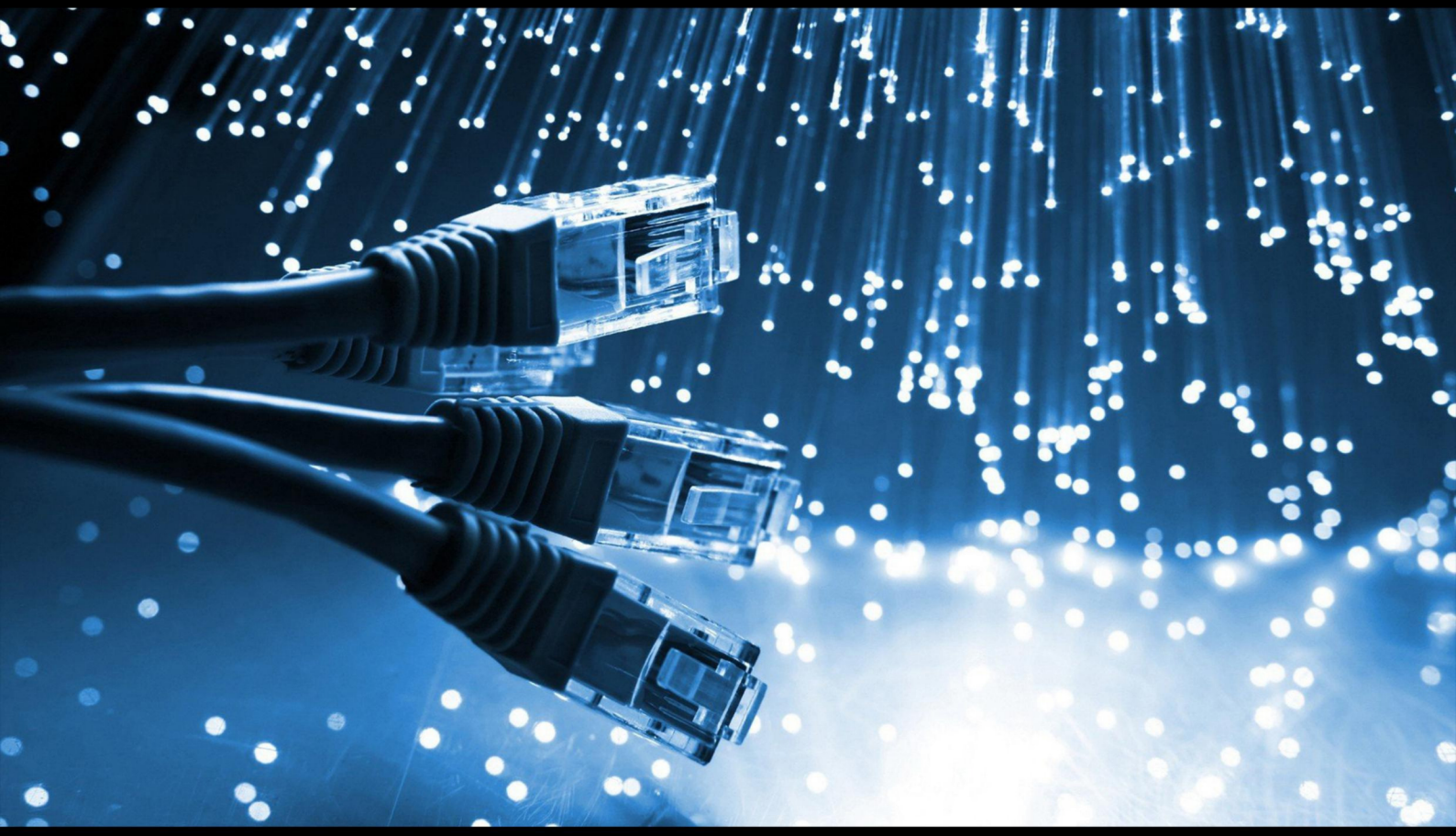
JCSI

Journal of Computer Sciences Institute

Volume 26/2023



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

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

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

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

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

Recenzenci numeru:

dr inż. Małgorzata Plechawska-Wójcik
dr inż. Maria Skublewska-Paszkowska
dr Mariusz Dzieńkowski
dr Rafał Stęgierski
dr Adam Kiersztyn
dr inż. Kamil Żyła
dr inż. Tomasz Nowicki
dr inż Sławomir Przyłucki
dr Michał Dolecki
dr Paweł Powroźnik
dr inż. Dariusz Gutek
dr inż. Tomasz Szymczyk
dr inż. Piotr Kopniak
dr inż. Marcin Badurowicz
dr inż. Maciej Pańczyk

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

Projekt okładki:
Marta Zbańska

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

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

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

Reviewers:

Małgorzata Plechawska-Wójcik
Maria Skublewska-Paszkowska
Mariusz Dzieńkowski
Rafał Stęgierski
Adam Kiersztyn
Kamil Żyła
Tomasz Nowicki
Sławomir Przyłucki
Michał Dolecki
Paweł Powroźnik
Dariusz Gutek
Tomasz Szymczyk
Piotr Kopniak
Marcin Badurowicz
Maciej Pańczyk

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

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL OF COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 1-6
Received: 10 September 2022
Accepted: 7 November 2022

Augmenting The Cloud Environment Security Through Blockchain Based Hash Algorithms

Ravi Kanth Motupalli[a,*] and Krishna Prasad K.[b]

[a]Research Scholar, Institute of Computer Science & Information Sciences, Srinivas University, Mangalore, Karnataka, India and Assistant Professor, Department of CSE, VNR VJIET, Hyderabad, Telangana, India.
[b]Associate Professor, Institute of Computer Science & Information Sciences, Srinivas University, Mangalore, Karnataka, India.

Abstract
Many techniques and algorithms are developed to enhance the security in the cloud environment. This helps the users to secure their server from malicious attacks. Hence the study and investigation of the performance enhanced security algorithms is a must demanded field in the research industry. When large number users using same server to store their information in cloud environment security is a must needed component to preserve the privacy and confidentiality of every individual user. This can be further strengthened by detecting the attacks in earlier stages and taking counter-measure to prevent the attack. Thus securing the data network without any leakage and loss of the information is a challenging task in the cloud environment. When the attacks or intrusion is detected after the occurrence there may be damage to the data in the form of data damage or theft. Hence it is necessary to predict and detect the attacks before the occurrence to protect the privacy and confidentiality of the user information.

Keywords: Cloud security; Data privacy; Data confidentiality; Hash Algorithm; Substitutional encryption

*Corresponding author
Email address: ravikanth_m@vnrvjiet.in (Ravi Kanth. M)

1. Introduction

The most preferred technology for the storage of large volumes of data with data privacy and confidentiality is cloud computing. The security risk in the cloud technology may cause collateral damage to the business applications and the numerous number of users. Hence there are many important security policies in the cloud applications like never leaving the cloud server open for other user which may attract the intruders to steal the data, all the security upgrades given by the vendors should be switched in time to prevent risking the data, Basic cyber security steps should be followed to protect the data from user side, use of strong password and efficient password management, Backing up the data to avoid data loss, should not modify the vendor provided permissions, etc.

When a network is designed to protect the user data or host large numbers of users simultaneously, its ability and competence to serve the users is deprived to have an extraordinary approach to its users. Fig.1 illustrates how the cloud environment protects the end user data against the threats and attacks.

It is the responsibility of the cloud service provider to ensure the authentication of the data access through physical path audit controls and the data protection assurance through the strong security algorithms [1]. The data repositories should be consistently watched for their activities and suspicious behaviours. When the service provider provides more services for multiple users there may arise situations where the details about the users will be backed up in the compatible server. During this case it is possible for any user to duplicate the identity of another user [2]. To avoid such damage to the user data two security algorithms using substitution approach and hash algorithm were devised in this study.

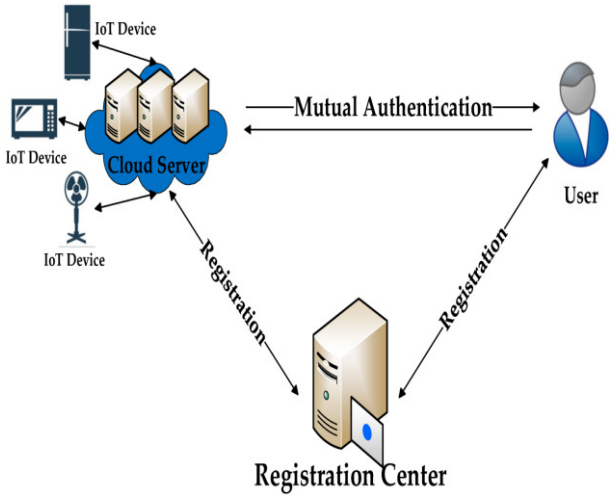


Figure 1: Authentication in Cloud environment.

2. Literature Survey

Many researchers had contributed their study in the cloud security domain. Cloud security involves the confidentiality of the user data and authentication to the user id. It also ensures safety against any type of attacks. Depending upon the applications many type of security algorithms were proposed by different authors and hybrid model of these algorithms were built [3]. Table 1. explains some of the literature work done by different authors on different security algorithm.

When there is any security breach in the cloud server the corresponding counter action should be taken to compensate the attack [4]. Encrypting the original data before transmission is a preferable choice to avoid the theft or data leak. Hence the service distributor should provide a strong security for the authenticated access of the data [5]. Double encrypted algorithm is the advanced version of the authentication algorithm to improvise the secure access [6].

Cloud security assures the susceptible cloud server against any type of attacks and intrusions that leads to the malicious attacks [7]. Hence it is responsibility of the service provider to ensure the secure authentication to the data access and assuring the data confidentiality of the user data. This model proposes a hybrid model of substitution algorithm to provide authenticated access and hash algorithm to protect the data confidentiality and avoid data loss or damage.

Table 1: Literature study on Security algorithms

	Focus/ Constructs	Reference
Cloud computing security algorithms	DES algorithm is the first devised algorithm for encryption. But due to weak cipher this is prone to many attacks	Hui et al,2019[8]; Srisakthi et al, 2020[9].
	The new AES algorithm was devised to tackle the brute force attacks.	Namasudra et al, 2020[10].
	AES algorithm is simple and rapid.	Sivakumar et al,2019[11].
	TDES algorithm is the customized version of the DES for triple protection. But the complexity of this algorithm makes it impossible to implement for larger data.	Gowtham et al,2021[12]; Kumar et al , 2017[13].
	Blowfish algorithm is a 64 bit block cipher algorithm known to be strong against any type of attacks. This algorithm was not patented and is owned by anyone to use in their application.	Yassein et al, 2017[14]; Bhuvaneshwaran et al, 2019[15].
	Homomorphic algorithms make use of two different public and private symmetric keys. Losing any one key may cause loss data in decryption.	Yahyaoui et al, 2018[16].
	RSA algorithm is also a symmetric key algorithm which generates two large prime numbers which are multiplied to produce private and public keys.	Mahmood et al, 2018[17]. Gayatri et al, 2020[18].

2.1. SECURITY CHALLENGES IN CLOUD COMPUTING

In cloud servers many users are being served with different services. Hence there arises many different issues in the security of the cloud server and user data. Data issue is considered as the foremost issue where the service provider is responsible for the confidentiality of the user data in the server [19]. Data audit should be conducted through automated algorithms to ensure the confidentiality of the data by avoiding data breach and data corruption.

Privacy of data is another issue in the cloud servers. More number of users being served at the same instance there is a chance of a data backlog that leads to the cross access of another user's data. In order to avoid this data breach and to maintain data privacy each user access should be verified using a two way authentication approach [20].

Security issue is a must addressable issue that avoids data tampering and data theft. The cloud server should nor be vulnerable to the external attacks and must ensure to detect the data attack at the early stage and avoid data damage or loss [21].

3. Objectives

The main objective of this research is to find an effective security algorithm to protect the data confidentiality and privacy of every individual user in the cloud environment using blockchain based Hash algorithm and to compare the performance analysis of the hash algorithm against the substitution algorithms.

i. The first and foremost objective of this study is to identify the security challenges in the cloud environment.
ii. Designing and developing the computational process for the substitution algorithm and hash algorithm.
iii. Comparing the performance evaluation of the hash algorithm against other substitutional algorithms in terms of time taken for the execution, amount of memory used and the throughput.

4. Proposed Methodology

Many researchers had contributed their study in the cloud

The flow chart representation of the proposed model to authenticate and protect the cloud data is illustrated in the Fig.2. Initially the user is registered to the cloud service provider through an authenticated server. Then the identification of the user was protected using the substitution algorithm. The user data in the cloud network is protected using the secure hashing algorithm. The change in the hash value is monitored to detect the data damage and data loss to secure the data.

4.1. Substitutional Algorithm

Caeser Cipher Substitution algorithm is considered to be the outstanding substitution algorithm in assuring security to the cloud networks. In this method the secret key is always given in numbers and the decrypted format of the cipher text is saved in the cloud server. This method is considered as a conventional and simplest method of encryption. In this encryption technique the shift method is used for the formation of the cipher text.

In the proposed substitutional technique double encryption of the dataset for the enhanced security is done. In order to generate the cipher text an obsessed numerical value is needed, which is termed as the shift value. Based on the numerical value of the shift the alphabets of the cipher are moved exactly shift times to generate the encrypted text.

The encryption equation based on the given shift is formulated as follows:

$$E_i(x) = (x + i)\ mod\ 26 \quad (1)$$

Where x represents the alphabetic character in the original data and n refers to the shift. Similarly the decryption equation of the cipher text is given as follows:

$$D_i(x) = (x - i)\ mod\ 26 \quad (2)$$

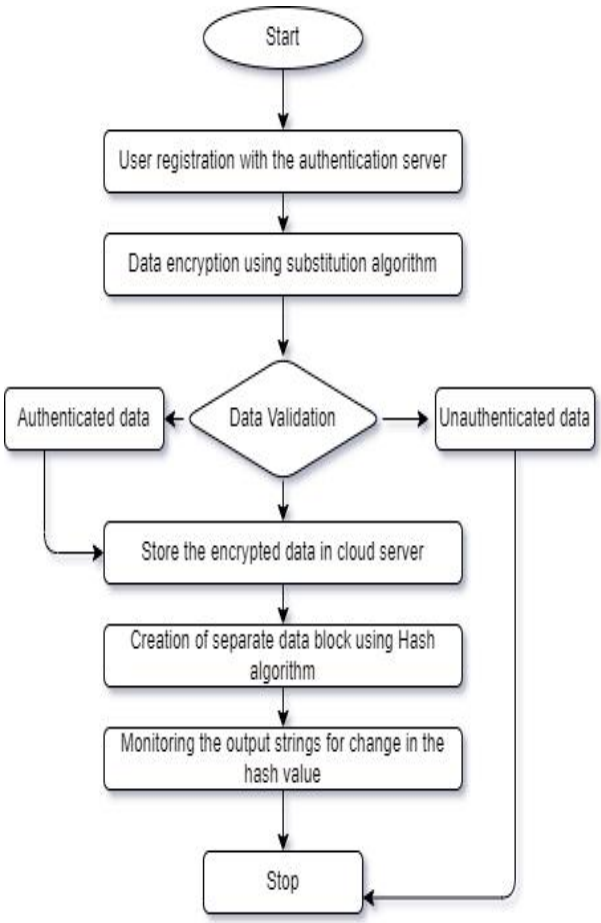


Figure 2: Block diagram of the proposed model.

During decryption the alphabets in the cipher text are shifted based on the number of shifts done in the encryption. The steps involved in the double encryption algorithm is explained as follows:

Encryption Algorithm

Step1: *The plain text to be encrypted is obtained from the user.*

Step 2: *The shift number 1 is given as the secret key 1 to the encryption.*

Step 3: *The number of shifts based on the secret key 1 is done to generate the cipher text 1.*

Step 4: *Now for the double encryption secret key 2 is given for the second encryption.*

Step 5: *Cipher text 1 is shifted based on the secret key 2 to generate the cipher text 2, which is the exact cipher text of the double encryption method.*

Decryption Algorithm

Step 1: *Two secret keys 1 and 2 with the known cipher rows are given for the decryption.*

Step 2: *Using secret key 1 first decryption is done and using the secret key 2 second decryption is done.*

Step 3: *The output of the second decryption is the original data given by the user.*

Through the double encryption substitution technique secure user authentication is done with less permutational complexity. This approach produces an output that is impossible to crypt analyse and the method is simple. But the complexity of the proposed substitutional algorithm is higher than the simpler Caesar algorithm with double approach in encryption and decryption. Multiple keys are involved in this procedure which should not be mixed up to produce the original data. In order to ensure the protected authentication in the cloud environment the password of each user is encrypted using the substitutional technique which avoids the storage of the original password in the cloud server.

4.2. HASH ALGORITHM

To protect the data in the cloud server against damage and loss due to malicious attacks and intrusions Hash algorithms are used. In this approach the arbitrary data block of the data set is processed to produce a standard string of the output. With the changes in the hash value damage or data loss can be identified.

There are many secure hashing algorithms which are designed one way to provide strong security. Thus this algorithm is strong enough to detect the intrusion or attack done on the data This algorithm is combined with the block chain technique which generates the hash value to protect the data. The hashing algorithm used in this model is SHA-512. Hash algorithm is notable for exhibiting overwhelming results. The algorithm works on the basic principle of the simpler modification made in the input block creates a complex impact on the output string and the distant blocks produce a comparable hash value. The hashing of the 512 bit block in the input is formulated as follows:

$$Ha(512) = \sum 512 \; p(x)(x \; mod \; y) \; (x - i) \qquad (3)$$

Where the Ha(512) represents the hash function of the 512 bit block and the p(x) represents parole with mean y. In this algorithm initially the input block is segregated as the fragments with 1024 bits. 64 bit hash value and the rounding constants are generated. This algorithm comprises 80 iterations where the message array is organized as 80 words, each with 64 bits. The iterating loop is scheduled from the 16th bit to 79th bit. The results and performance analysis of the hash algorithm is discussed in the following section.

5. RESULTS AND DISCUSSION

Cloud computing framework is notable for its harmless yield since its implementation in the market. With the advancing approaches and the technical methodologies the security and authentication of the user data and the user is augmented. Thus the added benefits of the advanced cloud computing servers are the perseverance of the user authentication and data confidentiality.

Cloud encryption by Caesar's algorithm provides an encounter model to the mutual and scheduled architecture which is prone to the security breaches. The Caesar's substitution technique used in the proposed model gives more effective security against two attacks namely dictionary attacks and rainbow attacks. These attacks are made to eavesdrop the data exchange, creating a fake browser and stealing the passwords of the users. When the cloud network is in active state, a protocol analyzer is used in the analysis of the activity of the end users. When the user is vulnerable to any attack the admin of the cloud server alerts the users.

As double key is used in the double encryption based Caesar's cipher method enhanced security is achieved. The processing speed is fast as the technique is not computationally complex, also requires less time to encrypt and decrypt making this as the time consuming algorithm. As the data size of the encrypted cipher is same as that of the original message, lesser storage space is required when compared with the other substitutional techniques. The key exchange process in this technique is simpler without any challenging issues.

The Hash algorithm 512 encryption is used in the cloud server to protect the data. In this model the user data is hashed by the string of hash values. Variance in these values detects the modification or loss of user data and alerts the user for the same. The Hash 512 algorithm proposed in this model highly contributes to the fake browser problems which detects the attacks in the early stage and prevents them. The output parameters of the single key Ceaser model, SHA 512 model and the proposed model are compared in Table 2.

5.1. Execution Time:

A process is defined as the end user application running at the user system which comprises single or multiple executions. The average time taken for the execution of a single process and the amount of energy spent on the execution of the process is calculated to find its efficiency. In the proposed model 1813 kbps of execution speed and 5813 kbps of the application running speed is obtained making it more efficient than the individual models.

5.2. Throughput:

Throughput is the measure of the productivity of the processors responding to any algorithm. It is measured

as the amount of data exchanged between the users. The throughput of the file uploading process should be high enough to upload all the data and the throughput of the usage should be small enough to be computationally simple and fast. The proposed model exhibits highest throughput for file upload and lowest for the application usage.

5.3.Memory Utilization:

This denotes the amount of field taken by the respective algorithms to complete their processes.

Table 2: Output attribute comparison

Output Parameters	Caeser's model	SHA 512 model	Proposed model
Execution speed (kbps)	322	1743	1813.45
Application running speed (kbps)	580	632	5813
Throughput (file upload) (kbps)	150	157	194
Throughput (Application Usage)(kbps)	75	62	45
Memory Utilization (kbps)	240	225	180

In comparison with the other model the proposed model take only 180 kbps to complete the execution of the process making this encryption model more effective and rapid enough

6. CONCLUSION

In the communication era huge data are being transacted through cloud servers. All these transactions are demanded to be secure and perfect. To ensure this data security and confidentiality certain security algorithms were developed. In this series, a hybrid algorithm comprising of double encrypting substitutional approach and Hash algorithm is implemented. This algorithm is proved to be more efficient in providing data security and data confidentiality in effective way than the existing algorithms. This algorithm is implemented to prove to overcome the security limitations in the cloud network by suppressing the risks like unauthorized data access, data damage, data theft and data leaks. Thus this study has achieved all the objectives outlined and the comparative analysis done with the existing algorithms proves the same.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, Above the clouds: A berkeley view of cloud computing, Technical Report No. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009).

[2] N. Fernando, S.W. Loke, W. Rahayu, Mobile cloud computing: A survey, Future generation computer systems 29(1) (2013) 84-106.

[3] C. Yang, Q. Huang, Z. Li, K. Liu, F. Hu, Big Data and cloud computing: innovation opportunities and challenges, International Journal of Digital Earth 10(1) (2017) 13-53.

[4] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, Journal of network and computer applications 34(1) (2011) 1-11.

[5] M.A. Rababaa, M.A. Kofahi, F.A. Saqqar, S.A. Rababavh, Hash algorithms for security on GSM system, International Review on Computers and Software 4 (2009) 698-703.

[6] M. Ali, S.U. Khan, A.V. Vasilakos, Security in cloud computing: Opportunities and challenges, Information sciences 305 (2015) 357-383.

[7] N. Santos, K.P. Gummadi, R. Rodrigues, Towards Trusted Cloud Computing, HotCloud 9(9) (2009) 3.

[8] H. Hui, D. McLernon, Design and Application of a Service Outsourcing Cloud for the Insurance Industry, Proceedings of the 9th International Conference on Information Communication and Management (2019) 1-5.

[9] S. Srisakthi, A.P. Shanthi,Towards the Design of a Stronger AES: AES with Key Dependent Shift Rows (KDSR), Wireless Personal Communications 114(4) (2020) 3003-3015.

[10] S. Namasudra, R. Chakraborty, A. Majumder, Securing Multimedia by Using DNA-Based Encryption in the Cloud Computing Environment, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16(3s) (2020) 1-19.

[11] P. Sivakumar, M. NandhaKumar, R. Jayaraj, A. Sakthi Kumaran, Securing Data and Reducing the Time Traffic Using AES Encryption with Dual Cloud, IEEE International Conference on System, Computation, Automation and Networking (ICSCAN) (2019) 1-5.

[12] A.K. Gautam, R. Kumar, A comprehensive study on key management, authentication and trust management techniques in wireless sensor networks, SN Applied Sciences 3(1) (2021) 1-27.

[13] S.P. Kumar, A. Aswini, M. Kavithadevi, S. Ramya, Improvised dedupication with keys and chunks in HDFS storage, Third International Conference on Science Technology Engineering & Management (ICONSTEM), (2017) 226-230.

[14] M.B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, Y. Khamayseh, Comprehensive study of symmetric key and asymmetric key encryption algorithms, International conference on engineering and technology (ICET) (2017) 1-7.

[15] A. Bhuvaneshwaran, P. Manickam, M. Ilayaraja, K. Sathesh Kumar, K. Shankar, Clustering Based Cybersecurity Model for Cloud Data, Cyber security and Secure Information Systems (2019) 227-240.

[16] A. El-Yahyaoui, M.D.E.C. El Kettani, Data privacy in cloud computing, 4th International Conference on Computer and Technology Applications (ICCTA) (2018) 25-28.

[17] Z.H. Mahmood, M.K. Ibrahem, New fully homomorphic encryption scheme based on multistage partial homomorphic encryption applied in cloud computing, 1st Annual International Conference on Information and Sciences (AiCIS) (2018) 182-186.

[18] R. Gayatri, Y. Gayatri, Detection of Trojan based DoS Attacks on RSA Cryptosystem using Hybrid Supervised Learning Models, IEEE Third International Conference on Smart Systems and Inventive Technology (ICSSIT) (2020) 1-5.

[19] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation computer systems 28(3) (2012) 583-592.

[20] A.T. Hashem, I. Yaqoob, N.B. Anuar, S. Mokhtar, A. Gani, The rise of big data on cloud computing: Review and open research issues, Information systems 47 (2015) 98-115.

[21] D. Sun, G. Chang, L. Sun, X. Wang, Surveying and analyzing security, privacy and trust issues in cloud computing environments, Procedia Engineering 15 (2011) 2852-2856.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 7-12
Received: 13 September 2022
Accepted: 11 October 2022

Comparison of an effectiveness of artificial neural networks for various activation functions

Porównanie efektywności sztucznych sieci neuronowych dla różnych funkcji aktywacji

Daniel Florek*, Marek Miłosz

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

Abstract

Activation functions play an important role in artificial neural networks (ANNs) because they break the linearity in the data transformations that are performed by models. Thanks to the recent spike in interest around the topic of ANNs, new improvements to activation functions are emerging. The paper presents the results of research on the effectiveness of ANNs for ReLU, Leaky ReLU, ELU, and Swish activation functions. Four different data sets, and three different network architectures were used. Results show that Leaky ReLU, ELU and Swish functions work better in deep and more complex architectures which are to alleviate vanishing gradient and dead neurons problems. Neither of the three aforementioned functions comes ahead in accuracy in all used datasets, although Swish activation speeds up training considerably and ReLU is the fastest during prediction process.

Keywords: activation functions; artificial neural networks; artificial intelligence

Streszczenie

Funkcje aktywacji, przełamując linową naturę transformacji zachodzących w sztucznych sieciach neuronowych (SSN), pozwalają na uczenie skomplikowanych wzorców występujących w danych wejściowych, np. w obrazach. Wzrost zainteresowania wokół SSN skłonił naukowców do badań wokół różnolitych aktywacji, które mogą dać przewagę podczas uczenia jak i przewidywania, ostatecznie przyczyniając się do powstania nowych, interesujących rozwiązań. W artykule przedstawiono wyniki badań nad efektywnością SSN dla funkcji ReLU, Leaky ReLU, ELU oraz Swish, przy użyciu czterech zbiorów danych i trzech różnych architektur SSN. Wyniki pokazują, że funkcje Leaky ReLU, ELU i Swish lepiej sprawdzają się w głębokich i bardziej skomplikowanych architekturach, mając za zadanie zapobieganie problemom zanikającego gradientu (ang. Vanishing Gradient) i martwych neuronów (ang. Dead neurons). Żadna z trzech wyżej wymienionych funkcji nie ma przewagi w celności (ang. Accuracy), jednakże Swish znacznie przyspiesza uczenie SSN, a ReLU jest najszybsza w procesie przewidywania.

Słowa kluczowe: funkcje aktywacji; sztuczne sieci neuronowe; sztuczna inteligencja

*Corresponding author

Email address: daniel.florek@pollub.edu.pl (D. Florek)

1. Introduction

In the last ten years Artificial Neural Networks (ANN) [1] have entered daily use in our lives, they are being utilized everywhere in the mobile phones and the internet. They are employed extensively particularly in the computer vision area, where they had most success and are irreplaceable with other machine learning algorithms. Currently ANNs are applied to solve perception problems such as object detection, image segmentation, image classification, speech recognition or language translation.

At heart ANN is a simple linear transformation combined with non-linear activation functions and backpropagation algorithm that allow it to learn complex patterns from the data provided during training. There are multiple activation functions designed for different purposes but the one that gained a lot of popularity during 2012 revolution is Rectified Linear Unit (ReLU) [2]. Due to its simplicity and speed, it is used widely in computer vision tasks that often process multiple frames per second.

Because of the role that activation functions fulfill there have been many attempts to find better performing functions in order to get higher accuracy, faster training and possibly speed up inference.

In this paper the effectiveness and efficiency of selected activation functions for different architectures and datasets is analyzed. For the purpose of this research thesis and hypotheses have been devised:

T. Use of different activation functions results in significant changes in effectiveness of ANNs for various datasets.
H1. Activation functions can be sorted by efficiency and effectiveness during the training process of ANN.
H2. Activation functions can be sorted by efficiency during prediction process.
H3. The order of activation functions doesn't change for different datasets.

2. Literature overview

There is few works that extensively compare activation functions performance across multiple datasets and architectures. Ramachandran et al. [3] propose Swish function as well as compare performance of several most known functions on CIFAR datasets [4], ImageNet and WMT 2014 English→German dataset using deep and wide architectures with high number of trainable parameters such as ResNet-164, Wide ResNet 28-10, Inception architectures, Mobile NASNet-A or attention based Transformer Architecture. According to the results Swish function wins or ties most of the time when compared to other functions, while ELU (Exponential Linear Unit) [5] and LReLU (Leaky ReLU) [6] don't seem that reliable and often underperform when compared to ReLU. Although [3] contains performance comparisons of different activation functions it mostly focuses on Swish function.

The paper [7] summarizes information on many currently used activation functions in deep learning. Authors list cons and pros of each function along with their formulas. Although the authors do not compare activation functions against each other, this work is nonetheless great source of information.

Empirical Evaluation of Rectified Activations in Convolution Network [6] compares ReLU, Leaky ReLU, PReLU (Parametric ReLU) and RReLU (Randomized Leaky ReLU) performance on CIFAR-10, CIFAR-100 and National Data Science Bowl Competition dataset. Results show that PReLU, RReLU and Leaky ReLU with $a = 5.5$ are better than ReLU. Leaky ReLU with $a = 100$ performs similar to ReLU. PReLU has lowest training error on all datasets but RReLU taking first place in test error which implies that PReLU may be overfitting while RReLU combats it and Leaky ReLU with lower a value seems to come close to RReLU.

3. Experiment Settings

3.1. Environment and Libraries

Experiment was conducted on an older machine (Table 1) using Tensorflow [8], Keras [9] and Scikit-Learn [10] libraries with Python programming language [11]. Environment was set up using Anaconda platform [12].

Table 1: Specification of the machine the experiment was conducted on

Name	Description / Version
CPU	Intel i5-4670K @4.2GHz
GPU	Nvidia GTX 1060 3GB
RAM	16GB 1600MHz
OS	Windows 10 Pro 64-bit 21H2
GPU drivers	Nvidia 511.23
CUDA	CUDA Toolkit 11.2
cuDNN	cuDNN SDK 8.1.0,
Python	Python 3.9.7
Tensorflow	Tensorflow 2.7.0
Scikit-Learn	Scikit-Learn 1.0.2
Anaconda	Anaconda 4.11.0

3.2. Selected Activation Functions

This experiment is focused on non-linear activation functions which are used in hidden layers of the ANN models. Those functions need to be fast, therefore they are simple but effective in deep learning.

Rectified Linear Unit [2] is most widely used activation function and is when looking for new functions it is often considered as a baseline in research. ReLU replaced tanh and Sigmoid activation functions because of higher performance, better generalization and being easy to optimize.

ReLU formula is:

$$f(x) = \max(0, x) \tag{1}$$

Leaky Rectified Linear Units [6] was supposed to be improvement over ReLU to alleviate potential dead neurons problem. LReLU sacrifices sparsity for a gradient which should be more robust during optimization but is less memory efficient which is important thing to consider in mobile systems. In the paper [6] LReLU performs comparably to ReLU, but authors notice slightly faster convergence. Negative values are controlled by α hyperparameter, which is usually set lower than 1.

Leaky ReLU formula is:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{2}$$

Exponential Linear Unit [5] is an activation function similarly to LReLU and tries to improve on ReLU and is supposed to speed up training by alleviating vanishing gradient problem. ELU allows negative values which saturation is controlled by α hyperparameter. In the paper [5] authors state that ELUs lead to faster learning and significantly better generalization compared to ReLUs and LReLUs.

ELU formula is:

$$f(x) = f(x) = \begin{cases} x, & x > 0 \\ \alpha \exp(x) - 1, & x \leq 0 \end{cases} \tag{3}$$

Swish [3] activation function was found using reinforcement learning-based search when looking for a better alternative to ReLU. Authors claim that Swish performs better in deeper models than ReLU and other activation functions.

Swish formula is:

$$f(x) = x \cdot sigmoid(x) = \frac{x}{1+e^{-x}} \tag{4}$$

Shapes of selected activation functions (1)-(4) are presented in Figure 1.

3.3. Used Datasets

Quality and nature of dataset determine the quality and performance of the trained ANN model. Size also matters as the bigger the dataset and, in classification problem, the number of output targets increases the longer the training process and possibly higher need of hardware resources. For this reason datasets of reasonable size, number of targets and that have been tested by the machine learning community or researchers, have been selected.

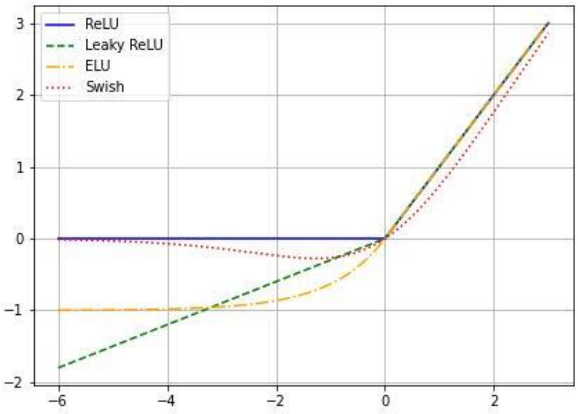


Figure 1: Activation functions shapes and values.

CIFAR-10 (Canadian Institute For Advanced Research) dataset [4] consists of 60,000 colour 32x32 pixel images split evenly between 10 classes. The targets are comprised of animals and transport vehicles. The dataset is split into training subset consisting of 50,000 images and 10,000 test images. The training subset is further split into five equal subsets for cross-validation.

CIFAR-100 dataset [4] has the same number of images and image size as CIFAR-10, but comes with 100 classes which are grouped into 20 superclasses such as fish, flowers, reptiles, people, trees, vehicles, household furniture etc. This dataset is also split in the same way as CIFAR-10, 50,000 training images which are further split into 5 equal size subsets and 10,000 test images.

Animals 10 dataset [13] has been posted on Kaggle website and it contains about 28,000 medium quality animal images with different sizes, split into 10 classes with varying number of images per class ranging from ~2,000 to ~5,000. In this research a subset of this dataset containing 1,400 of images per class has been used. Entire subsets contains 14,000 images resized to 112x112 pixels.

Intel Image Classification dataset [14] has also been put on Kaggle and it contains 14,034 training images, 3,000 test images across 6 classes and size 150x150. The dataset is almost evenly balanced and images are resized to 112x112 pixels. Classes consist of nature and urban areas scenes around the world.

3.4. Analyzed Artificial Neural Network architectures

Different architectures may have some impact on activation functions performance, therefore three architectures have been used. Two of those are well known among machine learning community and one is simple stacked convolutional layers. Layers use “same” padding with no bias vector, all layers use He normal kernel initializer [15] except logistic regression layer which uses Glorot uniform initialization [16]. During training all models use two data augmentation layers: random horizontal flip and random rotation. Strides are used instead of max pooling and after each layer batch normalization [17] and activation is applied.

ResNet [18] ANN network architecture is made of residual modules, each module consists of batch normalization, activation and convolutional layer with everything repeated two times, input of module is copied and goes through convolutional layer of size 1x1 and specified stride (1x1 or 2x2) and at the end is added to the output of the second convolutional layer. Strides in convolutional layers can be replaced by Max Pooling 2D layer. ResNet starts with convolutional layer, then several groups of residual modules with each group having one of its modules decrease the size by some factor (usually 2x2 stride), fully-connected (dense) block and one dense layer with Softmax activation (logistic regression layer) for classification.
An example of layers and connections in the residual module is presented in Figure 2.

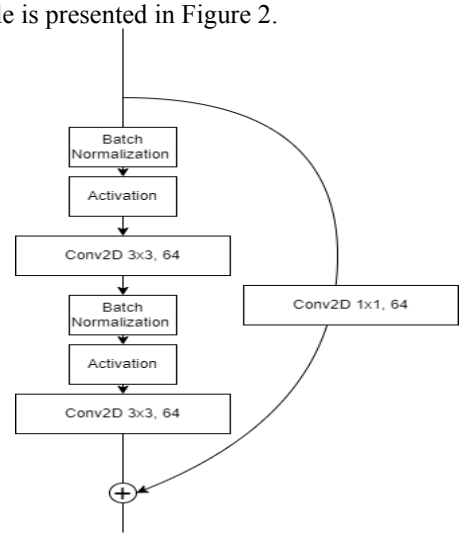


Figure 2: Residual module example.

Xception [19], similarly to ResNet, consists of Xception modules, which are residual modules but inside convolutional layers are replaced by depthwise separable convolutional layers. As the Xception architecture is based on the hypothesis that cross-channel and spatial correlations can be mapped completely separately, the input has to go through at least one convolutional layer which will help the hypothesis as cross-channel and spatial correlations tend to be very high in input images.

Simple convolutional neural network is straight flow, without any other connections consisting of multiple convolutional layers with batch normalization and activation, global max pooling, one or more fully-connected layers and logistic regression layer.

3.5. Data augmentation and supply

Data supply method can be a bottleneck during training process. One way to deal with this problem is to use tf.data.Dataset API [20] from Tensorflow library which can also help with data augmentation. In this experiment training data is loaded, cached then during training randomly augmented, shuffled and batched. Prefetch function is used to ready the data before the next epoch so the data preparation doesn’t happen at the start of the epoch. All the transformations applied in tf.data.Dataset

pipeline are executed on a CPU. Data augmentation consists of random hue, saturation, brightness and contrast adjustments within specified range. Random horizontal image flip and image rotation is applied by the model layers.

3.6. Research methodology

In this paper effectiveness of an activation function is defined to be the accuracy and value of the loss function computed on the test set. Efficiency on the other hand is defined as the prediction time, epochs required to converge and time required for a single epoch. Research has been conducted with the use of the stratified k-fold Cross Validation [21] for each dataset and activation function. For CIFAR-100 and Intel Image Classification we use variations of Xception architecture, while for CIFAR-10 and animals 10 ResNet and simple convolutional neural network has been employed respectively. Dataset is loaded, shuffled, split into training and test subsets (if there is no provided separate test set) then the training subset is split into 5 stratified subsets of even size. During training four of the splits are used for training and the remaining one for validation. This is repeated five times with different split used for validation. Each activation function is trained five times for every dataset. Training and validation loss and accuracy are recorded along with time spent on each epoch.

Each trained artificial network model uses cross-entropy as a loss function. Only the model with best loss value across all epochs is saved. Early stopping with 10 epoch patience is used to investigate the speed of convergence for each activation function. After training every model is evaluated for test loss, accuracy and prediction time.

During training process Adam [22] optimization algorithm with initial learning rate 0.001 and exponential decay for all models is used.

Activation function hyperparameters have been set to following values:

- $\alpha = 0.3$ for Leaky ReLU,
- $\alpha = 1.0$ for ELU.

4. Research results analysis

In this research more shallow models are trained, with fewer hidden layers and thus smaller number of trainable parameters compared to experiments conducted by Ramachandran et al. [3]. Results tables present mean values of all splits for each activation function.

Models trained on CIFAR-10 dataset use ResNet architecture with 283,595 trainable parameters and 1,628 non-trainable parameters (Table 2). It should be noted that it is deeper than wider as the number of filters is rather low. Batch size is set to 128. The results of experiment are presented in Table 3.

In this particular combination of dataset and architecture ELUs give the best accuracy and loss. It is important to note that ELUs have also trained for highest amount of epochs with mean of 65.4 (Table 3). Second come Swish activations with accuracy lower by value of 0.4% but with significantly lower number of epochs before training stop. ReLUs are fastest during prediction, ELU and LReLU have comparable prediction time. Difference in mean time spent on one epoch can be mostly attributed to resources taken by system.

Table 2: Model layers for CIFAR-10 dataset

Layer name	Layer description
Input	Shape (32,32,3)
Conv2D	Size 5x5, filters 32
Residual module 1	Size 3x3, filters 16, strides 2
Residual module 2	Size 3x3, filters 16
Residual module 3	Size 3x3, filters 16
Residual module 4	Size 3x3, filters 32, strides 2
Residual module 5	Size 3x3, filters 32
Residual module 6	Size 3x3, filters 32
Residual module 7	Size 3x3, filters 64, strides 2
Residual module 8	Size 3x3, filters 64
Residual module 9	Size 3x3, filters 64
Global Average Pooling 2D	
Dense	Units 128
Dense (Classification)	Units 128, activation Softmax

Table 3: Experiment results for CIFAR-10 dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.7758	0.6666	0.6526	0.55	46.2	21.99
LReLU	0.7806	0.6468	0.6275	0.58	60.6	21.38
ELU	0.7927	0.6159	0.6098	0.58	65.4	21.16
Swish	0.7881	0.6358	0.6132	0.62	51.2	22.36

Xception architecture was employed for training models on CIFAR-100 dataset with 1,421,956 trainable parameters and 6,208 non-trainable parameters (Table 4). Here Dropout layer has been used, for better generalization and Max Pooling for downsampling layers output instead of strides in convolutional layers. Batch size is set to 128.

Table 4: Model layers for CIFAR-100 dataset

Layer name	Layer description
Input	Shape (32,32,3)
Conv2D 1	Size 5x5, filters 64
Conv2D 2	Size 3x3, filters 96
Xception module 1	Size 3x3, filters 192, max pooling 2x2 strides 2
Xception module 2	Size 3x3, filters 256, max pooling 2x2 strides 2
Xception module 3	Size 3x3, filters 512, max pooling 2x2 strides 2
Separable Conv2D	Size 3x3, filters 512
Global Average Pooling 2D	
Dense	Units 512
Dropout	Rate 0.5
Dense (classification)	Units 100, activation Softmax

Table 5: Experiment results for CIFAR-100 dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.5736	1.7489	1.7613	2.17	33.4	53.4
LReLU	0.6118	1.6058	1.6264	2.45	43.6	52.6
ELU	0.6098	1.6210	1.6388	2.43	39.8	51.7
Swish	0.5981	1.6360	1.6647	2.84	26.6	59.9

We can see that LReLUs perform the best, with small difference to ELU activation (Table 5). Again we can see that worst performing function has trained for much lower number of epochs on average but Swish seems to have very fast convergence compared with other activa-

tions. Again Swish takes longest amount of time for predictions and single epoch.

For Animals 10 datasets we used Simple Convolutional neural network architecture with 598 218 trainable parameters and 1 792 non-trainable parameters (Table 6). In this particular architecture Global Max Pooling 2D has been used, instead of Global Average Pooling. Batch size is set to 8.

Table 6: Model layers for Animals 10 dataset

Layer name	Layer description
Input	Shape (112,112,3)
Conv2D 1	Size 7x7, filters 64, strides 2
Conv2D 2	Size 3x3, filters 64
Conv2D 3	Size 3x3, filters 128, strides 2
Conv2D 4	Size 3x3, filters 128
Conv2D 5	Size 3x3, filters 128, strides 2
Conv2D 6	Size 3x3, filters 128
Global Max Pooling 2D	
Dense	Units 128
Dense	Units 128
Dropout	Rate 0.5
Dense (classification)	Units 10, activation Softmax

Table 7: Experiment results for Animals 10 dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.6907	0.9131	0.9346	0.98	41.8	25.4
LReLU	0.6824	0.9405	0.9530	1.05	54.0	23.9
ELU	0.6782	0.9574	0.9937	1.05	44.2	26.5
Swish	0.6625	0.9897	0.9987	1.23	30.8	28.3

Surprisingly ReLUs result in the best mean test accuracy and loss, with second to Swish activations lowest number of epochs (Table 7). Again ReLUs have the shortest prediction time, LReLUs and ELUs perform similiarly in this regard and Swish activations take longest time to predict. Swish activations also have longest mean time spent on single epoch.

Models trained on Intel Image Classification dataset use Xception architecture with 668,038 trainable parameters and 4,736 non-trainable parameters (Table 8). In comparison to CIFAR-100 models here we have a bit deeper models but lower number of filters or units in layers. Batch size is set to 16.

Table 8: Model layers for Intel Image Classification dataset

Layer name	Layer description
Input	Shape (112,112,3)
Conv2D	Size 5x5, filters 64
Conv2D	Size 3x3, filters 64
Xception module 1	Size 3x3, filters 96, max pooling 2x2 strides 2
Xception module 2	Size 3x3, filters 128, max pooling 2x2 strides 2
Xception module 3	Size 3x3, filters 192, max pooling 2x2 strides 2
Xception module 4	Size 3x3, filters 256, max pooling 2x2 strides 2
Separable Conv2D	Size 3x3, filters 512
Global Average Pooling 2D	
Dense	Units 256
Dense	Units 128
Dropout	Rate 0.5
Dense (classification)	Units 6, activation Softmax

Table 9: Experiment results for Intel Image Classification dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.8546	0.4757	0.4770	1.70	32.4	44.5
LReLU	0.8561	0.4720	0.4546	1.76	41.4	46.8
ELU	0.8517	0.4663	0.4499	1.76	37.0	45.8
Swish	0.8594	0.4568	0.4485	2.05	31.4	49.6

In this combination of dataset and architecture we have the closest results between all activation functions in terms of accuracy and loss so far, with Swish taking the crown (Table 9). Again prediction time results are similar to previous results. Swish activations need lower number of epochs to converge but take the longest time for single epoch.

5. Conclusions

From the results a conclusion can be drawn that H1 and H2 are true but H3 is only partially true as the activation functions order of effectiveness and efficiency changes in every dataset and architecture combinations.

In regards to H1 it is true that although Swish activations take the longest time to train for a single epoch they also need significantly lower number of epochs to converge thus decreasing total training time. More complex models seem to favor activations that try to alleviate vanishing gradient and dead neurons problems although the functions are not consistent in their effectiveness across all datasets.

Without a doubt the H2 is true and the order doesn't change for different datasets as is stated in H3. Swish activations slow down prediction time by value ranging from ~12.7% up to ~30.9% compared to ReLUs. LReLUs and ELUs are comparable in terms of prediction time but are still slower than ReLUs by varying value of ~3.5% to ~13%. This may have considerable impact on performance of cloud services that employ ANN models especially in moments of high traffic.

It should be noted that Swish may be comparable in terms of effectiveness or possibly better than LReLU and ELU since Early Stopping ends training early before learning rate has a chance to decrease to a value which can make finer changes to the model. Another thing to note is that authors of Searching for activation functions [5] have made their experiments on deep models with higher number of parameters than the models presented in this paper. This has effect on the performance of activation functions as the selected functions try to alleviate vanishing gradient and dead neurons problems which are less likely to occur in shallow models.

Ultimately research on performance of activation functions is not nearly completed by the results presented in this article. Future experiments should focus on verifying activation functions efficiency and effectiveness for different value of learning rate and its decay, parameters affecting activation function themselves such as α in LReLU or ELU, broader selection of architectures and datasets.

References

[1] A. Abraham, Artificial neural networks. Handbook of measuring system design, John Wiley and Sons Ltd., London (2005) 901-908, https://doi.org/10.1002/0471497398.mm421.

[2] V. Nair, G. E. Hinton, Rectified Linear Units Improve Restricted Boltzmann Machines in Proceedings of the 27th International Conference on International Conference on Machine Learning, Omnipress, Madison (2010) 807-814.

[3] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv (2017), https://doi.org/10.48550/arXiv.1710.05941.

[4] A. Krizhevsky, V. Nair, G. E. Hinton, CIFAR-10 and CIFAR-100 datasets http://www.cs.toronto.edu/~kriz/cifar.html, [14.06.2022].

[5] D. A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), Published as a conference paper at ICLR 2016 (2015), https://doi.org/10.48550/arXiv.1511.07289.

[6] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv (2015), https://doi.org/10.48550/arXiv.1505.00853.

[7] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, arXiv (2018), https://doi.org/10.48550/arxiv.1811.03378.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean et al., TensorFlow: A System for Large-Scale Machine Learning, OSDI 16 (2016) 265-283.

[9] Keras, https://keras.io , [14.06.2022].

[10] F. Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12 (2011) 2825-2830, https://doi.org/10.48550/arXiv.1201.0490.

[11] G. Van Rossum, F. L. Drake, Python 3 Reference Manual, CA: CreateSpace, Scotts Valley, 2009.

[12] Anaconda platform website https://anaconda.org/, [14.06.2022].

[13] Animals 10 dataset https://www.kaggle.com/datasets/alessiocorrado99/animals10 , [14.06.2022].

[14] Intel Image Classification dataset https://www.kaggle.com/datasets/puneet6060/intel-image-classification , [14.06.2022].

[15] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, arXiv (2015), https://doi.org/10.48550/arXiv.1502.01852.

[16] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Journal of Machine Learning Research - Proceedings Track 9 (2010) 249-256.

[17] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, International conference on machine learning, PMLR 37 (2015) 448-456, https://doi.org/10.48550/arXiv.1502.03167.

[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition (2016) 770-778, https://doi.org/10.48550/arXiv.1512.03385.

[19] F. Chollet, Xception: Deep learning with depthwise separable convolutions, Proceedings of the IEEE conference on computer vision and pattern recognition (2017) 1251-1258, https://doi.org/10.1109/CVPR.2017.195.

[20] tf.data.Dataset API https://www.tensorflow.org/api_docs/python/tf/data/Dataset , [20.06.2022].

[21] P. Refaeilzadeh, L. Tang, H. Liu, Cross-Validation. Encyclopedia of Database Systems. Springer, Boston (2009), https://doi.org/10.1007/978-0-387-39940-9_565.

[22] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv (2014), https://doi.org/10.48550/arXiv.1412.6980.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 13–17
Received:14 September 2022
Accepted: 7 November 2022

Performance analysis of user interface implementation methods in mobile applications

Analiza wydajności metod implementacji interfejsów użytkownika w aplikacjach mobilnych

Jakub Szczukin*

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

Abstract
The purpose of this article is to analyze the impact of Jetpack Compose on user interface performance in mobile applications. A relatively new technology, Jetpack Compose, has not seen much research on its performance. The study used applications written in Kotlin, using the Jetpack Compose toolkit and views. The applications were tested with performance tests, using the Macrobenchmark tool, UI Automator 2 and JUnit 5. A literature review examining the impact of many factors on UI and Android performance was performed. In the end, upon completion of the testing, it was concluded that Jetpack Compose is slightly inferior in performance compared to interfaces built with views, in return it offers faster and easier code development.

Keywords: testing; user interface; jetpack compose; performance

Streszczenie
Celem artykułu jest analiza wpływu zastosowania Jetpack Compose na wydajność interfejsu użytkownika w aplikacjach mobilnych. Jetpack Compose jest stosunkowo nową technologią, przez co nie ma wiele badań na temat jej wydajności. Do badania zostały wykorzystane aplikacje napisane w języku Kotlin, przy użyciu zestawu narzędziowego Jetpack Compose oraz widoków. Aplikacje były testowane przy użyciu testów wydajnościowych, przy użyciu narzędzia Macrobenchmark, UI Automator 2 oraz JUnit 5. Został wykonany przegląd literatury badającej wpływ wielu czynników na wydajność interfejsu użytkownika i systemu Android. Po zakończeniu badań wywnioskowano, że Jetpack Compose jest nieznacznie gorszy wydajnościowo w porównaniu do interfejsów zbudowanych za pomocą widoków, w zamian oferuje szybszą i łatwiejszą pracę nad kodem.

Słowa kluczowe: testowanie; interfejs użytkownika; jetpack compose; wydajność

*Corresponding author
Email address: jakub.szczukin@pollub.edu.pl (J. Szczukin)

1. Wstęp

Historia urządzeń mobilnych jest stosunkowo krótka – pierwsze telefony komórkowe zostały pokazane w 1973r. w Nowym Jorku. Telefony zostały wykonane przez firmę Motorola, były dużych rozmiarów i ważyły około 2 kilogramy [1]. W 1979 r. Japonia uruchomiła pierwszą na świecie sieć telefonii komórkowej. Parę lat później,
w roku 1983 został wypuszczony na rynek pierwszy, masowo produkowany telefon komórkowy DynaTAC 8000x.

W następnych latach technologia urządzeń mobilnych szybko się rozwijała, głównie w kierunku rozmów głosowych oraz wiadomości tekstowych, do momentu prezentacji iPhone z systemem iOS oraz opublikowania pierwszej wersji systemu operacyjnego Android w roku 2007. Od tego momentu, rozwój urządzeń mobilnych kierował się w stronę wielofunkcyjnych urządzeń multimedialnych. Obecnie, obydwa systemy wyparły konkurencję i zajmują odpowiednio 29,24% oraz 70,01% urządzeń na rynku światowym [2].

Wraz z rozwojem technologii i rozpowszechnieniem urządzeń mobilnych wśród gospodarstw domowych, wzrosło zapotrzebowanie na coraz wydajniejsze urządzenia. Rozwiązaniem wydają się nowoczesne podzespoły, a także zoptymalizowane sposoby pisania nowych aplikacji. W tym artykule, starano się pokazać wpływ dwóch sposobów wdrażania interfejsu użytkownika na wydajność urządzenia mobilnego. Została postawiona teza, że interfejs użytkownika stworzony za pomocą Jetpack Compose jest wydajniejszy od dotychczasowych alternatyw. Potwierdzono wpływ zastosowania odpowiednio zoptymalizowanego kodu oraz techniki wykonania interfejsu użytkownika na jego wydajność.

2. Przegląd literatury

Autorzy artykułu „How resource utilization influences UI responsiveness of Android software” [3] prowadzali testy wytrzymałościowe oraz testy na aplikacji na urządzeniach z systemem Android w celu zrozumienia, jak wykorzystanie zasobów wpływa na responsywność systemu Android. Z przeprowadzonych badań wywnioskowano, że pojemność urządzenia oraz wielkość pamięci wpływają na responsywność UI tylko wtedy, gdy ich użycie zbliża się do 100%, natomiast najważniejszym aspektem dla wydajności aplikacji jest odpowiednie zarządzanie priorytetami wątków.

Następnym wybranym artykułem jest „Performance of Hybrid Mobile Application UI Frameworks" [4], skupiającym się na porównaniu najczęściej stosowanych, hybrydowych interfejsach użytkownika dla aplikacji mobilnych. Na zbudowanych aplikacjach przeprowadzono testy wydajnościowe, gdzie szczególną uwagę zwracano na szybkość ładowania elementów, płynność przewijania i płynność przechodzenia między stronami. W badaniu wzięło udział 4 ekspertów, którzy ocenili kryteria w stosunku od najmniej ważnych do najważniejszych. Do wyników wykorzystano obliczone wagi kryteriów. W otrzymanych wynikach zauważono korelację pomiędzy prostotą używanej biblioteki, a jej wydajnością.

Artykuł „The Effect of Representational UI Design Quality of Mobile Shopping Applications on Users' Intention Shop" [5] bada wpływ zwięzłości i spójności interfejsu użytkownika na użyteczność aplikacji i użytkowników. W badaniu wzięło udział ponad 230 studentów,
a otrzymane wyniki potwierdziły założone hipotezy – zwięzły i spójny UI pozytywnie wpływa na użyteczność aplikacji oraz intencję użytkowników do ponownego korzystania z aplikacji do zakupów.

Możliwość dostosowania interfejsu użytkownika jest również ważnym aspektem, który został zbadany w artykule pt. „Enhancing user experience through customization of UI design" [6]. W eksperymencie wzięło udział 50 studentów w wieku 18-23 lat, którzy odpowiadali na pytania z kwestionariusza dot. Komfortu korzystania
z UI. Na podstawie otrzymanych wyników, wywnioskowano, że możliwość dostosowania interfejsu przez użytkownika wpływa na komfort jego użytkowania.

Ostatnim artykułem odnoszącym się do niniejszej pracy jest „User Interface Matters: Analysing the Complexity of Mobile Applications from a Visual Perspective" [7]. Autorzy zaproponowali analizę interfejsu poprzez wprowadzenie odpowiednich wskaźników. W celu pokazania ich użyteczności, oceniono za ich pomocą aplikacje mobilne napisane podczas kursów programowania. Pozwoliły w prosty i przejrzysty sposób pokazać złożoność interfejsu, logiki i wydajności.

W powyższej literaturze, nie znaleziono porównań takich jak w niniejszym artykule, jednakże, stanowiły one bazę, na której zostały opracowane oraz zinterpretowane badania użyte w niniejszym artykule.

3. Plan i kryteria badań

Badania zostały przeprowadzone na aplikacji mobilnej napisanej w języku Kotlin [8] na system operacyjny Android. Aplikacja została podzielona na wiele aktywności, w skład których wchodzi lista z elementami (Listingi 1 - 3) oraz krótka animacja (Rysunek 1). Lista elementów została napisana przy użyciu Jetpack Compose [9], widoków oraz ich kombinacji. Do uruchomienia animacji jest wymagana interakcja użytkownika z wyświetlaczem urządzenia. Polega ona na przemieszczeniu w losowe miejsce 10 kolorowych kółek o identycznej średnicy. Do stworzenia animacji, ponownie wykorzystano zestaw Jetpack Compose oraz widoki wraz z elementami zapisanymi w formacie XML.

