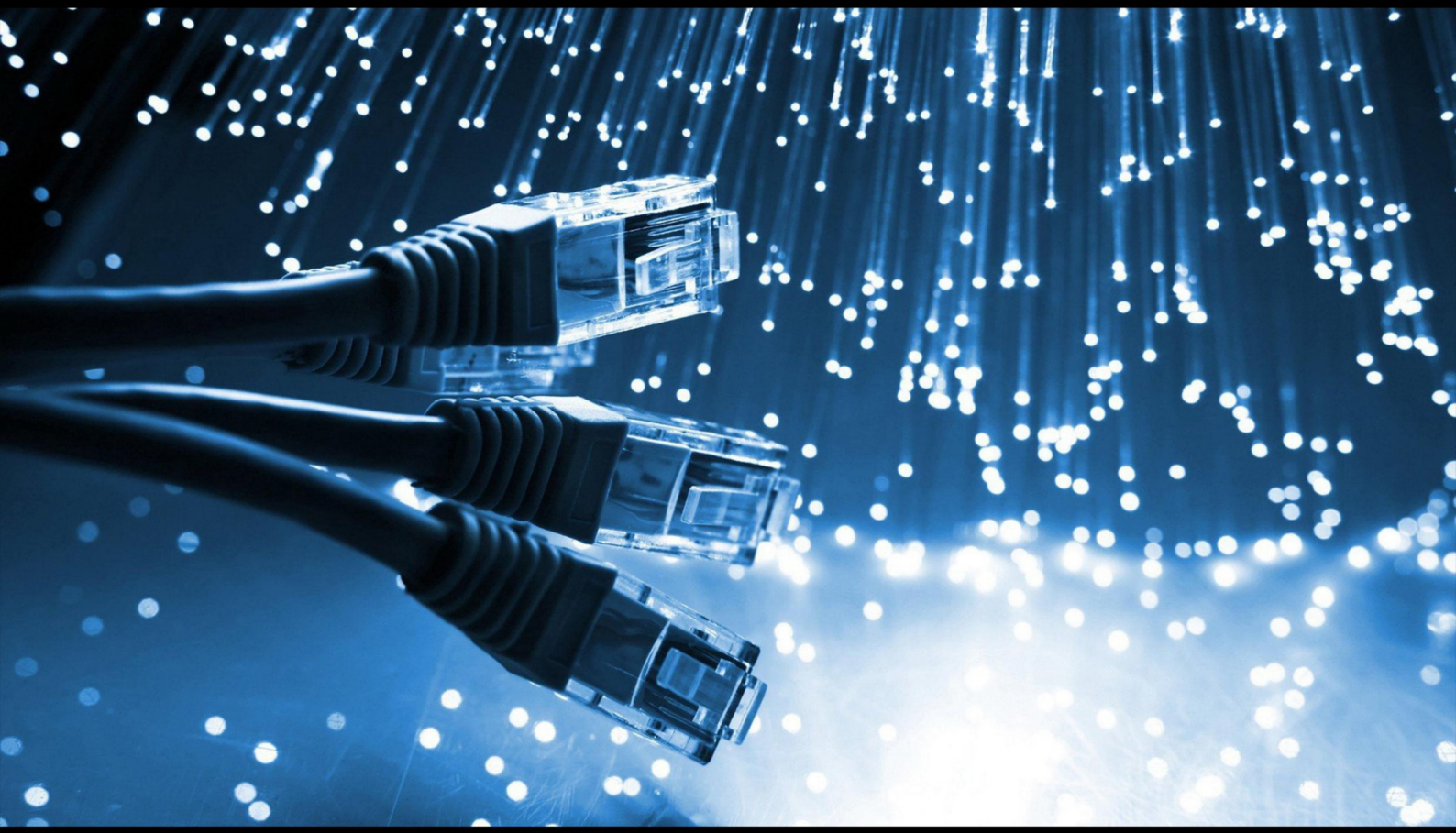
JCSI

Journal of Computer Sciences Institute

Volume 22/2022



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

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

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

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

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

Recenzenci numeru:

dr inż. Maria Skublewska-Paszkowska
dr inż. Małgorzata Plechawska-Wójcik
dr inż. Dariusz Gutek
dr inż Sławomir Przyłucki
dr Paweł Powroźnik
dr inż. Piotr Muryjas
dr inż. Kamil Żyła
dr inż. Jacek Kęsik
dr inż. Tomasz Nowicki

