela 2: Średnie opóźnienie we wszystkich systemach na łączu 72 Mbps (opracowanie własne)

System	Prędkość łącza nadawcy (Mbps)	Prędkość łącza odbiorcy (Mbps)	Średnie opóźnienie (s)
Microsoft Teams	72	72	1
Zoom	72	72	1
Signal	72	72	0
WhatsApp	72	72	1
Cisco Webex	72	72	1

To jakie jest średnie opóźnienie w systemach najlepiej widać na wykresie zamieszczonym poniżej (Rysunek 3). Na Rysunku 3 łatwo widać, że takie samo opóźnienie ma Cisco Webex, WhatsApp, Microsoft Teams i Zoom wynoszące 1 s.

Ze wszystkich systemów najmniejszą wartość opóźnienia wykazuje Signal. Opóźnienie 0 s na łączu 0,72 Mbps jest zaskakujące. Można stwierdzić, że rozmowa na tym łączu dla Signal odbywa się w czasie rzeczywistym.



Rysunek 3: Wizualizacja średniego opóźnienia w systemach na łączu 72 Mbps (opracowanie własne).

Następnym pomiarem jest łącze o większej prędkośći, równe 86 przy paśmie łącza 2,4 Ghz. Wyniki badania są dostępne w Tabeli 3. Pierwszym krokiem analizy było obliczenie średniego opóźnienia w czasie między nadaniem komunikatu głosowego, a jego odbiorem.

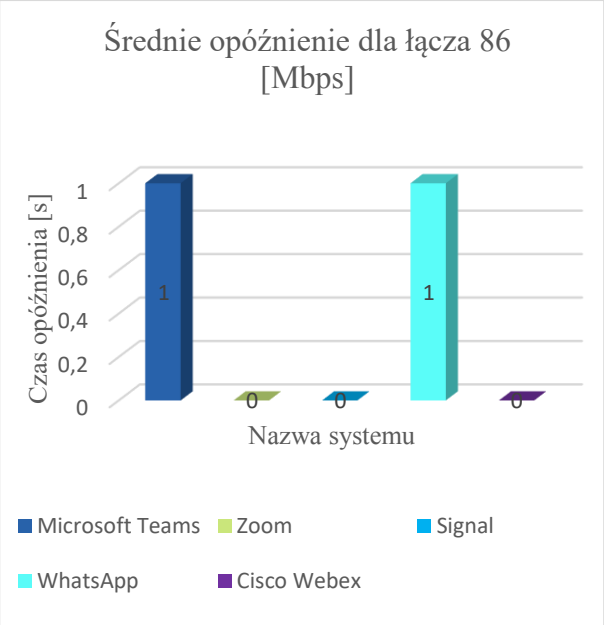
Na Rysunku 4 przedstawione jest średnie opóźnienie wszystkich badanych systemów. Widać na nim wyraźnie, że największe opóźnienia na lepszym łączu (od

wcześniejszego 72 [Mbps]) ma WhatsApp i Microsoft Teams, których opóźnienie wynosi 1s.

Zdecydowanie najlepszy wynik posiada Signal, Zoom i Cisco Webex, bo 0 s opóźnienia. Cisco Webex jest to pewnego rodzaju zaskoczeniem, ponieważ na słabszym łączu generował opóźnienia, a łączu 86 Mbps już nie.

Tabela 3: Średnie opóźnienie we wszystkich systemach na łączu 86 Mbps (opracowanie własne)

System	Prędkość łącza nadawcy (Mbps)	Prędkość łącza odbiorcy (Mbps)	Średnie opóźnienie (s)
Microsoft Teams	86	86	1
Zoom	86	86	0
Signal	86	86	0
WhatsApp	86	86	1
Cisco Webex	86	86	0



Rysunek 4: Wizualizacja średniego opóźnienia w systemach na łączu 86 Mbps (opracowanie własne).

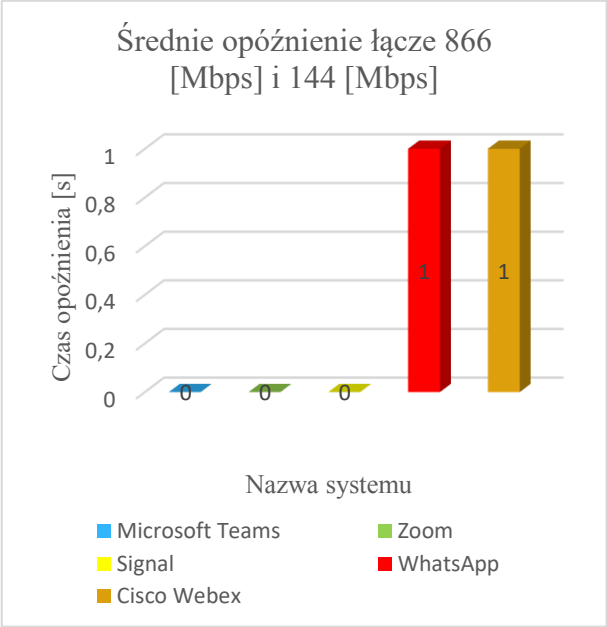
Kolejnym badanym łączem było łącze zróżnicowane – nadawca dysponował prędkością 866 Mbps i pasmem sieci 5 Ghz, a odbiorca prędkość transmisji 144 Mbps i pasmem sieci 2,4 Ghz. Badanie przeprowadzono celowo na takich parametrach w celu sprawdzenia, czy pasmo ma wpływ na funkcjonowanie systemów i czy systemy działają przy takich parametrach. Uzyskane wyniki są wyjątkowo ciekawe. Poniżej w Tabeli 4. pokazane są wyniki: z 10 pomiarów obliczono średnią opóźnienia systemów.

Chcąc zwizualizować wyniki przygotowano wykres pokazujący systemy i opóźnienia (Rysunek 5). Wyraźnie widać, że podobnie jak w poprzednich pomiarach WhatsApp uzyskuje jedno z większych opóźnień z wynikiem 2. Mimo szybszego łącza opóźnienie jest większe niż przy łączu wolniejszym, a jest to spowodowane zróżnicowaniem pasm sieciowych. Niektóre systemy gorzej sobie radzą z takim zróżnicowaniem.

Tabela 4: Średnie opóźnienie we wszystkich systemach na łączu 866 Mbps i 144 Mbps (opracowanie własne)

System	Prędkość łącza nadawcy (Mbps)	Prędkość łącza odbiorcy (Mbps)	Średnie opóźnienie (s)
Microsoft Teams	866	144	1
Zoom	866	144	0
Signal	866	144	0
WhatsApp	866	144	2
Cisco Webex	866	144	1

Nieco niższe opóźnienie uzyskał Cisco Webex (1 s), Microsoft Teams (1 s) i Zoom (1 s). Można stwierdzić, że na tej transmisji danych najlepiej wypadł Zoom z najmniejszym przesunięciem i Signal (0 s).



Rysunek 5: Wizualizacja średniego opóźnienia w systemach na łączu 866 Mbps i 144 Mbps (opracowanie własne).

3.4. Analiza użyteczności i popularności systemów wśród użytkowników

Badanie analizy użyteczności i popularności systemów wśród użytkowników było elektronicznym badaniem ankietowym. Ankieta miała na celu ocenę aplikacji przez użytkowników pod kątem jakości, użyteczności i popularności. Ankieta została przeprowadzona w formie elektronicznej za pomocą Google Forms.

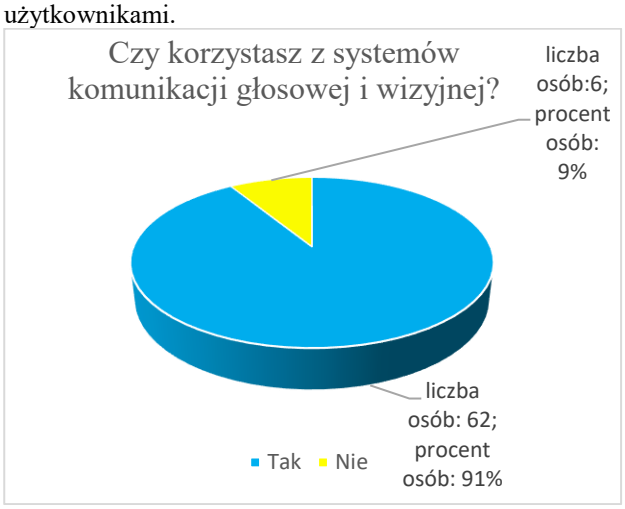
Grupa respondentów to osoby w przedziałach wiekowych 5-13 lat, 14-18 lat, 19-26 lat, 27-45 lat, 46-55 lat, 56-80 lat. Byli to studenci, uczniowie szkół podstawowych, uczniowie szkół ponadgimnazjalnych, osoby pracujące, osoby bezrobotne i emeryci. Wśród osób pracujących były to osoby pracujące w branżach: finanse, IT, elektronika, elektryka, administracja, medycyna, handel, transport, szkolnictwo, techniczne inne, górnictwo, chemiczna, metalowa, branża odzieżowa, ekonomiczna, bankowość, produkcyjna, spedycja, budownictwo.

Czynniki wpływające na wykorzystanie systemów to: zapotrzebowanie na systemy do nauki zdalnej w

szkołach wywołane pandemią, zapotrzebowanie na pracę zdalną, co powoduje korzystanie z aplikacji, potrzeba rozmowy z osobami przebywającymi daleko, potrzeba silnego kontaktu i uczestnictwa w życiu towarzyskim, a takie możliwości dają systemy przez możliwość ciągłej rozmowy z bliskimi.

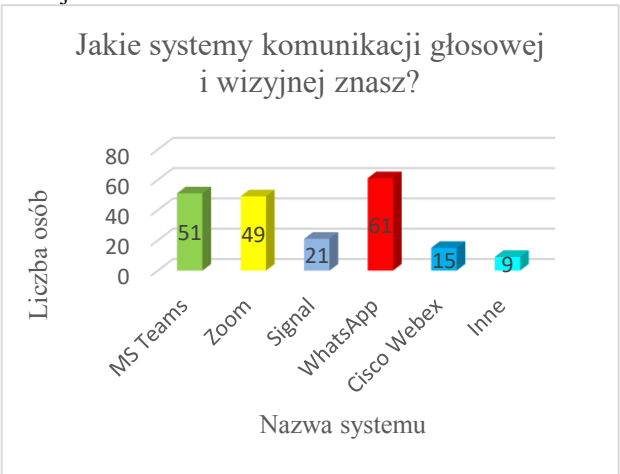
Rysunek 6 pokazuje korzystanie z systemów komunikacji głosowej i wizyjnej. Zapytano ankietowanych, czy korzystają z systemów komunikacji głosowej i wizyjnej i aż 91% ankietowanych opowiedziało twierdząco. Pokazuje to jak dużą popularnością cieszą się te systemy. Tylko 9% ankietowanych ich nie używa. Poniżej znajduje się wykres przedstawiający odpowiedzi na wykresie kołowym (Rysunek 6).

Rysunek 7 pokazuje z kolei, jaka jest znajomość tych systemów wśród ankietowanych. Postawiono pytanie: „Jakie systemy komunikacji głosowej i wizyjnej znasz?", przy którym możliwe było udzielenie więcej niż jednej odpowiedzi (Rysunek 7). Łatwo zauważyć, że najbardziej znany jest WhatsApp – uzyskał 61 głosów, kolejno Microsoft Teams z wynikiem 51 osób i Zoom – 49. Te trzy systemy są najpopularniejsze. Mniej znane systemy to: Signal, który zdobył 21 głosów i Cisco Webex z wynikiem 15 głosów. Ankietowani potwierdzili znajomość również innych systemów, których są użytkownikami.

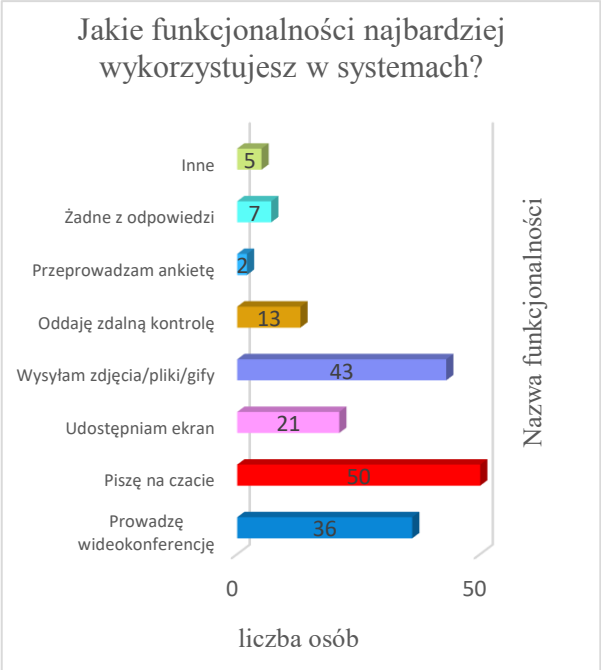


Rysunek 6: Korzystanie z systemów komunikacji głosowej i wizyjnej (opracowanie własne).

Rysunek 8 ukazuje wykorzystanie funkcjonalności omawianych systemów przez ankietowanych. Dla pytania: „Z jakich funkcjonalności systemów najczęściej korzystasz?" przygotowano poniższy wykres (Rysunek 8). To także pytanie wielokrotnego wyboru. Jedną z najpopularniejszych funkcjonalności okazało się pisanie na czacie, co zadeklarowało 50 osób. Większość osób komunikuje się ze sobą dzięki tym aplikacjom, stąd określa się je jako komunikatory. Bardzo dużo głosów zdobyła także funkcjonalność wysyłania zdjęć/gifów/plików z wynikiem 43 głosów. Prowadzenie konferencji wskazało 36 osób. Są to trzy najczęściej wykorzystywane funkcjonalności. Te cieszące się mniejszym zainteresowaniem to: udostępnianie ekranu (21 głosów), oddawanie zdalnej kontroli (13 głosów), przeprowadzanie ankiety (2 głosy). Na wykresie pokazano, że 5 osób korzysta z innych funkcjonalności, a 7 osób odpowiedziało, że nie korzysta z powyższych funkcjonalności.



Rysunek 7: Rozkład znajomości systemów komunikacji głosowej i wizyjnej (opracowanie własne).



Rysunek 8: Rozkład wykorzystania funkcjonalności wśród ankietowanych (opracowanie własne).

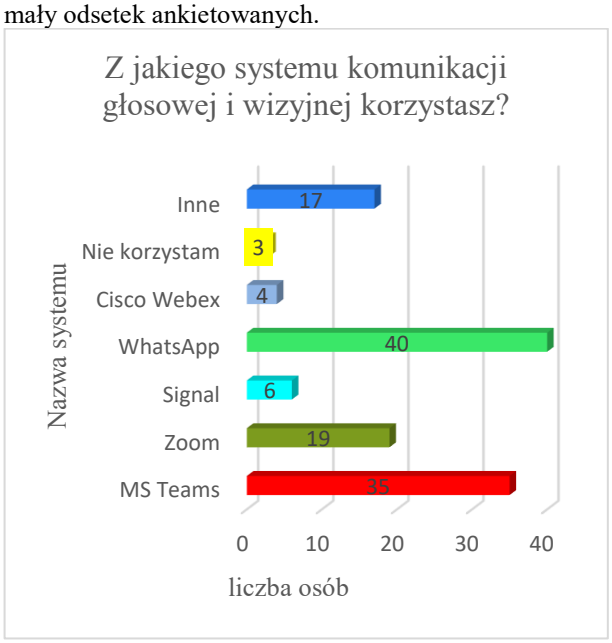
Przy pytaniu „Z jakiego systemu komunikacji głosowej i wizyjnej korzystasz?" ankietowani dostali możliwość wielokrotnego wyboru i zaznaczenia kilku systemów. Najczęściej wybierany system to WhatsApp (aż 40 osób), następnie Microsoft Teams (35 osób) i Zoom (19 głosów). Widać wyraźną dominację tych dwóch systemów – MS Teams i WhatsApp.

Przewaga Microsoft Teams bierze się zapewne z faktu, iż sprawdza się on najlepiej spośród innych systemów w szkołach. Napisano na ten temat artykuł – *Integrated Systems in Emergency Distance Education: The Microsoft Teams* [7], w którym autorzy opisują go jako system przeznaczony do wykorzystania przez „wirtualną" klasę. Autorzy pokazują m.in. możliwości zbie-

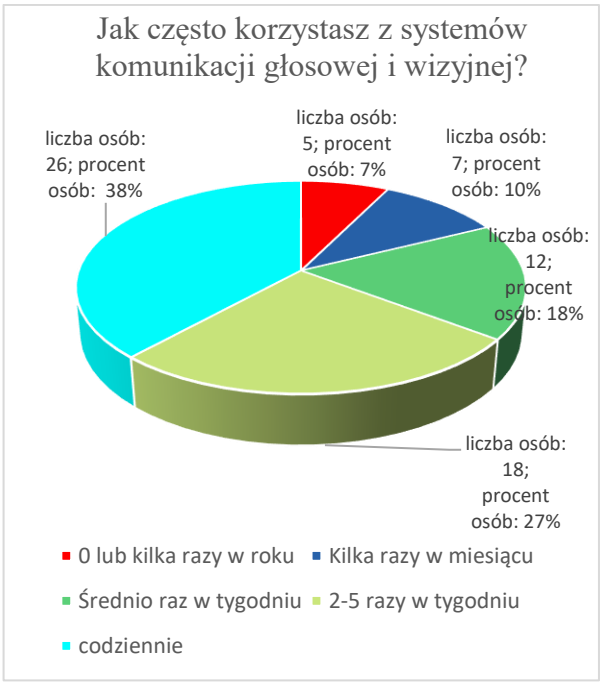
rania prac domowych, wypełniania testów, czy udostępniania plików.

Na Rysunku 9 widać, że Cisco Webex zdobył zaledwie 4 głosy, a Signal – 6. Przygotowano wykres odpowiedzi ankietowanych (Rysunek 9).

Na Rysunku 10 pokazano, jak często ankietowani korzystają z systemów komunikacji głosowej i wizyjnej. Należy podkreślić, że aż 38% ankietowanych korzysta z systemów komunikacji codziennie, następnie 27% korzysta z nich 2-5 razy w tygodniu. Część osób korzysta z aplikacji raz w tygodniu lub kilka razy w miesiącu. Niecałe 8% ankietowanych odpowiedziało, że nie korzysta z aplikacji wcale lub tylko raz w roku, co stanowi mały odsetek ankietowanych.



Rysunek 9: Rozkład systemów z jakich korzystają ankietowani (opracowanie własne).



Rysunek 10: Częstotliwość korzystania z systemów przez ankietowanych (opracowanie własne).

4. Wnioski

Podsumowując analizę funkcjonalności systemów komunikacji głosowej i wizyjnej, będących tematem tej pracy – wykonano serię specjalistycznych badań, jak: porównanie wymaganych danych osobowych w systemach, porównanie funkcjonalności systemów, analizę prędkości łącza i odbioru komunikatu głosowego, analizę badanych aplikacji pod względem jakości, użyteczności i popularności wśród użytkowników.

Pierwsza analiza, czyli porównanie wymaganych przez aplikacje danych osobowych pokazała, że najwięcej danych osobowych użytkowników na etapie instalacji wymaga Microsoft Teams. Na drugim miejscu uplasowały się ex aequo Zoom i WhatsApp. Nie świadczy to jednak o najwyższym poziomie bezpieczeństwa, ponieważ na jego poziom ma wpływ więcej czynników. Najmniej danych osobowych zbierają Cisco Webex Meetings i Signal.

Analiza dotycząca funkcjonalności systemów była pracochłonnym zadaniem i pokazała wiele ciekawych kwestii. By zbadać systemy niezbędne okazały się osoby trzecie. Poproszono je o wykonywanie pewnych czynności z wykorzystaniem systemów. Zbadano aż 68 funkcjonalności w każdym z pięciu badanych systemów komunikacji głosowej i wizyjnej.

Zdecydowanym liderem wśród aplikacji posiadających najszerszy wybór funkcjonalności okazał się Zoom, dysponujący aż 96% funkcjonalności. Miejsce drugie zajął Microsoft Teams, którego możliwości funkcjonalne są również duże (94% funkcjonalności). Wielkim zaskoczeniem okazał się mało znany użytkownikom Cisco Webex Meetings, ponieważ zajął trzecie miejsce wśród systemów z wynikiem 90% funkcjonalności. Zaskakującą kwestią jest to, że system posiada wiele możliwości dedykowanych użytkownikowi, a wciąż jest mało znany. Pokazało to badanie ankietowe przeprowadzone w dalszej części pracy. Najmniej funkcjonalności wykazały dwie aplikacje: Signal oraz WhatsApp, ponieważ tylko 57% badanych funkcjonalności. Nie jest to dobry wynik w porównaniu z innymi aplikacjami.

Badanie prędkości łącza i odbioru komunikatu głosowego wymagało skupienia zarówno od nadawcy (autora artykułu), który wysyłał komunikat głosowy, jak i od odbiorcy odbierającego komunikat. Polegało to na zmierzeniu czasu dotarcia komunikatu, a następnie opóźnienia czasowego na różniej transmisji danych w systemach.

Przy łączu równym 74 Mbps zarówno u nadawcy i u odbiorcy takie samo opóźnienie wygenerował Cisco Webex, WhatsApp, Microsoft Teams i Zoom wynoszące 1 s. Nie są to znaczące opóźnienia i nie utrudniają korzystania użytkownika z aplikacji. Pomiar zdecydowanie wygrał Signal, którego opóźnienie równało się 0 s. Opóźnienie na tej transmisji danych dla Singala jest niezauważalne.

Nieco większa transmisja danych równa 86 Mbps u nadawcy i odbiorcy przyniosła nowy ranking. Największe opóźnienie wykazał WhatsApp i Microsoft Teams, wynoszące 1s. Co zaskakujące pozostałe systemy: Cisco

Webex, Signal i Zoom, nie generowały opóźnień w odbiorze. Zauważalna różnica występuje dla systemu Cisco Webex, który na słabszym łączu 72 Mbps generował opóźnienia, natomiast na łączu 86 Mbps już nie.

Zdecydowanie szybsza transmisja danych wynosząca 866 Mbps u nadawcy i 144 Mbps odbiorcy wykazała największe opóźnienie dla WhatsApp –2 s. Najmniejsze opóźnienie przypadło z kolei systemowi Zoom i Signal o wartości 0 s. Nie zawsze to prędkość transmisji danych ma największe znaczenie przy wartości opóźnienia. Jak wykazały badania – niektóre systemy osiągały większe opóźnienie na szybszym łączu. W przypadku tego pomiaru łącze 866 Mbps było na paśmie sieci wynoszącym 5 Ghz, a łącze 144 Mbps na paśmie 2,4 Ghz. Mogło to przynieść różnice w opóźnieniach między systemami. W kwestii opóźnień należy uwzględniać także m.in. parametry komputera, czy zużycie CPU. Istnieje wiele innych czynników, które wpływają na opóźnienie komunikatu w systemie.

Warto zauważyć, że przodownikiem w opóźnieniach komunikatu głosowego okazał się WhatsApp. Najlepiej pracującym systemem, na rożnej transmisji danych jest zdecydowanie Signal.

Ostatnie elektroniczne badanie ankietowe pokazało, że systemy komunikacji głosowej są popularne. Ich popularność wzrosła, ponieważ nastąpiło ogromne zapotrzebowanie na systemy do nauki zdalnej w szkołach, zapotrzebowanie na pracę zdalną, co powoduje korzystanie z aplikacji, potrzeba rozmowy z osobami przebywającymi daleko, potrzeba kontaktu i uczestnictwa w życiu towarzyskim, a takie możliwości dają systemy przez możliwość rozmowy i wideokonferencji.
Korzystanie z nich zadeklarowało aż 91% osób ankietowanych. Jako najczęściej wykorzystywane systemy wskazywano WhatsApp oraz Microsoft Teams. Najmniej znany okazał się Cisco Webex Meeting. Najpopularniejsze systemy z kolei to WhatsApp, Microsoft Teams i Zoom. Funkcjonalności najbardziej wykorzystywane w aplikacjach to pisanie na czacie, wysyłanie zdjęć/gifów/plików oraz prowadzenie wideokonferencji. Najmniej używaną funkcjonalnością jest przeprowadzanie ankiet. Aż 38% ankietowanych korzysta z systemów codziennie, a 27% kilka razy w tygodniu. Daje to dużą grupę osób i wiele godzin spędzonych z aplikacją w skali dnia, tygodnia, czy roku. 91% ankietowanych podkreśliło, iż kwestie bezpieczeństwa są dla nich bardzo ważne. Nie ma w tym nic zaskakującego. Przecież za pośrednictwem aplikacji często wysyła się prywatne zdjęcia, pliki; prowadzi się rozmowy osobiste, wymienia się poufne informacje. Badania pokazały też, że zaledwie 3% osób z ankietowanych nie korzysta z aplikacji komunikacji głosowej i wizyjnej. Jest to grupa respondentów w wieku od 56 do 80 lat i są to emeryci.
Powyższe badania poruszyły wiele kwestii związanych z systemami komunikacji głosowej i wizyjnej począwszy od informacji o danych osobowych użytkowników, przez funkcjonowanie systemów na różnej transmisji danych i rodzaje ich funkcjonalności, w oparciu o doświadczenia ankietowanych, a zarazem użytkowników aplikacji badanych na potrzeby powyższego artykułu.

Literatura

[1] Systemy do komunikacji, https://pl.wikipedia.org/wiki/Komunikator_internetowy, [12.04.2022]

[2] D. Bouhnik, M. Deshen, WhatsApp goes to school: Mobile instant messaging between teachers and students, Journal of Information Technology Education 13 (2014) 217-231.

[3] S. Ravinder, S. Awasthi, Updated Comparative Analysis on Video Conferencing Platforms- Zoom, Google Meet, Microsoft Teams, WebEx Teams and GoToMeetings, EasyChair Preprint 4026 (2020) 1-6.

[4] A. Correia, F. Xu, F. Chenxi, Evaluating videoconferencing systems for the quality of the educational experience, Distance Education 41 (2020) 429-452.

[5] R. Rahim, A. Siahaan, S. Lubis, Insecure Whatsapp Chat History, Data Storage and Proposed Security, International Journal of Pure and Applied Mathematics 119 (2018) 2481-2486.

[6] N. Patel, S. Patel, W. Tan, Performance Comparison of WhatsApp versus Skype on Smart Phones, International Telecommunication Networks and Applications Conference (2018) 1-3.

[7] S. Cenkaya, G. Durak, Integrated Systems in Emergency Distance Education: The Microsoft Teams. Necatibey Faculty of Education Electronic, Journal of Science & Mathematics Education 14 (2020) 889-920.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 218–223
Received: 20 May 2022
Accepted: 12 June 2022

Comparative analysis of the Cycles and Eevee graphics engines on the example of rendering 3D models of archaeological artifacts

Analiza porównawcza silników graficznych Cycles oraz Eevee na przykładzie renderowania modeli 3D artefaktów archeologicznych

Sebastian Dudek* , Krzysztof Dziedzic

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

Abstract

The purpose of the article is to conduct a comparative analysis of two graphics engines available in the Blender application. These are the Eevee and Cycles engines. The analysis will concern such elements as: efficiency, the ability to adjust the scene and the speed of image rendering by both engines. A test stand in the form of a desktop computer with a newly installed operating system has been prepared for the research.
Keywords: Blender; performance; graphic engine; comparison

Streszczenie

Celem artykułu jest przeprowadzenie analizy porównawczej dwóch silników graficznych dostępnych w aplikacji Blender. Są to silniki Eevee oraz Cycles. Analiza dotyczyć będzie takich elementów jak: wydajność, możliwość dostosowania sceny oraz szybkość renderowania obrazu przez oba silniki. Do przeprowadzenia badań przygotowane zostało stanowisko badawcze w postaci komputera stacjonarnego z nowo zainstalowanym systemem operacyjnym.
Słowa kluczowe: Blender; wydajność; silnik graficzny; porównanie

*Corresponding author
Email address: sebastian.dudek@pollub.edu.com (S. Dudek), k.diedzic@pollub.pl (K. Dziedzic)

1. Wstęp

Badania wydajności silników graficzne są obecne w literaturze naukowej. Jednymi z częściej porównywanych silników graficznych są Unreal Engine oraz Unity. Paweł Skop w swoim artykule z 2017 roku porównuje oba te silniki pod kątem wydajności na wybranych platformach. Na potrzeby artykułu dla obu silników stworzone zostały gry w wirtualnej rzeczywistości. Analiza wyników wykazała, że dostępne w momencie pracy nad artykułem ówczesne wersje tych popularnych silników różnią się od siebie ilością uzyskiwanych klatek obrazu na sekundę. Silnik Unity uzyskał lepsze rezultaty niż silnik Unreal Engine. Wyniki jakie uzyskano były odwrotne do tezy, w której autor zakładał, że to silnik Unity będzie charakteryzował się mniejszą wydajnością [1]. Przeprowadzone zostało również porównanie wydajności silnika Unity do Cry Engine, który również jest używany najczęściej w grach komputerowych [2]. Badania takie przeprowadził Huber Żukowski w 2019 roku. W swoim artykule porównywał on ilości uzyskanych klatek obrazu na sekundę, obciążenie procesora, oraz zajętość RAM. Również w tym przypadku silnik Unity okazał się rozwiązaniem wydajniejszym.

W erze, w której zdecydowana większość przemysłu filmowego wykorzystuje efekty komputerowe, projektowane gry stają się coraz bardziej rozbudowane i dopracowane, zwiększają się wymagania względem aplikacji wykorzystywanych do opracowania wszelkich materiałów graficznych [3]. Aby przenieść wizję artystów na ekrany potrzebne są odpowiednie narzędzia, które wspomagają i ułatwiają im pracę w szerokim zakresie zastosowań grafiki komputerowej. Dostępne dziś narzędzia i rozwiązania dają szansę wejścia w świat grafiki komputerowej nie tylko największym firmom, ale też każdemu użytkownikowi, który korzysta z komputera w domu. Dostępne są bezpłatne rozwiązania, które dają nieograniczone możliwości twórcze. Zainteresowane osoby nie muszą więc inwestować dodatkowych środków, aby sprawdzić siebie w tej dziedzinie lub możliwości oprogramowania. Tak szeroki dostęp może zachęcić zainteresowanych do zrozumienia i poszerzenia swoich umiejętności w tej dziedzinie.

Możliwości graficzne w grach wideo z roku na rok wzrastają ze względu na rosnącą moc obliczeniową komputerów domowych. Twórcy już od jakiegoś czasu wdrażają w swoich silnikach, grach ray tracing. Ta technika pozwala zwiększyć realizm obrazu dzięki ulepszonym odbiciom na powierzchniach metalicznych lub połyskujących, cieniowaniu i okluzji otoczenia. Jeszcze do niedawna ray tracing był zarezerwowany głównie do tworzenia fotorealistycznych wizualizacji wnętrz, reklam lub w przemyśle filmowym. Wynikało to z braku wystarczającej mocy obliczeniowej, aby wygenerować odpowiednią liczbę klatek obrazu na sekundę do utrzymania płynności obrazu na pożądanym poziomie. Wykorzystując nowoczesny sprzęt, fotogrametrię lub skanery 3D możemy również skanować i przenosić obiekty niemal jeden do jednego do programów graficznych [4]. Obiekt taki możemy skanować w różnych rozdzielczościach, co zwiększa możliwość dostosowania go do założonych wymagań.

Historia grafiki komputerowej rozpoczyna się w dwudziestym wieku, a dokładnie są to lata sześćdziesiąte [5, 6]. Początkowo grafika komputerowa była zarezerwowana głównie dla wojska lub dużych firm z odpowiednio wysokim budżetem. Armia wykorzystywała tę dziedzinę do symulacji otoczenia w symulatorach lotów. Pojęcie grafiki komputerowej pojawiło się dzięki panu Williamowi Fetterowi. Był on artystą, który pracował w fabryce samolotów. Pierwszym narzędziem pozwalającym na pracę z bryłami oraz wykorzystanie linii był Sketchpad. Powstanie tej aplikacji można uznać za przełom w grafice komputerowej. Następne lata to czas rozwoju technologii i zwiększania się możliwości sprzętu. W latach siedemdziesiątych stworzony został pierwszy model ludzkiej twarzy. Był on również animowany. Jednym z pierwszych programów do modelowania w trzech wymiarach był 3D Core Graphic System. Grafika komputerowa coraz śmielej wkraczała w świat reklamy, plakatów czy ulotkek. Jedną z większych zmian było stworzenie przez firmę Apple komputera o nazwie Macintosh II. Był to komputer osobisty do użytku domowego. Posiadał on również rozbudowany program graficzny, którym był Adobe Photoshop. Aplikacja ta pozwalała na tworzenie grafik z wykorzystaniem warstw. Dzięki temu istniała i istnieje do dzisiaj możliwość niezależnej edycji poszczególnych sekcji tworzonego obrazu. W kolejnych latach grafika komputerowa stawała się coraz większą gałęzią komputerowego świata. Lata osiemdziesiąte i dziewięćdziesiąte to rozwój grafiki trójwymiarowej. W tym czasie powstaje aplikacja Blender. Pierwsza wersja, czyli Blender 1.0 została opublikowana w styczniu 1995 roku. Projekt jest rozwijany do dzisiaj. Obecna najnowsza wersja produkcyjna dostępna na stronie producenta to Blender 3.1.2.

2. Aplikacja Blender

Blender, widoczny na rysunku 1 jest pakietem narzędzi, który rozwijany jest obecnie przez Blender Foundation [7]. Oprogramowanie jest darmowe oraz udostępnione na licencji GNU GPL. Daje możliwość tworzenia animacji obiektów trójwymiarowych i dwuwymiarowych oraz renderingu przygotowanych scen. W aplikacji Blender zaimplementowano wiele przydatnych narzędzi. Pomagają i ułatwiają przeniesienie wizji artysty lub projektanta na ekran komputera [8]. Z ich pomocą obiekty trójwymiarowe mogą być poddane licznym modyfikacjom.

Rysunek 1: Interfejs aplikacji Blender 3.0.

Silnik Cycles od swojego debiutu na rynku przeszedł wiele zmian [9]. Po 10 latach istnienia wraz z Blenderem w wersji 3.0 pojawiła się nowa odsłona tego silnika o nazwie Cycles-X. Wprowadzonych zostało wiele zmian, które na celu miały zastąpienie starych i niezoptymalizowanych rozwiązań oraz algorytmów tego silnika graficznego. Pozwoliło to znacząco zwiększyć wydajność Cycles. Zmiany przyniosły nawet o 50% szybsze czasy renderowania w stosunku do możliwości jakie oferowała poprzednia wersja silnika. Dodana została obsługa interfejsu API NVIDIA OptiX. Kiedy na rynku pojawiły się karty graficzne z rodziny Geforce Turing, pokazano też pierwsze modele z dedykowanymi jednostkami RT. Jednostki te mogą wspomagać i skracać czas w obliczeniach związanych ze śledzeniem promieni. Dającym duże możliwości poprawy obrazu jest wbudowany w aplikację Blender denoiser. Jest to narzędzie służące do odszumiania obrazu. Podczas renderowania, najczęściej w ciemnych miejscach powstaje szum. Może być to spowodowane niskim próbkowaniem w ustawieniach naszego silnika renderującego. Denoiser ma na celu usunięcie powstałego szumu za pomocą technik przetwarzania końcowego, które oparte są na uczeniu maszynowym. Ma on jednak swoje wady. Jeżeli próbkowanie będzie za niskie, a informacji na obrazie zbyt mało, efekt końcowy będzie lekko niewyraźny, rozmazany.

Silnik Eevee został wprowadzony w jednej z aktualizacji do aplikacji Blender, a dokładnie do wersji 2.80 [10]. Przy okazji tej zmiany odświeżony został również interfejs omawianego oprogramowania. Silnik ten działa na podobnej zasadzie co inne silniki renderujące w czasie rzeczywistym. Rozwiązania takie świetnie sprawdzają się w grach komputerowych. Przykładami konkurencyjnych silników mogą być Unreal Engine oraz Unity. Technologia i sposób generowania obrazu sprawia, że silnik ten w porównaniu do Cycles będzie generował gorszej jakości odbicia, cieniowanie czy efekty wolumetryczne. Eevee nie wykorzystuje ray tracingu. Jego przewagą jest znacznie, krótszy czas renderowania obrazów. Dużą zaletą jest też to, że obraz w podglądzie jest identyczny jak końcowy rezultat po wyrenderowaniu. Daje to możliwość wprowadzania poprawek w modelach, teksturach czy oświetleniu na bieżąco bez konieczności częstego renderwania sceny, aby zobaczyć efekt końcowy. Dzięki temu zaoszczędzony czas można poświęcić na inne aspekty projektu. Silnik o takiej specyfice jest przydatny i wygodniejszy w momencie tworzenia animacji. Po raz kolejny oferowana szybkość pozwala na lepszą kontrolę nad animacjami. Nie jest konieczne renderowanie czasami wielu setek czy nawet tysięcy klatek, aby móc wychwycić niedoskonałości. Eevee wspiera technikę Physically Based Rendering (PBR).

3. Opis badania

3.1. Obiekty badań

Do przeprowadzenia badań zeskanowane zostały rzeczywiste obiekty, artefakty archeologiczne. Przygotowane zostały różne ich wersje o różnej ilości trójkątów

oraz rozdzielczości tekstur. Za pomocą skanera na światło strukturalne uzyskana została chmura punktów, która wykorzystana została do przygotowania modeli 3D. Użytym skanerem był Artec EVA (Rysunek 2)[11].

Rysunek 2: Skaner Artec EVA.

Testy wykonano na trzech obiektach archeologicznych. Każdy z nich posiadał trzy warianty o różnym zagęszczeniu siatki:

- 500 wierzchołków,
- 3000 wierzchołków,
- 6000 wierzchołków.

Rysunek 3: Zeskanowany model, Model_01, 6000 wierzchołków.

Dla zeskanowanych obiektów przygotowane zostały trzy warianty tekstur o różnej rozdzielczości:

- 512 x 512 pikseli,
- 1024 x 1024 pikseli,
- 2048 x 2048 pikseli.

Zeskanowane modele przedstawione na Rysunkach 3, 4 oraz 5 zostały umieszczone na prostej w budowie scenie. Aby wykluczyć wpływ obiektów otaczających badane warianty modeli, środowisko jest maksymalnie uproszczone. Gotowa scena składa się z zeskanowanego obiektu w wybranym wariancie, podłogi w postaci obiektu plane, dostępnego domyślnie w aplikacji Blender oraz mapy HDRI o rozdzielczości 2160p.

Rysunek 4: Zeskanowany model, Model_02, 500 wierzchołków.

Rysunek 5: Zeskanowany model, Model_03, 3000 wierzchołków.

3.2. Metoda badań

Badanie wydajności polegało na sprawdzeniu czasów renderowania, danej sceny wraz z obiektami o różnej ilości wierzchołków. Dla każdego modelu o danej jakości wykonanych zostało 5 prób. Uzyskane czasy zapisywane były na podstawie czasomierza wbudowanego w aplikację. Każda próba renderowania była wykonana w ten sam sposób. Mając obiekt i scenę w podglądzie aplikacji Blender, używany był klawisz F12 odpowiedzialny za rozpoczęcie procesu renderowania. Po zakończonej operacji powracano ponownie do widoku podglądu i renderowanie rozpoczynano od nowa w

analogiczny sposób. Badanie różnic w generowanym obrazie polegało na porównywaniu dwóch wyrenderowanych przy użyciu silników Cycles oraz Eevee scen. Analizie podlegało cieniowanie, odbicia oraz oświetlenie.

3.3. Stanowisko badawcze

Badania przeprowadzone zostały na komputerze o specyfikacji technicznej podanej w tabeli (Tabela 1).

Tabela 1: Specyfikacja techniczna stanowiska badawczego

Podzespół	Model/Parametry
Procesor	AMD Ryzen 3600 6/12, 4,2GHz
Karta graficzna	Palit RTX 3060TI 8GB
Dysk SSD 1	Adata 8200SX Pro
Dysk SSD 2	Samsung SSD 870 QVO 1TB
Płyta główna	Asus ROG Strix B450-F Gaming
RAM	16GB (2x8GB) DDR4 3200 MHz
System operacyjny	MS Windows 11 Pro 64bit

4. Wyniki badań

Wyniki testów przeprowadzonych zgodnie ze scenariuszem badawczym przedstawiają tabele. Tabele z wynikami przedstawiają czasy renderowania dla silnika Cycles (Tabela 2) oraz Eevee (Tabela 3).

Tabela 2: Czasy renderowania dla silnika Cycles.

Model_01		
500 verts	3000 verts	6000 verts
13.25[s]	13,49[s]	13,64[s]
13,33[s]	13,30[s]	13,50[s]
13,25[s]	13,42[s]	13,49[s]
13,18[s]	13,40[s]	13,42[s]
13,13[s]	13,31[s]	13,46[s]
Model_02		
500 verts	3000 verts	6000 verts
13,27[s]	13,40[s]	13,56[s]
13,04[s]	13,43[s]	13,32[s]
12,86[s]	13,28[s]	13,45[s]
13,02[s]	13,23[s]	13,41[s]
13,09[s]	13,21[s]	13,33[s]
Model_03		
500 verts	3000 verts	6000 verts
12,30[s]	12,38[s]	12,40[s]
12,18[s]	12,29[s]	12,43[s]
12,10[s]	12,21[s]	12,25[s]
12,06[s]	12,09[s]	12,45[s]
12,08[s]	12,26[s]	12,32[s]

Tabela 3: Czasy renderowania dla silnika Eevee.

Model_01		
500 verts	3000 verts	6000 verts
0,76[s]	0,90[s]	0,97[s]
0,80[s]	0,81[s]	0,72[s]
0,88[s]	0,78[s]	0,81[s]
0,75[s]	0,75[s]	0,84[s]
0,71[s]	0,84[s]	0,79[s]
Model_02		
500 verts	3000 verts	6000 verts
0,81[s]	0,89[s]	0,91[s]
0,79[s]	0,71[s]	0,77[s]
0,78[s]	0,79[s]	0,81[s]
0,71[s]	0,73[s]	0,74[s]
0,79[s]	0,81[s]	0,82[s]
Model_03		
500 verts	3000 verts	6000 verts
0,65[s]	0,75[s]	0,77[s]
0,70[s]	0,84[s]	0,86[s]
0,69[s]	0,71[s]	0,81[s]
0,76[s]	0,83[s]	0,79[s]
0,75[s]	0,79[s]	0,85[s]

Po przeprowadzeniu badań wydajności, stworzona została dodatkowa scena pozwalająca na porównanie obu silników pod względem jakości generowanego obrazu. Na Rysunku 6 przedstawiono wykorzystane do badań trzy modele obiektów archeologicznych wkomponowane w wymodelowaną scenę. Nowa scena składała się z biblioteczki, podłoża, w którego skład wchodziła roślinność oraz kamienie, dodatkowo wymodelowane zostały elementy dekoracyjne otaczające zeskanowane obiekty archeologiczne. Za tło posłużył obiekt o nazwie plane, prostokąt złożony z czterech wierzchołków, z nałożoną teksturą gór. Użyta została również mapa HDRI generująca ostrzejsze światło i cienie. Rysunek podzielony został na cztery części. Widoczny jest obraz wygenerowany przez silnik Eevee oraz efekt pracy silnika Cycles.