Listing 1: Część kodu odpowiedzialna za wyświetlanie listy elementów za pomocą widoków

```
@Composable
fun List(listContent: List<DataRow>){
    LazyColumn{ this: LazyListScope
        items(listContent.size){ index ->
            ListRow(listContent[index])
        }
    }
}

@Composable
fun ListRow(obj: DataRow) {
    Row{ this: RowScope
        Column(modifier = Modifier.padding(10.dp)){ this: ColumnScope
            Text(text = obj.id.toString(), fontSize = 30.sp)
        }
        Column{ this: ColumnScope
            Row{ this: RowScope
                Text(text = "test", fontSize = 18.sp)
            }
            Row{ this: RowScope
                Text(text = obj.content, fontSize = 18.sp)
            }
        }
    }
}
```

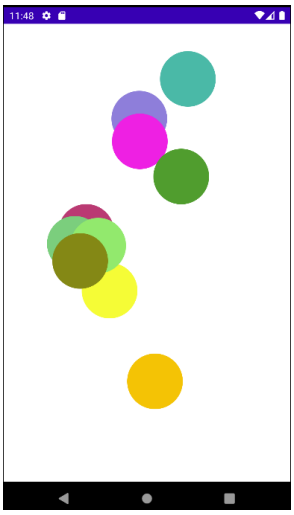
Listing 2: Część kodu odpowiedzialna za wyświetlenie listy elementów za pomocą widoków

```
    val linearLayoutManager = LinearLayoutManager( context: this)
    val adapter = ListAdapter(SampleData)
    val recyclerView = findViewById<RecyclerView>(R.id.itemList)
    recyclerView.layoutManager = linearLayoutManager
    recyclerView.adapter = adapter
```

Listing 3: Część kodu odpowiedzialna za wyświetlenie listy elementów za pomocą kombinacji Jetpack Compose i widoków

```
val composeView = findViewById<ComposeView>(R.id.compose_view)
val sampleData = SampleData.listContent

composeView.setContent{
    LazyColumn{ this: LazyListScope
        items(sampleData.size){ this: LazyItemScope
            ListRow(sampleData[it])
        }
    }
}
```

Do odpowiedniego przeprowadzenia eksperymentu zastosowano narzędzie Macrobenchmark [10] zintegrowany z IDE Android Studio do przeprowadzania testów wydajnościowych na platformę Android. Przy pomocy bibliotek JUnit oraz UIAutomator zostało przeprowadzone 30 testów mierzących czas uruchomienia aplikacji oraz 9 testów zliczających czas potrzebny procesorowi do wyrenderowania pojedynczej klatki. Każda aplikacja, została poddana identycznej liczbie testów obciążających interfejs jednakowymi działaniami, z których wyliczono średnie statystyki dotyczące wydajności.



Rysunek 1: Przykładowe położenie kulek po uruchomieniu animacji.

Przeprowadzone testy brały pod uwagę różne tryby uruchomienia aplikacji. Pierwszym z nich jest „tryb zimny" (ang. Cold), dla którego wpierw został tworzony proces aplikacji, a następnie utworzona nowa aktywność. Następnie badano „tryb ciepły" (ang. Warm), dla którego proces aplikacji istnieje, lecz jest potrzebne stworzenie aktywności. Ostatnim sposobem uruchamiania aplikacji jest „tryb gorący" (ang. Hot), gdzie istniejąca aktywność jest przenoszona na pierwszy plan, w już działającym procesie.

Badania zostały powtórzone na fizycznym urządzeniu, działającym pod kontrolą systemu Android. Wspomniane urządzenia mobilne posiadały różne parametry, tj. wersja systemu operacyjnego, procesor, procesor graficzny, pamięć RAM, rozdzielczość oraz liczba pikseli przypadających na cal długości wyświetlacza (Tabela 1).

Tabela 1: Parametry urządzeń mobilnych

	Xiaomi Redmi 9	Samsung Galaxy A3	Motorola Moto G50
Model	M2003J15SG	SM-A300FU	Moto G(50)
CPU	Octa-core (2x2.0 GHz Cortex-A75 & 6x1.8 GHz Cortex-A55)	Quad-core 1.2 GHz Cortex-A53	Gualcomm Snapdragon 480 8x2.00 GHz
GPU	Mali-G52 MC2	Adreno 306	Adreno 619
RAM	4 GB	1,5 GB	4 GB
System operacyjny	Android 11 RP1A.200720.011 (API level 30)	Android 6.0.1 (API level 23)	Android 12 S1RFS32.27-25-1 (API level 31)
Wyświetlacz	1080x2340 px, 395 ppi, 60Hz	720x1280px, 312 ppi, 60Hz	720x1600px, 270ppi, 90Hz

Przeprowadzone testy skupiały się głównie na zmierzeniu czasu potrzebnego urządzeniu, na przeprowadzenie danej operacji na interfejsie użytkownika, w danej technologii. Czasy wykonania testów zostały zmierzone za pomocą narzędzia Macrobenchmark, zintegrowanego z IDE Android Studio. Pierwszy z przeprowadzonych testów miał za zadanie zmierzyć czas potrzebny urządzeniu na uruchomienie aplikacji oraz przejście do aktywności z badanym interfejsem. Przykładowa część kodu została przedstawiona poniżej (Listing 4).

Następne przeprowadzone testy miały za zadanie zmierzyć czas zbudowania pojedynczej klatki przez procesor. W tym wypadku, ostatecznym wynikiem jest suma dwóch wątków, pracujących nad budową interfejsu graficznego. Do badania przygotowano dwie, testowane aktywności. Pierwsza z nich zawierała dynamiczną listę z elementami, druga – animację wykonaną w danej technologii. Na tych aktywnościach, zasymulowano aktywność użytkownika za pomocą biblioteki UIAutomator (Listingi 5 – 6).

Listing 4: Część kodu odpowiedzialna za mierzenie czasu startowego aplikacji

```
@LargeTest
@RunWith(Parameterized::class)
class StartupBenchmark(private val startupMode: StartupMode) {
    @get:Rule
    val benchmarkRule = MacrobenchmarkRule()

    @Test
    fun startup() = benchmarkRule.measureRepeated(
        packageName = TARGET_PACKAGE,
        metrics = listOf(StartupTimingMetric()),
        compilationMode = CompilationMode.DEFAULT,
        startupMode = startupMode,
        iterations = ITERATIONS,
        setupBlock = { this: MacrobenchmarkScope
            pressHome()
        }
    ){ this: MacrobenchmarkScope
        val intent = Intent()
        intent.action = "$packageName.ComposeActivity"
        startActivityAndWait(intent)
    }
```

Listing 5: Część kodu odpowiedzialna za symulację przewijania listy

```
){ this: MacrobenchmarkScope
    val column = device.findObject(By.scrollable( isScrollable: true))

    scrollList(column, device)
}
```

Listing 6: Część kodu odpowiedzialna za symulację kliknięć na ekranie

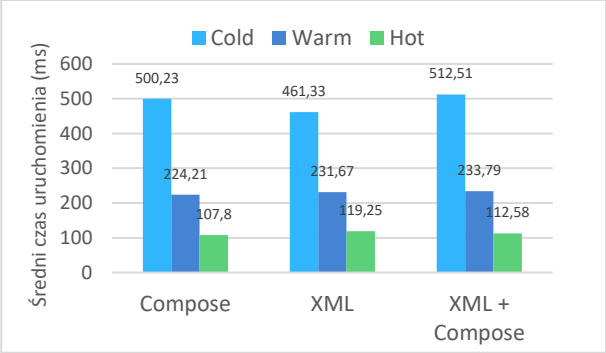
```
){ this: MacrobenchmarkScope
    val canvas = device.findObject(By.clazz( className: "android.view.View"))

    for(i in 0 .. 4){
        canvas.click()
        sleep( millis: 2000)
    }
}
```

4. Wyniki

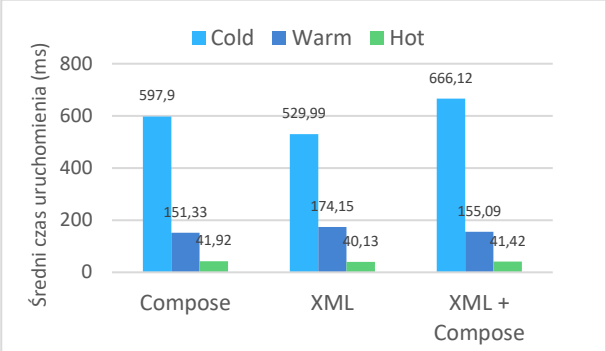
Początkowo zostały przeprowadzone testy mierzące czas startowy aplikacji. Badanie przeprowadzono na liście ze statycznymi elementami. Otrzymane wyniki zostały przedstawione poniżej. Na podstawie otrzyma-

nych wyników, zostały przeprowadzone odpowiednie obliczenia (Rysunki 2 – 4).

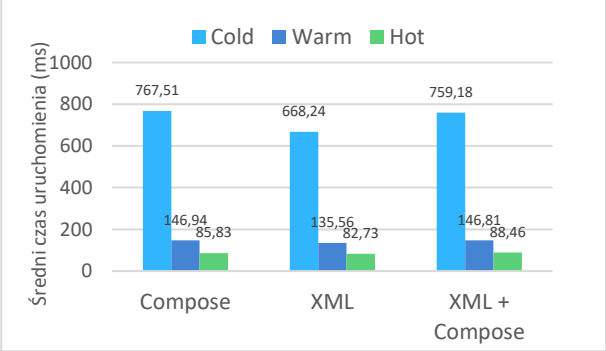
Biorąc pod uwagę poniższe wyniki, można zauważyć, że najlepsze, ogólne wyniki uzyskała aktywność zbudowana przy pomocy widoków i plików XML. Niezależnie od typu urządzenia, posiada również najniższy czas uruchomienia pierwszego, „zimnego" startu aplikacji. Jetpack Compose osiąga nieco gorsze wyniki, oscylując w granicach 5% - 10% wyników poprzednika. Najgorzej w całym zestawieniu poradziła sobie kombinacja obu technologii, osiągając największy czas uruchomienia dla większości trybów startowych.



Rysunek 2: Wyniki pomiarów czasu startowego aplikacji dla telefonu Xiaomi Redmi 9.



Rysunek 3: Wyniki pomiarów czasu startowego aplikacji dla telefonu Samsung Galaxy A3.



Rysunek 4: Wyniki pomiarów czasu startowego aplikacji dla telefonu Motorola Moto G50.

Następnym przeprowadzonym badaniem jest zliczenie czasu budowania pojedynczej klatki przez procesor. W tym celu badania zostały przeprowadzone na liście elementów, która podczas testów była automatycznie przewijana. Każdy test został wykonany na „zimnym" trybie startowym. Ponieważ interfejs użytkownika jest generowany na dwóch wątkach MainThread oraz RenderThread, zaprezentowane niżej wyniki, są sumą czasów działania tych wątków. Zostały one przedstawione w formie średniej arytmetycznej, odchylenia standardowego, percentyli wielkości 50, 90, 95 i 99 oraz wartości minimalnych i maksymalnych (Tabele 2-4).

Z otrzymanych wyników wynika, że technologia, w której wykonany został interfejs użytkownika nie ma wielkiego wpływu na jego płynność. Różnica pomiędzy wynikami z danych metod jest niewielka. Największe odchyły od pomiarów wykazał jednak Jetpack Compose, co może powodować niestabilność pracy w wybranych przedziałach czasowych. Najmniejszym czasem budowania klatki podczas animacji przewijania listy, cechuje się kombinacja obu badanych technologii. Osiągnęła ona najstabilniejsze i najszybsze wyniki spośród badanych.

Tabela 2: Wyniki czasu potrzebnego do zbudowania pojedyńczej klatki przez procesor dla telefonu Xiaomi Redmi 9

	Compose	XML	XML + Compose
P50	23,94 ms	29,13 ms	13,66 ms
P90	30,82 ms	32,71 ms	30,06 ms
P95	32,51 ms	33,85 ms	31,27 ms
P99	61,53 ms	61,27 ms	48,82 ms
Śr.	22,61 ms	27,19 ms	17,34 ms
Odch. St.	10,99 ms	9,44 ms	8,42 ms
Min	10,32 ms	8,45 ms	8,73 ms
Max	125,58 ms	90,05 ms	64,15 ms

Tabela 3: Wyniki czasu potrzebnego do zbudowania pojedyńczej klatki przez procesor dla telefonu Samsung Galaxy A3

	Compose	XML	XML + Compose
P50	9,20 ms	10,11 ms	8,95 ms
P90	16,07 ms	16,45 ms	15,94 ms
P95	21,24 ms	20,85 ms	21,25 ms
P99	34,01 ms	28,61 ms	34,32 ms
Śr.	11,12 ms	11,12 ms	10,82 ms
Odch. St.	5,49 ms	5,05 ms	5,55 ms
Min	4,83 ms	4,83 ms	2,66 ms
Max	48,56 ms	51,41 ms	54,74 ms

Tabela 4: Wyniki czasu potrzebnego do zbudowania pojedyńczej klatki przez procesor dla telefonu Motorola Moto G50

	Compose	XML	XML + Compose
P50	6,13 ms	5,59 ms	6,12 ms
P90	8,83 ms	7,53 ms	8,66 ms
P95	10,65 ms	9,13 ms	10,20 ms
P99	24,80 ms	13,39 ms	21,22 ms
Śr.	6,57 ms	5,94 ms	6,45 ms
Odch. St.	4,02 ms	1,83 ms	3,54 ms
Min	2,32 ms	2,38 ms	2,63 ms
Max	46,33 ms	16,84 ms	39,63 ms

Powyższe badanie, zliczające czas zbudowania pojedynczej klatki, zostało również wykonane na aktywnościach zawierających ok. 10-sekundową animację, poruszających się kół. Zebrane wyniki przedstawiono poniżej (Tabele 5-7).

Tabela 5: Wyniki czasu potrzebnego do zbudowania pojedynczej klatki przez procesor dla telefonu Xiaomi Redmi 9

	Compose	XML
P50	11,55 ms	7,39 ms
P90	14,60 ms	8,29 ms
P95	15,64 ms	8,72 ms
P99	17,49 ms	10,48 ms
Śr.	12,04 ms	7,23 ms
Odch. St.	6,34 ms	1,17 ms
Min	5,06 ms	2,59 ms
Max	143,95 ms	14,14 ms

Tabela 6: Wyniki czasu potrzebnego do zbudowania pojedynczej klatki przez procesor dla telefonu Samsung Galaxy A3

	Compose	XML
P50	10,02 ms	7,34 ms
P90	16,39 ms	10,58 ms
P95	20,63 ms	12,96 ms
P99	28,49 ms	15,54 ms
Śr.	11,26 ms	7,92 ms
Odch. St.	5,33 ms	2,14 ms
Min	4,48 ms	4,22 ms
Max	84,67 ms	24,41 ms

Tabela 7: Wyniki czasu potrzebnego do zbudowania pojedynczej klatki przez procesor dla telefonu Motorola Moto G50

	Compose	XML
P50	6,85 ms	5,57 ms
P90	8,38 ms	7,29 ms
P95	9,12 ms	7,72 ms
P99	13,60 ms	8,83 ms
Śr.	6,63 ms	5,57 ms
Odch. St.	2,83 ms	1,36 ms
Min	2,75 ms	1,62 ms
Max	55,62 ms	10,72 ms

Wyniki ostatniego badania ponownie pokazują niewielką niestabilność pracy podczas animacji wykonanych w Jetpack Compose. Osiąga zarówno najwyższy, maksymalny jak i minimalny czas budowania klatki od alternatywnej opcji. Pomimo tego, obydwie metody wykonały płynne animacje, z małą różnicą pomiędzy ich średnim czasem wykonania zadania.

6. Wnioski

Przeprowadzone badania pozwalają na wyciągnięcie następujących wniosków:

Zastosowanie widoków w aplikacji jest efektywniejszą opcją od zestawu narzędziowego Jetpack Compose, oraz związanymi z nim opcjami. Zarówno czas startowy, jak i czas potrzebny do wyrenderowania pojedynczej klatki okazał się mniejszy w aktywnościach zbudowanych przy pomocy widoków. Nie dotyczy to aplikacji działających głównie w tle, gdzie przywrócenie ich na pierwszy plan zajmuje krócej, gdy interfejs użytkownika jest zbudowany w Jetpack Compose.

Stabilność animacji wydaje się problemem w obecnej wersji 1.1.1 Jetpack'a Compose. Starsze urządzenia osiągają znacznie gorsze wyniki korzystając z animacji Compose. Wpływ na to mogą mieć na to zarówno parametry testowanych urządzeń fizycznych, jak i obecne w testowanej wersji Jetpack Compose, eksperymentalne animacje. W pozostałych przypadkach, obie metody osiągają podobne rezultaty.

Literatura

[1] Wywiad z Martinem Cooperem, wynalazcą telefonów komórkowych, http://news.bbc.co.uk/1/hi/programmes/click_online/8639590.stm, [21.09.2022].

[2] Udział w rynku mobilnych systemów operacyjnych na świecie, https://gs.statcounter.com/os-market-share/mobile/worldwide, [26.01.2021].

[3] J. Fu, Y. Wang, Y. Zhou, X. Wang, How resource utilization influences UI responsiveness of Android software, Information and Software Technology 141 (2022)1-11, https://doi.org/10.1016/j.infsof.2021.106728.

[4] R. Vala, R. Jasek, Performance of Hybrid Mobile Application UI Frameworks, in: V. Mladenov, I.Rudas, O. Martin, G. Tsenov, P. M. Pardalos, M. Hromada, Proceedings of the 2014 Intertnational Conference on Applied Mathematics, Computational Science & Engineering (AMCSE 2014), Varna, Bulgaria, September 13-15, 2014.

[5] W. Jung, The Effect of Representational UI Design Quality of Mobile Shopping Applications on Users Intention to Shop, Procedia Computer Science 121 (2017) 166-169.

[6] S. L. T. Hui, S. L. See, Enhancing user experience through customisation of UI design, Procedia Manufacturing 3 (2015) 1932–1937.

[7] L. Corrala, I. Fronzab, T. Mikkonenc, User Interface Matters: Analysing the Complexity of Mobile Applications from a Visual Perspective, Procedia Computer Science 191 (2021) 9-16.

[8] Informacje o języku kotlin, https://kotlinlang.org/docs/home.html, [21.09.2022].

[9] Informacje o narzędziu Jetpack Compose, https://developer.android.com/jetpack/compose/documentation, [21.09.2022].

[10] Dokumentacja narzędzia Macrobenchmark, https://developer.android.com/topic/performance/benchmarking/macrobenchmark-overview, [21.09.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 18–22
Received: 17 September 2022
Accepted: 29 September 2022

Comparative analysis of web applications implemented in: PHP and Python

Analiza porównawcza aplikacji webowych napisanych w językach: PHP oraz Python

Jakub Zborowski*, Maciej Pańczyk

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

Abstract

This article presents a comparative analysis of two web applications implemented in PHP and Python. Test applications were created and equipped with the same functionality used in tests consisting in measuring the server response times to INSERT, SELECT, UPDATE and DELETE requests - handling database operations. The purpose of the research was to compare both languages in terms of selected criteria. Their performance, source code volume and popularity were compared.

Keywords: web application; performance analysis; PHP; Python

Streszczenie

Artykuł przedstawia analizę porównawczą dwóch aplikacji webowych napisanych w językach: PHP oraz Python. Stworzono aplikacje testowe, które zostały wyposażone w tą samą funkcjonalność wykorzystaną w badaniach polegających na pomiarze czasów odpowiedzi serwera na żądania typu INSERT, SELECT, UPDATE i DELETE – obsługujące operacje na bazie danych. Celem badań było porównanie obydwu języków pod względem wybranych kryteriów. Porównywano ich wydajność, objętość kodu źródłowego oraz popularność.

Słowa kluczowe: aplikacja webowa; analiza wydajności; PHP; Python

*Corresponding author

Email address: **j_zborowski@o2.pl** (J. M. Zborowski)

1. Wstęp

Aplikacje internetowe w dzisiejszych czasach stają się coraz bardziej popularne wśród użytkowników co powoduje, że powoli wypierają one z naszego życia aplikacje desktopowe.

Bardzo prężnie rozwijająca się technologia umożliwia wykorzystanie dużej ich elastyczności. Aby móc korzystać z aplikacji internetowej, wymagane jest wyłącznie posiadanie przeglądarki – dzięki temu ignorujemy konieczność pobierania i instalowania programu na każdym z urządzeń, przez co zyskujemy wygodny dostęp do jego funkcji, bądź plików zarówno z telefonu, tableta jak i komputera. Pozwala to na szybki wgląd do plików oraz przesyłanie wiadomości z dowolnego miejsca. Aplikacje desktopowe borykają się również z koniecznością systematycznych aktualizacji - nie doświadczymy tego rodzaju problematycznych przerw w przypadku korzystania z wersji webowych. Warto pamiętać też o wymaganiach systemowych, koniecznych do zainstalowania oprogramowania desktopowego. Biorąc pod uwagę aplikacje internetowe wszystko, czego potrzebujemy, to przeglądarka, możemy więc korzystać z jej funkcji bez konieczności zamartwiania się, czy nasz sprzęt ma odpowiednie parametry, aby poradzić sobie z wymaganiami programu. Wszystkie te zalety składają się na dużą popularność aplikacji internetowych.

Dodatkowo pisząc aplikację możemy użyć szkieletów programistycznych, co znacząco ułatwi proces tworzenia. Zaprojektowano je w taki sposób, aby zapewnić elastyczność, mają one również dobrą logikę wewnętrzną, którą narzucają stworzonej aplikacji. Korzystając z tego rozwiązania programista pisze mniej kodu, a szkielety programistyczne same w sobie są dobrze zaprojektowane i przetestowane – co czyni je niezawodnymi.

2. Przegląd literatury

Języki PHP oraz Python są językami, które zajmują czołowe pozycje w zakresie tworzenia stron i aplikacji internetowych. Dla języka PHP jednym z najbardziej popularnych i profesjonalnych szkieletów programistycznych jest szkielet Symfony, natomiast dla języka Python czołowym, a jednocześnie dosyć starym szkieletem jest szkielet Django. W przypadku aplikacji internetowych bardzo ważną kwestią jest ich jakość i związana z nią wydajność. W związku z tym testowanie aplikacji tego typu jest skomplikowanym oraz pracochłonnym procesem.

W artykule [1] autorzy Kai Lei, Yining Ma oraz Zhi Tan porównali i ocenili technologie służące do budowy aplikacji internetowych: PHP, Python i Node.js. Badania zrobiono, wykonując identyczne testy na aplikacjach posiadających te same funkcjonalności, ale zbudowanych przy użyciu tych trzech różnych technologii. Testy aplikacji wyświetlającej tekst „Hello World!”, obliczających i zwracających dziesiąty, dwudziesty i trzydziesty wyraz ciągu Fibonacciego, jak i przeprowadzających

operacje na bazie danych, w większości przypadków wykazały, że w stosunku do pozostałych testowanych technologii największą wydajnością cieszy się środowisko Node.js.

W artykule [2], autorzy Giuseppe Antonio Di Lucca oraz Anna Rita Fasolino podkreślają istotność testowania aplikacji pod względem wydajności przy zastosowaniu różnych obciążeń osiąganych poprzez zwiększanie liczby zapytań oraz wysyłanie różnej wielkości danych. W tego typu badaniach podstawową wykorzystywaną miarą jest czas wykonania poszczególnych czynności i odpowiedzi serwera.

W artykule [3], autor Klaus Purer porównał technologie służące do budowy aplikacji internetowych: PHP, Python oraz Ruby. Na początku została porównana semantyka i składnia, następnie pomocne cechy, takie jak: wyłapywanie błędów, abstrakcja relacyjnej bazy danych czy funkcjonalne cechy języków. Autor również porównuje języki pod względem bezpieczeństwa i finalnie również wydajności. Po przeprowadzonych badaniach okazało się, że jedynie w obszarze popularności i dostępności PHP był na pierwszym miejscu, a w kwestii wydajności wszystkie trzy technologie były na równi. W pozostałych, tj.: czytelność, użyteczność, bezpieczeństwo, abstrakcja bazy danych, wyłapywanie błędów oraz funkcjonalne cechy - Python delikatnie zwyciężył nad językiem Ruby, a język PHP zdecydowanie odstawał. Ostatecznie autor podkreśla, że wybór języka jest kwestią indywidualną, a dyskusje o językach są emocjonalne i irracjonalne.

Dotychczas żaden z badaczy nie pokusił się o porównanie popularności języków PHP oraz Python na podstawie liczby wyszukań, co zostało przeze mnie zrobione. Dodatkowo w wyżej wymienionych artykułach wykorzystane zostały starsze wersje języków.

3. Porównywane języki

3.1. PHP

Język PHP (ang. Hypertext Preprocessor) to skryptowy język programowania działający po stronie serwera, który został stworzony w roku 1994 przez Rasmusa Lerdorfa [4]. W rankingu popularności języków programowania TIOBE [5] z sierpnia 2022 roku, zajmuje on całkiem wysokie dziesiąte miejsce.

W zakresie zastosowania na stronach internetowych język PHP jest w ścisłej czołówce. Wedle ankiety przedstawionej w serwisie W3Techs język PHP jest używany przez 77.4% stron, dla których język programowania po stronie serwera jest znany [6]. Dodatkowo PHP jest wykorzystywany na wielu popularnych stronach, np.: Facebook, Instagram, WordPress czy Wikipedia.

3.2. Python

Język Python to język programowania wysokiego poziomu ogólnego przeznaczenia, którego pierwsza wersja ukazała się w 1991 roku, a jego twórcą jest Guido van Rossum [7]. Charakteryzują go takie cechy jak:

- jest w pełni obiektowy,
- umożliwia programowanie w różnych stylach (strukturalne, obiektowe, funkcyjne, itd.),
- jest językiem interpretowanym,
- wcięcia w kodzie są wykorzystywane do tworzenia bloków.

Obecnie jest on używany przez setki tysięcy programistów, co podkreśla jego miejsce w rankingu popularności języków programowania TIOBE z sierpnia 2022 roku, gdzie Python zajmuje pierwsze miejsce – tuż nad językiem C oraz językiem Java.

4. Metoda badań

4.1. Opis eksperymentu

Analizę porównawczą aplikacji webowych, które zostały napisane w językach PHP oraz Python przeprowadzono na podstawie trzech scenariuszy badawczych, w ramach których wykonano:

1. pomiar czasów wykonywania się aplikacji służących do operowania na bazie danych, zmierzony za pomocą dostępnych funkcji,
2. pomiar objętości kodu aplikacji zaimplementowanych na rzecz danego polecenia, polegający na zliczeniu liczby linijek kodu,
3. badanie popularności porównywanych języków poprzez sprawdzenie liczby ich wyszukań na wybranych platformach.

Badanie czasów wykonywania się aplikacji zostało podzielone na 4 główne operacje na bazie danych: INSERT, SELECT, UPDATE oraz DELETE. Każde z nich było badane w 5 scenariuszach różniących się liczbą rekordów, na których były wykonywane: 1, 10, 100, 1000 i 10000 rekordów. Dla każdego scenariusza aplikacja wykonywała się 10-krotnie, aby wyniki były bardziej wiarygodne. Następnie została wyliczona z nich średnia, która jest widoczna na rysunkach w kolejnym rozdziale.

Do generowania niezbędnych danych potrzebnych do operowania na bazie danych został wykorzystany pakiet Faker [8], który dla języka Python został stworzony na podstawie inspiracji biblioteką Faker, stworzoną dla języka PHP [9]. Przy wykorzystaniu odpowiednich funkcji umożliwiają one generowanie imion, nazwisk, liczb czy nawet adresów.

4.2. Środowisko testowe

W tabeli 1 pokazano środowisko testowe, na którym przeprowadzono badania. Oprócz parametrów komputera znajdują się tu również wersje analizowanych języków programowania.

Według artykułu opublikowanego na stronie Hackr.io [10] przez Amana Goela pracującego w Indyjskim Instytucie Technologii w Bombaju, który zgromadził ponad 50 tysięcy obserwatorów na swoim profilu na portalu społecznościowym dla programistów - Linkedin [11] - wersje 5.x języka PHP były wolne, jednakże nowsze wydanie, 8.1, czyni ten język zdecydowanie szybszym. Prawie 3-krotnie szybszym od typowego programu napisanego w języku Python.

Tabela 1: Specyfikacja urządzenia testowego

Sprzęt	
Procesor	AMD Ryzen 5 3600X
Pamięć RAM	16,0 GB
System operacyjny	Windows 10 Home 64bit
Dysk	TOSHIBA HDWD120
Oprogramowanie	
PHP	8.1.6
Python	3.7.0

4.3. Pomiar objętości kodu

W scenariuszu 2 zliczono liczbę linii kodu dla czterech porównywanych operacji na bazie danych, które w każdym z języków realizowały te same zadania. Wiersze zawierające nawiasy klamrowe lub puste wiersze zostały pominięte w pomiarach.

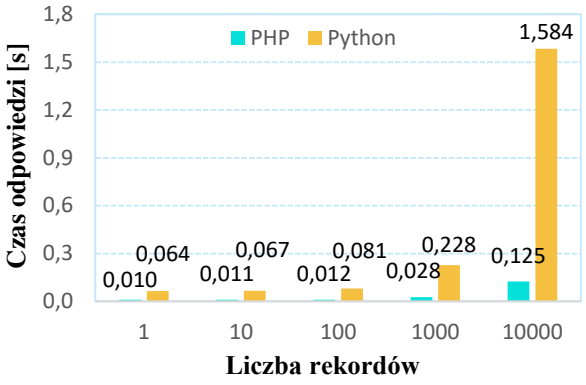
4.4. Badanie popularności

W scenariuszu 3 przeprowadzono badanie popularności języków poprzez sprawdzenie liczby ich wyszukań na trzech wybranych platformach. Pierwszą z nich jest najpopularniejsza wyszukiwarka – Google [12], która zazwyczaj jest wybierana jako pierwsza, gdy ludzie czegoś nie wiedzą – tutaj wzięto pod uwagę jedynie wyniki, które dokładnie posiadały szukane słowo. Jako drugą platformę wybrano jedno z najbardziej popularnych forów programistycznych - Stack Overflow [13], na którym brano pod uwagę jedynie wątki, które posiadały jakąkolwiek odpowiedź. Ostatnią platformą, która została wybrana jest GitHub [14] - hostingowy serwis internetowy przeznaczony do projektów programistycznych, będący obecnie najpopularniejszym hostem projektów open source. Dla GitHuba wzięto pod uwagę liczbę repozytoriów, aczkolwiek pod każdym innym względem również górował ten sam język.

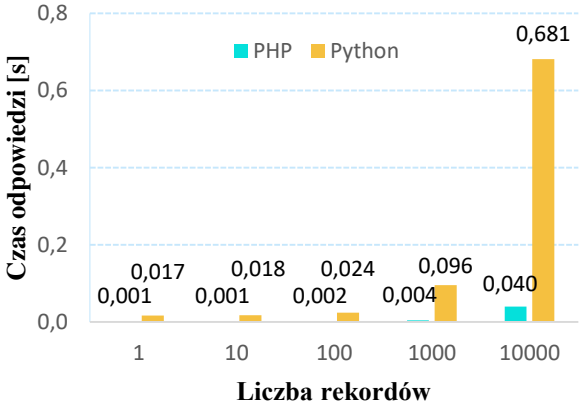
5. Wyniki

5.1. Pomiar czasów

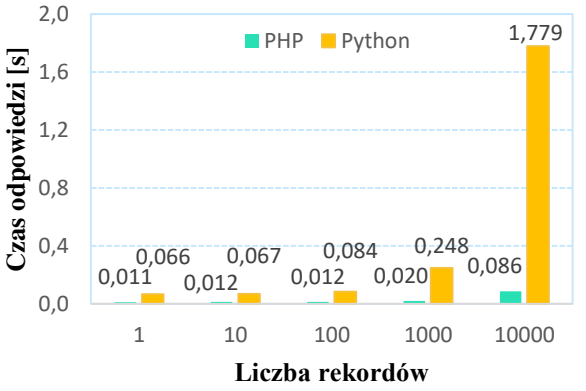
Na poniższych rysunkach przedstawiono zebrane wyniki dla scenariusza 1, w ramach którego przeprowadzono testy dla poszczególnych operacji na bazie danych: zapisu, odczytu, aktualizacji i usuwania. Zostały one wykonane na różnej liczbie rekordów, począwszy od 1, następnie 10, 100, 1000 oraz 10000 rekordów.



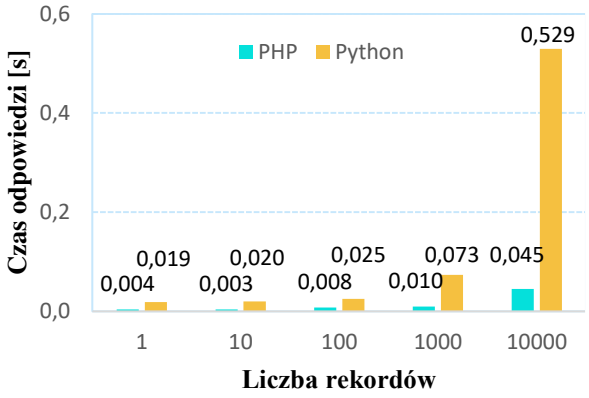
Rysunek 1: Czas wykonania zapytania INSERT [s].



Rysunek 2: Czas wykonania zapytania SELECT [s].



Rysunek 3: Czas wykonania zapytania UPDATE [s].



Rysunek 4: Czas wykonania zapytania DELETE [s].

Powyższe rysunki ukazują średnie czasy wykonywania się programów przy różnej liczbie rekordów, co jest równoznaczne ze zróżnicowanym obciążeniem.

Z wykresów tych widać, że przeskok pomiędzy 1, 10 oraz 100 rekordami dla każdego z języków jest stosunkowo niewielki, jednakże porównując je ze sobą widać zdecydowaną różnicę. Dla polecenia DELETE przy działaniu na jednym rekordzie PHP okazał się około 5 razy szybszy, biorąc pod uwagę polecenia INSERT oraz UPDATE wciąż tylko na jednym rekordzie jest to już 6-krotna przewaga na korzyść języka PHP. Gdy przejdziemy natomiast do polecenia SELECT przewaga rośnie dramatycznie i Python przy działaniu na jednym rekordzie jest już 17 razy wolniejszy.

Patrząc na różnicę pomiędzy 100 oraz 1000 rekordów dla języka PHP wynosi ona maksymalnie 16 milisekund - dla polecenia INSERT. Przechodząc do języka Python są to już różnice rzędu 147ms, 72ms, 164ms oraz 48ms dla poszczególnych poleceń. Porównując obydwa języki przy liczbie 1000 rekordów: przewaga na korzyść PHP jest 8-krotna dla polecenia INSERT, 24-krotna dla polecenia SELECT, 12-krotna dla polecenia UPDATE i 7-krotna dla polecenia DELETE.

Gdy spojrzymy na przeskok z 1000 na 10000 rekordów dla języka PHP: czas wykonywania się programu dla poleceń INSERT, UPDATE oraz DELETE zwiększył się około 4,5 raza, a dla polecenia SELECT 10-krotnie. Biorąc pod uwagę ten sam przeskok dla języka Python: dla każdego z poleceń wynosi on około 7-krotny wzrost.

Porównując wyniki przy 10000 rekordów: przy poleceniu INSERT język PHP okazał się ponad 12-krotnie szybszy, przy poleceniu SELECT 17-krotnie, przy poleceniu UPDATE ponad 20-krotnie, natomiast przy poleceniu DELETE jest to przewaga ponad 11-krotna.

5.2. Pomiar objętości kodu

Średnia liczba linii kodu przypadająca na implementację funkcjonalności w języku PHP wynosi: 19,75 (tabela 3). W przypadku języka Python średnia liczba linii kodu przypadająca na implementację funkcjonalności wynosi: 26,75 (tabela 4).

Tabela 3: Objętość kodu dla języka PHP

Funkcjonalność	Liczba linii kodu
INSERT	21
SELECT	19
UPDATE	23
DELETE	16

Tabela 4: Objętość kodu dla języka Python

Funkcjonalność	Liczba linii kodu
INSERT	27
SELECT	28
UPDATE	27
DELETE	25

Można zauważyć, że do zaimplementowania tych samych funkcjonalności język PHP potrzebował mniejszej objętości kodu. Dla języka Python było potrzebne o 35% więcej linijek kodu. Wynika to m.in. z tego, że w składni Pythona obowiązuje technika „wcięć", a przez to niektóre instrukcje, możliwe do zapisania w innych językach programowania w jednej linijce trzeba rozdzielać na kilka.

5.3. Badanie popularności

W tabeli 2 przedstawiono porównanie popularności języków PHP oraz Python poprzez sprawdzenie liczby wyszukań tychże języków w wyszukiwarce Google, wątków z udzieloną jakąkolwiek odpowiedzią na serwisie społecznościowym Stack Overflow oraz liczby repozytoriów na serwisie internetowym GitHub.

Tabela 2: Porównanie popularności języków

Język	Rok wydania	Google	Stack Overflow	GitHub
PHP	1994	844 000 000	1 275 000	851 000
Python	1991	712 000 000	1 722 000	2 708 000

Wyniki te pokazują, że PHP zgromadził więcej wyszukań w Google, jednakże Python posiada o wiele większe wsparcie społeczności, co nie powinno dziwić, gdyż jest on popularniejszym oraz bardziej uniwersalnym językiem.

6. Wnioski

Aplikacje internetowe stają się coraz bardziej popularne w obecnych czasach, zapotrzebowanie na nie wciąż rośnie, a im samym stawiane są coraz wyższe wymagania. Pojawiają się coraz to nowsze technologie do tworzenia tych aplikacji. Z uwagi na to programiści są zmuszeni do trudnego wyboru – jaką technologię wybrać do zaimplementowania swojej aplikacji.

Celem niniejszej pracy było porównanie dwóch popularnych języków programowania używanych do tworzenia aplikacji internetowych: języka PHP oraz języka

Python. Do przeprowadzenia badań przygotowano aplikacje testowe, które zostały zbudowane w podobny sposób, zgodnie z założeniami. Posiadały one identyczny zestaw funkcjonalności, te same dane, oraz były uruchamiane w jednakowy sposób. W analizie porównawczej wzięto pod uwagę trzy kryteria: popularność, wydajność oraz objętość kodu.

Biorąc pod uwagę uzyskane wyniki wydajności, można stwierdzić, że język PHP wypadł o wiele lepiej od języka Python. Przy pracy nad najnowszą, aktualnie dostępną wersją PHP programiści postarali się o zwiększenie szybkości jego działania i jak można zauważyć przyniosło to oczekiwany efekt.

Pod względem objętości kodu Python, który charakteryzuje się wyjątkowym sposobem pisania kodu, potrzebował średnio o 7 linijek więcej, co przez wspomnianą wcześniej technikę „wcięć" nie powinno dziwić. Implementacja aplikacji w języku Python była o wiele łatwiejsza, jednakże składała się z większej liczby linijek kodu źródłowego.

Porównując popularność badanych języków w wyszukiwarce Google zwyciężył PHP, jednak patrząc na wsparcie społeczności oraz liczbę projektów zdecydowanie przeważa ten drugi. Python jest o wiele bardziej uniwersalnym językiem stosowanym w wielu różnych dziedzinach technologicznych, podczas gdy PHP jest używany głównie do budowania skryptów po stronie serwera internetowego.

Wyniki przeprowadzonych badań wypadają na korzyść języka PHP. Wykonana analiza jest ograniczona, gdyż dotyczyła bardzo prostej aplikacji testowej, wykonującej jedynie cztery rodzaje operacji na bazie danych – zapis, odczyt, aktualizację oraz usuwanie. Aby przeprowadzić dokładniejszą analizę należałoby poszerzyć działanie aplikacji testowych i dodać kolejne kryteria do zbadania.

Literatura

[1] K. Lei, Y. Ma, Z. Tan, Performance comparison and evaluation of web development technologies in php, python, and node.js, In 2014 IEEE 17th international conf erence on computational science and engineering, IEEE (2014) 661-668.

[2] G. A. Di Lucca, A. R. Fasolino, Testing web-based applications: The state of the art and future trends, Information and Software Technology 48 (2006) 1172-1186.

[3] K. Purer, PHP vs. Python vs. Ruby–The web scripting language shootout, Vienna University of Technology, 2009.

[4] L. Welling, L. Thomson, PHP and MySQL Web development, Sams Publishing, 2003.

[5] TIOBE Programming Community Index, https://www.tiobe.com/tiobe-index/, [14.08.2022].

[6] W3Techs – Technology Surveys, https://w3techs.com/technologies/comparison/pl-aspnet,pl-php,pl-python, [24.08.2022].

[7] M. Lutz, Programming python, "O'Reilly Media, Inc.", 2001.

[8] Faker – generator danych dla języka Python, https://faker.readthedocs.io/, [17.08.2022].

[9] Faker – generator danych dla języka PHP, https://github.com/fzaninotto/Faker/blob/master/readme.md, [17.08.2022].

[10] Hackr.io – artykuł Amana Goela, https://hackr.io/blog/python-vs-php, [18.08.2022].

[11] Linkedin – profil Amana Goela, https://www.linkedin.com/in/goel-aman/, [18.08.2022].

[12] Wyszukiwarka Google, https://www.google.com/, [16.08.2022].

[13] Stack Overflow – forum programistyczne, https://stackoverflow.com/, [16.08.2022].

[14] GitHub – serwis internetowy, https://github.com/, [16.08.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 23–28
Received: 19 September 2022
Accepted: 21 November 2022

Comparative analysis of social media accessibility

Analiza porównawcza dostępności mediów społecznościowych

Bartosz Henryk Bocheński*

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

Abstract

The paper assesses the accessibility compliance level of social media websites and their usability in terms of universal design. The study was performed on "reddit" and a social media website created by the author of the article for the express purpose of meeting the WCAG requirements as closely as possible, named "Twittn't". The analysis has been performed with the use of a survey study, an automated WAVE tool and an eye-tracker. The research groups consisted of fifteen and twenty people for survey and eye-tracker study respectively. The test results have shown strong correlation between usability of the social media website interface and number of its WCAG errors.

Keywords: social media; eye tracker; WCAG 2.0; universal design

Streszczenie

W artykule zbadano poziom dostępności mediów społecznościowych i ich funkcjonalność pod względem projektowania uniwersalnego. Badania zostały przeprowadzone na stronie „reddit" oraz medium społecznościowym stworzonym na potrzeby artykułu, spełniającym zalecenia WCAG w najwyższym możliwym stopniu, nazwanym „Twittn't". Analiza stron została przeprowadzona przy pomocą ankiety, automatycznego narzędzia WAVE oraz okulografu. Grupy badawcze zawierały 15 osób w przypadku ankiety oraz 20 w przypadku badania okulograficzengo. Wyniki badań wykazały silną korelację między użytecznością interfejsu mediów społecznościowych i liczbą błędów WCAG.

Słowa kluczowe: media społecznościowe; okulograf; WCAG 2.0; projektowanie uniwersalne

*Corresponding author

Email address: bartosz.bochenski@pollubl.edu.pl (B H Bocheński)

1. Introduction

The concept of web accessibility relates to the need to help people suffering from disabilities with accessing the Web, regardless if the disability is temporary, permanent or related to person's age. However, Web accessibility is also supposed to create a better experience for people with slow internet and those without any disabilities. Thus, an accessible website is easy to navigate, understand and interact with regardless of disability or age [1].

The Web Content Accessibility Guidelines (WCAG) Version 2.0 [2] is the internationally accepted Web accessibility compliance standard. For the purpose of this article's research, the United States and Poland both enforce the WCAG standard, with former doing so from the year 1998 with WCAG 1.0 and updating to 2.0 in year 2008, but only applying it to government owned or financed websites, and latter, as a member of the European Union, applying the standard in version 2.0 to all public sector websites since 2016 [3]. The EU (European Union) law was updated to use WCAG 2.1 in year 2018.

The main uses of social media [4] include but are not limited to: company promotions, communicating with friends or family, sharing art or random thoughts online or simply arguing with strangers over a typo. Due to their overwhelming popularity and wide variety of users, the social media websites should be designed to meet everyone's needs, including people with disabilities. However, many social media websites either follow accessibility guidelines coincidentally or completely ignore them [5], causing frustration due to unintuitive or simply poorly designed interface. Furthermore, social media are not forced to adhere to accessibility standards. In the US (United States) social media aren't usually financed or related to the federal government and thus free from the US accessibility legislation, for it doesn't apply to private websites. In the EU only public sector websites, which are websites owned by state, regional and local authorities or bodies governed by public law are required to follow accessibility guidelines. Social media do not belong to any of these categories and are thus exempt from adhering to accessibility standards.

The following study aims to evaluate how quality and clarity of social media's interface relates to its accessibility. Thus, to analyse this correlation, a UD (Universal Design) standard has been chosen (WCAG 2.0) and a hypothesis has been put forward: applying the standards of universal design greatly improve quality of a social media's interface and its usability for all users.

2. Literature review

With web accessibility being a long-standing issue for design of websites a whole there are a myriad of publications assessing accessibility standards and their application.

The article by W Arasid et al [6] analyses websites of 13 universities with the aim of improving their accessibility based on the WCAG 2.0 guidelines. The evaluation was performed with the use of TAW (Test de Accesibilidad Web), which is an automated evaluation tool. Results were presented in a form of graph showing error rate of each WCAG category. The study found that almost all of the studied websites have repeated the same WCAG 2.0 errors.

Reliance on automated tools has its disadvantages. An article by Markel Vigo et al [7] shows the consequences of relying only on automated test when identifying accessibility barriers. While automated test allows for quick evaluation, forgoing user test or expert evaluation often causes negative consequences. A total of 6 state-of-the-art tools were tested in their coverage, completeness and correctness of WCAG 2.0 conformance. The coverage averaged at most 50%, completeness reaching up to 38% and those with higher completeness averaged lower correctness, due to finding as many violations as possible, thus causing false positives. It was summarised that while using only automated test means that half of success criteria will not be tested and only 40% of the analysed criteria will be caught at the risk of generating false positives.

While the guidelines for disabled people are available, most practitioners do not conform to them, believing that accessibility guidelines provide no benefit to majority of users (unimpaired people) or cause negative impact for them. The study by Schmutz, S. et al [8] analyses the impact of implementing Web accessibility guidelines for users without impairments. Higher level of Web accessibility led to better performance compared to low or very low. There was no discernible difference between low or very low. Contrary to common concerns high conformance with Web accessibility guidelines provides benefits to unimpaired users.

3. Study subject

The study analyses user interface of two social media websites. While the websites aren't close to each other in interface design nor do they fulfil the exact same role, they are similar enough to warrant a direct comparison.

The first website under the study is hosted on https://www.reddit.com and is thus named reddit (stylized in all lower case) [9]. It is a social media website founded in 2005 by University of Virginia and is written in Python and JavaScript. Reddit has been a popular social media platform worldwide for the last several years, reaching about 430 million active users monthly current year (2022). However, it is also well known for its user interface being infamously unintuitive and not conforming to the rules of universal design (UD).

The second studied website is hosted on https://kullublin.xyz, and while it has no official name, it has been titled "Twittn't" as a reference to "Twitter", one of the more accessible social media platforms. It has been created for the needs of the study for the express purpose of fulfilling the WCAG 2.0 accessibility requirements and general rules of universal design. Twittn't was created using Wordpress software and the Buddypress plugin.

4. Study methods

The two social media websites were tested by methods provided below during the period of February to April 2022. Both websites were tested in their light mode to maintain consistency.

4.1. WAVE automatic tool

The WAVE tool [10] was used to quickly assess the WCAG compliance level of both websites. This tool presents results of its analysis of elements of a website into six categories:

- Errors
- Contrast errors
- Alerts
- Features
- Structural elements
- ARIA labels

For the purposes of this study the only noteworthy categories are errors and contrast errors, unless the other categories show an outstanding result, e.g., a website with a grand total of 0 ARIA labels (e.g., 4chan.org).

The tool was used in a form of an extension for the Mozilla Firefox browser. The evaluation was performed on live versions of both websites during the period of April 2022. For consistency's sake, both sites had enough content for scrolling to be necessary to load more and the test was done without scrolling. This way, the website content will affect the test equally on both websites.

4.2. Survey study

The survey study was performed with the assistance of fifteen people with no prior experience with either website. They were provided a list of ten task to complete in sequence on both websites:

1. Enter the provided website
2. Sign up and login
3. Create a new post
4. Upvote a post
5. Comment under a post of choice
6. Check your profile
7. Change your profile picture
8. Go to main page
9. Switch the website to dark mode
10. Log out

After which they were asked to fill out an interface assessment survey prepared by researchers at the Lublin University of Technology [1] (survey is under address https://forms.gle/7jFe1EMjPLATHxpS9). The survey consists of 31 questions divided into five categories which are as follows:

- Navigation and structure
- Communication, feedback and user assistance

- Application interface
- Subpage text
- Data input.

The results of the survey are presented on a scale from 1 to 5. With 1 meaning a function of an interface is unintuitive and difficult to perform and 5 meaning that it works without any complications and is intuitive.

4.3. Eye-tracker

The eye-tracker [11] study was performed with the assistance of twenty users with no prior experience with either of the websites. The study group contained both near sighted people and those without any vision impairments. They were asked to locate ten different interface elements present on both websites which are:

1. Element used to add an image to a post
2. Element used to enter the comment section of a post
3. The name of the group in which the post on screen has been posted
4. Dark mode button
5. Profile image change button
6. Element used to start creating a new post
7. Button used to reply to a comment
8. Button used to create a new group
9. The username of the post's author
10. Button used to leave a group

The study was performed on a computer equipped with a Gazepoint GP-3 HD eye-tracker. The result categories analysed by the study are as follows:

- task completion time
- Time To First Fixation (TTFF) [12] on an Area of Interest (AOI)
- success rate of a task.

5. Results

5.1. WAVE automatic tool

The following figures (Figure 1, 2) show the WAVE tool result summaries for reddit (Figure 1) and Twittn't (Figure 2).

The following table (Table 1) describes the number of WCAG 2.0 error occurrences per category on both websites. Table 1 contains only the WCAG categories in which at least one of the websites had at least one error.

Table 1 Comparison of number of WCAG 2.0 errors for WAVE test

WCAG 2.0 Category	Occurrences on reddit	Occurrences on Twittn't
1.1 Text alternatives	42	8
1.2.1 Audio/Visual only	11	0
1.2.2 Captions pre-recorded	10	0
1.2.3 Audio description or media alternative pre-recorded	10	0
1.3.1 Info and Relationships	2	1
1.4.2 Audio control	11	0
1.4.3 Contrast minimum	147	1
2.1.1 Keyboard	1	1
2.4.1 Bypass blocks	1	1
2.4.4 Link purpose (in context)	59	7
2.4.6 Headings and labels	2	1
3.3.2 On input	1	0
4.1.2 Name, role, value	1	0
Total number	**298**	**20**

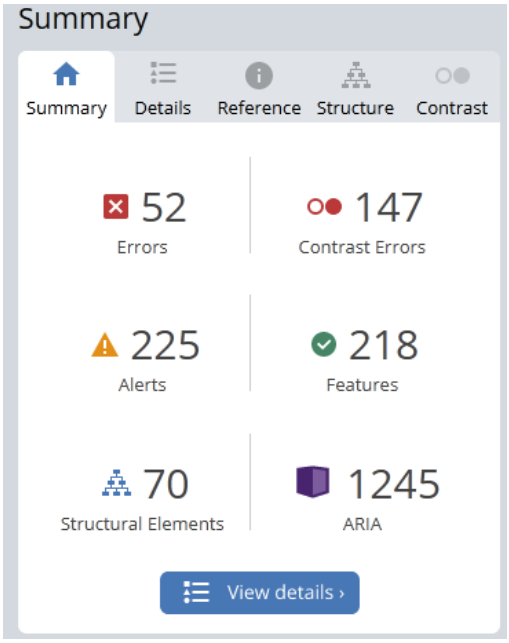


Figure 1: Summary of WAVE tool results for reddit.

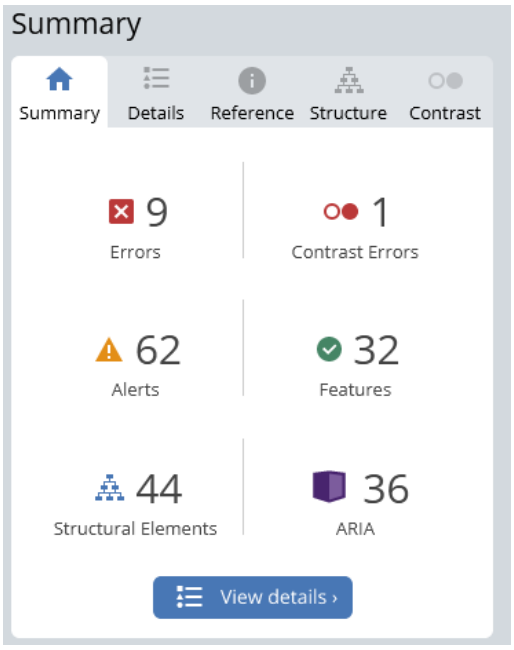


Figure 2: Summary of WAVE tool results for Twittn't.

The totals error counts between the summaries (Figure 1, 2) do not match the total amounts in the

table (Table 1) due to summaries only showing the number of elements with errors and not the total error count. For example, an empty button is a singular element but the error it causes falls under WCAG 2.0 category 1.1 and 2.4.4 thus causing 2 errors.

5.2. Survey study

The interface assessment survey has resulted in a total average of 2 points for reddit and 4.21 for Twittn't. The averages for each question of the survey are presented in Figure 3.

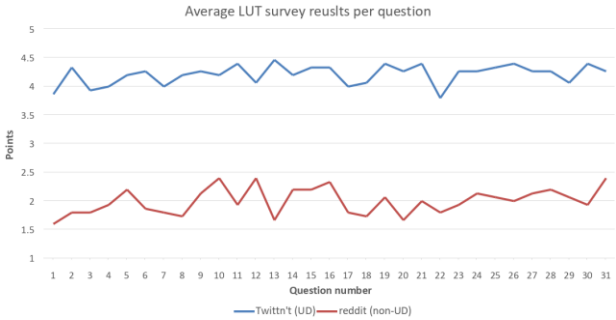


Figure 3: Average survey results on a scale from 1 to 5 for both websites.

The average results of each category are presented in Table 2.

Table 2 Average survey results for each category of the website assessment survey

Category	reddit	Twittn't
Navigation and structure	1.87	4.1
Communication, feedback and user assistance	2.07	4.19
Application interface	1.96	4.26
Sub page text	1.99	4.12
Data input	2.12	4.27

5.3. Eye-tracker study

The results of eye-tracker study have been compiled into tables task success rates and the averages for both TTFF and task completion time. Table 3 shows study results for reddit and the Table 4 displays the results for Twittn't. Shorter task completion times and TTFF and higher success rates indicate better quality of the interface.

The distinction between near sighted people and the unimpaired wasn't made due to vision impairment not causing any significant difference in overall results.

To visually represent the locations of users' fixations a set of heat maps [13] has been generated. The heat maps shown in Figure 4 are for reddit's stimulus number 2, the task of which was to locate the element or elements responsible for redirecting the user to post's comment section.The second heat map (Figure 5) shows the fifth stimulus for Twittn't, the task of which was to locate element or elements used to change user's profile photo.

Table 3: Eye-tracker data table for reddit containing average task completion times, average TTFF and task success rates

Stimulus	Task completion time	TTFF	Task success rate
1	12677.37	8184.44	35%
2	8670.37	4061.44, 6745.42	95%
3	11750.51	3673.82	85%
4	9395.05	7857.48	85%
5	6457.87	2911.54	80%
6	7812.689	2289.511	90%
7	6506.373	3792.282	55%
8	14198.73	9861.914	75%
9	6198.946	3646.215	75%
10	14549.77	5199.44	95%

Table 4: Eye-tracker data table for Twittn't containing average task completion times, average TTFF and task success rates

Stimulus	Task completion time	TTFF	Task success rate
1	5348.40	3417.50	70%
2	7046.20	3153.80	100%
3	4020.80	5295	65%
4	4278	3221.20	75%
5	5916	486.40, 4957.10	95%
6	2989	2420.90	100%
7	2781.70	2039.20	85%
8	6765.60	3426	65%
9	2212.40	1753.60	95%
10	2560.50	1701.90	95%

Figure 4: Heat map of stimulus 2 for reddit.

Figure 5: Heat map of stimulus 5 for Twittn't.

6. Discussion

For the automated WAVE test the total number of errors on reddit is quite high in comparison to Twittn't, 1490% of the Twittn't total error count (Table 1). The difference couldn't have come from user generated content due to less than 10 posts being loaded on each of the websites at the time of the tests. Social media websites tend to struggle with text alternatives and link purpose categories due to most of the content being generated by the user on the front-end side of website. However, the number of occurrences of these errors on reddit exceeds double the number of loaded posts, meaning that the errors, especially empty links, must originate from parts of the website interface. Surprisingly enough, part ofthe image-based posts did include proper alternative text. The only real outlier on the side of Twittn't is text alternatives and link purposes. The errors in these categories originate from user profile pictures not having their alternative text handled properly when they appear as a part of a post. This issue is slightly mitigated by profile picture never appearing without the username beside it. Those errors, while notable, will not affect the majority of users.

The contrast issues on the other hand will affect all users. Those errors on reddit originated from text colour being light grey while the background is either slightly lighter grey or white. The user error or malice doesn't contribute to the issue due to reddit not allowing the change of text colour on posts and WAVE tool not counting images for contrast category. For comparison, the reddit's text to background contrast ratio is 4.21:1 while the Twittn't is 8.59:1. In theory this means that reddit fails WCAG's AA requirement for normal text and barely passes it for large text, while the Twittn't passes the contrast test. In practice, this means that users might have issues reading the labels of interface elements or information such as post author's name or post's title on reddit and visually impaired users might be incapable of reading them.

The survey study has shown that the survey respondents, after completion of the tasks, were highly biased against reddit. The average difference between reddit and Twittn't was above 2 points on a 5-point scale (Figure 3).

Important detail to mention before the analysis of results is that the areas of interest set in the application used to collect eye-tracking data were highly lenient, i.e., the area of interest was about 20% larger than the element itself for all stimuli.

The task completion rates for the eye-tracker study only account for any fixation happening on a stimulus. This decision was made to reduce the risk of misinterpreting the reason for time difference between last fixation and the task completion.

The time difference between TTFF and task completion times is quite long for reddit, with task completion times being up to 240% longer than TTFF. Such difference indicates that the users had issues discerning the purpose of the element they were looking at. The time differences vary from 4 to 8 seconds (Table 3). The discrepancy was made clear with stimuli 1, 8 and 10. On those stimuli the users averaged over 30 total fixations per stimuli, while averaging 0.35, 4 and 3 fixations on the areas of interest (AOI) for stimulus 1, 8 and 10. An edge case happened in stimulus 4 where one respondent took slightly over 16 seconds and 100 (5 times the average) fixations to find the correct AOI. Furthermore, the subsequent fixations on AOI usually appeared early after the first fixation. Last fixations on the AOI usually happened over a second before the task completion time.