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:

Maria Skublewska-Paszkowska
Małgorzata Plechawska-Wójcik
Dariusz Gutek
Sławomir Przyłucki
Paweł Powroźnik
Piotr Muryjas
Kamil Żyła
Jacek Kęsik
Tomasz Nowicki

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 22 (2022) 1-7
Received: 30 September 2021
Accepted: 21 October 2021

Analysis of the impact of various gamification factors on the level of player satisfaction

Analiza wpływu różnych czynników grywalizacyjnych na poziom satysfakcji graczy

Krystian Demidiuk*, Małgorzata Plechawska-Wójcik

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

Abstract

Nowadays, gamification factors have many applications outside the gaming market, including in science, trade and everyday life, which is referred to as gamification. Despite its widespread use, due to the number of factors that can be used, the scale of potential benefits is still unknown. Although the very concept of gamification refers to areas not related to games, its effectiveness prompts to examine the source from which it comes, and thus the impact of its individual factors on the games themselves. The following work examines the influence of various gamification factors on a player's playing time and satisfaction in order to find the factor that brings the greatest benefit. The research was carried out using a game specially created for this purpose and on the basis of the results of the survey presented to the players. The results show that the five selected gamification factors are the challenges and the badges awarded for them have the greatest impact on both examined metrics.

Keywords: gamification; immersion; player satisfaction

Streszczenie

Czynniki grywalizacyjne mają w obecnych czasach wiele zastosowań poza rynkiem gier, między innymi w nauce, handlu jak i w codziennym życiu, co określane jest mianem grywalizacji. Pomimo jej powszechnego wykorzystywania, z racji na liczbę możliwych do wykorzystania czynników, skala potencjalnych korzyści wciąż pozostaje nieznana. Choć samo pojęcie grywalizacji odnosi się do obszarów nie związanych z grami, jej skuteczność skłania do zbadania źródła z którego się wywodzi, a zatem wpływu poszczególnych jej czynników na same gry. Poniższa praca bada wpływ różnych czynników grywalizacyjnych na czas gry gracza oraz uzyskaną przez niego satysfakcję, w celu znalezienia czynnika przynoszącego największe korzyści. Badanie zostało wykonane przy pomocy specjalnie utworzonej w tym celu gry oraz na podstawie wyników ankiety przedstawionej graczom. Wyniki pokazują, że pięciu wybranych czynników grywalizacyjnych, to wyzwania oraz przyznawane za nie odznaki mają największy wpływ na obie badane metryki.

Słowa kluczowe: grywalizacja; immersja; satysfakcja gracza

*Corresponding author

Email address: demidiuk.krystian@gmail.com (K. Demidiuk)

1. Introduction

Almost 50 years after the creation of the first video game (Pong, 1972), the gaming industry has gained immense popularity. In 2019 alone, it earned USD 148.8 billion and 8,290 new full-fledged titles appeared on the Steam platform. These numbers are definitely impressive and drive the further development of better and better technologies. However, such results also raise many new questions. Extensive research is still needed to understand the reason why players spend so much time at the computer, while other activities can tire them after a few moments, what exactly motivates or fascinates them, and of course how to use it in other areas.

Motivated by all these questions, scientists all over the world have been trying to understand how games affect the human mind. For this purpose, many metrics and terms have been defined, the most popular of which are the concepts of gamification and immersion.

The purpose of the study is to investigate which gamification factor has the greatest impact on game time and overall player satisfaction. Due to their wide application even outside the gaming market, the research results could be used in the development of gamification itself as it is the term used to describe the use of game mechanisms in non-gaming situations. The study is based on surveys conducted on a selected research group and on the basis of information collected during gaming sessions. The obtained results can also be used by the developers to determine the aspects of the game that they should focus on the most. The aim of the study is to answer the following research questions:

1. Does the quality of the graphics extend the player's playing time?
2. Does the quality of the soundtrack extend the player's playing time?
3. Does the presence of a scoreboard extend a player's playing time?
4. Does the presence of player levels extend the player's playing time?
5. Does the presence of achievements and badges increase the player's playing time?