Rysunek 6: Różnice w generowaniu obrazu przez badane silniki.

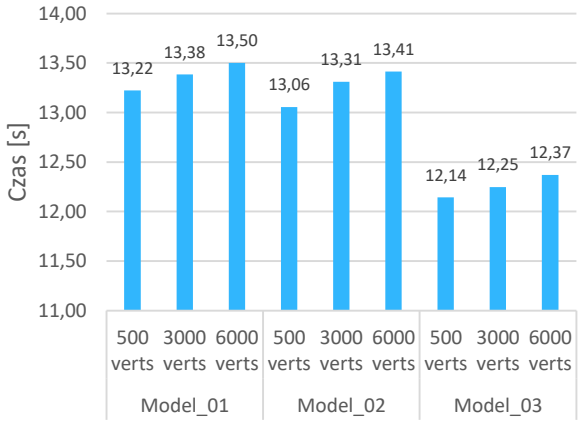
W tabeli (Tabela 4) przedstawione zostały czasy renderowania sceny dla różnych wariantów tekstur. Badanie zostało przeprowadzone na silniku Cycles. Scena składała się z dwóch obiektów. Użyty został Model_01, który składał się z 6000 wierzchołów, oraz tło w postaci obiektu plane dostępnego natywnie w aplikacji Blender.

Tabela 4: Pomiar czasu renderowania dla różnych wariantów tekstur

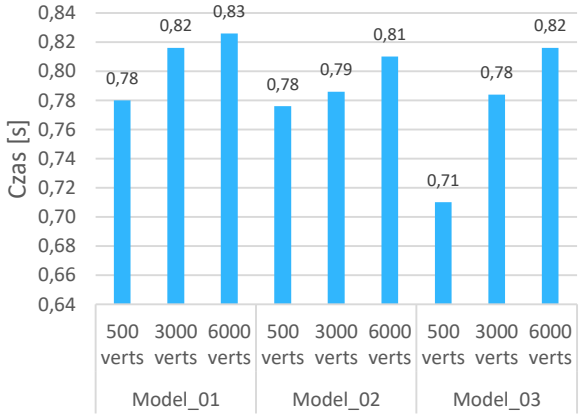
Model_01, 6000 verts		
512 x 512	1024 x 1024	2048 x 2048
13,66[s]	13,74[s]	13,66[s]
13,64[s]	13,59[s]	13,69[s]
13,63[s]	13,67[s]	13,58[s]
13,73[s]	13,65[s]	13,67[s]
13,62[s]	13,60[s]	13,69[s]

5. Analiza wyników

Uzyskane wyniki pozwoliły na obliczenie średnich czasów renderowania dla każdego przypadku. Na Rysunku 7 i 8 przedstawiono średnie wyniki obu silników graficznych. Do wyliczenia średniej użyto 5 wyników na dany wariant modelu.



Rysunek 7: Średni czas renderowania scen dla silnika Cycles.



Rysunek 8: Średni czas renderowania scen dla silnika Eevee.

Analizując wykresy oraz uzyskane wyniki stwierdzić można, że czas renderowania na silniku Cycles w stosunku do silnika Eevee jest znacznie dłuższy. Cycles uzyskał średni czas renderowania wszystkich scen na poziomie 12,96 s. Średni czas renderowania dla silnika Eevee wynosił 0,79 s. Wynika to z wykonywania o wiele bardziej skomplikowanych obliczeń światła, okluzji otoczenia oraz cieni. Silnik Eevee ma za to bardzo dużą przewagę jeżeli głównym kryterium wyboru jest czas wyrenderowania sceny. Z obliczeń średnich czasów wynika, że Eevee pozwolił na skrócenie czasu renderowania średnio o 93%. Zwiększenie gęstości siatki miało niewielki wpływ na czas renderowania sceny w przypadku porównywanych silników i stworzonej sceny. Uzyskane wyniki świadczą o tym, że w przypadku silników różnica w ilości poligonów na scenie musiałaby być wielokrotnie wyższa, żeby można było zobaczyć znaczny wzrost czasu renderowania. Pomimo niewielkich wzrostów uzyskanych wyników, wykresy jednoznacznie pokazują, że czas oczekiwania na gotowy obraz zwiększa się wraz ze wzrostem poligonów badanych obiektów archeologicznych. Konieczność dodatkowych obliczeń dla zwiększonej ilości ścian obiektu wydłuża oczekiwanie na końcowy rezultat. Wykonywane są obliczenia dotyczące światła, cieni oraz obić, a wraz z zagęszczeniem siatki wzrasta ilość płaszczyzn dla których konieczne jest wykonanie tych obliczeń. Porównując dwa silniki graficzne Eevee oraz Cycles, widać różnicę między czasem wyrenderowania tej samej sceny na korzyść silnika Eevee. Ma to związek ze sposobem działania obu silników. Silnik Cycles wykorzystuje do renderowania obrazu technikę zwaną ray tracing. Operacje jakie wykonuje są dużo bardziej skomplikowane. Obliczenia dotyczą światła, odbić oraz cieniowania. Pozwala to uzyskać obraz z poprawnymi fizycznie opisanymi aspektami względem silnika Eevee, ale kosztem dużo dłuższego czasu renderowania.

Analiza uzyskanych wyników dla różnych wariantów tekstur wykazała, że w przyjętym scenariuszu zwiększenie ich rozdzielczości nie spowodowało wydłużenia czasu renderowanie sceny. Różnica między najdłuższym (13,74[s]), a najkrótszym (13,58[s]) uzyskanym czasem dla trzech przygotowanych wariantów to 1,16%.

Analizę jakości obrazów warto rozpocząć od cieniowania. Na rysunku dostrzec można różnice w tym aspekcie, szczególnie we wnętrzu szafki. Silnik Eevee nie generuje cieni tak dokładnych jak Cycles lub brakuje ich w ogóle. Ustawione zostało oświetlenie na takie, które generuje ostrzejsze bardziej widoczne cienie [12]. Różnice dostrzec też można w cieniowaniu gdzie obiekty stykają się z podłożem. Silnik Eevee nie wygenerował cieni pod modelem szafki przez co obraz sprawia wrażenie płaskiego, bez głębi. Drugą różnicą jest oświetlenie i odbicie. Silnik Cycles na elementach połyskujących generował poprawne fizyczne odbicia czego nie robił silnik Eevee.

6. Podsumowanie

W artykule przedstawiono porównanie dwóch silników graficznych, Cycles oraz Eevee. Metoda badań wykazała, że czas renderowania przy użycia silnika Cycles jest znacząco dłuższy niż na drugim badanym silniku. Oba silniki są również powtarzalne w uzyskiwanych wynikach. Przeprowadzone badania wykazały, że mając na

scenie określoną ilość wierzchołków, trójkątów możemy spodziewać się zbliżonych czasów renderowania. Wpływ zwiększenia jakości tekstur w obranym przypadku był nieznaczny.

Po bezpośrednim porównaniu i przeanalizowaniu tej samej sceny wyrenderowanej przez oba silniki stwierdzić można, że silnik Cycles generuje obraz wyższej jakości niż silnik Eevee. Jest to związane ze sposobem działania obu silników. Użyty w silniku Cycles ray tracing pozwala na uzyskanie poprawnych fizycznie odbić, cieni oraz efektów wolumetrycznych. Zaletą silnika Eevee w tym wypadku jest znacznie skrócony czas renderowania przygotowanych scen.

Literatura

[1] P. Skop, Porównanie wydajności silników gier na wybranych platformach, Journal of Computer Sciences Institute 7 (2018) 116–119.

[2] H. Żukowski, Porównanie wydajności trójwymiarowych gier z użyciem silników CryEngine i Unity, Journal of Computer Sciences Institute, 13 (2019) 345–348 https://doi.org/10.35784/jcsi.1330.

[3] A. Thorn, Unity i Blender. Praktyczne tworzenie gier, Helion, Gliwice, 2015.

[4] E. Głowienka, B. Jankowicz, B. Kwoczyńska, P. Kuras, K. Michałowska, S. Mikrut, A. Moskal, I. Piech, M. Strach, J. Sroka, Fotogrametria i skaning laserowy w modelowaniu 3D, pod redakcją Sławomira Mikruta Ewy Głowienki, Wyższa Szkoła Inżynieryjno-Ekonomiczna z siedzibą w Rzeszowie (2015) 78–79.

[5] P. Chlipalski, Blender. Architektura i projektowanie. Wydanie II, Helion, Gliwice, 2018.

[6] D. Holcer, Ewolucja grafiki komputerowej w świecie rozrywki interaktywnej, Repozytorium Uniwersytetu Jagielońskiego, praca magisterska, 2015, https://ruj.uj.edu.pl/xmlui/handle/item/207622.

[7] B. Simonds, Blender. Praktyczny przewodnik po modelowaniu, rzeźbieniu i renderowaniu, Helion, Gliwice, 2014.

[8] K. Kuklo, J. Kolmaga, Blender. Kompedium, Helion, Gliwice, 2007.

[9] Strona z przewodnikiem silnika Cycles: https://docs.blender.org/manual/en/latest/render/cycles/index.html, [06.04.2022].

[10] Strona z przewodnikiem silnika Eevee: https://docs.blender.org/manual/en/latest/render/eevee/index.html, [06.04.2022].

[11] Instrukcja obsługi skanera Artec Eva: https://www.manualslib.com/manual/1755978/Artec-3d-Eva.html, [04.04.2022].

[12] J. Birn, Cyfrowe oświetlenie i rendering. Wydanie II, Helion, Gliwice, 2007.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 224–227
Received: 22 May 2022
Accepted: 12 June 2022

Comparative analysis of React, Next and Gatsby programming frameworks for creating SPA applications

Analiza porównawcza szkieletów programistycznych React, Next oraz Gatsby do tworzenia aplikacji typu SPA

Adam Świątkowski*, Karol Ścibior

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

Abstract

This article presents a performance analysis of some of the most popular development frameworks based on the React library. The aim of the study was to show which of the technologies used to create the visual parts of web applications is the most efficient. The research was conducted with the use of 3 applications representing the same research content but based on the above-mentioned frontend technologies. In order to evaluate the performance, web browser development tools and React library were used, which proved that vanilla React is the most efficient for rendering pages with a lot of data.
Keywords: SPA; React; Next; Gatsby

Streszczenie

W niniejszym artykule przeprowadzono analizę wydajnościową jednych z najpopularniejszych szkieletów programistycznych opartych na bibliotece React. Celem badania było wykazanie, która z technologii wykorzystywanych do tworzenia części wizualnych aplikacji internetowych jest najwydajniejsza. Badanie przeprowadzono z wykorzystaniem 3 aplikacji reprezentujących tę samą treść badawczą, ale opartych na tytułowych technologiach typu front-end. W celu oceny wydajności wykorzystano narzędzia deweloperskie przeglądarek internetowych oraz biblioteki React dzięki czemu dowiedziono, że czysty React jest najwydajniejszy przy generowaniu stron o dużej liczbie danych.
Słowa kluczowe: SPA; React; Next; Gatsby

*Corresponding author

Email address: adam.swiatkowski@pollub.edu.pl (A. Świątkowski)

1. Wstęp

Zapotrzebowanie na efektowne strony internetowe rośnie z dnia na dzień. Firmy prześcigają się w tym, by ich wizytówki w postaci stron internetowych prezentowały się jak najlepiej. Z perspektywy programistycznej ważnym aspektem jest, aby proces wytwarzania aplikacji internetowych, był również efektywny. Dzięki podejściu w tworzeniu aplikacji o nazwie SPA (ang. Single Page Application) można osiągnąć obydwa omawiane aspekty. Model SPA pozwala na tworzenie dynamicznych interfejsów użytkownika, co eliminuje konieczność przeładowania strony, przy chęci przejścia na podstronę oraz zapewnia zdecydowaną przewagę w szybkości ładowania treści względem klasycznych stron internetowych [1].

Jedną z najpopularniejszych bibliotek do tworzenia części wizualnych stron internetowych, napisanej w języku JavaScript, jest React. Biblioteka ta została stworzona przez programistów odpowiedzialnych za stworzenie sieci społecznościowej Facebook. Na dzień dzisiejszy pochwalić się ona może ponad 15 mln. pobrań tygodniowo [2].

Technologia Next jest szkieletem programistycznym opartym na bibliotece React. Fundamentalną różnicą względem działania tej technologii względem biblioteki React jest fakt, że treść prezentowana użytkownikowi jest wstępnie generowana na serwerze, dzięki czemu drastycznie zmniejsza się czas potrzebny na załadowanie strony [3].

Technologia Gatsby jest również szkieletem programistycznym opartym na bibliotece React. Głównym założeniem twórców tej technologii była jej zgodność ze wzorcem architektonicznym stworzonym przez firmę Google - PRPL (Push, Render, Pre-cache, Lazy-load). Pozwala to na zauważalne przyspieszenie załadowania treści przez klienta przeglądarkowego [4].

W celu oceny wydajności omówionych technologii wykorzystany zostanie panel deweloperski dostępny w przeglądarkach internetowych Chrome oraz Firefox. Narzędzia deweloperskie w obu przeglądarkach pozwalają na sprawdzenie zarówno czasu ładowania całej strony jak i poszczególnych jej elementów (komponentów).

2. Cel i zakres badań

Celem badań była ocena wydajności wybranych szkieletów programistycznych w porównaniu z biblioteką bazową React. Pozwoli to na sprawdzenie jak różne koncepcje implementacji tej samej biblioteki przenoszą się na rzeczywiste wyniki wydajności. Na tej podstawie wskazane zostanie najwydajniejsze rozwiązanie.

Postawione zostały następujące tezy badawcze:
T1: Szkielet programistyczny Next jest najbardziej wydajny przy tworzeniu stron o umiarkowanej liczbie danych od 100 do 1000 wierszy.
T2: Klasyczny React jest najbardziej wydajny dla aplikacji internetowych o dużej liczbie danych od 10000 wierszy.

3. Przegląd literatury

Analizując dostępną literaturę ciężko znaleźć zestawienie porównujące jednocześnie wszystkie tytułowe technologie. Zdecydowanie łatwiej było uzyskać źródła z wykorzystaniem ogólnodostępnych wyszukiwarek internetowych. Z uwagi na dosyć młody wiek badanych szkieletów programistycznych znalezienie odpowiednich źródeł akademickich może stanowić duże wyzwanie.

Przykładowym artykułem zbieżnym z zakresem badań jest artykuł pt. „create-react-app vs Gatsby.js vs Next.js" [5], w którym autor zestawia ze sobą wymienione szkielety programistyczne i porównuje je ze sobą pod kątem:

- wydajności przy prezentacji danych,
- wymagań w kontekście utrzymania i konserwacji,
- mechanizmów wspierających pozycjonowanie w wyszukiwarkach internetowych,
- częstotliwości aktualizacji wprowadzanych przez twórców technologii.

Wymienione kryteria są zbieżne z zakresem badawczym niniejszej pracy i pozwalają na wskazanie mocnych i słabych stron każdego z narzędzi.

4. Analizowane technologie

Do przeprowadzenia badań zostały wykorzystane technologie oparte na bibliotece React. Były to: zwykły React (szablon CRA (ang. Create React App)), NextJS oraz GatsbyJS. Wszystkie te technologie są do siebie bardzo podobne, jednak zostały stworzone do różnych celów. Proces twórczy aplikacji opartych na bibliotece React bazuje na modelu deklaratywnym, w którym każdy komponent definiuje fragment aplikacji używając składni JSX.

4.1. React

Szablon *create-react-app* to oficjalna, najprostsza metoda do utworzenia projektu aplikacji typu SPA opartego na bibliotece React. Oferuje on nowoczesną i minimalną konfigurację pozwalającą na natychmiastowe rozpoczęcie pracy bez wprowadzania dodatkowych ustawień i narzędzi.

4.2. Next

Szablon NextJS jest zbudowany na bibliotece React, dokładając do niej własne funkcjonalności i optymalizację. Między innymi pozwala na hybrydowe korzystanie ze statycznego generowania widoków oraz generowania widoków po stronie serwera. Dodatkowo posiada takie funkcjonalności jak automatyczna optymalizacja zdjęć, wbudowane funkcje językowe, wsparcie dla biblioteki TypeScript, czy możliwość pisania punktów końcowych REST API.

4.3. Gatsby

Szablon Gatsby również jest zbudowany w oparciu o bibliotekę React, oferując szybkie i elastyczne rozwiązanie do tworzenia stron internetowych opartych na treściach pobieranych z silników typu CMS. Gatsby najczęściej używany jest jako generator stron statycznych dzięki swojej szybkości i natywnym rozwiązaniom służącym do optymalizacji SEO (ang. Search Engine Optimization) czy zwiększania dostępności stron internetowych.

5. Metoda badań

Do przeprowadzenia badań stworzono trzy aplikacje oparte na narzędziach React, Next oraz Gatsby. Projekty utworzono kolejno za pomocą komend:

- *yarn create react-app react-performance-analysis*,
- *yarn create next-app next-performance-analysis,*
- *npm init gatsby gatsby-performance-analysis*.

W każdym z projektów zaimplementowano identyczne, minimalistyczne interfejsy użytkownika zawierające tabelę wyświetlającą dane wygenerowane przy pomocy biblioteki *hoaxer* [6]. Pomiary przeprowadzono korzystając z przeglądarki internetowej Brave [7] w wersji 1.36.119 z silnikiem Chromium w wersji 99.0.4844.83, ze względu na popularność tego silnika.

Listing 1: Kod odpowiedzialny za generowanie danych

```
import hoaxer from 'hoaxer'

export const getMockData = rows =>
 [...Array(rows).keys()].map(index => ({
   id: index + 1,
   name: hoaxer.name.findName(),
   email: hoaxer.internet.email(),
   avatar: hoaxer.internet.avatar(),
   job: hoaxer.name.jobTitle(),
   company: hoaxer.company.companyName(),
   phone: hoaxer.phone.phoneNumber(),
   country: hoaxer.address.country(),
 }))
```

Po zamontowaniu komponentu do struktury przeglądarki DOM (ang. Document Object Model), dane generowane są w sposób synchroniczny oraz ładowane są do stanu komponentu i wyświetlane w tabeli (Listing 2). Dla każdego wiersza wygenerowanych danych, wyświetlany jest wiersz w tabeli, gdzie każda komórka zawiera element HTML typu span z wyświetloną treścią.

Listing 2: Kod odpowiedzialny za załadowanie danych do stanu

```
const [data, setData] = React.useState([])
React.useEffect(() => {
   const mockData = getMockData(1000)
   setData(mockData)
 }, [])
```

Załadowane dane są wyświetlane analogicznie w każdej z napisanych aplikacji za pomocą JSX. Cała tabelka została opakowana w komponent React.Profiler, który pozwala sprawdzić czas w jakim ładowane są jego

komponenty potomne (w tym przypadku jest to tabela). W przypadku aplikacji pomiarowych dla każdego pomiaru wyświetlane są po dwie wartości. Najpierw wyświetlane są pomiary dla pierwszego zamontowania tabelki do strefy DOM (w momencie, gdy tabela jest pusta) oraz drugie wyświetlane w momencie załadowania kolejno 100, 1000 i 10000 wierszy wygenerowanych danych. Najważniejszą wyświetlaną informacją, na której bazowane są pomiary badawcze jest *actualTime*, czyli całkowity czas mierzony od startu do zakończenia ładowania komponentu (Listing 3).

Dzięki temu, że każdy z badanych szablonów programistycznych oparty jest na bibliotece React to sposób deklarowania komponentów jest identyczny. Największe różnice pojawiają się w momencie budowania aplikacji i generowania widoku.

Listing 3: Kod odpowiedzialny za wyświetlanie danych i dokonanie pomiaru

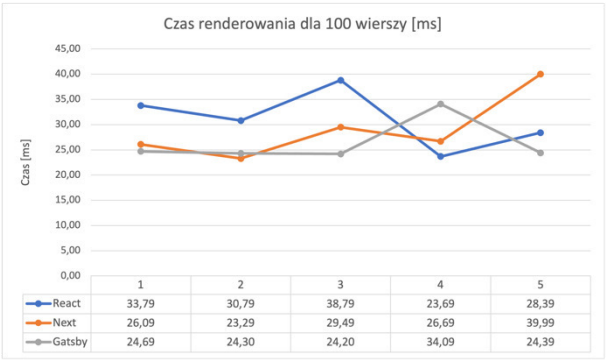
```
<div>
      <header>
        <h1>React Performance Analysis</h1>
      </header>
      <main>
        <React.Profiler id="test"
          onRender={(
            profilerId,
            mode,
            actualTime,
            baseTime,
            startTime,
            commitTime
          ) => {
            console.log({profilerId, mode,
actualTime, baseTime, startTime,commitTime})
          }}
        >
          <table>
            <thead>
              {labels.map(el => (
                <th key={el}>{el}</th>
              ))}
            </thead>
            <tbody>
              {data.map((el, i) => (
                <tr key={i}>
                  {Object.entries(el).map(([key,
label], j) => (
                    <td key={j}>
                      <span>{label}</span>
                    </td>
                  ))}
                </tr>
              ))}
            </tbody>
          </table>
        </React.Profiler>
      </main>
</div>
```

6. Platforma testowa i wyniki badań

Do badań zastosowano środowisko:

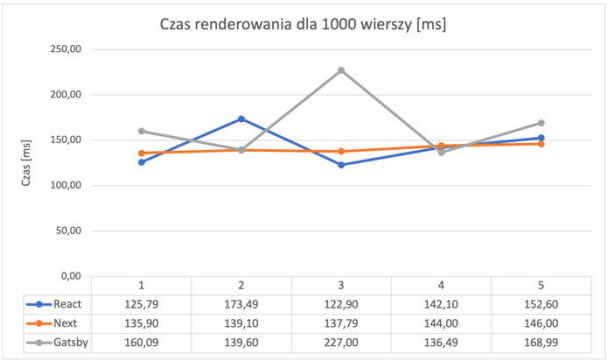
- Przeglądarka internetowa: Brave 1.36.119 Chromium: 99.0.4844.83,
- OS: MacOS Monterey 12.0.1,
- CPU: Intel Core i5 1.4GHz czterordzeniowy,
- RAM: 16GB 2133 MHz LPDDR3,
- GPU: Intel Iris Plus Graphics 645 1536 MB,
- Model: Macbook Pro 13 2020.

Na Rysunkach 1-3 zaprezentowano wyniki pomiarów czasu ładowania strony dla 100, 1000 oraz 10000 wierszy w tabeli.



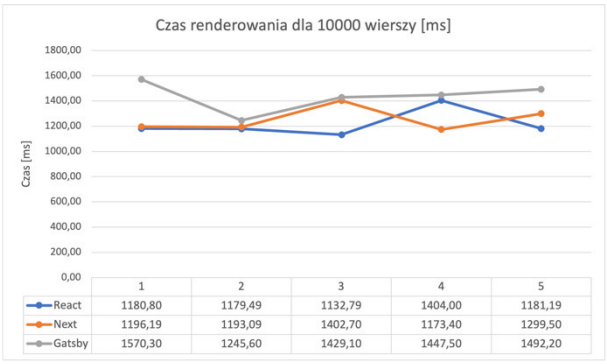
	1	2	3	4	5
React	33,79	30,79	38,79	23,69	28,39
Next	26,09	23,29	29,49	26,69	39,99
Gatsby	24,69	24,30	24,20	34,09	24,39

Rysunek 1: Wyniki pomiarów dla 100 wierszy.



	1	2	3	4	5
React	125,79	173,49	122,90	142,10	152,60
Next	135,90	139,10	137,79	144,00	146,00
Gatsby	160,09	139,60	227,00	136,49	168,99

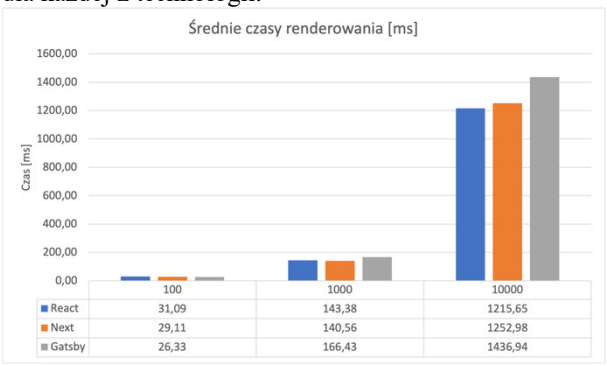
Rysunek 2: Wyniki pomiarów dla 1000 wierszy.



	1	2	3	4	5
React	1180,80	1179,49	1132,79	1404,00	1181,19
Next	1196,19	1193,09	1402,70	1173,40	1299,50
Gatsby	1570,30	1245,60	1429,10	1447,50	1492,20

Rysunek 3: Wyniki pomiarów dla 10000 wierszy.

Rysunek 4 prezentuje średni czas ładowania strony dla każdej z technologii.



	100	1000	10000
React	31,09	143,38	1215,65
Next	29,11	140,56	1252,98
Gatsby	26,33	166,43	1436,94

Rysunek 4: Średnie wyniki pomiarów dla 100,1000 i 10000 wierszy.

7. Analiza wyników

Na podstawie analizy uzyskanych wyników zauważyć można umiarkowane wahania w czasach ładowania strony dla różnej wielkości prezentowanej treści w zależności od badanej technologii.

Dla pierwszej przeprowadzonej serii, czyli badania czasu ładowania strony zawierającej 100 wierszy najlepszą technologią okazał się być szkielet programistyczny Gatsby. Uzyskał on średnio najniższy wynik względem pozostałych badanych technologii – 26,33 ms. Stanowi to różnicę względem technologii React o 4,76 ms. Next uzyskał wynik nieco gorszy od technologii Gatsby i wynosił on 29,11 ms.

Drugą serią badań, były próby czasów ładowania strony zawierającej 1000 wierszy. W tym przypadku patrząc na Rysunek 2, zauważyć można najbardziej stabilną wydajność technologii Next. Szkielet ten okazał się również najbardziej wydajną technologią, dla opisywanej próby i uzyskał średni wynik czasu ładowania strony na poziomie 140,56 ms. React uzyskał wynik na poziomie 143,38 ms, a Gatsby na poziomie 166,43 ms.

W trzeciej serii badań, czyli próbie czasów ładowania strony zawierającej 10000 wierszy, różnice czasów między technologiami stały się bardziej zauważalne. Najbardziej wydajna technologia w tej próbie, czyli React, okazała się być lepsza od najmniej wydajnej średnio, aż o 221,29 ms. Tym samym, najwydajniejszą technologią dla najtrudniejszej z prób okazała się być biblioteka React uzyskując średni wynik na poziomie 1215,65 ms.

8. Wnioski

Wyniki badań pokazują umiarkowane dysproporcje w wydajności badanych technologii względem liczby generowanych danych w widoku aplikacji testowej, do której je zastosowano.

Każdy z badanych szkieletów programistycznych bazuje na bibliotece React. Można więc sądzić, że każdy z autorów technologii Next oraz Gatsby, chciał ulepszyć bibliotekę React tworząc dla niej nakładkę, tak by uzyskać jak najlepsze i najbardziej wydajne działanie w różnych scenariuszach. Aplikacja testowa stworzona na potrzeby badań imituje właśnie scenariusze dla prostej aplikacji, gdzie zmienia się jedynie liczba wyświetlanych danych.

Każda z technologii okazała się świadczyć o tym czym kierowali się jej twórcy, co potwierdzają wyniki dwóch pierwszych serii, gdzie wygrały szkielety Next oraz Gatsby. Szkielety te nadają się idealnie do prostych aplikacji internetowych z niewielką ilością danych wyświetlanych na jednej stronie. Wyniki badań częściowo potwierdzają tezę T1 postawioną w niniejszym artykule. Technologia Gatsby okazała się najwydajniejsza dla stron o najmniejszej ilości danych, natomiast dla stron o umiarkowanej ilości danych najlepszy okazał się szkielet programistyczny Next.

Dla aplikacji przedstawiającej dużą liczbę danych najwydajniejsza okazuje się czysta biblioteka React, co potwierdza tezę T2. Warto zauważyć jednak, że wyświetlanie aż tak dużej ilości danych w aplikacjach internetowych, bez użycia metod optymalizacyjnych takich jak ładowanie na żądanie (ang. on-demand loading), znane również jako leniwe ładowanie (ang. lazy loading), lub wirtualizacji. Optymalizacja przez programistę tworzonego kodu, na pewno zmniejszyłaby różnice w wydajności poszczególnych technologii i bibliotek.

Wyniki badań są również zbieżne z wynikami badań przeprowadzonych w artykule pt.: „create-react-app vs Gatsby.js vs Next.js” [5], w którym to autor wskazał technologię Next jako najwydajniejszą, w zestawieniu z pozostałymi technologiami, ze względu na użycie w niej metody generowania widoku po stronie serwera, co przekłada się na wzrost szybkości jej ładowania. Autor zwrócił jednak uwagę, że metoda generowania widoków po stronie serwera jest dość obciążająca, co może przełożyć się na wydajność ładowania widoków, stron o dużej liczbie danych, powyżej 10000 wierszy.

Literatura

[1] E. Scott, SPA Design and Architecture: Understanding Single Page Web Applications, Manning Publications, 2015.

[2] Dokumentacja React, https://pl.reactjs.org/docs/getting-started.html, [22.04.2022].

[3] Dokumentacja Next, https://nextjs.org/docs, [22.04.2022].

[4] Dokumentacja Gatsby, https://www.gatsbyjs.com/docs/, [22.04.2022].

[5] CRA vs Gatsby vs Next, https://www.sensilabs.pl/2020/12/15/create-react-app-vs-gatsby-js-vs-next-js/, [22.04.2022].

[6] Biblioteka Hoaxer, https://www.npmjs.com/package/hoaxer, [22.04.2022].

[7] Przeglądarka internetowa Brave, https://brave.com/, [22.04.2022].

[8] J. Wagner, Web Performance in Action: Building Faster Web Pages, Manning Publications, 2017.

[9] C. R. Adams, Mastering JavaScript High Performance, Packt Publishing, 2015.

[10] R. Nowacki, M. Plechawska-Wójcik, Analiza porównawcza narzędzi do budowania aplikacji Single Page Application - AngularJS, ReactJS, Ember.js, Journal of Computer Sciences Institute 2 (2016) 98-103.

[11] A. M. Vipul, P. Sonpatki, ReactJS by Example - Building Modern Web Applications with React, 2016.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 228–235
Received: 7 June 2022
Accepted: 2 July 2022

Performance comparison between selected chess engines

Porównanie wydajności wybranych silników szachowych

Maciej Sójka*

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

Abstract
Selected chess engines were compared to each other in terms of performance, using Lucas Chess. The list of engines was divided into three categories, depending on strength in ELO points. The point of this study is to find the strongest and the lightest engines in each category. Then, each category was tested using three different starting positions. White, black and overall wins were highlighted. At the same time, data of CPU and RAM usage of each engine was collected. A script was developed to print CPU and RAM usage of a specific process. Maximum and average percent of used CPU and RAM were highlighted. Chess engines with most amount of wins were, from weakest to strongest: Bikjump, Rybka and Stockfish. Least amount of system resources was consumed by: Cinnamon, Demolito and Critter.

Keywords: chess; chess engines; performance comparison

Streszczenie
Wydajność wybranych silników szachowych została porównana z użyciem programu Lucas Chess. Listę silników podzielono na kategorie w zależności od punktów ELO. Celem badań było znalezienie najmocniejszych i najlżejszych silników w kategoriach. Każdą z kategorii zbadano pod kątem trzech ustawień szachownicy. Wyróżniono wygrane ogólne oraz wygrane jako różne kolory bierek. Równolegle przeprowadzono badanie zużycia zasobów komputera. Przygotowano skrypt zapisujący wartości procentowe wykorzystania pamięci RAM i procesora przez konkretne procesy. Wyróżniono średnie i maksymalne procentowe zużycie CPU i pamięci RAM. Silniki z największą liczbą wygranych, od najsłabszych do najsilniejszych, to: Bikjump, Rybka i Stockfish. Najmniejsze zapotrzebowanie na zasoby mają: Cinnamon, Demolito i Critter.

Słowa kluczowe: szachy; silniki szachowe; porównanie wydajności

*Corresponding author
Email address: maciej.sojka@pollub.edu.pl (M. Sójka)

1. Wstęp

Szachy współczesne powstały w piętnastym wieku w południowej Europie. Jest to gra dla dwóch graczy, która wykorzystuje kwadratową siatkę o szerokości ośmiu pól. Na początku rozgrywki każdy gracz posiada szesnaście typów bierek: króla, hetmana, dwie wieże, dwóch gońców, dwa konie i osiem pionków. Bierki oraz pola mają różne kolory (czarny i biały). Celem gry jest uniemożliwienie ruchu przeciwnika poprzez "szach mat", czyli zbicie króla.

Jednym z celów inżynierów i naukowców było stworzenie maszyny symulującej grę ludzkiego gracza. W tym celu stworzono tak zwane "silniki szachowe", czyli algorytmy analizujące pozycję bierek i zwracające ruch lub listę ruchów. Silniki szachowe zwykle sterowane są z poziomu linii poleceń i do interakcji w formie innej niż komendy tekstowe wymagany jest zewnętrzy interfejs graficzny (tak zwany "front-end").

Umiejętność graczy i silników szachowych wyrażana jest poprzez system punktowy ELO, który został stworzony przez Arpada Elo [1]. Punkty ELO są wyrażane poprzez niezerową liczbę. Zwycięzca dostaje punkty w przypadku wygranej, a przegrany traci tyle samo. W przypadku remisu wyżej oceniany gracz traci punkty na rzecz przeciwnika. Punkty ELO liczone są wewnątrz pewnej puli graczy. Ogólnoświatowy współczynnik ELO jest liczony przez Międzynarodową Federację Szachową (fr. Fédération Internationale des Échecs, FIDE [2]).

Celem badań jest porównanie silników szachowych w wyznaczonych kategoriach. Silniki będą grupowane na podstawie ich współczynnika ELO, a rozgrywki zostaną rozpoczęte na podstawie różnych pozycji startowych. Na każdą kategorię wyróżnione zostaną silniki:

- z największą ilością wygranych gier,
- wymagające najmniejszą ilość zasobów komputera.

2. Przegląd literatury

Jedną z pierwszych prac związaych z algorytmami znajdującymi optymalne ruchy jest aplikacja do rozwiązywania problemów "mat w x ruchów" opisana przez Vladan V. Vučković [3]. Logika programu wykorzystuje algorytm alfa-beta. Kod źródłowy został napisany w Asemblerze, co gwarantuje wysoką wydajność nawet na procesorze Celeron z taktowaniem 1GHz.

Wykorzystanie algorytmu alfa-beta w silniku szachowym zostało głębiej opisane w pracy W. B. Putra i L. Heryawan [4]. Użycie alfa-beta do redukcji niepotrzebnych krawędzi w drzewie możliwych ruchów zwiększyło wydajność badanego silnika.

Szersze wykorzystanie silnika szachowego zostało zrealizowane przez Hongyu Zang, Zhiwei Yu i Xiaojun Wan w formie automatycznego komentatora gry [5]. Do oceny ruchów został wykorzystany silnik stworzony na

bazie sieci neuronowej i uczenia maszynowego. Na podstawie oceny szachownicy przez silnik generowane są informacje tekstowe wyświetlane na ekranie.

Praca Marco Block, Maro Bader i innych [6] opisuje poszerzenie silnika szachowego wykorzystującego uczenie maszynowe o kategoryzację stanu szachownicy na podstawie aktualnego stanu gry (z ang. „otwarcie", „gra środkowa", „gra końcowa"). Implementacja kategoryzacji, wraz z powiększeniem bazy ruchów, na których uczy się silnik, pozwoliła na znaczne zwiększenie jego umiejętności.

W pracy Norhan Hesham, Osama Abu-Elnasr, i Samir Elmougy [7] został przedstawiony alternatywny silnik szachowy, wykorzystujący bibliotekę uprzednio zdeklarowanych optymalnych ruchów. Na początku baza ruchów była wypełniana danymi z rozgrywek wcześniejszych mistrzów szachowych, którą silnik stopniowo dopełniał własnymi grami. Użycie takiego rozwiązania zwiększyło możliwości silnika wobec innych rozwiązań.

Hybrydowa reprezentacja szachownicy [8] przedstawiona przez Sigit Kariagil Bimonugroho i Nur Ulfa Maulidevi miała na celu przyspieszenie procesu analizy rozgrywki i wybrania optymalnego ruchu. Wyniki rozgrywek szachowych pokazały, że opisywana reprezentacja jest szybsza i mocniejsza od niektórych alternatyw.

Opis i porównanie dwóch silników szachowych zostało dokonane w pracy Shiva Maharaj, Nicka Polsona i Alexa Turka [9]. Głównym celem badań była ocena rozwiązania Łamigłówki Plasketta przez silniki Stockfish i LCZero. Obydwa silniki i łamigłówka zostały szeroko opisane od strony teoretycznej. Stockfish ostatecznie uzyskał lepsze wyniki ze względu na szybkość algorytmu wyszukującego optymalny ruch.

3. Metodyka badań

Komputer, na którym przeprowadzano badania, to Lenovo IdeaPad L340 Gaming. Pełna specyfikacja sprzętu opisana jest przez Tabelę 1. Laptop pracował w następującej konfiguracji:

- w opcjach zasilania pobór mocy został ustawiony na tryb „najwyższej wydajności",
- w opcjach UEFI praca wiatraków została ustawiona na tryb „najwyższej wydajności",
- podłączona została dodatkowa podstawka chłodząca.

Tabela 1: Specyfikacja podzespołów komputera badawczego

Procesor	Intel Core i7-9750H
Karta graficzna 1	NVIDIA GeForce GTX 1650
Karta graficzna 2	Intel UHD Graphics 630
Pamięć RAM	16 GB
System operacyjny	Windows 11

Do wykonania eksperymentu użyty został program Lucas Chess w wersji R 2.01a. Lucas Chess jest to zestaw narzędzi do gry, nauki i treningu umiejętności szachowych napisany w języku Python [10]. Zawiera on między innymi funkcję turnieju silników szachowych z gotową listą komputerowych graczy. W ramach pracy wyodrębniona została część silników, podzielona później na następujące kategorie:

- Kategoria 1 (do 2500 ELO):
  - Bikjump 2.01 – napisany w języku C++ przez Aarta J. C. Bika; Bikjump oferuje możliwość korzystania z bazy danych gier końcowych,
  - Cdrill 1800 build 4 – zaprojektowany przez Ferdinanda Moscę, ma za zadanie symulowanie gracza poniżej poziomu 2000 ELO,
  - Cinnamon 1.2c – napisany w języku C++ ze wsparciem wielu architektur i systemów; Cinnamon jest również dostępny do wykorzystania w stronach internetowych jako otwartoźródłowa biblioteka JavaScript,
  - Clarabit 1.00 – napisany w 2008 roku przez Salvadora Pallaresa Bejarano; Clarabit wykorzystuje "magiczne tablice bitów" [11], dzięki którym możliwe ruchy gońca i wieży mogą zostać obliczone równocześnie,
  - Maia-1900 – zmodyfikowana wersja LCZero przeznaczona do jak najlepszej symulacji ludzkiego gracza, w tym ludzkich błędów. Maia jest silnikiem wykorzystującym sieci neuronowe i oferuje poziomy trudności od 1100 do 1900 ELO,
  - Tarrasch ToyEngine Beta V0.906 – prosty silnik napisany w języku C++ przez Billa Forstera na potrzeby interfejsu użytkownika o tej samej nazwie (Tarrasch).
- Kategoria 2 (2501 – 2999 ELO):
  - Arminius 2017-01-01 – zaprojektowany przez Volkera Annussa, wykorzystuje sieć neuronową, oraz od wersji 2017, bardziej skomplikowaną wersję "magicznych tablic bitowych", zwaną "tablicami czarnej magii" [11],
  - Cheng 4.40 – napisany w języku C++ przez Martina Sedlaka ze wsparciem wielu systemów operacyjnych; Cheng wykorzystuje technikę zwaną "leniwym SMP" [12], która umożliwia wzajemne "podglądanie" postępu pracy różnych wątków podczas znajdywania ruchów,
  - Daydreamer 1.75 JA – napisany w języku C przez Aarona Beckera; nazwa "Daydreamer" pochodzi od zachowania pierwszych wersji silnika, które w losowych momentach popełniały ogromne błędy, przypominając graczy "zamyślonych"; początkowo silnik wykorzystywał uproszczoną funkcję ewaluacyjną Tomasza Michniewskiego, przypisując numeryczną wagę pionkom i polom szachownicy,
  - Demolito 32bit – opracowany w języku C przez Lucasa Braescha; Demolito wykorzystuje "okienko aspiracji" [13], które wydziela przestrzeń możliwych ruchów w pewnej odległości od zgadywanej wartości tak, aby możliwie przyspieszyć wyszukiwanie; jeśli optymalny ruch znajduje się poza "oknem", program wydziela "okno" z większym zakresem,
  - Gambitfruit Beta 4bx – napisany przez Ryana Beniteza w języku C++ i inspirowany silnikami Toga oraz Fruit; Gambitfruit ma za zadanie być wyjątkowo agresywnym przeciwnikiem,

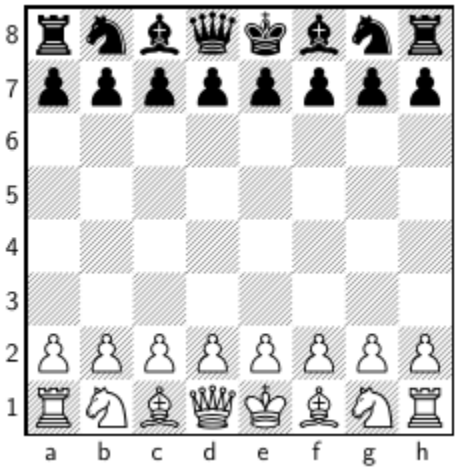
- Rybka 2.3.2a - rozwijany przez mistrza międzynarodowego Vasika Rajlicha, w latach 2006-2010 zdobywca czołowych miejsc międzynarodowych mistrzostw szachowych; śledztwo w 2011 roku zdyskwalifikowało Rybkę z późniejszych wydarzeń ze względu na oskarżenia o identyczność z silnikiem Fruit,
- Toga deepTogaNPS 1.9.6 – (początkowo sekretnie) oparty na silnik Fruit i zaprojektowany przez Fabien Letouzey; Toga po oskarżeniach o plagiat został wydany w wersji drugiej pod licancją GNU 2.0; najnowsze wydania Togi wykorzystują sieci neuronowe,
- Zappa 1.1 – zaprojektowany przez Anthony'ego Cozzie; po dyskwalifikacji w 2011 Zappa przejął część wygranych Rybki; na jego podstawie powstał komercyjny silnik szachowy Zap!Chess.