The eye-tracker results for Twittn't (Table 4) were far more stable, with times between TTFF and task completion time ranging between 500 and 3000 ms. The stimuli with longer delays between those times usually had an average of 2.2 subsequent fixations close to the task completion time. Additionally, the last fixation on the AOI usually occurred less than 500 ms before task completion time.

The notable outlier on the Twittn't eye-tracker data is stimulus 3 (Table 4), where the average TTFF ended up being higher than task completion time by over a second. The cause of this is a couple of respondents that decided they found the correct answer less than a second after the start of the stimulus and either missing the element or the eye-tracker not recording the fixation due to it being too short.

Due to high difference in eye-tracker study results between analysed websites and substantial difference in task completion times, especially for reddit, a Levene's test [14] was performed on the eye-tracker results. The test results for all of the stimuli were above 0.05, thus all of the eye-tracker data met the assumption of homogeneity of variance, with the highest result being stimulus 6 with p value of 0.915 and lowest being stimulus 9 with p value of 0.117.

On a side note, many of important interface elements of reddit are placed on a long dropdown menu on the top right side of a screen. The element that opens the menu has severe contrast issues. The notable elements hidden by the menu are: user profile options, community creation menu and dark mode. The unintuitive nature of the menu could have affected the results of the survey and eye-tracker study.

7. Conclusions

In summary, results of the study have confirmed the hypothesis put forward at the beginning of the article.

The automatic WAVE test revealed the first concerning detail about reddit: while total number of errors is quite high (298), there are websites that function properly with higher error counts. The real problem is that the half of them are contrast errors that make reddit difficult to navigate at best and painful for the eyes at worst. Twittn't on the other hand has low number of total errors (20) but only a singular contrast error, making it significantly easier to navigate and to read the website's content.

The survey has confirmed the predicted issues with interface due to contrast. The survey results per category (Table 2), in reddit's, case is the lowest for navigation and structure (1.87) and application interface (1.96), while on Twittn't the application interface is the second highest rated category at 4.26.

The eye-tracker study has shown that the users had significantly easier time locating element on Twittn't than on reddit. The task completion times being shorter on Twittn't with higher success rates among users shows a clear difference in interface clarity. Times to first fixation support that: on reddit the TTFF tends to be upwards of 8 seconds before task completion time. Either the user doubted that the element they found fulfilled the ascribed purpose or deemed the element to not be the correct one and continued their search. On the other hand, the TTFFs on Twittn't were closer to task completion times. A singular exception being a stimulus with two AOIs (Table 4, stimulus 5), meaning that unlike on reddit the users were not second guessing themselves once they located the correct element.

In conclusion simply following the rules of universal design, such as WCAG 2.0, greatly improves the quality and clarity of social media website interface.

References

[1] M. Miłosz, Ergonomia systemów informatycznych, Biblioteka Cyfrowa Politechniki Lubelskiej, 2014.

[2] W3C Recommendation, Web Content Accessibility Guidelines, 11 December, http://www.w3.org/TR/WCAG20/, [20.07.2022].

[3] Web Accessibility Initiative, Web Accessibility Laws & Policies, https://www.w3.org/WAI/policies/, [20.07.2022].

[4] A.M. Kaplan, M. Haenlein, Users of the world, unite! The challenges and opportunities of Social Media, Business Horizons, 53(1) (2010) 59-68, https://doi.org/10.1016/j.bushor.2009.09.003.

[5] M.G. Dinis, C. Eusébio, and Z. Breda, Assessing social media accessibility: the case of the Rock in Rio Lisboa music festival, International Journal of Event and Festival Management, 11(1) (2020) 26-46, https://doi.org/10.1108/IJEFM-02-2019-0012.

[6] W. Arasid, A.G. Abdullah, D. Wahyudin, C.U. Abdullah, I. Widiaty, D. Zakaria, N. Amelia, A Juhana. An Analysis of Website Accessibility in Higher Education in Indonesia Based on WCAG 2.0 Guidelines, IOP Conference Series: Materials Science and Engineering, 306(1) (2018) 012130, https://dx.doi.org/10.1088/1757-899X/306/1/012130.

[7] M. Vigo, J. Brown, V. Conway. Benchmarking web accessibility evaluation tools: measuring the harm of sole reliance on automated tests, In Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13). Association for Computing Machinery, New York, NY, USA, 1 (2013) 1–10, https://doi.org/10.1145/2461121.2461124.

[8] S. Schmutz, A. Sonderegger, & J. Sauer. Implementing Recommendations From Web Accessibility Guidelines: Would They Also Provide Benefits to Nondisabled Users, Human Factors, 58(4) (2016) 611–629, https://doi.org/10.1177/0018720816640962.

[9] K.E. Anderson, Ask me anything: what is Reddit?, Library Hi Tech News, 32(5) (2015) 8-11, https://doi.org/10.1108/LHTN-03-2015-0018.

[10] L.R. Kasday. A tool to evaluate universal Web accessibility. In Proceedings on the 2000 conference on Universal Usability (CUU '00), Association for Computing Machinery, New York, NY, USA (2000) 161–162, https://doi.org/10.1145/355460.355559.

[11] K. Holmqvist, M. Nyström, and F. Mulvey. Eye tracker data quality: what it is and how to measure it. In Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12), Association for Computing Machinery, New York, NY, USA (2012) 45–52, https://doi.org/10.1145/2168556.2168563.

[12] L.N. van der Laan, I.T.C. Hooge, D.T.D. de Ridder, M.A. Viergever, P.A.M. Smeets, Do you like what you see? The role of first fixation and total fixation duration in consumer choice, Food Quality and Preference, 39 (2015) 46-55, https://doi.org/10.1016/j.foodqual.2014.06.015.

[13] O. Špakov and D. Miniotas 2007. Visualization of Eye Gaze Data using Heat Maps, Elektronika ir Elektrotechnika, 74(2) (2007) 55-58.

[14] G.V. Glass. Testing Homogeneity of Variances, American Educational Research Journal, 3(3) (1966) 187–190, https://doi.org/10.3102/00028312003003187.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 29–32
Received: 19 September 2022
Accepted: 7 November 2022

Comparative analysis of transfer protocols asynchronous messages on systems queuing

Analiza porównawcza protokołów przesyłania wiadomości asynchronicznych w systemach kolejkowych

Grzegorz Derlatka*, Piotr Kopniak

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

Abstract
This article presents an analysis of the performance of two protocols supported by queuing systems, i.e. MQTT (MQ Telemetry Transport) and AMQP (Advanced Message Queuing Protocol). This analysis was performed using two message brokers - ActiveMQ and RabbitMQ. The time of sending the message was analyzed, determined on the basis of the time of sending and receiving the message for both protocols in both of the above-mentioned queuing systems. The tests were carried out using proprietary applications written in Java and the Spring application framework.

Keywords: message broker; asynchronous communication; AMQP protocol; MQTT protocol

Streszczenie
W tym artykule została przedstawiona analiza wydajności dwóch protokołów obsługiwanych przez systemy kolejkowe, tj. protokołu MQTT (ang. MQ Telemetry Transport) oraz AMQP (ang. Advanced Message Queuing Protocol). Analiza ta została przeprowadzona z użyciem dwóch brokerów wiadomości - ActiveMQ oraz RabbitMQ. Analizie został poddany czas przesłania wiadomości wyznaczony na podstawie czasu wysłania i odebrania komunikatu dla obu protokołów w obu przytoczonych systemach kolejkowych. Testy zostały przeprowadzone przy pomocy własnych aplikacji napisanych w języku Java oraz szkielecie aplikacji Spring.

Słowa kluczowe: system kolejkowy; broker komunikatów; komunikacja asynchroniczna; protokół AMQP; protokół MQTT

*Corresponding author
Email address: **grzegorz.derlatka@pollub.edu.pl** (G. Derlatka)

1. Wstęp

System kolejkowy to oprogramowanie, które umożliwia aplikacjom, systemom i usługom wzajemną komunikację oraz wymianę informacji. Broker komunikatów, bo również pod taką nazwą funkcjonuje, realizuje taką funkcję dzięki tłumaczeniu komunikatów pomiędzy formalnymi protokołami komunikacyjnymi. Dzięki temu usługi współzależne mogą ze sobą bezpośrednio „rozmawiać", nawet jeżeli zostały napisane w różnych językach lub wdrożone na różnych platformach [1].

System kolejkowy opiera się na komunikacji asynchronicznej. Polega ona na wysyłaniu danych przez nadawcę wiadomości bez konieczności przesłania informacji zwrotnej, co oznacza, że po drugiej stronie nie musi być odbiorcy w tym samym czasie. Taki rodzaj komunikacji nie wymaga współistnienia rozmawiających ze sobą procesów (bytów). Takie podejście sprawia, że poszczególne moduły, komponenty programu są luźniej powiązane, przez co podmiana ich jest łatwiejsza, bezpieczniejsza, a także awaria, któregoś z nich nie powoduje tylu błędów co w przypadku komunikacji synchronicznej.

Analizie zostały poddane dwa protokoły wykorzystywane w brokerach wiadomości – MQTT oraz AMQP.
Protokół MQTT jest lekkim protokołem transmisji danych, opartym o wzorzec publikacji /subskrypcji, pracujący na szczycie warstwy TCP (ang. Transmission Control Protocol) / IP (ang. Internet Protocol). Znajduje on zastosowanie głównie w systemach związanych z Internetem rzeczy. Natomiast protokół AMQP to protokół o otwartym standardzie, który umożliwia współdziałanie poprzez przesyłanie wiadomości między systemami, niezależnie od dostawcy lub używanej platformy brokera wiadomości. Dzięki standardowi AMQP do wymiany wiadomości można używać dowolnej biblioteki klienta zgodnej z AMQP oraz dowolnego brokera zgodnego z AMQP. Klienci wiadomości korzystający z AMQP są całkowicie niewiążący.

Motywacją do wykonania takiej analizy było bardzo częste oraz stale rosnące zastosowanie brokerów wiadomości w systemach komercyjnych.

W przedstawionej pracy zostały porównane protokoły systemów kolejkowych, ponieważ w dostępnej literaturze głównie porównywane są systemy kolejkowe same w sobie.

2. Przegląd literatury

W pierwszej pracy autorstwa J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate oraz P. Manzoni [2] zbadano protokoły AMQP i MQTT w takich przypadkach jak utrata wiadomości, opóźnienie, odchylenie od ustalonych, okresowych charakterystyk sygnału. Analiza została przeprowadzona przy użyciu aplikacji testowej, która nazywa się Amqperf, została opracowana w celu wygenerowania obciążenia dla systemu kolejkowania wiadomości u producenta. Amqperf korzysta z brokera wiadomości RabbitMQ, która jest implementacją protokołu AMQP oraz biblioteki Paho, która zawiera implementację protokołu MQTT. Badania wykazały, że użycie protokołu AMQP jest lepsze w przypadku budowania niezawodnych, skalowalnych i zaawansowanych infrastruktura przesyłania wiadomości w klastrze przez sieć WLAN.

W następnej pracy autorstwa N. Q. Uy oraz V. H. Nam [3] badano skuteczność użycia protokołów MQTT i AMQP w oparciu o eksperymenty, symulacje środowiska transmisji danych z różnym opóźnieniem i współczynnikiem utraty pakietów. Badaniu zostały poddane czasy, w którym pakiety poruszały się w środowisku z opóźnieniem oraz wskaźnikiem strat dla obu protokołów. Do przeprowadzenia eksperymentu zostały użyte dwa komputery z aplikacjami napisanymi w języku Python oraz bazą danych MySQL do zapisywania wyników wysyłania i odbierania wiadomości. Z przeprowadzonych badań wynika, że protokół MQTT przynosi pewne korzyści w przypadku bardzo nisko zasilających urządzeń, jest bardzo skuteczny jeśli skupiamy się głównie na wydajności, wymaga mniej zasobów. AMQP to bardziej zaawansowany protokół, bardziej niezawodny i zapewniający lepsze bezpieczeństwo.

W kolejnym badaniu autorstwa Mishra B., Mishra B. i Kertesz A. [4] poddano ocenie wydajność kilku implementacji brokerów dla protokołu MQTT przy użyciu testów obciążeniowych oraz analizy ich powiązań z projektem systemu. Analiza wyników testu odbywała się za pomocą trzech różnych metryk: obciążenie procesora, opóźnienie i szybkość wiadomości. Badania były prowadzone przy użyciu lokalnego komputera oraz Google Cloud Platform. Wyniki wykazały, że Mosquitto przewyższa inne rozważane rozwiązania w większości metryk. Jednak ActiveMQ jest najbardziej wydajny pod względem skalowalności dzięki wielowątkowej implementacji, podczas gdy Bevywise ma obiecujące wyniki dla scenariuszy z ograniczonymi zasobami.

Ostatnia praca autorstwa Kaciuczyk T., Korga T. oraz Smołka J. [5] dotyczyła badań wydajności wybranych brokerów komunikatów: Apache ActiveMQ, RabbitMQ oraz Apache Kafka. Analizie został poddany czas przesłania wiadomości wyznaczony na podstawie czasu wysłania i odebrania komunikatu. Testy zostały przeprowadzone za pomocą autorskich aplikacji klienckich napisanych w języku Java. Badania wykazały, że najbardziej wydajną kolejką wiadomości jest Apache Kafka.

Na podstawie przytoczonej literatury można stwierdzić, że podobne badania zostały już przeprowadzone. Jednak w tej pracy badane są głównie protokoły, nie systemy kolejkowe. Analizują one wydajność, tj. czas odbierania i wysyłania wiadomości dwóch protokołów wiadomości - AMQP oraz MQTT w dwóch systemach kolejkowych ActiveMQ i RabbitMQ.

Ponadto, na podstawie przeprowadzonej analizy literatury postawiono tezę: „Protokół MQTT jest wydajniejszy, tj. czas przesyłania wiadomości jest krótszy”

3. Aplikacje opracowane na potrzeby badań

Aplikacje do pomiarów czasów zostały napisane w języku Java oraz szkielecie aplikacji Spring. Programy rejestrują czasy wysłania i odebrania wiadomości oraz wyświetlają wyniki w konsoli. Takie testy zostały wykonane dla protokołu AMQP oraz MQTT przy wykorzystaniu systemów kolejkowych – ActiveMQ oraz RabbitMQ. Stąd, zostały stworzone odpowiednio aplikacje dla odbiorcy i wydawcy wiadomości w różnych modelach kolejek. Przesyłane wiadomości były w różnej liczbie jednocześnie. Na podstawie uzyskanych danych zostały wyznaczone średnie czasy przesyłu. Badania zostały przeprowadzone na komputerze o parametrach:

- procesor Intel Core i5-7200U CPU@2.5Ghz,
- 8GB pamięci RAM, dysk twardy typu SSD o pojemności 256 GB,
- system operacyjny Ubuntu 18.04.

4. Metoda badawcza

Średnie czasy wysyłania wiadomości zostały obliczone na podstawie 10 otrzymanych wyników dla różnych konfiguracji, aby ograniczyć ryzyko błędnych danych przez opóźnienia komputera. Testowano czas dla 1, 10, 1000 oraz 10000 wiadomości jednocześnie. Badania prowadzono dla wiadomości trwałych, czyli takich, które są przechowywane w systemie kolejkowym. Dla komunikatów powyżej jednego mierzono czas od momentu wysłania pierwszej wiadomości do momentu odebrania ostatniej wiadomości. Przesyłana wiadomość była stałej wielkości i wynosiła 10 kB.

Tabela 1: Scenariusz nr 1

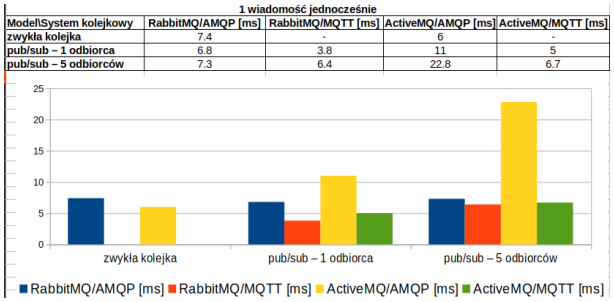
Nazwa: Badanie czasu wysłania wiadomości		
Cel badań: Ustalenie czasów wysyłania wiadomości dla różnych wielkości		
Warunki początkowe: Działające aplikacje, uruchomione systemy kolejkowe		
Warunki końcowe: Dane dot. czasów wysłania wiadomości zostaną wyświetlone oraz zapisane		
Liczba użytkowników biorących udział w badaniu: 1		
Kolejne czynności		
Lp.	**Opis czynności**	**Oczekiwany wynik**
1	Użytkownik uruchamia aplikację oraz system kolejkowy	W aplikacji zostają wyświetlone informacje o jej uruchomieniu oraz poprawnym połączeniu z systemem kolejkowym
2	Aplikacja zaczyna wysyłać wiadomości	W aplikacji zostają wyświetlone czasy wysłania wiadomości

Tabela 2: Scenariusz nr 2

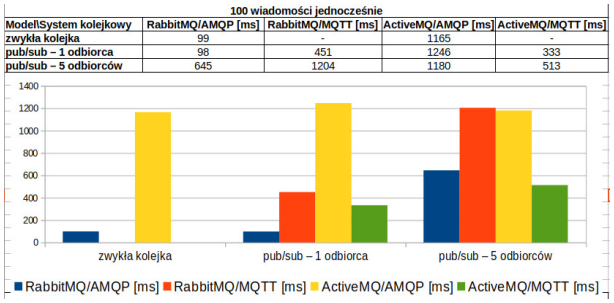
Nazwa: Badanie czasu odbierania wiadomości		
Cel badań: Ustalenie czasów odbierania wiadomości dla różnych wielkości		
Warunki początkowe: Dane dot. czasów odbierania wiadomości zostaną wyświetlone oraz zapisane		
Warunki końcowe: Dane dot. czasów wysłania wiadomości zostaną wyświetlone oraz zapisane		
Liczba użytkowników biorących udział w badaniu: 1		
Kolejne czynności		
Lp.	**Opis czynności**	**Oczekiwany wynik**
1	Użytkownik uruchamia aplikacje oraz system kolejkowy	W aplikacji zostają wyświetlone informacje o jej uruchomieniu oraz poprawnym połączeniu z systemem kolejkowym
2	Aplikacja, po jej uruchomieniu, zaczyna wysyłać wiadomości	W aplikacji nr 1 zostają wyświetlone daty wysłania wiadomości
3	Aplikacja nr 2 odbiera wiadomości wysłane przez aplikację nr 1	W aplikacji nr 2 zostają wyświetlone informacje o odebraniu wiadomości oraz daty odebrania

5. Wyniki

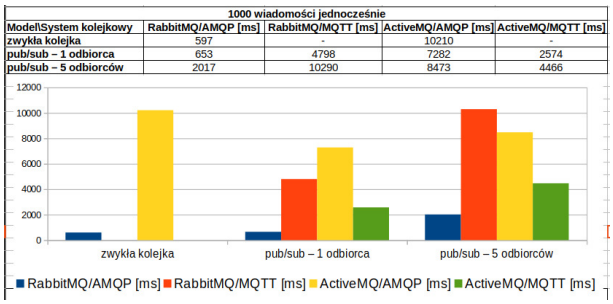
Pierwszym badaniom zostało poddane wysyłanie pojedynczych wiadomości w modelu zwykłej kolejki oraz w modelu publikacji i subskrypcji. W modelu publikacji i subskrypcji mierzono czasy dla jednego oraz pięciu subskrybentów. Natomiast z racji tego, że protokół MQTT obsługuje tylko model publikacji
i subskrypcji, w badaniach nie uwzględniono zwykłej kolejki. Wynik badania przedstawia Rys. 1. Pozostałe badania wykonano analogicznie do poprzedniego, z tym, że wysyłano różne ilości wiadomości jednocześnie. Wyniki przedstawiają kolejno Rys. 2, Rys. 3 oraz Rys. 4.



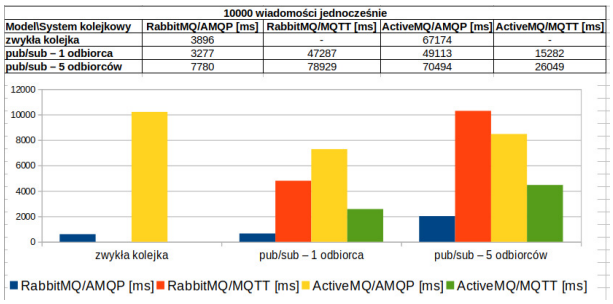
Rysunek 1: Czas przesłania pojedynczej wiadomości dla przedstawionych modeli, protokołów oraz kolejek.



Rysunek 2: Czas przesłania 100 wiadomości dla przedstawionych modeli, protokołów oraz kolejek.



Rysunek 3: Czas przesłania 1000 wiadomości dla przedstawionych modeli, protokołów oraz kolejek.



Rysunek 4: Czas przesłania 10000 wiadomości dla przedstawionych modeli, protokołów oraz kolejek.

6. Dyskusja

W modelu zwykłej kolejki przeprowadzone badania zostały z wykorzystaniem tylko protokołu AMQP. Zmierzone czasy wykazały, że lepiej poradził sobie system RabbitMQ. Dla pojedynczej wiadomości nie była to zauważalna różnica, gdyż wyniki różniły się
o zaledwie 1.4 ms. Natomiast dla większej ilości komunikatów czasy różniły się o rzędy wielkości.

W modelu publikacji i subskrypcji można zauważyć, że RabbitMQ przy użyciu protokołu AMQP radzi sobie najlepiej, zarówno dla jednego odbiorcy jak i pięciu. Najgorzej w tym zestawieniu wypada message broker ActiveMQ przy użyciu protokołu AMQP.

Analizując dane pod względem tylko protokołów można stwierdzić, że generalnie lepsze czasy tj. krótsze osiąga protokół MQTT.

Czas przesyłania pojedynczej wiadomości drastycznie spada w przypadku wysyłania wielu wiadomości. Może to być spowodowane długim czasem łączenia się aplikacji z systemem kolejkowym czy też optymalizacjami message brokera podczas wysyłania tych samych wiadomości.

Porównując badania z przeglądem literatury można stwierdzić, że przeprowadzona analiza prowadzi do podobnych wniosków. Przykładowo, badania autorstwa N. Q. Uy oraz V. H. Nam [3] wykazały również, że protokół MQTT jest wydajniejszy. Zatem, postawiona teza: ”Protokół MQTT jest wydajniejszy, tj. czas przesyłania wiadomości jest krótszy” została udowodniona i potwierdzona.

7. Podsumowanie

W powyżej pracy porównano czas przesyłania wiadomości dla dwóch protokołów używanych
w systemach kolejkowych - AMQP oraz MQTT. Analiza została przeprowadzona przy użyciu dwóch brokerów wiadomości - RabbitMQ oraz ActiveMQ. Analiza wykazała, że wydajniejszym (osiągającym krótsze czasy przesyłania wiadomości) systemem jest RabbitMQ.
W wyniku porównania bezpośrednio protokołów wykazano, że wydajniejszym protokołem jest MQTT. Stwierdzono to poprzez obliczenie średnich czasów nie biorąc pod uwagę brokera wiadomości, a sam protokół.

Literatura

[1] Zasada działania, cechy systemu kolejkowego, https://www.ibm.com/cloud/learn/message-brokers, [12.09.2022].

[2] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, P. Manzoni, A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks, 12th Annual IEEE Consumer Communications and Networking Conference 12 (2015) 931-936.

[3] N. Q. Uy, V. H. Nam, A comparison of AMQP and MQTT protocols for Internet of Things, 6th NAFOSTED Conference on Information and Computer Science (2019) 292-297, https://doi.org/10.1109/NICS48868.2019.9023812.

[4] B. Mishra, B. Mishra, A. Kertesz, Stress testing mqtt brokers: A comparative analysis of performance measurements, Energies 14 (2021) 5817-5837.

[5] T. Kaciuczyk, T. Korga, J. Smołka, Functional and performance analysis of selected message brokers in a distributed application, Journal of Computer Sciences Institute 14 (2020) 19-25.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 33–41
Received: 19 September 2022
Accepted: 17 October 2022

Usability analysis of advertising websites interfaces with the use of the universal design principles

Analiza użyteczności interfejsów serwisów ogłoszeniowych z uwzględnieniem zasad projektowania uniwersalnego

Jan Marciniec*, Dominik Kondraciuk

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

Abstract

The aim of the research was to observe how the usage of the universal design principles influences on the accessibility and usability of websites. As one of the analysis' subjects there was chosen the popular advertising website. It has been confronted with a newly implemented application offering the same functions, but improved with the requirements resulting from the universal design. In order to compare those two services there was conducted the study with the use of eye-tracking, LUT checklists and WAVE Evaluation Tool. These methods made it possible to demonstrate the differences between websites caused by the contrasting implementation methods. The results showed that an application of universal design principles has a positive impact on the websites' accessibility and usability.

Keywords: universal design; usability; accessibility; user experience

Streszczenie

Celem badań było zaobserwowanie w jaki sposób stosowanie zasad projektowania uniwersalnego wpływa na dostępność i użyteczność stron internetowych. Jako jeden z przedmiotów analizy wybrano popularny serwis ogłoszeniowy. Skonfrontowano go z nowo zaimplementowaną stroną oferującą te same funkcje, lecz udoskonaloną o wymagania wynikające z zasad projektowania uniwersalnego. W celu porównania obu serwisów przeprowadzono eksperymenty okulograficzne oraz badania jakości interfejsów z wykorzystaniem list kontrolnych LUT i walidatora dostępności cyfrowej WAVE. Metody te umożliwiły wykazanie różnic pomiędzy aplikacjami wynikających z innych metod implementacji. Wyniki pokazały, że stosowanie zasad projektowania uniwersalnego ma pozytywny wpływ na dostępność i użyteczność stron internetowych.

Słowa kluczowe: projektowanie uniwersalne; użyteczność; dostępność; doświadczenia użytkownika

*Corresponding author

Email address: **jan.marciniec@pollub.edu.pl** (J. Marciniec)

1. Introduction

According to the report [1], about 15% of all the people that live in the world, have some disabilities. It is really important to take actions that will make their lives easier, so they will not be excluded from society. Accessibility term means, that people with disability have exactly the same access to all of the social life's areas [2]. Nowadays, a lot of activities require using the Internet and when applications are designed for most typical users, people with disability might have some troubles using them [3]. For this reason, it is so important to implement websites using the universal design, which aims to increase accessibility. This technique does not only allow users with disability use websites, but it also makes it easier for all people [4]. The universal design makes the implementation of websites more difficult, but it makes them way more useful, so it is worth making the additional effort. Over the years, websites' interfaces have evolved from an ergonomic point of view. Recently, there is increasingly noticed the need to adapt applications for people with natural computer difficulties. All public websites implemented after 2015 should be adapted to the needs of people with disability. Despite this recommendation, many of them are designed in a way that makes it difficult for some users to access the whole content.

There are not so many papers about insightful research related to the topic of universal design influence. There can be found articles about the analysis of interfaces usability, but there is lack of papers referring to the comparison of websites with similar functionalities, but implemented with various design approaches.

The main purpose of this paper is to conduct a series of comparative experiments between two similar applications to observe how the universal design techniques affect an accessibility and usability of the website.

There has been specified the research designing a web application using the universal design rules increases its usability.

Some research hypotheses have been specified and they are as follows:

- Designing a web application using the universal design rules increases the simplicity and intuitiveness of its use.
- Designing a web application using the universal design rules increases its availability for users.

- The possibility of adjusting the contrast and the font size has a positive effect on the effectiveness of using the website.

2. Literature review

Article [5] contains a set of the eye-tracking patterns related to potential interface usability problems. In order to verify if these assumptions are correct, authors conducted the eye-tracking research, during which participants had to perform specific tasks with the use of two websites. It has been observed that there is a correlation between some patterns and the corresponding usability problems, so during interpretation of the eye-tracking results it is worth paying attention to the occurrence of the presented patterns.

The research about the websites' usability with the use of the eye-tracking technology has been also described in [6]. The aim of this article was to observe some typical users' behaviors during finding information at websites. The results have shown that the most users' attention is attracted by informational areas of the page.

In [7] there has been described the eye-tracking research with the use of five popular Norwegian websites. The results have shown that the intuitiveness of the interface affects the users' experience. Unfortunately, the research did not include guidelines of the universal design paradigm and the WCAG (Web Content Accessibility Guidelines) 2.0, so there cannot be made conclusions about the influence of following the recommendations relating to the application's design.

Paper [8] refers to the websites' accessibility in terms of meeting the WCAG 2.0 and 2.1 rules. To make the research, authors used various accessibility validators such as Achecker, SortSite and WAVE Evaluation Tool. They were used to verify 54 official websites for vaccination registration in USA. The results have shown the scale of problems with websites' accessibility, because all of examined applications are supported by the government and only two of them had been implemented in accordance with all the WCAG guidelines.

The topic of public websites is also included in the [9], which contains the description of the websites' accessibility research conducted with the use of Tenon and VoiceOver WCAG 2.0 validators. The results have shown that only one website was properly designed for people with disability. Authors also presented the most frequent usability problems which concern the structure of the document, the lack of correct text alternatives, insufficient information about the intended use of the links, unmodifiable font size and the lack of attributes indicating the purpose of the elements.

Article [10] is about the development process of e-learning platform. Three methods were applied to identify potential usability problems: heuristic evaluation, user experience questionnaires and the eye-tracking. After revealing some errors, all of them were corrected and then the level of platform's availability was re-verified. The results allowed to state that the usability of the modernized application is at the higher level compared to the original version.

Paper [11] presents research on a procedure for improving a GUI quality. The procedure was applied to the Sale Force Automation application. The authors presented several methods that can be used to evaluate the application. These include expert analysis and cognitive walkthrough. In addition, the authors proposed a new method that is a conglomerate of expert analysis and cognitive walkthrough – LUT checklists. The proposed method proved to be accurate for this type of application interface evaluation, but the authors believe that its versatility makes it applicable to different types of websites. The method helped the experts to carry out the evaluation and made it more structured and consistent.

In the [12] the author described a number of methods for evaluating the quality of software interfaces. Based on the way the evaluation was carried out, he divided them into manual and automatic ones. Among the manual testing methods, the author listed and described, among others, eyetracking. Various methods of presenting test results obtained by eyetracking are described, such as fixation maps, heat maps, point-of-view videos, but it is possible to present and analyze in even more sophisticated ways. Another described manual method was to collect user feedback. The author claimed that with just 5 people, 80% of problems can be detected. In order to properly carry out user feedback collection, it is worth using supporting tools, such as Nielsen-Molich heuristics or LUT checklists based on them. The author described LUT checklists and the metric developed based on them – WUP (Web Usability Points), and showed how they work and how to use them correctly to improve the interface evaluation.

3. Materials and methods

The object of research consists of two websites. One of them is a popular advertising service – Gumtree. It allows users to advertise items they want to sell and then other users can view those announcements. Looking at the number of advertisements, you can see that a lot of people use this website, so it should be accessible for all of them. However, it is not properly adapted in terms of the universal design. A lot of service's views are unintuitive, because of the elements' arrangement. At some parts of the website, it can be seen that the contrast level is too low and additionally it cannot be adjusted to the user's preferences or needs. Another drawback is that the font size is also unchangeable. Content of this website is only available in one language – Polish, so it cannot be understood by people speaking other languages. Blind people or anyone who will try to use a screen reader for this website will also have problems with access to the content, because there are lacks of ARIA attributes in some important places. There are also images, links and buttons that do not have alternative texts, which would be really helpful while using an assistive technology.

To observe, how those issues affect the website's availability, there has been implemented the similar

application to the Gumtree, but with some improvements. First, all the interfaces have been changed to increase the clarity of the whole website. All the views have been implemented using appropriate ARIA attributes and various alternative texts, so technologies like screen readers can be easily used with this service. Accessible websites should offer functions of customizing the interface, so the implemented application allows users to increase or decrease font size, enable the high contrast mode and switch the language of the content. Additionally, the application's color pallete was chosen in such a way that page elements meet requirements about contrast values between the foreground and the background.

Both websites have been analysed in terms of availability. To make this happen, there have been conducted a bunch of studies:

- measuring the time of finding a search panel for its various locations;
- measuring the time of selecting the announcement category for various locations of the menu elements;
- measuring the time of finding options to change the font size for buttons with various icons;
- checking the existence of alternative texts for pictures, links and buttons;
- checking the existence of ARIA attributes;
- checking the contrast level of the views;
- measuring the time of finding specified elements for various contrast modes;
- measuring the time of finding the option to enable the high contrast mode for various locations of the button;
- measuring the time of finding option to change the font size for various locations of buttons;
- measuring the time of finding the option to switch the language of the content for various locations of the button.

Three separate methods have been used to make this research: eye-tracking, LUT checklists and WAVE Evaluation Tool, which check the compliance with the WCAG 2.0 rules.

3.1. Eye-tracking

Eye-tracking is a method of collecting data concerning the eye movement that can be used to perform statistical analyzes. It relies on device that records the video of the eye. It can be remote, stationary or mobile. Any type is used for slightly different purposes. Remote and stationary devices can record the person's behavior and actions on the computer screen. On the other hand, mobile ones make it possible to record the scene in front of the person. All type of eye-trackers use diodes emitting infrared light that reflects of the eye surface. The contrast between the pupil and the iris makes it possible to precisely detect the image of the pupil whose center is used with the reference points to mark places of looking at the screen [13]. Eye-tracker uses two main states of the eye: a fixation (which consist in stopping the eye's movement) and a saccade (consisting in moving the eyesight between fixations). Duration of each fixation also matters during the analysis [14].

The device used to the research was Gazepoint GP3 HD. Basic parameters of this eye-tracker are presented in the Table 1.

Table 1: Basic parameters of the Gazepoint GP3 HD eye-tracker

View angle accuracy	0.5° – 1.0°
Sampling frequency	60 Hz or 150 Hz
Calibration	5-point or 9-point
Head movements allowable range	35 cm (vertical) x 22 cm (horizontal)
Head movement depth range	±15 cm
Dimensions	235 x 45 x 47 mm
Weight	125 g
Compatible screen sizes	Up to 24"

To start the eye-tracking research, about twenty scenarios have been prepared, containing screen shots linked with precise instructions for participants of the analysis. Each task was about finding a specific element at the displayed picture. The eye-tracker was cooperating with the iMotions 9.0 platform for the analysis. It made it possible to create the presentation, including all the prepared scenarios and present it to participants.

There was a group of ten students between ages of 23 and 25 years old that took part in the eye-tracking analysis. Several participants were wearing glasses, but this fact was not considered during analysis. All of them claimed that they use the Internet every day, but they have never visited any of examined websites before. All participants were informed about how the experiment will look like and how they should behave during it. This study allowed to observe times of exercising prepared scenarios. It also allowed to receive images containing paths of participants' looking places and generate heatmaps presenting their areas of interest.

Eye-tracking study was also used for example by authors of [5, 6, 15]. All of described researches consisted of finding usability problems of some websites. The participants were asked to perform defined tasks included in experiment scenarios. All results have shown some usability problems that can be found on public websites, but none of described researches included a project paradigm aspect.

3.2. LUT checklists

There have been also prepared LUT checklists, which aimed to test the availability of both websites. It concerned five separate areas [12]:

- navigation and structure;
- messages, feedback, help for the user;
- application interface;
- text of subpages;
- data input.

Each area consisted of a few subareas to which there were assigned various questions that made it possible to evaluate website's usability and accessibility.

Those checklists were submitted by the same group of participants as for the eye-tracking analysis. They had to answer all questions by selecting number from 1 to 5. Meaning of every rating value has been presented in the Table 2.

Table 2: Meaning of ratings [12]

Rating	Description
1	There were critical usability issues, preventing or discouraging the use of the website.
2	There were serious usability issues that could prevent most users from getting done things, they wanted to do.
3	There were minor usability issues that do not separately cause problems to most users, but their accumulation can affect the quality of the user's work.
4	There were single minor usability problems, that may reduce the quality of work with the website.
5	There were no usability issues or user performance issues.

3.3. WAVE Evaluation Tool

Publication [16] is really helpful while creating websites adapted to the widest possible audience [17]. There are some tools that support finding accessibility problems of website basing on guidelines included in this publication. An example of such technology that was used to the analysis is WAVE Evaluation Tool [4]. It allows to observe which WCAG 2.0 rules are not present at the website. For the research, most important information from WAVE Evaluation Tool was if there occurred errors at both websites and what type of errors they were, if there were problems with too low contrast and how many ARIA attributes there were in the code of each service.

WAVE Evaluation Tool was also used for example by authors of paper [8]. Of course, there are a lot of different usability validators and the same exact article includes experiments with Achecker and SortSite programs too. Research with the use of Tenon and VoiceOver tools was described in article [9]. Both publications show some examples of the most popular usability problems based on experiments conducted with the use of various accessibility validators. However, all of examined websites had just one version, so the impact of universal design could not be verified.

4. Results

4.1. LUT checklists

After the survey research, there has been calculated a WUP value for each checklist. It reflects the level of usability of the website and it is defined by formula 1 [12]:

$$WUP = \frac{1}{n_a}\sum_{i=1}^{n_a}\frac{1}{s_i}\sum_{j=1}^{s_i}\frac{1}{q_{ij}}\sum_{k}^{q_{ij}} p_{ijk} \qquad (1)$$

where:
n_a – number of question areas
s_i – number of question sub-areas of the j area
q_{ij} – number of questions inside the i area and the j sub-area
p_{ijk} – rating of the k question inside the i area and the j sub-area

The WUP value is in the range from 1 to 5, where higher value means better website's usability. The comparison of WUP values calculated for both examined services is presented in Figure 1. It shows two boxplots – first referring to the Gumtree and second concerning the newly implemented website.

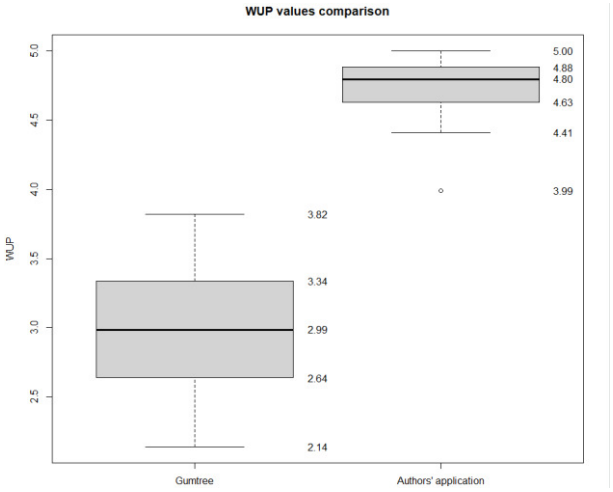


Figure 1: WUP values comparison.

It is clearly visible that for the authors' application, WUP values are at a higher level, which means that it received better ratings in surveys. For the Gumtree website, they range from 2.14 to 3.82. The competing app received 3.99 as the lowest value of the WUP, but it has been classified as an outlier. The lowest value calculated for the authors' application which is not an outlier is 4.41. The highest of them is 5, which is the highest possible value, meaning that the respondent considered all aspects of the application included in the survey as not causing usability problems. In the case of Gumtree, the median of WUP value is 2.99, while for the authors' applications it is at the level of 4.8. A clear advantage of the second one indicates a better user experience while using this website.

4.2. WAVE Evaluation Tool

As a result of the experiment using the digital accessibility validator there are presented two charts in Figure 2 and Figure 3. The first one contains a graphical comparison of both tested websites in terms of the total number of errors that may have a negative impact on accessibility, in particular for people with disability.

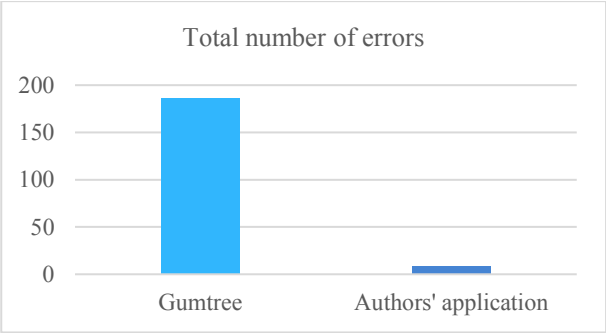


Figure 2: Total numbers of errors comparison.

The results show that significantly fewer problems were revealed for the application implemented by authors – a total of 9 errors, compared to 186 at the Gumtree website. The most common problems of the second one are: lack of text alternatives for images, very low contrast between the foreground and the background of elements and the shortage of labels assigned to form fields. Additionally, the content of many buttons and links consists only of graphics and almost every subpage does not have a properly defined document language.

Figure 3 shows a graph which present a comparison of the ARIA attributes number found in the source codes of both examined webpages.

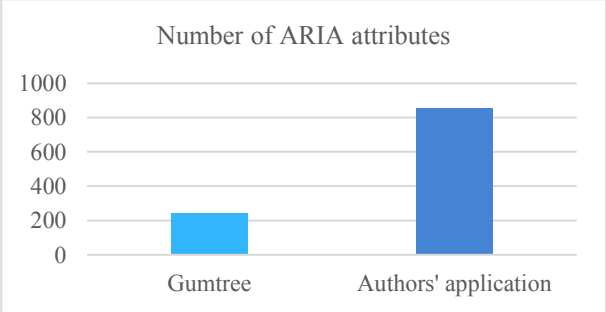


Figure 3: Numbers of ARIA attributes comparison.

It is worth mentioning that there are not such a number of ARIA attributes to ensure that the tested website is fully accessible, but the use of them directly affects the effectiveness of using the application with the support of assistive technologies. Because of that, it can be concluded that with the increase of the number of ARIA attributes, the level of website's availability increases too. In this aspect, the authors' application is much better adapted to the widest possible group of users – there were identified 853 ARIA attributes in its source code and Gumtree contains only 241 of them.

4.3. Eye-tracking

The analysis of the results of the eye-tracking research can be divided into two parts: qualitative and quantitative. The quantitative results of experiments are presented in Figure 4 and Figure 5. The first one shows differences in average time of completing several tasks on two sites compared: Gumtree and the site created by authors. Second one shows average numbers of fixations made by users during completing tasks. These tasks were to find some elements of the interface e.g., first task was: "Find the classifieds search field".

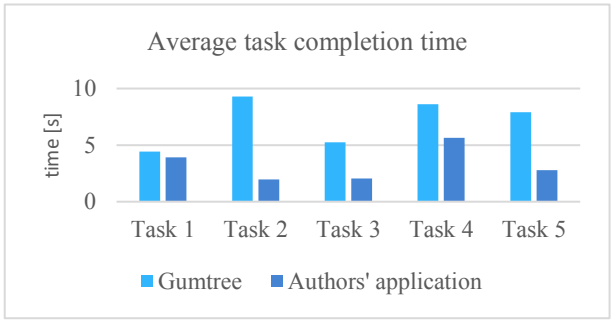


Figure 4: Average task completion times comparison.

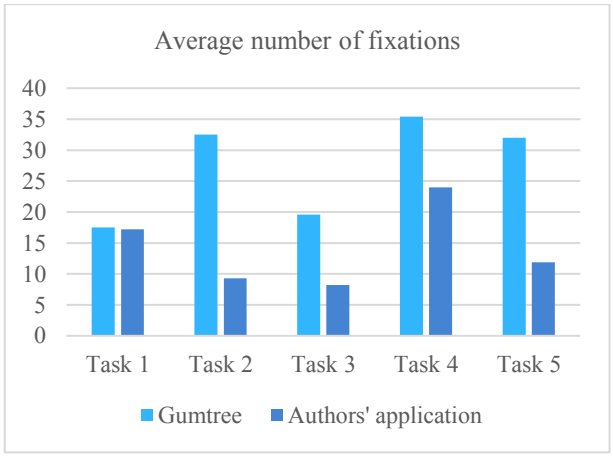


Figure 5: Average numbers of fixations comparison.

The charts clearly show that any task on the Gumtree page is more difficult to complete than on the other page and in most cases the time difference is significant. Only in the case of the first task, the differences are slight. Screenshots used in research for this task are showed in figures 6 and 7. Additionally, the desired places are marked with red ellipses, and as you can see, search field is positioned in two differed places.

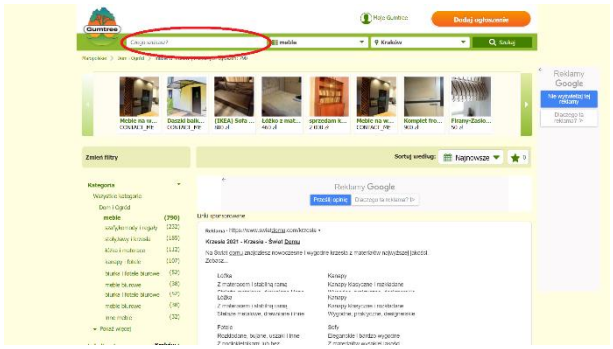


Figure 6: Search field location – Gumtree.

In case of the Gumtree site, search field is located on the left side of the top bar. In case of the authors' site, this field is located under the top bar, on the right side above the filters, which as occurred, it may be unintuitive place for it. To understand what is wrong with the placement, you need to know where users expect the search box to be placed. In this case, scanning path and heat maps are helpful and can allow solving this issue.

Examples of scanning paths are showed in Figure 8 and Figure 9. This path illustrates how a single user focused their eyesight.

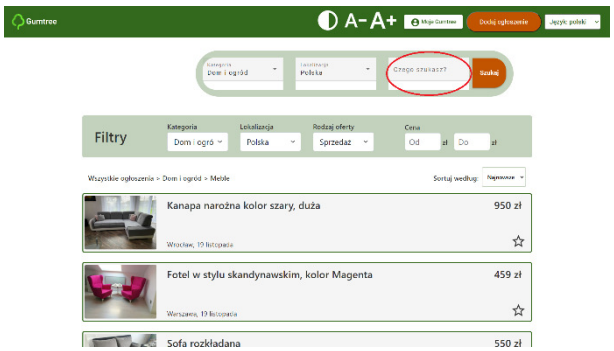


Figure 7: Search field location - authors' application.

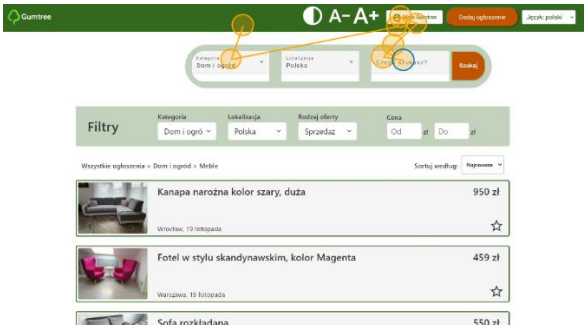


Figure 8: Sample scan path from the task 1 – authors' application.

Figure 9: Sample scan path from the task 1 – authors' application.

Both figures show that users have found the desired object. The first user expected to see the search box in the top bar, but it did not stop him or her from finding it quickly. It took a little longer for the second user to find the search box and it looks like he or she was not sure if this was what he or she was looking for, which can be deduced from the large number of fixations in the vicinity of the searched object. However, to get more general conclusions, heat maps presented in figures 10 and 11 should be analyzed.

Both heat maps show that users are looking for the search box on the right side of the top bar. Moreover, they seem to check all the distinctive bars, which means that on the authors' website, the filter bar also attracts their attention. That means that the search bar should stand out more. Yellow color on category and localization selects, may suggest that search bar should be simpler, or that question should be more specific.

Figure 10: Heat map from the task 1 – Gumtree.

Figure 11: Heat map from the task 1 - authors' application.

Results for the other tasks more significantly shows the influence of using the universal project design rules during designing the interface. Especially task 2 shows, how bad interface can affect on users experience – users were on average over 4 times longer to search for a category on Gumtree site than on a page provided by the authors. The heat maps for this task are presented in figures 12 and 13. They show that presenting the user with too much information makes the page difficult to read. Although on authors' site the categories are presented in a more concise way, users did not check all categories as in the case of Gumtree and quickly found the right category. This suggests that big icons, associated with categories catch users' attention and allow them to quickly find what they are looking for.

The second part of the eye tracker examination is to check how small changes affect the time to find the desired objects. These changes include making the icons more readable, moving the buttons around, increasing the default font size, or turning on the high-contrast version. Results of that research are presented in Figure 14.

The chart shows that in each case the task on the alternative version of the interface took less time, but in half of the cases the difference is very small. Actions which really makes difference were: changing icon, turning on high-contrast version and increasing font size. The first task was to find the option to change the font size using buttons with two different icons responsible for it. In Figure 15 both icons are presented. The

version of the icons at the bottom of this picture is easier for users to find.

Figure 12: Heat map from the task 2 – Gumtree.

Figure 13: Heat map from the task 2 - authors' application.

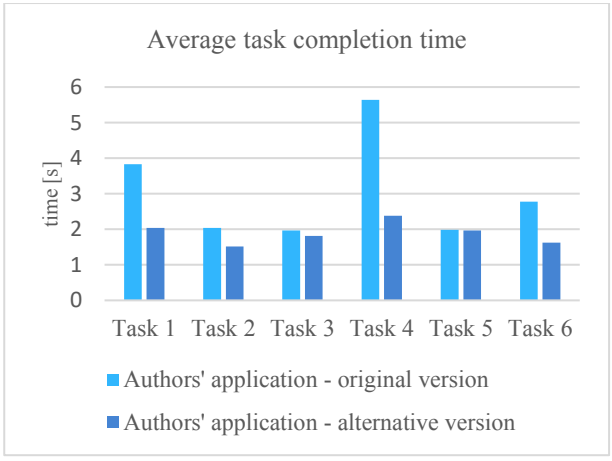


Figure 14: Average task completion times comparison.

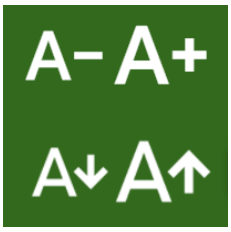


Figure 15: Alternative icons for changing the font size.

In the task 4, alternative version of site was with high contrast, which makes users easier to find desired object, which was localization given by author of the announcement. Basic version of interface is shown in Figure 16, alternative version is in Figure 17.

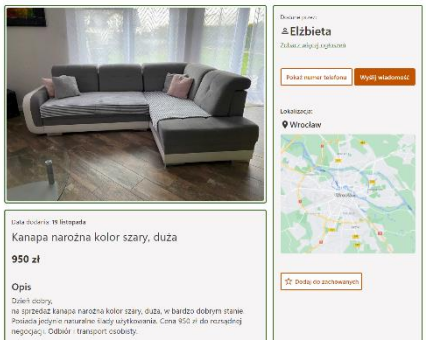


Figure 16: Interface with disabled the high-contrast option.

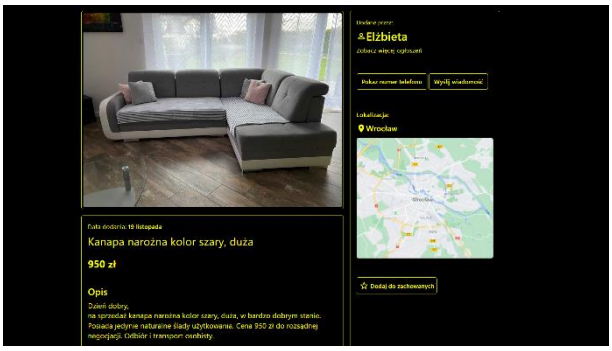


Figure 17: Interface with enabled the high-contrast option.

This result may suggest that allowing the user to enable a high-contrast version of the page may help the user to use the page more productively. It could also mean that an original version might have problems with contrast or colors. Although, the heat map presented in Figure 18 shows that users still have some problems to find proper object, and probably highlighting the field containing the location, or changing its font size may solve this problem, but this is an issue for further consideration and requires further research.

In the task 6, the respondents were to find the option of displaying all advertisements of a given user on interfaces with different font sizes. The interface with bigger font size, and marked desired option is shown in Figure 19.

Figure 18: Heat map from the task 4

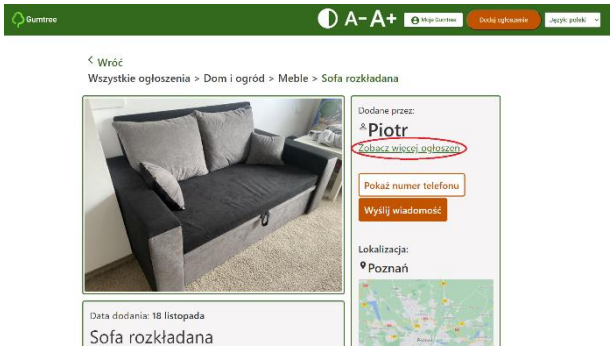


Figure 19: Interface with the increased font size

The results of these studies show that minor changes to the interface can slightly improve the user experience, but giving users the ability to customize the page by changing the font size and allowing the choice of high-contrast versions makes a real difference to the usability of the page.

5. Conclusions and future works

In this paper there has been deeply analysed an impact of the universal design paradigm application to websites. There have been conducted a series of experiments in order to compare two similar applications in terms of usability. All results have shown that applying the principles of universal design implicates increased level of interface's usability. The research made it possible to confirm all the hyoptheses, which were as follows:

- Designing a web application using the universal design rules increases the simplicity and intuitiveness of its use.
- Designing a web application using the universal design rules increases its availability for users.
- The possibility of adjusting the contrast and the font size has a positive effect on the effectiveness of using the website.

Future work directions may for example include increasing the number of participants of the experiments, because there were only ten of them. This amount of research group should allow to find almost all of the occurring errors. However, bigger population would authenticate received results and perhaps reveal some other usability problems. The number of eye-tracking scenarios could possibly be increased too. It would allow to compare some different elements in terms of the impact of universal design principles.

There have been compared just two websites. Another future work directions may consist of analyzing more classifieds services. In order to make the whole research more complete, it can be considered to examine some other webpages from various life areas. Research like that could prove that the universal design principles have a positive impact for all applications' target users.

References

[1] World Health Organization, World report on disability, World Health Organization, 2011.

[2] A. Buczek, J. Sikorski, Dostępność stron internetowych wybranych instytucji użyteczności publicznej dla osób z niepełnosprawnością, Niepełnosprawność 30 (2018) 281-294, https://doi.org/10.4467/25439561.NP.18.031.9869.

[3] M. Popiołek, Wykluczenie cyfrowe w Polsce, Nierówności społeczne a wzrost gospodarczy 32 (2013) 310-320.

[4] D. A. Ryley-Huff, Web Accessibility and Universal Design: A Primer on Standards and Best Practices for Libraries, Library Technology Reports 48 (2012) 29-35, https://doi.org/10.5860/ltr.48n7.

[5] C. Ehmke, S. Wilson, Identifying Web Usability Problems from Eye-Tracking Data, The 21st British HCI Group Annual Conference University of Lancaster (2007), https://doi.org/10.14236/ewic/HCI2007.12.

[6] Z. Xin, Y. Mei, W. Chenglong, W. Lirong, Study of sustainable factors in universal design, IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design (2009) 1412-1415, https://doi.org/10.1109/CAIDCD.2009.5374950.

[7] S. Khodambashi, H. Gilstad, Ø. Nytrø, Usability evaluation of clinical guidelines on the web using eye-tracker, Exploring Complexity in Health: An Interdisciplinary Systems Approach, IOS Press, Amsterdam, 2016.

[8] S. Alismail, W. Chipidza, Accessibility evaluation of COVID-19 vaccine registration websites across the United States, Journal of the American Medical Informatics Association 28 (2021) 1990-1995, https://doi.org/10.1093/jamia/ocab105.

[9] E. Scanlon, Z. W. Taylor, J. J. Chini, Physics webpages create barriers to participation for people with disabilities: Five steps to increase digital accessibility (2019), https://doi.org/10.48550/arXiv.1907.02906.

[10] B. A. Zardari, Z. Hussain, A. A. Arain, W. H. Rizvi, M. S. Vighio, QUEST e-learning portal: Applying heuristic evaluation, usability testing and eye tracking, Universal Access in the Information Society 20 (2021) 531-543, https://doi.org/10.1007/s10209-020-00774-z.

[11] 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, https://doi.org/10.1109/HSI.2013.6577815.

[12] M. Miłosz, Ergonomia systemów informatycznych, Politechnika Lubelska, Lublin, 2014.

[13] L. Cooke, Improving usability through eye tracking research, International Professional Communication Conference (2004) 195-198, https://doi.org/10.1109/IPCC.2004.1375297.

[14] A. Bojko, Eye Tracking the User Experience: A Practical Guide to Research, Rosenfeld Media, New York, 2013.

[15] P. Weichbroth, K. Redlarski, I. Garnik, Eye-tracking web usability research, Federated Conference on Computer Science and Information Systems 8 (2016), http://dx.doi.org/10.15439/2016F127.

[16] B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, Web content accessibility guidelines (WCAG) 2.0, WWW Consortium (W3C) 290 (2008), https://www.w3.org/TR/WCAG20/.

[17] C. Ślusarczyk, WCAG 2.0 jako podstawa uniwersalnego projektowania stron internetowych, Zeszyty Naukowe Uniwersytetu Szczecińskiego. Ekonomiczne Problemy Usług 112 (2014) 461-468.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 42–47
Received: 20 September 2022
Accepted: 11 October 2022

A comparative analysis of contemporary integrated Java environments

Analiza porównawcza współczesnych zintegrowanych środowisk do pracy w języku Java

Cezary Piotr Kaczorowski

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

Abstract

The purpose of this study was to compare three of the most popular Java development environments: IntelliJ IDEA, Netbeans, and Eclipse, in order to identify the best one for specific user archetypes. Using a program specially designed for this task, the execution time was measured, as well as the CPU and RAM usage during the execution of the program on each environment. The obtained data was collected and processed accordingly. Additionally, the environments were compared in a number of ways, including features that distinguish them from each other, ease learnability for a new user or support from the developers. The study concluded with IntelliJ IDEA being the best environment for all of the considered archetypes.

Keywords: Java environments; IntelliJ IDEA; NetBeans; Eclipse

Streszczenie

Celem pracy było porównanie trzech najpopularniejszych środowisk programistycznych do pracy w języku Java: IntelliJ IDEA, Netbeans i Eclipse, w celu wskazania najlepszego z nich dla konkretnych archetypów użytkowników. Przy pomocy stworzonego specjalnie programu zbadany został czas wykonywania programu, a także obciążenie procesora i pamięci w trakcie wykonywania programu w poszczególnych środowiskach. Dodatkowo środowiska zostały porównane pod wieloma innymi względami, m.in. pod kątem cech odróżniających je od siebie, łatwości opanowywania przez nowego użytkownika, czy wsparcia ze strony twórców oprogramowania. W rezultacie stwierdzono, że IntelliJ IDEA jest najlepszym środowiskiem dla wszystkich rozpatrywanych archetypów użytkowników.