2. Literature review

Some gamification elements are so common that users are not even aware of their presence. Today it is normal that people get points for shopping [1] [2], awards for helping other users [3] or marks for knowledge [4], they have become natural. The use of various game mechanics and mechanisms in various industries other than gaming in order to increase commitment and motivation[5] is referred to as gamification. There has been a lot of research on this phenomenon and its impact on achieving the intended goals [6] [7], such as increasing sales results [3] or improving academic performance [8][9]. The vast majority shows that implementing gamification is definitely worthwhile. Despite its many possibilities, gamification is not a solution to every problem and requires careful planning in order to achieve the desired results. A very important element of the planning process is the correct selection of factors and methods that will have the greatest impact on a given recipient group [10] [11], and it has been proven that the effect may differ depending on age or gender [12]. The way of rewarding the user should also be carefully considered, in order to properly meet his needs [13]. One of the most important concepts of gamification is immersion. It describes the state in which the player focuses on the game to such an extent that feels as if the world created in it is real. This allows the player to distract from the real world, potentially extending their playtime and overall satisfaction. There are many factors that affect the level of game inertia. Sound plays an important role in this situation. Research [14] shows that in order to increase inertia, the music in the game should be liked by the player. Papers [15] [16] clarify this requirement, stating that a good soundtrack should not fit the player's preferences, but it should be of good quality and reflect the atmosphere of the game. Graphics also play a key role in gameplay. The research described in [17] proves that proper adherence to the psychology of colors can significantly increase player satisfaction. The fact whether the level was procedurally generated or created manually by a graphic artist also has a slight difference [18]. The results of the research in [19] show the positive impact of graphic realism, although on the basis of [20], it can be concluded that it is more important how well the whole combines into one style. Moreover, [20] apart from the graphic factor, it also shows the influence of an appropriate reaction to the user's actions in order to increase the sense of influence on the surrounding world in the game. It is also worth keeping the user at the right level of tension, depending on the type of game [21], so that he is properly focused on the game. The game should not be too difficult, which leads to excessive frustration, or too simple, which causes the player to feel bored. In [22], it was investigated how uncertainty influences player satisfaction, unfortunately, despite great expectations, the results of the research did not allow to find a significant influence on inertia.

3. Theoretical Background

The theoretical overview is based on two main concepts, gamification with focus on its factors and immersion. They have a key impact on the obtained results, therefore their knowledge is necessary to conduct an experiment.

3.1. Gamification factors

There are many gamification factors, and the frequency of their use is so great that they are often considered normal, an integral part of life today. The most common of them are probably *the points* awarded for the correct performance of activities in a gamified environment. They can bring benefits in the form of discounts, possibility of exchange for some awards or serve only as information for the customer about his activity or loyalty to a given brand. They are often linked to another factor, namely *the scoreboard*. Users are assigned ranks, places on the list, depending on the number of points obtained. Depending on the application, they can also inform about the awards won or serve to obtain satisfaction themselves. Usually, their main task is to motivate users to perform certain actions. This approach, however, depends largely on how competitive the group of entities is. A similar action is brought by the application of the mechanism of *achievements and badges* granted for their completion. However, this approach differs in that in the case of challenges, competition with other users is limited, and the personality type of the person receiving the challenges is more important. They are often scaled up to several difficulty levels, where the simplest ones are done quite quickly, while with the increase of the level, the required amount of work and time increases dramatically. While it is not required for simple tasks, it is worth using another factor, which is a progress bar, for long tasks. It allows you to increase your motivation by visually representing the progress of an achievement or experience required to move to the next level. *Levels* are a frequently used mechanism for determining when new content is unlocked and how users are divided according to their familiarity with the service. Therefore, it is worth making the new level mean something good for the user, worth the effort. Effort in this case refers to the tasks that are most often performed in order to gain a level. They are usually easier to perform than the aforementioned achievements, and they can also be repetitive. Additionally, in most cases, challenges are optional, while tasks can be imposed without the possibility of skipping them. A special case of tasks are group tasks, which refers to the last of the most important gamification factors, the cooperation of participants. This factor allows to tighten social ties, which leads to building a harmonious community of users. Such a community is usually positive about helping, increasing motivation and commitment.