- Kategoria 3 (3000 ELO i więcej):
  - Andscacs 0.9432n – stworzony przez Daniela José Queraltó na przełomie 2013 i 2014 roku; na mistrzostwach WCRCC 2016 Andscacs otrzymał drugie miejsce nie przegrywając żadnej rozgrywki, co uniemożliwiło mistrzowi (Komodo) osiągnięcie perfekcyjnego wyniku [14],
  - Critter 1.6a – napisany w 2008 roku przez Richarda Vidę, początkowo w języku Pascal; w 2009 został przepisany na język C/C++; Critter jest w stanie wyszukiwać optymalny ruch używając do ośmiu wątków jednocześnie,
  - Gull 3 –stworzony przez ThinkingALot i inspirowany innymi silnikami szachowymi; od wersji 1.2 Gull jest opisany przez jeden plik źródłowy C++ i wspiera rekurencyjne wyszukiwanie ruchów; od wersji trzeciej dostępne są porty na systemy Linux i MacOS,
  - Hannibal 1.4b – stworzony przez Sama Hamiltona i Edsela Apostola ze wsparciem Audy Arandela; głównym celem Hannibala jest bycie najsilniejszym, jak to tylko możliwe; do wyszukiwania ruchów wykorzystuje on algorytm alfabeta, bazy danych końcowych ruchów oraz tablice bitowe,
  - Houdini 1.5a – zaprojektowany w 2010 roku przez Roberta Houdarta i inspirowany innymi silnikami, w tym Stockfish; od wersji 2.0 wzwyż Houdini jest dostępny wyłącznie w wersjach płatnych, a starsze wersje są darmowe dla celów innych niż komercyjne,
  - Komodo 12.1.1 – stworzony w 2010 roku przez Dona Daileya, od 2013 rozwijany przez Marka Leflera; książka otwarć Komodo powstała przy współpracy z ekspertem, Erdoganem Günesem; od 2011 roku najnowsze wersje Komodo są oprogramowaniem płatnym; wcześniejsze wersje dostępne są na platformy Windows, Linux, MacOS oraz Android; na jego podstawie powstał silnik Dragon, nauczony maszynowo na rozgrywkach Komodo,
  - LCZero 0.27 – adaptacja projektu silnika do gry w japońską grę "Go" pod zasady gry w szachy napisana w języku C++; LCZero jest otwartoźródłowy i ma na celu połączenie metody wyszukiwania Monte Carlo z technikami głębokich sieci neuronowych; do najbardziej optymalnego wykorzystania tych sieci polecane jest uruchamianie LCZero na karcie graficznej z rdzeniami CUDA [15],
  - Stockfish 14.1 – napisany w języku C++ i wydany pod otwartoźródłową licencją GPL 3.0, od 2018 roku faktycznie najsilniejszy powszechnie dostępny silnik szachowy; od 2019 roku, dzięki wsparciu ze strony społeczności skupionej wokół szacho-podobnej gry "Shogi", Stockfish uzyskał możliwość wykorzystania sieci neuronowych; Stockfish jest dostępny na platformach Windows, Linux, MacOS, iOS, Android oraz w prawie wszystkich serwisach oferujących grę szachową i szacho-podobną on-line.

3.1. Badanie liczby wygranych rozgrywek

Silniki szachowe były badane pod kątem liczby wygranych rozgrywek poprzez funkcję turniejów szachowych w programie Lucas Chess. Turnieje były przeprowadzane w postaci "każdy-z-każdym", przy czym dowolna para silników rozgrywała między sobą dokładnie trzy razy. Rozgrywki były objęte limitem czasowym, dając dokładnie jedną minutę na gracza. Silnikom przydzielano miejsce w rankingu na podstawie wyniku punktowego, gdzie wygrany dostaje 1 punkt, przegrany 0 punktów, a remisujący po połowie punktu. Przeprowadzono łącznie dziewięć turniejów, gdzie dla wcześniej wymienionych trzech kategorii przetestowano następujące startowe ułożenia pionków:

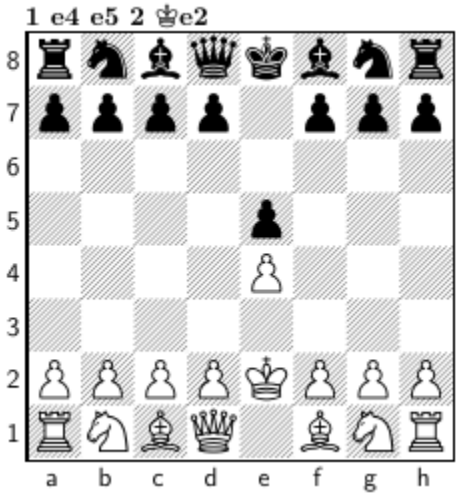
- klasyczna szachownica – standardowe ułożenie pionków według zasad gry w szachy, przedstawione na Rysunku 1,



Rysunek 1: Standardowe ułożenie szachownicy.

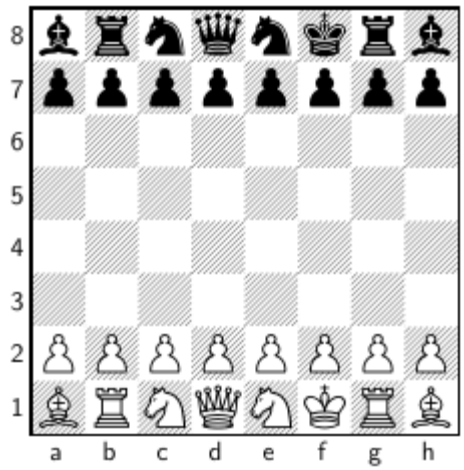
- otwarcie "Bongcloud" – przedstawione na Rysunku 2 i złożone z ruchów 1.e4 e5 oraz 2.Ke2, jest najgorszym rozpoczęciem rozgrywki z perspektywy gracza białego [16]; przesunięcie króla do przodu uniemożliwia późniejszą roszadę i blokuje hetmana oraz gońca na polu f1; uważa się, że „Bongcloud" został

wynaleziony przez jednego z użytkowników popularnego serwisu szachowego Chess.com o pseudonimie "Lenny_Bongcloud",



Rysunek 2: Otwarcie „Bongcloud".

- Chess960 – wariant szachowy stworzony przez Bobby'ego Fischera, w którym pozycja figur na pierwszej i ósmej linii szachownicy jest generowana losowo; aby rozgrywka była bardziej zbalansowana, obydwaj gracze mają takie samo ustawienie figur, gońce stoją na polach o przeciwnym kolorze, a król znajduje się pomiędzy wieże; w ten sposób możliwych jest 960 różnych ustawień pionków; Rysunek 3 przedstawia wykorzystane do badań jedno z możliwych ustawień.



Rysunek 3: Jedno z ustawień Chess960 użyte w badaniu.

3.2. Badanie zapotrzebowania na CPU i RAM

Do sprawdzenia wykorzystania procesora i pamięci systemowej wykorzystany został własny skrypt napisany w języku Python. Zadaniem skryptu miało być pobranie wartości procentowej wykorzystania CPU i RAM z procesu silnika i przesłanie danych do pliku .csv. Procesy wyszukiwane były po nazwie, którą należało odczytać z „menadżera zadań" systemu Windows. Lista nazw procesów wyszukiwanych przez skrypt opisana jest przez Listing 1. Proces o nazwie „maia" nie jest na liście, ponieważ silnik Maia jest oparty LCZero i system „widzi" go pod tą samą nazwą.

Listing 1: Lista procesów badanych silników szachowych

```
nazwy = [
  'CDrill_1800_Build_4.exe', 'cinnamon_1.2c-generic.exe',
  'lc0.exe', 'TarraschToyEngineV0.906.exe',
  'bikjump.exe', 'clarabit_100_x32_win.exe',
  'Arminius2017-01-01-32Bit.exe', 'cheng4.exe',
  'daydreamer-175-32-ja.exe', 'demolito_32bit_old.exe',
  'gfruit.exe', 'Rybka v2.3.2a.w32.exe',
  'DeepToga1.9.6nps.exe', 'zappa.exe',
  'andscacs_32_no_popcnt.exe', 'Critter_1.6a_32bit.exe',
  'Gull 3 w32 XP.exe', 'Hannibal1.4bx32.exe',
  'Houdini_15a_w32.exe', 'komodo-12.1.1-64-bmi2.exe',
  'Stockfish-14_1_x64-bmi2.exe'
]
```

Ciało skryptu widoczne na Listingu 2 składa się z pętli nieskończonej sprawdzającej dla każdego elementu listy, czy istnieje proces w systemie o takiej samej nazwie. Następnie, w przypadku zgodności nazw, otwierany jest plik .csv o nazwie takiej samej jak proces (jeśli takowego pliku nie ma, to jest on tworzony automatycznie). Do pliku dopisywane są informacje o procentowym użyciu procesora i pamięci systemowej oraz znak końca linii. Po zamknięciu pliku skrypt czeka sekundę, aby następnie przejść do kolejnej iteracji pętli. Dostęp do procesów systemowych umożliwia biblioteka psutil [17], a do oczekiwania użyto biblioteki time. Skrypt działał podczas rozgrywki silników w turniejach szachowych.

Listing 2: Ciało pętli zapisującej dane do plików

```
while True:
  for n in nazwy:
    for p in psutil.process_iter():
      if p.name() == n:
        f = open(p.name()+".csv", "a")
        f.write(str(p.cpu_percent())+','+str(p.memory_percent()))
        f.write("\n")
        f.close()
  time.sleep(1)
```

4. Wyniki

Informacje o silnikach i rozegranych partiach w turniejach szachowych są dostępne w programie Lucas Chess. Po pomyślnym przebiegu turnieju generowana jest tabela ze szczegółowymi informacjami na temat wygranych, przegranych i remisów. Dodatkowo program oferuje eksport danych o turnieju do pliku o własnościowym formacie .mvm. Do badań wykorzystano liczby wygranych łączne, wygranych jako gracz biały, wygranych jako gracz czarny oraz punkty ELO.

Dane o zużyciu zasobów komputera zostały przeniesione z plików .csv do arkusza kalkulacyjnego, gdzie wyliczono maksymalne zużycie, średnie zużycie oraz odchylenie standardowe dla procesora i pamięci RAM.

4.1. Kategoria 1

Turniej z szachownicą klasyczną zakończył się zwycięstwem silnika Carabit. Każdy silnik szachowy rozegrał 30 gier, z czego Clarabit i Bikjump wygrywali średnio co drugą rozgrywkę. Ze względu na pierwszeństwo ruchu, liczba wygranych jako gracz biały jest nieznacznie większa od liczby wygranych jako czarny we

wszystkich przypadkach, oprócz silnika Tarrasch, który przegrał prawie każdą grę. Liczba wygranych jest wprost proporcjonalna do współczynnika ELO, co widoczne jest w Tabeli 2.

Tabela 2: Wyniki turnieju szachowego kategorii 1 na klasycznej szachownicy

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Clarabit	21	11	10	2058
Bikjump	19	10	9	2026
Cinnamon	17	9	8	1930
Maia	13	7	6	1900
Cdrill	11	5	6	1800
Tarrasch	1	0	1	1481

Przejście z pozycji startowej na otwarcie „Boncgcloud" przerzuca przewagę na gracza czarnego, co jest najbardziej widoczne dla „słabszych" silników. Wyjątkiem stanowi Cinnamon, wygrywający częściej jako bracz biały. Silnik Cdrill wykazał największą różnicę wyników wobec wcześniejszego turnieju, ogólnie wygrywając o 3 gry więcej, co przeniosło go w tabeli na miejsce czwarte. Maia, z taką samą liczbą całkowitych wygranych co wcześniej, spadł na miejsce piąte, co widać w Tabeli 3.

Tabela 3: Wyniki turnieju szachowego kategorii 1 rozpoczynając od otwarcia „Bongcloud"

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Clarabit	22	11	11	2058
Bikjump	20	9	11	2026
Cinnamon	14	8	6	1930
Cdrill	14	7	7	1800
Maia	13	4	9	1900
Tarrasch	2	0	2	1481

W turnieju Chess960 zwycięzcą został silnik Bikjump, wygrywając dwie z trzech rozgrywek. Tak jak dla szachownicy klasycznej, obydwaj gracze mieli szanse zbliżone do równych. Wyjątkiem stanowią Maia i Tarrasch. Silnik Maia, tak jak w przypadku otwarcia „Bongcloud", wygrywał częściej jako gracz czarny. Tarrasch, dzięki nietypowemu ułożeniu pionków, udało się wygrać siedem rozgrywek, z czego pięć jako gracz biały. Wygrane silników dla Chess960 widoczne są w Tabeli 4.

Tabela 4: Wyniki turnieju szachowego kategorii 1 dla Chess960

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Bikjump	20	9	11	2026
Clarabit	15	7	8	2058
Cinnamon	16	8	8	1930
Maia	12	3	9	1900
Cdrill	11	6	5	1800
Tarrasch	7	5	2	1481

Silniki z kategorii pierwszej wykazują się znikomym zapotrzebowaniem na RAM, co widoczne jest w Tabeli 5. Tarrasch i Cdrill pobierają mniej niż jeden promil z 16 GB dostępnej pamięci. Wszystkie badane silniki są napisane w postaci aplikacji C/C++ bez interfejsu użytkownika, tak więc pamięć wykorzystywana jest głównie do tablic i obliczeń numerycznych. Inaczej wygląda zużycie procesora. Tylko Cinnamon i Clarabit nie wykorzystywały chwilowo w pełni jednego z wątków.

Tabela 5: Wyniki pomiarów wydajności dla kategorii 1

Silnik	CPU śr. [%]	CPU maks. [%]	RAM śr. [‰]	RAM maks. [‰]
Bikjump	49,6	100,0	1,244	1,252
Cdrill	51,4	100,0	0,558	0,562
Cinnamon	42,6	81,2	2,298	2,303
Clarabit	43,7	83,0	2,774	2,776
Maia	46,1	92,3	1,841	4,361
Tarrasch	45,9	100,0	0,247	0,248

4.2. Kategoria 2

W przeciwieństwie do innych kategorii, silniki z przedziału 2501-2999 ELO są na podobnym poziome, co przekłada się na zróżnicowane wyniki rozgrywek. Każdy silnik w tej kategorii rozegrał łącznie 42 rozgrywki. Dla szachownicy klasycznej zwycięzcą został Cheng z niewielką przewagą wobec Rybki, co przedstawionej jest w Tabeli 6. Silnik Demolito, pomimo dość wysokiej liczby wygranych, uzyskał przedostatnie miejsce ze względu na to, że przegrał 20 z pozostałych 26 gier.

Tabela 6: Wyniki turnieju szachowego kategorii 2 na klasycznej szachownicy

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Cheng	25	13	12	2750
Rybka	23	12	11	2936
Toga	15	9	6	2843
Dayderamer	12	7	5	2670
Gambitfruit	16	9	7	2750
Arminius	12	7	5	2662
Demolito	16	15	1	2627
Zappa	2	2	0	2581

Rozpoczynając od otwarcia „Bongcloud", większość silników w kategorii drugiej radziła sobie zdecydowanie lepiej jako gracz czarny. Wyjątek stanowi Demolito, który wygrywał wyłącznie jako gracz biały, co widoczne jest w Tabeli 7. Największą liczbę wygranych uzyskał Rybka. Silnik Cheng, pomimo zdobycia „tylko" 20 wygranych, uzyskał wyższy wynik od Togi ze względu na większą liczbę remisów. Analogicznie, Arminius znajduje się na wyższej pozycji od silnika Daydreamer z tego samego powodu.

W przypadku Chess960 silnik Rybka zdominował kategorię drugą wygrywając 3 na 4 rozgrywki. Toga, nauczony maszynowo głównie na rozgrywkach tradycyjnych, spadł z trzeciego miejsca na szóste, za sprawą wielu przegranych. Silnik Demolito, tak jak wcześniej,

faworyzuje rozgrywkę jako gracz biały, co widoczne jest w Tabeli 8.

Tabela 7: Wyniki turnieju szachowego kategorii 2 rozpoczynając od otwarcia „Bongcloud”

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Rybka	23	8	15	2936
Cheng	20	4	16	2750
Toga	22	9	13	2843
Demolito	20	20	0	2627
Gambitfruit	14	4	10	2750
Arminius	9	4	5	2662
Daydreamer	11	5	6	2670
Zappa	8	3	5	2581

Tabela 8: Wyniki turnieju szachowego kategorii 2 dla Chess960

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Rybka	32	17	15	2936
Cheng	21	11	10	2750
Demolito	20	20	0	2627
Gambitfruit	16	11	5	2750
Arminius	9	4	5	2662
Toga	15	7	8	2843
Daydreamer	11	5	6	2670
Zappa	5	3	2	2581

Pomiary wydajności pokazały, że silniki szachowe z kategorii 2 potrzebują więcej zasobów systemowych niż „słabsze” odpowiedniki. Wszystkie silniki, oprócz Demolito, momentalnie osiągały 100 procent zużycia CPU. Najmniejsze średnie zapotrzebowanie na wątek procesora wykazuje Rybka. Tak jak wcześniej, pamięć RAM komputera nie jest wykorzystywana w znacznym stopniu. Najbardziej pamięciożerny jest silnik Zappa, wykorzystując do pół procenta pamięci systemu, co widoczne jest w Tabeli 9.

Tabela 9: Wyniki pomiarów wydajności dla kategorii 2

Silnik	CPU śr. [%]	CPU maks. [%]	RAM śr. [‰]	RAM maks. [‰]
Arminius	45,0	100,0	1,549	1,554
Cheng	48,4	100,0	1,421	1,426
Daydreamer	47,7	100,0	2,819	2,839
Toga	43,9	100,0	1,334	1,407
Demolito	47,1	95,6	1,341	1,346
Gambitfruit	49,3	100,0	1,275	1,278
Rybka	40,4	100,0	1,422	1,424
Zappa	41,1	100,0	1,947	4,883

4.3. Kategoria 3

Podobnie jak dla kategorii pierwszej, silniki powyżej 3000 ELO uzyskały wyniki takie, jak ich „siła w punktach”. Każdy silnik rozegrał 42 rozgrywki, z czego ogólny zwycięzca, Stockfish, wygrał aż 39. Niska pozycja Gull, tak jak w podobnych przypadkach wcześniej, wynika z dużej liczby przegranych. Niewielka preferencja wobec gracza białego także tu występuje, co widoczne jest w Tabeli 10.

Tabela 10: Wyniki turnieju szachowego kategorii 3 na klasycznej szachownicy

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Stockfish	39	20	19	3500
Komodo	27	14	13	3300
LCZero	14	8	6	3300
Andscacs	7	4	3	3264
Gull	10	6	4	3125
Critter	6	5	1	3091
Houdini	8	5	3	3093
Hannibal	4	3	1	3000

Zmiana przewagi między graczami poprzez otwarcie „Bongcloud” wywarło największy wpływ na silniki z kategorii trzeciej, co widać w Tabeli 11. Tylko Stockfish, Komodo i LCZero były w stanie wygrać więcej niż dwie rozgrywki na 21 możliwych jako gracz biały. Przewagę gracza czarnego efektywnie wykorzystał silnik Houdini, „przebijając” rywali z większą liczbą punktów ELO.

Tabela 11: Wyniki turnieju szachowego kategorii 3 rozpoczynając od otwarcia „Bongcloud”

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Stockfish	35	14	21	3500
Komodo	30	10	20	3300
LCZero	13	5	8	3300
Houdini	12	1	11	3093
Critter	11	2	9	3091
Andscacs	4	0	4	3264
Gull	12	2	10	3125
Hannibal	6	0	6	3000

Turniej Chess960 dla kategorii trzeciej przebiegał podobnie do turnieju z szachownicą klasyczną z tą różnicą, że wcześniej wysoko pozycjonowany silnik LCZero przestał wygrywać. Wynika to z budowy silnika – tak jak wcześniej opisane Maia i Toga, LCZero został wytrenowany maszynowo wyłącznie na szachownicy klasycznej. Podobnie jak algorytm do rozpoznawania obrazów nie dający sobie rady, gdy wejście jest obrócone lub odbite w lustrze, silnikowi LCZero udało się wygrać tylko 6 na 42 rozgrywki Chess960, co widoczne jest w Tabeli 12.

Silniki z kategorii trzeciej wykazują się względnie wysokim zapotrzebowaniem na pamięć RAM, co widoczne jest w Tabeli 13. Jedynie silniki Critter i Andscacs pobierały zasoby systemowe w liczbie porównywalnej do kategorii poprzednich. Stockfish, LCZero i Komodo zostały napisane pod architekturę 64-bitową oraz domyślnie korzystały z dwóch wątków, co przełożyło się na większe zapotrzebowanie na zasoby. Wartość powyżej stu procent przy zużyciu CPU oznacza, że program korzysta z więcej niż jednego wątku.

Tabela 12: Wyniki turnieju szachowego kategorii 3 dla Chess960

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Stockfish	36	16	20	3500
Komodo	30	16	14	3300
Gull	11	5	6	3125
Andscacs	7	4	3	3264
Critter	9	5	4	3091
Houdini	8	4	4	3093
LCZero	6	2	4	3300
Hannibal	6	3	3	3000

Tabela 13: Wyniki pomiarów wydajności dla kategorii 3

Silnik	CPU śr. [%]	CPU maks. [%]	RAM śr. [‰]	RAM maks. [‰]
Andscacs	47,6	88,8	2,927	2,932
Critter	42,0	99,6	1,649	1,652
Gull	48,4	100,0	4,780	4,795
Hannibal	48,7	100,0	7,476	7,480
Houdini	44,3	100,0	2,475	2,478
Komodo	90,4	194,5	8,938	8,963
LCZero	90,8	190,5	2,589	2,783
Stockfish	83,5	200,0	9,624	9,718

5. Wnioski

- Dla kategorii 1:
  - „najmocniejszym" silnikiem okazał się być Bikjump, wygrywając łącznie 59 gier na 90 możliwych, Clarabit stanął na drugim miejscu z 58 wygranymi, Cinnamon trzeci z liczbą zwycięstw równą 49,
  - najmniejsze zużycie pamięci RAM wykazywał Cdrill, Cinnamon najmniej wykorzystywał procesor.
- Dla kategorii 2:
  - największą liczbę wygranych osiągnął silnik Rybka (78 ze wszystkich 126 gier), na drugim miejscu Cheng z 66 wygranymi, trzeci Demolito, wygrywając 56 rozgrywek,
  - najmniejsze zużycie RAM wykazywał Gambitfruit, Rybka z najniższym średnim zapotrzebowaniem na procesor, Demolito z najniższym zużyciem maksymalnym.
- Dla kategorii 3:
  - „najsilniejszym" był Stockfish, osiągając 110 zwycięstw na 126 rozgywek, na drugim miejscu stanął Komodo z 87 zwycięstwami, silniki LCZero i Gull osiągnęły ex aequo trzecie miejsce z 33 wygranymi rozgrywkami każdy,
  - silnik Critter zużywał w najmniejszym stopniu pamięć systemową i procesor.
- Dodatkowo:
  - silniki szachowe wykorzystują procentowo znacznie mniej pamięć RAM niż procesor,
  - losowanie pionków w Chess960 powoduje, że silniki szachowe zbudowane w oparciu o uczenie maszynowe i sieci neuronowe radzą sobie zdecydowanie gorzej niż w przypadku szachownicy klasycznej,
  - silnik Demolito ma wysoką różnicę umiejętności w zależności od strony szachownicy – ze wszystkich 56 wygranych tylko jedna z nich nie była zwycięstwem jako gracz biały,
  - dokonane porównanie nie jest w pełni dokładne – przyszłe badania mogłyby uwzględnić różne limity czasowe podczas rozgrywki.

Literatura

[1] A. Elo, The Proposed USCF Rating System, Its Development, Theory, and Applications, Chess Life 22 (1967) 242-247.

[2] International Chess Federation - strona główna, https://www.fide.com/, [25.05.2022].

[3] V. V. Vučković, Realization of the Chess Mate Solver Application., Yugoslav Journal of Operations Research 14 (2004) 273-288, https://doi.org/10.2298/YJOR0402273V.

[4] W. B. Putra, L. Heryawan, Applying Alpha-beta Algorithm In A Chess Engine, Jurnal Teknosains UGM 6 (2016) 37-43.

[5] H. Zang, Z. Yu, X. Wan, Automated chess commentator powered by neural chess engine, arXiv (2019), https://doi.org/10.48550/arXiv.1909.10413.

[6] M. Block, M. Bader, E. Tapia, M. Ramírez, K. Gunnarsson, E. Cuevas, D. Zaldivar, R. Rojas, Using Reinforcement Learning in Chess Engines, Research in Computing Science 35 (2008) 31-40.

[7] N. Hesham, O. Abu-Elnasr, S. Elmougy, A New Action-Based Reasoning Approach for Playing Chess, Computers, Materials and Continua 69 (2021) 175-190.

[8] S. K. Bimonugroho, N. U. Maulidevi, A Hybrid Approach to Representing Chessboard using Bitboard and Compact Chessboard Representation, IOP Conference Series: Materials Science and Engineering 803 (2020), https://doi.org/10.1088/1757-899X/803/1/012018.

[9] S. Maharaj, N. Polson, A. Turk, Chess AI: Competing Paradigms for Machine Intelligence, Entropy 24 (2022) 550, https://doi.org/10.48550/arXiv.2109.11602.

[10] Strona internetowa programu Lucas Chess, https://lucaschess.pythonanywhere.com/home, [25.05.2022].

[11] Magiczne tablice bitów - definicja, https://www.chessprogramming.org/Magic_Bitboards, [25.05.2022].

[12] Leniwe SMP - definicja, https://www.chessprogramming.org/Lazy_SMP, [25.05.2022].

[13] Okno aspiracji - definicja, https://www.chessprogramming.org/Aspiration_Windows, [25.05.2022].

[14] Mistrzostwa ACCA World Computer Rapid Chess Championship 2016, https://www.chessprogramming.org/WCRCC_2016, [25.05.2022].

[15] B. Steinbach, M. Werner, XBOOLE-CUDA -- Fast Boolean Operations on the GPU (2014).

[16] Double bongcloud: why grandmasters are playing the worst move in chess, https://www.theguardian.com/sport/2021/mar/18/bongcloud-meme-opening-carlsen-nakamura, [20.06.2022]

[17] Biblioteka psutil - dokumentacja, https://psutil.readthedocs.io/en/latest/, [25.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 236–241
Received:7 June 2022
Accepted: 2 July 2022

Comparison of the offer of selected cloud service providers from the point of view of implementing IT projects based on open code

Porównanie ofert wybranych dostawców usług chmurowych z punktu widzenia realizacji projektów informatycznych opartych o otwarty kod

Jan Baran*, Sławomir Przyłucki

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

Abstract

The article presents a comparison of the offers of selected cloud computing providers in terms of their use in the process of developing and implementing IT services based on the idea of open-source code. The research concerns two groups of cloud services. The first one is provided according to the IaaS model and has the form of a virtual machine lease. The second service based on the PaaS model and is represented by database instances. The analysis is both qualitative (subjective assessment) and quantitative. In the latter case, it consists in a series of measurements of the parameters of virtual instances. Based on this analysis, the best public cloud service provider for users starting to use cloud computing resources was selected.

Keywords: cloud computing; IaaS service model; PaaS service model

Streszczenie

Artykuł przedstawia porównanie oferty wybranych dostawców chmury obliczeniowych pod kątem ich wykorzystania w procesie przygotowywania i wdrażania usług opartych o idee otwartego kodu źródłowego. Przeprowadzone badania dotyczą usług dostarczanych według modelu IaaS w postaci dzierżawy maszyny wirtualnej oraz usług dostarczanych według modelu PaaS w postaci usług baz danych. Zaprezentowana analiza ma charakter tak jakościowy (ocena subiektywna) jak i ilościowy, który polega na szeregu pomiarów parametrów instancji wirtualnych. Na podstawie tej analizy, wybrany został najlepszy dostawca usług w chmurze publicznej dla użytkowników rozpoczynających wykorzystywanie zasobów chmur obliczeniowych.

Słowa kluczowe: chmury obliczeniowe; model usług IaaS; model usług PaaS

*Corresponding author

Email address: jan.baran@pollub.edu.pl

1. Introduction

Observation of trends in the IT (Information Technology) market shows the increasing importance of services, which are implemented on a cloud computing infrastructure. The availability of this infrastructure and the related offer of IT service providers are expanding practically overnight [15]. Consequently, among individual software developers as well as small and medium-sized IT companies, there is a growing interest in using cloud resources on daily basis. In this area, services offered on public cloud computing infrastructure play by far the most important role [8],[15]. Thanks to these services, the technological and economic barrier of entering the IT market has significantly decreased. Simultaneously, the competitiveness on the market has also increased rapidly and the development of innovative IT solutions has accelerated [13],[18].

Considering the above observations, this article is devoted to a multi-criteria comparative analysis of the offer of cloud service providers who are leaders in this market [19]. The analysis is focused on the offers addressed to individuals and companies wishing to join the on-going IT revolution, and thus want to use the possibilities and advantages of cloud computing.

The purpose of the research is a multi-criteria evaluation of cloud services that are most often used by novice users and developers od open-source applications [17]. Therefore, the tests based on cloud computing resources that are offered under free tiers. The research method used for this study is comparative method. The chosen features of the selected cloud computing services are tested and compared. The cloud computing offers are selected and analyzed based on their published documentation and provider web pages.

1.1. Literature Review

Research and analysis of cloud computing services are the subject of many publications presented both in scientific journals and online resources [14]. However, few of them refer to the issue of comparing the characteristics and parameters of services used by all those who can be described as beginner users of cloud services. In our opinion, this is a point of view that deserves a deeper analysis.

A similar approach to the one presented in this article can be found in the article [1], in which the authors set themselves the goal of comparing the offer of the two most popular cloud service providers. Based on the comparisons' results, it was indicated that the Microsoft

Azure cloud offers the best interface for managing and monitoring services (the friendliest, especially for novice users). However, by analyzing all the results describing a wide range of service features, the authors indicated the Amazon AWS cloud as the one with the most positive sub-assessments. On the other hand, publication [2] contains a description of a comparative analysis of three selected cloud service providers. The research covered 12 services that are particularly useful for programmers implementing their projects in an organizational structure typical of small and medium-sized companies. The conclusions that have been defined based on the conducted analysis indicate that the Amazon AWS cloud is the best offer when the implemented IT project enters the implementation phase. In this phase, the ability to scale and monitor the project based on a global infrastructure begins to play a key role efficiently and flexibly. On the other hand, Microsoft Azure was indicated as a very good solution for small companies whose service projects are based on Microsoft solutions and when migration to the cloud from its own IT infrastructure (from a private cloud) is considered. Another service provider included in the discussed comparison, Google, and its flagship service Google App Engine confirmed advantage in terms of configuration flexibility and a wide set of solutions dedicated to developers, while at the same time competitive conditions for using the resources offered by Google infrastructure. The last of the companies compared, IBM and its IBM Cloud Platform proved their high competitiveness thanks to the unique implementation of virtualization and based on it a wide and (in many cases) unique offer of services.

A completely different approach was presented in [3]. The author of the research presented in it conducted a comparison of the costs of using several of the most popular services offered by selected cloud computing providers. The article compares the following services: data warehouse and virtual machines. In the first of the above-mentioned categories (data warehouse services), the highest marks were awarded to services offered in Microsoft Azure. In the group of virtual machine lease services, the comparison of costs in various payment models, ranging from the PAYG (pay-as-you-go) model, through annual subscriptions to the lease in the spot scheme, did not indicate a clear winner. The results cited in the article preferred, alternately, Amazon AWS or Microsoft Azure as the most advantageous service offer with specific lease parameters.

1.2. Service delivery models in cloud environments

Each public cloud provider has the choice of providing services according to three models.[4] From the point of view of novice users, the most important are those that provide the ability to quickly configure the necessary virtualized infrastructure and, on its basis, configure the development environment for their own service [9], [10], [11]. Based on this assumption, services provided according to the IaaS and PaaS models play a key role. All three models are characterized below.

- IaaS model (Infrastructure as a Services). This is the basic model that is offered as part of services available in public clouds. This service offers on-demand network resources, data warehouses and computing resources that can be easily scaled to individual needs. An example of this type of service model is the lease of virtual machines and network switches. They create a foundation for defining routes and access rules for individual infrastructure components.
- PaaS model (Platform as a Services). The second model covers all the services available under the IaaS model, but additionally also provides services related to the possibility of using specific supporting software. In practice, this means that in this model, immediate availability of selected software components is offered, the delivery of a virtual machine with the appropriate operating system installed (along with the necessary licenses), or the delivery of a ready-made database platform managed by the provider. Usually, the implementation of this model is additionally associated with the service provider's guarantees that the condition for the provision of individual services will remain unchanged. These conditions are defined in SLA (Service Level Agreement) contracts.
- Saas Model (Software as a Service). The last of the discussed models of providing services based on public clouds is the SaaS model. This model allows for the provision of infrastructure dedicated to the customer's own service, which is to be provided by a given service provider. For the end user, this means that he can implement a developed application or service that can be used immediately, and the costs of its operation are related to the real time of its operation.

Table 1 presents a summary of the features of the above-discussed service delivery models which based on public clouds.

Table 1: Characteristics of service delivery models

Model	Typical services	Use cases
SaaS	e-mail, task automation, service acquisition, social media	platform for service providers
PaaS	development team collaboration, application design and testing, database integration	supplemental or alternative to locally managed tool sets configured per project/service
IaaS	virtual machines, clustering, load balancing, storage resiliency, log access, monitoring	high-performance computing (HPC), web application, data storage

According to the information in the previous parts of this chapter, the presented comparative analysis concerns services provided according to the IaaS and PaaS models.

2. Benchmarking of free virtual machine instance lease services.

During the comparative analysis, the most popular cloud service providers were considered, respectively: Amazon AWS, Microsoft Azure, Google GCP. As part of all the above-mentioned clouds, new users are offered a set of services that can be used for one year at no cost. This creates an easy and cheap way to test and implement IT solutions for a wide range of small businesses or newly launched projects. In addition, thanks to this offer, thousands of users can create applications and services based on open-source licenses [16].

This chapter presents a comparative analysis of the lease offers of free virtual machines provided according to the IaaS model. All the presented measurements were carried out based on virtual machines running under the Linux operating system, Ubuntu 20.04 LTS distribution. Each of the VMs (Virtual Machines) had the same or similar parameters. The list of these parameters is presented in Table 2.

Table 2: Parameters of the virtual machines used during the tests

Cloud provider	Service name	vCPU	RAM
AWS	AMAZON EC2 t2.mikro	1	1GB
AZURE	Virtual Machine - B1s	1	1 GB
GCP	Compute Engine E2-micro	2	1 GB

The bench.sh script [5] will be used to carry out the first group of performance tests on virtual machines. The role of this script is to collect information about the software available on the VM, to test the speed of writing and reading data on the VM disk, and to test the network connection parameters to the server based on the speedtest.net service. The Phoronix-Test-Suite Benchmark program will be used for the implementation of another group of tests, which will enable the performance of RAM memory performance tests [6].

2.1. Test results

Table 3 shows the virtual machine parameters collected with the bench.sh script. These data include information about the processor model, the number of processor cores and their clock frequency. Additionally, data on disk capacity and operating memory capacity as well as system architecture and operating system version were collected. The last data set contains information about the location of the VM and the type of the virtualization system.

When analyzing the data from Table 3, it can be noticed that individual machines use different hypervisors. A virtual machine running in the Amazon AWS cloud uses the XEN manager, the machine running in the Microsoft Azure cloud uses Hyper-V (Microsoft Virtual Machine), and the machine working in the Google cloud is based on the KVM (Kernel-based Virtual Machine) hypervisor.

Table 3: Specification of the tested virtual machines

Parameter	AWS EC2	Azure Virtual Machine Linux – B1s	GCP Compute Engine E2-micro
Processor	Intel® Xeon® CPU E5-2686 v4@2.30GHz	Intel® Xeon® Platinum 8272CL	Intel® Xeon® CPU@2.20GHz
No. of cores	1	1	2
CPU cache	46080 KB	36608 KB	56320 KB
Encryption standard AES-NI	on	on	on
HD size	29 GB (użyte 1,9GB)	32,9 GB (użyte 1,6 GB)	29 GB (użyte 2,6 GB)
RAM	967,9 MB	908,5 MB	968,1 MB
Operating system	Ubuntu 20.04.4 LTS	Ubuntu 20.04.4 LTS	Ubuntu 20.04.4 LTS
CPU architecture	x86_64 (64 bit)	x86_64 (64 bit)	x86_64 (64 bit)
Linux kernel	5.13.0-1022-aws	5.13.0-1023-azure	5.13.0-1024-gcp
Networking - TCP	TCP CC: cubic	TCP CC: cubic	TCP CC: cubic
VM standard	Xen-DomU	Microsoft Virtual Machine	KVM
Location	Frankfurt am Main Region: Hesse	Frankfurt am Main Region: Hesse	Frankfurt am Main Region: Hesse

The processors are of a similar class, but the machine provided by Microsoft Azure has the highest processor clock speed. However, in the case of GCP Compute Engine, the e2-micro machine can be launched for which Google offers 2 virtual processors and it should be emphasized that this is the lowest option. It should also be noted that each tested machine has the same operating system installed, but the load on the machine at the start of each cloud service provider differs, e.g., the disk usage in Amazon AWS is 1.9 GB, in the case of Google GCP it is already 2.6 GB, while in Microsoft Azure it is only 1.6 GB of the occupied space on the hard disk.

Figure 1 summarizes the results of I/O operation performance measurements for the hard disk for selected virtual machines,

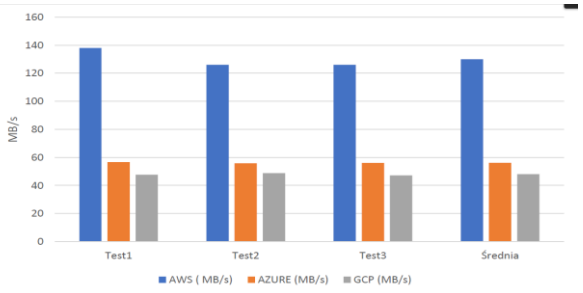


Figure 1: List of I/O operation parameters for disks on the three types of virtual machines.

Based on the obtained results, it can be concluded that the best parameters for reading data from the disk were obtained for a virtual machine running in the Amazon AWS cloud. The average of the obtained measurements was 130 MB/s. This means that it is 56.7% higher than the disk performance of a virtual machine running in the Microsoft Azure cloud and 63% higher than in the case of Google GCP. Disk performance is one of the key elements in a virtual machine, which affects the level of acceptance and comfort of work for each user, including beginners.

Figure 2 contains a summary of the speed measurement of the data download and upload along with the recorded delay values. The speedtest.net service was used in these tests.

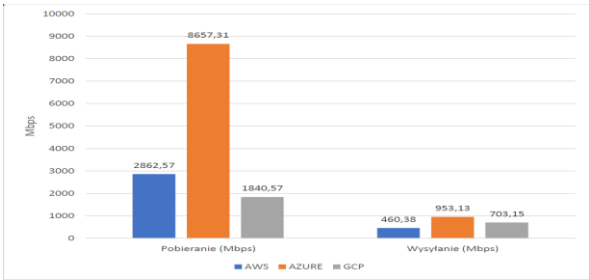


Figure 2: The results of network connection performance measurements.

The obtained results show that the virtual machine running in the Microsoft Azure cloud turned out to be the clear leader in terms of network connection parameters. The second place was taken by a virtual machine based on the Amazon AWS public cloud infrastructure, and the worst results were obtained for the Google cloud.

Another test concerned the measurements of the performance of individual virtual machines from the point of view of operations at the operating memory level. This test has been divided into 5 steps: add, copy, scale, triad and average, which provide a quantitative description of the performance of the process of reading and writing data blocks. Table 4 shows the results of the measurements obtained in this test.

Table 4: Results of operational memory performance measurements for selected virtual machines

Test	AWS (MB/s)	AZURE (MB/s)	GCP (MB/s)
Integers			
Add	9644,03	13141,87	1852,49
Copy	4221,35	12587,29	1610,35
Scale	1124,93	10198,95	1518,61
Triad	1072,1	12876,24	1570,3
Average	1128,44	12214,11	1565,42
Floating point			
Add	9430,72	11222,99	1834,53
Copy	8226,21	12742,88	1572,45
Scale	4238,46	9766,61	1534,29
Triad	992,47	12942,24	1598,51
Average	1038,32	11681,72	1582,43

In this test, the grade is better when the measured value is higher. The results of the operational memory performance measurements show that the best parameters are offered by a virtual machine running in the Microsoft Azure public cloud. At the same time, the virtual machine running in the Amazon AWS cloud showed much better parameters than the virtual machine running in the Google cloud.

3. Benchmarking of free database services

Analysis of the performance of database services is based on VM instances offered as part of free annual subscriptions, the so-called free tiers [12]. In addition, it was assumed that performance tests of subsequent database services are carried out based on a VM located in the same subnet as the database service. The database performance measurement was based on the Sysbench [7] program installed on Ubuntu 20.04 LTS amd64. This program measured the parameters of reading and writing to the database, using for this purpose data previously prepared especially for the needs of the presented comparative analysis. The use of Sysbench is illustrated by the following commands:

```
sysbench --db-driver=mysql --mysql-db=test \
--range_size=100 --table_size=10000 \
--tables=2 --threads=1 --events=0 \
--time=60 --rand-type=uniform \
/usr/share/sysbench/oltp_read_only.lua prepare
```

In turn, the command used to test the database is as follows:

```
sysbench /usr/share/sybench/oltp_read_write.lua
--mysql-db=test --threads=60 \
--db-driver=mysql -report-interval=1
```

3.1. Comparison of parameters of database services

The list of parameters of free database services offered by the analyzed cloud service providers is presented in Table 5.

Table 5: List of parameters of tested database services

Service provider	Service name	vCPU	RAM	HDD size
AWS	AMAZON RDS db.t3.micro	2	1 GB	20 GB
AZURE	Microsoft Azure for MySQL - B1s	1	2 GB	20 GB
GCP	Cloud SQL db-g1-small	1	1,7 GB	20 GB

When performing tests on database services, limitations in the number of possible simultaneous database connections were observed. Therefore, the test was limited to 60 simultaneous database connections. Table 6 summarizes the test results for the three analyzed cloud service providers. The test is divided into 4 parts (read, write, number of transactions per second and delay) and in each part 3 tests were performed, and the average value was calculated on their basis.

Regarding the average values for the process of reading data from a specific database, the best parameter values were obtained for the database service running on the Amazon AWS cloud. At the same time, the worst results were obtained for the case of tests for the database service working in the Microsoft Azure cloud.