Słowa kluczowe: środowiska Java; IntelliJ IDEA; NetBeans; Eclipse

*Corresponding author

Email address: cezary.kaczorowski@pollub.edu.pl (C. P. Kaczorowski)

1. Wstęp

W obecnych czasach programiści mają do wyboru wiele platform umożliwiających im tworzenie oprogramowania w najróżniejszych językach. Nie inaczej jest z tymi, którzy tworzą aplikację w oparciu o język Java[1, 2]. Dodatkowo częsta aktualizacja platform powoduje duże wahania użyteczności poszczególnych rozwiązań na tle konkurencji.

W związku z mnogością rozwiązań i towarzyszącą im dynamiką zmian, pojawia się potrzeba dostarczenia obiektywnej wiedzy, która pomoże programistom języka Java w wyborze zintegrowanego środowiska programistycznego. Niniejszy artykuł dotyczy porównania środowisk IntelliJ IDEA, NetBeans oraz Eclipse, pod kątem wydajności, funkcjonalności oraz dopasowania do potrzeb typowych archetypów użytkowników. Do porównania wybrano środowiska, które w [3] uznano za najpopularniejsze, a więc warte szczególnej uwagi. Warto wskazać, że nie udało się odnaleźć prac innych autorów, które ten temat analizowałyby w tak szerokim spektrum.

Porównywane środowiska programistyczne zostały ocenione w oparciu o autorskie kryteria uwzględniające archetypy ich potencjalnych użytkowników. Wykonano testy, mające na celu zbadanie czasu wykonywania programu oraz obciążenia komputera generowanego w trakcie wykonywania programu. Dodatkowo przeanalizowano najważniejsze cechy poszczególnych środowisk, ich funkcjonalności oraz cechy odróżniające je od siebie nawzajem.

Postawiono pytanie badawcze: Czy środowisko IntelliJ IDEA jest najlepszym środowiskiem programistycznym do pracy w języku Java? Towarzyszą temu następujące cztery hipotezy:

- H1: Środowisko IntelliJ IDEA jest wydajniejsze niż NetBeans czy Eclipse.
- H2: Środowisko IntelliJ IDEA jest bardziej popularne niż NetBeans czy Eclipse.
- H3: Środowisko IntelliJ IDEA jest najlepiej rozwijanym środowiskiem programistycznym.
- H4: IntelliJ IDEA jest najmniej przystępnym środowiskiem do pracy w języku Java.

2. Metodyka badań

W kolejnych podrozdział przedstawiono aspekty metodyki badań takie, jak: kryteria oceny, archetypy użytkowników, przebieg procedury testowej, sprzęt i oprogramowanie.

2.1. Kryteria oceny

Na potrzeby porównania wybranych środowisk programistycznych języka Java zdefiniowano następujące kryteria oceny:

1. Wydajność – czas w sekundach wykonywania testowego programu w poszczególnych środowiskach.
2. Obciążenie procesora – procentowe obciążenie procesora w trakcie wykonywania testowego programu.
3. Zajętość pamięci – zajętość pamięci RAM w MB w trakcie wykonywania testowego programu.
4. Popularność – zainteresowanie środowiskiem wśród populacji programistów (określone na podstawie zewnętrznych źródeł).
5. Funkcjonalność – mocne i słabe strony wybranych środowisk oraz odróżniające je od siebie możliwości.

2.2. Archetypy użytkowników

Podczas porównania wybranych środowisk programistycznych wspomagano się archetypami użytkowników. Archetypy użytkowników pozwalają zdywersyfikować oczekiwania wobec optymalnego wyboru środowiska, a także ustanowić hierarchię ich oczekiwań. Zdecydowano się zdefiniować następujące archetypy:

1. Zwykły użytkownik – nie ma specjalnych wymagań dotyczących środowiska, wszystkie kryteria stawia na równi.
2. Użytkownik ze słabszym komputerem - potrzebuje środowiska, które będzie najmniej obciążało komputer, mniej uwagi zwraca na szybkość działania i nie przykłada wcale uwagi do popularności czy funkcjonalności środowiska.
3. Doświadczony użytkownik – potrzebuje środowiska, które zaoferuje najwięcej funkcjonalności i nie zwraca uwagi na pozostałe kryteria.

Archetypy w połączeniu z kryteriami oceny umożliwiają wskazanie najlepszego środowiska dla danego użytkownika. Każdy archetyp posiada własną hierarchię potrzeb, w związku z tym najlepsze dla niego środowisko powinno osiągać najlepsze rezultaty w istotnych dla archetypu kryteriach oceny. W przypadku niejednoznacznych wyników dla najistotniejszych kryteriów dla danego archetypu, następne w kolejności będą brane pod uwagę kryteria pokrewne.

2.3. Przebieg procedury testowej

Dla celów przetestowania wybranych środowisk programistycznych został napisany specjalny program. W trakcie jego tworzenia postępowano zgodnie z zasadami pisania dobrego kodu [4, 5]. Zadaniem programu było wygenerowanie odpowiednio dużego obciążenia, aby uzyskać rezultaty mogące podlegać miarodajnej analizie.

Program testowy składał się z trzech testów, które polegały na przeprowadzaniu dużych obliczeń w pętli. Każdy z testów był przeprowadzany przez osobny wątek tak, aby testy mogły wykonywać się jednocześnie. Kod poszczególnych przypadków testowych został przedstawiony na listingach 1-3.

Procedura testowa była wykonywana według następującego schematu:

1. Wstępne uruchomienie programu testowego w aktualnie badanym środowisku – dzięki temu program był identyfikowany przez narzędzie VisualVM.
2. Wybranie pomiaru obciążenia procesora (CPU) i obciążenia pamięci RAM (Memory) w narzędziu VisualVM oraz uruchomienie pomiaru przyciskiem „Start”.
3. Uruchomienie przypadków testowych przedstawionych na listingach 1-3.
4. Odczekanie, aż program testowy się wykona a narzędzie VisualVM zbierze wszystkie potrzebne dane.
5. Zapisanie danych z pomiaru i wyeksportowanie ich do pliku z rozszerzeniem csv.

Listing 1: Kod pierwszego testu

```
public static void firstTest(int index) {
    int n = 10000*index;
    double var1 = 123456789;
    double var2 = 987654321;
    for(int j = 0; j < n; j++) {
        for(int i = 0; i < n; i++) {
            var1 = var1 * var2;
        }
    }
    System.out.println("\n---First test progress: "
            + index +  "0%");
}
```

Listing 2: Kod drugiego testu

```
public static void secondTest(int index) {
    int n = 10000*index, var = 0;
    String chain = "";
    List<String> list = new ArrayList<>();
    for(int i = 0; i < n; i++) {
        var = (int) pow(i,i);
        list.add(Integer.toString(var));
        chain += list.get(i);
    }
    chain = Double.toString(sqrt(
            Double.parseDouble(chain)));
    if(Double.parseDouble(chain)>var) {
        chain = Double.toString(pow(var,var));
    }
    System.out.println("\n---Second test progress: "
            + index +  "0%");
}
```

Listing 3: Kod trzeciego testu

```
public static void thirdTest(int index) {
    int n = 10000*(index-5);
    for(int j = n; j > 0; j-=2) {
        for(int i = 0; i < n; i+=2) {
            Double var = pow(i,j);
        }
    }
    System.out.println("\n---Third test progress: "
            + index +  "0%");
}
```

2.4. Sprzęt i oprogramowanie

Parametry komputera, na którym przeprowadzono testy były następujące:

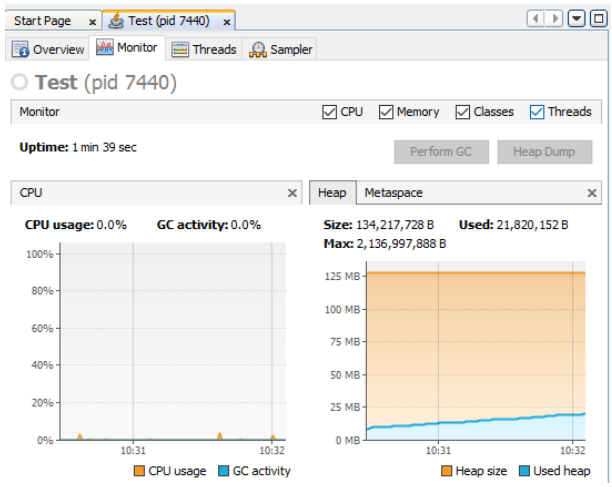
- Procesor: Intel Core i7-9700 3.00GHz.

- Karta graficzna: NVIDIA GeForce RTX 2060.
- Pamięć RAM: 32GB DDR4.
- System operacyjny: Windows 10 wersja 64 bitowa.
- Wersja środowiska Java: JDK 11[6].

Porównywane środowiska programistyczne języka Java były zainstalowane w następujących wersjach:

- Wersja IntelliJ IDEA: 2022.1.1.
- Wersja NetBeans: IDE 13.
- Wersja Eclipse: 2022-03(4.23.0).

W celu zebrania danych odnośnie czasu wykonywania programu testowego, obciążenia procesora i zajętości pamięci RAM, wykorzystano narzędzie VisualVM [7], które współpracuje z każdym z porównywanych środowisk programistycznych. Rysunek 1 przedstawia przykładowy pomiar przy użyciu tego programu.



Rysunek 1: Przykładowy zrzut ekranu, prezentujący pomiary w narzędziu VisualVM dla środowiska Eclipse.

3. Analiza wyników testów

Dla każdego pomiaru zebrano następujące dane dla każdego z porównywanych środowisk:

- czas wykonania programu [s],
- minimalne obciążenie procesora [%],
- maksymalne obciążenie procesora [%],
- średnie obciążenie procesora [%],
- mediana obciążenia procesora [%],
- minimalna zajętość pamięci [MB],
- maksymalna zajętość pamięci [MB],
- średnia zajętość pamięci [MB],
- mediana zajętości pamięci [MB].

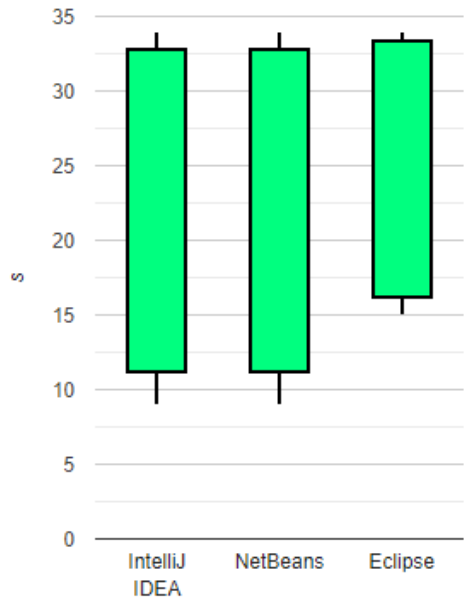
3.1. Czas wykonywania programu

Zestawienie czasów wykonywania programu testowego przedstawiono w Tabeli 1. Środowisko Eclipse okazało się najwolniejszym z porównywanych środowisk. Środowiska Netbeans i IntelliJ IDEA uzyskały bardzo zbliżone wyniki, z nieznaczną przewagą środowiska Netbeans.

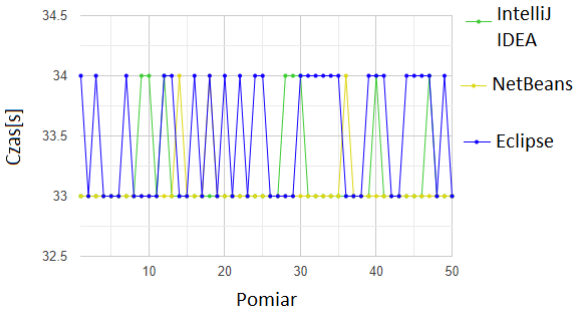
Tabela 1: Czas wykonywania programu testowego [s]

	IntelliJ IDEA	NetBeans	Eclipse
Minimum	9,00	9,00	15,00
Maksimum	34,00	34,00	34,00
Mediana	23,50	23,00	26,00
Średnia	33,27	33,19	33,66

Na Rysunku 2 przedstawiono wykres pudełkowy dla zebranych czasów wykonywania programu testowego. Z kolei Rysunek 3 przedstawia czasy wykonania poszczególnych pomiarów.



Rysunek 2: Wykres pudełkowy dla czasu wykonywania programu testowego [s].



Rysunek 3: Wykres liniowy dla czasu wykonywania programu testowego [s].

Wszystkie czasy różnią się od siebie o bardzo małe wartości, które mogą być bardziej widoczne przy bardziej złożonych i skomplikowanych projektach. Jednakże z zebranych danych wynika, że różnica pomiędzy dwoma wiodącymi w tej kategorii środowiskami jest marginalna, w związku z czym zarówno NetBeans, jak i IntelliJ IDEA mogą być uznane za równie szybkie.

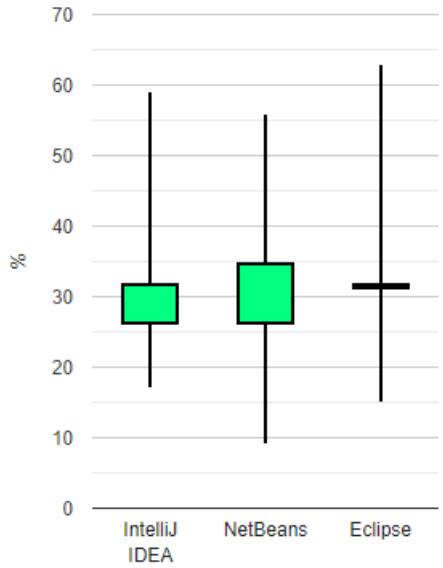
3.2. Obciążenie procesora

Zestawienie obciążenia procesora podczas wykonywania programu testowego przedstawiono w Tabeli 2. Środowisko Eclipse uzyskało zdecydowanie najgorszy wynik. Środowiska Netbeans i IntelliJ IDEA ponownie otrzymały zbliżone wyniki, z przewagą na rzecz tego pierwszego.

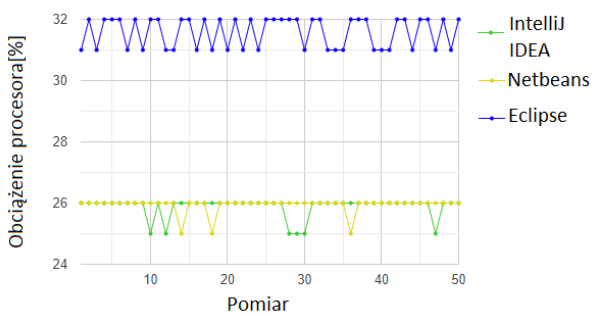
Na Rysunku 4 przedstawiono wykres pudełkowy dla zebranych obciążeń procesora podczas wykonywania programu testowego. Z kolei Rysunek 5 przedstawia obciążenia procesora podczas poszczególnych pomiarów.

Tabela 2: Obciążenie procesora podczas wykonywania programu testowego [%]

	IntelliJ IDEA	NetBeans	Eclipse
Minimum	17,00	9,00	15,00
Maksimum	59,00	56,00	63,00
Mediana	26,00	26,00	31,00
Średnia	26,32	26,32	32,07



Rysunek 4: Wykres pudełkowy dla obciążenie procesora podczas wykonywania programu testowego [%].



Rysunek 5: Wykres liniowy dla obciążenia procesora podczas wykonywania programu testowego [%].

Okazało się, że środowisko Eclipse najbardziej obciążało procesor komputera. Ponownie, jak w przypadku czasu wykonania, różnica pomiędzy IntelliJ IDEA i NetBeans jest na tyle niewielka, że oba środowiska można uznać za równie wydajne w kwestii obciążenia procesora.

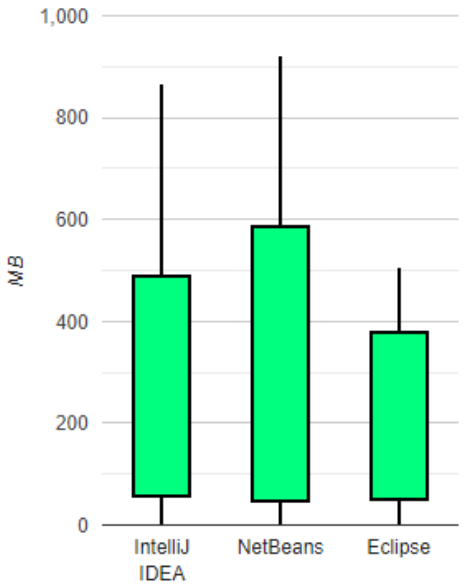
3.3. Zajętość pamięci

Zestawienie zajętości pamięci podczas wykonywania programu testowego przedstawiono w Tabeli 3. Tym razem środowisko Eclipse uzyskało zdecydowanie najlepszy wynik. Środowiska Netbeans i IntelliJ IDEA ponownie otrzymały zbliżone wyniki, z przewagą na rzecz tego drugiego.

Tabela 3: Tabela zawierająca najważniejsze wartości z wykresu pudełkowego średniego zużycia pamięci [MB]

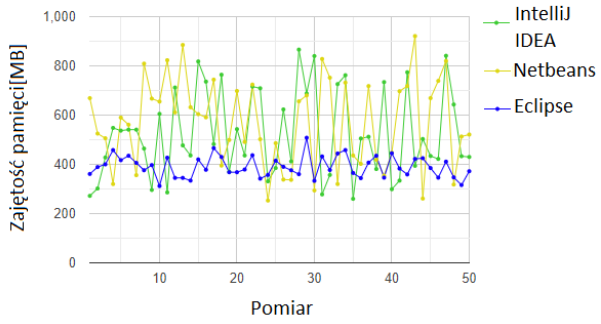
	IntelliJ IDEA	NetBeans	Eclipse
Minimum	13,21	7,54	14,13
Maksimum	866,00	921,00	508,00
Mediana	178,50	174,00	205,00
Średnia	524,19	570,15	391,43

Na Rysunku 6 przedstawiono wykres pudełkowy dla zebranych zajętości pamięci podczas wykonywania programu testowego. Z kolei Rysunek 7 przedstawia zajętość pamięci podczas poszczególnych pomiarów.



Rysunek 6: Wykres pudełkowy dla zajętości pamięci podczas wykonywania programu testowego [MB].

Zwycięzcą w kategorii zajętości pamięci okazało się środowisko Eclipse. Na kolejnym miejscu uplasowało się środowisko IntelliJ IDEA, a następnie NetBeans. Tym razem różnice okazały się znaczące.



Rysunek 7: Wykres liniowy dla zajętości pamięci podczas wykonywania programu testowego [MB].

4. Porównanie funkcjonalności środowisk programistycznych

W kolejnych podrozdziałach przedstawiono analizę funkcjonalności porównywanych środowisk programistycznych.

4.1. Eclipse

Eclipse [8] jest darmowym środowiskiem programistycznym, a więc jest ogólnodostępny dla wszystkich użytkowników. Posiada bogatą dokumentację, która ułatwia nowym użytkownikom nabycie wiedzy, jak korzystać z tego środowiska. Eclipse dostarcza wsparcie dla platformy GitHub, jednakże nie jest ono zbyt intuicyjne w obsłudze. Jest również dostępna opcja integracji projektu z narzędziami Docker czy Kubernetes. Eclipse posiada bogatą bibliotekę wtyczek (ang. plug-in) wspierających i ułatwiających pracę programisty. Zgodnie z danymi z Eclipse Marketplace jest ich ponad 1500. Co jednak odróżnia Eclipse od pozostałych środowisk, to wsparcie dla tworzenia własnych języków w oparciu o te już istniejące - opisano to chociażby w artykułach [9] i [10]. Choć jest to zdecydowanie ogromna zaleta tego środowiska, to jednak opcja tworzenia własnego języka nie jest do końca przydatna dla programisty, który zamierza pracować z językiem Java. Eclipse jest aktualizowany co 3 miesiące.

4.2. NetBeans

Środowisko Netbeans [11] tak, jak i Eclipse, jest w pełni darmowym zintegrowanym środowiskiem programistycznym. Jest ono dosyć proste, a więc względnie łatwe w nauce, jednak dla bardziej doświadczonych użytkowników nie oferuje zbyt wiele, zwłaszcza na tle pozostałych dwóch środowisk. Netbeans nie oferuje funkcjonalności, która by go szczególnie wyróżniała na tle konkurencji w postaci Eclipse czy IntelliJ IDEA. Środowisko posiada mało wtyczek (około 60) i nie oferują one niczego, czego nie oferowałyby pozostałe środowiska. Netbeans posiada wbudowany profiler oparty o VisualVM służący do analizy aplikacji, jednakże nie jest zbyt rozbudowany. Pozostałe środowiska oferują, choćby dzięki dodatkowym wtyczkom, znacznie więcej opcji. Jest dostępne wsparcie dla Git i konteneryzacji. Netbeans jest aktualizowany co około 3-4 miesiące

4.3. IntelliJ IDEA

Środowisko IntelliJ IDEA [12] jest dostępne w dwóch różnych wersjach: darmowej tzw. Community Edition i płatnej tzw. Ultimate Edition. Choć mogłoby się wydawać, że największe zalety tego środowiska będą niedostępne dla większości użytkowników poprzez zablokowanie wielu funkcjonalności w darmowej wersji, to sytuacja ma się inaczej. Znaczna większość funkcji oferowanych przez to środowisko jest dostępna w wersji Community [13]. IntelliJ IDEA posiada około 6035 wtyczek, z czego 5555 jest dostępne w darmowej wersji środowiska. Wyszukiwanie wtyczek jest bardzo intuicyjne, ze względu na wyszukiwarkę, która bardzo dobrze radzi sobie ze słowami kluczowymi. Takie rozwiązanie pozwala łatwo zaspokoić potrzeby projektów, ale może być przytłaczające dla początkującego użytkownika. IntelliJ IDEA, podobnie jak Eclipse, pozwala na integrację z platformą GitHub, ale zawiera ona o wiele więcej opcji i czytelniejszy interfejs. Wśród wielu użytecznych funkcjonalności można wyróżnić m.in.: inteligentne uzupełnianie kodu, generowanie diagramów na podstawie kodu, wyszukiwanie duplikatów kodu, czy integrację kodu w innym języku w projekcie. IntelliJ IDEA posiada wsparcie dla narzędzi Docker i Kubernetes, jednak ta druga jest dostępna jedynie w wersji Ultimate. Inną funkcjonalnością dostępną jedynie w płatnej wersji środowiska są „profiling tools" służące do analizy kodu. Największą wadą tego środowiska, oprócz zablokowania części funkcjonalności w bezpłatnej wersji, jest niezbyt dobra dokumentacja, która sprawia, że nauczenie się korzystania z tego środowiska nie jest zbyt łatwe. IntelliJ IDEA jest aktualizowany co miesiąc.

5. Podsumowanie

Przeprowadzone badania dostarczyły danych pozwalających na wyłonienie najlepszych środowisk w poszczególnych kryteriach:

1. Wydajność: Najlepszymi środowiskami okazały się zarówno NetBeans, jak i IntelliJ IDEA.
2. Obciążenie procesora: Najlepszymi środowiskami okazały się zarówno NetBeans, jak i IntelliJ IDEA.
3. Zajętość pamięci: Najlepszym środowiskiem okazał się bezsprzecznie Eclipse.
4. Popularność: Zgodnie z zewnętrznymi badaniami [3] Eclipse jest obecnie najpopularniejszym środowiskiem do pracy w języku Java.
5. Funkcjonalność: Najlepszym środowiskiem okazało się IntelliJ IDEA, które nawet w darmowej wersji oferuje o wiele więcej funkcjonalności wspomagających pracę programisty niż pozostałe środowiska, a w wersji płatnej jest bezkonkurencyjnie najlepszym środowiskiem pod względem oferowanych unikalnych funkcjonalności.

Po nałożeniu rezultatów w powyższych pięciu kryteriach na archetypy użytkowników, można wydać następujące rekomendacje:

- Zwykły użytkownik: Z racji tego, iż ten użytkownik nie ma żadnej hierarchii wśród kryteriów, najodpowiedniejszym dla takiego użytkownika środowi-

skiem wydaje się to, które okazało się najlepsze w największej liczbie kryteriów. Najlepszym więc wyborem dla zwykłego użytkownika jest IntelliJ IDEA, które jest najlepsze pod względem funkcjonalności oraz ex aequo najlepsze pod względem wydajności i obciążenia procesora.

- Użytkownik ze słabszym komputerem: Zgodnie z preferencjami tego użytkownika najlepszym wyborem będzie środowisko, które generuje najmniejsze obciążenie procesora oraz zajętość pamięci. Jednakże w obu tych kryteriach najlepsze okazują się różne środowiska. W takim wypadku trzeba rozstrzygnąć to za pomocą pokrewnego kryterium pomocniczego. Po wzięciu kryterium wydajności okazuje się, że najlepszym wyborem dla użytkownika ze słabszym komputerem jest IntelliJ IDEA bądź NetBeans.
- Doświadczony użytkownik: W tym przypadku nie ma problemu z wyłonieniem najlepszego środowiska. Jest nim bezsprzecznie IntelliJ IDEA, który okazał się bezsprzecznie najlepszy w tym jednym, ale znaczącym dla podanego archetypu kryterium.

Po przeprowadzeniu analiz i porównań można odpowiedzieć twierdząco na pytanie badawcze, że środowisko IntelliJ IDEA jest najlepszym środowiskiem programistycznym do pracy w języku Java. Można stwierdzić, że IntelliJ IDEA jest wydajniejsze niż pozostałe porównywane środowiska, jest najlepiej rozwijanym środowiskiem, jest najmniej przystępnym środowiskiem, nie jest najbardziej popularnym środowiskiem. Pomimo pewnych negatywnych aspektów, IntelliJ IDEA okazało się preferowanym środowiskiem dla każdego z trzech archetypów użytkowników.

Literatura

[1] B. Eckel, Thinking in Java, Wydanie IV Helion, 2006.

[2] C. S. Horstmann, Java. Podstawy, Wydanie X Helion, 2010.

[3] Ranking najlepszych środowisk IDE Java i kompilatorów języka Java, https://myservername.com/top-10-best-java-ides-online-java-compilers, [22.05.2022].

[4] N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo-Techniczne, 2002.

[5] R. C. Martin, Czysty kod. Podręcznik dobrego programisty, Helion, 2014.

[6] Dokumentacja języka Java w wersji JDK 11: https://docs.oracle.com/en/java/javase/11/docs/api/index.html, [22.05.2022].

[7] Strona główna programu VisualVM: https://visualvm.github.io, [22.05.2022].

[8] Strona Eclipse zawierająca opis środowiska dla deweloperów i najważniejszych funkcjonalności, https://scand.com/company/blog/eclipse-ide-for-java-developers/, [22.05.2022].

[9] A. Margheri, M. Masi, R. Pugliese, F. Tiezzi, A rigorous framework for specification, analysis and enforcement of access control policies, IEEE Transactions on Software Engineering, 45(2017), 2-33.

[10] L. Lopes, F. Martins, A safe-by-design programming language for wireless sensor networks, Journal of Systems Architecture, 63, (2016), 16-32.

[11] Strona NetBeans zawierająca opis środowiska dla deweloperów i najważniejszych funkcjonalności, https://netbeans.apache.org/download/nb120/, [22.05.2022].

[12] Strona IntelliJ IDEA zawierająca opis środowiska dla deweloperów i najważniejszych funkcjonalności, https://www.jetbrains.com/idea/features/, [22.05.2022].

[13] Strona IntelliJ IDEA zawierająca porównanie wersji płatnej (Utimate) z wersją darmową (Community), https://www.jetbrains.com/products/compare/?product=idea&product=idea-ce, [22.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 48–53
Received: 25 September 2022
Accepted: 17 October 2022

Choosing the optimal database system to create a CRM system

Wybór optymalnego systemu baz danych do stworzenia systemu CRM

Łukasz Szwałek*, Jakub Smołka

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

Abstract

MSSQL, MySQL and PostgreSQL are some of the most popular databases. The selection of the database for the creation of a CRM system is based mainly on the assessment of its effectiveness in terms of speed. The article aims to choose the optimal database system to create an effective CRM system. The literature review led to the hypothesis that MSSQL will be the fastest. A series of experiments using the app served as a test. During the research, a series of experiments were carried out using the order processing module, which is part of a larger CRM system aimed at the e-commerce industry. Each query was measured 10 times, the results were averaged. The research results did not confirm the hypothesis about the speed and significant advantage of the MSSQL database. The results showed the advantage of PostgreSQL over other databases.

Keywords: database system; CRM; Customer relationship management

Streszczenie

MSSQL, MySQL i PostgreSQL to jedne z najpopularniejszych systemów baz danych. Dobór bazy do stworzenia systemu CRM opiera się głównie na ocenie jej efektywności pod względem szybkości. Artykuł ma za zadanie przedstawić wybór optymalnego systemu baz danych do stworzenia efektywnego systemu CRM. Przegląd literatury skłonił do postawienia hipotezy, że MSSQL będzie najszybszy. Za badanie posłużyła seria eksperymentów z użyciem aplikacji. Podczas badań przeprowadzono serię eksperymentów z użyciem modułu przetwarzania zamówień będącego częścią większego systemu CRM skierowanego do branży e-commerce. Każde zapytanie zostało zmierzone 10-krotnie, wyniki uśredniono. Wyniki badań nie potwierdziły hipotezy o szybkości i znacznej przewadze bazy MSSQL. Wyniki pokazały przewagę bazy PostgreSQL nad innymi bazami.

Słowa kluczowe: system baza danych; CRM; Zarządzanie relacjami z klientami

*Corresponding author

Email address: lukasz.szwalek@pollub.edu.pl (Ł. Szwałek)

1. Wstęp

Baza danych jest sposobem długoterminowego przechowywania informacji, które zostały wygenerowane przez systemy informatyczne. Jest to jeden z podstawowych składników większości aplikacji lub programów. Dzięki obecności bazy danych możliwe staje się realizowanie podstawowych zadań, takich jak przechowywanie, przetwarzanie i pobieranie danych; jest to podstawa każdej aplikacji zorientowanej na dane.

CRM lub Systemy Zarządzania Relacjami z Klientami [1] charakteryzują się przetwarzaniem i generowaniem dużych ilości danych, integracją kilku systemów w jedną całość oraz kompatybilnością z innymi systemami. Gromadzenie danych ma na celu usprawnienie i automatyzację procesów sprzedaży, zarządzanie relacjami z klientami i planowanie zasobów przedsiębiorstwa. W bardziej zautomatyzowanych ustawieniach biznesowych transakcje dla całego przedsiębiorstwa są rejestrowane przez jeden system, a następnie mogą być integrowane z innymi systemami, takimi jak zasoby ludzkie, finanse i marketing. CRM to kluczowe narzędzie, dzięki któremu organizacje mogą czerpać korzyści ze zwiększonej produktywności, rentowności i przewagi konkurencyjnej. CRM jest szeroko stosowany przez firmy z branż m.in. telekomunikacyjnej, bankowej i finansowej oraz handlu detalicznego. Dane powinny być dostępne zawsze i wszędzie w przeciągu kilku, kilkunastu milisekund, dlatego tak ważne jest prawidłowe dobranie odpowiedniego silnika bazy danych. Aktualnie na rynku relacyjnych systemów zarządzania bazami danych jednymi z najpopularniejszych są MySQL, Microsoft SQL Server oraz PostgreSQL. Nierelacyjne systemy nie zostały uwzględnione w tym porównaniu ze względu na specyfikę CRM, gdzie struktura tabel i relacje między nimi są z góry ustalone.

2. Cel i zakres badań

Celem artykułu jest analiza wydajności trzech wybranych systemów bazodanowych w kontekście systemu CRM. Porównanie wyników tej analizy pozwoli określić, który system bazodanowy najlepiej współpracuje z tego typu systemami. Badania zostaną przeprowadzone na serii scenariuszy, które reprezentują typowe przypadki użycia. Przeprowadzane badanie będzie wykonywane przy użyciu programu Postman [2] umożliwiającego sprawdzenie czasów odpowiedzi API systemu CRM. Dzięki powyższemu rozwiązaniu możliwa będzie weryfikacja szybkości wykonywania zapytań i obserwacja wykorzystywanych zasobów przez poszczególne bazy danych. Porównanie to pomoże w wyborze konkretnego systemu bazodanowego, który najlepiej odpo-

wiada wybranemu zastosowaniu. Badania zostały oparte na aplikacji wykorzystującej bibliotekę Doctrine [3] oraz na badanych bazach danych. Ze względu na ograniczenia sprzętowe i zasobowe badanie zostało ograniczone do analizy wydajności baz danych. W niniejszym artykule, została sformułowana następująca teza badawcza:

T1: Baza danych oparta o MSSQL jest najwydajniejsza spośród testowanych systemów.

Prawdziwość zaprezentowanej tezy zostanie sprawdzona na podstawie wykonanych badań.

3. Przegląd literatury

W artykule „Performance analysis of NoSQL and relational databases with MongoDB and MySQL" [4] autorstwa Benymol'a Jose'a i Sajimon'a Abraham'a przedstawiono zalety baz nierelacyjnych względem baz relacyjnych, wyjaśniono różnice w czasie wykonywania operacji. Badania zostały przeprowadzone poprzez wykonywanie zapytań różnego typu za pomocą programów MySQL Workbench oraz MangoDB studio3T. Podsumowując informacje zwarte w artykule należy wskazać, iż testy zostały przeprowadzone na bazach zorientowanych na przechowywanie danych w postaci dokumentów bez zdefiniowanej struktury danych. Autorzy wykazali, że do baz przechowujących duże ilości danych w formie dokumentów warto stosować nierelacyjne bazy danych.

Kolejną pracą naukową poświęconą opisywanej tematyce jest „Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization" [5] autorstwa Rafała Kleweka, Wojciecha Truszkowskiego, Marii Skublewska-Paszkowskiej. Autorzy oceniali wydajność czterech relacyjnych baz danych: Oracle, MSSQL, MySQL oraz PostgreSQL. Bazy były testowane za pomocą aplikacji webowej umożliwiającej tworzenie i rozwiązywanie testów wielokrotnego wyboru. Testowano podstawowe operatory języka SQL, a także klauzulę WHERE w kilku wariantach. Postawiono hipotezę, że silnik Oracle będzie najszybszy. Przeprowadzone testy wykazały słuszność tej tezy. We wnioskach zostały zawarte następujące konkluzje:

- Bez względu na rozmiar bazy silnik Oracle osiągał porównywalne czasy.
- Silnik Oracle nagorzej radził sobie z zapytaniami zawierającymi podzapytania.

W artykule „Efficiency of databases in Django-based applications" [6] autorstwa Bartosza Nejmana i Beaty Pańczyk zostało przedstawione porównanie trzech systemów bazodanowych: MySQL, PostgreSQL oraz MongoDB. Autorzy przeprowadzili testy tworzenia, pobierania i warunkowego pobierania danych z testowanych baz danych przy pomocy aplikacji stworzonej na szkielecie Django [7]. We wnioskach artykułu zostały zawarte następujące stwierdzenia:

- Silnik MangoDB nie nadaje się do współpracy ze szkieletem Django ze względu na potrzebę zastosowania zewnętrznej biblioteki do przetwarzania zapytań ORM na składnię MangoDB.
- Z operacjami zapisu danych lepiej radzi sobie baza MySQL, natomiast z odczytem PostgreSQL.

Artykuł „Analiza wydajności relacyjnych baz danych Oracle oraz MSSQL na podstawie aplikacji desktopowej" [8], której autorami są Grzegorz Dziewit, Jakub Korczyński oraz Maria Skublewska-Paszkowska, zawiera porównanie wydajności baz danych Oracle i MS SQL w aplikacjach desktopowych. Artykuł zawiera informacje dotyczące metody wykorzystywanej podczas porównywania relacyjnych systemów bazodanowych w zakresie średniego czasu wykonywania poszczególnych zapytań. System opracowany przez badaczy umożliwia określenie, który system jest lepszy w zależności od funkcjonalności aplikacji. Autorzy w artykule udowodnili, iż baza danych MS SQL lepiej wykonuje operacje INSERT oraz UPDATE niż baza Oracle, natomiast w przypadku zapytań SELECT: krótszy czas wykonania uzyskał system Oracle.

Istnieje szereg badań ukazujących różnicę pomiędzy popularnymi bazami relacyjnymi jak i nierelacyjnymi. Jednak każdy omawiany artykuł bada w inny sposób - wykorzystując dedykowane narzędzia do każdej z baz, przez co wyniki mogą być zaburzone - lub przez pełnoprawne aplikacje webowe, gdzie na końcowy czas składa się także generowanie widoku aplikacji wraz z danymi. Powyższe uwagi przyczyniły się do stworzenia testów, w których mniejsze znaczenie mają czynniki środowiskowe.

4. Aplikacja testowa

Do przeprowadzenia badań stworzona została aplikacja typu REST API [9], której podstawowym założeniem jest możliwość odczytywania, aktualizacji i zapisywania danych. Do tego celu powstały trzy endpointy (Listing 1), po jednym dla każdej z operacji. Aplikacja została stworzona w oparciu o szkielet aplikacji Symfony [10].

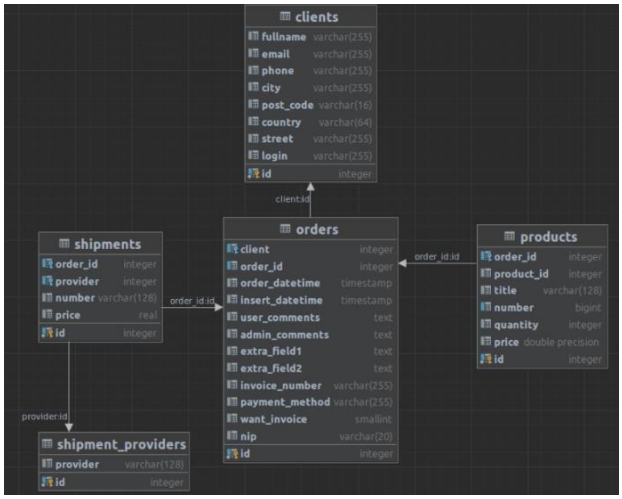
Listing 1: Endpointy API

```
getOrders:
  path: /api/getOrders
  controller: App\Controller\DefaultController::getOrders
  methods: [GET]

insertOrders:
  path: /api/insertOrders
  controller: App\Controller\DefaultController::insertOrders
  methods: [POST]

updateOrders:
  path: /api/updateOrders
  controller: App\Controller\DefaultController::updateOrders
  methods: [ PUT ]
```

Dzięki zastosowaniu biblioteki Doctrine w jednej aplikacji za pomocą zmiany jedynie konfiguracji ten sam kod źródłowy jest wstanie obsłużyć wszystkie testowane bazy danych. Testowane bazy danych zostały przygotowane w trzech silnikach bazodanowych:

- MySQL ver.8.0.29,
- PostgreSQL ver. 14.4,
- Microsoft SQL Server ver.2019-latest.

Każda z nich została napełniona tymi samymi danymi. Aby wyeliminować czynnik środowiskowy każda baza danych została uruchomiona w oddzielnym kontenerze Docker [11]. Każda baza została stworzona w oparciu o oficjalny obraz, bez modyfikacji zmiennych środowiskowych. Schemat bazy danych został zaprezentowany na Rysunku 1.



Rysunek 1: Schemat bazy danych.

5. Metoda badawcza

Eksperyment polegał na przetestowaniu i uruchomieniu testów przygotowanych w programie Postman. Testy zostały wykonane dla trzech scenariuszy, w których zostały wykorzystane polecenia SELECT, INSERT, UPDATE języka SQL:

- odczyt z bazy (SELECT),
- zapis do bazy (INSERT),
- modyfikacja danych (UPDATE).

Każdy scenariusz został wykonany w trzech seriach. Każda seria badała inny rozmiar zapytania, ilość danych w tabelach była za każdym razem taka sama i została przedstawiona w Tabeli 1. Zapytania były tworzone dla 10, 100 i 1000 wierszy z tabeli orders.

Tabela 1: Liczba wierszy w poszczególnych tabelach

Tabela	Ilość danych
orders	2 000 000
clients	1 750 000
products	3 300 000
shipments	2 000 000
shipment_providers	100

Testy były uruchamiane na systemie Ubuntu 20.04 LTS, a bazy danych w kontenerach platformy Docker w wersji 20.10. W Tabeli 2 została przedstawiona specyfikacja maszyny, na której zostały przeprowadzone testy. W trakcie wykonywania testów kontenery zawierające bazy danych były monitorowane za pomocą komendy „docker stats", która wyświetla takie dane jak: wykorzystanie procesora, pamięci RAM.
Sposób realizacji każdego z testów przez kod źródłowy interfejsu REST API został przedstawiony w poniższych podpunktach.

Tabela 2: Specyfikacja urządzenia testowego

Procesor	Procesor Intel(R) Core(TM) i7-7700HQ
Dysk	512GB SSD M.2
RAM	16GB 2133MHz

5.1. Odczyt z bazy

Listing 2 przedstawia funkcję odpowiadającą za testowanie polecenia SELECT. Przyjmuje dwa parametry pozwalające manipulować wielkością otrzymywanych wyników. Zapytanie łączy dwie tabele za pomocą operacji LEFT JOIN. Dla potwierdzenia zwracana jest jedynie liczba wierszy wygenerowanych przez zapytanie.

Listing 2: Ciało funkcji getOrders

```
public function getOrders(Request $request): Response
{
    $offset = $request->query->get('offset');
    $limit = $request->query->get('limit');
    $query = $this->em->createQueryBuilder()->select(['o', 'p'])
        ->from(Orders::class, 'o')
        ->leftJoin(Products::class, 'p', 'WITH', 'p.order = o.id')
        ->setFirstResult($offset)
        ->setMaxResults($limit);
    $results = $query->getQuery()->getResult();

    return new Response(count($results), 200);
}
```

5.2. Zapis do bazy

Funkcja przedstawiona na Listingu 3 odpowiada za testowanie polecenia INSERT. Parametrem wejściowym jest tablica zamówień w zawierająca dane zamówienia, zamawiającego i produktów. Ta funkcja testuje równoczesny i wielokrotny zapis do trzech tabel: orders, clients, products.

Listing 3: Ciało funkcji insertOrders

```
public function insertOrders(Request $request): Response
{
    $orders = json_decode($request->getContent(), true);
    foreach ($orders['orders'] as $orderData) {
        $client = $this->createClientObject($orderData);
        $this->em->persist($client);
        $this->em->flush();
        $order = $this->createOrderObject($orderData, $client);
        $this->em->persist($order);
        $this->em->flush();
        foreach ($orderData['products'] as $productData) {
            $product = $this->createProductObject($productData, $order);
            $this->em->persist($product);
        }
        $this->em->flush();
    }

    return new Response(count($orders['orders']), 200);
}
```

5.3. Modyfikacja danych

Listing 4 przedstawia funkcję odpowiadającą za aktualizowanie danych. Tak jak w przypadku odczytu: przyjmuje dwa parametry umożliwiające manipulację liczbą edytowanych danych. Funkcja pobiera wskazaną w teście liczbę wierszy. Następnie modyfikowana jest

jedna kolumna z tabeli orders - extra_fields2 - przechowująca zmienne tekstowe, a następnie jedna kolumna o nazwie quantity – przechowującą liczbę produktów, z tabeli products dla wszystkich powiązanych wierszy.

Test polecenia SELECT i UPDATE został wykonany dla trzech wielkości zapytań 10, 100, 1000 wierszy. Test polecenia UPDATE polegał na zapisaniu do bazy 10, 100, 1000 zamówień. Każdy z testów polegał na dziesięciokrotnym wykonaniu danego scenariusza w celu uzyskania możliwie jak najbardziej realnych wyników. Program Postman pozwala na przygotowanie i zautomatyzowanie testów, dzięki czemu każdy test był wykonany w jednakowy sposób. Czasy wykonania każdego z testu zostały odczytane z konsoli programu i uśrednione dla każdego wariantu w serii.

Listing 4: Ciało funkcji updateOrders

```
public function updateOrders(Request $request): Response
{
    $offset = $request->query->get('offset');
    $limit = $request->query->get('limit');
    $orders = $this->em->getRepository(Orders::class)
        ->findBy([], null, $limit, $offset);
    foreach ($orders as $order) {
        $order->setExtraField2($order->getOrderId());
        $products = $this->em->getRepository(Products::class)
            ->findBy(['order' => $order]);
        foreach ($products as $product) {
            $product->setQuantity(99);
        }
    }
    $this->em->flush();

    return new Response(count($orders), 200);
}
```

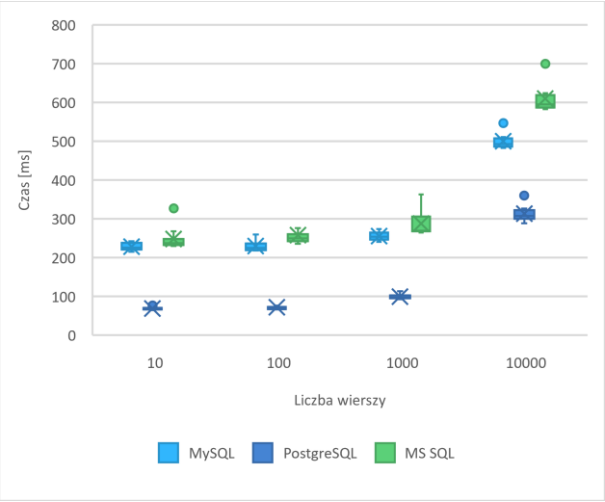
6. Analiza wyników

Przed rozpoczęciem badań został wykonany pomiar użycia zasobów przez bazy nie wykonywały żadnych operacji. Wynik obserwacji przedstawiony został w Tabeli 3.

Tabela 3: Zużycie zasobów w trakcie spoczynku

MySQL		PostgreSQL		MSSQL	
CPU [%]	RAM [MB]	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]
0,34	382	0,1	34	2,1	1167

W pierwszym badanym scenariuszu sprawdzany był czas wykonywania instrukcji z poleceniem *SELECT*, wyniki zostały przedstawione na Rysunku 2 a uśrednione wyniki serii w Tabeli 4 natomiast zużycie zasobów zostało przedstawione w Tabeli 5. Porównując poszczególne serie zapytań widać bardzo niewielką różnicę w czasie wykonania, dlatego dla tego scenariusza został wykonany dodatkowo test dla 10000 wierszy. Porównując silniki w każdej serii znaczącą przewagę ma baza oparta na silniku PostgreSQL.



Rysunek 2: Wyniki testu funkcji getOrders.

Tabela 4: Średnie wyniki testu funkcji getOrders

Liczba wierszy	MySQL		PostgreSQL		MSSQL	
	Średni czas [ms]	Std	Średni czas [ms]	Std	Średni czas [ms]	Std
10	228,20	9,0	68,6	3,6	248,2	29,7
100	229,70	13,5	72,4	8,5	258,3	27,4
1000	255,30	10,3	99	6,0	287,7	319
10000	499,9	18,3	313,2	19,9	609,9	34,9

Tabela 5: Zużycie zasobów w trakcie testu funkcji getOrders

Liczba wierszy	MySQL		PostgreSQL		MSSQL	
	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]
10	91,9	541	73,6	95	93,6	1225
100	89,7	541	70,1	95	93,1	1225
1000	80,5	541	55,9	95	84,6	1225
10000	49,7	541	23,3	95	58,3	1225

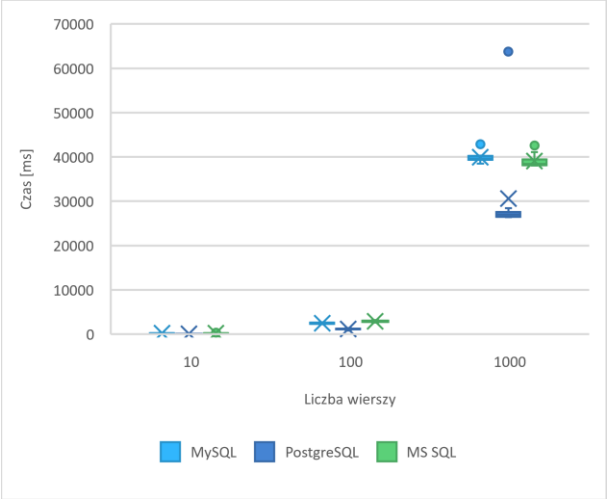
W następnym badaniu testującym scenariusz z poleceniem INSERT widać znacząco zwiększający się czas wykonania serii wraz ze zwiększającą się liczbą danych do zapisania, widać to zarówno na wykresie (Rysunek 3) jak i Tabeli 6 z uśrednionymi wartościami każdej serii. Natomiast z obserwacja zużycia zasobów (Tabela 7) pokazuje spadek użycia CPU i równoczesny przyrost użycia pamięci RAM wraz ze zwiększaniem liczby wierszy w teście. Porównując silniki ponownie dominuje PostgreSQL, jedynie przy ostatniej serii czasy wykonania były bardzo zbliżone.

Tabela 6: Średnie wyniki testu funkcji insertOrders

Liczba wierszy	MySQL		PostgreSQL		MSSQL	
	Średni czas [ms]	Std	Średni czas [ms]	Std	Średni czas [ms]	Std
10	222,6	43,1	85,1	16,9	201,2	63,4
100	2475,6	145,1	1173,4	177,7	2930,3	184,6
1000	39932,3	1183,8	30642,5	11664,9	39060,7	1512,5

Tabela 7: Zużycie zasobów w trakcie testu funkcji insertOrders

Liczba wierszy	MySQL		PostgreSQL		MSSQL	
	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]
10	22,4	536	28,4	103	60,3	1229
100	21,2	536	24,5	106	63,4	1251
1000	12,5	536	8,2	110	34,4	1255



Rysunek 3: Wyniki testu funkcji insertOrders.

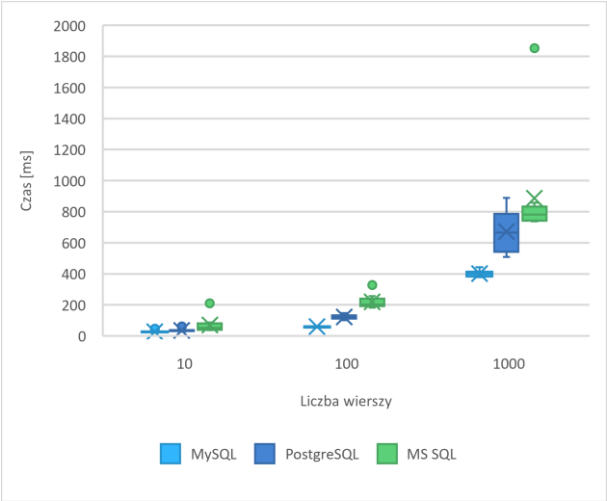
W ostatnim ze scenariuszy testowano polecenie UPDATE. Wyniki przedstawione na Rysunku 4 pokazują duży rozrzut między pomiarami w serii 1000 wierszy dla bazy PostreSQL. Tabela 8 przedstawia uśrednione wyniki serii, które także pokazują przyrost czasu między seriami. Jednocześnie wraz ze zwiększaniem liczby wierszy rosło wykorzystanie CPU i pamięci RAM dla każdej z baz (Tabela 9). W tym scenariuszu w każdej serii najniższe czasy osiągał silnik MySQL z niewielką przewagą nad silnikiem PostgreSQL jak i MSSQL.

Tabela 8: Średnie testu funkcji updateOrders

Liczba wierszy	MySQL		PostgreSQL		MSSQL	
	Średni czas [ms]	Std	Średni czas [ms]	Std	Średni czas [ms]	Std
10	27,7	7,4	35,6	9,8	69,8	52,3
100	59,5	6,6	122,6	13,4	219,1	44,4
1000	401	19,8	671,7	130,2	886,9	341,6

Tabela 9: Zużycie zasobów w trakcie testu funkcji updateOrders

Liczba wierszy	MySQL		PostgreSQL		MSSQL	
	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]	CPU [%]	RAM [MB]
10	35,4	522	33,4	347	47,1	1226
100	47,9	523	54,8	379	69,0	1259
1000	73,6	527	68,4	453	84,5	1318



Rysunek 4: Wyniki testu funkcji updateOrders.

7. Wnioski

W niniejszym artykule przeprowadzone zostały badania wydajności trzech najpopularniejszych baz danych dostępnych na rynku, czyli MySQL, PostgreSQL oraz MSSQL. Dla zapewnienia neutralnych warunków każdy z silników bazodanowych został uruchomiony w oddzielnym kontenerze oprogramowania Docker z wykorzystaniem oficjalnych, dedykowanych obrazów. Analizując wyniki przyprowadzonych badań jedynie w przypadku testu sprawdzającego wydajność wykonywania polecenia SELECT można zaobserwować znaczącą przewagę bazy danych opartej o silnik PostgreSQL. Przewaga jest widoczna zwłaszcza w pierwszych trzech seriach, gdzie testy bazy PostgreSQL uzyskiwały trzykrotnie krótsze czasy w porównaniu do pozostałych testowanych baz. W pozostałych dwóch testach poleceń INSERT i UPDATE nie została odnotowana tak znacząca przewaga którejkolwiek z testowanych baz. W teście polecenia INSERT także najlepsze wyniki osiągnął PostgreSQL lecz wraz ze wzrostem liczby przetwarzanych wierszy czasy każdej z baz były zbliżone. W ostatnim teście z poleceniem UPDATE z niewielką przewagą najszybciej poradził sobie silnik MySQL. Pod względem zużycia zasobów maszyny, baza oparta o silnik PostgreSQL także wypada najlepiej spośród testowanych. Bez obciążenia baza zużywa znikomy procent CPU jak i znikomą ilość pamięci RAM. Podczas wykonywania testów wartości te znacząco wzrastają, lecz nie osiągają wartości obserwowanych na pozostałych bazach. Teza, która została postawiona w artykule: „Baza danych oparta o MSSQL jest najwydajniejsza spośród testowanych systemów” zosta-

ła obalona. W wyniku analizy badań przeprowadzonych na interfejsie REST API stworzonym na szkielecie aplikacji Symfony, wykorzystującym bibliotekę Doctrine, najbardziej wydajnym systemem okazał się PostgreSQL. Jednakże w przypadku badań polecenia SELECT i INSERT zużycie procesora spada wraz ze wzrostem liczby wierszy co świadczy o większym zapotrzebowaniu na moc obliczeniową przez aplikację REST API. Jest to znak, iż badania symulujące większe systemy powinny być wykonywane na więcej niż jednej maszynie, aby zapewnić takie sam dostęp do mocy obliczeniowej każdemu elementowi podlegającemu badaniu.

Wydajność baz danych jest niezwykle istotna zwłaszcza w przypadku przechowywania gigabajtów danych w postaci relacyjnych baz. Z testu polecenia INSERT i UPDATE wynika jasno, że różnice w czasach wykonania dla poszczególnych baz są niewielkie. Warto również zaznaczyć, że wraz ze wzrostem wielkości zapytań i ilości przetwarzanych danych czasy wykonania wzrastają. Nawiązując do wstępu, system CRM ma usprawniać pracę z danymi, dlatego zapytania trwające prawie 40 sekund - w przypadku zapytania zapisującego 10 000 zamówień - zbliża się do bariery „przepływu myśli użytkownika" [12], nie uwzględniając czasu potrzebnego na przesłanie danych między serwerem a użytkownikiem i czasu potrzebnego do wygenerowania widoku. Biorąc pod uwagę powyższe stwierdzenie nasuwa się wniosek, iż na czasy wykonania zapytań składają się nie tylko wydajność bazy, ale także optymalizacja wykonywanych zapytań poprzez zmniejszenie wyników za pomocą klauzuli LIMIT i paginacji wyników za pomocą klauzuli OFFSET.

Literatura

[1] Czym jest system CRM, https://www.oracle.com/pl/cx/what-is-crm/, [2014-02-19].

[2] Dokumentacja Postman, https://learning.postman.com/docs/getting-started/introduction/, [2022-07-05].

[3] Dokumentacja biblioteki Doctrine, https://www.doctrine-project.org/, [2019-06-24].

[4] B. Jose, S. Abraham, Performance analysis of NoSQL and relational databases with MongoDB and MySQL, Materials Today: Proceedings 24, (2020) 2036-2043, https://doi.org/10.1016/j.matpr.2020.03.634.

[5] W. Truskowski, R. Klewek, M. Skublewska-Paszkowska, Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization, Journal of Computer Sciences Institute 16 (2020) 279-284, https://doi.org/10.35784/jcsi.2026.

[6] B. Nejman, B. Pańczyk, Efficiency of databases in Django-based applications, Journal of Computer Sciences Institute 11 (2019) 82-85, https://doi.org/10.35784/jcsi.142.

[7] Dokumentacja Django, https://docs.djangoproject.com/en/4.1/, [02-08-2022].

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

[9] REST API, https://www.ibm.com/pl-pl/cloud/learn/rest-apis, [06-04-2021].

[10] Dokumentacja Symfony, https://symfony.com/doc/5.4/index.html, [09-12-2021].

[11] Dokumentacja Docker, https://docs.docker.com/compose/, [2021-05-10].

[12] J. Nielsen, Usability Engineering, AP Professional, San Francisco, 1993.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 54–62
Received: 26 September 2022
Accepted: 10 October 2022

Analysis of the usability and accessibility of public transport online timetables in selected cities in Poland

Analiza użyteczności i dostępności internetowych rozkładów jazdy komunikacji miejskiej w wybranych miastach w Polsce

Piotr Wójtowicz*, Mariusz Dzieńkowski

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

Abstract
The purpose of the study was to verify various aspects related to the accessibility and usability of the timetable websites of two companies providing public transportation services. The issues addressed mainly concerned the graphic form of accessibility tools, the way of accessing these tools, their location in the site, the use of alternative names and descriptions. The objects of the research were two existing websites of two people transportation companies. In addition, two prototype sites were made for the purposes of the research. An experiment was prepared and conducted with 13 participants. A research method using the eye tracking technique was applied. The results were obtained in the form of heat maps, scan paths, task completion times, task performance ratings and the number of fixations on the object of interest. After analyzing the collected results, the following conclusions were drawn: the use of similar graphic symbols often confuses users; indirect access to accessibility tools generally makes it difficult for users to use them; the ill-conceived design of some page elements creates problems for content viewers; experimenting with the placement of some tools causes difficulty in finding them; a great deal of confusion on websites is introduced by unusual names, links, menu options; a large amount of content and the lack of prominence of what is most important cause problems in finding the right information; the lack of captioning of graphics often causes users to have to think longer and check if what they want to use is the right option; placing an item/information at the bottom of the page generally extends the time to reach it.