Table 6: Test results of selected parameters of database services

	AWS	AZURE	GCP
READ [no. of queries]			
test1	62748	25984	56924
test2	63098	25200	58002
test3	59136	28392	55944
average	61660,67	26525,33	56956,67
WRITE [no. of queries]			
test1	17835	7345	16104
test2	17946	7115	16447
test3	16843	8005	15846
average	17541,33	7488,33	16132,33
TRANSACTIONS [no. of transactions/s]			
test 1	440,66	177,41	393,19
test 2	443,04	171,27	403,77
test 3	415,3	190,98	385,66
average	433	179,89	394,21
DELAY [ms]			
test 1	135,51	332,85	151
test 2	134,65	343,79	147,34
test 3	143,6	307,43	153,7
average	137,92	328,02	150,68

The analysis of the measured average values of writes to the database leads to similar conclusions as in the case of data reading tests. Based on the tests, it can be indicated that the best one was the database running in the Amazon AWS cloud, while the worst results were obtained for the database operating in the Microsoft Azure cloud infrastructure. On the other hand, the database operating as part of the Google GCP service was characterized by the results relatively not much worse than in the case of the Amazon AWS cloud, but at the same time the results indicate that data reading was 53.58% faster than in the case of the Microsoft Azure cloud.

In the next measurement, the number of transactions per second was examined, and similarly to the previous tests, the more transactions, the better the result. Based on the results, the Amazon AWS cloud service was found to be 8.96% more efficient than the GCP cloud service and 58.46% more efficient than the Microsoft Azure cloud.

4. Conclusion

The article presents the comparative analysis of selected services offered by the key cloud service providers.

The performance tests of the virtual machine lease services can be concluded that the most efficient offer was the machine running on the IaaS service implemented in the Microsoft Azure cloud. For this machine, better values of parameters were obtained than for a virtual machine running in the Amazon AWS cloud and Google GCP. Virtual machines based on the Google cloud infrastructure offered the worst parameters among all three tested solutions.

The performed tests of the database service showed the superiority of the Amazon AWS infrastructure, which offered the highest performance. At the same time, the results obtained in this group of tests allow pointing to the Microsoft Azure cloud as the one that offered the worst parameters. In addition, it should be added that the Google cloud database service ranked among the above-mentioned clouds, but in terms of prices it is comparable to the analogous AWS cloud service.

To summarize all the tests performed, a final evaluation scheme was adopted. This scheme is based on the award of points for individual scheme: 10 - the best, 5 - average, 0 - worst. Following the above procedure, the best service can be indicated. Table 7 presents the collected and subjectively assessed results of the analysis of the tested offers of cloud providers. They base on the results presented in previous chapters.

Table 7: The results of subjective analysis of selected services

	AWS	AZURE	GCP
Registration	10	5	0
Documentation of VMs	5	10	0
Database service documentation	5	10	0
Running a Virtual Machine	5	0	10
Use of a database service	10	0	5
Sum	**35**	**25**	**15**

As a result of the subjective assessment and after summing up the points awarded, the Amazon AWS cloud came out best, with a total of 35 points. The Microsoft Azure cloud was in second place.

Table 8 contains a summary list of points awarded based on the results of the performed performance tests of selected virtual machine instances.

Table 8: Virtual machine lease service evaluation results

	AWS	AZURE	GCP
Virtual machines: free tier offers	5	5	10
Costs without free tier	10	0	5
HDD I/O tests	10	5	0
Networking	5	10	0
RAM tests	5	10	0
Sum	**35**	**30**	**15**

In virtual machine performance tests, a virtual machine running in the Amazon AWS public cloud wins with a slight advantage. It received 35 points. The Microsoft Azure cloud received a total of 30 points and the lowest score was received by a virtual machine based on the Google cloud.

In the last statement, Table 9 presents the scores resulting from the performance tests of database services.

Table 9: Database services evaluation results

	AWS	AZURE	GCP
Specification	10	5	0
Costs	5	10	5
Tests of simultaneous connections	0	5	10
Read tests	10	0	5
Write tests	10	0	5
Transactions tests	10	0	5
Delay tests	10	0	5
Sum	**55**	**20**	**35**

After summing up the points for the performance tests of the selected database service, the database service in

Amazon AWS was the best. The lowest points were scored for a database service based on Microsoft's Azure cloud solutions.

Summarizing all the scores presented above, Amazon and its Amazon Web Services cloud services turned out to be the clear leader, followed by the Microsoft Azure public cloud and Google Cloud Platform. Both scored the same number of points

References

[1] B. S. Dordevic, S. P. Jovanovic, V. V. Timcenko, Cloud Computing in Amazon and Microsoft Azure platforms: Performance and service comparison, 22nd Telecommunications Forum Telfor (TELFOR) (2014) 931-934.

[2] C. D. Opara, Cloud Computing in Amazon Web Service, Microsoft Windows Azure, Google App Engine and IBM Cloud Platforms: A Comparative Study, master thesis, Near East University Nicosia, 2020.

[3] CAST AI, Cloud Pricing Comparison: AWS vs. Azure vs. Google Cloud Platform in 2022, https://faun.pub/cloud-pricing-comparison-aws-vs-azure-vs-google-cloud-platform-in-2022-3dc402e3edff [16.04.2022].

[4] P. Mell, T. Grance, The NIST Definition of Cloud Computing, Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, (2011) MD 20899-8930.

[5] Script for Virtual machine benchmark, https://github.com/teddysun/across/blob/master/bench.sh [20.3.2022]

[6] Open-Source, Automated Benchmarking, http://www.phoronix-test-suite.com/ [02.02.2022]

[7] Sysbench, https://wiki.gentoo.org/wiki/Sysbench [05.02.2022]

[8] M. J. Kavis, Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS), John Wiley & Sons, Inc, Hoboken, New Jersey, 2014.

[9] V. Kantsev, Implementing DevOps on AWS, Packt Publishing Ltd., 2017.

[10] T. Vijayakumar, Practical Azure Application Development, Apress, 2017.

[11] M. Suren, G. Suraj, DevOps for Azure Applications, Apress, 2018.

[12] D. Mauri, S. Coriani, A. Hoffman, S. Mishra, J. Popovic, Practical Azure SQL Database for Modern Developers. Building Applications in the Microsoft Cloud, Apress. 2020.

[13] G. Haff, How Open Source Ate Software. Understand the Open Source Movement and So Much More, Apress, 2018.

[14] M. A. Kamal, H. W. Raza, M. M. Alam, M. M. Su'ud, Highlight the Features of AWS, GCP and Microsoft Azure that Have an Impact when Choosing a Cloud Service Provider, International Journal of Recent Technology and Engineering (IJRTE) 8(5) (2020) 4124-4132.

[15] M. Dewangan, R. K. Deshmukh, A. Mishra, Comparative Study Between Existing Cloud Service Providers. International Journal of Advanced Research in Computer Science 9(2) (2018) 537-539.

[16] S. Mohapatra, S. Mohanty, S. Pattanayak, A. Hota, Comparison of various platforms in cloud computing. International Journal of Computer Applications 162(7) (2018) 28-33.

[17] I. Odun-Ayo, A. Falade, V. Samuel, Cloud Computing and Open Source Software: Issues and Development, Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS I (2018) 14-16.

[18] I. Voras, B. Mihaljevic, M. Orlić, Criteria for evaluation of open source cloud computing solutions. IEEE eXplore 7 (2011) 137 – 142.

[19] K. Mahesh, M. Laxmaiah, Y. K. Sharma, A Comparative Study on Google App Engine, Amazon Web Service and Microsoft Windows Azure, International Journal of Computer Engineering & Technology IJCET 10(1) (2019), 54-60.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 242–249
Received: 13 June 2022
Accepted: 6 July 2022

Comparative analysis of reactive and imperative approach in Java web application development

Analiza porównawcza podejścia reaktywnego i imperatywnego w tworzeniu aplikacji internetowych w języku Java

Sebastian Iwanowski*, Grzegorz Kozieł

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

Abstract
The subject of this study was to compare web applications created using the imperative and reactive approaches in Java. For this purpose, two applications with the same functionalities were developed using both approaches. The study looked at the performance, stability and time-consumption of implementation of each application. Based on the obtained results, it was found that the reactive application processes queries faster, uses less CPU, and is more stable in the case of handling many simultaneous requests, where the processing time is greater than 10 seconds. No significant differences were observed in using the computer's RAM by the applications. In addition, the study showed that reactive application takes more time to create.

Keywords: imperative approach; reactive approach; web applications; Java

Streszczenie
Celem niniejszej pracy było porównanie aplikacji internetowych tworzonych przy pomocy podejścia imperatywnego oraz reaktywnego w języku Java. Do tego celu stworzono dwie aplikacje z takimi samymi funkcjonalnościami używając obu podejść. Badanie dotyczyło wydajności, stabilności oraz czasochłonności implementacji każdej z aplikacji. Na podstawie uzyskanych wyników, stwierdzono, że aplikacja reaktywna szybciej przetwarza zapytania, w mniejszym stopniu obciąża procesor oraz jest stabilniejsza w przypadku obsługi wielu jednoczesnych żądań, gdzie czas przetworzenia jest większy niż 10 sekund. W przypadku wykorzystania ilości pamięci RAM przez aplikacje nie zaobserwowano znaczących różnic. Ponadto badanie pokazało, że stworzenie aplikacji reaktywnej jest bardziej czasochłonne.

Słowa kluczowe: podejście imperatywne; podejście reaktywne; aplikacje internetowe; Java

*Corresponding author
Email address: **sebastian.iwanowski@pollub.edu.pl** (S.Iwanowski)

1. Wstęp

Aplikacje internetowe są obecnie jednymi z podstawowych form dostarczania usług. Aplikacje takie coraz częściej tworzone są z myślą o dużej liczbie odbiorców, a co za tym idzie z ukierunkowaniem na większy zysk. Programiści muszą systematycznie mierzyć się z zadaniem doboru odpowiednich technologii i narzędzi, aby jak najlepiej sprostać oczekiwaniom narzuconym im przez klientów. Niejednokrotnie dobór nieadekwatnych rozwiązań, może wiązać się z dużymi kosztami, będącymi skutkiem konieczności przebudowy części lub, co gorsza, całej aplikacji.

Jedną z popularniejszych obecnie technologii wybieranych do tworzenia oprogramowania internetowego jest język Java, oraz tworzony i rozwijany od wielu lat, wzorzec programistyczny Spring Framework. Szkielet ten, pozwala w stosunkowo prosty i szybki sposób, tworzyć różnego rodzaju aplikacje internetowe. Z racji tego, że wzorzec ten jest bardzo rozbudowany, programista korzystający z niego musi wybrać odpowiednie biblioteki, które decydować będą o tym jak działać będzie w przyszłości dany produkt. Głównymi bibliotekami w przypadku aplikacji internetowych są Spring MVC oraz Spring WebFlux, będącymi odpowiednikami imperatywnego i reaktywnego stosu technologicznego. Oba wybory mają zalety i wady. Aplikacje tworzone w oparciu o drugą z wymienionych bibliotek, jak opisują sami autorzy, powinny być głównie przeznaczone dla oprogramowania, które posiadać będzie znaczące opóźnienia, np. powolne operacje wejścia/wyjścia lub dla programów, które muszą zachować wysoką stabilność. Dla klasycznych aplikacji, nie posiadających wymienionych wcześniej cech, zalecają pozostanie przy Spring MVC, który jest prostszy i szybszy w zrozumieniu niż jej reaktywna alternatywa [1].

Twórcy obu bibliotek nie udostępniają wielu porównań wydajnościowych obu produktów. Jeżeli różnica w szybkości czy też stabilności okazałaby się niewielka, kosztem wydłużenia czasu wytwarzania oprogramowania, spowodowanym wyborem trudniejszej technologii, to należałoby przeprowadzić analizę czy byłoby to opłacalne zarówno dla programistów, którzy musieliby najprawdopodobniej przeznaczyć dodatkowy czas na pracę z oprogramowaniem oraz dla klientów którzy otrzymali by nieznacznie lepszą aplikację w zamian za dłuższy czas oczekiwania.

Celem artykułu jest potwierdzenie tezy: aplikacje reaktywne są wydajniejsze i stabilniejsze niż aplikacje imperatywne w przypadku jednoczesnej obsługi wielu zapytań. Dodatkowo zdefiniowano cztery hipotezy:

1. Zapytania w aplikacjach reaktywnych są przetwarzane szybciej niż w aplikacjach imperatywnych w przypadku obsługi wielu użytkowników jednocześnie,
2. Aplikacje reaktywne wykorzystują mniej zasobów procesora oraz pamięci RAM komputera,
3. Aplikacje reaktywne są bardziej stabilne i posiadają mniejszy procent nieobsłużonych zapytań w przypadku większej liczby zapytań,
4. Stworzenie aplikacji reaktywnej jest bardziej czasochłonne niż stworzenie aplikacji imperatywnej.

2. Analiza literatury

Analiza literatury wykazała niewielką popularność badań dotyczących podejścia reaktywnego na przykładzie języka Java. Dodatkowo, przeprowadzony przegląd nie wykazał, żadnej publikacji, która porównywała by podejście imperatywne oraz reaktywne w języku Java, zarówno pod względem wydajności jak i czasochłonności implementacji.

W artykule [2] autorzy porównują paradygmat reaktywny z tradycyjnym paradygmatem obiektowym, w odniesieniu do zrozumienia programu. Swoje badania przeprowadzili na 127 osobach podzielonych na 2 grupy, które miały do wykonania 10 zadań i ankietę dotyczącą aplikacji napisanych zarówno przy użyciu podejścia reaktywnego jak i obiektowego. Eksperyment brał pod uwagę takie czynniki jak, poprawność zrozumienia, wymagany czas oraz poziom umiejętności jaki był niezbędny do prawidłowej odpowiedzi. Twórcy wykazali, że aplikacje oparte o paradygmat reaktywny są prostsze w zrozumieniu, a dodatkowo poziom umiejętności programistycznych jaki wymagany jest do prawidłowej analizy był mniejszy niż w przypadku aplikacji opartych o paradygmat obiektowy. Autorzy dodają również, że przyczynami które wpływają na taki rezultat mogą być zmniejszona powtarzalność kodu oraz jego lepsza czytelność.

Artykuł [3] przedstawia porównanie aplikacji REST (Representational State Transfer) stworzonych w technologii Spring Boot oraz MS.NET. Badane programy posiadały identyczne funkcjonalności, odpowiadające czterem podstawowym operacjom: tworzenia, odczytywania, aktualizowania oraz usuwania (CRUD). Badania zostały przeprowadzone z wykorzystaniem narzędzia Apache JMeter i uwzględniały średnie czasy odpowiedzi, liczbę błędów w aplikacjach oraz wykorzystanie zasobów CPU oraz pamięci komputera. Na podstawie uzyskanych wyników, autorzy stwierdzili, że aplikacje oparte o technologię MS.NET zapewniają szybszy czas odpowiedzi, niż aplikacje stworzone w Spring Boot w każdym z testowanych scenariuszy. Twórcy zaobserwowali również, że program ten wykorzystuję mniej zasobów CPU oraz pamięci komputera. Ponadto, na podstawie analizy liczby błędów, twórcy uznali, że obie aplikacje są równie stabilne.

Autorzy artykułu [4] przedstawiają rezultaty badań dotyczących analizy porównawczej dwóch wielowątkowych paradygmatów: programowania reaktywnego oraz stylu kontynuacji-przejścia. Kryteriami jakimi posłużyli się twórcy w celu przeprowadzenia badania były utrzymywalność, wydajność oraz testowalność. Badanie zostało przeprowadzone na zaimplementowanych w oparciu o wymienione wyżej paradygmaty aplikacjach pozwalających na kompresję plików wideo. Twórcy wykorzystali do tego biblioteki RxJava oraz Kotlin Coroutines. Na podstawie uzyskanych wyników, twórcy stwierdzili, że obie technologie osiągają podobne wyniki, jeśli chodzi o wydajność oraz testowalność, lecz istnieją znaczące różnice pod względem utrzymywalności. Z przeprowadzonego przez nich badania, korzystającego z metryk Halsteada, wywnioskowali oni również, że aplikacja oparta na metodzie Coroutines jest łatwiejsza w zaimplementowaniu niż aplikacja oparta na RxJava. Zaznaczają, że jest to najprawdopobodniej spowodowane faktem, że będący przedmiotem badań styl kontynuacji-przejścia przypomina tradycyjny, imperatywny styl programowania.

3. Omówienie pojęć

3.1. Aplikacje imperatywne

Aplikacja imperatywna jest to program, którego kod wykonuje się w sposób sekwencyjny. Oznacza to, że każda linia kodu może się wykonać, jedynie po tym jak poprzednia instrukcja została zakończona. Idea w programach imperatywnych, opiera się więc na wstrzymywaniu wykonywania się kolejnych instrukcji.

W niniejszej pracy, program imperatywny zbudowany został w oparciu o wzorzec programistyczny Spring Boot i Spring MVC. Korzysta on z domyślnego dla tych bibliotek serwera – Tomcat [5]. Serwer ten jest przykładem serwletu, który został napisany z myślą o obsłudze kodu imperatywnego. Jednym z podstawowych schematów jego działania jest tworzenie wątku dla każdego odebranego żądania. Domyślnie jego maksymalna pula wątków wynosi 200, co oznacza, że maksymalnie jest w stanie obsłużyć 200 jednoczesnych zapytań, po przekroczeniu tego limitu, serwer zaczyna odrzucać lub kolejkować inne żądania. Serwer imperatywny jest serwerem blokującym co oznacza, że w momencie przypisania wątkowi sekwencji instrukcji do wykonania, jest on blokowany aż do czasu zakończenia ostatniej z nich. Dodatkowo w przypadku, gdy dane żądanie będzie powiązane z operacją wejścia/wyjścia np. zapis do pliku, i będzie zajmować stosunkowo dużą ilość czasu. Wątek po zleceniu wykonania tej czynności jest przez pewien czas niedostępny, pomimo, ze nie wykonuje praktycznie żadnej operacji [6].

3.2. Aplikacje reaktywne

Aplikacja reaktywna to aplikacja, która używa asynchronicznego podejścia do wykonywania szeregu instrukcji. Głównym założeniem programów reaktywnych jest to, aby sekwencja wykonywania się kolejnych bloków kodu była od siebie jak najbardziej niezależna oraz aby w żadnym momencie nie nastąpiło zablokowanie wątku odpowiedzialnego za jej wykonanie. W odróżnieniu od aplikacji imperatywnych, aplikacje reaktywne nie mają jednoznacznie określonej kolejności wykonywania się instrukcji.

Serwerem użytym w aplikacji reaktywnej, będącej celem badań pracy jest Netty, który został skonfigurowany i dostarczony przez wzorce projektowe Spring Boot oraz Spring WebFlux. Netty zbudowany został na potrzeby aplikacji reaktywnych, potrzebujących zarówno asynchroniczności jak i nieblokowanego przepływu instrukcji [7]. W odróżnieniu od serwera imperatywnego, jego schemat działania nie jest oparty na przypisywaniu żądań do konkretnych wątków, a jak najbardziej optymalnym zarządzaniu stosunkowo małą liczbą wątków. Serwer ten tworzy pętle zdarzeń, która odpowiedzialna jest za odbieranie i przesyłanie żądań. Pętla ta sprawdza, który wątek jest dostępny, a następnie oddelegowuje do niego odebrane zapytanie, przechodząc tym samym do dalszego obsługiwania serwera [8]. W przypadku opisywanego serwera, liczba pętli zależna jest od liczby dostępnych rdzeni procesora, zazwyczaj jest to jedna lub dwie pętle na rdzeń [9].

4. Metodyka badań

Przedmiotem badań jest porównanie wydajności, stabilności oraz czasochłonności implementacji aplikacji imperatywnej oraz reaktywnej w takich samych środowiskach oraz przy użyciu tych samych próbek badawczych. Do tego celu stworzone zostały dwie aplikacje napisane w wymienionych wyżej technikach programowania, posiadające te same funkcjonalności. Zbadane zostały czasy przetwarzania zapytań, wykorzystanie zasobów środowiska oraz sprawdzone zostało ile zapytań uda się poprawnie obsłużyć. Ponadto zliczono linie kodu, które były niezbędne do stworzenia każdej z aplikacji, w celu wyznaczenia czasochłonności implementacji takich projektów.

Porównanie zostało przeprowadzone poprzez zmierzenie czasu jaki będzie potrzebny każdej z aplikacji na przetworzenie określonej liczby żądań HTTP. Funkcjonalnościami które były wykorzystywane w celu przeprowadzenia pomiarów są:

- pobranie listy wszystkich pracowników z bazy danych,
- pobranie pojedynczego pracownika z bazy,
- aktualizacja pojedynczego pracownika w bazie danych,
- tworzenie nowego pracownika w bazie,
- pobranie elementu z opóźnieniem serwera równym 10 sekund.

Dla każdej z funkcjonalności badanie zostało powtórzone 50-krotnie w celu uśrednienia otrzymanych wyników. Próby przeprowadzone zostały w takich samych warunkach, dla 1000 elementów w bazie danych oraz odpowiednio 30, 300, 2000 jednoczesnych zapytań do aplikacji, w celu symulacji małej, średniej i dużej liczby zapytań. Dodatkowo przeprowadzono analizę liczby linii kodu obu aplikacji, co było miernikiem czasochłonności tworzenia kodu przy wykorzystaniu danej metody. Przeprowadzone badania zostały powtórzone dla aplikacji o dostępnych do wykorzystania zasobach:

- 2 rdzenie procesora, 4GB RAM,
- 4 rdzenie procesora, 8GB RAM.

Badania zostały przeprowadzone z wykorzystaniem dwóch maszyn o podanych w tabeli 1 parametrach.

Tabela 1: Parametry środowisk testowych

	Maszyna 1	**Maszyna 2**
Rodzaj urządzenia	Laptop	Komputer stacjonarny
Procesor	Intel Core i7 7700HQ, 2.8GHz	Intel Core i7 12700K,3.6GHz
Pamięć RAM	8GB DDR4	32GB DDR4
Dysk	SSD 256GB, interfejs IDE	SSD 2TB, interfejs NVMe
System operacyjny	Windows 10 Home 64-bit	Windows 10 Pro 64-bit

Obie maszyny posiadały zainstalowane JDK w wersji 11 oraz podłączone były do tej samej sieci o szybkości łącza 100MB/s. Maszyna 1 odpowiadała za uruchomienie skryptów badających szybkości odpowiedzi aplikacji oraz zapisanie tych danych w postaci plików HTML. Z kolei maszyna 2 pełniła rolę serwera dla obu aplikacji. Utrzymywała zarówno instancję aplikacji oraz przypisaną do niej bazę danych, jak i narzędzia Prometheus i Grafana, służące do pomiarów wykorzystania zasobów sprzętowych.

5. Wyniki badań

5.1. Czasy odpowiedzi aplikacji

Przeprowadzone badania pozwoliły wyznaczyć średnie czasy odpowiedzi oraz stosunek liczby poprawnie do niepoprawnie obsłużonych zapytań, który oznaczony jest w tabelach jako „OK/KO”. W przypadku każdej z funkcjonalności, średnie czasy odpowiedzi zostały wyliczone uwzględniając jedynie poprawnie obsłużone zapytania.

5.1.1. Ograniczenie 2CPU, 4GB pamięci RAM

W tabelach 2 oraz 3 zawarte zostały wyniki dotyczące aplikacji ograniczonych do 2 rdzeni procesora oraz 4GB pamięci RAM w środowisku Docker Desktop.

Tabela 2: Uśrednione wyniki pomiarów czasu odpowiedzi aplikacji imperatywnej ograniczonej do 2CPU, 4GB RAM

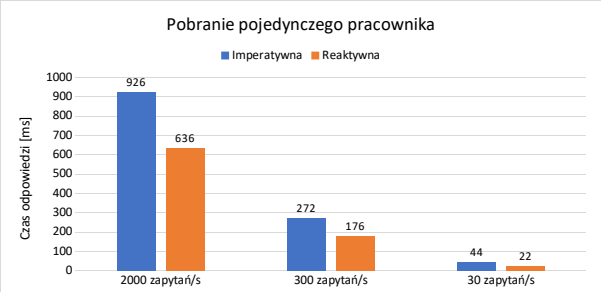
	Aplikacja imperatywna					
	Średni czas odpowiedzi [ms]			**OK/KO [%]**		
Liczba jednoczesnych zapytań	**2000**	**300**	**30**	**2000**	**300**	**30**
Pobranie pojedynczego pracownika	926	272	44	100	100	100
Pobranie wszystkich pracowników	7200	1110	148	96	100	100
Aktualizacja pojedynczego pracownika	1758	384	73	100	100	100
Utworzenie nowego pracownika	966	226	32	100	100	100
Pobranie elementu z opóźnieniem serwera równym 10 sekund	30494	13443	10198	50	100	100

Na podstawie Rysunków od 1 do 5, odczytać można, że w większości scenariuszy, aplikacja reaktywna osiągnęła lepsze wyniki. W przypadku wyników operacji pobierania pojedynczego pracownika przedstawionych na Rysunku 1, widać znaczącą różnicę pomiędzy programami. Aplikacja reaktywna jest w stanie zwrócić odpowiedź o 290ms szybciej niż aplikacja imperatywna w przypadku 2000 zapytań, co stanowi 31% wzrost szybkości.

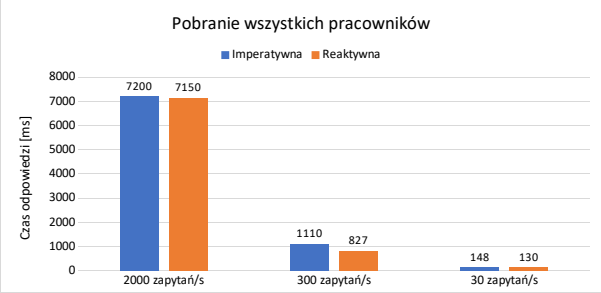
Tabela 3: Uśrednione wyniki pomiarów czasu odpowiedzi aplikacji reaktywnej ograniczonej do 2CPU, 4GB RAM

	Aplikacja reaktywna					
	Średni czas odpowiedzi [ms]			OK/KO [%]		
Liczba jednoczesnych zapytań	**2000**	**300**	**30**	**2000**	**300**	**30**
Pobranie pojedynczego pracownika	636	176	22	100	100	100
Pobranie wszystkich pracowników	7150	827	130	98	100	100
Aktualizacja pojedynczego pracownika	788	171	26	100	100	100
Utworzenie nowego pracownika	719	211	44	100	100	100
Pobranie elementu z opóźnieniem serwera równym 10 sekund	11296	10137	10136	100	100	100

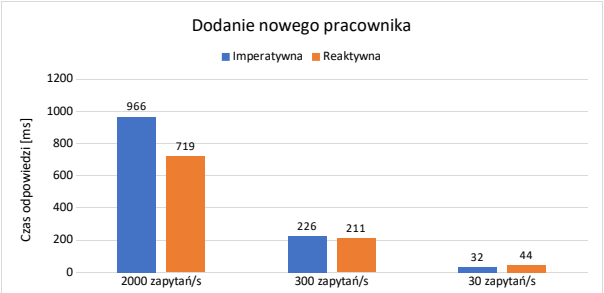
Wykres z Rysunku 2, dotyczący pobierania wszystkich użytkowników, również pokazuje, że stos reaktywny jest szybszy niż imperatywny, lecz w tym przypadku różnica ta jest mniej widoczna. Dla 2000 zapytań, wynosi ona 50 ms, co daje wynik lepszy o niecały procent. W przypadku 300 i 30 zapytań, aplikacja reaktywna jest szybsza o odpowiednio 25% i 12% w stosunku do aplikacji imperatywnej. Wykres z Rysunku 5 przedstawia porównanie czasów w przypadku pobrania elementu, przy którym serwer czeka 10 sekund na jego odesłanie, symulując czasochłonne operacje typu operacje wejścia/wyjścia. W tym przypadku widoczna jest największa różnica, jeśli chodzi o wysokie obciążenie serwera – 2000 zapytań. Aplikacja reaktywna jest w stanie obsłużyć zapytania niemal 3 razy szybciej niż jej imperatywny odpowiednik. Dodatkowo, tak jak widoczne to jest w Tabelach 2 oraz 3, serwer imperatywny odrzucił 50% zapytań, gdzie serwer reaktywny był w stanie obsłużyć każde z nich. W przypadku 300 i 30 zapytań obie aplikacje wypadły podobnie, lecz wyniki nadal świadczą na korzyść stosu reaktywnego.



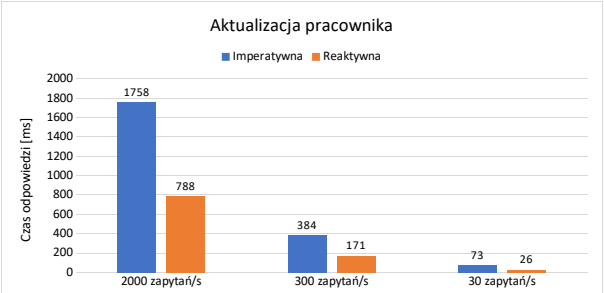
Rysunek 1: Średnie czasy pobrania danych pojedynczego pracownika przy ograniczeniu 2CPU, 4GB RAM.



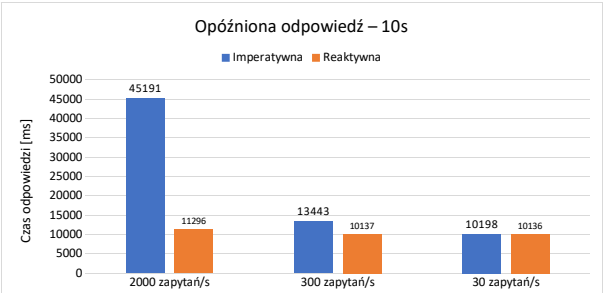
Rysunek 2: Średnie czasy pobrania danych wszystkich pracowników przy ograniczeniu 2CPU, 4GB RAM.



Rysunek 3: Średnie czasy dodania nowego pracownika przy ograniczeniu 2CPU, 4GB RAM.



Rysunek 4: Średnie czasy aktualizacja danych pracownika przy ograniczeniu 2CPU, 4GB RAM.



Rysunek 5: Średnie czasy odpowiedzi przy pobieraniu elementu opóźnionego o 10 sekund przy ograniczeniu 2CPU, 4GB RAM.

5.1.2. Ograniczenie 4CPU, 8GB pamięci RAM

Tabele 3 oraz 4 przedstawiają rezultaty badań dotyczące aplikacji ograniczonych do 4 rdzeni procesora oraz 8GB RAM, tak samo jak w poprzednim wariancie, w środowisku Docker Desktop.

Tak jak widoczne to jest na rysunkach od 6 do 10, w przypadku środowiska z dostępnymi zasobami: 4 rdzenie procesora oraz 8 GB RAM, czasy odpowiedzi również okazały się krótsze dla aplikacji reaktywnych w przypadku większości funkcjonalności.

Na podstawie rysunku 6 można odczytać, że wzrost szybkości odpowiedzi serwera reaktywnego w przypadku obciążenia 2 tysięcy zapytań dla funkcjonalności pobrania danych pojedynczego pracownika wynosi 33%, czyli 2% szybciej niż w przypadku wyniku dla poprzedniego ograniczenia. Na rysunku 7 widoczne jest, że operacja pobierania danych wszystkich pracowników z bazy danych zajmuje więcej czasu w przypadku zastosowania stosu reaktywnego. W przypadku 2000 oraz 300 zapytań, aplikacja imperatywna okazała się szybsza o odpowiednio 7% i 2%. W przypadku 30 zapytań, wynik świadczy jednak o przewadze programu reaktywnego.

Tabela 3: Uśrednione wyniki pomiarów czasu odpowiedzi aplikacji imperatywnej ograniczonej do 4CPU, 8GB RAM

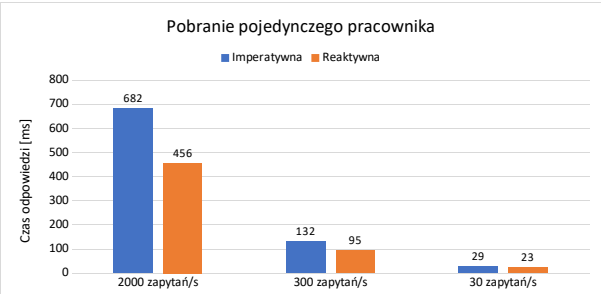
	Aplikacja imperatywna					
	Średni czas odpowiedzi [ms]			OK/KO [%]		
Liczba jednoczesnych zapytań	2000	300	30	2000	300	30
Pobranie pojedynczego pracownika	682	132	29	100	100	100
Pobranie wszystkich pracowników	5720	831	163	96	100	100
Aktualizacja pojedynczego pracownika	874	200	34	100	100	100
Utworzenie nowego pracownika	583	122	29	100	100	100
Pobranie elementu z opóźnieniem serwera równym 10 sekund	30485	13487	10063	51	100	100

Tabela 4: Uśrednione wyniki pomiarów czasu odpowiedzi aplikacji reaktywnej ograniczonej do 4CPU, 8GB RAM

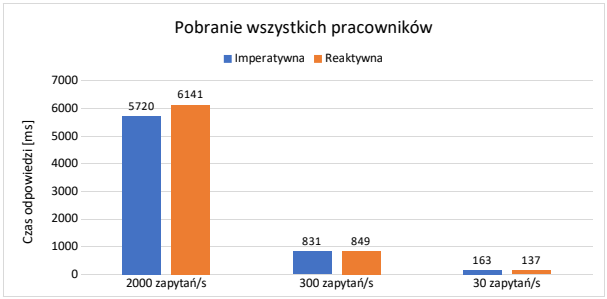
	Aplikacja reaktywna					
	Średni czas odpowiedzi [ms]			OK/KO [%]		
Liczba jednoczesnych zapytań	2000	300	30	2000	300	30
Pobranie pojedynczego pracownika	456	95	23	100	100	100
Pobranie wszystkich pracowników	6141	849	137	97	100	100
Aktualizacja pojedynczego pracownika	531	128	23	100	100	100
Utworzenie nowego pracownika	107	94	25	100	100	100
Pobranie elementu z opóźnieniem serwera równym 10 sekund	11394	10139	10131	100	100	100

Dla funkcjonalności przedstawionych na Rysunkach 8 oraz 9, różnica jest najbardziej widoczna w scenariuszach z 2000 zapytaniami na sekundę, gdzie w przypadku dodania nowego pracownika, wynik ten jest ponad pięciokrotnie lepszy dla aplikacji reaktywnej. Dla 300 i 30 zapytań, rezultaty są lepsze średnio o około 30% dla aktualizacji i o 19% w wypadku dodawania nowego pracownika do bazy, również na rzecz stosu reaktywnego.

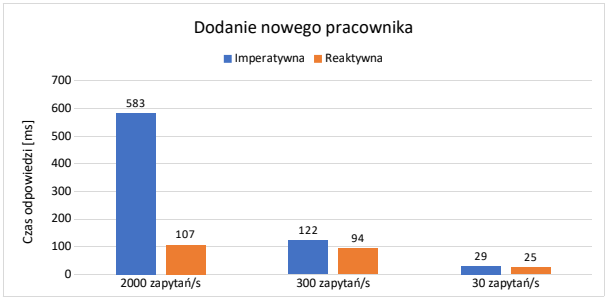
Analizując wykres przedstawiony na Rysunku 10, można dostrzec identyczną sytuację jak w poprzednim wariancie, aplikacja reaktywna uzyskała niemal 3 razy lepszy wynik w przypadku wariantu z 2000 zapytań.



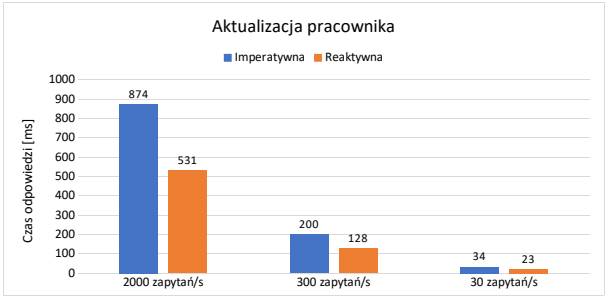
Rysunek 6: Średnie czasy pobrania danych pojedynczego pracownika przy ograniczeniu 4CPU, 8GB RAM.



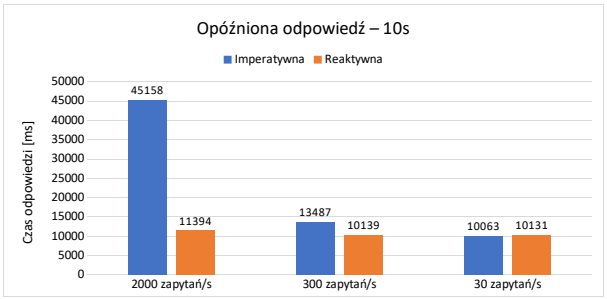
Rysunek 7: Średnie czasy pobrania danych wszystkich pracowników przy ograniczeniu 4CPU, 8GB RAM.



Rysunek 8: Średnie czasy dodania nowego pracownika przy ograniczeniu 4CPU, 8GB RAM.



Rysunek 9: Średnie czasy aktualizacja danych pracownika przy ograniczeniu 4CPU, 8GB RAM.



Rysunek 10: Średnie czasy odpowiedzi przy pobieraniu elementu opóźnionego o 10 sekund przy ograniczeniu 4CPU, 8GB RAM.

5.2. Wykorzystanie zasobów sprzętowych

5.2.1. Ograniczenie 2CPU, 4GB pamięci RAM

Tabela 5 oraz Tabela 6 przedstawia średnie zużycie zasobów sprzętowych przez testowane aplikacje ograniczone do 2 rdzeni procesora oraz 4GB RAM.

Na podstawie analizy obu tabel powiedzieć można, że aplikacja reaktywna wykorzystuje moc procesora w znacznie mniejszym stopniu niż aplikacja imperatywna. Różnice sięgają od 7%, w przypadku dodawania nowego pracownika przy 30 zapytaniach na sekundę, do

nawet 31%, w momencie pobierania danych wszystkich pracowników z bazy.

Na wykresie przedstawionym na Rysunku 11 widoczne jest, że średnie wykorzystanie procesora ze wszystkich scenariuszy wynosi 52.8% dla aplikacji imperatywnej, a 33.7% dla aplikacji reaktywnej, co stanowi ponad 36% różnicę w stosunku do programu imperatywnego. Jedynie w przypadku ostatniej funkcjonalności w tabelach widoczne jest, że aplikacja reaktywna wykorzystuję nieznacznie więcej dostępnych zasobów procesora niż jej imperatywny odpowiednik. W tym przypadku spowodowano jest to głównie tym, że aplikacja imperatywna odrzuca 50% zapytań, podczas gdy reaktywna obsługuje każde z nich. Widać, że dla 300 i 30 zapytań różnica wynosi jedynie 1%. Jedynie przypadek 2000 zapytań wykazuje 4% mniejsze wykorzystanie CPU przez program imperatywny.

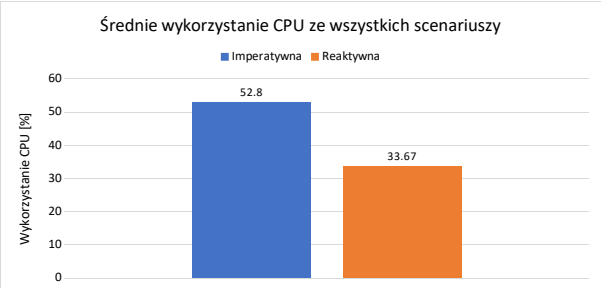
Analizując zużycie pamięci RAM obu aplikacji trudno jednoznacznie określić, która z nich zużywa jej mniej. Oba programy uzyskały podobne wyniki jeśli chodzi o średnią ze wszystkich scenariuszy, rysunek 12 pokazuje, że wynik aplikacji imperatywnej to 11%, a reaktywnej 10%. W przypadku scenariusza dotyczącego pobrania danych pojedynczego pracownika, zaobserwować można, że aplikacja reaktywna zużywa mniej pamięci dla 2000 i 300 zapytań, a aplikacja imperatywna dla 30 zapytań/s. Różnice w pozostałych scenariuszach są niewielkie, często zaledwie 1 procentowe.

Tabela 5: Uśrednione wyniki pomiarów wykorzystania zasobów komputera przez aplikację imperatywną ograniczoną do 2CPU, 4GB RAM

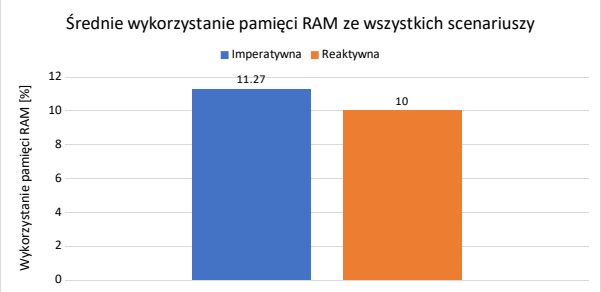
	Aplikacja imperatywna					
	Wykorzystanie CPU [%]			Wykorzystanie pamięci RAM [%]		
Liczba jednoczesnych zapytań	**2000**	**300**	**30**	**2000**	**300**	**30**
Pobranie pojedynczego pracownika	82	86	29	22	11	5
Pobranie wszystkich pracowników	62	63	31	32	13	7
Aktualizacja pojedynczego pracownika	94	92	61	16	7	6
Utworzenie nowego pracownika	83	68	35	11	12	5
Pobranie elementu z opóźnieniem serwera równym 10 sekund	3	2	1	9	7	6

Tabela 6: Uśrednione wyniki pomiarów wykorzystania zasobów komputera przez aplikację reaktywną ograniczoną do 2CPU, 4GB RAM

	Aplikacja reaktywna					
	Wykorzystanie CPU [%]			Wykorzystanie pamięci RAM [%]		
Liczba jednoczesnych zapytań	**2000**	**300**	**30**	**2000**	**300**	**30**
Pobranie pojedynczego pracownika	55	55	12	12	6	9
Pobranie wszystkich pracowników	31	55	18	31	13	5
Aktualizacja pojedynczego pracownika	57	68	10	11	7	6
Utworzenie nowego pracownika	57	47	28	13	8	6
Pobranie elementu z opóźnieniem serwera równym 10 sekund	7	3	2	9	7	7



Rysunek 11: Średnie wykorzystanie CPU ze wszystkich scenariuszy przy ograniczeniu 2CPU, 4GB RAM.



Rysunek 12: Średnie wykorzystanie pamięci RAM ze wszystkich scenariuszy przy ograniczeniu 2CPU, 4GB RAM.

5.2.2. Ograniczenie 4CPU, 8GB pamięci RAM

W Tabelach 7 i 8 przedstawione zostały wyniki badań dotyczących aplikacji ograniczonych do 4 rdzeni procesora oraz 8GB pamięci RAM.

Tabele te przedstawiają analogiczną sytuację jak w przypadku poprzedniego ograniczenia. Aplikacja reaktywna w mniejszym stopniu obciąża CPU we wszystkich, poza ostatnim scenariuszem dotyczącym pobierania elementu z opóźnieniem. Najmniejsza różnica w wykorzystaniu procesora wynosi jednak tutaj 2% dla przypadku dodawania nowego użytkownika podczas obciążenia 30 zapytań/s, a największa równa jest 36% w przypadku aktualizacji danych użytkownika dla 2000 zapytań. W przypadku funkcjonalności pobierania elementu z opóźnieniem, sytuacja jest jednakowa jak poprzednio, aplikacja reaktywna wykorzystuje nieznacznie więcej zasobów w zamian oferując lepszą stabilność aplikacji.

Z Rysunku 13 odczytać można, że średnie wykorzystanie procesora wynosi 33.6% dla aplikacji imperatywnej, a 24.3% dla aplikacji reaktywnej, co stanowi 27% spadek zużycia w stosunku do programu imperatywnego.

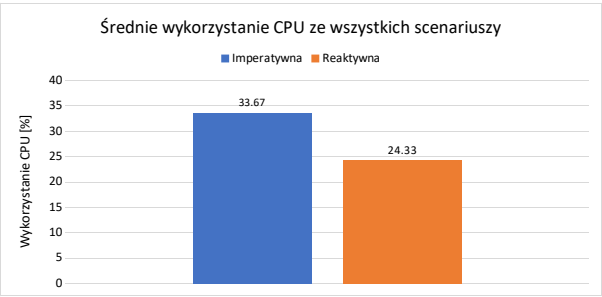
Tak jak w poprzednim przypadku zużycie pamięci RAM jest tutaj bardzo podobne dla obu aplikacji, nie widać jednak różnicy w przypadku funkcjonalności pobierania danych pojedynczego pracownika. Średnie wartości ze wszystkich scenariuszy przedstawione zostały na Rysunku 14 i wynoszą odpowiednio 6.6% oraz 6.4% dla aplikacji imperatywnej i reaktywnej.

Tabela 7: Uśrednione wyniki pomiarów wykorzystania zasobów komputera przez aplikację imperatywną ograniczoną do 4CPU, 8GB RAM

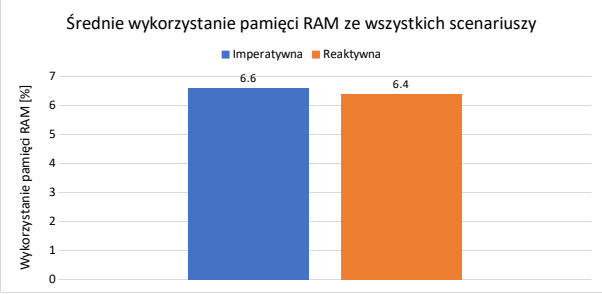
	Aplikacja imperatywna					
	Wykorzystanie CPU [%]			Wykorzystanie pamięci RAM [%]		
Liczba jednoczesnych zapytań	2000	300	30	2000	300	30
Pobranie pojedynczego pracownika	54	43	18	9	5	6
Pobranie wszystkich pracowników	41	40	37	15	6	6
Aktualizacja pojedynczego pracownika	87	69	17	8	5	5
Utworzenie nowego pracownika	53	28	12	6	8	4
Pobranie elementu z opóźnieniem serwera równym 10 sekund	2	2	2	6	5	5

Tabela 8: Uśrednione wyniki pomiarów wykorzystania zasobów komputera przez aplikację reaktywną ograniczoną do 4CPU, 8GB RAM

	Aplikacja reaktywna					
	Wykorzystanie CPU [%]			Wykorzystanie pamięci RAM [%]		
Liczba jednoczesnych zapytań	2000	300	30	2000	300	30
Pobranie pojedynczego pracownika	47	27	13	10	6	4
Pobranie wszystkich pracowników	24	26	27	15	5	5
Aktualizacja pojedynczego pracownika	51	57	5	7	7	5
Utworzenie nowego pracownika	38	31	10	5	6	4
Pobranie elementu z opóźnieniem serwera równym 10 sekund	5	2	2	6	6	5



Rysunek 13: Średnie wykorzystanie CPU ze wszystkich scenariuszy przy ograniczeniu 4CPU, 8GB RAM.



Rysunek 14: Średnie wykorzystanie pamięci RAM ze wszystkich scenariuszy przy ograniczeniu 4CPU, 8GB RAM.

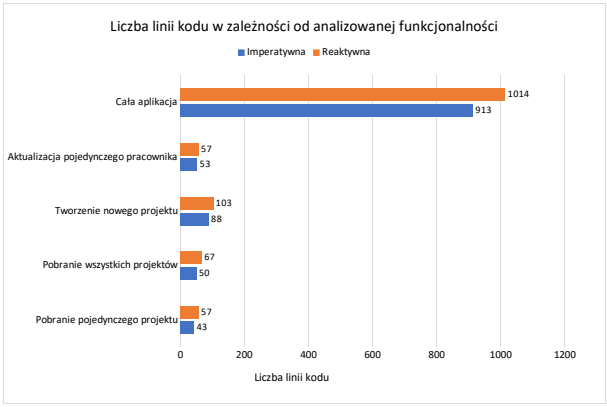
5.3. Czasochłonność implementacji

Czasochłonność implementacji została wyznaczona poprzez porównanie liczby linii kodu potrzebnej do stworzenia wybranych funkcjonalności oraz całej aplikacji dla każdego z programów. Zliczanie linii zostało przeprowadzone wyłączając puste linie oraz uwzględniając domyślne formatowanie dostarczone przez środowisko programistyczne IntelliJ IDEA. Tabela 9 pokazuje rezultaty tego badania.

Tabela 9: Wyniki pomiaru liczby linii potrzebnych do stworzenia wybranych funkcjonalności oraz całej aplikacji

	Aplikacja imperatywna	Aplikacja reaktywna
Pobranie pojedynczego projektu	43	57
Pobranie wszystkich projektów	50	67
Tworzenie nowego projektu	88	103
Aktualizacja pojedynczego projektu	53	57
Cała aplikacja	913	1014

Rysunek 15 przedstawia wizualne porównanie liczby linii kodu w zależności od analizowanej funkcjonalności. Ostatni diagram dotyczy całej aplikacji, z którego wynika, że różnica w ogólnej ilości kodu potrzebnej do stworzenia aplikacji reaktywnej wynosi o 10% więcej niż aplikacji imperatywnej. Na podstawie wykresu, można również stwierdzić, że aplikacja reaktywna wymaga większej ilości kodu do stworzenia takiej samej funkcjonalności co aplikacja imperatywna. Zależnie od funkcjonalności różnica ta wynosi od 4 do 17 linii kodu. Na podstawie otrzymanych w tym badaniu wyników, można stwierdzić, że stworzenie aplikacji reaktywnej wymaga więcej czasu niż stworzenie aplikacji imperatywnej. Wynika to głównie z braku odpowiedników dla uproszczeń i bibliotek używanych w stosie imperatywnym.



Rysunek 15: Liczba linii kodu w zależności od analizowanej funkcjonalności.

6. Podsumowanie i wnioski

W niniejszym artykule przedstawiono i omówiono wyniki badań dotyczących aplikacji internetowych tworzonych przy pomocy podejścia reaktywnego i imperatywnego w języku Java.

Na podstawie otrzymanych rezultatów można zauważyć kilka różnic w zachowaniu obu aplikacji. Pierwszą jest to, że aplikacje reaktywne okazały się szybsze nie tylko w przypadku obsługi wielu użytkowników jednocześnie, tak jak zakładano w pierwszej hipotezie, lecz w niemal każdym badanym w tej pracy scenariuszu. Znacząca różnica widoczna jest na przykładzie badania dotyczącego funkcjonalności pobierania elementu z 10 sekundowym opóźnieniem, gdzie stos reaktywny osiągnął prawie 3 razy szybszy czas odpowiedzi, względem aplikacji napisanej za pomocą stosu imperatywnego.

Seria badań, dotycząca porównania wydajności aplikacji, pod względem zużywanych zasobów komputera, wykazała, że obie aplikacje, niezależnie od dostępnych zasobów, potrzebują tyle samo pamięci RAM komputera do prawidłowego działania. W przypadku wykorzystania mocy obliczeniowej procesora, aplikacja imperatywna używała średnio 43.2%, gdzie aplikacja reaktywna wykorzystywała średnio jedynie 28.9%. Wynika z tego, że program reaktywny jest lepiej zoptymalizowany pod kątem redukcji zapotrzebowania na moc obliczeniową procesora niż program imperatywny, co z kolei częściowo potwierdza hipotezę numer 2.

W przypadku hipotezy 3 stwierdzono, że obie aplikacje są równie stabilne przy większości badanych operacji, wyłączając wspominaną wcześniej funkcjonalność pobierania elementu z opóźnieniem, w której stos reaktywny obsłużył o 50% więcej zapytań w przypadku wielu jednoczesnych żądań niż aplikacja imperatywna. Wynika to głównie z omawianego modelu działania serwera reaktywnego, który na żadnym etapie działania programu nie blokuje wątku wykonywalnego, a zamiast tego polega na pętli zdarzeń do informowania o odbywających się w systemie operacjach oraz o ich zakończeniu.

Ostatnie z badań dotyczące czasochłonności implementacji, potwierdziły hipotezę mówiącą o tym, że aplikacje reaktywne są bardziej czasochłonne do stworzenia niż aplikacje imperatywne. Otrzymane rezultaty pokazały, że każda z funkcjonalności w programie reaktywnym wymaga większej ilości kodu niż w programie imperatywnym, a dodatkowo kod całej aplikacji reaktywnej jest o średnio 10% dłuższy niż w przypadku aplikacji imperatywnej.

Podsumowując, postawiona w pracy teza „aplikacje reaktywne są wydajniejsze i stabilniejsze niż aplikacje imperatywne w przypadku jednoczesnej obsługi wielu zapytań" została częściowo potwierdzona, ponieważ aplikacje reaktywne są wydajniejsze, wyłączając zużycie pamięci RAM oraz są stabilniejsze, lecz tylko w jednym ze scenariuszy, na podstawie reszty z nich nie udało się stwierdzić tej różnicy. Dodatkowo badanie pokazało, że liczba zapytań nie ma tutaj znaczenia, ponieważ wyniki w każdym przypadku świadczą na korzyść aplikacji reaktywnej.

Literatura

[1] Dokumentacja Spring Framework odnośnie reaktywnych bibliotek, https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html, [28.05.2022].

[2] G. Salvaneschi, S. Proksch, S. Amann, S. Nadi, M. Mezini, On the positive effect of reactive programming on software comprehension: An empirical study. IEEE Transactions on Software Engineering, 43(12) (2017) 1125-1143.

[3] H. K. Dhalla, Benchmarking the performance of RESTful applications implemented in spring boot Java and MS. Net core, Journal of Computing Sciences in Colleges, 36(3) (2020) 178-178.

[4] S. Komolov, N. Askarbekuly, M. Mazzara, An empirical study of multi-threading paradigms Reactive programming vs continuation-passing style. 2020 the 3rd International Conference on Computing and Big Data, Taichung, 2020.

[5] G. Amuthan, Spring MVC. Przewodnik dla początkujących, Helion, 2015.

[6] J. Brittain, I. F. Darwin, Tomcat: The Definitive Guide, 2nd Edition, O'Reilly Media, 2003.

[7] O. Dokura, I. Lozynskyi, Hands-On Reactive Programming in Spring 5, Packt Publishing, 2018.

[8] Dokumentacja Spring Framework odnośnie reaktywnych bibliotek, https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html, [28.05.2022].

[9] Opis Spring WebFlux, https://piotrminkowski.com/2020/03/30/a-deep-dive-into-spring-webflux-threading-model/, [28.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 250–257
Received: 14 June 2022
Accepted: 6 July 2022

Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine libraries

Analiza wydajności relacyjnych baz danych MySQL, PostgreSQL oraz Oracle z zastosowaniem bibliotek Doctrine

Marcin Choina* , M. Skublewska-Paszkowska

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

Abstract
In modern applications, databases perform a very important function but the choice of a database system and additional libraries may affect the speed of the operations. The paper presents a time analysis concerning the performing of insert, update, delete and select operations on three database systems, MySQL 8.0, PostgreSQL 14.1 and Oracle 21c, cooperating with an application using Doctrine libraries. The obtained results showed differences between performing operations with and without object-relational mapping. In cooperation with the application, the operations were carried out the fastest using the PostgreSQL system. The Oracle system performed data selection faster without mapping on a large data set.

Keywords: relational databases; Doctrine; ORM; PHP

Streszczenie
We współczesnych aplikacjach, bazy danych pełnią bardzo ważną rolę, jednak wybór systemu bazodanowego i dodatkowych bibliotek może wpływać na szybkość wykonywania operacji. W niniejszej pracy przedstawiono czasową analizę dotyczącą wykonywania operacji bazodanowych insert, update, delete i select dla trzech systemów baz danych MySQL 8.0, PostgreSQL 14.1 i Oracle 21c, współpracujących z aplikacją wykorzystującą biblioteki Doctrine. Badania wykazały różnice między wykonywaniem operacji wraz z mapowaniem obiektowo-relacyjnym, a wykonywaniem samych zapytań. Przy współpracy z aplikacją, operacje najszybciej przeprowadzono korzystając z systemu PostgreSQL. System Oracle szybciej wykonywał operacje pobierania danych bez udziału mapowania na dużym zbiorze danych.

Słowa kluczowe: relacyjne bazy danych; Doctrine; ORM; PHP

*Corresponding author
Email address: marcinchoina1997@gmail.com (M. Choina)

1. Introduction

Database systems are a common method of data storage today. They are used in various types of IT systems, computer programs and applications. Currently, databases can be divided into several types (e.g. noSQL, object-oriented, hierarchical), but the most common choice for application development are relational databases [1]. There are many free and commercial relational database management systems available on the market, differing in capabilities and speed of operations. The performance is also influenced by the technology in which the application using the database is created and the operating system on which it is run.

This paper aims to compare the time performance of MySQL 8.0, PostgreSQL 14.1 and Oracle 21c database systems running with an application using Doctrine libraries, one of the most popular and fast technologies using object-relational mapping, allowing for the connection of an application written in PHP with the database. The performed tests allow to verify the time of operations and assess which database systems are the fastest and the slowest in carrying out operations of inserting, updating, deleting or selecting data, using, among others, the mechanisms of sorting, grouping and joining tables.

1.1. Doctrine

Doctrine is an open source library set for PHP technology under the MIT (the Massachusetts Institute of Technology) license. It is the default communication mechanism with the database for the Symfony framework and can be much more efficient compared to other mechanisms, e.g. Propel library [2]. The Doctrine abstraction layer (DBAL) enables cooperation with many available database systems. Additionally Doctrine is based on object-relational mapping (ORM) technology that allows to present information retrieved from the database using entity class objects. This solution allows to significantly improve the quality of the code and enables faster information management in the database [3]. Doctrine supports relationship mechanisms, which is achieved through associations between class objects. They make it possible to refer to other objects, and the library ensures that the relevant data is retrieved from the database [4].

1.2. MySQL

MySQL is a database system licensed under the GNU GPL license, currently developed by Oracle [5]. It is supported by a large number of technologies as well as

by the most popular programming languages such as PHP, Java, C ++, and it can be used on all popular operating systems [6]. It is also part of the server environment on the Linux - LAMP platform.

1.3. PostgreSQL

PostgreSQL is an advanced database system released under the PostgreSQL license, which is very similar to the MIT license. The system introduces many proprietary extensions to the implementation of the SQL standard. It enables the creation of stored procedures in various programming languages such as Python or Perl. PostgreSQL is a combination of a relational and object-oriented database system, which allows for a more precise adaptation of the database to the needs of the application being created [7].

1.4. Oracle

The Oracle database management system is a commercial solution released by Oracle Corporation under a paid license [8]. There is also a free version, the Express Edition, with limited functionality. Like PostgreSQL, it contains elements of an object-oriented database system. It also provides its own PL / SQL language, which enables the extension of standard functionalities to automate some activities carried out in the database.

2. Related works

Many scientists analyzed the differences between individual database solutions, their capabilities and performance. The authors of the article [9] chose to compare various relational database systems, Oracle Database 19c, SQL Server 2019, PostgreSQL 12 and MySQL 8. For this purpose, on each of them they created a database according to the same scheme, and then tested on it data selecting, grouping, and inserting operations as well as backup and restoring data. These tests showed differences in the performance of databases, where the most efficient in terms of time turned out to be the Oracle and SQL Server systems, and the MySQL system was the worst.

Oracle and SQL Server databases are commercial solutions which were compared by the authors of articles [10], [11]. The first one presents an analysis of the differences and performance of the servers in the given databases. The research presents the advantages of the Oracle database, which has better accessibility, in terms of the possibility of installation on various operating systems and support by a larger number of programming languages, as well as having multi-layer security. On the other hand, MS SQL is supported by simpler language syntax and better query performance for single and joined tables. In the second of the mentioned articles, the performance of these systems was compared with the use of a desktop application. In the study, an external application was used to perform various actions on the database, and the execution time of a given actions was taken from the views of the database system such as V$SQL for the Oracle database and sys.dm_exec_query_stats for MS SQL. The operations were performed on a set of 500-100,000 records for text or numeric data and on a set of 1-50 records for binary data larger than 50MB, and the cache was cleared after each action. The results showed that the MS SQL database performs DML (Data Manipulation Language) operations such as inserting and updating data better. The Oracle database is much better suited for DQL (Data Query Language) operations, which means data selecting.

Authors of many articles compare Oracle, SQL Server and MySQL databases. Examples of this comparison are the articles [12], [13]. In the first one, the time of updating one column of a table, transferring records to another table, and selecting records using sorting, grouping and joining tables in a database system running on a home computer was examined. The MySQL was the best for most operations in this case. In the second article, the authors compared these relational databases with non-relational databases such as Mongo, Redis, GraphQL and Cassandra, where the times of basic operations were compared. The obtained results confirmed that queries to non-relational databases were executed much faster than in the case of relational databases. Considering relational systems, in the Oracle system there has been achieved one of the fastest data selection times.

Currently, various frameworks or libraries are used to support the application development process, as well as connecting to the database and managing the data stored. Since the libraries are designed so that they can cooperate with various database systems, they can have an impact on the efficiency of performed operations. One of the most popular frameworks for creating web applications in PHP language are Symfony and Laravel. For Symfony, the preferred way to communicate with the database is to use Doctrine libraries, while Laravel has its own mechanism Eloquent ORM [14]. In the article [15], the authors examine the performance of relational databases cooperating with an application written using the Laravel framework. The test consisted in comparing the average execution times of operations such as insert, update, delete and select on a database running on mid-range hardware. To the research open source databases such as MySQL 8 and PostgreSQL 12 and the commercial Microsoft SQL Server 2017 were used. The results showed that in the case of a small number of records stored in the database (up to 1000 records), MySQL was the best solution, while with a larger number of records, the PostgreSQL system turned out to be better.

Another article presents the analysis of MySQL, MS SQL and PostgreSQL database systems in the context of web applications using the Spring framework, the Hibernate library and the JDBC interface [16]. In the case of database cooperation with a web application, PostgreSQL turned out to be the most efficient system for complicated operations on joined tables, using both the Hibernate library and the JDBC interface. In the case of a single table, the average times of performing opera-

tions were similar, and when performing operations directly on the database, MS SQL was the most optimal system.

In all above-mentioned articles, authors focused mostly on comparing the average times of performing basic operations on the database. In the article [17] the authors also analyzed the CPU load and the memory usage, and the authors of the article [9] examined the times of making and restoring a database backup. In most studies one model to build the database was used for each tested system. Some researchers used an additional application [11] or technology [15, 16], but few studies were related to the cooperation of databases with Doctrine libraries.

Based on the literature analysis, the following hypotheses were made:

- Insert, update and delete operations should be performed the fastest by the PostgreSQL database in cooperation with the Doctrine library,
- The Oracle database should perform the operations of selecting data the fastest with a large number of records (100,000) in the table.

3. Research method

In this paper, it was decided to compare the average times of performing select, insert, update and delete operations. These times were measured for the query with object-relational mapping on the data and without it. For the needs of the research, an application that uses Doctrine libraries has been created. It contained a module that allows to test the database. On each tested system, a database based on the model shown in Figure 1 was created.

The research was carried out according to a scenario. The purpose of the study was to verify the time of performing subsequent operations specified in Table 1, depending on the used database system and the number of records stored in the database tables. The initial condition for the study was a fixed number of records in each table of the tested database, which was 1,000, 10,000 or 100,000 records, respectively.

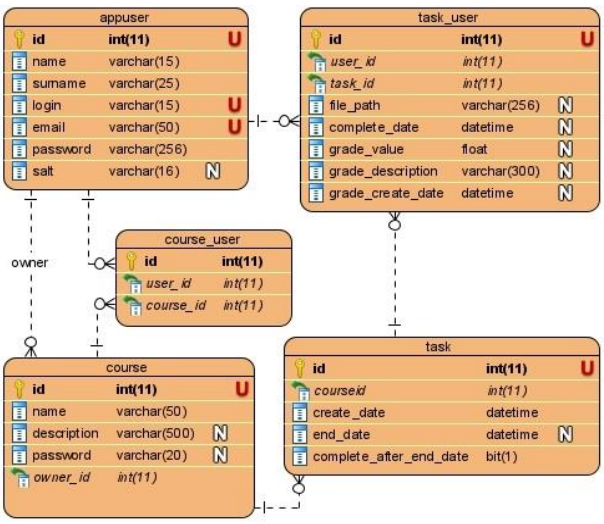


Figure 1: Database model.

The following activities were performed during the study:

- selecting one of the operations (Table 1),
- test of operations on MySQL database,
- test of operations on PostgreSQL database,
- test of operations on Oracle database,
- restoring the initial state of databases.

Before each test, the cache of the database system and application was cleared, and the entire test was repeated 10 times for each operation. Computer with the following parameters was used for the tests:

- Intel® Core™ i5-10300H processor,
- 8GB DDR4 2400 MHz RAM,
- Xioxia BG4 512GB NVMe SSD,
- Windows 10 64-bit operating system.

Table 1: Database operations

Operation	Query
Insert one row into the table	INSERT INTO appuser values (100001, “Adam”, ”Nowak”, ”anowak”, ”anowak@pollub.pl”, ”zaq1@WSX”)
Update one row in a table	UPDATE appuser SET name = “Marek” WHERE id = 567
Delete one row from the table	DELETE FROM task_user WHERE id = 567
Select all rows from one table	SELECT * FROM appuser
Select all rows sorted	SELECT * FROM appuser ORDER BY surname, name
Select rows using pattern search	SELECT * FROM appuser WHERE UPPER(name) LIKE ‘%ADA%’
Select one row from a table based on the primary key	SELECT * FROM appuser WHERE id = 567
Select rows with joining tables using JOIN construction	SELECT * FROM course c INNER JOIN appuser u ON u.id = c.owner_id INNER JOIN task t ON t.course_id = c.id
Select rows with joining tables using WHERE construction	SELECT * FROM course c, appuser u, task t WHERE u.id = c.owner_id AND t.course_id = c.id
Select rows using grouping functions	SELECT c.* FROM course c LEFT JOIN task t ON t.course_id = c.id GROUP BY c.* HAVING count(t.id) = 0
Select rows using a correlated query	SELECT t.* FROM task_user t WHERE grade_value = (SELECT MAX(grade_value) FROM task_user tu1 WHERE t1.user_id = t.user_id)

4. Results

The first tested operations were DML queries. First, insert of single user operation were tested (Figure 2-3). The results do not vary significantly depending on the number of records in the tables for each database system. This type of operations for both the query with mapping and without it were executed the quickest in the PostgreSQL database. With the other systems, insert operations were performed much slower. By performing the query without mapping to the database, using the MySQL database the shorter execution time was achieved than for Oracle database, but for this system the application processed the entity object into a query much longer. The application did not take a long time (18-21ms) to process the object to insert it into the Oracle database, which shows that this system works quicker with the application than MySQL.

The results of the update and data delete operation (Figure 4-7) were similar to the data insert operation. The fastest execution times were achieved using PostgreSQL database.The results of the other two systems differ depending on whether the object-relational mapping was considered. The shorter time was gained performing

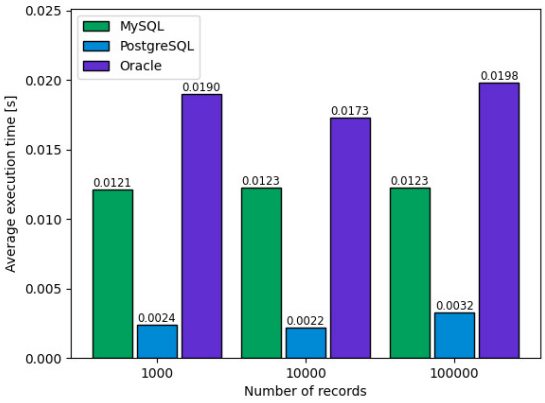


Figure 2: Average time of INSERT operation.

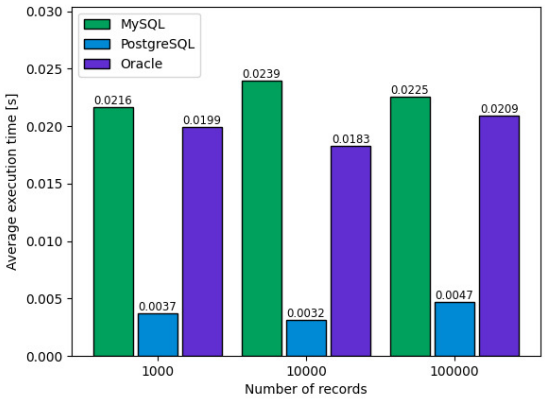


Figure 3: Average time of INSERT operation with ORM.

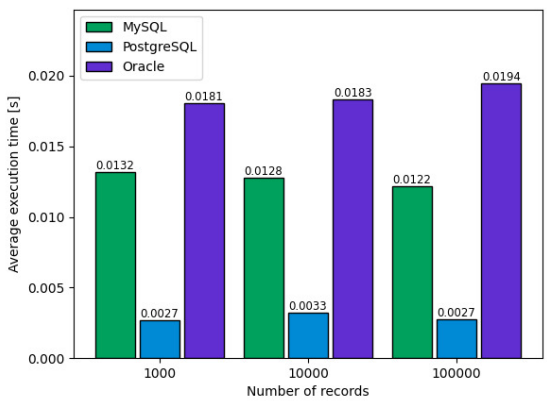


Figure 4: Average time of UPDATE operation.

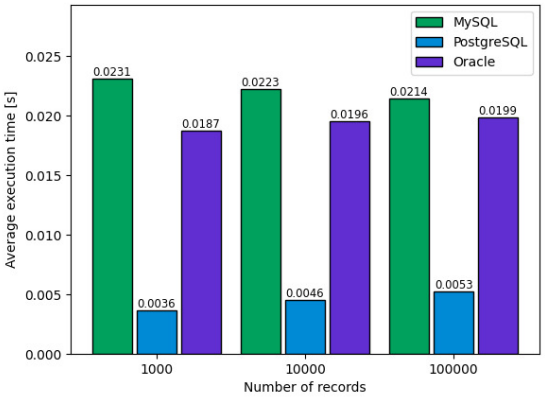


Figure 5: Average time of UPDATE operation with ORM.

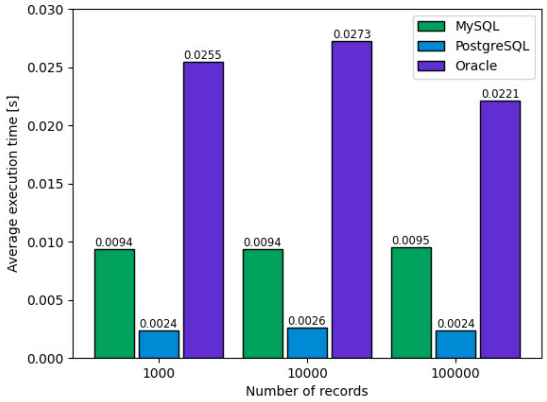


Figure 6: Average time of DELETE operation.

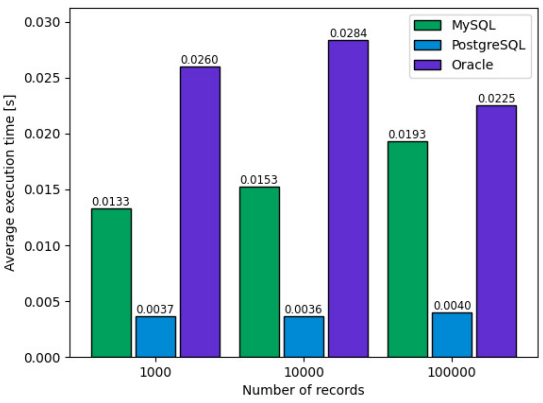


Figure 7: Average time of DELETE operation with ORM.

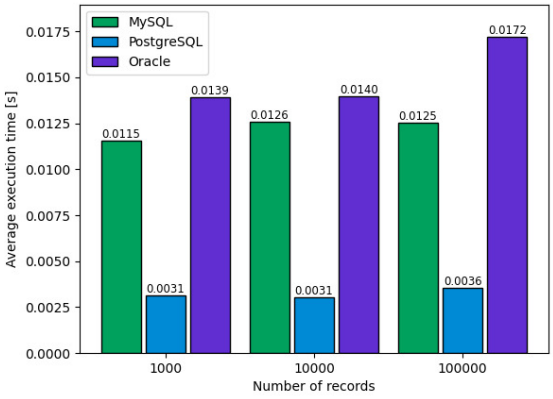


Figure 8: Average time of select one row operation.

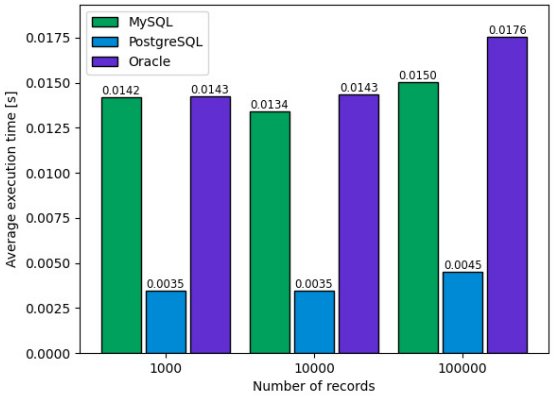


Figure 9: Average time of select one row operation with ORM.

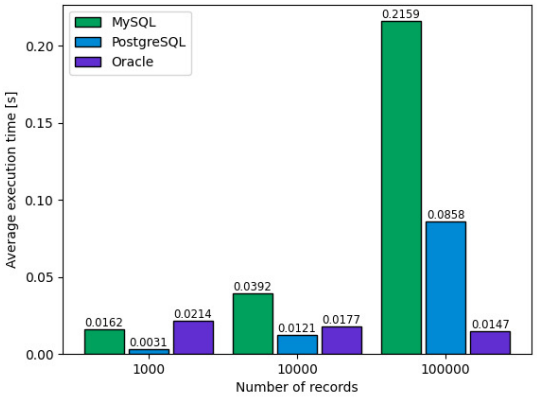


Figure 10: Average time of select all data operation.

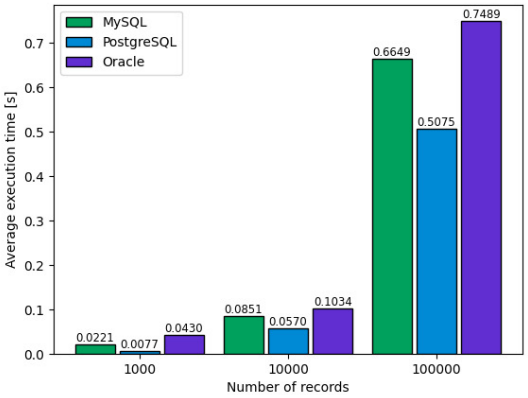


Figure 11: Average time of select all data operation with ORM.

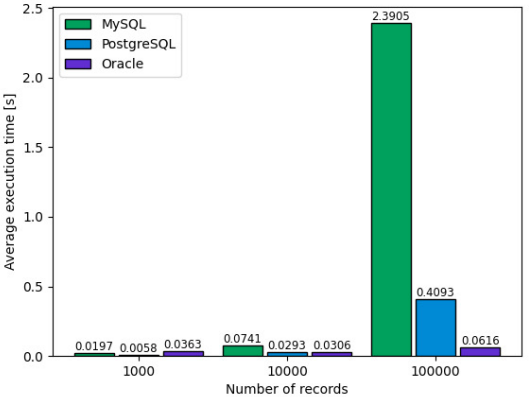


Figure 12: Average time of select sorted data.

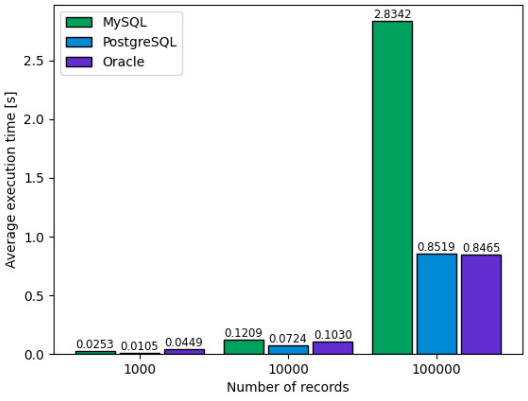


Figure 13: Average time of select sorted data with ORM.

update operation in cooperation with the application using the Oracle database than MySQL. Using the MySQL system theshorter time was obtained performing the update query without cooperating with the application. Aditionally the MySQL system performed the delete operation with and without object-relational mapping faster than the Oracle system.

Next operations belong to the group of DQL instructions that allow to retrieve specific data from the database. One of the most commonly used operations is selecting one row from a table based on its primary key. The results of this operation were similar to the results of DML operation studies (Figures 8-9). In the PostgreSQL database the data from the database has been obtained the fastest, but regarding to other databases, the performing query wihout mapping was faster with MySQL database than whit Oracle. While cooperating with the application, the longest times were obtained for a large number of records (i.e. 100,000) using the Oracle system. With less number of records in the tables, its results were similar to the MySQL system.

When selecting all the rows from the database (Figures 10-11), with PostgreSQL the shortest average time was obtained when the number of records in the

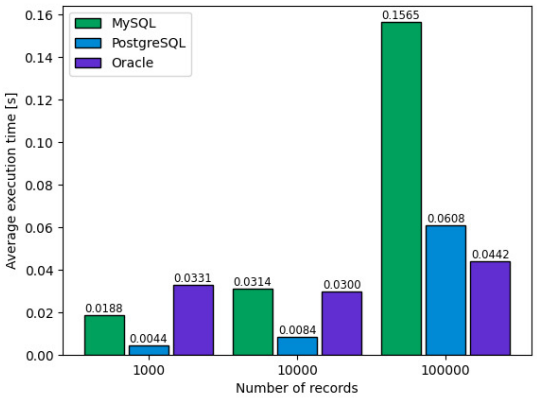


Figure 14: Average time of select operation using pattern search.

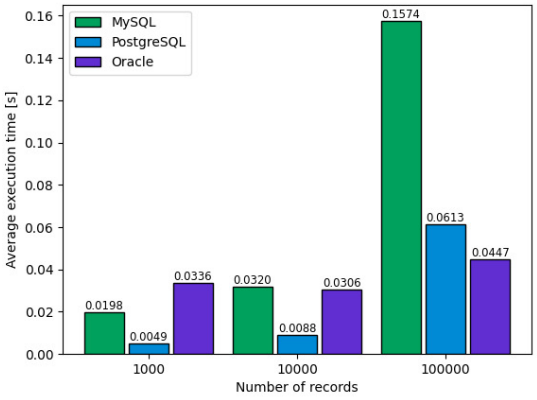


Figure 15: Average time of select operation using pattern search with ORM.

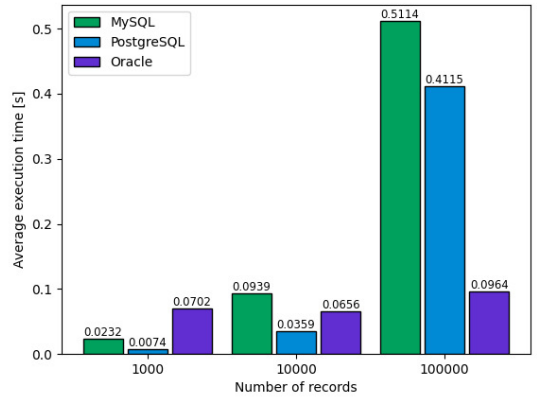


Figure 16: Average time of join operation using JOIN construction.

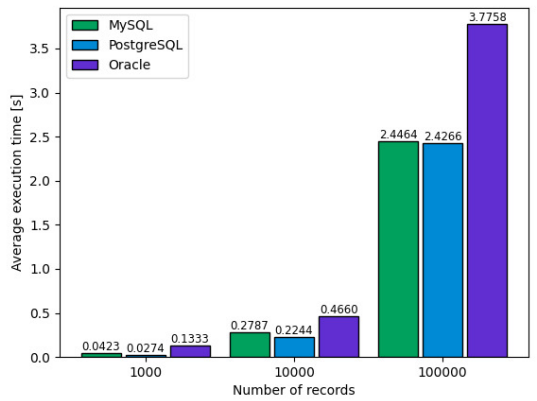


Figure 17: Average time of join operation using JOIN construction with ORM.

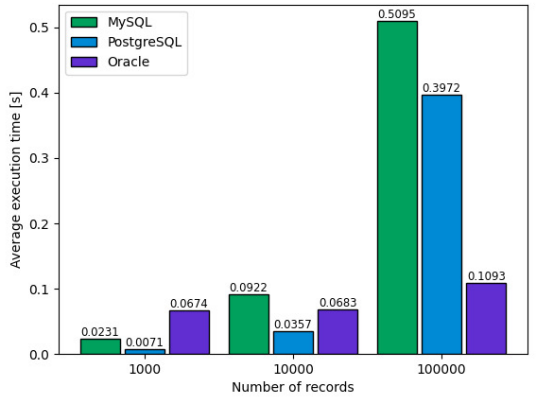


Figure 18: Average time of join operation using WHERE construction.

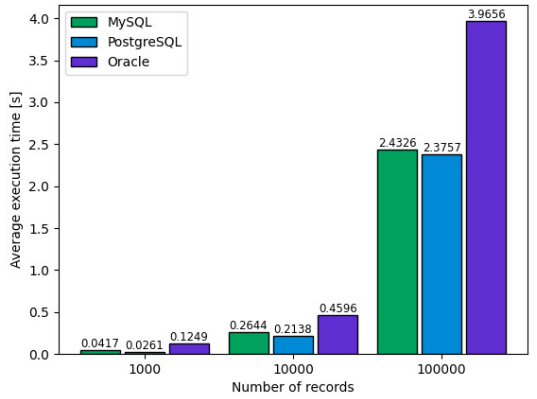


Figure 19: Average time of join operation using WHERE construction with ORM.

tables was low or medium (up to 10,000). With a larger number of records, the operations were performed much faster using the Oracle system, for which the average operation execution time decreased while the number of records increased. However, when working with the application, using the Oracle database the longest times were achieved and the best times were obtained using the PostgreSQL database, regardless of the number of records in the tables.

In the Oracle the best times were obtained when executing select queries using sort or pattern search (Figures 12-15) on a table with a large number of rows (i.e. 100,000). With fewer numbers, PostgreSQL performed queries the quickiest. The MySQL system with a large number of records in the tables, carried out both operations several times longer than the other systems. Regarding mapping with a large number of records (i.e. 10,000), there have been obtained a very similar result for the Oracle and the PostgreSQL system when sorting. When searching for pattern, the time was up to 27% shorter using Oracle system.

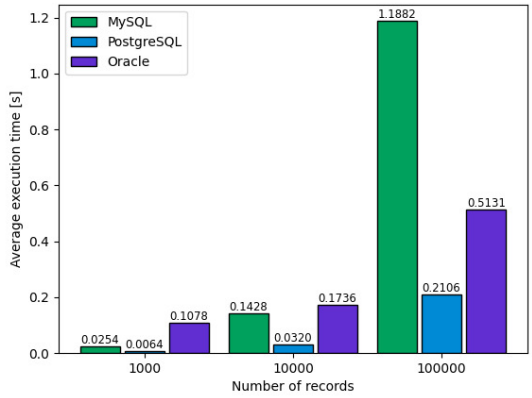


Figure 20: Average time of group operation.

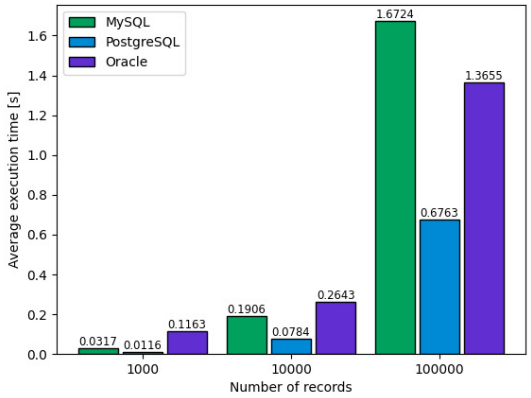


Figure 21: Average time of group operation with ORM.

When operations of joining the tables were performed, the results for both tested methods were similar (Figures 16-19). The Oracle system was very quick at executing a query for a large number of records in the database, while for a smaller number of records, the shortest time was achieved using the PostgreSQL system. When cooperating with the application, in the PostgreSQL database the best time was achieved, regardless of the number of records, but also using the MySQL system a very similar average execution time was acheved.

For the grouping operations, the best times were obtained using PostgreSQL database with and without mapping (Figures 20-21). The Oracle system performed operations for small and medium number of records (up to 10,000) the slowest, while in MySQL system the slowest execution time was obtained for a large number of records stored in tables.

For the select operation using correlated query (Figures 22-23), while executing the query without object-relational mapping on a large number of records the shortest time was obtained using Oracle database. With a smaller number of records in the table, or when the object-relational mapping was performed, the quickest times were gained using PostgreSQL database.

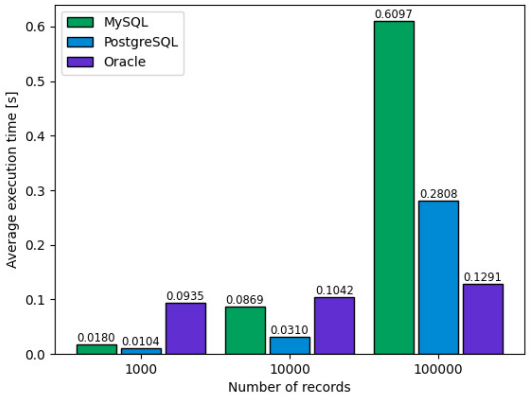


Figure 22: Average time of select operation using correlated query.

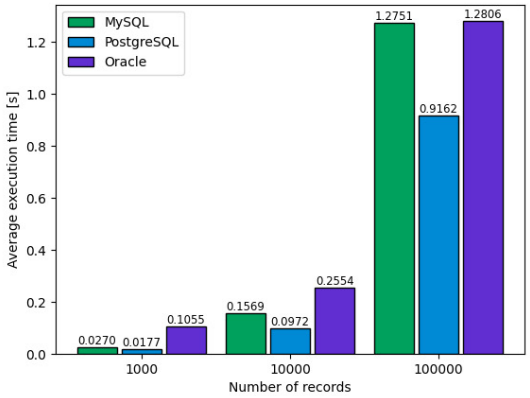


Figure 23: Average time of select operation using correlated query with ORM.