Keywords: usability; accessibility; websites analysis; eye tracking

Streszczenie
Celem pracy była weryfikacja różnych aspektów związanych z dostępnością i użytecznością serwisów internetowych zawierających rozkład jazdy dwóch firm realizujących publiczne usługi komunikacyjne. Podjęte kwestie dotyczyły przede wszystkim formy graficznej narzędzi dostępności, sposobu dostępu do tych narzędzi, umiejscowienia ich w witrynie, zastosowania alternatywnych nazw i opisów. Obiektami badań były dwa istniejące serwisy www dwóch firm przewozowych ludzi. Dodatkowo dla celów badań wykonano dwa prototypowe serwisy. Przygotowano i przeprowadzono eksperyment, w którym wzięło udział 13 osób. Zastosowano metodę badawczą wykorzystującą technikę eye-trackingową. Uzyskano wyniki w postaci map cieplnych, ścieżek skanowania, czasów realizacji zadań, ocen wykonania zadań oraz liczby fiksacji na obiekcie zainteresowania. Po przeprowadzeniu analiz zebranych wyników wyciągnięto następujące wnioski: użycie podobnych symboli graficznych często wprowadza w błąd użytkowników; niebezpośredni dostęp do narzędzi dostępności na ogół utrudnia użytkownikom z nich skorzystanie; nieprzemyślany wygląd niektórych elementów strony stwarza problemy dla odbiorców treści; eksperymentowanie z umiejscowieniem niektórych narzędzi powoduje trudności z ich znalezieniem; dużą dezorientację na stronach internetowych wprowadzają nietypowe nazwy, linki, opcje menu; duża ilość treści i brak wyeksponowania tego co najważniejsze powoduje problemy ze znalezieniem właściwej informacji; brak podpisów grafiki powoduje, że użytkownicy często muszą się dłużej zastanowić i sprawdzić czy to czego chcą użyć jest właściwą opcją; umieszczenie elementu/informacji na dole strony na ogół wydłuża czas dotarcia do nich.

Słowa kluczowe: użyteczność; dostępność; analiza stron internetowych; eyetracking

*Corresponding author
Email address: piotr.wojtowicz9@pollub.edu.pl (P. M. Wójtowicz)

1. Wstęp

W dzisiejszych czasach, ludzie coraz częściej podróżując, korzystają z możliwości wyszukiwania informacji przy pomocy Internetu. Pomaga to im szybko odnaleźć się w sytuacji, zdobyć niezbędne informacje, gdy inne źródła są w danej chwili niedostępne. Na przykład, gdy zdarzy się potrzeba skorzystania z transportu autobusowego, zaistnieje potrzeba sprawdzenia rozkładu jazdy. Wówczas można to szybko zrobić korzystając z internetowej lub mobilnej wersji rozkładu – używając do tego komputera lub smartfona albo osobiście wybrać się na przystanek. To ostatnie rozwiązanie wymaga czasu, a po przybyciu na przystanek może okazać się, że rozkładu nie ma lub jest trudny do odczytania. Z tych powodów informacje dotyczące kursów autobusów są obecnie najczęściej pozyskiwane przez osoby korzysta-

jące z transportu publicznego za pośrednictwem Internetu. Dzięki temu potencjalni pasażerowie komunikacji mogą w szybki sposób sprawdzić potrzebne im informacje, a także dowiedzieć się więcej na temat zakupu biletów, zniżek, tras autobusów czy linii obsługiwanych przez pojazdy niskopodwoziowe. Często w przypadku serwisów internetowych oferowanych przez przewoźników publicznych, sposób przedstawienia treści stanowi utrudnienie w poszukiwaniu potrzebnych danych. Najczęstszym problemem jest samo znalezienie niezbędnych informacji. Wynika to z tego, że interfejsy stron są mało czytelne, są przeładowane informacjami, nie eksponują najważniejszych elementów, nie posiadają udogodnień i narzędzi umożliwiających dostęp do nich dla osób starszych czy niepełnosprawnych. Serwisy tego typu, które są przeznaczone dla szerokiej grupy użytkowników, powinny być inteligentnie projektowane, powinny mieć przemyślaną strukturę i wspierać różne rozwiązania także dla osób z różnymi niepełnosprawnościami. Witryny takie powinny być użyteczne, proste w obsłudze, przejrzyste i ascetyczne. Dlatego też strony z takim przeznaczeniem powinny być koniecznie testowane i oceniane pod względem jakości, użyteczności, dostępności, satysfakcji i doświadczenia użytkownika. Badania powinny być więc realizowane już na etapie projektowania i implementacji, a następnie także po wdrożeniu i udostępnieniu informacji w Internecie.

Wśród wielu sposobów testowania serwisów internetowych, w ostatnich latach na popularności zyskuje technika eyetrackingowa. Umożliwia ona analizę stanów oczu osób badanych: zatrzymań (fiksacji), szybkich ruchów (sakad), zmian średnicy źrenicy oraz mrugnięć. Technika ta pozwala na przeprowadzenie zarówno analizy jakościowej – dokonywanej na bazie map cieplnych i ścieżek skanowania, a także analizy ilościowej realizowanej na podstawie danych liczbowych. W ramach tej pracy wykonano obie analizy na dwóch istniejących serwisach internetowych oraz dwóch prototypowych odpowiednikach, które zostały opracowane z uwzględnieniem aspektów użyteczności i dostępności. Wszystkie analizowane serwisy zawierały rozkłady jazdy komunikacji miejskiej z wybranych miast. W badaniach wykorzystano wyłącznie technikę eyetrackingową.

2. Przegląd literatury

Strony i aplikacje internetowe muszą obecnie zawierać zestaw funkcjonalności związanych z zapewnieniem dostępności dla szerokiej grupy użytkowników, a szczególnie osób z różnymi niepełnosprawnościami [1].

Dlatego konieczne są narzędzia umożliwiające nietypowe sposoby obsługiwania komputera i serwisów internetowych. Bardzo dużym wyzwaniem jest sprawienie, żeby system czy strona była dostępna dla wszystkich rodzajów niepełnosprawności. Ważne jest także, aby aplikacje i serwisy były użyteczne, proste i łatwe w obsłudze. Elementy te trzeba kontrolować od momentu projektowania, w czasie implementacji oraz po wdrożeniu.

Istnieje wiele metod badania użyteczności i dostępności. W pracy [2] autorzy wykorzystali trzy narzędzia automatyczne (walidatory: Utilitia, Tingun Page Checker, Google PageSpeed Insights) oraz listę kontrolną składającą się z ośmiu kryteriów oceniających responsywność, możliwość zmiany kontrastu, rozmiaru czcionki, wyboru języka, umożliwienie opcji drukowania i wyświetlania w formacie PDF, posiadanie funkcji lektora, wyszukiwarki oraz dokumentów wymaganych przez prawo. Do oceny serwisów pod względem dostępności wykorzystano dwa wskaźniki i ustalono dla nich progi, po przekroczeniu których dana strona uważana była jako dostępna. W badaniach wzięto pod uwagę 190 serwisów www urzędów gmin województwa lubelskiego. Okazało się, że tylko 33 serwisy gmin posiadają strony przystosowane do potrzeb osób niepełnosprawnych.

W artykule [3] została dokonana ocena serwisów trzech uczelni wyższych w Polsce: Katolickiego Uniwersytetu Lubelskiego, Politechniki Krakowskiej oraz Zachodniopomorskiego Uniwersytetu Technicznego. Do badań wykorzystano dwie metody: kwestionariuszową – sprawdzającą satysfakcję użytkowników podczas interakcji z wybranymi serwisami oraz wykorzystującą technikę eyetrackingową. Podczas analizy danych eyetrackingowych posłużono się mapami cieplnymi i ścieżkami skanowania oraz miarami liczbowymi takimi jak czas do pierwszej fiksacji na obiekcie zainteresowania, czas realizacji zadania i liczba fiksacji. Dodatkowo nagrania sesji użytkowników były przeglądane i oceniane przez ekspertów pod kątem poprawności wykonania zadań.

W ramach niniejszej pracy do analiz także wykorzystano wyniki w postaci map cieplnych i ścieżek skanowania oraz miary takie jak czas realizacji zadań, poprawność ich wykonania, liczba wizyt w obszarze zainteresowania oraz liczba fiksacji w obszarze zainteresowania. Ostatnia miara może być interpretowana na różne sposoby. Duża liczba fiksacji w obszarze zainteresowania z jednej strony może wskazywać na wysoki poziom zainteresowania obiektem lub obszarem [4]. Z drugiej jednak strony może być oznaką trudności w zrozumieniu, rozproszeniu myśli oraz dezorientacji. Poza tym większa liczba fiksacji w określonym AOI (ang. Area of Interest) może wskazywać na to, że ten AOI jest ważniejszy i/lub bardziej zauważalny przez użytkownika od innych [5]. Miara ta może także wskazywać na to, że wyszukiwanie było nieefektywne [6]. Natomiast miara jaką jest liczba wizyt określa liczbę uczestników badań, którzy skupili się na określonym AOI. Jeśli niektórzy badani przegapili dane AOI i nie skupili się na nim, prawdopodobnie oznacza to, że albo ten obszar był trudny do zauważenia (np. nie był wyraźny w porównaniu z innymi obszarami na ekranie), albo uczestnicy nie szukali go, ponieważ prawdopodobnie zawierał informacje, które uznali za nieistotne.

3. Cel i zakres pracy

Celem pracy jest weryfikacja różnych aspektów związanych z dostępnością i użytecznością serwisów internetowych ogólnego przeznaczenia, zawierających rozkłady jazdy autobusów komunikacji miejskiej. Podjęte kwestie dotyczyły przede wszystkim formy graficznej narzędzi dostępności, sposobu dostępu do tych narzędzi (pośredni/bezpośredni), umiejscowienia na witrynie, zastosowania alternatywnych nazw i opisów. Do celów badawczych wybrano dwa funkcjonujące serwisy www. Przy wyborze tych stron kierowano się pochodzeniem firm świadczących usługi komunikacyjne oraz niedomaganiami ich serwisów w zakresie dostępności i użyteczności. Dla porównania przygotowano dwa podobne serwisy, będące odpowiednikami tych istniejących w Internecie, ale różniące się lokalizacją i wyglądem narzędzi dostępności, ważnych linków czy istotnych informacji. Ocena istniejących i zaproponowanych rozwiązań zarówno na stronach firm komunikacyjnych i prototypowych odpowiednikach została dokonana przy pomocy techniki eyetrackingowej. W eksperymencie wzięli udział użytkownicy, którzy wykonywali proste zadania, pracując i wchodząc w interakcje z przygotowanymi serwisami. Dla uproszczenia zostały wykonane zrzuty ekranowe wybranych stron, a osoby badane miały tylko odnaleźć pewne informacje tekstowe lub elementy graficzne powiązane z określonymi funkcjonalnościami. Sesje badawcze oraz aktywność oczna była rejestrowana przez eyetracker.

Zakres prac obejmował:

- analizę literatury dotyczącej podobnych badań eyetrackingowych, głównie w obszarze użyteczności i dostępności stron internetowych,
- wybór obiektów do badań – dwóch serwisów internetowych z głównym elementem, jakim był rozkład jazdy autobusów komunikacji miejskiej,
- wykonanie dwóch prototypowych serwisów uwzględniających kwestie użyteczności i dostępności,
- przygotowanie stanowiska badawczego,
- sformułowanie zadań do wykonania przez osoby uczestniczące w eksperymencie,
- przeprowadzenie pilotażu i korekta eksperymentu,
- przeprowadzenie badań i ich rejestracja,
- analizę wyników oraz sformułowanie wniosków.

Przed rozpoczęciem badań zostały postawione następujące pytania badawcze:

1. Czy użycie alternatywnych form graficznych dla narzędzi dostępności utrudni ich używanie?
2. Czy dobrym rozwiązaniem jest zgrupowanie narzędzi dla niepełnosprawnych w jednym miejscu, tak aby dostęp do nich odbywał się za pośrednictwem specjalnej niebieskiej ikony z rysunkiem osoby na wózku?
3. Czy zastosowanie niestandardowych wyglądów pól tekstowych oraz pominięcie podpowiedzi skomplikuje użytkownikom korzystanie z nich?
4. Czy umieszczenie funkcji dostępności w innych miejscach niż te zwyczajowo przyjęte spowolni użytkownikom dotarcie do nich?
5. Czy zastosowanie alternatywnych nazw, opcji menu, linków spowoduje dezorientację użytkowników i wydłuży im pracę?
6. Czy brak wyeksponowania ważnych treści przyczyni się do powstania problemów z ich odnalezieniem?
7. Czy forma tekstowa elementów dostępu do treści w serwisie jest lepsza od formy graficznej?
8. Czy lokalizacja elementów stron na dole wydłuża czas pracy użytkownika?

4. Metoda badawcza

Eksperyment przeprowadzany z wykorzystaniem techniki eyetrackingowej składał się z następujących etapów:

- przedstawienie uczestnikom celu badania i jego przebiegu,
- wyrażenie zgody przez uczestników na udział w badaniu i rejestrację ich działań,
- kalibracja urządzenia,
- wyświetlanie naprzemiennie treści poleceń i bodźców w postaci zrzutów ekranowych z badanych serwisów,
- zorganizowanie grupy badawczej,
- przeprowadzenie eksperymentu na grupie badawczej,
- weryfikacja i przetwarzanie zgromadzonych danych,
- analiza wyników za pomocą platformy iMotions.

4.1 Obiekty badań

Do przeprowadzenia badań zostały wybrane dwie strony internetowe zawierające informacje dotyczące autobusowych rozkładów jazdy. Głównym kryterium przy wyborze tych stron była lokalizacja firmy komunikacyjnej powiązanej z daną stroną internetową na terenie województwa lubelskiego. Drugim kryterium było posiadanie przez stronę www funkcjonalności dostępności. Do badań wybrano serwis MPK Lublin (rysunek 1) oraz serwis Chełmskich Linii Autobusowych (rysunek 2).

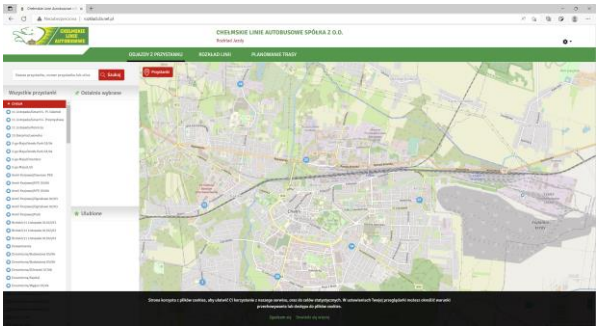
Do celów badań opracowano dwa serwisy będące odpowiednikami tych istniejących. Są to prototypowy serwis – Rozkład Jazdy Lublin (rysunek 3) oraz prototypowy serwis ChLA (rysunek 4). Witryny zostały wykonane za pomocą systemu CMS WordPress – najpopularniejszej platformy do zarządzania treścią. Daje ona szerokie możliwości kontrolowania treści, dzięki ogromnej liczbie motywów i wtyczek. Przykładem jednej z takich wtyczek ułatwiającej tworzenie stron jest wtyczka *„WP Accessibility”*, pozwalająca na szybkie dodanie zestawu funkcji dostępności [7].

Strony prototypowe są dostępne w Internecie dzięki wykorzystaniu darmowego hostingu (tabela 4). Strony zostały zaprojektowane w ten sposób, aby ich schemat graficzny różnił się od witryny istniejącej.

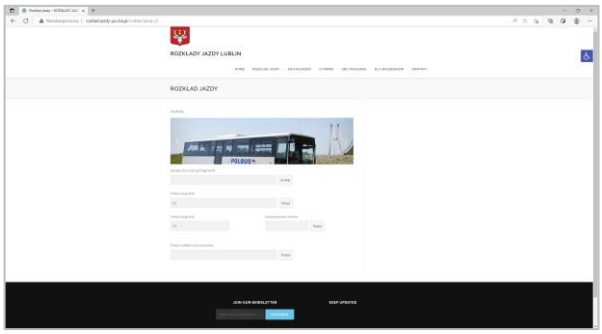
Tabela 1: Nazwy i adresy stron

Skrót	Użyta strona
S1	MPK Lublin – oryginał https://mpk.lublin.pl/
S2	MPK Lublin – prototyp http://rozklad-jazdy-pu.cba.pl/
S3	ChLA – oryginał https://www.cla.net.pl/page/357/trasy.html
S4	ChLA – prototyp http://rozklad-jazdy-pu2.cba.pl/trasy/

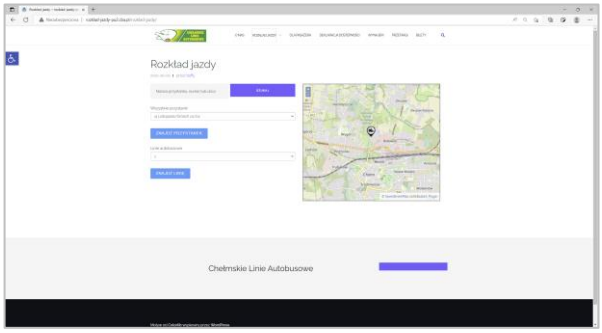
Rysunek 1: Widok oficjalnego serwisu MPK Lublin.



Rysunek 2: Widok oficjalnego serwisu ChLA.



Rysunek 3: Widok prototypowego serwisu Rozkład Jazdy Lublin.



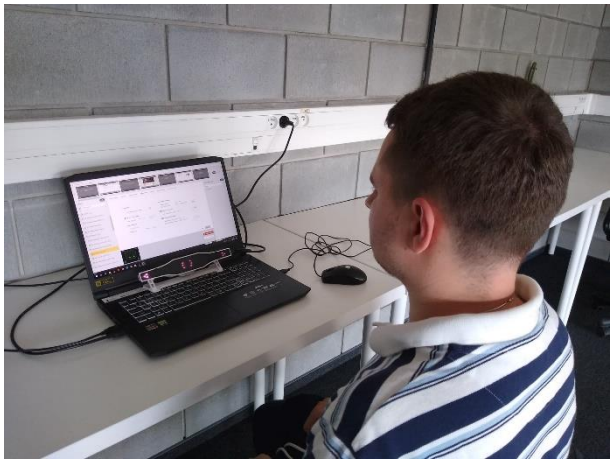
Rysunek 4: Widok prototypowego serwisu ChLA.

4.2 Grupa badawcza

W badaniu wzięło udział 13 osób (10 mężczyzn oraz 3 kobiety). Wiek badanych osób mieścił się w przedziale 23–27 lat. Byli to dawni studenci Politechniki Lubelskiej, studenci Uniwersytetu Medycznego w Lublinie oraz osoby pracujące. Z wywiadu przeprowadzonego przed eksperymentem, ustalono, że uczestnicy nie mieli wcześniej żadnego doświadczenia z dobranymi obiektami badań.

4.3 Stanowisko badawcze

Eksperyment został przeprowadzony w laboratorium należącym do Katedry Informatyki znajdującym się w Centrum Innowacji i Zaawansowanych Technologii Politechniki Lubelskiej. W pomieszczeniu zostały zapewnione odpowiednie warunki oświetleniowe do przeprowadzenia badań. Każdy z uczestników siedział na regulowanym krześle, zapewniającym właściwą pozycję uczestnika podczas badań. W każdej z przeprowadzonych sesji badawczych brał udział moderator, który wyjaśnił uczestnikom szczegóły badania oraz monitorował poprawny przebieg jego wykonania.



Rysunek 5: Przebieg badania eyetrackingowego.

Stanowisko badawcze pokazane na rysunku 3 jest wyposażone w eyetracker stacjonarny Gazepoint GP3 HD [8], którego szczegółową konfigurację przedstawia tabela 2.

Tabela 2: Specyfikacja eyetrackera

Częstotliwość próbkowania	60Hz lub 150Hz
Kalibracja	5-cio lub 9-cio punktowa
Detekcja stanów oka	fiksacji, sakad, zmian średnicy źrenicy
Technika śledzenia	jasna źrenica
Zakres swobody poruszania głową	35 cm w poziomie oraz 22 cm w pionie
Zakres odległości oczu od eyetrackera	~65cm
Dokładność w idealnych warunkach	0,5–1° obuocznie

Eyetracker jest podłączony do laptopa, którego specyfikacja znajduje się w tabeli 3. Na komputerze było zainstalowane oprogramowanie Gazepoint Control, które jest niezbędne do działania eyetrackera. Natomiast

do zaprojektowania eksperymentu zostało użyte oprogramowanie iMotions 9.0. Program ten pozwala na rejestrowanie sesji badawczych użytkowników biorących udział w eksperymencie, jak i wizualizację wyników w postaci: ścieżek skanowania, map cieplnych, map uwagowych i obszarów zainteresowania. iMotions pozwala również na generowanie statystyk i eksport danych [9].

Tabela 3: Specyfikacja laptopa

Model	ACER NITRO 5 AMD
Procesor	AMD Ryzen 7 5800H 3.20 GHz
Karta graficzna	NVIDIA Geforce RTX 3060
Pamięć RAM	32 GB
Monitor	Full HD 144 Hz
Rozdzielczość ekranu	1920 x 1080
Częstotliwość odświeżania	144 Hz
Przekątna ekranu	43,9 cm (17,3")
Typ wyświetlacza	LCD
System operacyjny	Windows 10 x64

4.4 Eksperyment

W ramach pracy przygotowano eksperyment, który składał się z następujących etapów:

1. Określenie celu badań, sformułowanie pytań badawczych.
2. Opracowanie zadań, które będą realizowane przez użytkowników.
3. Zaprojektowanie eksperymentu na platformie iMotions.
4. Przygotowanie stanowiska badawczego.
5. Przeprowadzenie badania pilotażowego i dopracowanie szczegółów eksperymentu.
6. Rejestracja użytkowników podczas realizacji zadań.
7. Eksport wyników i analiza danych.
8. Sformułowanie wniosków i ocena zgromadzonego materiału badawczego.

Eksperyment zawierał sześć zadań, które użytkownicy wykonywali na każdej ocenionej stronie internetowej (dwie istniejące i dwa prototypy). Treść tych zadań przedstawia tabela 4.

Tabela 4: Zadania do wykonania podczas eksperymentu

L.p.	Treść zadania
Zad1	Zlokalizuj narzędzie do powiększenia czcionki na prezentowanej stronie.
Zad2	Znajdź wyszukiwarkę.
Zad3	Gdzie znajduje się *Deklaracja dostępności*?
Zad4	Znajdź pole w formularzu, które umożliwi wyświetlenie całego rozkładu na przystanku o wybranym numerze.
Zad5	Znajdź na stronie www informacje o sprzedaży biletów.
Zad6	Znajdź narzędzie umożliwiające wyświetlenie galerii.

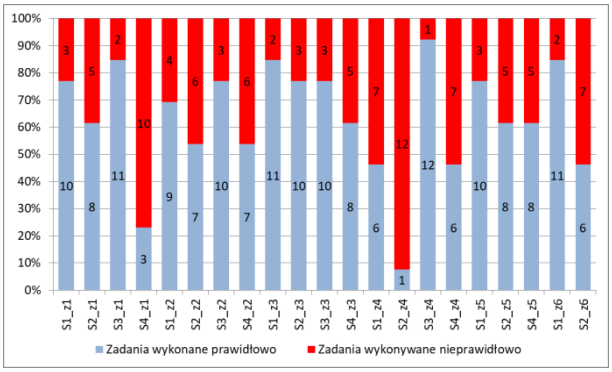
Grupa badawcza miała łącznie do wykonania dwadzieścia dwa zadania. Liczba ta wynika z występowania galerii tylko na jednej z oficjalnych stron. Eksperyment został tak zaprojektowany, aby była możliwość porównania i oceny serwisów istniejących z ich prototypami. Każda sesja składała się z takiej samej serii instrukcji i zadań, wyświetlanych za każdym razem w tej samej kolejności. Poza tym każda sesja obejmowała następujące czynności:

- wyjaśnienie badanemu sposobu przeprowadzenia eksperymentu,
- wyrażenie zgody przez badanego na przeprowadzeniu eksperymentu,
- kalibracja urządzenia oraz odpowiednie posadzenie uczestnika przed stanowiskiem badawczym,
- nagranie przez eyetracker ruchów gałek ocznych osób badanych,
- poinformowanie uczestnika o końcu badania.

5. Wyniki badań

5.1 Poprawność wykonania zadań

Po wyeksportowaniu danych z oprogramowania iMotions przeprowadzono analizę wyników. Najpierw skupiono się na zbadaniu poprawności wykonania poszczególnych zadań. Na rysunku 6 znajdują się wyniki pokazujące jak poradzili sobie respondenci z realizacją poszczególnych zadań na określonej, testowanej witrynie. Na wykresie, symbolami S1–S4 oznaczone są serwisy, które wyjaśnione są w tabeli 3. Natomiast skróty z1–z6 to kolejne zadania o treści przedstawionej w tabeli 4.



Rysunek 6: Rezultaty wykonania zadań.

Komentarze do uzyskanych wyników przedstawionych na powyższym wykresie znajdują się w tabeli 5.

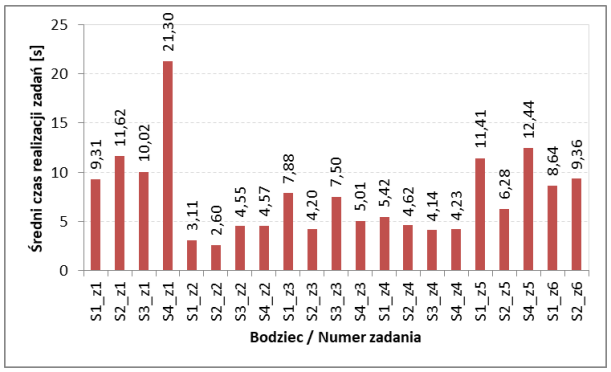
Tabela 5: Komentarze do realizacji poszczególnych zadań

Zadanie	Komentarz
S1_z1	Bezproblemowe odnajdowanie narzędzia, niektórzy respondenci nie znali tej możliwości.
S2_z1	W tym przypadku zadziałał efekt uczenia, większość uczestników domyśliła się, że szukane narzędzie jest dostępne po naciśnięciu niebieskiej ikony znajdującej się przy prawej krawędzi przeglądarki. W wynikach wystąpiło dużo niedokładności między fiksacją, a rzeczywistą lokalizacją ikony, więc niezbędna była ocena ekspercka.
S3_z1	Uczestnicy szybko znajdowali znane im narzędzie – dużą literę A z plusem; umiejscowienie tego narzędzia w górnej, prawej części strony ułatwiało to zadanie.
S4_z1	Przeważająca liczba badanych uważała, że do powiększenia czcionki służy ikona lupy, odpowie-

	dzialna za uruchomienie wyszukiwarki. Także literę A znajdującą się na końcu pola adresu, z jego prawej strony (przeglądarka Microsoft Edge), służącego do czytania strony na głos, uważano za szukane narzędzie. Dużą trudnością było domyślenie się, że narzędzie powiększenia czcionki znajduje się w tzw. menu dostępności znajdującym się po naciśnięciu niebieskiej ikony z osobą na wózku, znajdującą się po lewej stronie okna przeglądarki. Uczestnicy nie znają rozwiązania, które stosuje WordPress.
S1_z2	Duża większość badanych z łatwością znajdowała panel, w którym znajduje się narzędzie do wyszukiwania. Trzy osoby prawdopodobnie nie zrozumiały polecenia lub przypadkowo za szybko kliknęli, przechodząc do następnej planszy.
S2_z2	Wystąpiły duże problemy ze znalezieniem pola wyszukiwarki, choć ulokowane było ono w centralnej części strony. Być może problem wynikał z faktu, że pole to miało szary kolor, nie miało placeholdera, a przycisk zawierał etykietę w jęz. angielskim.
S3_z2	Uczestnicy sprawnie lokalizowali położenie wyszukiwarki, która znajdowała się w środkowej, górnej części strony.
S4_z2	Respondenci mieli problem z wyborem narzędzia do wyszukiwania: mylili ikonę lupy z ikoną zawierającą narzędzia dostępności.
S1_z3	Znalezienie linku *Deklaracja dostępności*, pomimo wielu treści na stronie i ulokowaniu go w prawym dolnym rogu, nie stanowiło problemu dla respondentów.
S2_z3	Znalezienie deklaracji nie stwarzało problemów. Lokalizacja deklaracji na dole strony wymagała cierpliwości w przeglądaniu obszernej treści strony.
S3_z3	Wyraźny napis *Deklaracja dostępności*, umieszczony w dolnej części strony, na ogół nie stwarzał problemu z jego odszukaniem.
S4_z3	Respondenci, którzy nie odnaleźli *Deklaracji dostępności*, prawdopodobnie nie zrozumieli polecenia. Link był umieszczony w menu górnym, w jego centralnej części, co w większości przypadków nie sprawiało problemów z jego odnalezieniem.
S1_z4	W kilku przypadkach problemem było to, że badani mylili opcję *Pokaż trasę linii na przystanku* z *Pokaż rozkład na przystanku*. Prawdopodobnie wynikało to z kolejności – pierwsza opcja znajdowała się nad drugą.
S2_z4	Wystąpiły podobne problemy jak w S1_z4, ale z dużo większym nasileniem, ponieważ tylko jedna osoba wykonała zadanie prawidłowo.
S3_z4	Miejsce umieszczenia pola z placeholderem oraz fakt, że było to jedyne pole na stronie, spowodowało, że tylko jedna osoba nie wskazała tego pola, najpewniej na skutek niezrozumienia zadania.
S4_z4	Prawie połowa uczestników miała problemy z tym zadaniem. Wynikały one z faktu, że chociaż pole to znajdowało się na początku formularza i zawierało placeholder, to tekst w nim był niezrozumiały, a ponadto pole miało kolor szary, przez co mogło sprawiać wrażenie, że jest ono tylko do odczytu.
S1_z5	W tym przypadku problemem mogło być umiejscowienie szukanej opcji – *Sprzedaż biletów*, na dole w lewym menu (przedostatnia opcja). Drugą problematyczną kwestią była duża ilość treści wyświetlanej na tej stronie.
S2_z5	Badani znajdowali opcję *Bilet elektroniczny* i zaprzestawali poszukiwania właściwej opcji, tzn. *Sprzedaż biletów*.
S4_z5	Główną trudnością w znalezieniu właściwej opcji było jej inne nazwanie.
S1_z6 S2_z6	Łatwiej było uczestnikom znaleźć tekstowy link do *Galerii* niż jedną z wielu ikon, pod którą krył się link do *Galerii*.

5.2 Pomiar czasu wykonania zadań

Kolejna część wyników dotyczy zmierzonych i uśrednionych czasów wykonania poszczególnych zadań (rysunek 7).



Rysunek 7: Średnie czasy wykonania zadań.

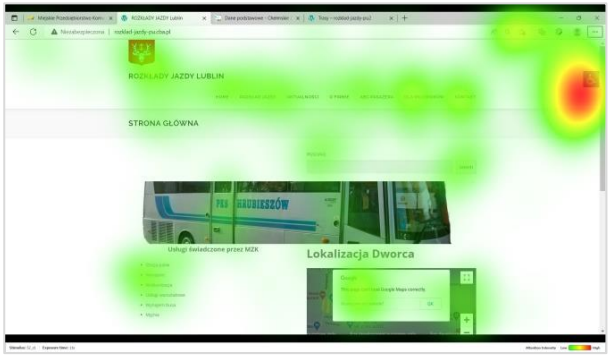
Z powyższego wykresu wynika, że w przypadku zadania pierwszego, znalezienie i domyślenie się, że niebieski przycisk umożliwia powiększenie czcionki na stronie S4, zajmowało uczestnikom średnio aż 21,30 sekund. Dla pozostałych stron czasy te były prawie o połowę krótsze. Dla strony S2 zadziałał prawdopodobnie efekt uczenia się, ponieważ bodziec ten był wyświetlany jako ostatni. Najdłuższy czas wykonywania zadania drugiego wystąpił dla bodźca S4. Uczestnicy musieli się tutaj domyśleć, że wyszukiwarka znajduje się po kliknięciu na ikonie z grafiką lupy. Z kolei w zadaniu trzecim najdłuższy czas wyszukiwania dla stron S1 i S3 wynikał z tego, że szukane elementy znajdowały się na dole stron, w miejscach w których było dużo treści. Natomiast czasy realizacji zadania czwartego były zbliżone, pomimo tego, że strona S3 miała bardzo prosty formularz, składający się z jednego pola, natomiast w formularzu ze strony S4 szukane pole było na samym początku formularza. W przypadku zadania piątego czasy realizacji zadania były warunkowane złożonością struktury treści. Strona S2 miała najprostszą strukturę i zawierała mało treści, tak więc czas wykonywania tego zadania dla tej strony był o połowę krótszy od pozostałych. Podczas realizacji zadania szóstego uczestnicy badań mieli problem na stronie S2 w określeniu miejsca, w którym znajduje się galeria, ponieważ należało dokonać wyboru jednej z wielu znajdujących się na stronie ikon/grafik.

5.3 Analiza jakościowa na podstawie map cieplnych

Badania jakościowe polegały na analizie map cieplnych i wybranych ścieżek skanowania. Pokazują one zagre-

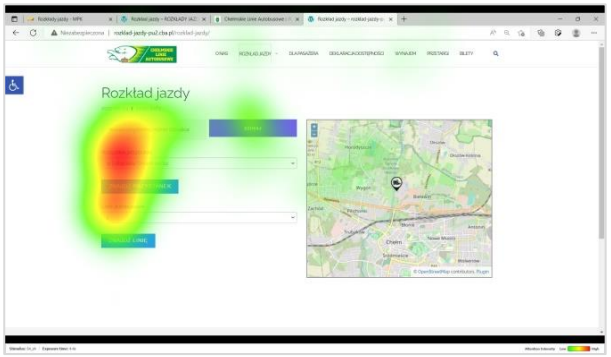
gowane wyniki dla określonego zadania dla wszystkich badanych. Czerwonym kolorem oznaczone są obszary, w których respondenci najczęściej i/lub najdłużej skupiali swoją uwagę. Dzięki barwom: zielonej, żółtej i czerwonej oraz przeźroczystości można zobaczyć elementy prezentowanej sceny, które przyciągały uwagę badanych, a które zostały przez nich pominięte.

W zadaniu pierwszym, na podstawie mapy cieplnej i lokalizacji gorącego obszaru można wywnioskować, że szukana funkcjonalność powiększenia czcionki znajduje się w grupie opcji, do której dostęp jest za pośrednictwem niebieskiej ikony z rysunkiem osoby na wózku inwalidzkim (rysunek 8). Duży obszar bodźca pokryty jest barwą zieloną, co świadczy o dokładnej eksploracji przez badanych całej zawartości strony www. Na oficjalnej stronie funkcjonalność ta może być uruchamiana bezpośrednio po kliknięciu na literze A. W prototypie dostęp do tej funkcjonalności można było uzyskać za pośrednictwem wcześniej opisanej ikony, co sprawiło respondentom duże trudności, gdyż nie mieli wcześniej do czynienia z takim rozwiązaniem. Warto również zwrócić uwagę na to, że badanym znacznie łatwiej było odaleźć te funkcje, które znajdowały się po prawej stronie ekranu.



Rysunek 8: Wynik badania eyetrackingowego w postaci mapy cieplnej dla zadania S2_z1.

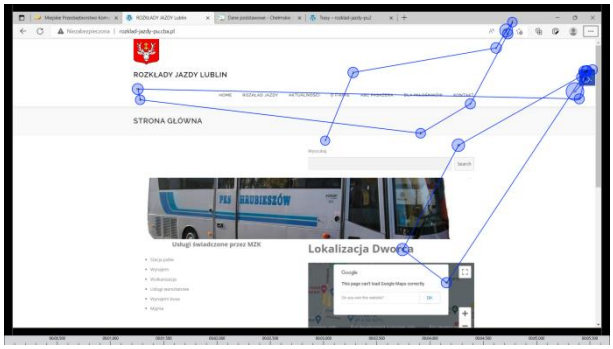
Mapa cieplna na rysunku 9 pokazuje, że uczestnicy przeskanowali wszystkie etykiety i pola formularza, od góry do dołu, choć szukane pole znajdowało się na samej górze. Największy czerwony obszar pokazuje, że badani łatwo i szybko odnajdowali szukane pole, ale dla pewności przeglądali także pozostałe pola znajdujące się w formularzu.



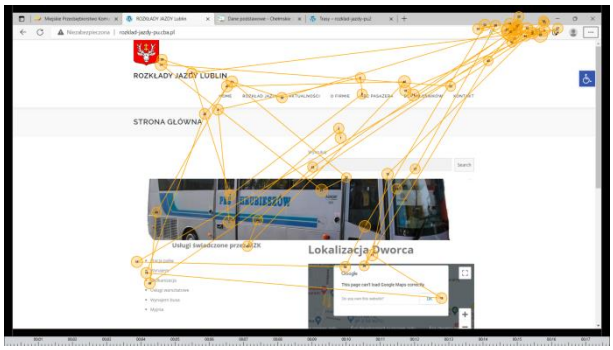
Rysunek 9: Wynik badania eyetrackingowego w postaci mapy cieplnej dla zadania S4_z4.

5.4 Analiza jakościowa na podstawie ścieżek skanowania

Ścieżki skanowania przez wzrok, w postaci linii reprezentują ruch sakadowy, a w postaci kółek zatrzymania (fiksacje) na scenie wizualnej. Rysunki 10 i 11 są przykładami takiego obrazowania, z tym że pierwszy z nich pokazuje prawidłowe i sprawne dotarcie do szukanego celu (dwadzieścia jeden kroków), natomiast drugi – długie skanowanie (kilkadziesiąt kroków) zakończone niepowodzeniem, czyli nie odnalezieniem szukanego narzędzia.



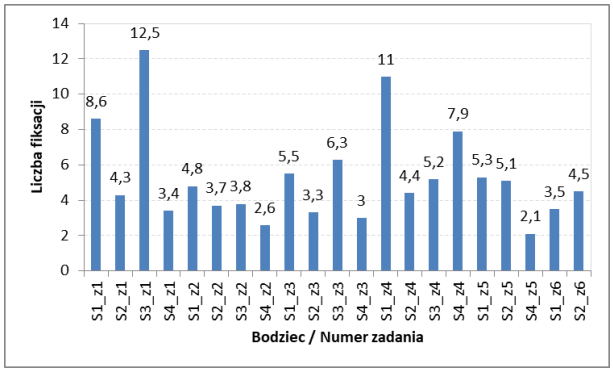
Rysunek 10: Ścieżka skanowania dla S2_z1_JK.



Rysunek 11: Ścieżka skanowania dla S2_z1_MK.

5.5 Analiza ilościowa na podstawie liczby fiksacji w obszarze zainteresowania

Na rysunku 12 przedstawiono średnie liczby fiksacji dla wyznaczonych obszarów zainteresowania na wyszukiwanych obiektach w ramach poszczególnych zadań.

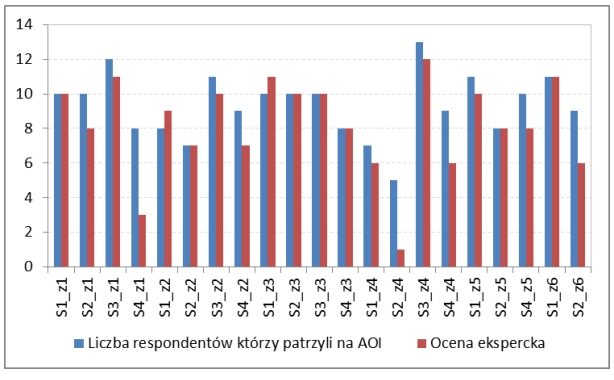


Rysunek 12: Średnia liczba fiksacji w obszarach AOI, w których znajdowały się szukane elementy.

W zadaniu pierwszym dużo fiksacji wystąpiło w obszarach, na stronach, na których bezpośrednio znajdowały się litera i znak A+ (strony S1 i S3). Gdy funkcjonalność powiększenia czcionki była ukryta (uruchamiana po wcześniejszym kliknięciu na przycisku otwierającym listę narzędzi dostępności) liczba fiksacji była znacznie mniejsza. Ta mała średnia liczba fiksacji wynikała również z faktu, że często nie znaleziono lub nie domyślono się, jak dotrzeć do narzędzia służącego do powiększania czcionki. W drugim zadaniu najmniejsza liczba fikasacji była na elemencie, który pośrednio dawał dostęp do wyszukiwarki (ikona lupy). W tym przypadku o małej liczbie fiksacji zdecydowały także niewielkie rozmiary szukanej grafiki. Zaskakująco mała liczba fiksacji miała miejsce dla strony S4, gdzie poszukiwano *Deklaracji dostępności* znajdującej się na środku menu głównego. Natomiast w przypadku zadania czwartego bardzo duża liczba fiksacji dla bodźca S1 mogła wynikać z tego, że uczestnicy badań długo zastanawiali się, czy wybrane pole jest tym właściwym. Podobna sytuacja zachodziła dla zadania czwartego w przypadku strony S4, pomimo tego że szukany element znajdował się na górze formularza i miał znaczne rozmiary. Z kolei mała liczba fiksacji w zadaniu piątym, również dla bodźca S4, wynikała z faktu, że badani mieli duży problem z realizacją zadania, ponieważ musieli się domyślić, że to czego szukają, znajduje się pod nazwą *Bilety*. W ostatnim, szóstym zadaniu większa średnia liczba fiksacji dla strony S2 była efektem tego, że badani zastanawiali się, czy pod niebieską ikoną ze schematem aparatu fotograficznego znajduje się szukana przez nich galeria.

5.6 Analiza ekspercka respondentów

Na rysunku 13 przedstawiono analizę ekspercką, polegającą na zliczeniu respondentów, których uwaga znalazła się w danym obszarze zainteresowania. Równolegle do tego, za pomocą platformy iMotions sprawdzono, wzrok ilu badanych znalazł się w wyzanczonym AOI.



Rysunek 13: Uwaga na określone AOI zestawiona z ekspercką oceną realizacji zadania.

Ocena ekspercka zazwyczaj była bardziej rygorystyczna i eliminowała sytuacje, w których osoby badane patrzyły na szukany element, ale nie rozpoznawały w nim rozwiązania zadania. W niektórych przypadkach, takich jak S4_z1 i S2_z4 różnice były bardzo duże ze względu na fakt, że badani nie znali odpowiedzi lub nie byli jej pewni. W dwóch przypadkach: S1_z2 i S1_z3 ocena ekspercka była wyższa, choć punkty fiksacji nie znajdowały się w konkretnym AOI. Spowodowane było to pewnymi niedokładnościami, które mogły wynikać z kilku przyczyn, na przykład: wady wzroku, problemów z kalibracją czy poruszania się uczestnika w czasie badań.

6. Wnioski

W wyniku zrealizowanych badań nasunęło się wiele wniosków, które mogą stanowić dobre praktyki w zakresie użyteczności i dostępności oraz mogą być przydatne twórcom serwisów www. Są one jednocześnie odpowiedziami na pytania badawcze postawione na początku pracy.

1. Użycie podobnych symboli graficznych często wprowadza w błąd użytkowników. Na przykład użycie dużej litery A jako narzędzia służącego do odczytu na głos zawartości strony, mylone jest z literą A z plusem służącą do powiększenia czcionki na stronach www. Inny przypadek to ikona lupy, która powodowała dezorientację, gdyż niekiedy służyła do powiększenia zawartości strony, a czasami do wyświetlenia pola wyszukiwania.
2. Pośredni dostęp do narzędzi dostępności na ogół utrudnia użytkownikom skorzystanie z nich. Tak było w przypadku wtyczki WordPressa oferującej szeroką gamę narzędzi dla osób niepełnosprawnych.
3. Nieprzemyślany wygląd niektórych elementów strony stwarza problemy dla odbiorców treści, np. szare tło i brak placeholdera w polu tekstowym wyszukiwarki
4. Eksperymentowanie z umiejscowieniem niektórych narzędzi powoduje trudności z ich znalezieniem.
5. Dużą dezorientację na stronach internetowych wprowadzają nietypowe nazwy, linki, opcje menu, itd.
6. Duża ilość treści i brak wyeksponowania najważniejszych elementów powoduje problemy ze znalezieniem właściwej informacji.
7. Brak podpisów grafiki sprawia, że użytkownicy często muszą się dłużej zastanowić i sprawdzić, czy to czego chcą użyć, jest właściwą opcją.
8. Umieszczenie elementu/informacji na dole strony na ogół wydłuża czas dotarcia do nich.

Literatura

[1] D. Paszkiewicz, J. Dębski, Dostępność serwisów internetowych. Dobre praktyki w projektowaniu serwisów internetowych dostępnych dla osób z różnymi niepełnosprawnościami, Stowarzyszenie Przyjaciół Integracji, Warszawa, 2013, https://www.power.gov.pl/media/13588/Dostepnosc-serwisow-internetowych-Dominik-Paszkiewicz-Jakub-Debski.pdf, [18.08.2022].

[2] M. Bednarczyk, M. Dzieńkowski, Evaluation of the availability of websites of communes in the Lubelskie

Province, Journal of Computer Sciences Institute, 19 (2021) 114–120, [18.08.2022].

[3] K. Kałan, D. Karpiuk, M. Dzieńkowski, Analiza użyteczności z uwzględnieniem aspektów dostępności wybranych serwisów internetowych uczelni wyższych, Journal of Computer Sciences Institute, 21 (2021) 259–302, [18.08.2022].

[4] S. Cullipher, S.J.R. Hansen, J.R. VandenPlas, Eye Tracking as a Research Tool: An Introduction, [w] Eye Tracking for the Chemistry Education Researcher, (2018) 1–9.

[5] A. Poole, L.J. Ball, P. Phillips, In search of salience: A response time and eye movement analysis of bookmark recognition, People and Computers XVIII – Design for Life: Proceedings of HCI 2004, London, (2004) 363–378.

[6] J. H. Goldberg, X. P. Kotval, Computer interface evaluation using eye movements: methods and constructs, International Journal of Industrial Ergonomics, 24 (6) (1999) 631–645.

[7] Oficjalna strona WordPressa, https://pl.wordpress.org/, [18.08.2022].

[8] Gazepoint, https://www.gazept.com/product/gp3hd/, [18.08.2022].

[9] iMotions, https://imotions.com, [24.09.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 63–67
Received: 26 September 2022
Accepted: 7 November 2022

Analysis of configuration distribution methods in service application environments

Analiza metod dystrybucji konfiguracji w środowiskach aplikacji usługowych

Arkadiusz Bryczek*, Piotr Kopniak

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

Abstract
This work is an analysis of methods and solutions enabling centralization of the storage of configuration values in service application environments. It presents a comparison focused on the user's interaction with the available solutions and performance tests during the operation. Consul, ZooKeeper tools and the implementation of the configuration server with the use of the Spring Cloud Config library were tested. Created test environment with the use of Java Spring, Kafka and Python technologies was used for this purpose. The aim of the research is to determine the best tool for working in large microservice networks and the most optimal in the context of working with the user. The presented results confirm the advantage of the Consul tool in terms of efficiency and interface quality over other solutions.

Keywords: configuration distribution; microservices; cloud solutions

Streszczenie
Niniejsza praca jest analizą metod oraz rozwiązań umożliwiających centralizację przechowywania wartości konfiguracyjnych w środowiskach aplikacji usługowych. Prezentuje ona porównanie ukierunkowane na współpracę użytkownika z dostępnymi rozwiązaniami oraz badanie wydajność podszas wykonywania operacji. Testom poddano narzędzia Consul, ZooKeeper oraz implementację serwera konfiguracyjnego z wykorzystniem biblioteki Spring Cloud Config. Wykorzystane do tego celu zostało autorskie środowisko testowe stworzone z użyciem technologii Java Spring, Kafka oraz Python. Celem jest wyznaczenie najlepszego z narzędzi do pracy w dużych sieciach mikroserwisowych oraz najoptymalniejszego w kontekście pracy z użytkownikiem. Zaprezentowane wyniki potwierdzają przewagę narzędzia Consul w aspekcie wydajności oraz jakości interfejsu nad pozostałymi rozwiązaniami.

Słowa kluczowe: dystrybucja konfiguracji; mikroserwisy; rozwiązania chmurowe

*Corresponding author
Email address: **arkadiusz.bryczek@pollub.edu.pl** (A. Bryczek)

1. Wstęp

Rozwiązania chmurowe zmieniają sposób rozwoju i wdrażania aplikacji ze względu na ich unikalne wymagania. Deweloperzy dążą do uzyskania wyższej skalowalności i niezawodności. Dynamiczne środowiska jakimi są środowiska chmurowe mogą wymagać posiadania magazynu przechowującego konfigurację [1]. Autonomiczny charakter nieodłącznie związany z mikrousługami wymaga od twórcy aplikacji wzięcia odpowiedzialności za aspekty operacyjne, takie jak łatwa konfigurowalność [2]. W przypadku architektury monolitycznej, gdzie posiadamy jedną główną aplikację, a co za tym idzie jeden plik przechowujący wartości konfiguracji nie jest wymagane wydzielanie jej do wspólnego miejsca. Posiadając wiele mikroserwisów koniecznym byłoby powielenie tych zasobów, a w momencie edycji, wykonanie jej w wielu miejscach dla części wspólnych. Pomocne w tej sytuacji stają się narzędzia, które umożliwiają aplikacjom na jednym środowisku pobranie konfiguracji po HTTP. Wybór odpowiedniego narzędzia staje się ważnym elementem analizy w tworzeniu architektury systemu.

Technologie do zarządzania konfiguracją służą do przechowywania wartości konfiguracyjnych i umożliwiają dynamiczną zmianę wartości. Budując środowisko zastosowanie technologii przechowujących konfigurację takich jak Consul lub ZooKeeper to powszechna praktyka podczas tworzenia sieci mikroserwisów [2].

Celem artykułu jest przeprowadzenie i prezentacja wyników badań określających wydajność oraz jakość interfejsu narzędzi Consul, ZooKeeper oraz implementacji serwera konfiguracyjnego z wykorzystaniem biblioteki Spring Cloud Config, które wzbogacają aplikację budowaną w architekturze mikroserwisowej o scentralizowany magazyn przechowujący konfigurację oraz umożliwiający jej szybką podmianę. Zważając na open source'ową dystrybucje narzędzia Consul, łatwość konfiguracji oraz wbudowany graficzny interfejs można wysunąć tezę, że charakteryzuje go lepsza wydajność od pozostałych badanych aplikacji oraz jakościowo lepszy interfejs.

Rozdział drugi przybliży badany temat oraz źródła wiedzy wykorzystanej do stworzenia prezentowanej pracy. W rozdziale trzecim zaprezentowane zostanie

środowisko przygotowane do przeprowadzenia badań. Metody badawcze oraz opis scenariuszy testowych zostanie zawarty w rozdziale czwartym. Rozdział piąty przedstawi wyniki otrzymane jako rezultat przeprowadzonych testów oraz ich omówienie. Dyskusja i porównanie zebranych danych z uprzednio publikowanymi badaniami oraz dostępnymi źródłami zostaną zawarte w rozdziale szóstym. Siódmy rozdział zawierał będzie podsumowanie oraz wnioski zebrane w czasie eksperymentu.

2. Przegląd literatury

W artykule Omar'a Al.-Debagy'ego i Peter'a Martinek'a [3] przedstawiono porównanie architektury monolitycznej z architekturą mikroserwisową. Udowodniono, że zastosowanie podejścia budowy sieci aplikacji składającej się z wielu współpracujących serwisów jest bardziej optymalne w aspekcie użytkowania przez większą liczbę użytkowników. Korzyści są również widoczne na płaszczyźnie ładowania aplikacji, szybkości restartu w przypadku błędu. Zbadany został również czas odkrywania aplikacji w środowiskach usługowych, potwierdzono tym wysoką wydajność oraz wsparcie w komunikacji między serwisami oraz to, że architektura mikroserwisowa nie utrudnia połączenia i zależnej współpracy aplikacji.

Artykuł [4] dotyczy ewolucji narzędzia ZooKeeper oraz zawiera opis możliwej do realizacji metody dynamicznego przeładowywania konfiguracji. Wykorzystany w badaniu został schemat pracy, w którym aplikacja kliencka zamiast regularnego odpytywania serwera o zmiany ustawia na obszary, które chce monitorować obserwatorów. Zadaniem ich jest nasłuchiwanie na zdarzenia w subskrybowanych gałęziach, do których ZooKeeper wysyła powiadomienia w momencie aktualizacji kluczy konfiguracji. W momencie, gdy klient odbierze takie powiadomienie, wysyła zapytanie do aplikacji w celu pobrania aktualnych danych oraz przeładowuje swoje ustawienia. Podejście to odstępuje od tradycyjnego, lecz pozwala to ograniczyć obciążenie poprzez reakcję aplikacji klienckiej tylko w sytuacji, gdy zmiana została dokonana w interesującym go obszarze. Optymalizuje to obciążenie i szybkość pracy w przypadku wysokiej liczby podpiętych klientów.

W artykule autorstwa Andisheh Feizi oraz Chui Yin Wong [9] zawarte zostały badania porównujące współpracę użytkowników z interfejsami graficznymi oraz wierszem poleceń. Testom poddana została efektywność pracy osób początkujących oraz wdrożonych w użytkowanie danego oprogramowania. Interfejs graficzny był postrzegany jako łatwiejszy do nauczenia dla obu grup. Potwierdzone zostało, że praca z nim jest efektywniejsza, jeśli jest poprawnie zbudowany. Pod uwagę wzięte zostały aspekty takie jak odpowiednia treść etykiet, odpowiednio dobrane ikony odnoszące się do zawartości zakładek oraz głębokość, oznaczająca ilość podstron dla danej strony. Stwierdzono również, że najlepszym rozwiązaniem jest udostępnienie graficznego interfejsu o możliwościach analogicznych do interfejsu konsolowego.

Badania dotyczące komunikacji narzędzi do dystrybucji konfiguracji takich jak ZooKeeper lub Consul z klientami, uwzględniające blokowanie połączenia przez każdego klienta oraz zwalnianie go zostały uwzględnione w artykule Piotra Grzesika i Dariusza Mrozka [5]. Został tam zbadane czasy uzyskiwania przez klienta blokady podczas połączenia oraz zwalniania jej w określonych środowiskach oraz pod obciążeniem. Badania pokazują, że narzędzia wypadają niekorzystnie w przypadku, gdy kilku klientów próbuje uzyskać dostęp do tego samego klucza. Najlepszą wydajnością charakteryzuje się jednak ZooKeeper, który w najgorszym scenariuszu zaprezentował najlepszy średni czas blokady i zwolnienia.

Powyższe publikacje zawierają eksperymenty potwierdzające, że wykorzystanie magazynu konfiguracyjnego jako mikroserwisu nie wpływa negatywnie na ogólną wydajność całego systemu. Potwierdzone zostało również, że za pomocą jednego narzędzia można zastosować różne podejście w zbudowaniu dynamicznego systemu dystrybuującego wartości zdefiniowane w konfiguracji. Żadne z nich nie prezentuje badania wydajności narzędzi podczas pracy w dużej sieci mikroserwisów. Brakuje również porównania wydajności w scenariuszach, gdy aplikacja jest powiadamiana o zmianie wartości w magazynie, a regularnym odpytywaniu magazynu o stan przechowywanych w nim wartości. Badaniu nie została też poddana współpraca użytkownika z narzędziem poprzez udostępniony interfejs.

3. Środowisko badawcze

Bazą do przeprowadzenia badania są wspomniane narzędzia do przechowywania konfiguracji oraz aplikacja testowa napisana w języku Java z wykorzystaniem szkieletu programistycznego Spring Boot oraz jego integracji z rozwiązaniami chmurowymi [6]. W aplikacji testowej wykorzystane zostaną gotowe biblioteki Spring Cloud oraz zmienne umożliwiające uzyskanie połączenia z aktualnie badanym narzędziem do przechowywania konfiguracji oraz kolejką wiadomości Kafka opisaną w dalszej części rozdziału. Aplikacja zawierać będzie jedną klasę posiadającą pole, pod które będą wczytywane wartość z pliku konfiguracyjnego przechowywanego w magazynie. W momencie odebrania informacji o zmianie wartości klucza, aplikacja zapisze czas zdarzenia oraz prześle go do tymczasowego magazynu danych. Dodatkowo utworzona została w języku Python aplikacja pomocnicza zapisującą czas zatwierdzenia zmian w konfiguracji.

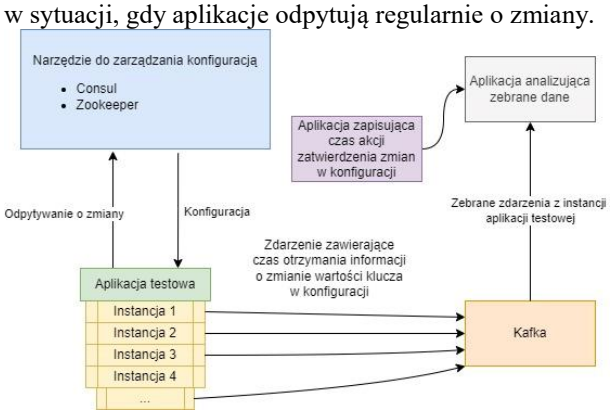
Osobną częścią będzie konfiguracja kolejki do strumieniowego przesyłania wiadomości Kafka [7]. Posłuży ona jako magazyn tymczasowo przechowujący zebrane z instancji aplikacji testowej czasy, które będzie wysyłała aplikacja w momencie, gdy zostanie odebrana informacja o zmienionej wartości klucza w konfiguracji.

Równie dobrym rozwiązaniem byłby zapis do bazy danych, umożliwiający przechowywanie otrzymanych wyników oraz przechowywanie ich. Obrane podejście wykorzystania kolejki pozwala na bieżącą analizę danych i monitorowanie testu podczas jego przebiegu.

Ostatnią częścią jest aplikacja analizująca zebrane w kolejce zdarzenia. Skonfigurowane zostanie połączenie oraz nasłuchiwanie na zdarzenia w kolejce. Serwis będzie przyjmował dane z kolejki oraz zwracał je jako rezultat zapytania.