5. Discussion

In the study, in addition to various database systems, the number of records in the tables of a given database, and whether object-relational mapping was used, was taken into account. For different database systems, when executing the query without mapping, for data manipulation operations and for the rest of operations where the number of records in database tables was less than 100,000, PostgreSQL perform operations the fastest. With the Oracle system the shortest times was obtained for almost all data select operation when the number of records in the tables was 100,000, except for retrieving a record using a primary key and using grouping. The obtained result confirmed the results of the research in articles [9] and [13], in which the Oracle system performed select operations the fastest, when the operation was performed directly on the database.

Results vary depending on whether cooperation with application was used, that means executing query with object-relational mapping. Regarding the select operation with a like clause, on a table containing 100,000 records, using Oracle database system the significantly shorter time was obtained comparing to other analyzed database systems. Analising query execution with obiect-relational mapping, operations were performed even several times faster using the

PostgreSQL system. That confirmed the results obtained in the article [16], in which, the PostgreSQL system performed the operations the fastest using additional technology.

The operation execution time increases when object-relational mapping is performed. Regarding the DML operations executed by the MySQL database, the performed queries with mapping lasted even 10ms longer than the queries without it. The difference for time query execution with and without object-relational mapping for other analyzed database systems varied from 1 to 2 ms. For the select operations, the mapping time depended on the number of records stored in the database. The biggest difference between performing operations with and without mapping occurred for the Oracle database for a large number of records in the tables (i.e. 100,000). This database system executed the query without mapping the fastest, but with the mapping the execution times were the longest. When all rows were selected, the system performed operation up to 50 times longer with mapping than without it.

Based on the gathered results, it can be concluded that the first hypothesis, which is that insert, update and delete operations should be performed the fastest by the PostgreSQL database in cooperation with the Doctrine library, was confirmed. PostgreSQL always performed operations of inserting, updating and deleting data the fastest. The second hypothesis, that the Oracle database should perform the operations of selecting data the fastest with a large number of records (i.e. 100,000) in the tables, could be partially confirmed. Oracle database executed most of the queries for selecting data from tables with a number of 100,000 records the fastest. However, in cooperation with the application, the operation time was significantly increased in favor of the PostgreSQL system.

6. Conclusions

In the presented paper, the times of various database systems were tested and compared while working with the Doctrine library, which allowed to verify the hypotheses. The study showed that the PostgreSQL system is the fastest solution when working with an application using Doctrine libraries. The Oracle system works the best when performing select queries without mapping to the database. The research focused on the time of performed operations, which is the most noticeable feature of databases for the end user. Other aspects such as CPU load and memory utilization were not analyzed, which may be an interesting direction of future research.

References

[1] T. Connolly, C. Begg, Database Systems, A practical approach to Design, Implementation, and Management, sixth edition, Pearson, 2015.

[2] K. Sawłuk, M. Miłosz, Comparison of object-relational data mapping technology in Symfony 3 framework, Journal of Computer Sciences Institute 8 (2018) 235-240, https://doi.org/10.35784/jcsi.687.

[3] M. Lorenz, G. Hesse, J. Rudolph, Object-relational Mapping Revised - A Guideline Review and Consolidation, Proceedings of the 11th International Joint Conference on Software Technologies - ICSOFT-EA, (2016) 157-168, https://doi.org/10.5220/0005974201570168.

[4] Doctrine documentation, https://www.doctrine-project.org/index.html, [03.11.2021].

[5] MySQL documentation, https://dev.mysql.com/doc/refman/8.0/en/introduction.html, [26.05.2022].

[6] Supported Platforms: MySQL Database, https://www.mysql.com/support/supportedplatforms/database.html, [24.01.2022].

[7] PostgreSQL website, https://www.postgresql.org/about/, [26.05.2022].

[8] Oracle documentation, https://docs.oracle.com/en/database/oracle/oracle-database/21/cncpt/introduction-to-oracle-database.html, [26.05.2022].

[9] A. Solarz, T. Szymczyk, Oracle 19c, SQL Server 2019, Postgresql 12 and MySQL 8 database systems comparison, Journal of Computer Sciences Institute 17 (2020) 373-378, https://doi.org/10.35784/jcsi.2281.

[10] M. Ilić, L. Kopanja, D. Zlatković, M. Trajković, D. Ćurguz, Microsoft SQL Server and Oracle: Comparative performance analysis, The 7th International conference Knowledge management and informatics (2021) 33-40.

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

[12] K. Islam, K. Ahsan, S. Bari, M. Saeed, S. Ali, Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer, Journal of Basic & Applied Sciences 13 (2017) 481-490, https://doi.org/10.6000/1927-5129.2017.13.79.

[13] R. Čerešňák, M. Kvet, Comparison of query performance in relational a non-relation databases, Transportation Research Procedia 40 (2019) 170-177, https://doi.org/10.1016/j.trpro.2019.07.027.

[14] Eloquent documentation, https://laravel.com/docs/5.0/eloquent/, [26.05.2022].

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

[16] K. Lachewicz, Performance analysis of selected database systems: MySQL, MS SQL, PostgerSQL in the context of web applications, Journal of Computer Sciences Institute 14 (2020) 94-100, https://doi.org/10.35784/jcsi.1583.

[17] Y. Bassil, A Comparative Study on the Performance of the Top DBMS Systems, Journal of Computer Science & Research 1 (1) (2012) 20-31, https://doi.org/10.48550/arXiv.1205.2889.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 258–264
Received: 14 June 2022
Accepted: 2 July 2022

A comparative analysis of tools dedicated to project management

Analiza porównawcza narzędzi dedykowanych zarządzaniu projektami

Piotr Pawłowski*, Małgorzata Plechawska-Wójcik

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

Abstract

This article presents a comparative analysis of selected tools dedicated to project management: Asana, Trello, Wrike, Monday.com and Bitrix24. The aim of the work is to answer the research question: which project management tool is the most appropriate for members of development teams using classic and agile methodologies. Popular methods for analyzing the quality of interfaces have been reviewed. The research was carried out using the survey method conducted among users using project management tools and cognitive walkthrough - a method used to assess the usability of application and website interfaces.

Keywords: project management; interface usability analysis; cognitive walkthrough; LUT's list

Streszczenie

Niniejszy artykuł poddaje analizie porównawczej wybrane narzędzia dedykowane zarządzaniu projektami tj.: Asana, Trello, Wrike, Monday.com oraz Bitrix24. Celem pracy jest uzyskanie odpowiedzi na pytanie badawcze: które narzędzie do zarządzania projektami jest najbardziej odpowiednie dla członków zespołów programistycznych zarządzanych z wykorzystaniem klasycznych i lekkich metodyk zarządzania. W pracy dokonano przeglądu popularnych metod służących analizie jakości interfejsów. Eksperyment badawczy wykonano z wykorzystaniem ankiety przeprowadzonej w gronie użytkowników korzystających z narzędzi wspomagających zarządzanie projektami oraz wędrówki poznawczej – metody służącej do oceny użyteczności interfejsów aplikacji i serwisów internetowych.

Słowa kluczowe: zarządzanie projektami; analiza użyteczności interfejsu; wędrówka poznawcza; lista LUT

*Corresponding author

Email address: piotr.pawlowski@pollub.edu.pl (P. A. Pawłowski)

1. Wstęp

Dynamiczny rozwój gospodarki spowodował duże zapotrzebowanie na dostarczenie użytkownikom różnego rodzaju oprogramowania. Ważne jest, aby istniejące oraz nowo powstałe firmy zajmujące się wytwarzaniem oprogramowania korzystały z narzędzi, które będą odpowiednie zarówno dla kierowników projektu jak i członków zespołów projektowych.

Istnieje wiele artykułów na temat wyboru odpowiedniego narzędzia do zarządzania projektami. W artykule [1] dokonano przeglądu takich narzędzi. Ogólnopolskie badania przeprowadzono wśród tysiąca losowo wybranych członków Project Managment Institute. Badania wykazały, że na rynku istnieje wiele narzędzi do zarządzania projektami, chociaż większość ankietowanych kierowników projektów używa tylko części z nich. Najpopularniejszym narzędziem okazał się Microsoft Project. Wszystkie programy mają podobne zastosowania i koncentrują się na podstawowych zadaniach tj. planowanie, śledzenie, kontrolowanie i harmonogramowanie. W badaniu [2] przeprowadzono analizę empiryczną dotyczącą wykorzystania narzędzi i technik zarządzania projektami w całym cyklu życia projektu. Zbadano także ich wpływ na powodzenie projektu. Według autorów artykułu obecne narzędzia i techniki zarządzania projektami PMTT (ang. *Project Managment Tools and Techniques*) są powszechnie znane, jednak do tej pory nie były dokładnie zbadane. Badania ankietowe wykazały, że wykorzystanie PMTT zależy od poszczególnych faz życia projektu.

Na rynku potrzeba narzędzi, które pozwolą menedżerom na łatwiejszą administrację w trakcie cyklu życia projektu. Taki wniosek można wysnuć na podstawie badań przeprowadzonych przez różnych autorów. W pracy [3] opracowano krótkie porównanie aż 20 takich narzędzi – zarówno płatnych jak i na licencji darmowej (ang. *open source*). Wyniki przedstawiono w formie tabeli, gdzie określono funkcje poszczególnych programów oraz sposób ich licencjonowania. Stwierdzono, że dobór odpowiedniego narzędzia zależy przede wszystkim od wielkości projektu. Podobne zestawienie porównawcze przedstawiono w artykule [4]. Opracowano zestaw 16 narzędzi m.in.: Asana, Trello i Wirke, które stosowane są w projektach prowadzonych przy użyciu metodyk zwinnych. Również tutaj znalazły się narzędzia płatne i darmowe. Stwierdzono, że niektóre z narzędzi używane są do różnych typów metodyki Agile m.in. Scrum, Kanban czy XP (ang. *Xtreme Programming).*

Rosnąca dynamika i wymagania obecnego rynku powoduje, że organizacje i firmy informatyczne zajmujące się zarządzaniem projektami stają przed coraz to większymi wyzwaniami. Zarządzanie projektem w sposób jedynie intuicyjny, bez skutecznych i powszechnie znanych schematów działania, może być powodem ukończenia projektu w stopniu niezadowalającym i przeciętnym. Zastosowanie odpowiednich praktyk i metodyk zarządzania projektami nie tylko wspo-

maga pracę organizacji ale również zapewnia wzrost wskaźnika sukcesu projektu.

W celu usprawnienia zarządzania projektami na rynku powstało wiele standardów takich jak PMBOK, PRINCE2 oraz SCRUM. Znajdują one zastosowania w dużych organizacjach, gdzie komunikacja między członkami zespołu oraz zarządzanie poszczególnymi zadaniami musi przebiegać sprawnie. W dużej mierze minimalizuje to ryzyko niepowodzenia projektu [5]. W artykule [6] podjęto próbę opracowania jednostki instruktażowej służącej do nauczania narzędzi zarządzania projektami dostosowaną do standardu PMBOK. Efektem pracy autorów było również rozszerzenie możliwości narzędzia typu *open source* do zarządzania projektami jakim jest dotProject. Badani studenci stwierdzili, że instruktaż w dużym stopniu ułatwił im nauczenie się korzystania z narzędzi do zarządzania projektami. Natomiast nauczyciele stwierdzili, że chcieliby użyć opracowanego instruktażu ponownie i polecić go do nauki innych narzędzi. Proponowane rozwiązanie może wypełnić braki w zakresie nauczania korzystania z narzędzi do zarządzania projektami.

W ostatnich czasach zwinne metodyki zarządzania projektami przyciągnęły uwagę inżynierów i badaczy głównie ze względu na ich szybki rozwój. Celem artykułu [7] jest analiza sześciu najbardziej popularnych metodyk zwinnych. Metodyki porównano pod względem ważnych czynników definiujących projekt tj. typ podejścia, czas jednej iteracji, wielkość drużyny, rozmiar projektu, rodzaj komunikacji zespołu oraz zaangażowanie klienta.

Obecna sytuacja pandemiczna SARS-CoV-2 nie oszczędziła nowo tworzących się firm informatycznych. W artykule [8] przeprowadzono badania na średniej wielkości firmie typu Startup Di2Win z siedzibą w Brazylii w okresie od 15 marca do 6 maja 2020 roku. Badanie miało na celu wdrożenie praktyk, które ograniczą zagrożenia związane z niepowodzeniem projektu. Wyniki działań, które podjęto w trakcie eksperymentu mogą być pomocne dla firm, które borykają się z konsekwencjami spowodowanymi przez SARS-CoV-2 oraz studentom, którzy nie są przyzwyczajeni do pracy zdalnej.

Tworząc lub analizując jakość interfejsów aplikacji i narzędzi ważne jest, aby dobrać odpowiednie metodyki badawcze. Warto podkreślić, że istnieje wiele technik które temu służą. Do najbardziej popularnych należą:

1. Heurystyki Nielsena – stosowana przy projektowaniu witryn internetowych, audytach UX (ang. User Experience) jak również w optymalizacji SEO aby zwiększyć indeksowanie witryny [9].
2. Lista LUT – oparta o szeroko stosowaną w kwestionariuszach skalę Likerta, która zawiera się w wartościach liczbowych od 1 do 5. Uzupełnieniem jej jest skala WUP, która określa stopień atrakcyjności interfejsu [10].
3. Wędrówka poznawcza - pomocna przy przeprowadzaniu audytu użyteczności (ang. User Experience) danego interfejsu czy strony internetowej z perspektywy użytkownika, który po raz pierwszy ma kontakt z danym środowiskiem [11].

W pracy badawczej [12] do oceny narzędzi zarządzania projektami wykorzystano ocenę heurystyczną Neilsena. W badaniu wzięło udział 8 osób, które miało wykonać 17 zadań w narzędziach Jira, VersionOne, AgileZen i ZebraPlan. Zadania były ściśle związane z tematyką zarządzania projektami m.in. tworzenie projektu, dodawanie członków, tworzenie zadań aż po zmianę ich priorytetów. W celu identyfikacji problemów z użytecznością interfejsu poproszono uczestników o wyrażanie swoich opinii na głos. Uczestnicy badania oceniali też trudność każdego zadania w 5-cio punktowej skali Likerta. Przed wykonaniem zadań przeprowadzono ankietę, w celu oceny znajomości inżynierii oprogramowania oraz narzędzi do zarządzania projektami. Autorzy uważają, że przeprowadzone badania mogą pomóc firmom zajmującym się tworzeniem oprogramowania w doborze odpowiednich narzędzi. W artykule [13] podjęto próbę analizy narzędzi służących do badania jakości interfejsów użytkownika. Jednym z przedstawionych sposobów testowania interfejsu jest wędrówka poznawcza oraz rozwinięta wędrówka poznawcza. Uproszczona wędrówka poznawcza ma na celu zdefiniowanie zadań do wykonania dla badanego użytkownika. Następnie zadania można podzielić na mniejsze kroki a ich trudność może być oceniana w 5-cio stopniowej skali. Aby lepiej ocenić użyteczność interfejsu można wykonać badanie rozwiniętą wędrówką poznawczą. Jednak według autora artykułu zastosowanie wielu kryteriów złożonych może znacznie wydłużyć proces badawczy. Wnioskiem z podjętej analizy jest fakt, że należy przyjąć odpowiednie metodyki badawcze interfejsu. W innym wypadku twórca aplikacji może definiować swoje własne standardy czego skutkiem może być zła ocena jakości.

W artykule [14] użyto rozwiniętej wędrówki poznawczej w celu oceny projektów interfejsów sprzętu medycznego. Wprowadzono szereg ulepszeń w stosunku do podstawowej wędrówki poznawczej m.in. podział pytań na dwa poziomy, stopniowanie zadań i kategoryzacja na typy problemów oraz projekcja wyników w postaci macierzy. Zastosowana metoda pozwoliła na lepszą identyfikację problemów z użytecznością interfejsu i zdaniem autorów można ją stosować również w produktach konsumenckich. W badaniu [15] podjęto próbę oceny narzędzia AntTracks służącego do analizy wycieków pamięci, które powszechnie występują w wielu językach programowania. Jako metodę badawczą użyto wędrówki poznawczej. Autorzy zdefiniowali 6 zadań które następnie samodzielnie wykonywali testując ich poziom trudności. Następnie zaangażowali osobę, która nigdy wcześniej nie korzystała z narzędzia AntTracks żeby na podstawie informacji zwrotnych dostosować metodę badania (w szczególności niejasności w instrukcjach). Do badania zaproszono czternastu uczestników, którzy nie korzystali z narzędzia AnTracks. Przed badaniem zapoznano użytkowników z zadaniami oraz podano ich treść. W trakcie badania proszono ich o werbalne wyrażanie czynności, które

będą wykonywać oraz komentowanie na bieżąco swoich problemów. Po odbytej sesji badawczej uczestnicy wypełnili kwestionariusz opracowany na podstawie atrybutów użyteczności Nielsena. Zastosowane metody badawcze pozwoliły na uzyskanie oczekiwanych rezultatów. Przeprowadzone badanie wędrówką poznawczą pozwoliło na uzyskanie sugestii i komentarzy od badanych osób. Powyższe kwestie pozwolą autorom artykułu w przyszłości ulepszyć tworzone narzędzie. Zastosowanie tego typu badania może służyć jako przewodnik lub wsparcie dla nowych użytkowników aplikacji.

W artykule [16] przedstawiono ocenę interfejsu systemu ERP służącego do integracji danych biznesowych. W celu wyboru metody badawczej dokonano porównania między analizą ekspercką a wędrówką poznawczą. Stwierdzono, że badanie wędrówką poznawczą należy stosować jeśli chcemy wykonać analizę interfejsu w krótkim czasie przy niewielkim nakładzie kosztów. Według autorów artykułu taką analizę możemy stosować jako dodatek do bardziej szczegółowych metod badawczych. Ostatecznie oceniono aplikację względem listy LUT. Taki podział interfejsu na podobszary pozwolił wskazać ekspertom kluczowe problemy interfejsu badanego systemu ERP.

2. Cel i zakres pracy

Celem pracy jest przeprowadzenie analizy porównawczej narzędzi dedykowanych zarządzaniu projektami. Eksperyment ma na celu stwierdzić, które narzędzie jest najbardziej odpowiednie dla użytkowników. Przetestowano wybrane elementy narzędzi pasujące do obu metodyk. Praca badawcza została wykonana z wykorzystaniem ankiety przeprowadzonej w gronie użytkowników korzystających z tego typu oprogramowania oraz wędrówki poznawczej – metody służącej do oceny użyteczności interfejsów aplikacji i serwisów internetowych.

Teza: Asana jest obecnie najlepszym rozwiązaniem dla członków zespołów programistycznych zarządzanych z wykorzystaniem klasycznych i lekkich metodyk zarządzania.

Szczegółowe pytania badawcze:

1. Czy narzędzie Asana do zarządzania projektami posiada najlepsze funkcjonalności?
2. Czy interfejs użytkownika narzędzia Asana jest oceniany jako użyteczny wśród początkujących użytkowników?
3. Czy Asana jest najpopularniejszym narzędziem wśród użytkowników korzystających z niego w codziennej pracy?

3. Plan badań

Plan eksperymentu badawczego składa się z następujących etapów:

1. Stworzenie ankiety dotyczącej tematyki narzędzi do zarządzania projektami na około 20 pytań
   a) Zebranie danych personalnych
   b) Przygotowanie pytań z zakresu branży w jakiej stosuje się tego typu aplikacje
   c) Przygotowanie pytań z zakresu opinii i spostrzeżeń z zakresu wybranego narzędzia
   d) Przygotowanie pytań z zakresu wymaganych cech i funkcjonalności narzędzi
2. Przeprowadzenie badania ankietowego wśród użytkowników korzystających z takich narzędzi
   a) Grupa badawcza: użytkownicy/osoby pracujące w branży wykorzystującej aplikacje do zarządzania projektami
   b) Oczekiwany przedział wiekowy: osoby w wieku produkcyjnym
   c) Branża: informatyczna lub pokrewna związana z zarządzaniem projektami
   d) Wykształcenie: preferowane wyższe
   e) Minimalna liczba osób badanych: 25
3. Analiza wyników badań ankietowych i wybranie listy narzędzi do badania praktycznego
4. Stworzenie badania praktycznego – wędrówki poznawczej
   a) Grupa badawcza: studenci
   b) Oczekiwany przedział wiekowy: 21-25 lat
   c) Branża użytkownika: brak
   d) Wykształcenie: w trakcie studiów
   e) Minimalna liczba osób badanych: 5
5. Przeprowadzenie badania praktycznego – wędrówki poznawczej
   a) Przygotowanie zadań i podzadań
   b) Rozszerzenie badania o ankietę LUT i pomiary czynności wykonywanych przez użytkownika
6. Analiza wyników wędrówki poznawczej
7. Wnioski

3.1. Badanie ankietowe

Celem badania ankietowego było pozyskanie informacji o użytkownikach korzystających z narzędzi do zarządzania projektami. Ankietę wśród respondentów przeprowadzono drogą elektroniczną. Badanie udostępniono na grupach społecznościowych związanych z zarządzaniem projektami oraz wysłano je osobiście do firm w Lublinie realizujących projekty informatyczne.

W pierwszej części ankiety zebrano podstawowe informacje takie jak: płeć, wiek, wielkość przedsiębiorstwa, branża czy możliwość wyboru przez pracownika narzędzia wspierającego jego pracę.

Respondenci proszeni byli o wskazanie znanych narzędzi oraz określenie narzędzia, z którego najczęściej korzystają. Na podstawie tego wyboru każdy uczestnik badania ankietowego ocenił i wyraził opinię o jego interfejsie. Dodatkowo zbadano m.in. jak osoba badana nauczyła się obsługi narzędzia, jej biegłość oraz sposób w jaki najczęściej rozwiązuje problemy napotykane w codziennej pracy z narzędziem.

Ostatnia część ankiety dotyczyła ważnych cech i funkcjonalności, jakie według użytkownika dane narzędzie powinno posiadać.

3.2. Obiekty badania praktycznego

Badanie praktyczne wędrówką poznawczą przeprowadzono na następujących narzędziach: Asana, Trello, Wrike, Monday.com oraz Bitrix24.

3.3. Wędrówka poznawcza

Badanie przeprowadzono na sześciu osobach, które nie miały doświadczenia w pracy z tego typu programami. Uczestnicy badania posiadali wykształcenie wyższe lub byli absolwentami studiów inżynierskich na Politechnice Lubelskiej. Stworzono cztery główne zadania z podziałem na podzadania. Tak przygotowaną listę kroków przedstawiono poniżej.

1. Wejście na stronę webową narzędzia
   a) Wyszukanie opcji zmiany języka
   b) Logowanie do konta użytkownika
2. Konfiguracja projektu
   a) Dodanie nowego projektu
   b) Zaproszenie dodatkowych członków do projektu
   c) Dodanie nowego widoku projektu
   d) Utworzenie dwóch zadań w projekcie
3. Zarządzanie zadaniami
   a) Przypisanie zadań członkom projektu
   b) Ustawienie terminu wykonania zadań
   c) Odnalezienie opcji dodania załączników
4. Zakończenie pracy z programem
   a) Wylogowanie użytkownika

Przy każdym podzadaniu uczestnicy badania odpowiadali na następujące cztery pytania:

1. Czy użytkownik wiedział jak rozpocząć zadanie?
2. Czy użytkownik połączył akcję z oczekiwanym rezultatem?
3. Czy użytkownik zauważy postęp w kierunku oczekiwanego rezultatu?
4. Czy użytkownik prawidłowo ukończył zadanie?

W trakcie trwania danego podzadania zliczana była liczba błędów oraz poinformowano użytkowników badania o tym, aby werbalnie wyrażali swoje opinie i spostrzeżenia. W każdym badaniu nagrywany był ekran oraz włączony mikrofon w celu rejestrowania uwag użytkowników w czasie rzeczywistym. Podczas nagrywania obrazu skorzystano z programu komputerowego *Mousotron 12.1* w celu analizy danych takich jak: czas wykonania zadania, liczba naciśnięć klawiatury komputerowej, liczba naciśnięć lewego przycisku myszy, przebyta droga kursora po interfejsie narzędzia. Otrzymane wyniki pozwoliły na lepszą interpretację wyników i wniosków z przeprowadzonej analizy.

3.4. Badanie za pomocą listy LUT

Grupa badawcza biorąca udział w badaniu przy pomocy wędrówki poznawczej dodatkowo oceniała jakość interfejsów badanych narzędzi za pomocą opracowanej listy LUT, która odnosiła się do następujących obszarów:

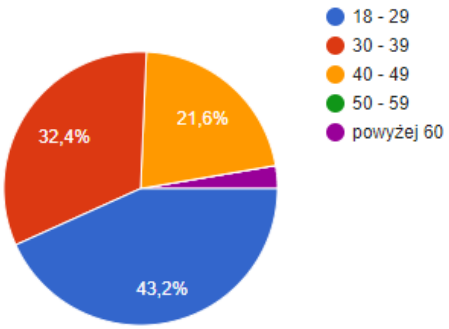
1. Nawigacja i struktura
2. Komunikaty, feedback, pomoc dla użytkownika
3. Interfejs aplikacji
4. Treść podstron
5. Wprowadzanie danych

Dla każdego narzędzia i każdego z pytań odnoszącego się do danego obszaru wyliczono średnią ocenę w skali od 1 do 5. Następnie wyznaczono średnią ocenę ogólną, aby obliczyć metrykę WUP (ang. *Web Usability Points*).

4. Wyniki badań

4.1. Wyniki badań ankietowych

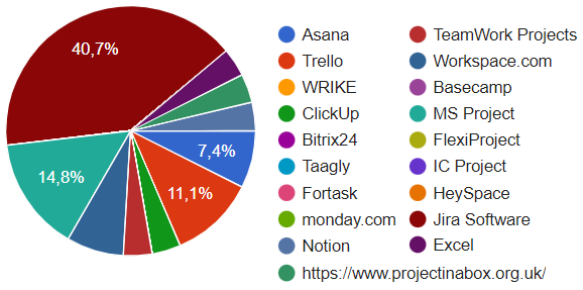
W badaniu ankietowym łącznie uzyskano 37 odpowiedzi. Po wykluczeniu grupy respondentów, która nie korzystała z aplikacji do zarządzania projektami, zarówno w przedsiębiorstwie i do celów prywatnych, łączna liczba ankiet wyniosła 27. Zainteresowanie ankietą wyraziło 16 mężczyzn i 21 kobiet. Największą grupą badawczą były osoby stosunkowo młode (16 osób) w wieku 18-29 lat (43,2%). Kolejną co do wielkości grupę stanowiły osoby w przedziale wiekowym 30-39 lat (32,4%). Na trzecim miejscu znalazła się grupa respondentów w przedziale wiekowym 40-49 lat (21,6%). Udział procentowy grup wiekowych respondentów pokazano na Rysunku 1.



Rysunek 1: Wykres kołowy prezentujący udział procentowy wieku ankietowych.

Ponad połowa ankietowanych biorących udział w badaniu pracuje w korporacjach zatrudniających powyżej 250 osób. Dominującym sektorem w którym pracują respondenci okazała się branża IT.

Prawie połowa respondentów (40,7%) korzysta z narzędzia Jira Software. Na drugim miejscu uplasowało się narzędzie MS Project z wynikiem 14,8%. Trzecią lokatę zajęło Trello (11,1%) natomiast czwarte miejsce przypadło Asanie (7,4%). Pozostałe narzędzia zdobyły łącznie mniej niż 7% wszystkich głosów. Wyniki obrazuje Rysunek 2.



Rysunek 2: Wykres kołowy prezentujący udział procentowy najczęściej używanych narzędzi.

W kolejnym segmencie ankiety zebrano wyniki opinii o wybranym narzędziu. Pod względem zadowolenia korzystania z narzędzia, respondenci ocenili wybrane aplikacje wysoko. Najbardziej pożądanymi cechami takich narzędzi okazały się: intuicyjny interfejs, rozbudowana wersja darmowa, system śledzenia historii

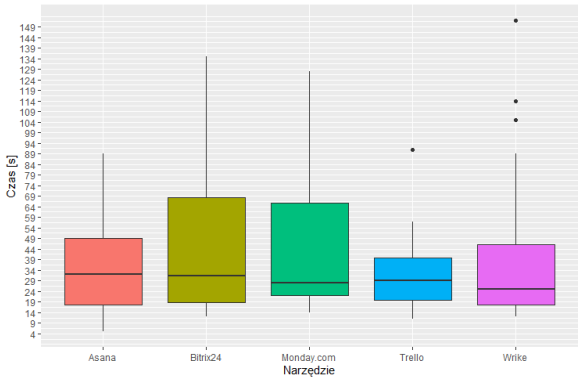
zmian oraz istnienie aplikacji mobilnej dla narzędzia. Natomiast to, co przeszkadza ankietowanym w tego typu aplikacjach jest czas nauki programu oraz koszt jego pełnej wersji. Jeśli chodzi o naukę obsługi narzędzia większość ankietowanych korzystała ze wsparcia kolegów z pracy oraz samodzielnie poznawała interfejs programu. Nieznaczna część badanych korzystała z profesjonalnego wdrożenia. Z analizy zebranego materiału wynika, że wiedza pozyskana z różnych źródeł nauki okazała się w dużym stopniu wystarczająca do opanowania obsługi i rozpoczęcia samodzielnej pracy z programem.

W końcowej części ankiety zapytano o priorytet poszczególnych funkcjonalności jakie powinny posiadać narzędzia do zarządzania projektami. Najbardziej istotnymi funkcjonalnościami według opinii respondentów są tablice Kanban oraz możliwość pracy z wieloma projektami.

4.2. Wyniki badań wędrówką poznawczą

Na podstawie danych liczbowych zebranych w trakcie przeprowadzania badania stworzono wykresy pudełkowe (ang. *box plot*). Długość pudełka zależy od wartości kwartyli Q1 i Q3. Długość dolnych i górnych wąsów zależy odpowiednio od wartości minimalnej i maksymalnej w zbiorze. Im wąsy są dłuższe, tym wyniki obserwacji są mniej skoncentrowane. Za pomocą wykresów tego typu można ilustrować duże ilości danych oraz zaobserwować wartości odstające w formie graficznej. Wykresy wygenerowano za pomocą programu *RStudio*. Poniżej zebrano wyniki dla każdej analizowanej cechy (trasa myszy, lewy przycisk myszy, liczba wciśnięć klawiszy oraz czas wykonania) w odniesieniu do najbardziej złożonego zadania (Zadanie 2: Konfiguracja projektu).

Biorąc pod uwagę czas potrzebny na wykonanie zadania, narzędzie Trello wypadło najlepiej, posiadając najbardziej skoncentrowane wyniki (Rysunek 3). Trzy wartości odstające dla narzędzia Wrike spowodowane były problemem z ukończeniem zadania.

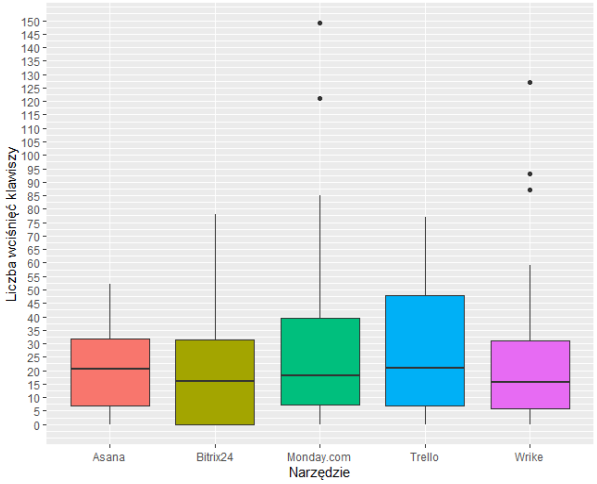


Rysunek 3: Rozkład danej czas zadania w dla wszystkich narzędzi.

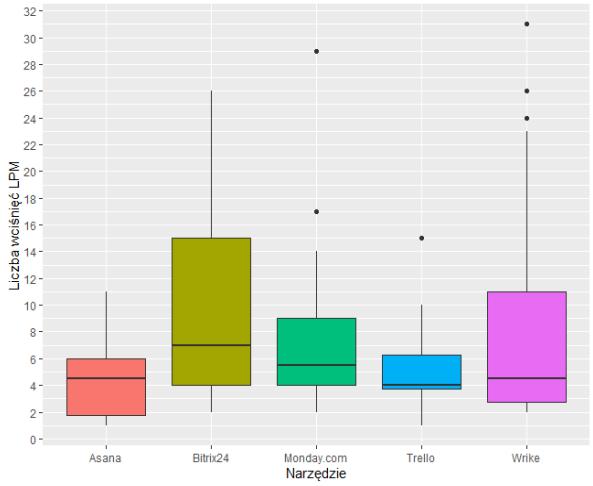
Analizę danych naciśnięć klawiszy przeprowadzono tylko dla zadania 1 i 2. Badani mieli do wpisania konkretne frazy tekstu a ewentualne wartości odstające są spowodowane pomyłkami użytkowników. Rozkład danej wciśnięć klawiszy zadania 2 dla wszystkich narzędzi pokazano na Rysunku 4.

Analizując liczbę wciśnięć lewego przycisku myszy można wywnioskować, iż narzędzie Asana wymagało najmniejszej liczby interakcji z użytkownikiem w celu osiągnięcia sukcesu. Ponownie narzędzie Bitrix24 wymagało od użytkownika poświęcenia więcej uwagi nad interfejsem i dłuższą eksplorację serwisu. Rozkład danej liczba wciśnięć lewego przycisku myszy dla zadania 2 pokazuje Rysunek 5.

Ostatnią cechą poddaną analizie był dystans pokonany w obrębie badanego interfejsu przez kursor myszy. Dla zadania 2 (Rysunek 6) zaobserwowano dużą liczbę wartości odstających. Najniższą wartość mediany odnotowano dla narzędzia Trello.



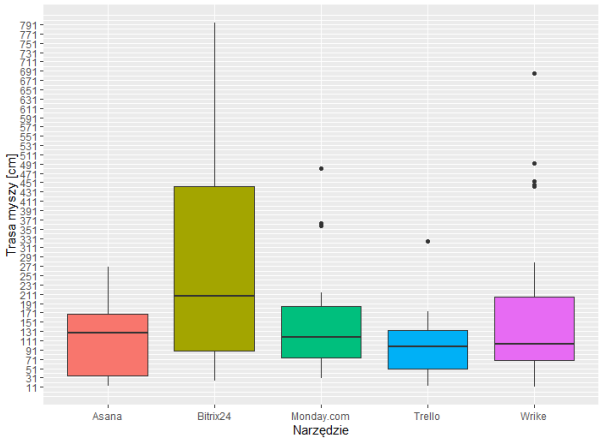
Rysunek 4: Rozkład danej wciśnięć klawiszy zadania 2 dla wszystkich narzędzi.



Rysunek 5: Rozkład danej liczba wciśnięć LPM zadania 2 dla wszystkich narzędzi.

Pod względem poprawności wykonanych zadań oraz liczby popełnionych błędów narzędzie Trello osiągnęło najlepszy wynik (liczba błędów 4). Drugie miejsce przypadło narzędziu Asana (8 błędów). Na miejscu trzecim uplasowało się narzędzie Wrike z liczbą błędów równą 19. Miejsce czwarte przypadło narzędziu Monday.com z liczbą błędów równą 28. Różnica między narzędziem, w którym popełniono najwięcej błędów (Bitrix24) a narzędziem, w którym tych błędów popełniono najmniej wynosi aż 17,04%. Tabela 1 zawiera

procentowe zestawienie liczby błędów z 44 kroków dla każdego narzędzia.



Rysunek 6: Rozkład danej trasa myszy zadania 2 dla wszystkich narzędzi.

Tabela 1: Procentowe zestawienie liczby błędów z 44 kroków dla każdego narzędzia

	Asana	Trello	Wrike	Monday.com	Bitrix24
U1	1	0	3	4	8
U2	0	2	0	4	0
U3	1	1	4	4	10
U4	1	0	3	4	5
U5	3	1	4	6	4
U6	2	0	5	6	7
Liczba błędów dla narzędzia	8	4	19	28	34
Średnia liczba błędów z 44 kroków	2	1	4,75	7	8,5
Liczba błędów w %	4,55	2,28	10,8	15,91	19,32

4.3. Wyniki badań listą LUT

Na podstawie ocen ogólnych z poszczególnych obszarów i podobszarów wyliczono metrykę jakości aplikacji webowej WUP a jej wyniki zaprezentowano w Tabeli 2.

Tabela 2: Wyliczone wartości metryki WUP

Narzędzie	WUP
Asana	3,97
Trello	3,48
Wrike	3,64
Monday.com	3,17
Bitrix24	2,82

Podsumowując zestawienie ocen WUP pierwszą lokatę zajęło narzędzie Asana z wynikiem 3,97. Druga lokata przypadła narzędziu Wrike. Zaskoczeniem może się okazać miejsce trzecie, które zdobyło narzędzie Trello. W badaniu wędrówką poznawczą narzędzie to miało najmniejszą liczbę błędów oraz największy procent wykonanych zadań. Na czwartej pozycji uplasowało się narzędzie Monday.com z wynikiem WUP na poziomie 3,17. Natomiast narzędziem znacznie odstającym od pozostałej grupy okazało się narzędzie Bitrix24. Jako jedyne osiągnęło ocenę WUP na poziomie poniżej 3. Potwierdzają to badania przeprowadzone wędrówką poznawczą, gdzie czasy i pokonany dystans w tym narzędziu znacznie odbiegały od pozostałej grupy.

5. Wnioski

Główny problem przedstawiony w niniejszej pracy sprowadzał się do postawionej tezy: *Asana jest obecnie najlepszym rozwiązaniem dla członków zespołów programistycznych zarządzanych z wykorzystaniem klasycznych i lekkich metodyk zarządzania*. Aby potwierdzić lub obalić tezę, sformułowano trzy szczegółowe hipotezy badawcze. Otrzymane wyniki badań pozwoliły obalić tezę oraz uzyskać odpowiedź na sformułowane hipotezy.

Pierwszą hipotezą badawczą było pytanie: *Czy narzędzie Asana do zarządzania projektami posiada najlepsze funkcjonalności?* Analizując wyniki ankiety przeprowadzonej wśród osób korzystających z narzędzi do zarządzania projektami można zaobserwować, iż wśród zapytania o najbardziej znane narzędzie, Asana zajęła dopiero czwartą lokatę. Tą samą pozycję osiągnięto w zapytaniu o narzędzie, z którego osoba najczęściej korzysta. Odpowiedzi otrzymane dla Asany w dalszej części ankiety stanowiły zaledwie 7,4% wszystkich odpowiedzi o ocenę funkcjonalności narzędzia. Tak mały odsetek danych dla Asany, utrudnia ich analizę. Dokonując analizy funkcjonalnej narzędzia można zauważyć, iż posiada ono pewne ograniczenia funkcjonalne względem porównywanych narzędzi w wersji darmowej.

Drugą hipotezą badawczą było pytanie: *Czy interfejs użytkownika narzędzia Asana jest oceniany jako użyteczny wśród początkujących użytkowników?* Badanie wędrówką poznawczą wykazało, iż najbardziej przyjaznym interfejsem dla początkującego użytkownika okazało się narzędzie Trello. Osiągnęło ono najmniejszą liczbę problemów wynoszącą zaledwie 2,28% i było jedynym narzędziem, w którym trzy z sześciu osób wykonało poprawnie wszystkie zadania. Trello było narzędziem, w którym zgłoszono najmniej uwag słownych a błędy użytkowników wynikały z drobnych pomyłek. Jedynie w badaniu listą LUT Asana osiągnęła podium z wynikiem WUP wynoszącym 3,97, jednak różnice w porównaniu do pozostałych narzędzi były niewielkie. Wyjątek stanowiło narzędzie Bitrix24, osiągając WUP na poziomie tylko 2,82.

Trzecia hipoteza badawcza dotyczyła pytania: *Czy Asana jest najpopularniejszym narzędziem wśród użytkowników korzystających z niego w codziennej pracy?* Badanie ankietowe wykazało, iż najchętniej wybieranym narzędziem przez respondentów była Jira Software z wynikiem 40,7% głosów. Drugą lokatę zajęło narzędzie MS Project z wynikiem 14,8%, natomiast miejsce trzecie przypadło narzędziu Trello z wynikiem 11,1%. Łącznie odpowiedzi te stanowiły 66,6% zebranych ankiet. Ogólna ocena interfejsów narzędzi została oceniona wysoko a z komentarzy badanych osób

wynika, iż ww. narzędzia spełniają wymagania osób korzystających z nich w codziennej pracy.

Przeprowadzone badania i otrzymane rezultaty w pracy magisterskiej mogą być pomocne dla współczesnych organizacji korzystających z tego typu aplikacji. Wyniki badań w pracy pozwoliły na wykazanie podstawowych błędów dotyczących użyteczności interfejsów badanych narzędzi oraz mogą stanowić solidną podstawę przy projektowaniu nowych rozwiązań lub udoskonalaniu programów już istniejących.

Literatura

[1] T. L. Fox, J. W. Spence, Tools of the trade: A survey of project management tools, Project Management Journal 29(3) (1998) 20-27.

[2] P. Patanakul, B. Iewwongcharoen, D. Milosevic, An empirical study on the use of project management tools and techniques across project life-cycle and their impact on project success, Journal of General management 35(3) (2010) 41-66.

[3] A. Mishra, D. Mishra, Software project management tools: a brief comparative view, ACM SIGSOFT Software Engineering Notes 38(3) (2013) 1-4.

[4] D. Özkan, A. Mishra, Agile Project Management Tools: A Brief Comprative View, Cybernetics and Information Technologies 19(4) (2019) 17-25.

[5] Metoda PMBOK, https://inzynieria.com/budownictwo/wiadomosci/55874,metoda-zarzadzania-pmbok-dlaczego-warto-ja-wdrozyc-w-kazdej-firmie, [03.05.2022].

[6] R. Q. Gonçalves, C. G. Von Wangenheim, An instructional unit for teaching project management tools aligned with PMBOK, Proceedings of IEEE 29th International Conference on Software Engineering Education and Training (2016) 46-55.

[7] R. V. Anand, M. Dinakaran, Popular agile methods in software development: Review and analysis, International Journal of Applied Engineering Research 11(5) (2016) 3433-3437.

[8] R. da Camara, M. Marinho, S. Sampaio, S. Cadete, How do Agile Software Startups deal with uncertainties by Covid-19 pandemic?, International Journal of Software Engineering & Applications 11(4) (2020).

[9] Heurystyki Jacoba Nielsena, https://primeo.pl/pl/462/Audyty-uzytecznosci-Wedrowka-poznawcza-i-10-heurystyk-Nielsena-, [15.05.2022].

[10] M. Miłosz, Ergonomia systemów informatycznych, Wydawnictwo Politechnika Lubelska, Lublin, 2014.

[11] Wędrówka poznawcza, https://www.k2.pl/blog/wedrowka-poznawcza-10-heurystyk-nielsena-i-analiza-zachowan-uzytkownikow-czyli-o-audycie-ux, [20.05.2022].

[12] N. Alomar, N. Almobarak, S. Alkoblan, S. Alhozaimy, S. Alharbi, Usability engineering of agile software project management tools, Proceedings of International Conference of Design, User Experience, and Usability (2016) 197-208.

[13] M. Laskowski, Propozycje metodyk badania użyteczności interfejsów aplikacji, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska (4b) (2012) 21-24.