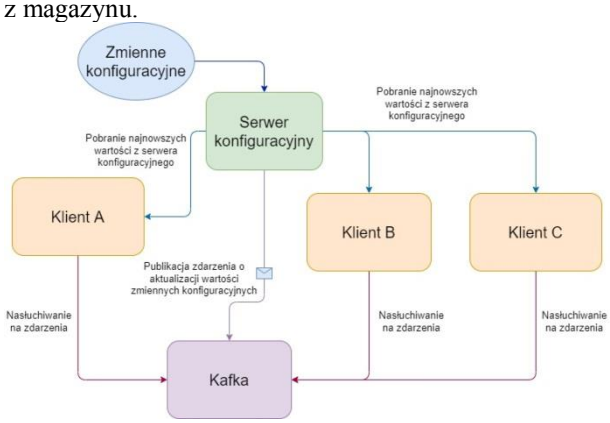
Uruchomienie środowiska zostanie zrealizowane z wykorzystaniem platformy konteneryzacji Docker. Każda z aplikacji uruchomiona zostanie jako osobny kontener pomijając aplikację pomocniczą, która będzie uruchamiana przed każdym testem. Pomocna w badaniu będzie funkcjonalność skalowania kontenerów. Dzięki niej w prosty sposób za pomocą flagi „scale" powielone zostaną instancje aplikacji testowej. Pozwoli to zachować równość aplikacji w środowisku związaną z przydzieleniem zasobów oraz poprawi rzetelność badań.

Na Rysunku 1 przedstawiono połączenia oraz informacje o danych przekazywanych pomiędzy poszczególnymi aplikacjami w utworzonym środowisku w sytuacji, gdy aplikacje odpytują regularnie o zmiany.



Rysunek 1: Połączenia w środowisku testowym – odpytywanie przez aplikacje o zmiany.

Zawarty na Rysunku 2 diagram zawiera sieć komunikacyjną aplikacji w sytuacji, gdy instancje aplikacji testowej są powiadamiane o zmianach, a następnie wykonywane jest pobranie danych z magazynu.



Rysunek 2: Połączenia w środowisku testowym – poinformowanie aplikacji o wprowadzonych zmianach.

4. Metody badawcze

Zaprezentowane środowisko posłuży do przeprowadzenia dwóch typów badań. Wykonane zostanie obiektywne badanie wydajność metod oraz subiektywne badanie interfejsu przeprowadzone na grupie testowej składającej się z dziesięciu studentów informatyki.

Analiza wydajności zostanie wykonana poprzez zbadanie czasu odebrania informacji o wprowadzeniu zmian oraz reakcji na to zdarzenie w dwudziestu sześciu instancjach aplikacji testowej. Wszystkie zostaną podpięte do jednej instancji narzędzia do przechowywania konfiguracji. Na wydajność składa się zarówno szybkość reakcji badanego narzędzia oraz jego optymalizacja w aspektach obciążenia systemu. W badanej technologii dokonana zostanie zmiana wartości klucza, która zapoczątkuje przeładunek we wszystkich instancjach zaimplementowanego rozwiązania. Każda z instancji w momencie uzyskania informacji o zdarzeniu zmiany wartości klucza zapisze pod zmienną dokładny co do wartości tysiącznych sekundy czas. Następnie wszystkie instancje prześlą zapisaną wartość do kolejki, która posłuży jako tymczasowy magazyn. W tle uruchomiona będzie aplikacja nasłuchująca na dane wpływające do tego magazynu. Dla większej dokładności badania, test zostanie powtórzony dziesięć razy dla każdej z metod. Obliczenia oparte będą o zebrane dane czasowe oraz czas startu badania zapisanego przez wspomnianą aplikację pomocniczą.

Na podstawie otrzymanych wiadomości zostanie obliczony średni czas przeładunku konfiguracji we wszystkich instancjach aplikacji testowej dla danej metody. Analogiczny scenariusz zostanie wykonany dla każdego z badanych rozwiązań.

Badanie interfejsu użytkownika zostanie przeprowadzone na grupie dziesięciu osób, które wcześniej nie miały kontaktu z żadnym z badanych narzędzi. Każdy badany otrzyma przed rozpoczęciem adekwatną dla narzędzia instrukcję opisującą krok po kroku w jaki sposób odnaleźć wartość klucza, którą będzie miał za zadanie zmienić. Użytkownik zapoznaje się z instrukcją, a następnie deklaruje gotowość do odbycia badania. Każda badana osoba wykona osobne scenariusze dla każdego z narzędzi.

Grupa dokona również oceny interfejsu poprzez indywidualne wypełnienie ankiety określającej jakość interfejsu na podstawie ankiety LUT (ang. Lublin University of Technology) [8]. Wybrane zostały w tym celu z ankiety obszary:

- nawigacja i struktura składająca się z podobszarów: łatwość nawigowania, hierarchia informacji oraz struktura informacji,
- komunikaty, pomoc dla użytkownika uwzględnia elementy takie jak: komunikaty ogólne, komunikaty o błędach oraz informacje zwrotne i pomoc,
- interfejs aplikacji obejmujący dwa podobszary: layout oraz dobór barw,

- wprowadzanie danych, który składa się z podobszarów odnoszących się do formularzy oraz wprowadzanych danych.

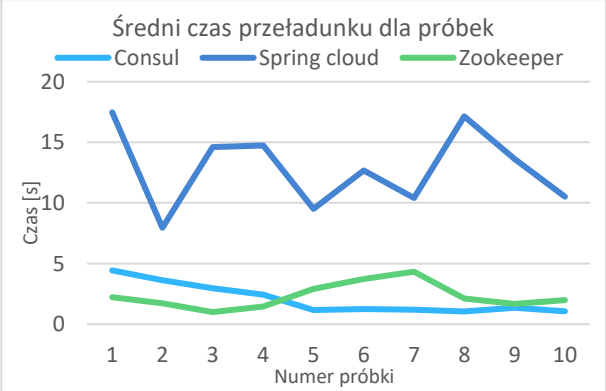
Każda z sekcji zawiera pytania oraz odpowiedzi w postaci oceny określanej jako wartość liczbowa z przedziału od jeden do pięć, gdzie jeden oznacza słabe wsparcie danego aspektu w interfejsie badanego narzędzia, a pięć jego łatwość i intuicyjność oraz prostotę w wykorzystaniu. Rezultatem obliczenie współczynnika WUP (ang. Web Usability Points) [8]. Wykorzystany zostanie do tego wzór:

$$WUP = \frac{1}{n_a}\sum_{i=1}^{n_a}\frac{1}{s_i}\sum_{j=1}^{s_i}\frac{1}{q_{i_j}}\sum_{k}^{q_{i_j}} p_{ijk} \qquad (1)$$

gdzie n_a oznacza liczbę obszarów w ankiecie, S_i liczbę podobszarów w obszarze i, q_{i_j} liczbę pytań w obszarze i oraz podobszarze j a p_{ijk} to ocena przyznania przez osobę badaną pytaniu o numerze k w podobszarze j obszaru i.

5. Wyniki

Badania wydajności pozwoliły zapisać czas od startu do zakończenia operacji przeładowania w każdej z instancji aplikacji testowej. Na Rysunku 3 przedstawiono wykres średnich czasów przeładowania każdej próbki dla każdej badanej metody. Można zaobserwować na nim, że Consul oraz ZooKeeper wykazały podobne wyniki oraz rozbieżności zebranych czasów.



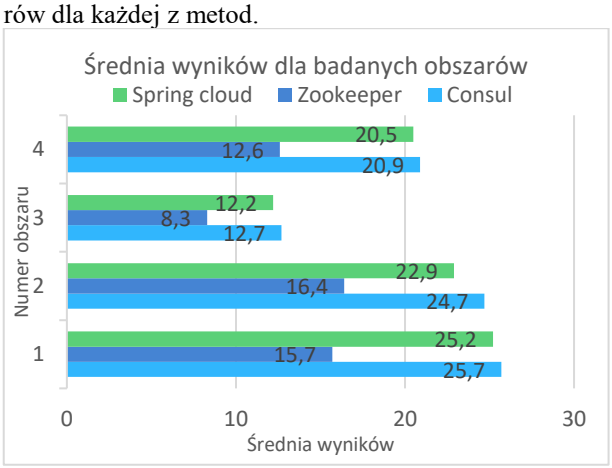
Rysunek 3: Średni czas przeładunku dla próbek.

Obliczając średnią z czasów próbek dla każdej z metod wyznaczono średni czas przeładunku oraz wartość rozbieżności dla danych.

Tabela 1: Wyniki czasowe badania wydajności metod

Metoda	Czas[s]	Odchylenie standardowe
Consul	2.056	1,233
ZooKeeper	2.314	1,043
Spring Cloud Config Server	12.865	3,223

Badanie jakości interfejsu wykonane przy pomocy ankiety LUT[8] skutkowało zebraniem zestawu danych na podstawie, którego obliczony został współczynnik jakości interfejsu WUP. Rysunek 4 przedstawia wykres średniej wyników dla każdego z poszczególnych obszarów dla każdej z metod.



Rysunek 4: Średnia wyników dla badanych obszarów.

W Tabeli 2 zawarte zostały średnie z obliczonych dla każdej badanej osoby współczynników, dla każdej z badanych metod. Uwzględniona została również rozbieżność otrzymanych wyników.

Tabela 2: Wyniki badania jakości interfejsu

Metoda	WUP	Odchylenie standardowe
Consul	4.23	0.53
ZooKeeper	2.68	0.22
Spring Cloud Config Server	4.03	0.38

6. Dyskusja

Celem przeprowadzonych badań było potwierdzenie tezy o przewadze narzędzia Consul nad podobnymi mu metodami tj. narzędziem ZooKeeper oraz implementacją serwera konfiguracyjnego z wykorzystaniem biblioteki Spring Cloud.

Wyniki badań wydajnościowych jednoznacznie wskazują na przewagę narzędzia oraz zastosowania metody, w której aplikacja pobiera informację o zmianach. Magazyny konfiguracyjne Consul oraz ZooKeeper uzyskały zbliżone wyniki w badaniu czasu reakcji, gdzie nieznaczną przewagą uzyskał serwer konfiguracyjny Consul. Obie aplikacje były notyfikowane o zmianach w ten sam sposób. Potwierdza to, że podejście, gdzie aplikacja odpytuje o zmiany serwer konfiguracyjny jest optymalniejsze oraz bardziej wydajne. Czas reakcji aplikacji oraz czas przeładowania wartości konfiguracyjnych jest krótszy, więc cała operacja zostanie również wykonana szybciej we wszystkich podpiętych pod serwer serwisach. Wyniki dla implementacji serwera konfiguracyjnego są znacząco gorsze. Jest to spowodowane zastosowanym mechanizmem notyfikacji, który mógłby okazać się lepszy, gdyby zwiększyć skalę testu. Podobne wyniki zaprezentowane zostały w artykule [5],

gdzie podobnie porównane zostały oba narzędzia w kontekście uzyskiwania klucza blokady oraz rywalizacji o klucz. Obie aplikacje podobnie wykazały nieznacznie różne wyniki, lecz w tym przypadku wydajniejszy był ZooKeeper.

Wyniki badania jakości interfejsu użytkownika otrzymane dzięki wynikom ankiety oraz policzonemu na ich postawie współczynnikowi zaprezentowały, że narzędziem posiadającym najlepszy jakościowo interfejs jest Consul. Analizując zebrane wyniki zaobserwować można przewagę interfejsów graficznych nad interfejsami konsolowymi. Użytkownicy podobnie wskazali, że interfejsy, gdzie użytkownik przeprowadza operacje poruszając się oraz wybierając elementy widoczne na ekranie są bardziej intuicyjne oraz łatwiejsze w użytkowaniu. Artykuł [9] prezentuje podobne wyniki potwierdzając, że praca z interfejsem graficznym aplikacji charakteryzuje się większą efektywnością w aspekcie pracy zarówno dla osoby początkującej oraz posiadającej doświadczenie w pracy z narzędziem.

7. Wnioski

Analizując wyniki przeprowadzonych badań można jednoznacznie stwierdzić, że w obydwu przypadkach najlepszymi wynikami charakteryzuje się Consul. Narzędzie uzyskało czas lepszy o 258 milisekund od serwera konfiguracyjnego ZooKeeper. Różnica jest nieznaczna, lecz skala testu zakładała przeładunek konfiguracji w 26 instancjach aplikacji testowe. Większa skala testu mogłaby spowodować uzyskanie większej różnicy w wynikach dla badanych serwerów konfiguracyjnych. Najgorszy wynik uzyskała implementacja serwera konfiguracyjnego z wykorzystaniem biblioteki Spring Cloud Config. Serwer charakteryzuje się czasem reakcji ponad 10 sekund większym od pozostałych badanych narzędzi. Związane jest to z innym mechanizmem powiadamiania, który charakteryzuje się mniejszym obciążeniem ze względu na brak konieczności ciągłego odpytywania o zmiany. Z drugiej strony zastosowanie kolejki wiadomości jako pośrednika do powiadamiania aplikacji o wprowadzonych zmianach znacznie wpływa na czas reakcji aplikacji.

Badania jakości interfejsu wypadły dla narzędzia Consul równie korzystnie. W każdym z obszarów aplikacja otrzymywała najwyższe wyniki. Współczynnik WUP wyniósł dla niego 4,23 co jest wynikiem o 0,2 lepszym od implementacji serwera konfiguracyjnego. Biblioteka zakłada wykorzystanie systemu kontroli wersji Git, jako interfejsu. W przypadku narzędzia Consul interfejs przystosowany edycji wartości konfiguracyjnych, poprzez przeznaczony do tego ekran. Wpływa to na większą intuicyjność oraz czytelność, której brakuje w systemach kontroli wersji ze względu na ich odmienne przeznaczenie. Najgorsze wyniki w tej sekcji otrzymał ZooKeeper, którego współczynnik WUP jest o 1,35 niższy od serwera Spring Cloud Config. Nie posiada wbudowanego graficznego interfejsu przez co praca z nim w kontekście scenariuszy okazała się najtrudniejsza, a jego interfejs najmniej intuicyjny. Trudności sprawiało użytkownikom również wyszukiwanie informacji w menu pomocy zawierającego polecenia wraz z opisem parametrów przez nie wykorzystywanych.

Literatura

[1] J. Wu, T. Wang, Research and Application of SOA and Cloud Computing Model, 2014 Enterprise Systems Conference, Shanghai (2014) 294-299, http://doi.org/10.1109/ES.2014.58.

[2] S. Kehrer, W. Blochinger, AUTOGENIC: Automated generation of self-configuring microservices, Department of Computer Science, Reutlingen University (2018) 35-46, http://dx.doi.org/10.5220/0006659800350046.

[3] O. Al-Debagy, P. Martinek, A comparative review of microservices and monolithic architectures, IEEE 18th International Symposium on Computational Intelligence and Informatics (2018) 149-154, http://dx.doi.org/10.1109/CINTI.2018.8928192.

[4] C. M. Pham, Z. Kalbarczyk, R. K. Iyer, V. Dogaru, R. Wagle, C. Venkatramani, An evaluation of zookeeper for high availability in system S, Proceedings of the 5th ACM/SPEC international conference on Performance engineering (2014) 209-217, http://doi.org/10.1145/2568088.2576801.

[5] P. Grzesik, D. Mrozek, Evaluation of key-value stores for distributed locking purposes, Beyond Databases, Architectures and Structures, Silesian University of Technology (2019) 70-81, https://doi.org/10.1007/978-3-030-19093-4_6 .

[6] Dokumentacja Spring Cloud w integracjach z innymi rozwiązaniami chmurowymi, https://spring.io/projects/spring-cloud, [30.01.2022].

[7] Dokumentacja Apache Kafka, https://kafka.apache.org/, [30.01.2022].

[8] M. Miłosz, Ergonomia systemów informatycznych, Lublin University of Technology, Lublin, 2014.

[9] A. Feizi, C. Y. Wong, Usability of user interface styles for learning a graphical software application, Faculty of Creative Multimedia, Malaysia (2012) 1089-1094, http://dx.doi.org/10.1109/ICCISci.2012.6297188.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 68–74
Received: 30 September 2022
Accepted: 26 October 2022

Researching users' knowledge in the Field of Instant Messengers Security

Badanie wiedzy użytkowników w zakresie bezpieczeństwa komunikatorów internetowych

Yevhenii Tsyliurnyk*, Oleksandr Tomenchuk, Grzegorz Kozieł

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

Abstract

Nowadays, many people contact other people through various types of social networks and instant messaging. However, these platforms are often not a secure way to exchange information among people. In recent times, most people have started to pay attention to the level of security offered by communicators and the technologies used in them, which results in the emergence of communicators focused primarily on safe communication. However, the question arises: "Are user actions not negatively affecting security?" Do users consciously choose instant messaging? Do they use modern methods to protect accounts? The conducted study confirmed the relationship between the user's actions and the security of communication. Additionally, the aspects most influencing the choice of communicator were found.

Keywords: security; instant messengers; privacy

Streszczenie

W dzisiejszych czasach wiele osób kontaktuje się z innymi ludźmi poprzez różnego typu portale społecznościowe i komunikatory internetowe. Platformy te nie są jednak często bezpiecznym sposobem wymiany informacji pomiędzy ludźmi. W ostatnich czasach większość osób zaczęła zwracać uwagę na poziom bezpieczeństwa oferowany przez komunikatory oraz wykorzystane w nich technologie, wskutek czego powstają komunikatory nastawione przede wszystkim na bezpieczną komunikację. Pojawia się jednak pytanie: „Czy działania użytkowników nie wpływają negatywnie na bezpieczeństwo?”. Czy użytkownicy świadomie wybierają komunikatory? Czy wykorzystują nowoczesne sposoby ochrony kont? Przeprowadzone badanie potwierdziło zależność między działaniami użytkownika, a bezpieczeństwem komunikacji. Dodatkowo zostały znalezione aspekty, najbardziej wpływające na wybór komunikatora.

Słowa kluczowe: bezpieczeństwo; komunikatory internetowe; prywatność

*Corresponding author

Email address: **yevhenii.tsyliurnyk@pollub.edu.pl** (Y. Tsyliurnyk)

1. Wstęp

Obecnie coraz więcej rozmów realizujemy za pomocą komunikatorów internetowych. Wykorzystujemy je zarówno w życiu prywatnym, jak i przy rozmowach służbowych. Przy komunikacji służbowej, pracodawcy często sami decydują, z jakiego narzędzia komunikacji chcą korzystać w firmie, kierując się przy tym przede wszystkim takimi aspektami jak bezpieczeństwo i wykorzystane technologie szyfrujące. Natomiast w życiu codziennym użytkownicy podczas wyboru komunikatora kierują się najczęściej popularnością wśród znajomych oraz dodatkowymi funkcjami takimi jak np. animowane emotki, możliwość prowadzenia rozmów wideo lub połączenia głosowego. Istotna jest również możliwość powiązania komunikatora z profilem na portalach społecznościowych.

Nasuwa się więc pytanie, czy dla użytkowników prywatnych istotny jest stopień bezpieczeństwa i czy wpływa to na wybór komunikatora do celów prywatnych? Aby zgłębić ten problem, postawiono 2 hipotezy robocze:

H1. Użytkownicy, dla których ważne jest bezpieczeństwo, przede wszystkim zwracają uwagę na zabezpieczenia zastosowane w komunikatorach, a dopiero później na interfejs oraz dostępną funkcjonalność.

H2. Użytkownicy dbający o swoje bezpieczeństwo używają dodatkowych funkcji komunikatorów, aby zwiększyć poziom bezpieczeństwa.

W celu zweryfikowania hipotez przeprowadzono ankietę na grupie 100 osób. Respondenci biorący udział w badaniu posiadali różny stopień wiedzy dotyczący komunikatorów oraz ich bezpieczeństwa. Połowa przebadanej grupy to studenci Politechniki Lubelskiej z kierunku Informatyka, więc możemy założyć, że ich wiedza w tym zakresie jest na wysokim poziomie. Druga połowa ankietowanych nie posiada wykształcenia kierunkowego, ale regularnie korzysta z komunikatorów. Tak dobrana grupa badawcza pozwoliła otrzymać wyniki odzwierciedlające prawdziwe grono użytkowników komunikatorów. Respondenci mogli wskazać dowolną liczbę komunikatorów, co pozwoliło dokładniej oszacować popularność każdego z dostępnych rozwiązań. Celem ankiety było sprawdzenie świadomości użytkowników o istniejących zagrożeniach, metodach zwiększenia swojego bezpieczeństwa oraz doświadczenia związanego z korzystaniem z komunikatorów internetowych.

2. Przegląd literatury

Publikacja została użyta jako przykład analizy porównawczej komunikatorów [1]. Ułatwiła ona sprecyzowa-

nie aspektów, na jakie trzeba zwrócić uwagę oraz polepszyła rozumienie współcześnie wykorzystywanych zabezpieczeń w komunikatorach. Praca opisuje jakie zagrożenia nadal występują w aplikacjach i jak ich kombinacje pozwalają złamać nawet szyfrowanie end-to-end [2]. Podobne problemy opisuje również publikacja [3]. Dla lepszego rozumienia zagrożeń musimy znać cele, jakie atakują hakerzy. Obecnie duża część ataków przeprowadzana jest nie tylko w celu zarobkowych, ale również dla pokazania swoich możliwości wpływu za pomocą szantażu na pewne grupy ludzi i ich świadomość. Znane są przypadki, w których hakerzy są ściśle związani z polityką, bezpieczeństwem narodowym i działaniami wojennymi. W pracach zostały opisane najbardziej rozpowszechnione motywy hakerów oraz to, w jaki sposób wybierają swoje cele i jakie strategie wykorzystują [4-5]. Książka „Computer security literacy: Staying safe in a digital world" pokazuje, że niemal każde działanie użytkownika sieci wpływa na jego bezpieczeństwo w Internecie [6]. Można to również wywnioskować z prac, które opisują czynniki wpływające na bezpieczeństwo systemów komputerowych [7-8]. Istnienie zagrożenia wynikającego z nieświadomych działań samych użytkowników jest obecnie dużym problemem, dlatego zadaniem twórców oprogramowania, oprócz doskonalenia bezpieczeństwa systemu, jest również poszerzanie wiedzy użytkowników o istniejących zagrożeniach i sposobach przeciwdziałania im.

W celu sprawdzenia świadomości użytkowników o istniejących zagrożeniach i wiedzy na temat cyberbezpieczeństwa została stworzona ankieta. Wyniki ankiety pozwolą na weryfikację poprawności założonych hipotez.

Książka ułatwia zrozumienie technologii, które już dawno są używane na rynku i obecnie stały się standardem szyfrowania [9]. Praca opisuje mechanizm kontroli uprawnień aplikacji, wykorzystany w OS Android i związane z nim problemy [10]. Jednym z wymienionych problemów jest nieumyślne nadanie wszystkich uprawnień, do których aplikacja żąda dostępu oraz brak świadomości użytkowników, jakie uprawienia dają dostęp do prywatnych danych.

Wiedza zaczerpnięta z wyżej wymienionych prac pozwoliła przygotować się do napisania artykułu, dobrać kryteria oceniania bezpieczeństwa badanych komunikatorów oraz prawidłowo zinterpretować wyniki z przeprowadzonej ankiety.

3. Omówienie oraz porównanie komunikatorów

Nikogo nie dziwi, że rynek komunikatorów internetowych staje się coraz większy i pojawiają się nowe aplikacje do komunikacji. Użytkownicy mają duży wybór dostępnych funkcji, dopasowanych do własnych potrzeb. Trudno opisać wszystkie funkcje każdego z komunikatorów, jednak na podstawie przeprowadzonej ankiety zostały przeanalizowane i omówione najczęściej wybierane aplikacje. Do analizy wybrane zostały najpopularniejsze [11] komunikatory, takie jak znane wszystkim: Facebook Messenger, Instagram, popularne głównie na wschodzie Telegram i Viber, oraz Microsoft Teams. Każdy komunikator jest unikalny względem wyglądu oraz dostępnych funkcji, lecz wszystkie posiadają wspólne podstawowe funkcje: wysłanie natychmiastowej wiadomości, możliwość dodania do wysyłanej wiadomości emotki, wysyłania plików graficznych, oraz możliwość połączenia głosowego lub wideo. Facebook Messenger pochodzi z największej sieci społecznościowej na całym świecie - Facebooka, na skutek czego cieszy się ogromną bazą użytkowników. Użytkownicy mogą wysyłać wiadomości, zdjęcia, filmy i inne pliki, a także reagować na wiadomości i posty znajomych. Aplikacja Facebook Messenger oferuje możliwość szyfrowania wiadomości oraz ich usuwania. Instagram należy do fotograficznych serwisów społecznościowych hostingu zdjęć, co znaczy, że komunikator pomimo wysyłania natychmiastowych wiadomości umożliwia użytkownikom edycję zdjęć i filmów, stosowanie do nich filtrów cyfrowych oraz udostępnianie ich w innych serwisach społecznościowych. W 2012 roku serwis został kupiony przez Facebook. Wielkim problemem Instagrama jest duża ilość fałszywych kont, wykorzystywanych do wysyłania spamu na platformie. Telegram to darmowy bazujący na chmurze obliczeniowej komunikator internetowy, który posiada funkcjonalność zbliżoną do mikroblogów. Użytkownicy mogą wysyłać wiadomości, zdjęcia, filmy oraz pliki dowolnego typu i rozmiaru, a także tworzyć kanały do nadawania wiadomości do nieograniczonej liczby odbiorców. Zaletą komunikatora Telegram jest szyfrowane metodą end-to-end połączeń głosowych i wideo. Jako opcję dodatkową można włączyć szyfrowanie wiadomości „punkt-punkt", co zapewnia jeszcze wyższy poziom bezpieczeństwa, jednak taka opcja nie jest dostępna dla rozmów grupowych oraz wersji desktopowej programu. Viber jest komunikatorem internetowym do prowadzenia rozmów telefonicznych. Wykorzystuje technologię Voice over IP (VoIP) przeznaczoną dla smartfonów oraz komputerów. Technologia ta jest rozwijana przez firmę Viber Media. Podobnie jak w przypadku omówionych wyżej komunikatorów użytkownicy Viber mogą przesyłać zdjęcia, filmy oraz pliki audio. Jedną z zalet tego komunikatora jest jego wieloplatformowość. Aplikacja kliencka jest dostępna na platformy Mac OS, Android, BlackBerry OS, iOS, Series 40, Symbian, Bada, Windows Phone, i Microsoft Windows. Wersja 64-bitowa dla systemów Linux jest dostępna w dwóch repozytoriach: jako .deb i .rpm przeznaczonych dla systemów Debian i Ubuntu oraz Fedora i openSUSE. Microsoft Teams jest usługą internetową opartą na chmurze, która zawiera w sobie narzędzia i usługi pozwalające na wygodną pracę zespołową. Funkcjonalność innych produktów firmy Microsoft, na przykład takich jak Microsoft Office czy Skype, które wchodzą
w skład Microsoft 365, jest bezpośrednio połączona z omawianym komunikatorem. Dzięki Microsoft Teams można pracować w trybie online w obrębie plików Excel, Word oraz PowerPoint. Opisane wyżej komunikatory internetowe różnią się nie tylko dostępnymi funkcjami, ale również podejściem twórców do bezpieczeństwa

i prywatności użytkowników. Wyniki analizy porównawczej komunikatorów przedstawiono w tabeli numer 1.

Tabela 1: Analiza porównawcza wybranych komunikatorów

Nazwa komunikatora	**Wykorzystane techniki zabezpieczające**
Facebook Messenger	Szyfrowanie podstawowe; szyfrowanie powiadomień technologią end-to-end jest wykorzystywane tylko w tajnej konwersacji; możliwość wysłania wiadomości znikających po pewnym czasie; dodatkową opcją jest możliwość włączenia powiadomień o nierozpoznanych logowaniach; możliwość włączenia uwierzytelniania dwuskładnikowego podczas logowania.
Instagram	Uwierzytelnianie dwuskładnikowe – dodatkowa opcja, która dodaje do warstwy logowania element uwierzytelniania, można ją włączyć w ustawieniach; możliwość wysłania zdjęć znikających po obejrzeniu; szyfrowanie podstawowe.
Telegram	Opiera się na protokole MTProto, który zapewnia zgodność zabezpieczeń z szybką dostawą i niezawodność przy połączeniach niskiej jakości; możliwość wysłania wiadomości znikających po pewnym czasie; szyfrowanie klient-serwer jest używane w czatach Telegram; specjalne sekretne czaty zabezpieczone szyfrowaniem end-to-end, które nie pozostawiają żadnych śladów na serwerach Telegram.
Viber	Podstawowe szyfrowanie podczas przesyłania danych; szyfrowanie end-to-end; możliwość wysłania „tajnych" wiadomości znikających po pewnym czasie; możliwość ustawienia kodu PIN, bez którego dostęp do wiadomości jest niemożliwy; funkcja ukrywania numeru telefonu.
Microsoft Teams	Do szyfrowania danych zostały wykorzystane technologie, będące standardami branżowymi, takie jak TLS i SRTP; regulowany jest dostęp do zespołu, poprzez kontrolowanie ustawień prywatności i roli gościa zespołu; technologia Advanced Threat Protection pozwala chronić użytkowników przed złośliwym oprogramowaniem ukrytym w plikach; usługa Cloud App Security identyfikuje i zapobiega podejrzanej aktywności w chmurze.

Analiza pokazała, że wszystkie badane komunikatory internetowe posiadają niezbędne szyfrowanie podstawowe. Rozwój technologii w dzisiejszych czasach powoduje, że szyfrowanie podstawowe jest bardzo proste do złamania przez hakerów, którzy potrzebują do tego coraz mniej czasu. Niektóre z wyżej wymienionych komunikatorów korzystają z szyfrowania end-to-end, co daje dużo większy poziom zabezpieczenia w porównaniu z podstawowym szyfrowaniem. Wielką zaletą szyfrowania end-to-end jest to, że tylko osoby komunikujące się mogą odczytać wiadomości w formie jawnej. Oznacza to, że przy korzystaniu z takiej formy szyfrowania wiadomość przekazywana jest bezpośrednio do finalnego odbiorcy w formie zaszyfrowanej, a po otrzymaniu wiadomości odbiorcy odszyfrowują ją samodzielnie. Ważnym elementem bezpieczeństwa jest również możliwość włączenia usuwania wiadomości po wybranym czasie.

Niestety wykorzystywane przez producenta technologie nie mogą na sto procent zagwarantować bezpieczeństwa naszych danych osobowych. Na bezpieczeństwo wpływa również zachowanie użytkownika, wykorzystywanie przez niego dodatkowych funkcji oraz kontrola nad dostępem do urządzenia. W celu sprawdzenia wpływu opisanych czynników na poziom bezpieczeństwa przeprowadzono badanie opisane w następnym rozdziale.

4. Badania

4.1. Metodyka badań

Badanie zostało przeprowadzone w dwóch etapach. Pierwszy polegał na porównaniu komunikatorów na podstawie ich specyfikacji oraz informacji dostępnych w Internecie. Wyniki porównania zostały przedstawione w rozdziale 3. Drugi etap polegał na przeprowadzeniu ankiety sprawdzającej wiedzę użytkowników w dziedzinie bezpieczeństwa użytkowania komunikatorów.

4.2. Dobór grupy badawczej

W badaniu wzięło udział 100 osób, 87 z nich w wieku 18-25 lat. Wśród uczestników 45 osób to studenci Politechniki Lubelskiej studiujące na kierunku Informatyka, pozostałych 55 respondentów to osoby studiujące na innych kierunkach i uczelniach, których życie codzienne nie jest związane z informatyką. Wszyscy respondenci na co dzień korzystają z komunikatorów internetowych, 94 osoby mają konta w co najmniej trzech z nich.

4.3. Przebieg badań

W celu uzyskania kompletnej informacji o posiadanej wiedzy, preferencjach oraz doświadczeniu respondentów zapytano ich o :

- wiedzę o bezpieczeństwie w Internecie,
- preferowane komunikatory,
- oczekiwany poziom bezpieczeństwa w różnych sytuacjach,
- aspekty wpływające na wybór komunikatora,
- korzystanie z dodatkowych funkcji komunikatorów,

- popularne sposobach podnoszenia poziomu bezpieczeństwa,
- doświadczenia związane z wyciekiem danych.

Wyniki ankiety poddano analizie, a uzyskane wnioski opisano w dalszej części artykułu. Dla lepszego rozumienia wyników, odpowiedzi na niektóre pytania analizowano dzieląc respondentów na grupy.

5. Dyskusja wyników

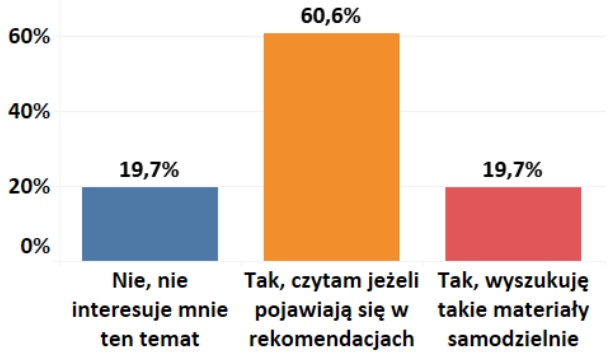
Ankieta została przeprowadzona na różnorodnej grupie respondentów, część z nich jest ściśle związana ze sferą IT, inni mają z nią do czynienia sporadyczne. Taka grupa pozwoliła na otrzymanie wyników charakteryzujących szerokie grono użytkowników komunikatorów.

Na początku zapytano ankietowanych o to, czy dbają o swoje bezpieczeństwo w Internecie. Takie pytanie wprost pozwala zobaczyć, ile osób jest świadomych istnienia zagrożeń oraz gotowych do przeciwdziałania im. Wyniki zaprezentowano na Rysunku 1.

Rysunek 1: Grupy użytkowników o różnym zaangażowaniu w dbałość o własne bezpieczeństwo w Internecie.

Zdecydowana większość (71%) odpowiedziała „Tak". Tylko 8% ankietowanych zadeklarowało, że nie zwraca uwagi na zagrożenia pochodzące z sieci. Część osób (26%) nie potrafiła konkretnie odpowiedzieć na to pytanie. Wyniki pokazują, że temat bezpieczeństwa jest znany większości osób i są one gotowi dokonywać pewnych działań, aby chronić swoją prywatność.

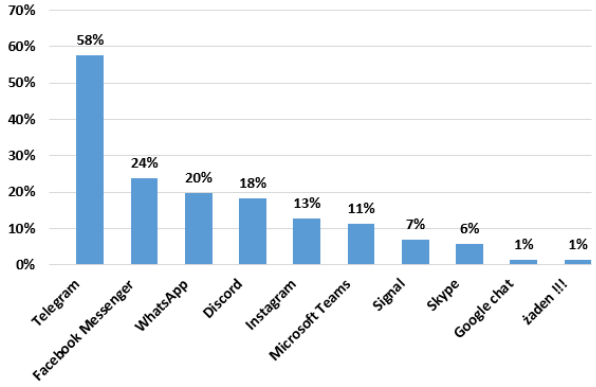
Następnie zapytano respondentów, czy interesują się informacjami na temat cyberbezpieczeństwa, wycieków danych lub ataków hakerskich. Na Rysunku 2 zaprezentowano odpowiedzi osób, które uważają, że dbają o swoje bezpieczeństwo w Internecie.



Rysunek 2: Zainteresowanie informacjami na temat bezpieczeństwa.

Jak widać na wykresie, tylko 20% respondentów omija ten temat, a zdecydowana większość osób badanych rozszerza swoją wiedzę w zakresie bezpieczeństwa. Warto zauważyć, że 60% badanych nie wyszukuje samodzielnie informacji, ale są zainteresowani artykułami, kiedy te pojawiają się w rekomendacjach lub jako reklama. Sugeruje to, że twórcy oprogramowania mogą znacząco wzbogacić wiedzę użytkowników w temacie bezpieczeństwa, dodając do swoich komunikatorów rekomendacje lub sugestie dotyczące bezpieczeństwa.

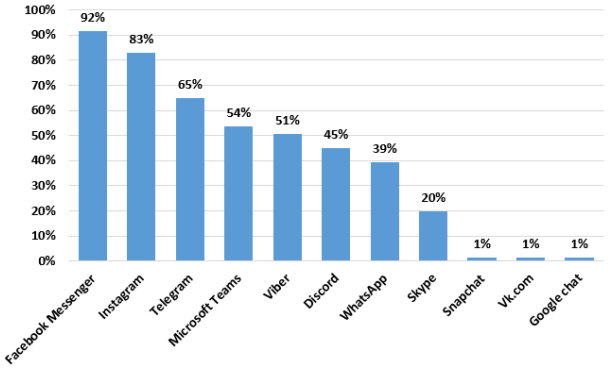
Respondentów zapytano również, który ze znanych im komunikatorów uważają za najbardziej bezpieczny. W tym pytaniu respondenci mogli wskazać dowolną liczbę komunikatorów, które one uważają za bezpieczne. Na Rysunku 3 przedstawiono wyniki pochodzące od grupy osób, które raportują, że dbają o swoje bezpieczeństwo w Internecie.



Rysunek 3: Jaki komunikator respondenci uważają za bezpieczny.

Widzimy, że większość pytanych osób uważa za najbardziej bezpieczny komunikator Telegram. Opierając się na przedstawionej powyżej analizie porównawczej komunikatorów, możemy stwierdzić, że jest to odpowiedź zbliżona do faktycznych danych. Kolejnym komunikatorem uważanym za bezpieczny jest FB Messenger, na który oddało głos 24% respondentów, niestety podczas analizy okazało się, że komunikator ten nie używa szyfrowania end-to-end, a w Internecie jest dużo informacji o wyciekach danych osobowych. Trzecie miejsce zajmuje WhatsApp, mimo tego, że ten komunikator również miał duże problemy z wyciekami danych. Ciekawe, że Viber, Signal i MS Teams mają dość przeciętne oceny, chociaż ich bezpieczność jest na wysokim poziomie, porównywalnym z Telegram-em.

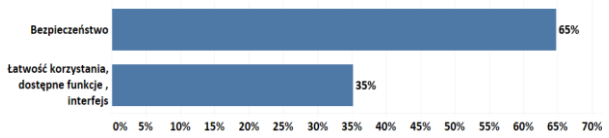
W następnym punkcie grupę respondentów raportujących zainteresowanie własnym bezpieczeństwem zapytano, z jakich komunikatorów korzystają, odpowiedzi przedstawiono na Rysunku 4.



Rysunek 4: Używane przez respondentów komunikatory.

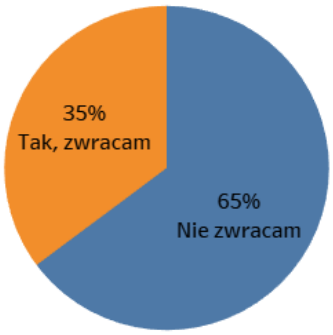
Jak widzimy, największą popularnością cieszą się komunikatory wywodzące się z sieci społecznościowych: Messenger i Instagram. Jest to spowodowane przede wszystkim tym, że wiele osób korzysta tylko z tych komunikatorów, co zmusza innych użytkowników do rozmów za pomocą tych aplikacji. Trzy kolejne miejsca zajmują Telegram, MS Teams oraz Viber, które są komunikatorami oferującymi wysoki stopnień bezpieczeństwa.

Aby określić czy użytkownicy dbający o swoje bezpieczeństwo w Internecie wybierają komunikatory ze względu na ich zabezpieczenia, zapytano ich: „Czy łatwość korzystania, dostępne funkcje oraz interfejs komunikatora mają dla Ciebie większe znaczenie niż bezpieczeństwo?”. Uzyskane wyniki zostały umieszczone na Rysunku 5.



Rysunek 5: Procent użytkowników wybierających łatwość korzystania, dostępne funkcje oraz interfejs komunikatora a nie bezpieczeństwo.

Około 65 procent użytkowników zadeklarowało, że bezpieczeństwo jest ważniejsze niż łatwość korzystania i dostępność dodatkowych funkcji w komunikatorze. Ważnym aspektem wpływającym na bezpieczeństwo komunikatora jest przesyłanie danych w postaci zaszyfrowanej. Respondentów zapytano, czy przy wyborze komunikatora zwracają uwagę na obecność szyfrowania przesyłanych danych oraz, czy rodzaj użytego szyfrowania ma dla nich znaczenie. Na Rysunku 6 i Rysunku 7 przedstawiono uzyskane wyniki.



Rysunek 6: Procent użytkowników zwracających uwagę na obecność szyfrowania przy wyborze komunikatora.

Z uzyskanych wyników możemy wywnioskować, że tylko 35% użytkowników przy wyborze komunikatora zwracają uwagę na obecność szyfrowania, a rodzaj użytego szyfrowania ma znaczenie tylko dla 8% odpytanych. Kolejnym punktem ważnym z punktu widzenia bezpieczeństwa jest wykorzystywanie aktualnych wersji oprogramowania oraz bieżące instalowanie aktualizacji bezpieczeństwa. W ankiecie zapytano użytkowników: „Czy dla Ciebie jest ważne korzystanie z najnowszej wersji komunikatora?”, odpowiedzi przedstawiono na Rysunku 8.

Rysunek 7: Procent użytkowników dla których ma znaczenie rodzaj użytego szyfrowania.

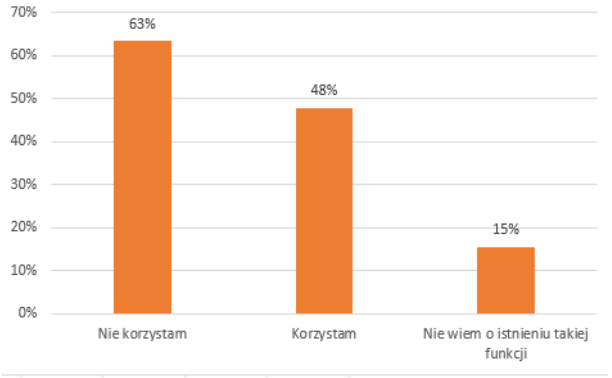
Rysunek 8: Procent użytkowników dla których ważne jest korzystanie z najnowszej wersji oprogramowania.

Jak i w poprzednich pytaniach, mimo zadeklarowanego dbania o swoje bezpieczeństwo, większość respondentów nie zwraca uwagi na aktualność używanego oprogramowania. Jednak w przypadku korzystania z komunikatorów na urządzeniach mobilnych domyślnie jest ustawiona automatyczna aktualizacja, co powoduje, że większość użytkowników i tak posiada aktualne wersje programów.

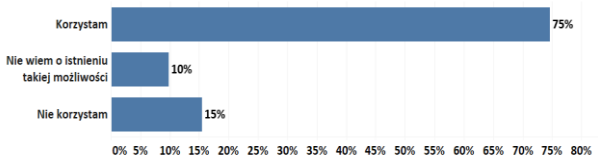
Uzyskane odpowiedzi pozwalają obalić hipotezę H1. Mimo deklarowanego dbania o swoje bezpieczeństwo większość użytkowników nie zwraca uwagi na podstawowe aspekty wpływające na zabezpieczenie komunikatora. Oprócz tego przy wyborze komunikatora użytkownicy kierują się jego popularnością i nadają przewagę komunikatorom wywodzącym się z sieci społecznościowych.

Niektóre komunikatory posiadają dodatkowe funkcjonalności pozwalające zwiększyć ochronę przesyłanych danych. Jednak problem polega na tym, że użytkownik musi wiedzieć o ich istnieniu oraz używać w odpowiednim momencie. Jedną z takich funkcji jest usuwanie wiadomości po określonym czasie. Rysunku 9 przedstawia jaki procent respondentów dbających o swoje bezpieczeństwo z niej korzysta.

Wśród wybranej grupy mniej niż połowa osób używa tej funkcji. W przypadku uzyskania fizycznego dostępu do urządzenia pozwoli to na przeczytanie wszystkich wysłanych wcześniej wiadomości. Następnym ważnym elementem bezpieczeństwa jest dwuetapowa autentyfikacja. Ten mechanizm pozwala chronić konto nawet w przypadku gdy hasło zostanie skompromitowane. Na Rysunku 10 przedstawiono procent użytkowników korzystających z dwuetapowej autentyfikacji.



Rysunek 9: Procent osób wykorzystujących funkcje usuwania wiadomości po określonym czasie.



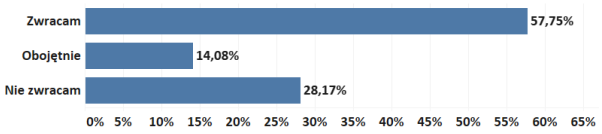
Rysunek 10: Procent osób korzystających z dwuetapowej autentyfikacji.

W tym przypadku możemy z pewnością konstatować, że zdecydowana większość grupy dba o bezpieczeństwo swojego konta. Dla potwierdzenia tego wniosku zapytano, czy dla różnych komunikatorów używane są różne hasła.

Rysunek 11: Procent użytkowników wykorzystujących różne hasła do różnych komunikatorów.

Diagram widoczny na Rysunku 11 pokazuje podobną tendencję do poprzedniego pytania, 76% użytkowników wykorzystują unikatowe hasła. Kombinacja unikatowego hasła i dwuetapowej autentyfikacji radykalnie zmniejsza ryzyko niepożądanego dostępu osób trzecich do naszego konta, i uniemożliwia wykonanie działań na koncie bez naszej wiedzy.

Często jednak oprócz samego komunikatora i zachowania użytkownika pojawia się trzeci czynnik wpływający na bezpieczeństwo informacji – system, na którym działa aplikacja. Najczęściej komunikatory używane są na urządzeniach mobilnych, na których zainstalowano wiele innych aplikacji. Podczas instalowania aplikacji użytkownik ma możliwość nadania jej pewnych uprawnień lub też odmówienia zgody na nadanie tychże uprawnień. Zwracanie uwagi na przydzielane uprawienia jest ważnym elementem bezpieczeństwa, gdyż niepożądany dostęp do systemu plików, kontaktów lub powiadomień może spowodować wyciek danych mimo wszystkich użytych w komunikatorze zabezpieczeń. Rysunek 12 przedstawia procent użytkowników zwracających uwagę na uprawnienia aplikacji podczas jej instalowania.

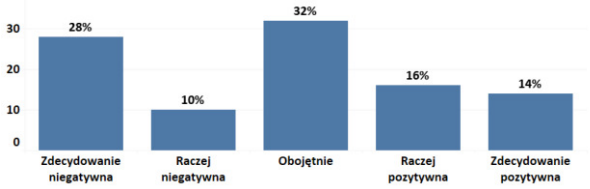


Rysunek 12: Procent użytkowników zwracających uwagę na uprawienia aplikacji pod czas instalowania.

Prawie 58% respondentów kontroluje, jakie uprawnienia są nadawane aplikacjom podczas instalowania. Jednak więcej niż 28% nie zwraca na to uwagi, czym naraża na niebezpieczeństwo nie tylko siebie, a również swoich współrozmówców.

Biorąc pod uwagę uzyskane wyniki, możemy częściowo potwierdzić hipotezę H2. Zdecydowana większość respondentów dobrze zabezpiecza swoje konta przed włamaniami. Również duża część użytkowników dbających o swoje bezpieczeństwo używa dodatkowych funkcji komunikatorów zwiększając swoje bezpieczeństwo. Jednak nadal pozostaje od 25 do 50 procent osób, zaniedbujących pewne aspekty bezpieczeństwa komunikatorów.

Dodatkowo zdecydowano się sprawdzić opinie użytkowników odnośnie możliwości analizy wysyłanych wiadomości w celach marketingowych. Respondentom zadano pytanie: „Jaka jest Twoja opinia na temat możliwej analizy wysłanych wiadomości przez twórcę oprogramowania?”. Respondenci udzielali odpowiedzi w pięciostopniowej skali - od „Zdecydowanie negatywnej” do „Zdecydowanie pozytywnej”. Wyniki zaprezentowano na Rysunku 13.



Rysunek 13: Opinia użytkowników na temat możliwej analizy wysłanych wiadomości.

Jak wynika z przeprowadzonego badania, liczna grupa respondentów odnosi się obojętnie do faktu analizy wysyłanych wiadomości, a kolejne 30% pozytywnie. Sama z siebie analiza wiadomości nie niesie dużego zagrożenia dla użytkowników, jednak świadczy o tym, że w konkretnym komunikatorze nie jest wykorzystywana metoda szyfrowania end-to-end, a wiadomości są w jakiś sposób otwierane i zapisywane na serwerach. Kombinacja tych dwóch czynników zwiększa ryzyko wycieku prywatnych danych w przypadku ataku na serwery komunikatora oraz robi go bardziej atrakcyjną celą dla hackerów.

6. Wnioski

Celem artykułu było zbadanie wiedzy użytkowników w zakresie bezpieczeństwa komunikatorów internetowych. Analizując dane przeprowadzonej ankiety, udało się zweryfikować pierwszą hipotezę i częściowo po-

twierdzić drugą. W procesie analizy wyników udało się wysunąć wnioski przydatne dla twórców komunikatorów oraz rekomendacje dla użytkowników.

Istotne jest to, że duża liczba użytkowników jest świadoma istniejących zagrożeń i gotowa wykonywać dodatkowe czynności, aby zwiększyć swoje bezpieczeństwo. Około 80% użytkowników zwracają uwagę na informacje o wyciekach danych, atakach hakerskich i wykrytych lukach bezpieczeństwa. Temat ten jednak nie jest dla większości użytkowników na tyle ciekawy, by samodzielnie poszukiwali wiedzy z tej dziedziny, więc zadaniem twórców oprogramowania jest dostarczanie użytkownikom informacji pozwalających na zwiększenie poziomu ich świadomości.

Istotny jest fakt, że mimo wiedzy o zagrożeniach oraz istnieniu lepiej zabezpieczonych komunikatorów użytkownicy dalej użytkują FB Messenger i Instagram, ze względu na ich popularność. Duża grupa użytkowników i dostępność niektórych znajomych wyłącznie na tych portalach w większości przypadków przeważa nad bezpieczeństwem danych. Natomiast trzy kolejne miejsca zajmują komunikatory z wysokim poziomem bezpieczeństwa, co świadczy o świadomym wyborze tych komunikatorów jako alternatywy i stopniowym przejściu na lepiej zabezpieczone kanały komunikacji.

Warto również zauważyć, że respondenci ceniący sobie bezpieczeństwo wykorzystują więcej dodatkowych możliwości podnoszenia poziomu własnego bezpieczeństwa i dobrze chronią swoje konta i urządzenia przed włamaniem.

Podsumowując, możemy powiedzieć, że wybór bezpiecznego komunikatora nie jest wystarczający do zapewnienia bezpiecznej komunikacji. Użytkownik, jak i jego współrozmówca muszą być świadomi istniejących zagrożeń i możliwości przeciwdziałania im, gdyż grono użytkowników i ich działania bezpośrednio wpływają na poziom bezpieczeństwa całego systemu.

Literatura

[1] J. Botha, C. Van't Wout, L. Leenen, A comparison of chat applications in terms of security and privacy, 18th European Conference on Cyber Warfare and Security (2019) 55-62.

[2] S. Prabhune, S. Sharma, End-to-end encryption for chat app with dynamic encryption key, 3rd International Conference on Advances in Computing Communication Control and Networking (2021) 1361-1366.

[3] J. Farnden, B. Martini, K. R. Choo, Privacy risks in mobile dating apps, 21st Americas Conference on Information Systems (2015) 118635-118647.

[4] S. Chng, H. Y. Lu, A. Kumar, D. Yau, Hacker types, motivations and strategies: A comprehensive framework, Computers in Human Behavior Reports 5 (2022) 100167 – 100175.

[5] K. Owen, M. Head, Motivation and demotivation of hackers in selecting a hacking task, Journal of Computer Information Systems 1 (2022) 1-15.

[6] D. Jacobson, J. Idziorek, Computer security literacy: Staying safe in a digital world, CRC Press, US, 2016.

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

[8] A. Kovacevic, N. Putnik, O. Toskovic, Factors related to cyber security behavior, IEEE Access 8 (2020) 125140-125148.

[9] M. Karbowski, Podstawy Kryptografii, Wydawnictwo Helion, 2006.

[10] Z. Fang, W. Han, Y. Li, Permission based android security: Issues and countermeasures, Computers and Security 43 (2014) 205-218

[11] Wyniki badania popularności komunikatorów za czerwiec 2022, https://pbi.org.pl/badanie-mediapanel/wyniki-badania-mediapanel-za-czerwiec-2022/, [29.09.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2022) 75–82
Received: 30 September 2022
Accepted: 14 October 2022

Performance analysis of working with databases with Spring and Symfony

Analiza wydajności pracy z bazami danych na przykładzie Spring i Symfony

Ewa Wieleba*, Bartłomiej Wieleba

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

Abstract

The article presents a comparative analysis of the efficiency of work with MySQL and PostgreSQL databases, using the popular Spring (Java) and Symfony programming frameworks. The research was carried out with the use of proprietary test applications that perform CRUD operations on a different number of records. The test results showed that the execution time of writing and deleting data using the Spring application is longer than when performing the same operations in Symfony. On the other hand, in the case of UPDATE and SELECT operations, the operation execution time with the Spring application turned out to be shorter than in the case of Symfony. The test results also confirmed that, regardless of the development framework, MySQL is less efficient than PostgreSQL while operating on 10 000 records except for DELETE, where MySQL combined with Symfony is the fastest.

Keywords: comparative analysis; Spring; Symfony; relational databases

Streszczenie

W artykule przedstawiono analizę porównawczą wydajności pracy z bazami danych MySQL i PostgreSQL, z wykorzystaniem popularnych szkieletów programistycznych Spring (Java) i Symfony. Badania przeprowadzono z wykorzystaniem autorskich aplikacji testowych, realizujących operacje typu CRUD na różnej liczbie rekordów. Wyniki testów wykazały, że czas wykonywania operacji zapisu i usuwania danych przy użyciu aplikacji Spring jest dłuższy niż przy wykonywaniu tych analogicznych operacji w Symfony. Natomiast w przypadku operacji UPDATE i SELECT, czas wykonywania operacji za pomocą aplikacji Spring okazał się krótszy niż w przypadku Symfony. Wyniki testów dowiodły, że niezależnie od szkieletu programistycznego MySQL jest mniej wydajny w stosunku do PostgreSQL przy wykonywaniu poleceń na dużej liczbie (10 000) rekordów dla wszystkich operacji poza operacją DELETE, gdzie MySQL z Symfony jest najszybszy.

Słowa kluczowe: analiza porównawcza; Spring; Symfony; relacyjne bazy danych

*Corresponding author

Email address: ewa.szewczak@pollub.edu.pl (E. Wieleba)

1. Wstęp

Wiele systemów informatycznych składa się z mikroserwisów i od nich zależy ich wydajność. Mikroserwisy odbierają i przetwarzają żądania, łączą się z innymi mikroserwisami lub bazami danych, na których wykonują wiele operacji. Liczba przetwarzanych operacji potrafi być ogromna w związku z czym rośnie obciążenie na mikroserwisach i bazach danych, co wiąże się z ich zawodnością i konsekwentnie z zawodnością systemów, które z nich korzystają.

Często zawodność wynika z opóźnień, kiedy współpracujące ze sobą serwisy zbyt długo czekają na odpowiedź na zadane żądania. Programiści starają się zapobiegać takim przypadkom, ale wszystko wciąż sprowadza się do jednego: szybkości obsługi żądań przez mikroserwisy i konsekwentnie, do szybkości obsługi zapytań przez bazy danych, z którymi współpracują te serwisy. Ważnym elementem decydującym o szybkości jest wybór języka programowania mikroserwisów oraz silnika baz danych. W niniejszym artykule przeprowadzono analizę porównawczą wydajności czasowej aplikacji współpracującej z danymi w relacyjnych bazach danych (MySQL i PostgreSQL), utworzonych w języku PHP (Symfony) i Java (Spring).

2. Przegląd literatury

Autorzy artykułu [1] badali możliwości optymalizacji zapytań SQL na silnikach baz danych Oracle, MySQL, Microsoft SQL Server i PostgreSQL z użyciem własnej aplikacji napisanej w języku Java z wykorzystaniem Hibernate. Z przeprowadzonych badań wynika, że wydajność baz danych zależy przede wszystkim od ilości danych przechowywanych przez bazę, stopnia skomplikowania wykonywanych zapytań oraz liczby zwracanych kolumn i rekordów w zapytaniu. Na szybkość wykonywania zapytań ma także wpływ funkcja grupująca i użycie klauzuli UNION.

Autorka artykułu [2] porównuje wydajność systemów bazodanowych MySQL, PostgreSQL i MS SQL w połączeniu z aplikacją internetową stworzoną w języku Java z Hibernate i JDBC. Autorka zwraca uwagę na powiązania między tabelami. W przypadku zapytań, w których występowało wykorzystanie powiązań między tabelami najlepiej wypadł silnik PostgreSQL, ale przy porównaniu czasów wykonywania zapytań dla pojedynczych tabel okazywało się, że są one do siebie

zbliżone zarówno dla silnika baz danych PostgreSQL, MySQL i MS SQL.

Autorzy artykułu [3] przy pomocy aplikacji Laravel (PHP) sprawdzali wydajność baz danych SQL Server, MySQL oraz PostgreSQL. Badania polegały na pomiarze czasu wykonywania prostych zapytań i operacji z użyciem konkatenacji kolumn oraz tabel. Zebrane wyniki pokazały, że w przypadku wykonywania zapytań dla małej liczby rekordów (do 1000) bardzo dobrze wypada baza MySQL.

Autorzy artykułu [4] porównywali dwa szkielety programistyczne PHP: Laravel i Symfony pod kątem architektury, organizacji kodu, czytelności dokumentacji. Badania wykazały, że oba szkielety doskonale sprawdzają się do tworzenia aplikacji internetowych. Symfony okazał się dobrym wyborem przy budowie rozbudowanych projektów i jest uważany za najbardziej stabilny szkielet programistyczny PHP, natomiast Laravel jest najbardziej popularnym szkieletem programistycznym do tworzenia aplikacji w języku PHP i jest najprostszy do nauki.

Autorzy artykułu [5] porównali silniki baz danych MySQL i PostgreSQL. Badania polegały na łączeniu się z bazą danych, tworzeniu tabel, następnie uzupełnieniu tabel danymi, wykonywaniu instrukcji SELECT, a następnie usuwaniu wszystkich danych z bazy. Polecenia były przeprowadzane na 5000, 10000, 15000, 20000, 25000, 30000, 35000 i 40000 rekordach. Wyniki wskazały, że operacje INSERT, SELECT i DELETE są wykonywane szybciej przy użyciu silnika bazy danych MySQL niż przy użyciu silnika baz danych PostgreSQL.

W artykule [6] autorzy przeprowadzili badania wydajności baz danych MySQL, MS SQL, PostgreSQL i Oracle uruchomionych na platformie Docker. Stworzyli do tego celu aplikację testową w języku PHP z użyciem szkieletu programistycznego Symfony z interfejsem graficznym napisanym w Vue.js. Do przeprowadzenia badań stworzyli w aplikacji klasy pozwalające na wykonanie pojedynczych operacji na bazach danych, następnie do tych klas napisali testy jednostkowe. Każdy test, uruchamiający odpowiednią klasę, został wykonany 100-krotnie. Autorzy ~~po przeglądzie literatury~~ założyli, że najszybsze są silniki Oracle i MS SQL. Uzyskane wyniki badań częściowo potwierdziły ich tezę, ponieważ w większości operacji najlepiej wypadał silnik Oracle, a trochę słabszy był MS SQL.

Autorzy artykułu [7] przeprowadzili badania ukierunkowane na pomiar szybkości wykonywania zapytań, obciążenia procesora i pamięci na dysku dla baz danych MySQL, PostgreSQL oraz Firebird na systemie operacyjnym Windows 10 Pro 64-bit. Badania polegały na powtórzeniu 10 razy tego samego scenariusza i obliczeniu średniej wartości pomiarów. Przeprowadzone badania dowiodły, że najmniejsze obciążenie procesora występuje przy użyciu bazy Firebird, natomiast baza MS SQL okazała się najszybsza, zajmują najmniej miejsca na dysku, ale bardzo obciążając procesor.

Autorzy artykułu [8] przeprowadzili badania wydajności dwóch szkieletów programistycznych PHP: Yii i CodeIgniter. Badania wykazały, że Yii jest szybszy w porównaniu z innymi frameworkami i autorzy polecili go do budowania dużych projektów, gdzie istotnym jest czas. Dodatkową jego zaletą jest prostota i automatycznie generowany kod kodu. Natomiast framework CodeIgniter jest łatwiejszy w instalacji, jednak z badań wynika, że może być trudniejszy do nauki, dlatego autorzy zalecają go bardziej doświadczonym programistom.

Autorzy artykułu [9] przeprowadzali badania mające na celu porównanie frameworków: Symfony2 i PhalconPHP pod kątem ich popularności, dostępności dokumentacji, możliwości i narzędzi przyśpieszających proces budowy oprogramowania. Badania polegały na wysyłaniu 2000 żądań przy ustanowionych 10 jednoczesnych połączeniach. Badania wykazały, że Phalcon i Symfony2, mają podobne możliwości routingu, użycia adnotacji i wyrażeń regularnych, ale Symfony2 zapewnia większy wybór formatów plików takich jak XML lub YAML. Silnik szablonów Twig w Symfony2 pozwala na lepszą organizację struktury szablonów. Z kolei silnik szablonów Volt w Phalcon jest bardziej przyjazny dla programistów zaczynających pracę z silnikami szablonów i ładuje się szybciej niż Twig.

W zależności od silnika baz danych, od zastosowanego języka programowania po stronie serwera i od wybranego frameworka może zależeć efektywność działania aplikacji internetowej.

3. Cel badań

Celem badań podjętych w niniejszym artykule jest porównanie wydajności pracy z relacyjnymi bazami danych MySQL i PostgreSQL na przykładzie Spring i Symfony.

Postawiono następujące tezy badawcze:

- Czas wykonywania operacji SELECT, DELETE, UPDATE, INSERT przy użyciu Spring jest krótszy niż przy wykonywaniu tych samych operacji przy użyciu szkieletu programistycznego Symfony w połączeniu z bazami danych PostgreSQL i MySQL.
- Czas wykonywania operacji SELECT dla silnika baz danych MySQL jest krótszy niż dla silnika PostgreSQL dla mniejszej liczby danych.

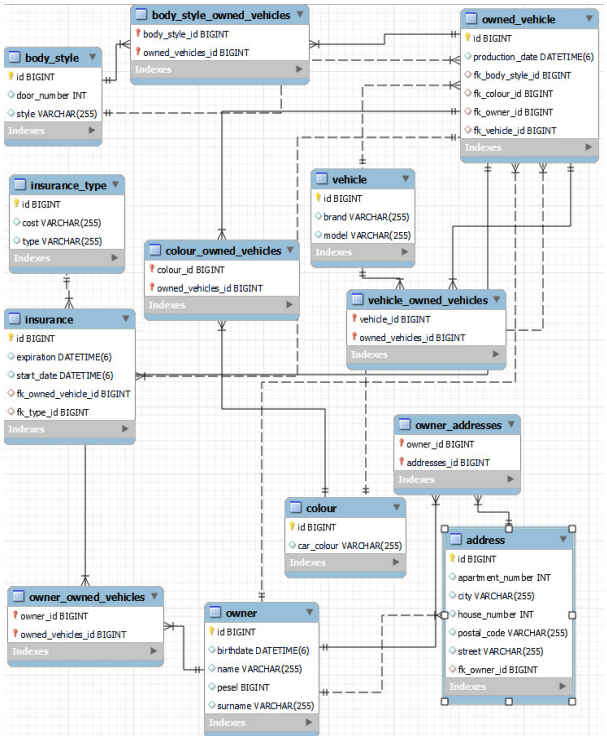
W artykule użyto szkieletu programistycznego Symfony [12] i Spring [11], ponieważ według rankingów [14-15] są one najbardziej popularnymi szkieletami programowania odpowiednio dla języka Java i PHP. Podobnie w przypadku baz danych, wybrano MySQL [19] i PostgreSQL [13] jako jedne z najpopularniejszych silników bazodanowych [16]. W artykule przedstawiono analizę porównawczą dla dwóch popularnych szkieletów programistycznych języków Java i PHP.

4. Metoda i środowisko badań

Do realizacji badań opracowano 3 aplikacje testowe:

- Java ze Spring – aplikacja typu REST,
- PHP z Symfony – aplikacja typu REST,
- aplikacja do pomiaru czasu wykonywania operacji, przeznaczona do zbierania wyników badań w języku Java.

Operacje na bazie danych realizowano za pomocą narzędzi ORM – Hibernate w Spring oraz Doctrine w Symfony. Schemat bazy danych samochodów, na której wykonano testy, przedstawiony jest na Rys. 1. Nazwy tabel i pól przygotowano w j. angielskim, aby zachować zgodność językową z nazwami klas, wykorzystywanych w kodzie aplikacji. .



Rysunek 1: Schemat ERD bazy danych.

Testy polegały na wykonywaniu zapytań zapisywania rekordów do tabeli *owned_vehicle*, aktualizowania danych takich jak marka, model, początek i koniec okresu na który ubezpieczono pojazd, ilość drzwi jaką posiada pojazd, styl karoserii pojazdu, adres właściciela pojazdu i kolor pojazdu, usuwania kaskadowego rekordów z tabeli *owned_vehicle*, dodawania rekordów do tabeli *owned_vehicle* . Testy realizowano dla różnej liczby rekordów (100, 1000 i 10000). Każda z operacji była powtarzana 50 razy, Czas wykonania operacji mierzony był poprzez obliczenie różnicy czasu przed i po wykonaniu danej operacji.

Do testów został użyty komputer o specyfikacji:

- procesor Ryzen 4600h,
- 24GB RAM DDR 4 3200MHz,
- dysk SSD,
- system operacyjny Windows 10 Home Edition.

Aplikacje testowe zaimplementowano w językach:

- Java (JDK 18) i Spring Boot 2.7.0,
- PHP 8.1.6 i Symfony 6.1.0.

5. Elementy implementacji aplikacji testowych

Przykładowe instrukcje UPDATE i SELECT dla Spring zostały pokazane na przykładach 1 i 2. Analogiczne operacje zostały zaimplementowane za pomocą Doctrine w Symfony.

Przykład 1: Instrukcja update w Spring Boot

```
@Query(value = "update owned_vehicle set
fk_colour_id =?1, fk_body_style_id =?2 where
  id=?3", nativeQuery = true)
void updateOwnedVehicleById(Long colour,
   Long bodyStyle, Long id);

@Transactional
public void updateOwnedVehicle(OwnedVehicleDto dto)
{
  ownedVehicleRepository
  .updateOwnedVehicleById(dto.getFkColourId(),
    dto.getFkBodyStyleId(), dto.getId());
}
```

Przykład 2: Instrukcja SELECT z JOIN w Spring Boot

```
@Query(value = "select " + "v.brand, " +
       "v.model, " + "i.start_date, " +
        "i.expiration, " + "bs.door_number, " +
        "bs.style, " + "a.city, " +
        "a.street, " + "a.apartment_number, " +
        "a.house_number, " + "c.car_colour " +
        "from owned_vehicle " +
        "union all vehicle v on v.id =
         owned_vehicle.fk_vehicle_id " +
        "union all insurance i on
       owned_vehicle.id = i.fk_owned_vehicle_id " +
       "union all body_style bs on bs.id =
        owned_vehicle.fk_body_style_id " +
       "union all address a on
      owned_vehicle.fk_owner_id = a.fk_owner_id " +
           "union all colour c on c.id =
   owned_vehicle.fk_colour_id", nativeQuery = true)

List<OwnedVehicleResponse> selectWithJoin();
//-----------------------------------------------
public List<OwnedVehicleResponse> selectWithJoin()
{
  return
     ownedVehicleJoinRepository.selectWithJoin();
}
```

6. Wyniki badań

W tabeli 1 przedstawiono wyniki porównania średnich czasów wykonywania operacji do pobierania i aktualizacji danych dla systemów bazodanowych MySQL i PostgreSQL w aplikacji Spring Boot.

W tabeli 2 przedstawiono analogiczne wyniki dla aplikacji Symfony.

6.1. Operacje dla 100 rekordów

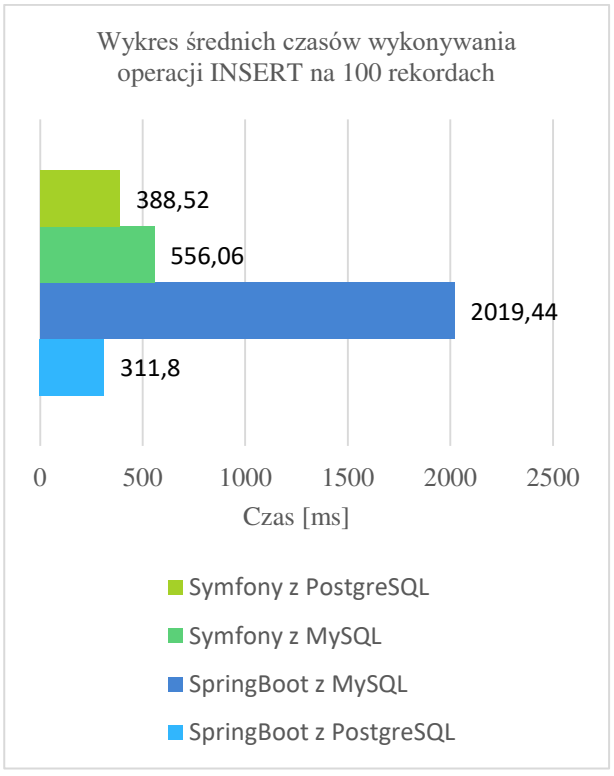
Średnie czasy wykonywania operacji na bazie danych dla 100 rekordów przedstawiono na rysunkach 2-5.

Tabela 1: Średnie czasy wykonywania zapytań w SpringBoot

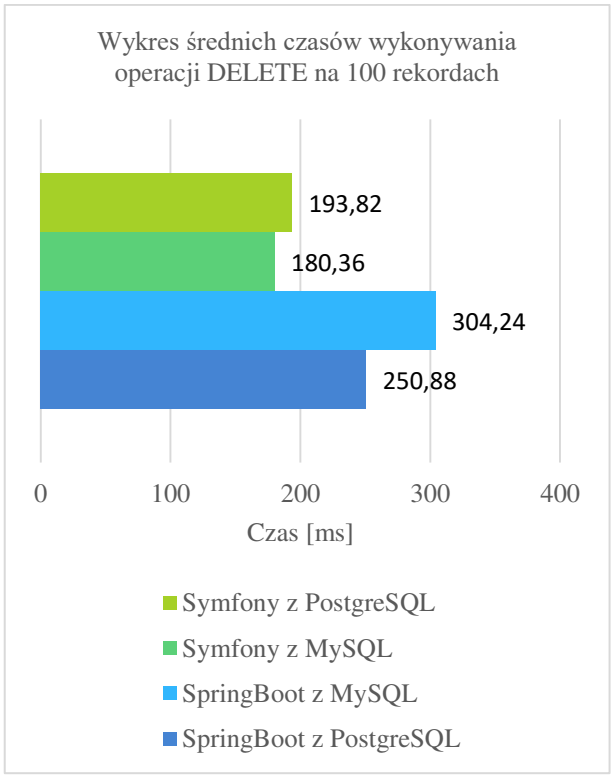
Tabela średnich czasów wykonywania zapytań w SpringBoot w PostgreSQL i MySQL						
	SpringBoot z PostgreSQL			SpringBoot z MySQL		
	100 rekordów	1000 rekordów	10000 rekordów	100 rekordów	1000 rekordów	10000 rekordów
INSERT	311,8 ms	2000,22 ms	18661,44 ms	2019,44 ms	19842,42 ms	204407,14 ms
DELETE	250,88 ms	2000,24 ms	18511 ms	304,24 ms	2711,88 ms	27825,04 ms
UPDATE	8,16 ms	20,2 ms	143,52 ms	11,4 ms	30,82 ms	232,92 ms
SELECT	31,58 ms	41,94 ms	180,42 ms	39,62 ms	55,78 ms	219,62 ms

Tabela 2: Średnie czasy wykonywania zapytań w Symfony

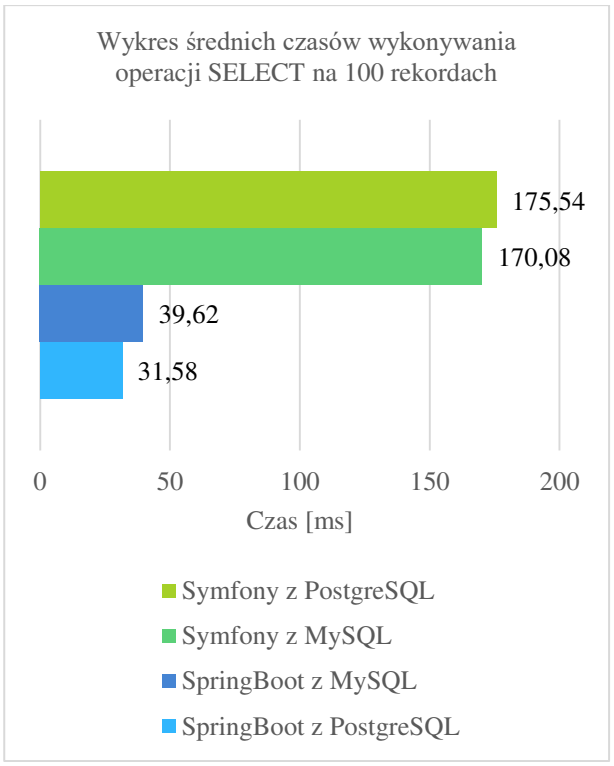
Tabela średnich czasów wykonywania zapytań w Symfony w PostgreSQL i MySQL						
	Symfony z PostgreSQL			Symfony z MySQL		
	100 rekordów	1000 rekordów	10000 rekordów	100 rekordów	1000 rekordów	10000 rekordów
INSERT	388,52 ms	1835,84 ms	17403,34 ms	556,06 ms	3373,82 ms	32157,26 ms
DELETE	193,82 ms	225,22 ms	627,76 ms	180,36 ms	193,96 ms	331,02 ms
UPDATE	175,98 ms	191,34 ms	376,4 ms	174,42 ms	196,9 ms	389,18 ms
SELECT	175,54 ms	187,24 ms	355,72 ms	170,08 ms	185,36 ms	378,48 ms



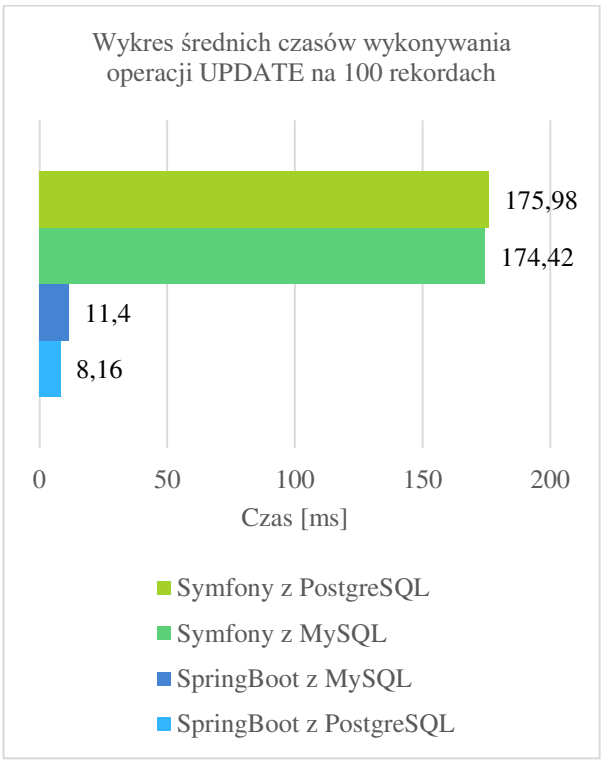
Rysunek 2: Wykres średnich czasów wykonania operacji INSERT na 100 rekordach.



Rysunek 3: Wykres średnich czasów wykonywania operacji DELETE na 100 rekordach.



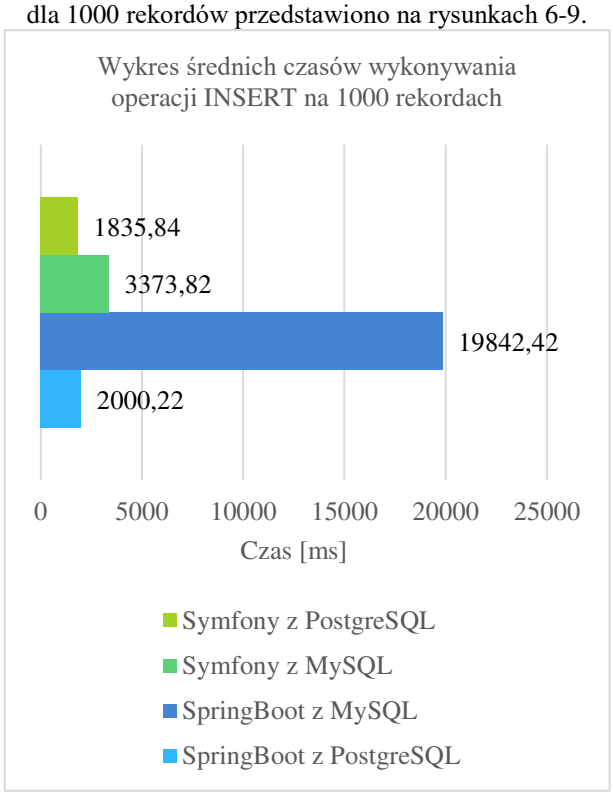
Rysunek 4: Wykres średnich czasów wykonywania operacji SELECT na 100 rekordach.



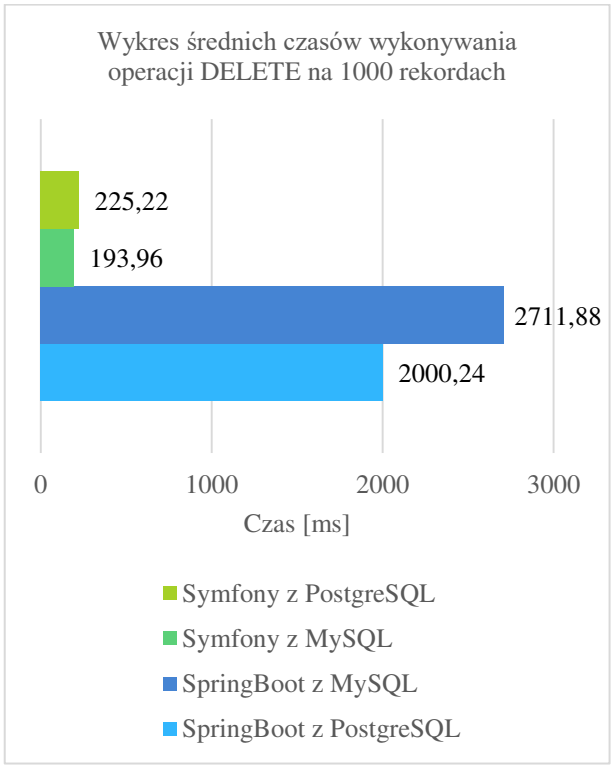
Rysunek 5: Wykres średnich czasów wykonywania operacji UPDATE na 100 rekordach.

6.2. Operacje dla 1000 rekordów

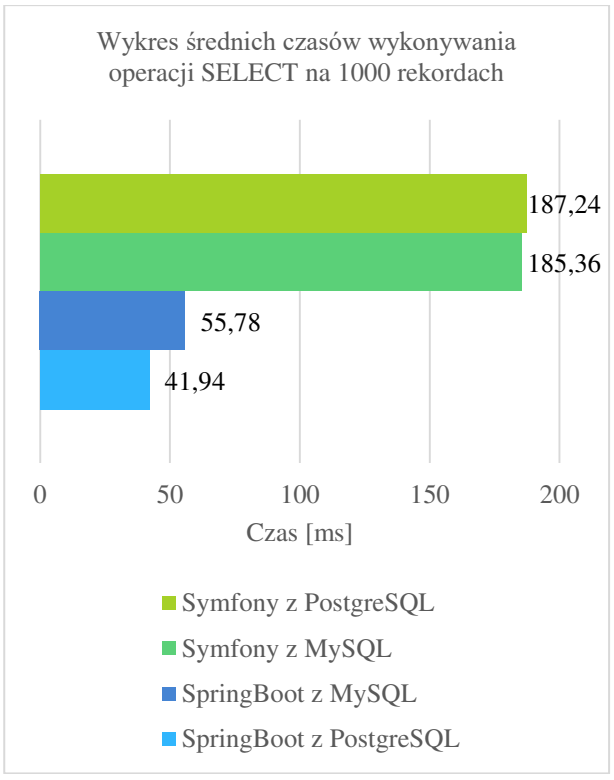
Średnie czasy wykonywania operacji na bazie danych dla 1000 rekordów przedstawiono na rysunkach 6-9.



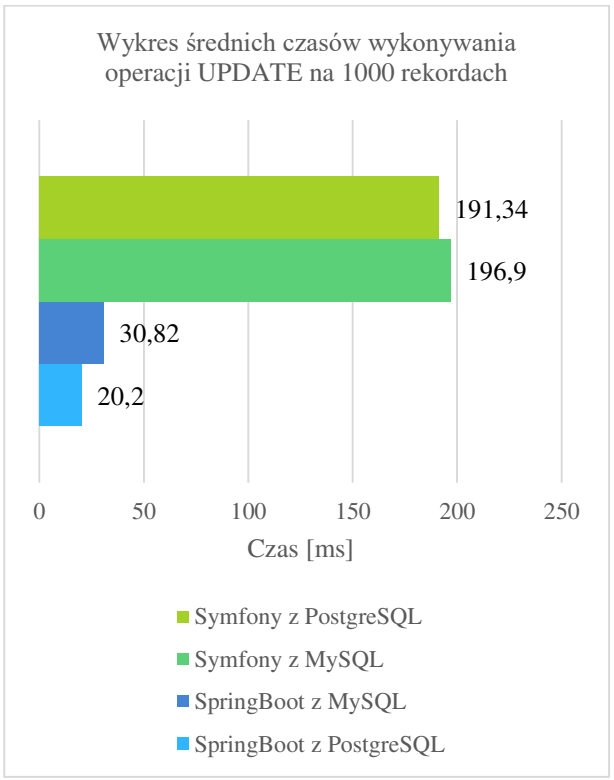
Rysunek 6: Wykres średnich czasów wykonywania operacji INSERT na 1000 rekordach.



Rysunek 7: Wykres średnich czasów wykonywania operacji DELETE na 1000 rekordach.



Rysunek 8: Wykres średnich czasów wykonywania operacji SELECT na 1000 rekordach.

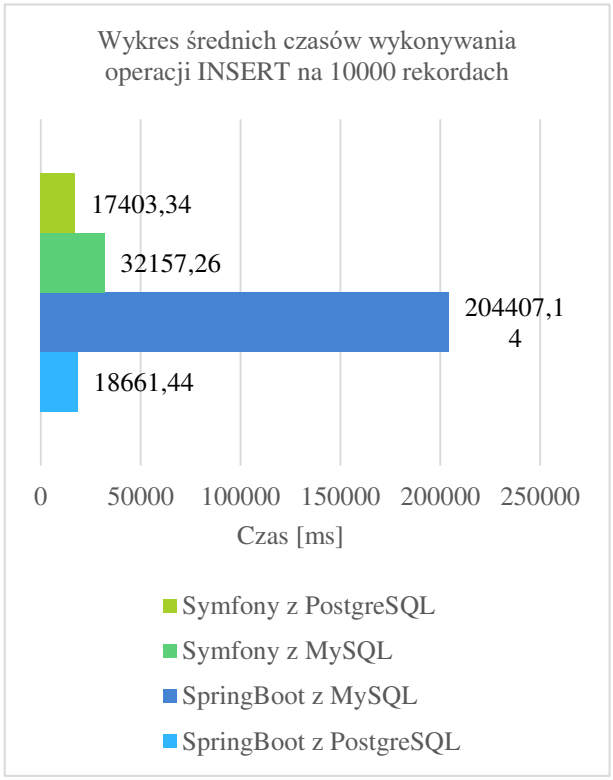


Rysunek 9: Wykres średnich czasów wykonywania operacji UPDATE na 1000 rekordach.

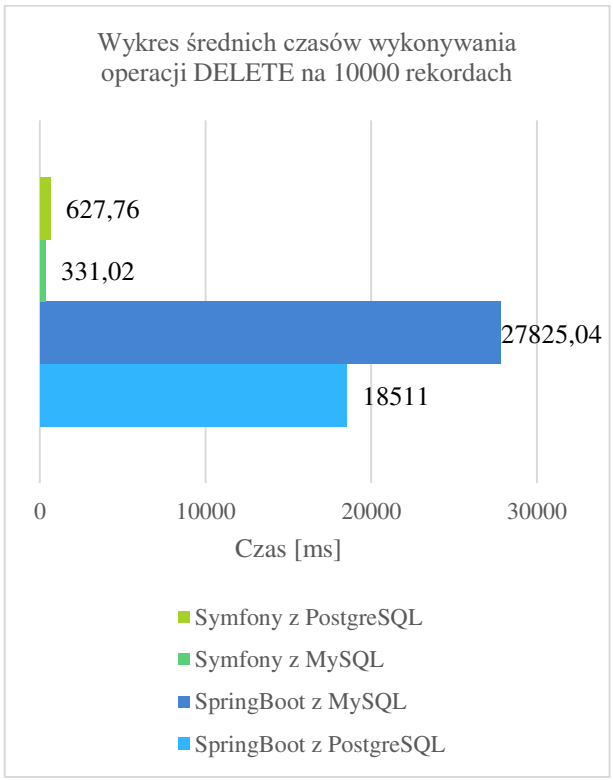
6.3. Operacje dla 10000 rekordów

Średnie czasy wykonywania operacji na bazie danych dla 10000 rekordów przedstawiono na rysunkach 10-13.

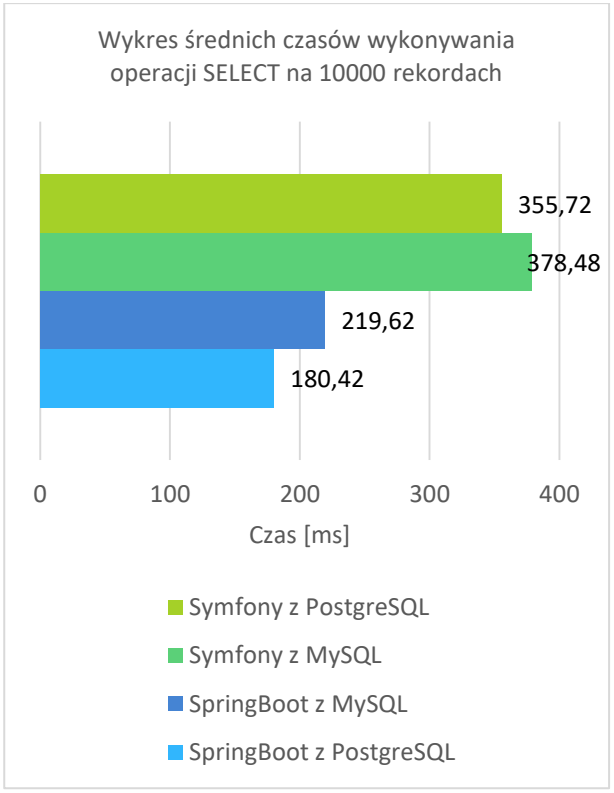
.



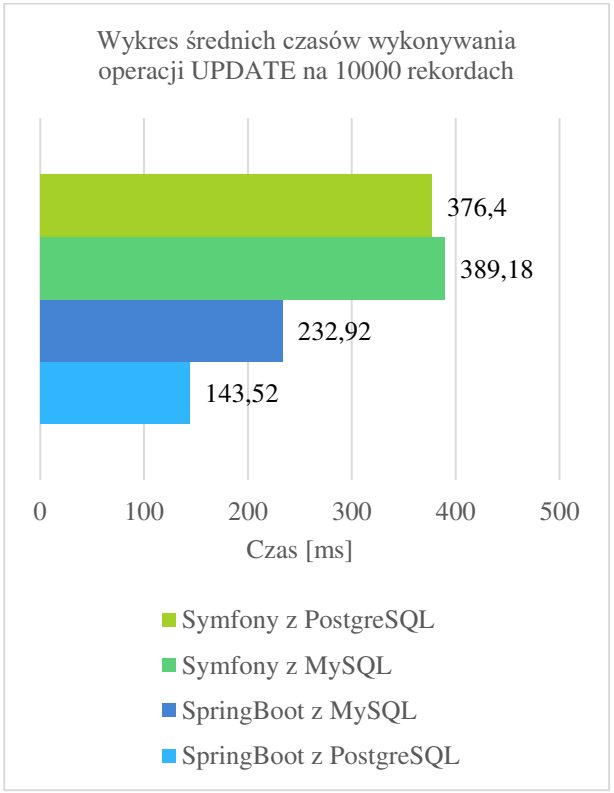
Rysunek 10: Wykres średnich czasów wykonywania operacji INSERT na 10000 rekordach.



Rysunek 11: Wykres średnich czasów wykonywania operacji DELETE na 10000 rekordach.



Rysunek 12: Wykres średnich czasów wykonywania operacji SELECT na 10000 rekordach.



Rysunek 13: Wykres średnich czasów wykonywania operacji UPDATE na 10000 rekordach.

7. Wnioski

Zebrane w tabeli 1 dane wskazują, że w połączeniu z frameworkiem Spring Boot, silnik PostgreSQL jest zdecydowanie szybszy od silnika MySQL. Najbardziej widać to przy wykonywaniu operacji INSERT, gdzie czas wykonywania operacji dla 10000 rekordów w PostgreSQL jest zdecydowanie krótszy niż w przypadku silnika MySQL. Znacząca różnica występuje również dla operacji DELETE, zwłaszcza dla 10 000 rekordów, gdzie czas wykonywania operacji dla PostgreSQL jest znacznie krótszy niż w przypadku MySQL.

Nieco mniejsza różnica występuje przy porównywaniu czasów operacji aktualizacji i pobierania danych (tabela 1). Tutaj również szybszy okazał się PostgreSQL. Dla 10 000 rekordów przy użyciu aplikacji napisanej w języku Java w SpringBoot operacja UPDATE wykonuje się o około 38,4% krócej dla PostgreSQL niż w przypadku MySQL i operacja SELECT o około 17,8% krócej.

Wyniki badań z tabeli 2 dla Symfony wskazują, że czas trwania operacji INSERT jest zdecydowanie krótszy dla bazy PostgreSQL niż MySQL. Dla pozostałych operacji można zaobserwować, że czas zależy od liczby rekordów. W Symfony z MySQL czasy trwania operacji DELETE, UPDATE i SELECT na 100 rekordach są krótsze niż dla analogicznych zapytań w bazie PostgreSQL. Natomiast czas trwania operacji DELETE, UPDATE i SELECT na 1000, 10000 rekordach jest krótszy dla przypadku Symfony z PostgreSQL.

Podsumowując silniki baz danych MySQL i PostgreSQL w połączeniu z aplikacją Spring można dojść do wniosku, że silnik PostgreSQL jest wydajniejszy od silnika MySQL w przypadku wykonywania operacji na dużej liczbie rekordów.

Biorąc pod uwagę wszystkie operacje przeprowadzone na 100, 1000 i 10000 rekordach, czasy wykonania operacji dla MySQL i Spring są dłuższe w porównaniu do innych wariantów aplikacji. Szczególnie można to zaobserwować na 10000 rekordach przy operacji INSERT (Rys 10) i DELETE (Rys 11) gdzie średni czas wykonania operacji dodawania i usuwania nowego rekordu do bazy PostgreSQL jest zdecydowanie krótszy (Tabela 1, Tabela 2, Rys. 10, Rys. 11). Mniejsza różnica występuje przy porównywaniu średnich czasów wykonywania operacji UPDATE i SELECT, jednak wciąż szybszy okazał się silnik PostgreSQL (Rys 13, Rys 12).

Wyniki badań przeprowadzonych przy użyciu aplikacji Symfony, czas wykonywania operacji INSERT niezależnie od liczby rekordów, krótszy jest przy użyciu bazy danych PostgreSQL. W przypadku pozostałych operacji: UPDATE, SELECT, DELETE w przypadku wykonania operacji na 100 rekordach, czas zapytania jest krótszy dla bazy MySQL. Natomiast przy wykonywaniu operacji na większej ilości danych zdecydowanie lepiej sprawdza się baza danych PostgreSQL (Tabela 2, Rys 3, Rys 4, Rys 5).

Na podstawie przeprowadzonej analizy porównawczej obalono słuszność postawionych tez, że „Czas wykonywania operacji SELECT, DELETE, UPDATE, INSERT przy użyciu szkieletu programistycznego Spring jest krótszy niż przy wykonywaniu tych samych operacji przy użyciu szkieletu programistycznego Symfony w połączeniu z bazami danych PostgreSQL

i MySQL". Nie potwierdziła się też teza, że „Czas wykonywania operacji SELECT dla silnika MySQL jest krótszy niż dla silnika PostgreSQL przy operowaniu na mniejszej ilości danych.".

Szkielet programistyczny Spring okazał się wydajniejszy od Symfony w połączeniu zarówno z bazą PostgreSQL jak i MySQL dla operacji SELECT i UPDATE (Tabela 1, Rys 4, Rys 5, Rys 8, Rys 9, Rys 12, Rys 13). Natomiast porównując czas wykonywania operacji INSERT i DELETE zdecydowanie wydajniejszy okazał się Symfony (Tabela 2, Rys 2, Rys 3, Rys 6, Rys 7, Rys 10, Rys 11).

Zebrane wyniki udowadniają, że czas wykonywania operacji SELECT dla silnika baz danych MySQL jest krótszy niż dla silnika PostgreSQL, w połączeniu z aplikacją Symfony niezależnie od liczby rekordów. Natomiast czas wykonywania polecenia SELECT przy użyciu aplikacji Spring jest dłuższy w połączeniu z bazą danych MySQL niż PostgreSQL (Tabela 1, Tabela 2).

Na podstawie przeprowadzonych badań można stwierdzić, że budując aplikacje, w których główną rolę grać będzie pobieranie dużych ilości danych i ich modyfikowanie, lepszym rozwiązaniem będzie wybór języka Java i Spring w połączeniu z bazą danych PostgreSQL. Rezultaty częściowo zgadzają się z wynikami badań przeprowadzonymi w artykule [2], gdzie PostgreSQL był wydajniejszy dla większych zestawów danych jednak badania przeprowadzone w tym artykule wskazują, że jest też wydajniejszy dla mniejszych zestawów danych (100 rekordów). Wyniki przeprowadzonych badań jednak zaprzeczają wynikom uzyskanym w artykule [5], PostgreSQL jest wydajniejszy niż MySQL.

Trudno też jest jednoznacznie wskazać najlepszą konfigurację.

Literatura

[1] P. Rymarski, G. Kozieł, Analiza możliwości optymalizacji zapytań SQL, Journal of Computer Sciences Institute 19 (2021) 151–158.

[2] K. Lachewicz, Analiza wydajności systemów bazodanowych: MySQL, MS SQL, PostgreSQL w kontekście aplikacji internetowych, Journal of Computer Sciences Institute 14 (2020) 94–100.

[3] R. Wodyk, M. Skublewska-Paszkowska, Porównanie wydajności relacyjnych baz danych SQL Server, MySQL oraz PostgreSQL z zastosowaniem aplikacji webowej i frameworku Laravel, Journal of Computer Sciences Institute 17 (2020) 358–364.

[4] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi, A. E. Yamami, A comparative study of laravel and symfony PHP frameworks, International Journal of Electrical and Computer Engineering 9 (2019) 704-712.

[5] S. Andjelic, S. Obradovic, B. Gacesa, A performance analysis of the dbms – mysql vs postgresql, Komunikacie 10 (2008) 53-57.

[6] R. Kleweka, W. Truskowski, M. Skublewska-Paszkowska, Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji, Journal of Computer Sciences Institute 16 (2020) 279–284.

[7] S. Stets, G. Kozieł, Comparative analysis of databases working under the control of Windows system, Journal of Computer Sciences Institute 13 (2019) 298–301.

[8] S.T Ali, J. Long, Quality Evaluation of PHP Frameworks, International Journal of Scientific & Engineering Research 10 (2019) 1454-1458.

[9] N. Prokofyeva, V. Boltunova, Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems, Procedia Computer Science 104 (2017) 51 – 56.

[10] Baza danych PostgreSQL, https://vavatech.pl/technologie/bazy-danych/postgresql, [06.06.2022].

[11] Dokumentacja SpringBoot, https://spring.io/projects/spring-boot , [07.06.2022].

[12] Dokumentacja Symfony, https://symfony.com/doc/current/index.html, [15.09.2022].

[13] Dokumentacja PostgreSQL, https://www.postgresql.org/docs/, [15.09.2022].

[14] Najpopularniejsze szkielety programistyczne do tworzenia aplikacji przy użyciu języka PHP w 2022 roku, https://www.linkedin.com/pulse/2022-most-popular-php-frameworks-infogenlabsinc/, [15.09.2022].

[15] Najpopularniejsze szkielety programistyczne do tworzenia aplikacji przy użyciu języka Java, https://www.geeksforgeeks.org/top-10-most-popular-java-frameworks-for-web-development/, [15.09.2022].

[16] Ranking popularności silników baz danych, https://db-engines.com/en/ranking, [12.10.2022].

[17] Dokumentacja szkieletu programistycznego Hibernate, https://hibernate.org/orm/documentation/6.1/, [12.10.2022].

[18] Dokumentacja biblioteki PHP Doctrine, https://symfony.com/doc/current/doctrine.html, [12.10.2022].

[19] Dokumentacja silnika baz danych MySQL, https://dev.mysql.com/doc/, [12.10.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2022) 83–87
Received: 5 October 2022
Accepted: 25 January 2023

The analysis of Blender open-source software cloth simulation capabilities

Analiza możliwości programu Blender pod kątem symulowania tkanin

Wojciech Kogut

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

Abstract

This article explores Blender open-source software capabilities in the area of cloth simulations. Simulation performance with different scene complexity and quality settings is tested with a script that automates the testing. System resource utilization is measured and the appearance of visual artifacts is taken into consideration. Cloth simulation features of Blender are compared to commercial software.

Keywords: Computer graphics; Blender; cloth simulation; physics simulation

Streszczenie

Ten artykuł stanowi analizę otwarto-źródłowego programu Blender w dziedzinie symulacji tkanin. Wydajność przeprowadzania symulacji jest badana dla różnej złożoności scen oraz różnych ustawieniach jakości przy pomocy skryptu automatyzującego pomiary. Użycie zasobów systemu jest mierzone, brane pod uwagę jest również występowanie artefaktów wizualnych. Funkcjonalność Blendera pod kątem symulacji tkanin jest porównywana do oprogramowania komercyjnego.

Słowa kluczowe: Grafika komputerowa; Blender; symulacje tkanin; symulacje fizyki

Email address: **wojciech.kogut@pollub.edu.pl**

1. Introduction

Due to the movie industry shift from practical effects to computer generated imagery (CGI) and video games becoming more and more visually complex, as well as other use cases, e. g. de-sign mockups or architectural visualisation, there is a great focus in the field of 3D computer graphics in chasing what is known as „photo-realism". The goal is to make rendered images indistinguishable from real photographs, or, in the case of moving pictures, real videos. One area that is crucial in achieving that, especially when it comes to motion pictures is physics simulation. There are several types of physics simulation – there are fluid simulation, soft body physics, cloth simulation etc. In recent years, simulations that look sufficiently convincing have been achieved, but it comes at a cost – accurate physics simulations tend to be one of the most computationally demanding tasks in 3D graphics, even for modern hardware.

This article will focus on cloth simulations, more specifically, on how well, the most popular open-source 3D graphics software on the market handles them in terms of performance and how does it fare in comparison to other, commercial solutions. To achieve that, a series of benchmarks will be ran and the results of that tests will be presented throughout this article.

2. Research goals and methods

This article's goal is to examine the functionality of Blender open-source program in cloth simulation area as well as the simulation performance. The goal of the tests performed is to prove the following hypothesis: the more cores the CPU used for simulating cloth in Blender has, the less time those simulations take on average.

For that purpose, three benchmark scenes where created:

- A sheet of fabric hanging in the air from its corners with additional wind force object
- A sheet of fabric falling onto a sphere
- An animated character wearing a simple dress

Every scene has two versions differing in polygon count: less complex scene with simulated object having around 4000 vertexes and more complex one with simulated object having around 15000 vertexes. Each variant will be tested with two sets of simulation quality settings – 5 quality steps, 2 collision quality and 10 quality steps and 5 collision quality. The tests will be automated using a script in Python that runs the animation 100 times, clears memory cache during each run and gets the elapsed time at the end.

The results will be presented in a table, with percent usage of the computer resources during testing and 1-10 grade representing the occurrence of graphical artifacts such as object clipping.

Blender's functionality in cloth simulation will be examined in relation to other, paid programs – Autodesk Maya, Autodesk 3DSMax and Cinema4D. The analysis of functionality will be presented as a table where the rows will be representing cloth simulation features and column will be representing particular programs.

3. Automating tests

Blender does not provide users with ways to control the number of animation loops nor does it show the run time of the animation. The only information the user is

given is current framerate, which is not sufficient information.

In order to address those issues, there was a need for an algorithm that will automatically run the animation and measure the elapsed time.

Blender gives its users the option to run scripts in Python under a dedicated tab. Those scripts can control most aspects of a Blender project. Because of that, the algorithm written for the purpose of this paper was written in Python inside the tab that was mentioned earlier.

Listing 1: Python script for benchmark automation

```
import datetime
import bpy
from bpy.app.handlers import persistent

@persistent
def run_after_frame_change(dummy):
    if bpy.context.scene.frame_current > frame_end:
        bpy.app.handlers.frame_change_post.remove(run_after_frame_change)
        bpy.ops.screen.animation_cancel()
        loop()

def loop():
    global n
    global a, b
    if n==0:
        a = datetime.datetime.now()
    n += 1
    if n == number_of_time_you_want_loop:
        b = datetime.datetime.now()
        print(b - a)
        return

    bpy.context.object.modifiers["Cloth"].point_cache.use_library_path = True
    bpy.context.object.modifiers["Cloth"].point_cache.use_library_path = False
    bpy.ops.ptcache.free_bake_all()
    bpy.context.scene.frame_current = frame_start
    bpy.app.handlers.frame_change_post.append(run_after_frame_change)
    bpy.ops.screen.animation_play()

n = 0
a = 0
b = 0
number_of_time_you_want_loop = 100
frame_start, frame_end = 1, 60
bpy.ops.screen.animation_play()

bpy.app.handlers.frame_change_post.append(run_after_frame_change)
```

The algorithm loops the number of times specified by number_of_time_you_want_loop variable. When the loop ends, it stops the animation. At the start of the execution of the script, variable a is initialized with current date and time and where the script reaches the end of the loop, current date and time is assigned to variable b. To get the elapsed time, variable a is subtracted from b at the after looping the number of times specified earlier. Aside from that, during each loop, the data cached in memory is cleared and using library path is disabled. It has to be re-enabled and disabled again each time because of the possible overlook on developers' part.

4. Performance benchmarks

The tests were performed on two different systems. The first benchmarking platform had the following specification: Intel Core i5-11400F with 6 cores and 12 threads, Nvidia GeForce RTX 3070 and 32 GB of DDR4 RAM. The second platform had the following specification: Intel Xeon E5-2630v3 (8 cores, 16 threads), 64 GB of DDR3 RAM and an integrated Intel GPU. The tests were performed in Blender version 3.1, which was the latest stable release at the time.

All scenes were tested with 2 sets of settings: quality steps set to 5 with collision quality set to 2 and also quality steps set to 10 with collision quality set to 5. The first two scene had collision detection distance set to 0.015 m, whist the third one had it set to 0.001 m. The rest of the settings were left on the default values provided by the software.

4.1. Execution times

Figure 1 presents the average times of simulation for 5 quality steps and 2 collision quality settings on the first benchmarking platform. The graph shows that increasing the polygon count results in significant increase in execution times. The simulation took from 3 to 9 times longer when going from lower to higher mesh density with the same quality settings across all scenes. However, despite not having any collision object, the scene with a sheet of fabric takes more time to calculate than the scene with a sphere when comparing higher polygon count versions of those benchmarks.

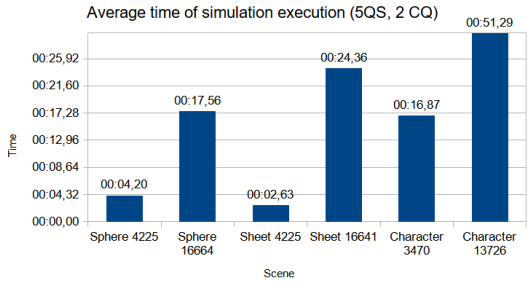


Figure 1: Average time of execution for 5 quality steps and 2 collision quality on the first machine.

Figure 2 showcases the result obtained with the second benchmarking platform with the lower simulation quality settings. The simulation times are slightly higher than on the first machine with the exception for the more complex version of the sheet of fabric benchmark, which is slightly faster on the second machine, although the results follow the similar pattern as with the first platform.

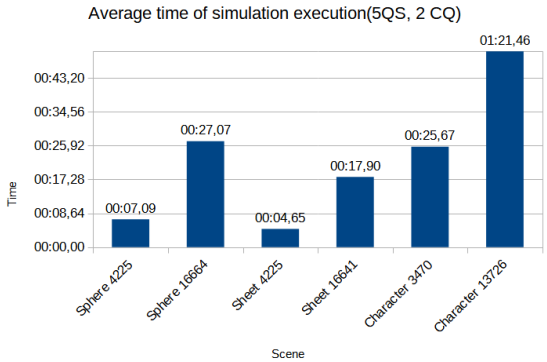


Figure 2: Average time of execution for 5 quality steps and 2 collision quality on the second machine.

Figure 3 shows the average times for quality steps set to 10 and collision quality set to 5 on the first machine. As expected, increasing those values significantly affects execution times – in case of this research there was a 1,5 to 5 times increase compared to the tests shown in figure 1.

For example, the animation for the character scene with 13736 polygon cloth object is 3,04 times slower than 3470 version with 5QS/2CQ settings, but 6,22 times slower for 10QS/5CQ. The distribution of values is comparable to graph seen on figure 1, but, in case of the most demanding of the scenes, the one with a character, the simulation time is far longer and the difference between less complex version of it and more complex one is also far greater.

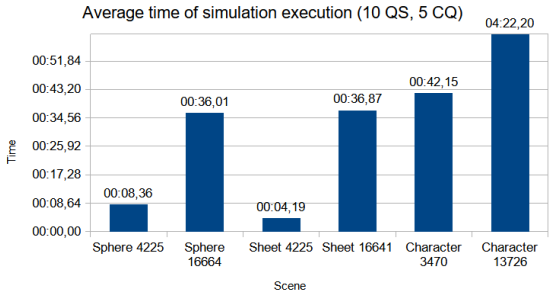


Figure 3: Average times of execution for 10 quality steps and 5 collision quality on the first machine.

Figure 4 shows the execution times for the scenes with higher simulation quality settings obtained on the second platform. Similarly to the corresponding graph for the first test bench, the results are mostly slightly higher than those acquired on the first computer, although the last scene, the character benchmark with higher polygon count ran faster by almost exactly 1 minute on the second platform.

This may be because of utilizing 2 more cores and, as a result, 4 more threads of the CPU that the second platform had available to it, albeit at lower clock speed.

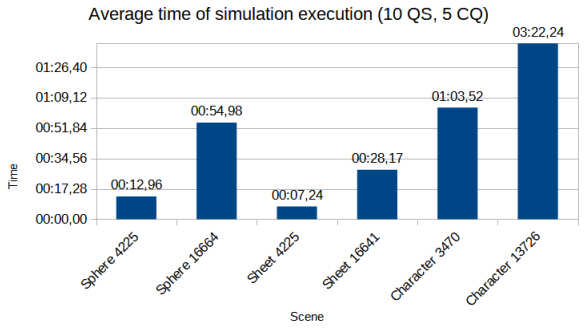


Figure 4: Average times of execution for 10 quality steps and 5 collision quality on the second machine.

In all of the tests performed, the character benchmark took the most time to complete due to having the most complex collision object, that was also animated. This causes the simulation engine to take the movement of that object into consideration and increases the time of the calculations.

4.2. Resource usage and artifacts

Table 1 presents the use of computer resources during the testing and perceived level of graphical arti-facts such as inter-object clipping.

As expected, more complexity equals more strain on the computer – e.g., sheet scene that has no interactions between objects uses less processing power even in the heaviest version than other scenes that have collision objects thrown into the mix. However, it can be noticed that in some cases the more complex scene may use less CPU power than the less complex one. This is likely due to the engine utilizing more cores and spreading the load more evenly for more demanding scenes. For example – the sheet scene with 4225 polygons with 10 QS/5CQ used 1 thread at 100% with others being idle, whilst the sheet scene with the same QS and CQ, but 16641 polygons used one thread at 80% and another at 78%. Sheet and sphere scenes used only one or two threads entirely or close to entirely, leaving the rest at idle and the character scene utilized all CPU threads above 50% with at least one exceeding 80%, however 13726 version of the character benchmark actually used 8 threads at close to 100%.

The simulation impact on RAM usage was negligible, since the simulation caching was disabled for the purpose of this research. Any variation in this category was most likely due to other programs running in the background and other components of Blender.

It can be observed that the polygon count of the simulated object impacts the cache size. The quality settings, however, do not impact the cache size, since there was no difference in the cached memory between higher settings and lower settings versions of scenes that had the same number of vertexes.

Measuring the level of graphical artifacts observed is a challenging task, because of it being subjective and depending on one's own perception which varies from person to person, however it is important when considering the outcome of the simulation. For that reason, the group of 20 anonymous respondents aged 17-33, mostly male, have graded all the scenes via a form. The respondents assigned the score from 1 to 10 to each scene – 1 meaning there are none or barely any visible artifacts and 10 meaning that there are a lot of noticeable artifacts. The value in Table 1 is the average grade given to a scene by the respondents.

Table 1: Computer resources used and graphical artifacts observed by the respondents

Scene	Vertex count	Quality Steps	Collision Quality	CPU	RAM	Cache (MiB)	Artifacts
Sphere	4225	5	2	33,00%	12,00%	8,9	4,6
Sphere	4225	10	5	41,00%	12,00%	8,9	4
Sphere	16664	5	2	30,00%	13,00%	34,9	5
Sphere	16664	10	5	50,00%	12,00%	34,9	4,85
Sheet	4225	5	2	27,00%	14,00%	8,9	3,65
Sheet	4225	10	5	25,00%	14,00%	8,9	3,45
Sheet	16641	5	2	23,00%	15,00%	34,9	3,45
Sheet	16641	10	5	21,00%	15,00%	34,9	3,25
Character	3470	5	2	60,00%	17,00%	7,3	5,75
Character	3470	10	5	75,00%	17,00%	7,3	5,75
Character	13726	5	2	55,00%	17,00%	28,9	5,55
Character	13726	10	5	67,00%	14,00%	28,9	4,95

5. Feature analysis

Blender as an open-source software may be falsely perceived by professionals that use commercial solutions as a basic tool that lack functionality and flexibility those programs have, when in reality more and more features are coming to Blender in each new release and it is now one of the most complete 3D tool available.

In this chapter, the features of Blender will be compared to other, paid programs, namely Autodesk 3DS Max, Autodesk Maya and Cinema4D. The results will be shown in a table.

Cloth simulation options in Blender are accessible under the „Cloth" section of the „Physics" tab, while in case of 3DSMax it is available from within the modifiers section as a cloth modifier [1]. It should be noted that you can also find cloth simulation in the modifiers stack in blender as well [2]. Additionally, 3DSMax only allow interactions between objects that are under one instance of cloth modifier. In case of Maya, all of cloth simulation features belong to nCloth plugin [3] and in Cinema4D they can be found under simulation tags section [4].

One of the most important factors that determines both quality and performance of the simulation is choosing a numerical integration method. There are four main categories of integration methods: explicit, implicit, high-order and low-order methods. The authors of the paper [5] concluded that implicit methods like Backward Euler method result in faster calculation, while explicit ones like Runge-Kutta produce the most accurate results. However, none of the software tested offer an option to choose the integration method.

Another important thing to consider is the number of steps of the simulation. As pointed out in paper [6], traditionally small time steps had to be used to avoid numerical instability. The authors of this publication proposed an algorithm that can use larger steps while mitigating instability, but generally the more time steps, the more accurate the result will end up being, at the cost of a slower calculation. All of the tested software enable the user to set the number of steps (in case of Maya the options are called „iterations" and „subsamples", but ultimately it is the same as choosing time steps).

As the paper [7] concludes, running the simulation on a GPU can be up to 60 times faster than on the CPU, but despite the benefits, none of the tested programs allow choosing to simulate on a graphics card outside of rendering and use a CPU computing instead.

All of the selected programs offer an option to control internal characteristics of a cloth object like stiffness, bending, damping, compression and pressure, while Blender and both of Autodesk products also have pre-defined presets for common material types.

Additionally, Blender and Maya offer an option to choose a bending model – angular and linear in the case of Blender.

All of the software tested allow to enable or disable both inter-object and internal collisions and all except 3DS Max also have an option to control the parameters of those collisions, like the distance at which the software detects collisions, friction between colliding geometry etc.

All of the programs enable the user to constrain certain parts of the mesh to either stay in their place or stick to another part of the mesh or a different mesh, while only Blender has an option that enables the simulated object to dynamically react to deformations of the meshes it collides with, here called dynamic mesh.

One way of creating clothing for virtual characters is a method called sewing – which like the real-world sewing involves designing the clothing item in parts that are later joined together. This technique in 3D graphics comes from a well-known tool for creating garment for 3D characters, Marvelous Designer and it is present in all of selected programs except for Maya.

Tearing cloth, although possible in Blender with a use of pin groups is not explicitly available from the simulation options as opposed to 3DSMax, where you can also select individual seams that will be teared and Cinema4D.

All of the programs have a functionality to cache the simulation results into memory and onto a storage disk (which, for distinction will be referred to as baking). It should be pointed out that Maya, for example, require the user to manually save cache each time there is anything changed, while Blender recalculates automatically after every change to the object or its properties.

Table 2 presents the comparison of the cloth-simulation functionality between all of the chosen programs in detail.

Table 2: Cloth simulation functionalities and adjustable properties in Blender, 3DSMax, Maya and Cinema4D

Feature	Blender	3DSMax	Maya	Cinema4D
Time steps	Y	Y	Y	Y
Subsampling	N	Y	Y	Y
Speed control	Y	N	Y	N
Mass	Y	Y	Y	Y
Bending Model	Y	N	Y	N
Integration alogorythm	N	N	N	N
Bending parameters	Y	Y	Y	Y
Damping parameters	Y	Y	Y	Y
Internal springs	Y	Y	Y	Y
Friction	Y	Y	Y	Y
Pressure	Y	Y	Y	Y
Compression	Y	Y	Y	N
Stiffness	Y	Y	Y	Y
Tearing	N	Y	N	Y
Tear strength	N	Y	N	Y
Keep shape	N	Y	Y	N
Seam parameters	N	Y	N	Y
Order of interactions	N	Y	N	N
Cloth type presets	Y	Y	Y	N
Simulation cache'ing	Y	Y	Y	Y
Simulation baking	Y	Y	Y	Y
Cloth constraints	Y	Y	Y	Y
Sewing	Y	Y	N	Y
Advanced pinching	N	Y	N	N
Weld group	N	Y	N	N
Dynamic mesh	Y	N	N	N
Inter-object collisions	Y	Y	Y	Y
Self-collisions	Y	Y	Y	Y
Collision quality	Y	N	Y	Y
Colission distance	Y	N	Y	Y
Collision friction	Y	N	Y	N
Collision clamping	Y	N	N	Y
Edge collisions	N	N	N	Y
GPU Compute	N	N	N	N
Weights	Y	N	Y	Y

This comparison shows that the cloth simulation functionalities provided by Blender out of the box are on par with those offered by commercial soft-ware and considering the open-source nature of this program it will likely offer even more features in years to come.