[14] L. O. Bligård, A. L. Osvalder, Enhanced cognitive walkthrough: Development of the cognitive walkthrough method to better predict, identify, and present usability problems, Advances in Human-Computer Interaction 4 (2013) 9-26.

[15] M. Weninger, P. Grünbacher, E. Gander, A. Schörgenhumer, Evaluating an Interactive Memory Analysis Tool: Findings from a Cognitive Walkthrough and a User Study, Proceedings of the ACM on Human-Computer Interaction (EICS) 4 (2020) 1-37.

[16] M. Miłosz, M. Plechawska-Wójcik, M. Borys, M. Laskowski, Quality improvement of ERP system GUI using expert method: A case study. 6th International Conference on Human System Interactions (2013) 145-152.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 265–272
Received: 14 June 2022
Accepted: 2 July 2022

Performance analysis of Laravel and Yii2 frameworks based on the MVC architectural pattern and PHP language

Analiza wydajności szkieletów programistycznych Laravel oraz Yii2 opartych na wzorcu architektonicznym MVC oraz języku PHP

Konrad Sławomir Węgrzecki*, Mariusz Dzieńkowski

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

Abstract
The subject of this paper is the performance analysis of two PHP programming frameworks in the latest versions - Laravel 9.6 and Yii 2.0.45. It was carried out with the help of test applications prepared for this purpose, which have identical functionalities: they generate all prime numbers from a given range and create a book ranking system based on CRUD operations. The request handling time of each application was used as a comparison criterion. To check the performance, dedicated debugbars - bottom bars displaying information about the executed request - were used. The results obtained after the tests showed that in terms of performance, Laravel is a better technology than Yii for building web applications.

Keywords: Laravel; Yii2; programming frameworks; performance comparative analysis

Streszczenie
Przedmiotem pracy jest analiza wydajnościowa dwóch szkieletów programistycznych języka PHP w najnowszych wersjach – Laravel 9.6 i Yii 2.0.45. Została ona przeprowadzona przy pomocy przygotowanych do tego celu aplikacji testowych, posiadających identyczne funkcjonalności. Aplikacje generują wszystkie liczby pierwsze z podanego zakresu oraz tworzą system rankingowy książek, oparty na operacjach CRUD. Jako kryterium porównawcze przyjęto czas obsługi żądań przez każdą aplikację. Do sprawdzenia wydajności użyto dedykowanych szkieletom tzw. debugbarów – dolnych pasków wyświetlających informację o wykonanym żądaniu. Wyniki uzyskane po przeprowadzonych badaniach wykazały, że biorąc pod uwagę wydajność, Laravel jest lepszą technologią niż Yii do budowy aplikacji internetowych.

Słowa kluczowe: Laravel; Yii2; szkielety programistyczne; analiza wydajności

*Corresponding author
Email address: **konrad.wegrzecki@pollub.edu.pl** (K. S. Węgrzecki)

1. Wstęp

Obecnie z oprogramowania można korzystać używając Internetu, albo posiadając zainstalowaną aplikację na urządzeniu mobilnym lub komputerze. Ostatnia opcja jest w tej chwili najrzadziej wybierana ze względu na czasochłonny proces instalacji oraz utrudnione przenoszenie danych. W związku z tym, najpopularniejszym rozwiązaniem stały się dziś aplikacje internetowe, ponieważ są one bezpieczne - nie przechowują danych na lokalnym urządzeniu oraz użytkownik nie musi ich samodzielnie instalować i modernizować.

Na rynku dostępnych jest bardzo dużo technologii umożliwiających tworzenie aplikacji internetowych. Wśród języków programowania dominującą pozycję na rynku w tym segmencie zajmują języki JavaScript, Python, Java, C# oraz PHP [1]. Na bazie języka PHP powstała największa liczba różnorodnych szkieletów programistycznych, które można zakwalifikować do jednej z trzech podgrup. Do pierwszej należą szkielety programistyczne posiadające rozbudowany zestaw narzędzi wspomagający tworzenie aplikacji oraz automatyzujący proces implementacji funkcjonalności. Drugą są szkielety, których pakiet narzędzi jest mniej rozbudowany, a programista ma większy udział w procesie tworzenia oprogramowania. Trzecią grupę stanowią systemy zarządzania treścią CMS (ang. Control Management System). Szkielety z tej grupy różnią się od szkieletów z dwóch pierwszych grup tym, że tworzenie aplikacji odbywa się głównie za pomocą wtyczek, a nie pisania kodu. Szkielety programistyczne języka PHP wchodzące w skład dwóch pierwszych grup wykorzystują wzorce architektoniczne, dzięki którym pisanie kodu przebiega szybciej, a proces tworzenia aplikacji jest tańszy. Obecnie najpopularniejszym wzorcem, ze względu na swoją prostotę oraz szybkość działania, jest MVC (ang. Model View Controller).

W ramach niniejszej pracy została przeprowadzona analiza porównawcza dwóch technologii do budowy aplikacji webowych opartych na języku PHP. Do porównań wybrano Laravela - najpopularniejszy szkielet programistyczny należący do pierwszej oraz Yii - najpopularniejszy szkielet należący do drugiej grupy [2].

2. Przegląd literatury

Duża liczba technologii pozwalających na tworzenie aplikacji po stronie serwera doprowadza do częstych dyskusji i rozważań na temat jakości oraz użyteczności technologii. Spowodowało to powstanie wielu publika-

cji naukowych poruszających tematykę tworzenia aplikacji internetowych. Szkielety programistyczne Laravel i Yii były już analizowane w pracy *Comparison Between Yii Frameworks and Laravel in 3 Different Version for Viewing Large Data of Shipyard Industry in Indonesia*. Na potrzeby publikacji zostały wykonane aplikacje testowe bazujące na tych dwóch szkieletach programistycznych i oparte na dwóch wersjach języka PHP [3]. W badaniach pod uwagę wzięto kilka miar: czas wykonania żądania, czas reakcji serwera, rozmiar wykorzystywanej pamięci oraz liczbę przetwarzanych rekordów przez aplikację testową w ciągu jednej sekundy. Wyniki badań pokazały, że użycie wersji 7 języka PHP poprawia wydajność nawet dwukrotnie w przypadku obu szkieletów programistycznych. Poza tym okazało się, że wydajniejszym szkieletem o około 7% jest Laravel.

Publikacja *Selecting a PHP framework for a web application project-The method and case study* podejmuje tematykę analizy sześciu szkieletów programistycznych opartych na języku PHP: Laravel, Symfony, CodeIgniter, Yii2, Zend, CakePHP [4]. Autorzy pracy w swoich badaniach podeszli do tematu szeroko, zaczynając od badań popularności szkieletów, poprzez zidentyfikowanie wspieranych przez nie technologii, na testach opracowanych aplikacji kończąc. Do oceny szkieletów wykorzystano system punktowy w skali od 0 do 1. Oceny dokonano w dziewięciu kategoriach. Wyniki pozwoliły na wskazanie najbardziej uniwersalnego rozwiązania, którym okazał się zdaniem autorów Yii2. Szkielet ten uzyskał 76 procent maksymalnych punktów, dzięki czemu wypadł znacznie lepiej niż Laravel, który zdobył tylko 31 procent.

Ostatnią pracą dotyczącą analizy szkieletów programistycznych Yii2 i Laravel jest *Porównanie możliwości tworzenia aplikacji PHP na przykładzie Yii2 i Laravel*. Na potrzeby badań zostały przygotowane aplikacje, które następnie analizowano pod względem ich struktury, łatwości i możliwości tworzenia interfejsu graficznego oraz współpracy z systemami bazodanowymi [5]. Oba szkielety były oceniane w ośmiu kategoriach w skali punktowej od 1 do 5. Końcowy wynik aplikacji opracowanej w Laravelu to 33 punkty, natomiast Yii2 - 31 punktów. Według autorki szkielet programistyczny Laravel jest bardziej zautomatyzowany i to on powinien być pierwszym wyborem dla mniej doświadczonych programistów, choć Yii2 jest jego silnym konkurentem.

Wyniki badań omówionych prac [3-5] nie pozwalają na jednoznaczne wskazanie technologii lepszej lub takiej która jest bardziej uniwersalna. Podczas zrealizowanych porównań były brane pod uwagę: wymagania technologiczne, stopień trudności tworzenia oprogramowania, obsługa systemów bazodanowych, szybkość działania aplikacji, wsparcie technologii poprzez dokumentację, popularność, użyta architektura oraz organizacja kodu. Głównym wnioskiem płynącym z tych prac jest to, iż wybór szkieletu programistycznego powinien być uzależniony od przeznaczenia tworzonej aplikacji. Laravel jest zdecydowanie prostszym szkieletem do rozpoczęcia programowania, ponieważ struktura kodu jest bardziej zwięzła, a cały proces bardziej zautomatyzowany. Yii2 stanowi jednak poważną konkurencję jako szkielet posiadający największą liczbę wspieranych technologii.

W publikacjach [6-7] większy nacisk został położony na stronę koncepcyjną tworzenia aplikacji, czyli zastosowane rozwiązania w danym szkielecie, strukturę kodu, możliwości rozszerzania aplikacji oraz ich wsparcie poprzez dokumentację czy specjalistyczne fora dyskusyjne. Szczególną uwagę zwrócono na wsparcie dla innych technologii, takich jak systemy bazodanowe czy system szablonów szaty graficznej oraz zastosowane ułatwienia dla programisty. W zależności od pracy wyniki końcowe różnią się od siebie, głównie ze względu na przyjęty system oceniania. Powiązane jest to z tym, że część elementów była oceniana subiektywnie przez autora danej pracy.

Autorzy publikacji [8-10] porównali szkielety programistyczne na podstawie aplikacji testowych używając przyjętych przez siebie kryteriów. Dotyczyły one różnic w budowie oraz wydajności poszczególnych szkieletów programistycznych. Do głównych miar użytych w tych porównaniach należały czasy reakcji serwera, czasy wykonania zadania zleconego użytkownikom oraz czasy odpowiedzi serwera. Dodatkowo badania te często były wykonywane pod różnym wirtualnym obciążeniu aplikacji testowych, w celu sprawdzenia jej pod kątem przyszłego użytkowania. Badania prowadzone były na aplikacjach wykorzystujących bazę danych. Zazwyczaj weryfikowany był odczyt wielu rekordów oraz zapis, edycja i usuwanie pojedynczego rekordu. Testy takich aplikacji pozwoliły zbadać szkielet programistyczny w środowisku pracy przypominającym funkcjonowanie przeciętnej aplikacji internetowej.

Porównywanie technologii wytwarzania aplikacji internetowych stało się ostatnio dosyć popularne. Autorzy prac w swoich badaniach poruszają kwestie technologiczne i wydajnościowe, celem wybrania najbardziej uniwersalnego rozwiązania. Działania te mają pomóc programistom wybór technologii zanim przystąpią do budowy systemu. W ramach niniejszej pracy autor także zajął się problemem analizy narzędzi do tworzenia aplikacji internetowych, poprzez utworzenie, a następnie analizę działania aplikacji testowych. Tym co odróżnia zrealizowane badania od badań wykonanych przez innych autorów są użyte wersje szkieletów programistycznych, użyta wersja języka PHP oraz przygotowane aplikacje testowe zawierające niezbędne funkcjonalności, na których podstawie będzie możliwa pełna ocena porównywanych technologii.

3. Cel i hipoteza pracy

Celem pracy jest analiza porównawcza wydajności szkieletów programistycznych w ich najnowszych wersjach w momencie pisania pracy czyli Laravel 9.6 i Yii 2.0.45. Za główną miarę wydajności przyjęto średni czas realizacji zadania przez aplikację opracowaną za pomocą danego szkieletu.

Za hipotezę badawczą przyjęto stwierdzenie:

Szkielet programistyczny Laravel może być lepszym rozwiązaniem niż Yii, ze względu na jego większą wydajność oraz bardziej zautomatyzowany proces tworzenia aplikacji.

4. Metodologia badań

4.1. Aplikacje testowe

Do przeprowadzenia badań wymagane było przygotowanie dwóch aplikacji testowych posiadających identyczny zestaw funkcjonalności - jedna aplikacja wykonana była przy pomocy szkieletu Laravel oraz druga przy pomocy Yii2. Obie aplikacje korzystały z tej samej bazy danych oraz wykorzystywały wyłącznie język znaczników HTML do budowy widoków. Porównywane aplikacje testowe posiadają dwa oddzielne moduły. Pierwszy z nich służy do wyznaczania liczb pierwszych z podanego przez użytkownika zakresu. Do tego celu nie został wykorzystany żaden optymalny algorytm, a jedynie podwójna, zagnieżdżona pętla z przerwaniem, w momencie gdy dzielnik liczby zostanie znaleziony. Ten prosty, naiwny algorytm realizuje maksymalną liczbę przejść i powoduje maksymalne obciążenie. Drugim modułem jest system rankingowy książek, oparty na operacjach CRUD, realizowanych na bazie danych oraz zawierający system uwierzytelniania użytkowników z podziałem na role użytkownika oraz administratora. Moduł ten został pozbawiony atrakcyjnej warstwy wizualnej, bowiem zrealizowano ją za pomocą podstawowych znaczników HTML, które umożliwiły także interakcję z użytkownikiem.

Wymagania funkcjonalne aplikacji testowych

- Zwykły użytkownik:
  - tworzenie konta,
  - logowanie w systemie,
  - wylogowanie,
  - przeglądanie listy książek,
  - przeglądanie rankingu książek,
  - przeglądanie szczegółów książki,
  - ocena książki.
- Administrator – oprócz funkcji użytkownika może również:
  - dodawać książki,
  - edytować książki,
  - usuwać książki.

Struktura bazy danych

Baza danych opracowana na potrzeby aplikacji testowych składa się z trzech tabel przechowujących wszystkie niezbędne dane do przeprowadzenia testów. W skład bazy danych wchodzą tabele:

- *users* – przechowująca dane kont użytkowników,
- *books* – zawierająca szczegółowe dane książek,
- *rating* – zawierająca oceny książek przez użytkowników systemu.

Za wyjątkiem danych logowania wszystkie dane w bazie danych zostały wygenerowane losowo według określonego klucza.

- Tabela *books*:
  - tytuł książki – ciąg losowy (15 znaków),
  - imię i nazwisko autora – generowane funkcją *faker*,
  - rok wydania – liczba losowa z przedziału od 1980 do 2022,
  - opis – losowy ciąg o długości od 150 do 250 znaków.
- Tabela *ratings*:
  - identyfikator użytkownika – liczba losowa z przedziału od 1 do 100,
  - identyfikator książki – liczba losowa z przedziału od 1 do największego identyfikatora książki,
  - ocena książki – liczba losowa z zakresu od 1 do 10.

4.2. Scenariusze testowe

Analizę porównawczą wydajności szkieletów programistycznych Laravel i Yii2 przeprowadzono za pomocą scenariuszy testowych, podczas których dokonano pomiarów czasów potrzebnych do:

- wyznaczenia liczb pierwszych do wartości: 100, 500, 1000, 5000, 10000, 50000, 100000,
- wyświetlenia listy książek zawierającej 100, 500, 1000, 5000, 10000 pozycji,
- wyświetlenia rankingu książek zawierającego: 100, 200, 500, 1000, 2000 książek oraz odpowiednio: 500, 1000, 2500, 5000, 10000 ocen,
- wyświetlenia, dodania, edycji i usunięcia książki znajdującej się na pozycji: 100, 500, 1000, 5000, 10000 w bazie danych,
- dodania oceny książki przy zapełnieniu tabeli ocen liczbą rekordów: 500, 1000, 2500, 5000, 10000.

Podczas eksperymentu czas liczony był od momentu otrzymania żądania przez daną aplikację do zakończenia pracy i wysłania odpowiedzi do przeglądarki. W celu zobiektywizowania wyników, pomiary były wykonywane na dwóch urządzeniach, na których przeprowadzono po 5 prób każdego z testów, a następnie wyniki uśredniono dla każdego urządzenia.

Wydajność rozwiązania nie jest jedynym aspektem branym podczas wyboru technologii. Dodatkowymi, istotnymi kwestiami są:

- możliwości technologii – zastosowane rozwiązania, wspierane technologie zewnętrzne (np. systemy bazodanowe) oraz usprawnienia dla programisty,
- jakość dokumentacji technicznej,
- wsparcie społeczności – zrzeszanej na forach tematycznych np. Stack Exchange i ogólnych np. grupy na Facebook'u i LinkedIn,
- objętość kodu źródłowego – pisanego przez programistę oraz generowanego przez narzędzia wbudowane, niezbędnego do zapewnienia wymaganych funkcjonalności, który może być oszacowany za pomocą biblioteki *sloc* pochodzącej z repozytorium pakietów *npm* [11].

4.3. Środowisko testowe

Obie aplikacje testowe zostały uruchomione na dwóch niezależnych środowiskach testowych. Różnica w wydajności procesorów użytych komputerów w testach

syntetycznych wyniosła 30% [12]. Obydwa środowiska posiadały zainstalowany pakiet XAMPP oraz przeglądarkę Google Chrome w tych samych wersjach. Jako narzędzia testowe użyto dedykowanych debugbarów wbudowanych w porównywane szkielety. Debugbary to narzędzia deweloperskie pozwalające dokładnie sprawdzić działanie aplikacji na wszystkich jej etapach. Umożliwiają także pomiar czasu, w którym aplikacja realizuje określony scenariusz. Debugbary są narzędziami dokładnymi, odpornymi na opóźnienia oraz wpływ czynników zewnętrznych. Wersje użytych technologii oraz parametry środowisk testowych zawarte są w tabelach 1, 2 i 3.

Tabela 1: Specyfikacja użytych technologii

Wersja Google Chrome	99.0.4844.74
Wersja pakiet XAMPP	8.1.4
Wersja serwer Apache	2.4.48
Wersja PHP	8.0.7
Wersja MySQL	10.4.19

Tabela 2: Specyfikacja pierwszego środowiska testowego

Procesor	Intel Core i7-9750H
System operacyjny	Windows 10 Home 64 bit wersja 21H2
Pamięć RAM	16 GB, 2666MHz
Dysk	NVMeSKHynix_HSF256

Tabela 3: Specyfikacja drugiego środowiska testowego

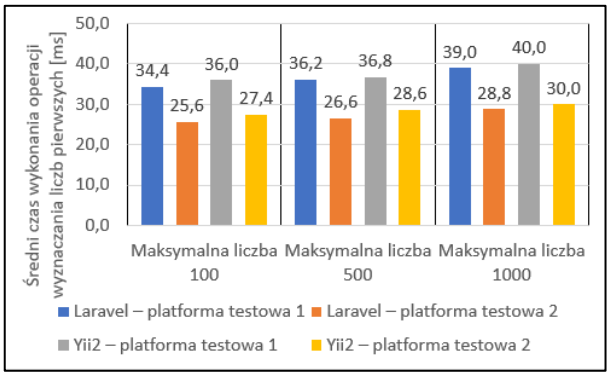
Procesor	Intel Core i7-11700K
System operacyjny	Windows 10 Pro 64 bit wersja 21H2
Pamięć RAM	16 GB, 3600MHz
Dysk	WDS100T3X0C-00SJG0

5. Wyniki badań

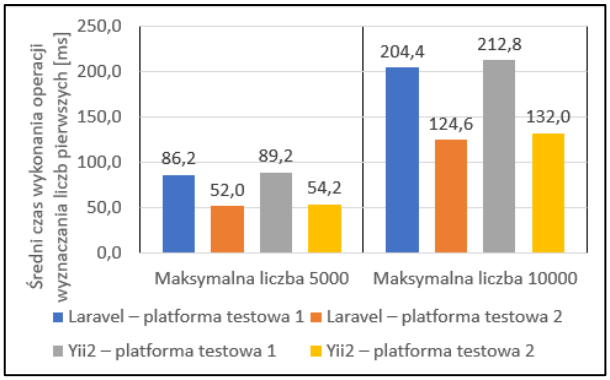
5.1. Eksperyment 1 - pomiar czasów wyznaczania liczb pierwszych

Uśrednione wyniki czasów wykonywania operacji wyznaczania liczb pierwszych, wyrażone w milisekundach, przedstawiają rysunki 1-3. Wykresy pokazują, że wyniki dla obu testowanych aplikacji są porównywalne niezależnie na jakiej platformie badawczej były wykonane.

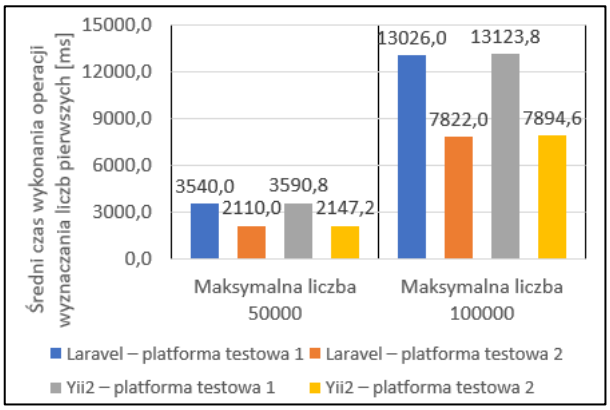
We wszystkich siedmiu przebadanych przypadkach szkielet programistyczny Laravel okazał się rozwiązaniem wydajniejszym, choć różnice były niewielkie. Na pierwszej platformie testowej maksymalna zaobserwowana różnica w wydajności między szkieletami to 5%, a na drugiej 8%. Wyznaczanie liczb pierwszych powyżej liczby 5000 powoduje, że różnice w czasach wyraźnie rosną.



Rysunek 1: Wyniki pomiarów wyznaczania liczb pierwszych dla wartości 100, 500 i 1000.



Rysunek 2: Wyniki pomiarów wyznaczania liczb pierwszych dla wartości 5000 i 10000.



Rysunek 3: Wyniki pomiarów wyznaczania liczb pierwszych do wartości 50000 i 100000.

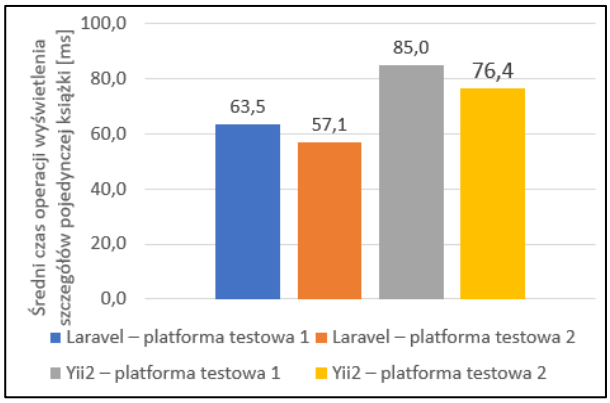
Porównując platformy testowe podczas wyznaczania liczb pierwszych na platformie 2 (mocniejszej) do wartości 100, 500 i 1000 wydajność Laravela wzrosła o 26,13% natomiast Yii do 23,73%. W przypadku wyznaczania liczb pierwszych dla zakresu 5000, 10000, 50000, 100000 nastąpił w równym stopniu wzrost wydajności dla obu platformy - dla Laravela o 39,8%, a dla Yii2 o 39,32%.

5.2. Pomiar czasów operacji CRUD

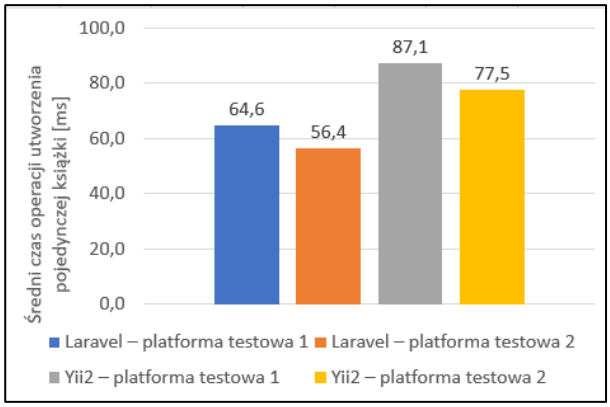
Operacje wykonywane przez moduł rankingowania książek podzielono na te, które były przeprowadzane na jednym rekordzie oraz te, które były realizowane na wielu rekordach.

Operacje wykonywane na jednym rekordzie

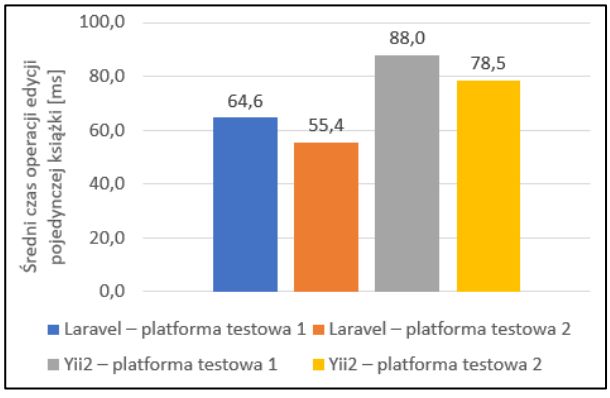
Operacje dodawania, wyświetlania, edycji i usuwania książki oraz dodawania oceny sprawdzono przy różnych poziomach zapełnienia bazy danych. Miało to jednak niezauważalny wpływ na czas realizacji zadania przez aplikację. W przypadku szkieletu programistycznego Laravel maksymalne odchylenie od średniej wynosiło 3%, a Yii2 2%. Wyniki tych operacji, wyrażone w milisekundach, przedstawiają rysunki 4-8.



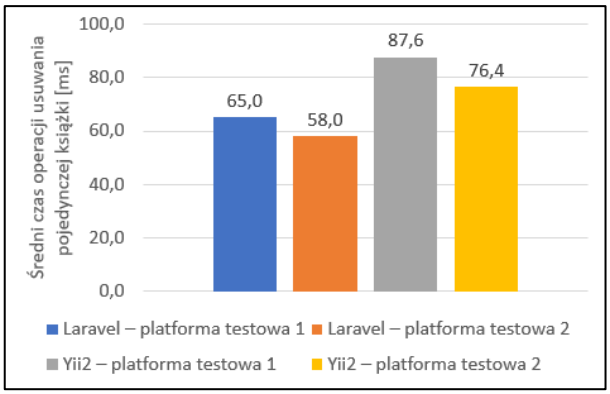
Rysunek 4: Wyniki pomiarów wyświetlenia książki.



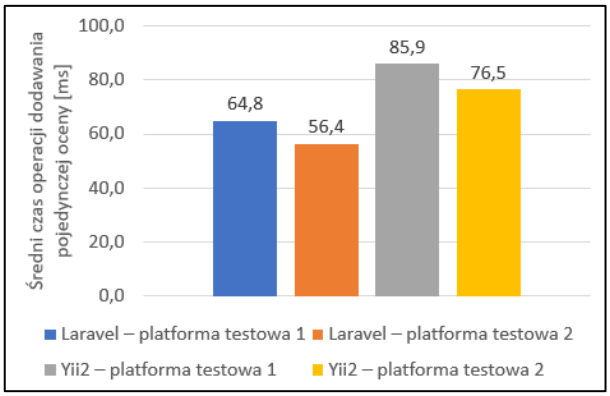
Rysunek 5: Wyniki pomiarów dodania książki.



Rysunek 6: Wyniki pomiarów edycji książki.



Rysunek 7: Wyniki pomiarów usunięcia książki.

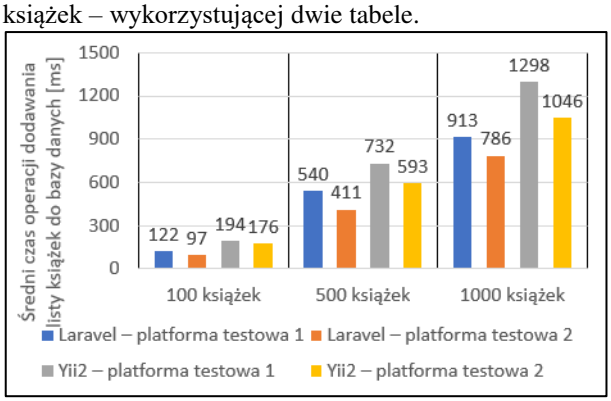


Rysunek 8: Wyniki pomiarów dodania oceny.

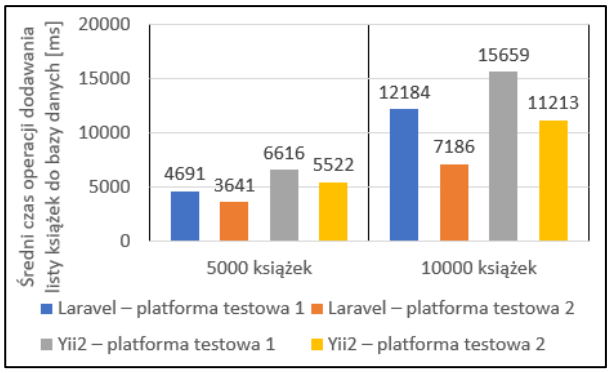
Wyniki badań pokazują większą wydajność aplikacji zbudowanej na bazie szkieletu Laravel w każdej z badanych operacji. W przypadku pierwszego stanowiska testowego średnia różnica czasu to około 38%, a w przypadku drugiego - 35% na korzyść Laravel'a. Prawdopodobnie jest to spowodowane zastosowaniem ORM (ang. Object-Relation Mapping) Eloquent, który jest znacznie wydajniejszy niż ActiveRecord stosowany w Yii2 od pierwszej jego publikacji. Zastosowanie wydajniejszej platformy testowej w tym wypadku działa na korzyść Laravela, który notuje 13% wzrostu wydajności, a Yii2 11%.

Operacje wykonywane na wielu rekordach

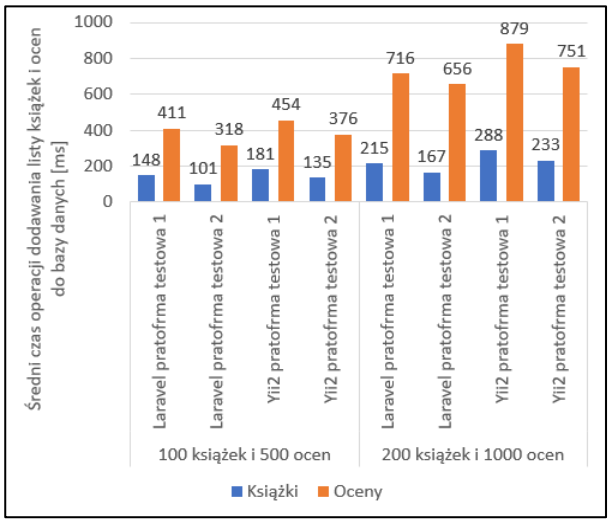
Operacjami wykonywanymi na wielu rekordach były zapis oraz wyświetlanie. Dotyczą one listy książek – operacji na jednej tabeli w bazie danych oraz rankingu książek – wykorzystującej dwie tabele.



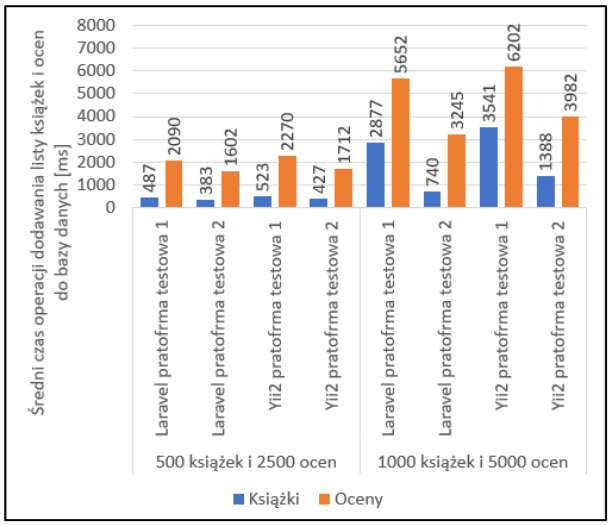
Rysunek 9: Wyniki pomiarów dodania 100, 500 i 1000 książek.



Rysunek 10: Wyniki pomiarów dodania 5000 i 10000 książek.

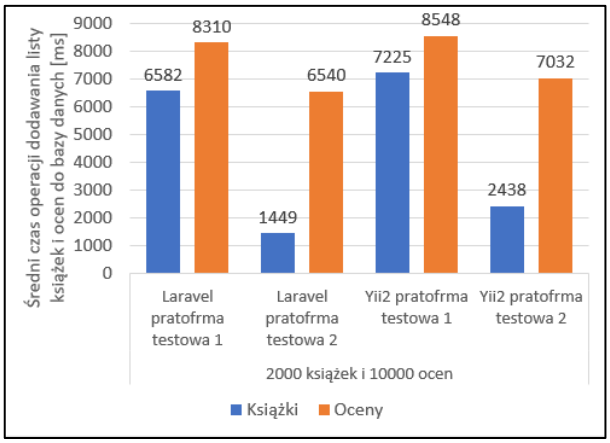


Rysunek 11: Wyniki pomiarów dodania 100 i 200 książek oraz odpowiednio 500 i 1000 ocen.



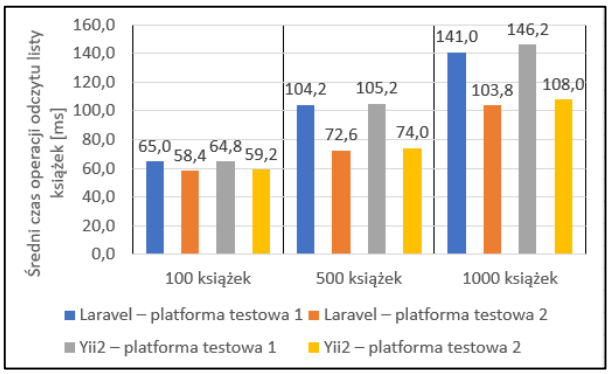
Rysunek 12: Wyniki pomiarów dodania 500 i 1000 książek oraz odpowiednio 2500 i 5000 ocen.

W przypadku zapełniania tabel użyto funkcji *random* i *faker* dla obydwu szkieletów programistycznych. Pozwalają one na generowanie wartości, które zostały zapisane w bazie danych. W przypadku Laravel'a wykorzystano dedykowane narzędzie do zapełniania bazy danych – *seeder*. Yii2 nie posiada jednak podobnego pakietu. Z tego względu przygotowana została metoda o identycznej funkcjonalności jak w szkielecie Laravel z użyciem kontrolera. Wyświetlanie pozycji odbywa się w ten sam sposób niezależnie od szkieletu. W przypadku rankingu poza samym pobraniem danych z bazy jest również liczona średnia ocen dla każdej książki. Wyniki czasów wyrażone w milisekundach zapisu danych zawierają rysunki 9-13 natomiast odczytu 14-17.

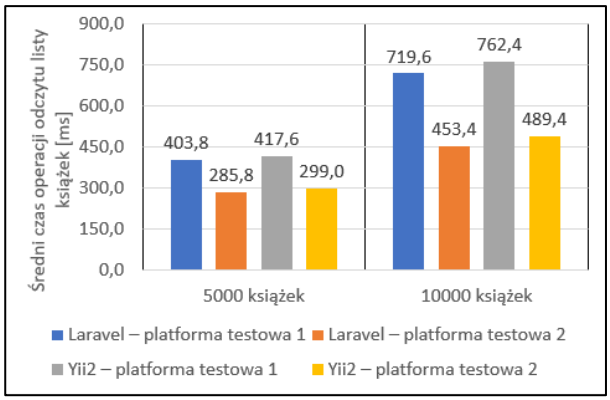


Rysunek 13: Wyniki pomiarów dodania 2000 książek i 10000 ocen.

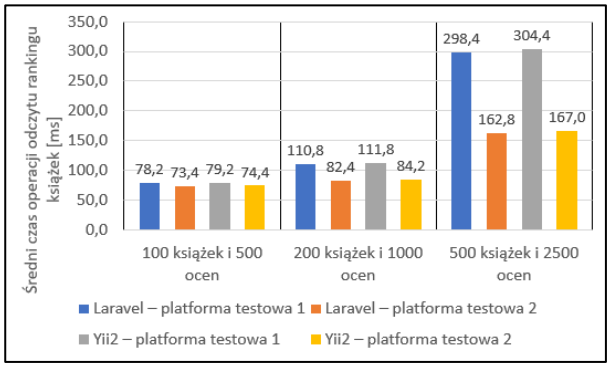
Wyniki tej części badań ponownie wykazują większą wydajność po stronie szkieletu programistycznego Laravel. Rezultaty zapełniania tabeli książek na pierwszej platformie testowej ukazują maksymalną różnicę w wydajności między szkieletami na poziomie 59%, a na drugiej platformie nawet 81%. W przypadku zapisywania danych do dwóch tabel różnice nie są tak duże, chociaż wyraźnie. Wyniki badań przeprowadzonych na urządzeniu pierwszym wykazują różnice między szkieletami Laravel – Yii2 od 7% do 25% dla tabeli książek i od 3% do 19% dla tabeli ocen. Dla drugiej platformy testowej różnice wydajności zwiększają się względem pierwszej i oscylują pomiędzy 10% a 47% dla tabeli książek oraz 6% do 19% dla tabeli ocen. W tym wypadku znaczna większość straty wydajności spowodowana jest zastosowaniem ORM Eloquent dla Laravel'a. Drugą mniej znaczącą różnicą jest dedykowane rozwiązanie do zapełniania bazy danych w szkielecie Laravel – *seeder*.



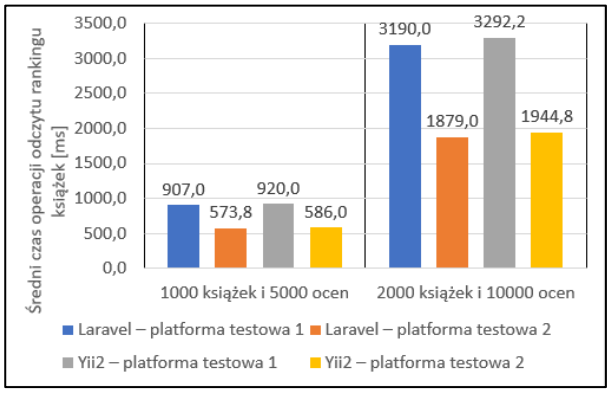
Rysunek 14: Wyniki pomiarów wyświetlenia 100, 500 i 1000 książek.



Rysunek 15: Wyniki pomiarów wyświetlenia 5000 i 10000 książek.



Rysunek 16: Wyniki pomiarów wyświetlenia 100, 200 i 500 książek oraz odpowiednio 500, 1000 i 2500 ocen.



Rysunek 17: Wyniki pomiarów dodania 1000 i 2000 książek oraz odpowiednio 5000 i 10000 ocen.

W przypadku masowego odczytu danych wydajniejszy był Laravel. Różnica uzyskanych czasów była jednak mniejsza niż dla masowego zapisu danych. W przypadku wyświetlania 100 książek Yii2 uzyskał średni czas lepszy o 0,2 milisekundy. Dla wszystkich innych pomiarów wydajność szkieletu Laravel jest wyższa o 6% na pierwszej platformie testowej oraz 7% na drugiej. Różnica zwiększa się w przypadku obydwu platform testowych im więcej danych jest przetwarzanych. Rozbieżność wydajności dla rankingu jest mniejsza i oscyluje pomiędzy 1% a 3% niezależnie od platformy testowej. Mniejsza dysproporcja wynika ze sposobu w jaki zastosowane ORM'y pracują na danych. ActivRecords pod względem odczytu dużej ilości danych częściowo dorównuje Eloquent. Dysproporcja uzyskanych wyników zwiększa się na niekorzyść Yii2 im większa jest baza danych.

5.3. Poza wydajnościowe kryteria oceny szkieletów programistycznych

Głównym celem w pracy jest badanie wydajności szkieletów programistycznych, jednak istotne są też inne kryteria. W związku z tym podrozdział ten został poświęcony analizie takich kryteriów jak: możliwości technologii, dokumentację techniczną, wsparcie społeczności oraz objętość kodu źródłowego. Parametry te pomimo braku wpływu na wydajność mają duże znacznie podczas wyboru technologii.

Możliwości technologii

Szkielet programistyczny Yii2 cechuje się większą swobodą podczas doboru technologii wspomagających. Pozwala on na zastosowanie większej liczby systemów bazodanowych, technologii tworzenia warstwy wizualnej oraz wspieranych narzędzi do pisania testów jednostkowych. Instalacja jest możliwa przy zastosowaniu starszej wersji serwera PHP. Elementy takie jak domyślny system szablonów, ORM oraz sposób debugowania różnią się w tych szkieletach jednak nie zapewniają dodatkowych funkcjonalności. Yii2 posiada generator kodu Gii, który pozwala wygenerować modele, kontrolery a nawet kompletne metody CRUD wraz z widokami. W szkielecie Yii2 bardziej rozbudowane elementy nie posiadają znaczących usprawnień. Laravel pomimo braku tak rozbudowanych udogodnień wprowadza usprawnienia dla programisty na każdym etapie pisania aplikacji.

Dokumentacja techniczna

Dokumentacje obydwu szkieletów programistycznych pod względem struktury są zbudowane bardzo podobnie. W przypadku Yii2 dokumentacja wyjaśnia szczegółowo tylko główne metody szkieletu. Dla bardziej szczegółowych danych przedstawiona jest tylko logika funkcji bez dokładniejszego wyjaśnienia oraz przykład kodu źródłowego pozwalającego osiągnąć określoną funkcjonalność. Dokumentacja szkieletu programistycznego Laravel zawiera szczegółowe wyjaśnienia oraz wiele przykładów użycia kodu. Dla każdej funkcji i metody można wybrać pożądaną wersję szkieletu co zwiększa wygodę pracy użytkownika. Wartą uwagi zaletą szkieletu Yii2 jest to, że posiada on dokumentację w języku polskim i innych 11 językach, natomiast Laravel ma dokumentację jedynie w języku angielskim.

Wsparcie społeczności

Największa platforma zrzeszająca wiele różnych forów tematycznych – Stack Echange pozwala sprawdzić liczbę zapytań posiadających określoną frazę. Na dzień 22.05.2022 roku Laravel posiadał 544878 wątków na wszystkich forach a Yii2 43359. Podobne dysproporcje wykazuje najpopularniejsze internetowe narzędzie interakcji międzyludzkich w Polsce – Facebook. Na dzień 22.05.2022 roku polskojęzyczna grupa Laravel'a posiada 1286 członków a Yii2 10. Na grupach anglojęzycznych jest to 59982 użytkowników Laravela i 3440 w przypadku Yii2. Podobna sytuacja występuje w przy-

padku LinkedIn – serwisu społecznościowego dedykowanemu kontaktom zawodowym. Na dzień 22.05.2022 roku grupa Laravela zrzeszała około 18000, a Yii2 1100 członków.

Objętość kodu źródłowego

Dla obydwu aplikacji testowych zliczona została liczba linii kodu. Zostało zastosowane jednakowe formatowanie kodu w celu otrzymania możliwie najdokładniejszych wyników. Do zaimplementowania funkcjonalności opisanych w rozdziale 4.1 Laravel wymagał napisania 743 linii kodu a Yii2 812. Dla testowanych aplikacji Laravel wymagał o 8,5% mniej linii kodu. W przypadku Yii2 znacznie większa część została wygenerowana.