6. Conclusion

In this article, the performance of cloth simulations in different scenarios was measured. The research confirmed that more vertexes and higher quality settings significantly impact the execution times – the tests shown up to 6 times increase in execution time when going from 5 quality steps and 2 collision quality to 10 quality steps with the same polygon count and 5 collision quality and up to 9 times increase when going from 4000 polygons to 16000 polygons with the same quality settings. At the same time increasing those mesh density and quality setting, doesn't remove all visual artifacts during the animation.

The initial hypothesis could not be proven as the results turned out ambiguous. Most benchmarks were faster on the system with less CPU cores that had higher clock-speed and only the most demanding of the benchmarks, the character scene with 14000 vertices cloth object was faster on the system with more CPU cores with less clock speed.

Although a fast modern CPU was used during the testing, it is still impossible to run complex cloth simulation in real time on a personal computer with a use of a CPU. The next step in this research could be testing how simulating using the GPU compares to our results, but as of now, Blender only allows changing the graphics processor for rendering.

The second part of research shown that Blender's functionality in the field of cloth simulations is comparable to commercial solutions. Most of the programs tested provided similar functionalities with mostly minor differences.

References

[1] Autodesk, 3DS Max Documentation, https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-80CAE782-8EFD-4E9E-ABA2-C53127D6659A-htm.html, [30.08.2022]

[2] Blender Foundation, Blender Documentation, https://docs.blender.org/manual/en/latest/physics/cloth/introduction.html, [30.08.2022]

[3] Autodesk, Maya Documentation, https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/Maya-CharEffEnvBuild/files/GUID-ED791F1C-8412-4785-829F-9925F2604E8A-htm.html, [30.08.2022]

[4] Maxon, Cinema4D Documentation, https://help.maxon.net/c4d/en-us/#html/5953.html?Highlight=cloth, [30.08.2022]

[5] P. Volino, N. Magnenat-Thalmann, Comparing efficiency of integration methods for cloth simulation, Proceedings of Computer Graphics International (2001) 265-272.

[6] D. Baraff, A. Witkin, Large steps in cloth simulation, SIGGRAPH '98 (1998) 43-54.

[7] T. Min, T. Ruofeng, R. Narain, M. Chang, D. Manocha, A GPU-based Streaming Algorithm for High-Resolution Cloth Simulation, Computer Graphics Forum 32(7) (2013) 21-30.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 88–92
Received: 13 October 2022
Accepted: 7 November 2022

Comparative analysis of selected programming frameworks dedicated to SPA applications

Analiza porównawcza wybranych szkieletów dedykowanych budowie aplikacji SPA

Kinga Dybowska*, Małgorzata Plechawska-Wójcik

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

Abstract
The purpose of this study was to compare two popular programming frameworks dedicated to building SPA applications. Two proprietary applications with identical functionalities were analyzed, one created in React and the other in Angular. A ready-made programming tool was used, an original script comparing the speed of adding elements to the database, and the popularity of the technology was checked on the Stack Overflow website and in the Google search engine.
Keywords: Angular; React; SPA

Streszczenie
Celem niniejszej pracy było porównanie dwóch popularnych szkieletów programistycznych dedykowanych budowie aplikacji typu SPA. Analizie zostały poddane dwie autorskie aplikacje o identycznych funkcjonalnościach, jedna z nich powstała w React, natomiast druga w Angularze. Wykorzystano gotowe narzędzie developerskie, autorski skrypt porównujący szybkość dodawania elementów do bazy danych oraz sprawdzono popularność technologii w serwisie Stack Overflow oraz w wyszukiwarce Google.
Słowa kluczowe: Angular, React, SPA

*Corresponding author
Email address: kinga.mikolajczuk@pollub.edu.pl (K. Dybowska)

1. Wstęp

Celem niniejszego artykułu jest analiza różnic i podobieństw wybranych szkieletów programistycznych dedykowanych aplikacjom typu SPA (Singe Page Application). SPA w odróżnieniu od innych systemów korzystają tylko z jednego pliku HTML, nie przeładowują więc stron podczas ich obsługi. Treści są generowane dynamicznie [1]. Do przeprowadzenia badania stworzone zostały dwie aplikacje, jedna z nich w technologii Angular [2], natomiast druga w React [3]. Powstałe systemy posiadają takie same funkcjonalności oraz podobny wygląd. Analiza została przeprowadzona na podstawie specjalnie przygotowanych scenariuszy badawczych. Do porównania stron wykorzystano gotowe narzędzia deweloperskie oraz specjalnie napisany skrypt.

Na podstawie celu pracy określono kilka pytań badawczych:

- Który szkielet programistyczny jest szybszy?
- Która aplikacja jest bardziej wydajna?
- Który szkielet programistyczny jest bardziej popularny?

Otrzymane wyniki badań pozwolą odpowiedzieć na pytanie który szkielet programistyczny jest lepszy.

1.1. React

React jest biblioteką stworzoną do budowania interfejsów użytkownika. Powstał na podstawie JavaScript, zaprojektowany przez programistów Meta, dawniej Facebook. Obecnie jest jednym z popularniejszych szkieletów programistycznych wykorzystywanych przy aplikacjach typu SPA. Używa wirtualnego drzewa DOM (Document Object Model) oraz izolowanych komponentów zarządzających własnym stanem. Są one opisane za pomocą JSX (JavaScirpt XML), jest to połączenie JavaScript oraz HTML (HyperText Markup Language). Dzięki temu możliwe jest wykorzystanie znaczników HTML bezpośrednio w kodzie, zaraz obok logiki aplikacji.

Każdy komponent jest wielokrotnego użytku i posiada cykl życia, czyli montowanie, aktualizację oraz usuwanie. Wymienione etapy posiadają swoje odzwierciedlenie w metodach umożliwiających reakcję na zmiany związane z cyklem życia komponentu [4].

1.2. Angular

Biblioteka ta również jest polecana przy tworzeniu stron typu SPA. Jednak w odróżnieniu od React została napisana w języku TypeScirpt, dzięki czemu łatwiej jest utrzymać czysty i zrozumiały kod. Cechuje się tym, że od razu po uruchomieniu jest gotowa do użycia. Szkielet został opracowany i aktualnie jest wspierany przez Google.

Biblioteka składa się z podstawowych elementów, jakimi są komponenty, zorganizowanych w NgModules. Moduły te zbierają powiązany kod w zestawy funkcjonalne. Aplikacja zawsze posiada co najmniej jeden

moduł, główny, umożliwia on wstępne ładowanie i zazwyczaj posiada kolejne moduły funkcji. Komponenty natomiast definiują widoki, czyli zestawy elementów ekranu. Angular może je wybierać i modyfikować zgodnie z logiką aplikacji. Posiada również mechanizm wiązania danych. Jest to automatyczny sposób aktualizowania widoku przy każdej zmianie modelu. Działa on w dwie strony, odświeża więc też model przy każdej zmianie widoku. Mechanizm ten ułatwia manipulację wirtualnym drzewem DOM [5].

2. Przegląd literatury

Analiza dostępnych artykułów o podobnej tematyce pozwoliła wybrać odpowiednie metody badawcze do porównania szkieletów programistycznych.

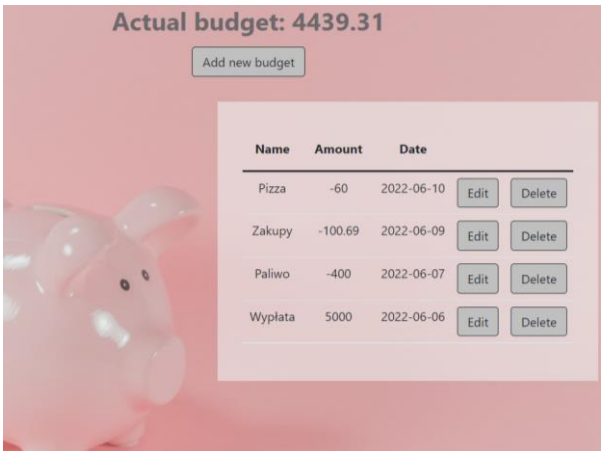
W niektórych pracach do porównania bibliotek wykorzystano średni czas wysyłania żądań do serwera oraz szybkość renderowania komponentów [6-8]. Bardzo wygodnym sposobem przeprowadzenia eksperymentu jest wykorzystanie wirtualnych narzędzi deweloperskich dostępnych online. Dzięki nim można porównać zgodność ze standardami, czas ładowania aplikacji oraz wygodę korzystania na urządzeniach o różnych rozdzielczościach ekranu [9].

Autorzy artykułów o podobnej tematyce często do porównania szkieletów programistycznych wykorzystywali autorskie skrypty [10]. Sprawdzały one między innymi szybkość podstawowych operacji na komponentach, czy też czas ładowania całej aplikacji. Niektóre prace posiadały bardzo dużo scenariuszy badawczych [11]. Oprócz wymienionych wcześniej aspektów w wymienionej pracy sprawdzono również obsługę popularnych zabezpieczeń, obsługę różnych formatów danych, integrację z portalami społecznościowymi takimi jak Facebook, obsługę cache. A także dostęp do baz danych, między innymi Firebase czy SQLite, szybkość współpracy z innymi bibliotekami, takimi jak Google Charts oraz popularność użycia. Efekty eksperymentów zostały przedstawione na wykresach.

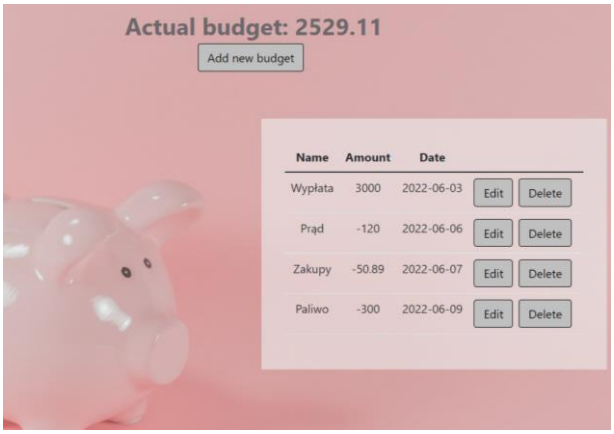
Innym podejściem wykorzystywanym do porównania szkieletów programistycznych było zbadanie szybkości działania aplikacji przy coraz większej liczbie użytkowników jednocześnie korzystających z systemu [12]. Zbadano operacje odczytu danych oraz ich modyfikację. Do tego celu wykorzystano specjalne narzędzie JMeter.

3. Metody badawcze

Badanie zostało przeprowadzone na przykładzie prostej aplikacji internetowej umożliwiającej zarządzanie oraz kontrolowanie budżetu domowego. W tym celu powstały dwa systemy, jeden przy użyciu biblioteki React, natomiast drugi przy wykorzystaniu szkieletu programistycznego Angular. Posiadają one identyczne funkcjonalności oraz podobny wygląd. Na Rysunku 1 została zaprezentowana strona stworzona w Angularze, natomiast na Rysunku 2 – w React.



Rysunek 1: Aplikacja Angular.

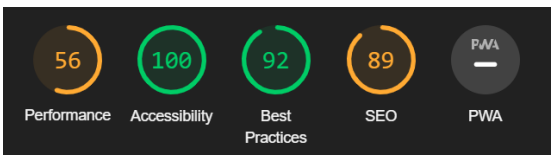


Rysunek 2: Aplikacja React.

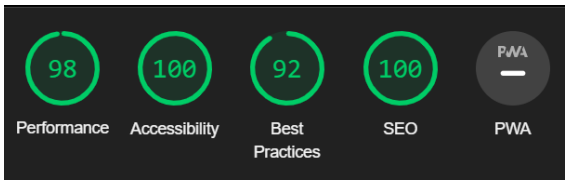
Aby porównać obie strony opracowano specjalne scenariusze badawcze, testują one różne właściwości aplikacji.

Scenariusz 1: Analiza wydajnościowa z wykorzystaniem narzędzi deweloperskich

Do przeprowadzenia tego badania wykorzystano dedykowane wirtualne narzędzie deweloperskie, dostępne w przeglądarce Google Chrome, jakim jest *Lighthouse* [13]. Pozwala ono przeprowadzić szczegółową analizę strony. Ocenia jej wydajność, dostosowanie do wyszukiwarek oraz potrzeb użytkowników. Pokazuje ono również błędy wykryte podczas pracy aplikacji, a także czas ładowania elementów na stronie. Wyniki analizy przedstawiane są w postaci wykresów. Na Rysunku 3 przedstawiono wyniki analizy dla aplikacji Angular, a na Rysunku 4 dla aplikacji React.



Rysunek 3: Wyniki analizy aplikacji Angular.



Rysunek 4: Wyniki analizy aplikacji React.

Na powyższych rysunkach można zauważyć, że aplikacja React w tym badaniu osiągnęła dużo lepsze wyniki. Wszystkie mierzone statystyki oznaczone zostały kolorem zielonym. Natomiast Angular otrzymał żółte wyniki, oznaczające wartość do poprawy w dwóch przypadkach. Analiza metryk ukazała duże różnice w szybkości działania stron. Angular dużo wolniej ładował aplikację jak i jej pojedyncze elementy, wpłynęło to na wyniki pierwszej metryki. Drugą widoczną różnicą jest SEO, czyli optymalizacja pod kątem wyszukiwarek internetowych. Aplikacja React byłaby wyświetlana wcześniej niż Angular.

Scenariusz 2: Analiza wydajnościowa z wykorzystaniem skryptu

Scenariusz ten sprawdza szybkość zapisu określonej liczby rekordów do bazy danych. Do przeprowadzenia tego badania stworzono autorski skrypt, mierzący czas dodawania elementów od momentu naciśnięcia przycisku uruchamiającego formularz, aż do umieszczenia rekordu w bazie. Skrypt został napisany w języku Python, z wykorzystaniem pakietu Selenium. Rozwiązanie to jest bardzo często wykorzystywane do analizy zadań wykonywanych w przeglądarce internetowej.

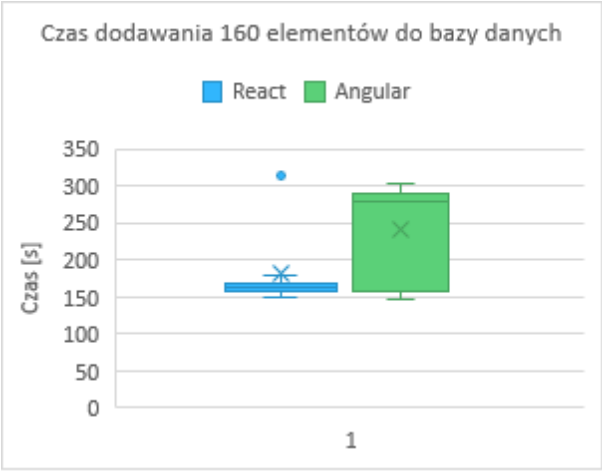
Pierwsze badanie zostało wykonane dla dodawania kolejno coraz większej liczby rekordów do bazy. Zaczynając od 10, aż do 250, zwiększając liczbę elementów o 10. Podczas ostatniego etapu dodawane jest więc 250 pozycji, a w bazie danych znajduje się już 3000 rekordów. Test ten pozwoli więc zaobserwować działanie aplikacji przy rosnącym obciążeniu bazy danych. Badanie to pokazało różnice w dodawaniu określonej liczby danych. Na początku aplikacje umieszczały rekordy w podobnym czasie. Podczas próby w której testowane było dodawanie 140 elementów Angular zaczął wykonywać zadanie coraz wolniej, natomiast React od tego momentu, aż do dodawania 220 rekordów prezentował dużo mniejszy czas. Jednak w przypadku 230, 240 oraz 250 elementów widać duży spadek jego wydajności, w tych punktach Angular wykonał zadanie znacznie szybciej.

Kolejne badanie jest bardzo podobne do poprzedniego, w tym przypadku jednak do bazy danych dodawane było 20 rekordów, umieszczanie każdej serii danych zostało powtórzone 30 razy. W tym przypadku, po każdej próbie wszystkie elementy były usuwane. Aplikacja za każdym razem działała więc na pustej bazie danych. Dla otrzymanych w ten sposób wyników przeprowadzono test Manna-Whitneya, ponieważ po analizie Shapiro-Wilka okazało się, że dane nie pochodzą z rozkładu normalnego i nie mają rozkładu zbliżonego do normalnego. Niemożliwe było więc wykonanie badania T Studenta, w związku z tym zdecydowano przeprowadzić test nieparametryczny. Badanie nie wykazało różnic w średnim czasie dodawania elementów do bazy danych. Na rysunku 5 zaprezentowano wyniki testu wykonanego w środowisku *RStudio*. Wartość p była większa niż, 0.05, co oznacza, że nie było podstaw do odrzucenia hipotezy H_0, że nie średni czas dodawania rekordów do bazy danych jest porównywalny dla obydwu technologii.

```
        Wilcoxon rank sum test with continuity correction

data:  Czas by Technologia
W = 394, p-value = 0.4119
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -1.711097  0.748720
sample estimates:
difference in location
            -0.4801567
```

Rysunek 5: Wyniki testu Manna-Whitneya dla 20 elementów.

Ostatnim badaniem wykonanym przy użyciu skryptu było przeprowadzenie identycznego testy jak w poprzednim przypadku, tym razem jednak w bazie danych umieszczane było 160 rekordów. Tutaj również wykorzystano test Manna-Whitneya. Analiza wykazała, że nie ma podstaw do odrzucenia hipotezy H_0, że średni czas dodawania elementów do bazy danych jest porównywalny dla obydwu aplikacji. Na rysunku 6 zaprezentowano wykres pudełkowy danych wykorzystanych w tym badaniu. Rozmiar pudełka określony jest na podstawie rozstępu ćwiartkowego danych, czyli różnicy między pierwszym, a trzecim kwartylem. Z definicji wynika, że w tym przedziale znajduje się 50% wszystkich obserwacji. Większy rozmiar pudełka świadczy więc o bardziej zróżnicowanych danych. Wąsy, reprezentowane przez linie połączone z pudełkiem oznaczają najmniejszą i największą wartość lub półtorej wartość rozstępu ćwiartkowego. Wartości znajdujące się poza tym zakresem określane są jako punkty odstające. Na wykresie można zauważyć też poziomą linię wewnątrz pudełka, która oznacza medianę. Widoczny jest również znak „x”, który reprezentuje średnią wszystkich danych.



Rysunek 6: Czas dodawania 160 elementów do bazy danych.

Na powyższym rysunku widać duże różnice w średnich czasach dodawania 160 rekordów do bazy danych poszczególnych aplikacji. Zauważalna jest duża różnica w wysokości pudełka reprezentującego wyniki. W przypadku React jest ono kilkukrotnie mniejsze od Angulara. Na tej podstawie oraz analizie średniej i mediany, które również są znacznie mniejsze dla technologii React można stwierdzić, że dodawał on elementy do bazy danych znacznie szybciej niż Angular. Na wykresie widoczne są jednak punkty odstające, które znacznie zawyżają średnią dla aplikacji React. Przez te wartości znalazła się ona poza pudełkiem.

Scenariusz 3: Popularność użytkowania

W tym przypadku przenalizowano popularność obydwu technologii, za pomocą jednego z najpopularniejszych serwisów wykorzystywanych przez programistów, jakim jest Stack Overflow [14] oraz narzędzie Google Trends [15]. Ze strony Stack Overflow wykorzystano coroczne badanie pokazujące popularność technologii frontendowych wykorzystywanych przez użytkowników serwisu. Dane pochodzą z 2022 roku. Respondenci zostali podzieleni na grupy, dzięki czemu możliwe jest zobaczenie odpowiedzi dla wszystkich respondentów, profesjonalnych programistów, a także osób uczących się programowania. Na Rysunku 7 przedstawiony został wycinek wykresu, obrazujący siedem najpopularniejszych technologii wśród wszystkich respondentów.

Rysunek 7: Najpopularniejsze technologie według użytkowników Stack Overflow [https://survey.stackoverflow.co/2022/#technology-most-popular-technologies].

Badanie wykazało, że React był popularniejszy w przypadku wszystkich grup respondentów. Na każdym z wykresów zajmował drugie miesjcez wynikiem powyżej 42%. Natomiast Angular wśród wszystkich respondentów oraz w przypadku profesjonalnych programistów zajmował piątą pozycję z wynikiem ponad 20%. Jednak w przypadku osób uczących się programowania znalazł się dopiero na dziewiątej pozycji. Używało go zaledwie 10% badanych. Wyprzedziły go biblioteki takie jak Django, Flask, Vue.js oraz Next.js. Przeanalizowano również znaczniki, którymi oznaczane są pytania na Stack Overflow. Okazało się, że od 2019 roku hasła związane w *react.js* są znacznie bardziej popularne, niż *angular*.

Drugie narzędzie, jakim jest Google Trends pozwala na porównanie popularności wyszukiwanych haseł przez użytkowników z całego świata. Analizowane dane są frazami wpisywanymi w wyszukiwarkę Google w ciągu ostatnich 12 miesięcy na całym świecie. Badanie to wykazało, że hasła związane z React są dużo bardziej popularne. Angular jest popularniejszy jedynie w trzech regionach z 76. We wszystkich pozostałych większym zainteresowaniem cieszy się *React.js*. Największą przewagę ma w Korei Południowej, Chinach, Nigerii, Indonezji oraz Bangladeszu. Proporcje wyszukiwania w tych krajach wynoszą w zaokrągleniu 9:1, na korzyść *React.js*.

4. Wnioski

Celem badań, była analiza różnic i podobieństw wybranych szkieletów programistycznych dedykowanych aplikacjom typu SPA. Porównano więc technologie React i Angular na podstawie specjalnie przygotowanych serwisów. Przeprowadzone badania pozwoliły odpowiedzieć na postawione na początku pracy pytania badawcze. Na pytanie: „Która aplikacja jest bardziej wydajna?" pozwoliło odpowiedzieć pierwsze badanie. Wykazało ono, że aplikacja React szybciej ładuje poszczególne elementy oraz ma krótszy czas ładowania strony. Można więc stwierdzić, że jest ona bardziej wydajna. Drugie badanie pozwoliło odpowiedzieć na kolejne pytanie: „Który szkielet programistyczny jest szybszy?". Analiza wykazała, że nie ma istotnych statystycznie różnic między szybkością dodawania elementów do bazy danych w badanych technologiach. Ostanie pytanie brzmiało: „Który szkielet programistyczny jest bardziej popularny?". Ostatnie badanie pokazało, że React cieszy się dużo większą popularnością. W większości regionów na świecie jest dużo częściej wyszukiwany. Programiści chętniej z niego korzystają, jest to zauważalne zwłaszcza wśród osób uczących się programowania. Bardzo rzadko decydują się one na technologię Angular.

Przeprowadzona analiza pozwoliła również odpowiedzieć na główną hipotezę, który szkielet programistyczny jest lepszy. Podsumowując powyższe badania można stwierdzić, że lepszą technologią jest React. Jest bardziej wydajny, szybciej ładuje elementy na stronie oraz cieszy się większą popularnością. Wśród osób uczących się programowania jest zdecydowanie częściej wybieraną technologią niż Angular. W przypadku średniego czasu dodawania elementów do bazy danych nie wykryto statystycznie istotnych różnic, jednak wykres pudełkowy dla umieszczania 160 rekordów pokazuje, że średni czas wykonywania tego zadania dla React jest mniej zróżnicowany i zdecydowanie niższy niż w technologii Angular. Wszystkie powyższe badania pozwalają więc stwierdzić, że React jest lepszym szkieletem programistycznym, niż Angular.

Literatura

[1] M. Mikowski, J. Powell, Single Page Web Applications. Programowanie aplikacji internetowych z JavaScript, Helion, 2015.

[2] A. Freeman, Angular. Profesjonalne techniki programowania. Wydanie II, Helion, 2018.

[3] P. Kamiński, React. Wstęp do programowania, Helion, 2021.

[4] E. Porcello, A. Banks, React od podstaw. Nowoczesne wzorce tworzenia aplikacji. Wydanie II, Helion, 2021.

[5] A. Freeman, Angular. Profesjonalne techniki programowania. Wydanie IV, Helion, 2021.

[6] R. Baida, M. Andriienko, Analiza porównawcza wybranych technologii internetowych w kontekście tworzenia gier na przykładzie gry planszowej Munchkin, Praca magisterska, Politechnika Lubelska, Lublin 2019.

[7] K. Boczkowski, Porównanie narzędzi do tworzenia aplikacji typu SPA na przykładzie React i Vue.js, Praca magisterska, Politechnika Lubelska, Lublin 2019.

[8] M. Jusięga, Analiza porównawcza wybranych wersji szkieletu programistycznego Symfony, Praca magisterska, Politechnika Lubelska, Lublin 2020.

[9] M. Wrońska, Analiza porównawcza biblioteki jQuery Mobile i frameworka Bootstrap w wytwarzaniu stron responsywnych, Praca magisterska, Politechnika Lubelska, Lublin 2016.

[10] G. Białecki, B. Pańczyk, Analiza wydajnościowa aplikacji Svelte i Angular, JCSI 19 (2021) 139-143.

[11] J. Palak, Analiza porównawcza zastosowania szkieletów Angular2 i Ember.js, Praca magisterska, Politechnika Lubelska, Lublin 2017.

[12] N. Kovalchuk, Porównanie wybranych szkieletów aplikacji internetowych dla technologii Java, Praca magisterska, Politechnika Lubelska, Lublin 2016.

[13] Opis rozszerzenia Lighthouse, https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=pl , [27.09.2022].

[14] Najpopularniejsze technologie według serwisu Stack Overflow, https://survey.stackoverflow.co/2022/#technology-most-popular-technologies , [27.09.2022].

[15] Google Trends popularność haseł Angular i React, https://trends.google.com/trends/explore?q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9 , [27.09.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 93–99
Received: 13 October 2022
Accepted: 26 October 2022

Comparative analysis of C and Python on the basis of the execution time of applications implementing selected algorithms

Analiza porównawcza języków C oraz Python na podstawie czasu wykonania aplikacji realizujących wybrane algorytmy

Paweł Rysak*

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

Abstract
The article deals with a comparative analysis of the speed of code execution written in the C language and Python. Its primary purpose was not to seek a simple answer to the question of which language would be more efficient, but what is the scale of differences in the performance of these languages. In order to determine the performance of compiled and scripting languages, a comparison of the languages was made using the following algorithms: the algorithm for solving the Hanoi tower problem, the Huffman encoding algorithm and the algorithm for converting numbers into text. Each of the listed algorithms was implemented in both languages. Then the execution time of the programs was measured and the results were obtained to determine the extent of the differences in their execution speed. C language applications executed 6 to 188 times faster than Python language applications.

Keywords: performance; algorithms; C; Python

Streszczenie
Artykuł dotyczy analizy porównawczej szybkości wykonywania kodu przez język C oraz Python. Jej podstawowym celem nie było szukanie prostej odpowiedzi na pytanie, który z języków będzie wydajniejszy, tylko jaka jest skala różnic w wydajności tych języków. W celu określenia wydajności języka kompilowanego oraz skryptowego dokonano zestawienia języków na przykładzie następujących algorytmów: algorytm rozwiązujący problem wieży Hanoi, algorytm kodowania Huffmana oraz algorytm zamiany liczb na tekst. Każdy z wymienionych algorytmów został zaimplementowany w obydwu językach. Następnie dokonano pomiaru czasu realizacji programów, którego wyniki pozwoliły na określenie skali różnic w szybkości ich wykonania. W języku C aplikacje wykonywały się od 6 do 188 razy szybciej niż aplikacje w języku Python.

Słowa kluczowe: wydajność; algorytmy; język C; Python

*Corresponding author
Email address: pavelrysak@gmail.com (P. Rysak)

1. Wstęp

Ze względu na różnorodność języków programowania, wybór odpowiedniego języka do osiągnięcia zamierzonych celów jest bardzo ważny, szczególnie w dzisiejszych czasach, gdy rozmiar i poziom skomplikowania aplikacji oraz systemów ciągle rosną. Przy wytypowaniu odpowiedniego języka do stworzenia aplikacji priorytety są zróżnicowane. Może to być łatwość rozszerzalności i utrzymania kodu lub właśnie wysoka wydajność i niskie zużycie zasobów. W przypadku chęci uzyskania kodu, który będzie łatwy w utrzymaniu i rozszerzaniu, powinno się wziąć pod uwagę języki z czytelnym i klarownym kodem np. Python. Jeżeli ważna jest wydajność i niskie zużycie zasobów, pod uwagę powinno wziąć się języki kompilowane np. C.

Główną różnicą pomiędzy językami C oraz Python jest to, że C jest językiem kompilowanym, a Python językiem interpretowanym. Program napisany w języku kompilowanym przed uruchomieniem musi zostać przetłumaczony na język zrozumiały dla komputera, natomiast język skryptowy charakteryzuje się tym, iż w momencie uruchomienia programu kod jest interpretowany linia po linii. Takie podejście utrudnia znalezienie błędów w kodzie programu, gdyż będą one widoczne dopiero w trakcie uruchomienia programu, w przeciwieństwie do języka kompilowanego, gdzie informacje o błędach są wyświetlane jeszcze przed uruchomieniem programu.

W niniejszym artykule dokonano porównania reprezentantów tych dwóch grup języków. Celem badań było określenie skali różnic wydajności języka C oraz Python. W pracy zestawiono ze sobą czas wykonania algorytmów zaimplementowanych w ww. językach: algorytm rozwiązujący problem wieży Hanoi, algorytm kodowania Huffmana oraz algorytm zamiany liczb na tekst.

2. Przegląd literatury

Pierwszym przeanalizowanym artykułem jest praca autorstwa Farzeen Zehra, Maha Javed, Darakhshan Khan, Maria Pasha pt. *Comparative Analysis of C++ and Python in Terms of Memory and Time* [1]. Podjęto się tutaj porównania technik zarządzania pamięcią w języku C++ oraz Python. Dokonano analizy czasu oraz wykorzystania pamięci w czterech algorytmach:

algorytm szukający, algorytm sortujący, algorytm dodający nowe elementy do struktury danych oraz algorytm usuwający elementy. Algorytmy zaimplementowano w ww. językach. Autorzy doszli do wniosku, że kod zaimplementowany w języku Python potrzebuje więcej czasu na wykonanie, gdyż konwertuje on kod wiersz po wierszu do kodu maszynowego. W języku C++ kod programu w całości ulega kompilacji jeszcze przed uruchomieniem programu.

Kolejny artykuł napisany przez Lutz Prechelt pt. *An empirical comparison of seven programming languages* dotyczy porównania czasu wykonania kodu, zużycia pamięci oraz długości kodu algorytmu zaimplementowanego w siedmiu językach, w tym także C oraz Python [2]. Program ma za zadanie ładować do pamięci słownik oraz plik z numerami telefonów, a następnie konwertować numery telefonów na tekst. Badania pokazały, że Python w większości przypadków lepiej radzi sobie z obsługą złożonych zbiorów danych.

Następnym przeanalizowanym artykułem jest praca napisana przez Hailong Zhang oraz Jun Nie pt. *Program Performance Test based on Different Computing Environment* [3]. W tej pracy przeprowadzono pomiar czasu wykonania algorytmu obliczającego odległość sferyczną pomiędzy dwoma punktami. Przeanalizowano różne warianty kodu, m.in. Python, C oraz Cython czyli zastosowanie typów danych oraz funkcji z języka C w Python. Wyniki jednoznacznie wskazały, że programy zaimplementowane w języku Cython mogą być o wiele bardziej wydajne w porównaniu do języka Python.

W kolejnym artykule pt. *A comparison of implementations of basic evolutionary algorithm operations in different languages* [4] napisanym przez Juan-Julián Merelo-Guervós i inni, zdecydowano się na zmierzenie prędkości wykonywania operacji algorytmów ewolucyjnych, zaimplementowanych w różnych językach, m.in. w C oraz w Python. Wybrano typowe dla algorytmów ewolucyjnych operacje: mutacja Bitflip, krzyżowanie oraz operacja OneMax. Polegają one odpowiednio na zmianie wartości konkretnego bitu w łańcuchu, zmianie wartości bitów pomiędzy łańcuchami oraz na zapętleniu łańcuchów i zliczaniu wystąpień wartości 1. Autorzy doszli do wniosku, że w większości przypadków algorytmy szybciej wykonywały się w języku C. Wyjątkiem była tu operacja krzyżowania.

Ostatnią pracą w przeglądzie literatury jest artykuł pt. *Investigating the performance of MicroPython and C on ESP32 and STM32 microcontrollers* [5] napisany przez Valeriu Manuel Ionescu oraz Florentina Magda Enescu. Dotyczy on wydajności języka C oraz MicroPython w odniesieniu do dość popularnych mikrokontrolerów: STM32 oraz ESP32. Wykorzystano tutaj algorytm szyfrowania SHA-256 oraz algorytm wyznaczania sum kontrolnych CRC-32. Wyniki otrzymane przez autorów jednoznacznie stwierdzają, że wydajność MicroPython jest zdecydowanie mniejsza niż języka C. Na korzyść języka MicroPython przemawia łatwość implementacji algorytmów na mikrokontrolerach, bez konieczności wprowadzania specjalnych zmian w kodzie.

3. Opis środowiska

Testy przeprowadzono na komputerze z zainstalowanym systemem operacyjnym Windows 11, z procesorem Intel Core i7-8750h o bazowej częstotliwości 2,2 GHz zaś maksymalnej 4,1 GHz oraz pamięcią RAM o pojemności 16 GB.

Python został poddany badaniu w wersji 3.10.4, a C w wersji C17. W obydwu językach zastosowano optymalizację. Jako kompilator języka C użyto MSVC w wersji 14.32.

4. Metody badawcze

Podjęto się zaimplementowania algorytmu rozwiązującego problem wieży Hanoi, algorytmu kodowania Huffmana oraz algorytmu zamiany liczb na tekst w językach C oraz Python. Wartościami mierzalnymi był czas wykonania algorytmów. W przypadku języka C stworzony uprzednio kod programu został skompilowany w trybie „release" do pliku exe. Następnie w czasie testów był uruchamiany bez skorzystania ze środowiska programistycznego przy użyciu wyłącznie konsoli wiersza poleceń systemu Windows. Kod napisany w Pythonie również był uruchamiany w ten sam sposób. Takie podejście zapewniło zminimalizowanie błędów pomiarowych spowodowanych przez środowisko programistyczne poprzez chwilowe „zawieszenie się". Każdy z algorytmów uruchomiono 30 razy.

Pomiaru czasu wykonania w języku C dokonano przy pomocy biblioteki time, deklarując zmienne typu całkowitego clock_t w odpowiednich miejscach programu i przypisując do nich wartość zwracaną przez funkcję clock [6]. Następnie odjęto od siebie te wartości i podzielono przez ilość cykli wykonywanych przez procesor w ciągu sekundy (ang. „ticks") [7]. Użycie metody w kodzie przedstawione jest na Listingu 1. Dokładność tej metody wynosi 0,001 s. Wynika to z wartości przechowywanej w wyrażeniu makro CLOCKS_PER_SEC, która w obecnej konfiguracji środowiska wyniosła 1000.

Listing 1: Przykład wykorzystania metod pomiaru czasu w języku C

```c
#include <stdio.h>
#include <time.h>

int main() {
    double time_spent = 0.0;
    clock_t begin = clock();

    for (int i = 0; i < 1000; i++)
    {
        printf("%d \n", i);
    }

    clock_t end = clock();
    time_spent += (double)(end - begin)
                  / CLOCKS_PER_SEC;
    printf("Time is %f  seconds ",
           time_spent);
    return 0;
}
```

W języku Python, pomiaru czasu wykonania programu dokonano przy użyciu biblioteki time, w podobny sposób co w języku C, odejmując od siebie wartości zwrócone przez funkcję time [8]. Dokładność tej metody jest różna dla każdego komputera i zależy od ilości cykli wykonywanych przez procesor w ciągu sekundy. Na komputerze, na którym przeprowadzano badania otrzymano dokładność 10^{-17} s. Użycie metody przedstawiono na Listingu 2.

Listing 2: Przykład wykorzystania metod pomiaru czasu w języku Python

```python
import time

if __name__ == '__main__':
    start_time = time.time()

    for x in range(1000):
        print(x)

    print("--- %s ---" % (time.time()
                          - start_time))
```

5. Algorytmy

5.1. Algorytm rozwiązujący problem wieży Hanoi

Rozwiązanie problemu wieży Hanoi polega na przeniesieniu wieży zbudowanej z krążków o różnych średnicach ze słupka A na słupek C posługując się słupkiem B [9]. Przy wykonywaniu tego zadania należy przenosić tylko jeden krążek jednocześnie oraz trzeba pamiętać, aby zawsze mniejszy krążek leżał na większym.

W przypadku implementacji tego algorytmu zastosowano 25 krążków. Została użyta funkcja rekurencyjna, w której liczba krążków (n) ulega stopniowemu zmniejszaniu.

Listing 3: Kod algorytmu rozwiązującego problem wieży Hanoi w języku Python

```python
import time

def Hanoi(n, _from, _to, _aux):
    if n == 1:
        move(n, _from, _to)
        return
    Hanoi(n-1, _from, _aux, _to)
    move(n, _from, _to)
    Hanoi(n-1, _aux, _to, _from)


def move(n, _from, _to):
    global step
    step = step + 1


if __name__ == '__main__':
    start_time = time.time()
    step = 0
    n = 25

    Hanoi(n, 'A', 'C', 'B')
    print("--- %s ---" % (time.time()
                          - start_time))
```

Kod algorytmu rozwiązującego problem wieży Hanoi [10] w języku Python został zaprezentowany na Listingu 3. Dodatkowo umieszczono w tym programie metodę pomiaru czasu wykonania programu, czego dokonano przed wywołaniem właściwej funkcji o nazwie Hanoi oraz zaraz za nią.

Kod algorytmu [10] w języku C został przedstawiony na Listingu 4, gdzie także umieszczono użycie metody mierzącej czas wykonania programu. Dokonano tego bezpośrednio przed oraz za wywołaniem właściwej funkcji o nazwie Hanoi. Taki schemat pomiaru czasu został zastosowany do pozostałych algorytmów.

Listing 4: Kod algorytmu rozwiązującego problem wieży Hanoi w języku C

```c
#include <stdio.h>
#include <time.h>

void hanoi(int n, char _from,
           char _to, char _aux);
void move(int n, char _from, char _to);

void hanoi(int n, char _from,
           char _to, char _aux) {
    if (n == 1) {
        move(n, _from, _to);
        return;
    }
    hanoi(n - 1, _from, _aux, _to);
    move(n, _from, _to);
    hanoi(n - 1, _aux, _to, _from);
}

int step = 0;
void move(int n, char _from, char _to)
           { ++step; }

int main() {
    double time_spent = 0.0;
    int n = 25;
    clock_t begin = clock();
    hanoi(n, 'A', 'C', 'B');
    clock_t end = clock();
    time_spent += (double)(end - begin)
                  / CLOCKS_PER_SEC;
    printf("Time is %f  seconds ",
           time_spent);
    return 0;
}
```

5.2. Algorytm kodowania Huffmana

W informatyce algorytmy kompresujące mają za zadanie zmniejszać rozmiar danych. Spośród metod kompresji danych możemy wyróżnić dwa rodzaje: kompresja stratna oraz kompresja bezstratna. W algorytmie kompresji stratnej niektóre fragmenty danych są tracone. W algorytmach kompresji bezstratnej dane sprzed kompresji są identyczne jak po kompresji. Do tej właśnie grupy należy algorytm kodowania Huffmana.

Algorytm [11] jako dane wejściowe przyjmuje tekst, w którym zliczane są wystąpienia poszczególnych liter. Następnie ze zbioru liter zawierających przyporządko-

wane im liczby wystąpień, dwie litery o najmniejszej liczbie wystąpień są łączone w nowy liść w drzewie bianrnym, którego wartość będzie odpowiadać sumie wystąpień wcześniej wspomnianych liter. W kolejnym kroku brana jest litera o najmniejszej wartości wystąpień, z pominięciem tych, które zostały już wykorzystane, a następnie znowu tworzony jest nowy liść mający wartość odpowiadającą sumie wartości wystąpień dobranej litery i utworzonego wcześniej liścia. Kroki te są powtarzane aż do momentu, kiedy pozostanie jeden element. W ten sposób powstaje drzewo binarne, w którym skrajne elementy (liście) mają przypisaną wartość zero, jeśli znajdują się po lewej stronie lub wartość 1, jeśli znajdują się po prawej stronie. Kody poszczególnych liter są otrzymywane w wyniku przypisywania odpowiadających im wartości binarnych idąc od góry drzewa.

Zaimplementowany algorytm ma za zadanie przyjąć tekst o dowolnej długości, a następnie wyznaczyć kody poszczególnych liter. Program uruchomiono dla tekstu wejściowego, anglojęzycznego zawierającego 1074000 słów, czyli 6158000 znaków.

Na Listingu 5 dotyczącym fragmentu kodu w Python [12,13] przedstawiono użycie biblioteki heapq, dostarczającej metody pozwalające łatwiej zaimplementować algorytm. Dostarcza ona funkcje umożliwiające utworzenie stosu (heapify) i wstawienie do niego posortowanych elementów (heappush). Kod ze względu na te usprawnienia stał się znacznie krótszy od kodu w języku C.

Listing 5: Fragment kodu algorytmu kodowania Huffmana w języku Python

```
def buildHuffmanTree(text):
    if len(text) == 0:
        return

    freq = {}
    for c in text:
        freq[c] = freq.get(c, 0) + 1

    pq = [Node(k, v) for k, v in
          freq.items()]
    heapq.heapify(pq)

    while len(pq) != 1:

        left = heappop(pq)
        right = heappop(pq)

        total = left.freq + right.freq
        heappush(pq, Node(None, total,
                          left, right))

    root = pq[0]

    huffmanCode = {}
    encode(root, '', huffmanCode)
```

Listing 6 przedstawia funkcję sortującą elementy przed wstawieniem ich do drzewa (minHeapify) oraz funkcję zamieniającą węzły (swapNode) zaimplementowaną w języku C [12,13]. W tym przypadku wszystkie funkcje napisano bez użycia gotowych bibliotek. W związku z tym kod algorytmu stał się bardziej zrozumiały, lecz w konsekwencji uzyskano dość sporą liczbę linii kodu, co może przełożyć się na ogólną wydajność algorytmu.

Listing 6: Fragment kodu algorytmu kodowania Huffmana w języku C

```
void minHeapify(struct MinHeap* minHeap,
                int idx) {
    int smallest = idx;
    int left = 2 * idx + 1;
    int right = 2 * idx + 2;

    if (left < minHeap->size &&
        minHeap->array[left]->freq
        < minHeap->array[smallest]->freq)
        smallest = left;

    if (right < minHeap->size &&
        minHeap->array[right]->freq
        < minHeap->array[smallest]->freq)
        smallest = right;

    if (smallest != idx) {
        swapNode(&minHeap->array[smallest]
            , &minHeap->array[idx]);
        minHeapify(minHeap, smallest);
    }
}

void swapNode(struct Node** a,
              struct Node** b) {
    struct Node* t = *a;
    *a = *b;
    *b = t;
}
```

5.3. Algorytm zamiany liczb na tekst

Istota algorytmu polega na wczytaniu pliku tekstowego, w którym zapisane są losowo wygenerowane, 20-cyfrowe numery oraz wzorzec dekodowania. Fragment pliku wejściowego został przedstawiony na listingu 7. Program korzystając ze wzorca wyświetla w wyniku tekst. Uruchomiono go dla 10000 wierszy.

Listing 7: Fragment pliku wejściowego

```
1 78048452524596100000:13=Z;17=Y;23=
  X;21=W;35=V;40=U;42=T;48=S;56=R;57
  =Q;65=P;70=O;75=N;79=M;84=L;85=K;9
  6=J;0=X;1=A;2=B;3=C;4=D;5=E;6=F;7=
  G;8=H;9=I
2 53601617947958700000:13=Z;17=Y;23=
  X;21=W;35=V;40=U;42=T;48=S;56=R;57
  =Q;65=P;70=O;75=N;79=M;84=L;85=K;9
  6=J;0=X;1=A;2=B;3=C;4=D;5=E;6=F;7=
  G;8=H;9=I
3 79669760798126900000:13=Z;17=Y;23=
  X;21=W;35=V;40=U;42=T;48=S;56=R;57
  =Q;65=P;70=O;75=N;79=M;84=L;85=K;9
  6=J;0=X;1=A;2=B;3=C;4=D;5=E;6=F;7=
  G;8=H;9=I
4 37117332951862000000:13=Z;17=Y;23=
  X;21=W;35=V;40=U;42=T;48=S;56=R;57
  =Q;65=P;70=O;75=N;79=M;84=L;85=K;9
  6=J;0=X;1=A;2=B;3=C;4=D;5=E;6=F;7=
  G;8=H;9=I
```

Kod algorytmu zaimplementowanego w języku C przestawiono na listingu 8. Kod okazał się znacznie dłuższy niż w języku Python. Najpierw została zliczona liczba wystąpień podciągu w słowie wejściowym, następnie przy użyciu obliczonej wartości została alokowana określona ilość pamięci na wynik. Takie podejście zapewnia zarezerwowanie optymalnej ilości pamięci w celu jak najefektywniejszego działania programu.

Listing 8: Fragment kodu algorytmu zamiany liczb na tekst w języku C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>

char* replaceWord(const char* source,
                  const char* oldWord,
                   const char* newWord);
char* rtrim(char* s);

char* replaceWord(const char* source,
                  const char* oldWord,
                   const char* newWord) {
    char* result;
    int i, cnt = 0;
    int newWordlen = strlen(newWord);
    int oldWordlen = strlen(oldWord);

    for (i = 0; source[i] != '\0'; ++i) {
        if (strstr(&source[i], oldWord)
            == &source[i]) {
            ++cnt;
            i += oldWordlen - 1;
        }
    }

    result = (char*)malloc(i + cnt *
             (newWordlen - oldWordlen)
              + 1);
```

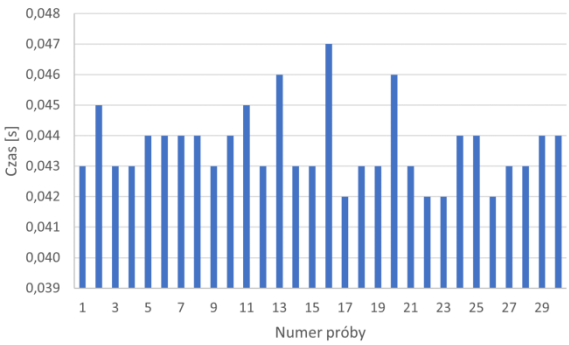
Kod algorytmu zaimplementowanego w języku Python przestawiono na Listingu 9. W porównaniu z kodem programu w C kod okazał się bardzo krótki. Jest to spowodowane tym, że Python posiada wiele gotowych funkcji, z których można skorzystać. W tym przypadku skorzystano z funkcji replace, która umożliwiła zamianę liczby na tekst. Funkcja jest wywoływana na łańcuchu znaków. Przyjmuje ona w parametrach ciąg znaków, który ma być zamieniony oraz nowy ciąg, który zastąpi poprzedni. Dodatkowym opcjonalnym parametrem jest liczba określająca, jak dużo wystąpień podciągu w słowie ma być zamienione. W opisywanym algorytmie parametr tej funkcji został pominięty, co oznacza, że replace będzie zamieniać wszystkie napotkane podciągi. W algorytmie użyto również funkcji split, która, umożliwia podzielenie napisu. Funkcja przyjmując w parametrach określony znak oraz liczbę oznaczającą ilość podzieleń tekstu, dzieli tekst w miejscach, w których napotka na wcześniej zdefiniowany znak, pod warunkiem, że nie przekroczy ona określonej liczby podziałów tekstu. W omawianym algorytmie jako parametr funkcji podano jedynie pierwszy z nich.

Listing 9: Kod algorytmu zamiany liczb na tekst w języku Python

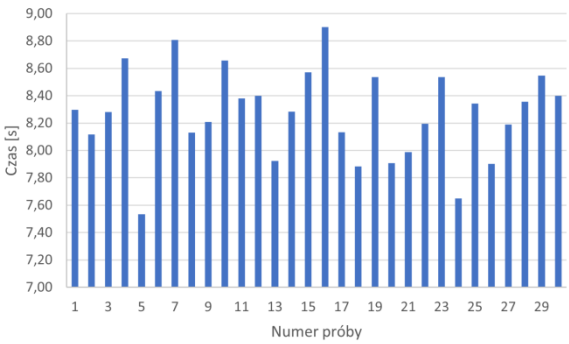
```python
import time

if __name__ == '__main__':
    start_time = time.time()

    file = open("input.txt", "r")

    for line in file.readlines():

        inputLine = line.strip()

        split = inputLine.split(":")
        answer = split[0]
        for rule in split[1].split(';'):

            ruleSplit = rule.split('=')

            answer = answer.replace
            (ruleSplit[0], ruleSplit[1])
```

6. Wyniki badań

W tym rozdziale zaprezentowano uzyskane wyniki badań w postaci wykresów. Wartości przedstawione na wykresach dotyczą każdej z 30 podjętych prób uruchomienia każdego algorytmu.



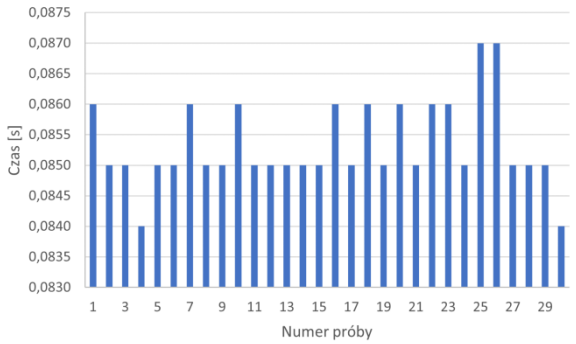
Rysunek 1: Czas wykonania algorytmu rozwiązującego problem wieży Hanoi w języku C.



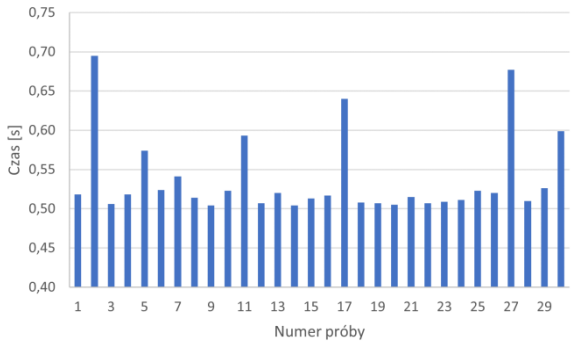
Rysunek 2: Czas wykonania algorytmu rozwiązującego problem wieży Hanoi w języku Python.

Jak widać na Rysunku 1 znaczna część uzyskanych wartości czasu oscyluje w granicach 0,043 s. Górna granica uzyskanych czasów wynosi 0,047 s, zaś dolna

osiąga 0,042 s. Na podstawie Rysunku 2, przedstawiającego czasy uzyskane przez algorytm wieża Hanoi w języku Python nie można jednoznacznie stwierdzić w okolicy jakiej wartości oscyluje znaczna część czasów. Wynika to ze zbyt dużej rozbieżności pomiędzy uzyskanymi wynikami. Odchylenie standardowe wyników języka C oraz Python wynosi odpowiednio 0,001 s i 0,317 s. W przypadku języka C średni czas wykonania algorytmu wyniósł 0,044 s, jeśli chodzi o Python uzyskano średnią wynoszącą 8,271 s. Wyniki te pokazują, że język C okazał się w tym przypadku szybszy blisko 188 razy.



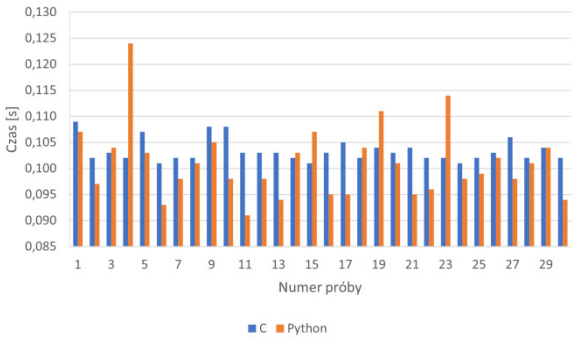
Rysunek 3: Czas wykonania algorytmu kodowania Huffmana w języku C.



Rysunek 4: Czas wykonania algorytmu kodowania Huffmana w języku Python.

Analizując wykres przedstawiony na Rysunku 3 można stwierdzić, że większość wartości oscyluje w granicach 0,085 s. Najdłuższy czas wynosi 0,087 s, a najkrótszy 0,084 s. W przypadku tego samego algorytmu zaimplementowanego w języku Python (patrz Rysunek 4), widać, że czas wykonania większości prób waha się od 0,50 s do 0,55 s. Odchylenie standardowe w przypadku języka C wynosi 0,0007 s, a w przypadku języka Python sięga 0,0500 s. Różnica między tymi wartościami nie jest już tak duża, jak w przypadku algorytmu rozwiązującego problem wieży Hanoi. Średnia wartość czasu wykonania algorytmu w języku C wyniosła 0,085 s, zaś w przypadku języka Python 0,537 s. Otrzymane średnie pozwalają stwierdzić, że język C okazał się szybszy około 6 razy.

Wyniki przedstawione na powyższych rysunkach jednoznacznie dowodzą uzyskania większej wydajności przez język C, pomimo różnej złożoności obliczeniowej algorytmu rozwiązującego problem wieży Hanoi oraz algorytmu kodowania Huffmana. W pierwszym z ww. algorytmów różnice w czasie wykonania są zdecydowanie większe niż w przypadku drugiego algorytmu. Wynika to z tego, że algorytm rozwiązujący problem wieży Hanoi jest zbudowany z wykorzystaniem funkcji rekurencyjnej, która jest często mniej optymalna niż funkcja wykorzystująca pętle.



Rysunek 5: Czas wykonania algorytmu zamiany liczb na tekst w języku C oraz Python.

W przypadku algorytmu zamiana liczb na tekst, wyniki zdecydowanie się różnią od wcześniej opisywanych. Na powyższym Rysunku 5 widać wyraźnie zbliżoną wydajność tych języków. Wyniki dotyczące czasu uzyskanego przez Python są nieco bardziej zróżnicowane. Różnica między najkrótszym a najdłuższym czasem wynosi aż 0,033 s. Jest ona zdecydowanie mniejsza niż w przypadku języka C, gdzie wynosi ona 0,008 s. Odchylenie standardowe wyników jest znowu większe dla algorytmu w języku Python, które wynosi 0,007 s, podczas gdy w języku C osiąga ono 0,002 s. Równocześnie różnica między odchyleniami standardowymi jest najmniejsza i wynosi zaledwie 0,005 s. Wyniki są zbliżone do tego stopnia, że na podstawie wykresu nie można jednoznacznie wskazać wydajniejszego języka programowania. Stwierdzić to można na podstawie porównania średnich wartości czasów, które wynoszą dla języka C 0,103 s oraz dla języka Python 0,101 s.

7. Wnioski

Biorąc pod uwagę specyfikę algorytmów zaimplementowanych w obydwu językach oraz uzyskane wyniki, można stwierdzić, że C jest językiem szybszym od 6 do 188 razy od języka Python. Jak widać skala różnic pomiędzy czasem wykonania programów z zaimplementowanym algorytmem rozwiązującym problem wieży Hanoi oraz kodowania Huffmana jest bardzo duża. Kiedy złożoność obliczeniowa algorytmu wzrasta, tak jak w przypadku algorytmu rozwiązującego problem wieży Hanoi, różnice w czasie są jeszcze bardziej widoczne. Jedyny przypadek, w którym Python uzyskał nieco lepszy czas to algorytm zamiany liczb na tekst. Może to wynikać z faktu, że Python lepiej sobie radzi z przetwarzaniem złożonych zbiorów danych. Na korzyść języka Python przemawia mniejsza liczba linii kodu co wyraźnie widać na zamieszczonych listingach W celu uzyskania lepszych czasów wykonania programów dla tego języka, można zastosować język zawierający elementy składni Python i C, czyli Cython. Jest on

o wiele bardziej wydajny od tradycyjnego języka Python.

Problem, którego rozwiązania podjęto się w tej pracy jest bardzo szeroki i podejmowany przez wielu badaczy. Niniejszy artykuł jest tylko małym dodatkiem, który być może chociaż w małym stopniu przyczyni się pośrednio do udoskonalania programów użytkowych.

Literatura

[1] F. Zehra, M. Javed, D. Khan, M. Pasha, Comparative Analysis of C++ and Python in Terms of Memory and Time, Preprints (2020), https://doi.org/10.20944/preprints202012.0516.v1.

[2] L. Prechelt, An empirical comparison of seven programming languages, Computer 33 (10) (2000) 23-29, https://doi.org/10.1109/2.876288.

[3] H. Zhang, J. Nie, Program Performance Test based on Different Computing Environment, 2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS) (2016) 174-177.

[4] J. -J. Merelo-Guervós et al., A comparison of implementations of basic evolutionary algorithm operations in different languages, 2016 IEEE Congress on Evolutionary Computation (CEC) (2016) 1602-1609.

[5] V. M. Ionescu, F. M. Enescu, Investigating the performance of MicroPython and C on ESP32 and STM32 microcontrollers, 2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME) (2020) 234-237.

[6] Sposób pomiaru czasu wykonania programu w języku C, https://levelup.gitconnected.com/8-ways-to-measure-execution-time-in-c-c-48634458d0f9 , [01.02.2022].

[7] Opis wyrażenia makro CLOCKS_PER_SEC, https://www.educative.io/answers/what-is-clockspersec-in-c , [25.09.2022].

[8] Sposób pomiaru czasu wykonania programu w języku Python, https://pynative.com/python-get-execution-time-of-program/ , [01.02.2022].

[9] Opis problemu dotyczącego algorytmu rozwiązującego problem wieży Hanoi, https://pl.wikipedia.org/wiki/Wie%C5%BCe_Hanoi , [01.02.2022].

[10] Kod algorytmu rozwiązującego problem wieży Hanoi w języku C oraz Python, https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/ , [03.04.2022].

[11] Opis przebiegu algorytmu kodowania Huffmana, https://binarnie.pl/kodowanie-huffmana/ , [01.02.2022].

[12] Kod algorytmu kodowania Huffmana w języku C oraz Python, https://www.programiz.com/dsa/huffman-coding, [03.04.2022].

[13] Kod algorytmu kodowania Huffmana w języku C oraz Python, https://www.techiedelight.com/huffman-coding/ , [03.04.2022].