6. Wnioski

Analiza wydajności szkieletów programistycznych Laravel i Yii2 wykazała wyraźną przewagę pierwszego z nich. W 31 na 32 przypadki testowe wydajniejszy okazał się Laravel. Operacje wykonywane bez wykorzystania bazy danych wymagały zbliżonych czasów z przewagą Laravel'a o maksymalnie 8%. Większą dysproporcję wydajności wykazują operacje CRUD na pojedynczych rekordach gdzie różnica wydajności wynosi nawet 26%. Masowy odczyt danych wykazuje różnice do 7% wydajności podczas wyświetlania listy książek i 3% dla rankingu. Zapis danych do bazy wynika z opóźnienienia technologicznego szkieletu Yii2. Różnice w wydajności wzrastają do poziomu maksymalnego 81% dla pojedynczej tabeli oraz 47% w przypadku dwóch tabel. Ponadto niemal w każdym wypadku różnice wydajności wzrastają w przypadku zastosowania wydajniejszej platformy testowej. Wyniki pozwalają stwierdzić zdecydowaną przewagę szkieletu programistycznego Laravel pod kątem wydajności.

Analiza dodatkowych parametrów oceny wydajności nie pozwala jasno wskazać lepszego narzędzia. Zaletą Yii2 jest większa dowolność podczas wyboru technologii wspierających np. system bazodanowy. W przypadku zastosowania rozwiązań konkurencyjnych np. ORM większą wydajnością cechują się te zastosowane w szkielecie Laravel. Jakość dokumentacji technicznej w obydwu przypadkach jest bardzo wysoka. Większą przejrzystość dla niedoświadczonego programisty zapewnia szkielet programistyczny Laravel. Posiada też on zdecydowanie większą społeczność co widać na przykładzie grup Facebookowych oraz liczby zapytań na stronie Stack Exchange. Parametrem, którego nie można określić jako jednoznacznie lepszy jest objętość kodu źródłowego. W przypadku szkieletu Yii2 linii kodu jest więcej, jednak moduł Gii wygenerował znaczną jego większość. Szkielet Laravel również pozwala generować fragmenty kodu, jednak nie posiada dedykowanego rozwiązania. Cechuje się on jednak bardziej zwięzłą strukturą kodu źródłowego. Podsumowując dodatkowe parametry oceny można stwierdzić, że Yii2 jest odpowiednim wyborem dla prostych aplikacji pisanych przez programistów z doświadczeniem w pisaniu aplikacji. Laravel jest odpowiednio przygotowany do tworzenia rozbudowanych aplikacji oraz dla mniej doświadczonych programistów.

Uzyskane wyniki oraz analiza dodatkowych parametrów potwierdzają postawioną na początku pracy hipotezę, że Laravel wydajnościowo jest lepszym rozwiązaniem. Szkielet Yii2 może być konkurencyjny tylko w wypadku prostych aplikacji realizujących podstawowe operacje CRUD dzięki modułowi generowania kodu Gii, gdzie wydajność nie jest głównym kryterium podczas wyboru technologii. W każdym innym aspekcie lepszą technologią jest Laravel ze względu na swoją wydajność oraz zautomatyzowany proces tworzenia funkcjonalności i to on powinien być pierwszym wyborem podczas doboru technologii przez programistę.

Literatura

[1] Dane statystyczne wykorzystywania języków programistycznych w 2021 roku, https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/, [22.05.2022].

[2] Dane statystyczne wykorzystywania szkieletów programistycznych języka PHP w 2021 roku, https://www.jetbrains.com/lp/devecosystem-2021/php/, [22.05.2022].

[3] U. K. Latif, T. F. Kusumasari, Comparison Between Yii Frameworks and Laravel in 3 Different Version for Viewing Large Data of Shipyard Industry in Indonesia, International Journal of Innovation in Enterprise System 2 (2018) 13-18.

[4] A. Zurkiewicz, M. Milosz, Selecting a PHP framework for a web application project-The method and case study. In Conference: 9th International Technology, Education and Development Conference (2015).

[5] O. Sydorchuk, Porównanie możliwości tworzenia aplikacji PHP na przykładzie Yii2 i Laravel, Journal of Computer Sciences Institute 11 (2019) 125-130.

[6] K. Benmoussa, M. Laaziri, S. Khoulji, K. M. Larbi, A. El Yamami, A new model for the selection of web development frameworks: application to PHP frameworks, International Journal of Electrical and Computer Engineering (IJECE) 9(1) (2019) 695 – 703.

[7] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi, A. El Yamami, A comparative study of laravel and symfony PHP frameworks. International Journal of Electrical and Computer Engineering (IJECE) 9(1) (2019) 704-712.

[8] K. Kuflewski, M. Dzieńkowski, Symfony and Laravel - acomparative analysis of PHP programming frameworks. Journal of Computer Science Institute 21 (2021) 367-372.

[9] D. Drabik, Comparison of new ways of creating PHP applications using Laravel and CodeIgniter example. Journal of Computer Sciences Institute 10 (2019) 71-76.

[10] N. Prokofyeva, V. Boltunova, Analysis and practical application of PHP frameworks in development of web information systems, Procedia Computer Science 104 (2017) 51-56.

[11] Opis biblioteki SLOC, https://www.npmjs.com/package/sloc, [22.05.2022].

[12] Porównanie wydajności platform testowych, https://cpu.userbenchmark.com/Compare/Intel-Core-i7-9750H-vs-Intel-Core-i7-11700K/m766364vs4107, [22.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 273–279
Received: 15 June 2022
Accepted: 22 June 2022

Comparative analysis of Java and Dart programming languages in terms of suitability for creating mobile applications

Analiza porównawcza języków programowania Java oraz Dart pod kątem przydatności do tworzenia aplikacji mobilnych

Łukasz Kozłowski*, Grzegorz Kozieł

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

Abstract
This paper presents the results of a comparative analysis of Java and Dart programming languages in terms of suitability for creating mobile applications. The research was carried out on two proprietary applications with identical functionalities, which were implemented in the analyzed languages. The analysis covers areas such as: CPU load, RAM consumption, battery consumption and program execution time. Additionally, on the basis of proprietary applications, the code structure, number of lines of code and community support were considered. The results of the analysis indicate that it is difficult to clearly determine which language is more efficient, so the choice between Java and Dart should be analytical and best suited to the requirements of a given application.

Keywords: Java; Dart; Android

Streszczenie
W niniejszej pracy przedstawiono wyniki analizy porównawczej języków programowania Java i Dart pod kątem przydatności do tworzenia aplikacji mobilnych. Badania przeprowadzono na dwóch autorskich aplikacjach o identycznych funkcjonalnościach, które zostały zaimplementowane w rozpatrywanych językach. Analiza obejmuje obszary takie jak: obciążenie procesora, zużycie pamięci RAM, zużycie baterii oraz czas wykonania programów. Dodatkowo na podstawie autorskich aplikacji rozpatrzono budowę kodu, liczbę linii kodu i wsparcie społeczności. Rezultaty analizy wskazują, że trudno jednoznacznie określić, który język jest wydajniejszy, dlatego wybór pomiędzy językiem Java, a Dart powinien być analityczny i jak najlepiej dopasowany do wymagań danej aplikacji.

Słowa kluczowe: Java; Dart; Android

*Corresponding author
Email address: lukasz.kozlowski@pollub.edu.pl (Ł. Kozłowski)

1. Wstęp

Dynamiczny rozwój technologii sprawił, że urządzenia mobilne są nierozłącznym elementem codziennego życia dla wielu ludzi. Do funkcjonowania smartfonów niezbędne są aplikacje mobilne, które współcześnie występują niemal w każdej dziedzinie życia. Ze względu na dynamiczny rozwój technologii konieczne jest udoskonalanie aplikacji mobilnych, aby przystosować je do pracy na coraz wydajniejszych smartfonach z różnorodnymi systemami operacyjnymi.

Na dzień dzisiejszy można wyróżnić dwa najpopularniejsze systemy operacyjne przeznaczone na urządzenia mobilne: Android oraz iOS, gdzie ponad 69% urządzeń mobilnych na świecie posiada system Android [1]. Obecnie firmom zależy na pozyskaniu dla swoich projektów mobilnych zarówno użytkowników posiadających system Android jaki iOS. Rozwiązaniem takiego aspektu jest zaprogramowanie aplikacji działającej pod kontrolą obu systemów. Umożliwia to język Dart przy współpracy z narzędziem Flutter. Takie rozwiązanie pozwala zminimalizować koszty oraz zasoby ludzkie, które należałoby przeznaczyć na utworzenie identycznych aplikacji z różnymi bazami kodu w różnych językach. W tym przypadku programista tworzy jedną implementację, a finalnie otrzymuje aplikację działającą na dwóch systemach operacyjnych.

W niniejszym artykule dokonano analizy porównawczej dwóch języków programowania: języka Java, który jest uważany za prekursora wśród języków programowania przeznaczonych dla aplikacji mobilnych oraz język Dart, który oferuje możliwość uruchomienia tego samego kodu na różnych platformach. Przeprowadzone porównanie ukierunkowane było na badania wydajnościowe, którym poddane były dwie identycznie działające autorskie aplikacje mobilne napisane w rozpatrywanych językach programowania. Przy porównywaniu wydajności skupiono się na takich parametrach jak obciążenie procesora, wykorzystanie pamięci RAM, zużycie baterii oraz czas wykonania programów. Dodatkowo pod uwagę wzięto również budowę kodu, liczbę linii z kodem i wsparcie społeczności. Rozważania podjęte w tym artykule mają ułatwić twórcom oprogramowania podjęcie decyzji w momencie, gdy muszą dokonać wyboru technologii, która zostanie użyta do opracowania aplikacji mobilnej.

2. Przegląd literatury

Rozpatrując budowanie aplikacji mobilnych należy wziąć pod uwagę, że istnieje możliwość programowania

takich aplikacji na dwa sposoby - w sposób natywny oraz wieloplatformowy. Aplikacja napisana natywnie charakteryzuje się tym, że jest dostępna na jedną platformę. Podejście wieloplatformowe umożliwia wykorzystanie jednego kodu do pracy na wielu platformach [2]. Aspekt ten zbadali: P. Kotarski, K. Śledź i J. Smołka w publikacji [3]. Elementy procesu badawczego odnosiły się do narzędzi informatycznych służących do generowania aplikacji przeznaczonych dla systemu Android. [3]. W swoich eksperymentach autorzy zaimplementowali programy w różnych technologiach, aby sprawdzić ich wydajność dotyczącą czasu posortowania tablic z elementami liczbowymi oraz zapisu i odczytu dużego pliku w urządzeniu fizycznym. Podsumowując dokonane wyniki badań twórcy zauważają, że rozwiązania międzyplatformowe nie wykazują właściwości, które mogłyby przyczynić się do ich zdeklasowania w porównaniu z metodami natywnymi. Ponadto stwierdzają, że oba rozwiązania są do siebie zbliżone pod względem wydajnościowym. W pracy nie stwierdzono jednoznacznie, które narzędzie jest bardziej efektywne dla programistów [3].

Jako drugie źródło wiedzy w analizowanym temacie oparto się na artykule D. Gałana, K. Fisza i P. Kopniaka [4]. Praca przedstawia konfrontację dwóch aplikacji, z których pierwsza została napisana w języku Kotlin przy wykorzystaniu zestawu narzędzi Android, zaś druga w języku Dart z zastosowaniem Flutter SDK [4]. Jako środowisko programistyczne posłużyło narzędzie Android Studio [5]. W badaniach autorzy biorą pod uwagę takie aspekty jak czas wykonywania operacji i obciążenie procesora urządzenia mobilnego, które mierzone są podczas przeprowadzania opracowanych scenariuszy testowych. Są to odpowiednio: sortowanie zbioru liczbowego, zapisywanie i odczytywanie danych z lokalnej bazy danych oraz zapisywanie i odczytywanie znaków z pliku tekstowego [4]. Ponadto autorzy analizują również biblioteki będące do dyspozycji programistów jak i objętość kodu źródłowego oraz wsparcie społeczności. Na podstawie przeprowadzonych eksperymentów twórcy pracy dochodzą do wniosku, że aplikacja zaimplementowana w języku Kotlin przy użyciu zestawu narzędzi Android SDK jest wydajniejsza, a ponadto posiada optymalny kod źródłowy i większe wsparcie społeczności oraz bibliotek niż aplikacja napisana za pomocą zestawu narzędzi Flutter SDK [4].

W publikacji [6] G. Kozieła oraz D. Sulowskiego wykonano badanie porównawcze języków programowania Java i Kotlin. W związku z badaniami zaimplementowano autorską aplikację mobilną oddzielnie dla każdego języka. Jej funkcjonalność polegała na sortowaniu zbiorów liczbowych przy użyciu wybranych algorytmów oraz wyświetleniu animacji. W analizie porównawczej skupiono się na takich aspektach jak czasy wykonania i kompilacji, obciążenie pamięci RAM jak też procesora. Poza tym do analizy włączono strukturę kodu, dostępność baz danych, a także wsparcie wspólnoty społeczności programistycznej. Należy dodać, że analiza postawionych kryteriów odbywała się podczas obserwacji projektowania, implementacji oraz działania stworzonej aplikacji. Otrzymane wyniki pozwoliły autorom pracy na wysunięcie wniosku, że zarówno jeden jak i drugi język nie wyróżnia się znacząco, aby stwierdzić, który z nich jest wydajniejszy przy rozpatrywaniu tworzenia aplikacji mobilnych działających na urządzeniach z systemem Android.

Spośród znalezionych publikacji za najbardziej dopasowaną do niniejszej pracy uznano publikację *Java and Dart programming languages: Conceptual comparison* autorstwa A. M. Hassana [7]. Autor w swojej pracy dokonuje omówienia języków programowania Dart oraz Java w celu określenia różnic oraz ich wspólnych cech. W artykule rozpatrywano właściwości dotyczące składni, typów danych, dziedziczenia, operatorów logicznych, instrukcji warunkowych oraz innych związanych z procesem kodowania aplikacji. Dodatkowo wskazano rozbieżność tych języków w zakresie używanych kompilatorów oraz dostępnych bibliotek. Poza tym przedstawiono także fragmenty kodu dla obu technologii. Z rozważań autora wynika, że język Dart ma ogromne możliwości programistyczne i może być równie atrakcyjny jak Java [7].

Z dotychczasowych rozważań wynika, że zarówno Java jak i Dart mają swoje atuty w konkretnych zastosowaniach. Należy zwrócić uwagę, że na popularności zyskują narzędzia wieloplatformowe wśród, których jest Flutter korzystający z języka Dart [8]. Dlatego rodzi się pytanie czy język Dart używany w rozwiązaniu dostarczającym aplikacje mobilne na Android i iOS ze wspólnego kodu jest wydajniejszy niż tradycyjne podejście z użyciem Javy i uzyskaniem aplikacji tylko na jedną platformę. Dodatkowo nie znaleziono publikacji dokonujących analizy tejże problematyki i z tego względu w niniejszym artykule podjęto się jej zbadania.

3. Metodyka badawcza

Na potrzeby badań zostały utworzone dwie autorskie aplikacje mobilne zaimplementowane w języku Java - przy użyciu standardowego zestawu narzędzi Android SDK oraz w języku Dart, którego używa narzędzie Flutter. Obie aplikacje posiadają takie same funkcjonalności, które posłużyły jako scenariusze do wykonania testów. Działanie aplikacji polegało na wykonaniu operacji po naciśnięciu odpowiedniego przycisku. Zaimplementowane operacje to wstawianie, edytowanie, odczytywanie i usuwanie tysiąca rekordów z nielokalnej bazy danych, a ponadto sortowanie tysiąca i dziesięciu tysięcy dodatnich liczb losowych.

Wyniki obciążeniowe były odczytywane podczas działania aplikacji z daną operacją przy wykorzystaniu środowiska programistycznego Android Studio. Środowisko to posiada wbudowane funkcje do monitorowania zachowania aplikacji działającej na urządzeniu fizycznym lub emulatorze, które określane są jako moduł Android Profiler. Do pomiaru czasów wykonania programów zastosowano wbudowane metody oferowane przez biblioteki badanych technologii. Należy jeszcze uwzględnić, że dla każdego z aspektów wydajnościowych wykonano po 100 prób.

W badaniu dotyczącym objętości kodu została rozpatrzona składnia języków Java i Dart, liczba linii kodu potrzebnych do napisania takiej samej autorskiej aplikacji w jednym jak i drugim języku. Kryterium, które obejmuje wsparcie społeczności polegało na zbadaniu liczby wpisów na portalach internetowych poświęconych społeczności programistycznej danego języka programowania.

Parametry urządzenia mobilnego wykorzystanego w badaniu przedstawiono w Tabeli 1. Dodatkowo należy wspomnieć, że telefon współpracował z bezprzewodową siecią - Wi-Fi o prędkości do 300 Mb/s przy pobieraniu i do 50 Mb/s przy wysyłaniu danych. Transmisja danych została wyłączona, aby urządzenie korzystało tylko z sieci bezprzewodowej.

Tabela 1: Parametry stanowiska badawczego

Urządzenie mobilne	Samsung Galaxy A52
Procesor	Qualcomm Snapdragon 720G, Ośmiordzeniowy
Pamieć RAM	6 GB
Pamięć ROM	128 GB
System	Android 11
Bateria	4500 mAh

Przed przystąpieniem do testów telefon został sformatowany i pozbawiony czynników, które mogłyby wpłynąć na wynik badania. Oprogramowanie na smartfonie zostało zaktualizowane do najnowszej wersji, aby wyeliminować sytuację niekontrolowanych połączeń z Internetem podczas wykonywania badań. W każdym z testów bateria była naładowana do poziomu nie mniejszego niż 85 %. Dodatkowo z urządzenia zostały usunięte wszystkie aplikacje, które prowadziłyby do nieoczekiwanych oddziaływań na jego zasoby.

4. Porównanie cech wybranych języków

Aplikacja napisana przy użyciu języka Java składa się ze zbioru klas. Klasy posiadają pola do przechowywania danych, które są określane mianem zmiennych, a co więcej posiadają także metody, aby wykonywać operacje na danych. Tworzenie klasy następuje zazwyczaj w osobnym pliku, który posiada taką samą nazwę jak klasa, ale nie jest to regułą. W Javie podczas deklaracji zmiennych należy zdefiniować rodzaj danych jakie będą przechowywane. Należy dodać, że od wersji Java 10 pojawił się typ specjalny var, dzięki któremu nie ma obowiązku podawania typu danych, ponieważ kompilator sam określi typ danej zmiennej.

Poznając język Dart należy pamiętać, że wszystko co można umieścić w zmiennej jest obiektem, a obiekt jest instancją klasy [9]. W tym języku klasy także posiadają zmienne oraz metody do pracy nad nimi. Instancje klasy można deklarować z określeniem typu danych, lecz nie ma takiego przymusu, gdyż od momentu powstania ten język posiada typ var, który pozwala na zadeklarowanie zmiennych bez określania typu.

Konstruktor w Javie posiada nazwę identyczną jak nazwa klasy i strukturę podobną do metod wraz z argumentami, które zostaną przypisane do obiektów klasy. Analogicznie jest w przypadku języka Dart, ponieważ domyślnie dla klasy również dostępny jest konstruktor bezargumentowy o tej samej nazwie co stworzona klasa. Konstruktor może przyjmować parametry w celu zainicjalizowania zmiennych klasy przy użyciu własnych wartości. W języku Java konstruktor nie może zwracać wartości. Natomiast w języku Dart istnieje taka możliwość przy użyciu konstruktora fabrykującego. Utworzenie takiego konstruktora jest możliwe przy użyciu słowa kluczowego factory przed nazwą konstruktora. Wywołanie takiego konstruktora jest identyczne jak w przypadku zwykłych konstruktorów.

Gettery i settery to metody, które pozwalają pobierać i inicjować wartości pól klas. W Dart, aby zdefiniować metodę pobierającą wartość pola klasy - getter, należy użyć słowa kluczowego get, które umieszczane jest po określeniu typu danych zwracanych, ale przed nazwą metody zwracającej dane pole klasy. Identycznie wygląda sytuacja podczas tworzenia metody inicjalizującej pola klasy - setter, ale tutaj słowem kluczowym jest set. Metoda ustawiająca setter posiada jeden parametr i nie zwraca żadnej wartości. Natomiast w Javie settery i gettery mogą mieć dowolną nazwę i nie potrzebują modyfikatora określającego set lub get. Wystarczy pamiętać, że metoda ustawiająca nic nie zwraca, posiada parametr i przypisuje go do pola klasy, a metoda pobierająca zawsze zwraca jakąś wartość i posiada typ jak zmienna zwracana.

4.1. Struktura projektów

Aplikacja zaimplementowana w języku Java została skompilowana w trybie debug, a w języku Dart w trybie profile. Do analizy struktury projektów autorskiej aplikacji napisanej w Java i Dart wykorzystano wtyczkę Statistic dla środowiska Android Studio [5]. Za pomocą tego dodatku sprawdzono m.in. liczbę plików, rozmiar projektów i liczbę linii z kodem. Narzędzie Statistic posiada możliwość filtrowania danych, które mają zostać poddane analizie między innymi po rozszerzeniu plików. Na potrzeby tej pracy skorzystano z filtrowania plików z rozszerzeniem .dart dla aplikacji napisanej w języku Dart oraz .java dla aplikacji korzystającej z Javy. W tym przypadku były to pliki napisane przez programistę. Dodatkowo dla aplikacji zaimplementowanej w języku Java poddano analizie pliki z rozszerzeniem .xml, które definiują graficzny interfejs aplikacji. W języku Dart interfejs graficzny jest definiowany w kodzie Dart. W Tabeli 2 zaprezentowano uzyskane wyniki.

Tabela 2: Statystyka plików Java, Xml oraz Dart

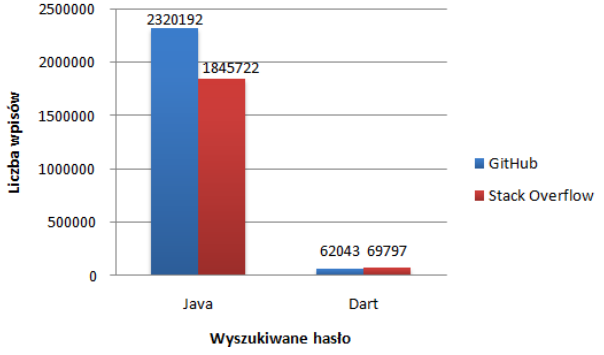
Kryterium	**Typ plików**		
	Dart	Java	XML
Liczba plików	9	7	3
Liczba linii z kodem	525	420	279
Rozmiar plików na dysku [KB]	28	32	20
Rozmiar całego projektu aplikacji po instalacji [MB]	47,56	13,35	

W statystyce ukierunkowanej tylko na dane z rozszerzeniami badanych języków to jest .java oraz .dart wynika, że liczba plików języka Dart jest o 2 większa w tym projekcie niż dla projektu zaimplementowanego w języku Java. Ponadto liczba linii zawierających kod jest większa o sto dla autorskiej aplikacji w języku Dart. Różnica w rozmiarze rozpatrywanych plików jest znikoma, ponieważ wynosi 4 KB i w tym przypadku to pliki języka Java zajmują więcej miejsca na dysku. Rozpatrując tworzenie graficznego interfejsu użytkownika widać, że dla aplikacji zaimplementowanej w języku Java niezbędne są dodatkowe pliki xml, w których zdefiniowany jest wygląd aplikacji. W języku Dart wygląd aplikacji jest zdefiniowany w kodzie plików z rozszerzeniem .dart. Dodatkowo należy zwrócić uwagę, że aplikacja zaimplementowana w języku Dart przy użyciu Fluttera posiada prawie cztery razy większy rozmiar niż autorska aplikacja w języku Java.

Analiza statystyki dla struktury projektu aplikacji napisanej przy współpracy Darta i Fluttera pokazuje, że prawie we wszystkich rozpatrywanych kryteriach ów projekt przyjmuje znacząco większe wartości niż projekt Java z Android SDK. Jednakże ze względu na to, że aplikacja zaimplementowana w języku Java potrzebuje dodatkowo plików definiujących wygląd to statystyka plików projektów jest korzystniejsza dla aplikacji w języku Dart. Można przypuszczać, że większy rozmiar aplikacji zaimplementowanej w języku Dart jest spowodowany wieloplatformowymi możliwościami Fluttera, który potrzebuje większej liczby plików oraz bibliotek, aby projekt napisany w jednym języku mógł działać na dwóch systemach to jest Android i iOS.

4.2. Wsparcie społeczności

W niniejszej pracy badanie dotyczące wsparcia społeczności obejmuje analizę popularności języków Java i Dart na wybranych portalach społeczności informatycznej. Podczas nauki bądź pracy programiści napotykają wiele problemów dotyczących języków, bibliotek oraz innych narzędzi programistycznych, którymi dzielą się na forach internetowych w celu zwiększenia szansy na rozwiązanie danego problemu i podzielenia się nim z innymi członkami społeczności. Aby ocenić popularność badanych języków zestawiono liczbę wpisów zawierających nazwą danego języka programowania z dwóch najbardziej popularnych serwisów społeczności to jest GitHub oraz Stack Overflow [10].



Rysunek 1: Popularność języków programowania.

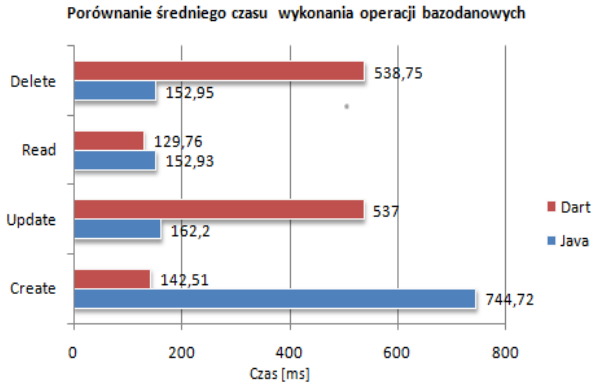
Analizując zebrane dane przedstawione na Rysunku 1 można zauważyć, że większą popularność osiąga Java. Badania popularności zostały przeprowadzone w kwietniu 2022 roku i dla Javy w serwisie GitHub odnotowano 2320192 wpisów, a w StackOverflow 1845722. Natomiast wyniki dla języka Dart są wyraźnie mniejsze i są to odpowiednio 62043 zapytania na portalu GitHub oraz 69797 dla StackOverflow [11, 12]. Interpretując uzyskane dane należy zwrócić uwagę na fakt, iż język Java miał swój początek 31 lat temu w 1991 roku [13]. Jest on zatem znacznie starszy niż język Dart, który pojawił się dopiero w 2011 roku [14]. Ze względu na swój młody wiek Dart nie jest jeszcze tak dobrze rozpowszechniony jak Java, która obecnie jest już widoczna niemal we wszystkich gałęziach przemysłu, biznesu i wielu innych kluczowych sektorach.

Warto dostrzec, że język Dart od 2015 roku jest ściśle rozwijany pod kątem aplikacji mobilnych w platformie Flutter [15] co może przyczynić się do wzrostu popularności tego języka wśród programistów tworzących aplikacje mobilne. Co więcej w 2021 roku Flutter, w którym wykorzystywany jest język Dart był najczęściej wybierany spośród narzędzi do tworzenia aplikacji mobilnych [8].

5. Wyniki wydajnościowe

5.1. Czas wykonania programów

Na Rysunku 2 przedstawiono zestawienie średnich czasów wykonania operacji, którymi są wstawianie rekordów do bazy danych - create, edytowanie - update, odczytanie - read i usuwanie - delete.

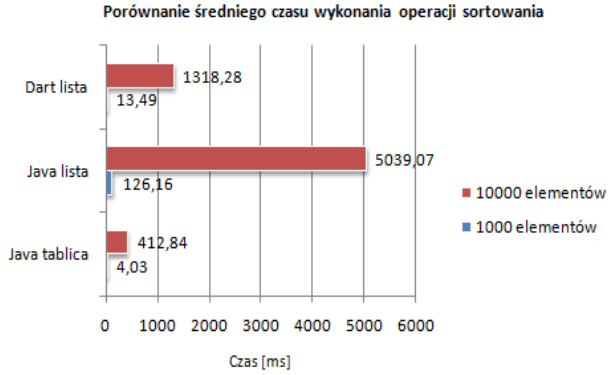


Rysunek 2: Porównanie średniego czasu wykonania operacji bazodanowych.

Aplikacja zaimplementowana w języku Java potrzebuje najwięcej czasu na wykonanie operacji wstawiania rekordów do bazy danych. Jest to wartość o 602,21 milisekundy dłuższa niż wynik dla tej samej funkcjonalności zaimplementowanej w języku Dart. Ostatnią operacją, w której Java osiąga lepsze wyniki czasowe nad językiem Dart jest proces odczytywania danych. Różnica jest znikoma, ponieważ wynosi zaledwie 23,17 milisekund na korzyść języka Dart. W pozostałych dwóch doświadczeniach: delete i update dla

języka Dart średni czas wykonania jest wyraźnie większy niż dla języka Java.

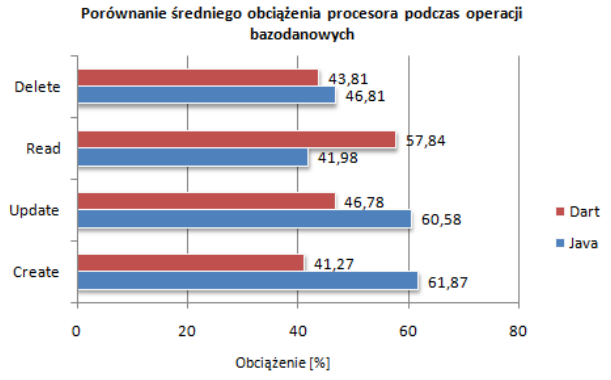
Kolejny scenariusz testowy dotyczył pomiaru czasu wykonania funkcji zaimplementowanej do sortowania zbiorów liczbowych przy wykorzystaniu algorytmu sortowania bąbelkowego. Zbiory danych zawierały odpowiednio 1000 oraz 10000 losowych dodatnich elementów typu int. Ich zawartość była identyczna zarówno dla aplikacji napisanej w języku Java jak i Dart. Dane przechowywane były w plikach, a przed sortowaniem następowało ich wczytanie i zapisanie liczb do odpowiednich struktur danych, to jest: dla języka Dart zbadano listę, a dla Javy tablicę oraz listę, ponieważ język Dart nie posiada oddzielnej struktury zwanej tablicą.



Rysunek 3: Porównanie średniego czasu sortowania liczb.

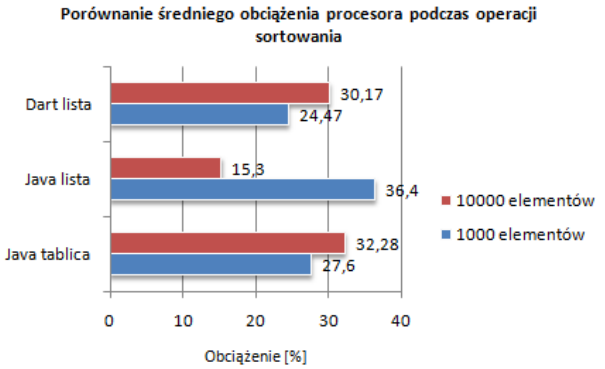
Porównując średnie czasy wykonania operacji sortowania algorytmem bąbelkowym zaprezentowane na Rysunku 3 można zauważyć, że sortowanie listy danych w języku programowania Java jest najmniej efektywne, ponieważ do zrealizowania tego zadania potrzeba 126,16 milisekund dla tysiąca elementów i aż 5037,07 milisekund dla dziesięciu tysięcy elementów co daje ponad 5 sekund. Java zdecydowanie szybciej poradziła sobie podczas sortowania zbioru liczbowego zawartego w tablicy, gdzie średni czas wykonania wynosił dla tysiąca elementów zaledwie 4,03 milisekund i tylko 412,84 milisekundy dla dziesięciu tysięcy. Język Dart podczas sortowania listy uzyskał wynik średnio o 3 razy większy niż Java podczas sortowania tablicy.

5.2. Obciążenie procesora



Rysunek 4: Porównanie średniego obciążenia procesora podczas operacji bazodanowych.

Na Rysunku 4 wyszczególniono średnie obciążenia procesora wszystkich wykonanych kryteriów testowych dotyczących operacji bazodanowych. Na jego podstawie można dostrzec, że aplikacja napisana w języku Java obciąża bardziej procesor podczas operacji create, update oraz delete niż taka sama aplikacja utworzona w języku Dart. Natomiast podczas operacji read Java osiągnęła mniejszy wynik obciążenia procesora niż język Dart.

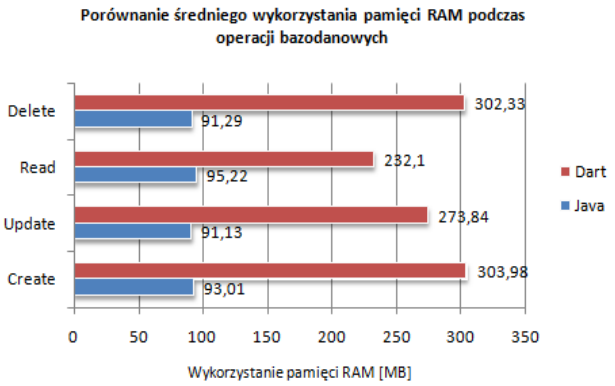


Rysunek 5: Średnie obciążenie procesora podczas sortowania.

Średnie obciążenia procesora podczas sortowania dwóch zbiorów liczbowych dla struktur danych w języku Java odpowiednio tablicy i listy oraz dla języka Dart listy prezentuje Rysunek 5. Analizując to zestawienie można zauważyć, że aplikacja zaimplementowana w języku Dart uzyskała najlepszy wynik podczas sortowania zbioru danych o małej liczbie elementów - tysiąc. Natomiast biorąc pod uwagę drugi zbiór liczb zawierający dziesięć tysięcy elementów widać jednoznacznie, że lista w języku Java w najmniejszym stopniu obciąża procesor podczas operacji z większą liczbą danych. Warto zauważyć, że podczas sortowania listy w języku Dart zarówno dla tysiąca jak i dziesięciu tysięcy elementów średnio uzyskano wyniki mniej obciążające procesor niż dla tablicy w języku Java, a to mogłoby tłumaczyć, dlaczego w języku Dart tablice zostały zastąpione listami.

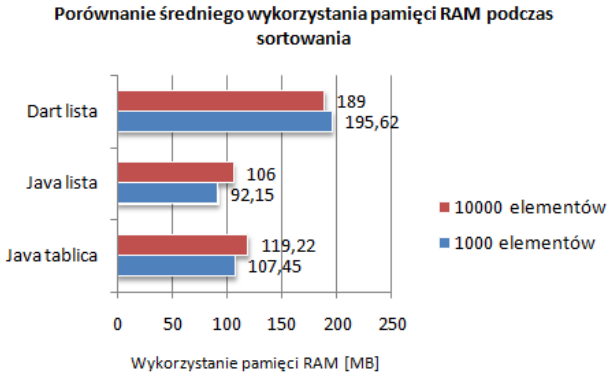
5.3. Wykorzystanie pamięci RAM

Rysunek 6 przedstawia średnie wykorzystanie pamięci RAM przez badane aplikacje podczas wykonywania operacji bazodanowych to jest create - wstawiania rekordów do bazy danych, update - edytowania, read - odczytania oraz delete - usuwania. Jednoznacznie widać, że język Java bezkonkurencyjnie osiągnął lepsze wyniki bez względu na wykonywaną operację. Różnica pomiędzy średnimi wykorzystania pamięci w danych operacjach wynosiła maksymalnie 4 megabajty. Aplikacja zaimplementowana w języku Dart wypadała gorzej podczas wykonywania procesów bazodanowych. Operacja wstawiania rekordów do bazy danych generowała największe wykorzystanie pamięci RAM, a najmniejsze w operacji odczytywania danych, lecz różnica w tym przypadku była niewielka, gdyż wynosiła zaledwie 71,88 megabajtów.



Rysunek 6: Porównanie średniego wykorzystania pamięci RAM podczas operacji bazodanowych.

Ostatnie doświadczenie jakie przeprowadzono w celu sprawdzenia wykorzystania pamięci RAM dotyczyło sortowania zbiorów liczbowych. Rysunek 7 prezentuje średnie wykorzystanie pamięci RAM dla całego eksperymentu dotyczącego sortowania danych i widać, że struktura danych zwana listą w języku Java wykorzystywała średnio najmniej pamięci RAM, a różnica liczebności zbioru nieznacznie wpłynęła na zwiększenie wykorzystania pamięci. Minimalnie większe wykorzystanie pamięci odnotowano dla tablicy w tym języku, średnio o 10 megabajtów więcej niż dla listy. Zarówno proces sortowania listy jak i tablicy w języku Java wraz ze wzrostem liczby elementów w zbiorze wykazywał zwiększone wykorzystanie pamięci RAM. Sytuacja jest odwrotna w przypadku listy dla języka Dart, ponieważ podczas sortowania liczniejszego zbioru średnie wykorzystanie pamięci było mniejsze niż dla mniejszego zbioru elementów.



Rysunek 7: Porównanie średniego wykorzystania pamięci RAM podczas sortowania.

5.4. Zużycie baterii

Przy wykorzystaniu narzędzia Android Profiler określenie wykorzystania baterii było możliwe poprzez trzy poziomy: Low, Medium oraz High, gdzie Low oznacza najmniejsze wykorzystanie, a High największe w danym momencie pracy aplikacji na urządzeniu fizycznym.

Wyniki uzyskane z przeprowadzonych scenariuszy testowych wskazują, że podczas działania autorskich aplikacji bez względu na rodzaj testu bateria była wykorzystywana na stałym poziomie Medium i nie wykazywała tendencji do zmiany tej wartości.

6. Wnioski

Analizując wyniki otrzymane ze zrealizowanych badań można spostrzec, że wybór języka programowania do tworzenia aplikacji mobilnych pomiędzy językiem Java, a Dart jest istotnym elementem, który może przyczynić się do zwiększenia popularności finalnej aplikacji. Analizując czasy wykonania programów ze scenariuszy testowych dotyczących operacji bazodanowych można zauważyć, że programy napisane w języku Java charakteryzują się większą wydajnością. Z kolei język Dart odznaczył się mniejszym obciążeniem procesora podczas tych eksperymentów. Rozpatrując poziom wykorzystania pamięci RAM należy zwrócić uwagę, że aplikacja zaimplementowana w języku Java wyraźnie osiągnęła lepsze wyniki, a co za tym idzie aplikacja mogła wykonywać zadania ze zwiększoną szybkością. Natomiast aplikacja napisana w języku Dart znacząco wykorzystywała pamięć RAM, nawet do trzech razy więcej niż bliźniacza aplikacja utworzona w języku Java. Prawdopodobnie zwiększone zapotrzebowanie na pamięć RAM mogło przyczynić się do wydłużenia czasów wykonania programów dla języka programowania Dart.

Rozważając aspekt sortowania danych również można zauważyć znaczące różnice, które dotyczą wydajności porównywanych języków programowania podczas pracy na różnych strukturach danych. Należy zwrócić uwagę, że język Java dla funkcji sortującej dane umieszczone w tablicy osiągnął najmniejszy czas wykonania, gdzie proces ten w języku Dart dla listy osiągnął czasy większe około 3 razy. Największy czas wykonania sortowania uzyskała struktura danych zwana listą w języku Java, ale jeśli chodzi o obciążenie procesora i wykorzystanie pamięci RAM można stwierdzić, że w najmniejszym stopniu obciążała urządzenie mobilne.

Pomimo tego, że autorskie aplikacje posiadały identyczne funkcjonalności to większy rozmiar osiągnęła aplikacja zaimplementowana w języku Dart.

Aby jednoznacznie stwierdzić, który język programowania jest efektywniejszy należałoby wykonać dodatkowe badania z uwzględnieniem większej liczby urządzeń posiadających różne podzespoły i systemy. Warto zwrócić uwagę, że język Dart przy współpracy z narzędziem Flutter przy jednej bazie kodu dostarcza programiście aplikację finalną działającą pod systemem Android oraz IOS. Wieloplatformowość tej technologii pozwala na zaoszczędzenie czasu oraz środków przeznaczonych do utworzenia potencjalnej aplikacji. Takiej zalety nie posiada język Java z zestawem narzędzi Android. Dlatego jest to ważny aspekt podczas rozpatrywania przydatności porównywanych języków programowania do tworzenia aplikacji mobilnych. Podsumowując dokonane analizy można stwierdzić, że wybór pomiędzy językiem Java, a Dart powinien być analityczny i najbardziej pasujący do wymagań danego projektu.

Literatura

[1] Popularność mobilnych systemów operacyjnych, https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009, [12.10.2021].

[2] Cechy Aplikacji natywnych i wieloplatformowych, https://www.uptech.team/blog/native-vs-cross-platform-app-development, [11.12.2021].

[3] P. Kotarski, K. Śledź, J. Smołka, Analiza wydajności aplikacji mobilnych przy zastosowaniu różnych narzędzi programistycznych do ich budowy, Journal of Computer Sciences Institute 6 (2018) 68-72. https://doi.org/10.35784/jcsi.642

[4] D. Gałan, K. Fisz, P. Kopniak, Porównanie aplikacji mobilnych zbudowanych przy zastosowaniu zestawów narzędzi programistycznych Android oraz Flutter z użyciem wielu kryteriów, Journal of Computer Sciences Institute 19 (2021) 107-113. https://doi.org/10.35784/jcsi.2614

[5] Android Studio, https://developer.android.com/studio [11.12.2021].

[6] D. Sulowski, G. Kozieł, Analiza porównawcza języków Kotlin i Java używanych do tworzenia aplikacji na system Android, Journal of Computer Sciences Institute 13 (2019) 354-358. https://doi.org/10.35784/jcsi.1332

[7] A.M. Hassan, JAVA and DART programming languages: Conceptual comparison, Indonesian Journal of Electrical Engineering and Computer Science 17 (2020) 845-849. http://doi.org/10.11591/ijeecs.v17.i2.pp845-849

[8] Popularność platform mobilnych na świecie, https://www.statista.com/statistics/869224/worldwide-software-developer-workinghours/, [11.01.2022].

[9] Dokumentacja języka programowania Dart, https://dart.dev/guides/language/language-tour, [10.05.2022].

[10] Ranking serwisów społeczności informatycznej, https://www.closeriq.com/blog/2020/06/top-developer-communities [11.04.2022].

[11] Serwis Stack Overflow, https://stackoverflow.com/, [11.04.2022].

[12] Serwis Git Hub, https://github.com/, [11.04.2022].

[13] C.S. Horstmann, Java. Tom I - Podstawy, Helion, 2016.

[14] Wprowadzenie do języka programowania Dart, https://en.wikipedia.org/wiki/Dart_(programming_language)#cite_note-12, [11.05.2022].

[15] Wprowadzenie do technologii Flutter, https://en.wikipedia.org/wiki/Flutter_(software)#cite_note-6, [22.04.2022].