3.2. Immersion

Immersion, often also called spatial presence, describes the state in which the user is absorbed by the environment presented to him to such an extent that he feels as if he were physically in this environment. As this is a very desirable behaviour, through numerous analyzes and studies, a number of issues that are worth paying special attention to have been developed. The first one of them is the game soundtrack. The key in this situation is to match the music to the atmosphere in the game. For example, racing games are often accompanied by strong, energetic music, while strategy games are usually endowed with calm music that allows the player to concentrate. It is also very common to adjust the music to the current events in the game, it is most visible in role-play games, where in moments of danger, the music changes drastically, thus signalling the current situation. In addition to the background music, there are many different sound elements that allow to enhance immersion. The rustle of leaves in the forest, the splash of water by the lake, or the sounds of nearby animals are just a few examples. It is also extremely important that the sounds delivered to the player at a given moment are sounds that are consistent with what the player expects. Sounds triggered by the player's actions, such as the creaking of a door opening or the sounds of footsteps while traveling, are also often included. In games that allow some form of conversation, or at least narration, the creator should also remember about the right choice of voice. The bored, sleepy tone of the narrator during the climactic scene will certainly be noticed by the player, thus bringing him out of the state of immersion. An inseparable part of designing immersive games is the process of choosing the graphic style. The game does not have to be photorealistic in order to provide the player with the best possible experience, but it does need to provide a consistent level of detail and style. The ultra realistic dragon model in a square game will undoubtedly grab the player's attention, thus destroying the overall gameplay experience. It is also worth paying attention to providing the player with the right perspective. In strategy games, the top view is the most common, while in action or role-play games, the view from the first or third person perspective dominates, with the possibility of changing these settings depending on the player's preferences. An important element is also providing a clear and legible interface, giving the player access to the most important information and a sense of control over current events. Of course, matching the sound and graphics is not enough if the goal is to provide the player with the greatest satisfaction. It is also necessary to take care of the gameplay itself. The game should consist of interesting tasks that allow the player to avoid monotony, preferably with different difficulty levels, so as to keep the player at the expected level of interest. This is possible thanks to a well-developed plot telling a story that can interest the player. Depending on the type of game, the plot is presented in text form or through all kinds of animations. Some game types also allow for ludonarrative behaviour. This allows the player to spend time in the game to perform other activities and tasks not directly related to the main story. This procedure allows the player to better master the game mechanics, understand the world presented to him and, in many situations, develop the character. By analyzing the above examples, one can easily see that there are many additional factors to consider when designing an immersive game. In addition to technological limitations, the design and the type of game itself also play an important role. For example, strategy games are usually characterized by high complexity and connections between various game mechanisms, which requires the player's constant attention and continuous decision-making. In turn, role-play games are distinguished by a high sense of development, with visible effects of work and commitment, thus motivating to continue the game. The factors that bring the player out of immersion are also important to discuss. The most obvious on this list are all kinds of errors and inconsistencies, overlapping models, distinctive textures, and stuttering characters. In addition, which is very easy to see especially in mobile games, there are too many ads that show up, which cannot be avoided.

4. Method

This chapter provides information about the game that is the basis for the research and describes the method of preparing and conducting the research, as well as the mechanisms influencing the quality of the obtained results.

4.1. The Game

The game was created by hand only for the purposes of research, using a ready-made, free template available on the Unity platform. As a genre, a tower defense game with a top view was selected. The main task of the player is to place the title towers in places intended for this purpose, so as to stop the waves of hostile objects moving along the marked route to the object referred to as the player base. A total of six levels have been prepared, the first of which was the base mode with no upgrades. Each of the other 5 modes was based on the base version and offered one of the possible improvements, which were:

- enhanced graphics,
- enhanced soundtrack,
- scoreboards,
- player levels,
- achievements with badges.

The game content was generated automatically with a gradually increasing difficulty level. An exemplary look of the game, presented in the mode with extended graphics, is shown in Figure 1.

Figure 1: Exemplary look of the game in mode with extended graphics.

4.2. Protocol

After agreeing to participate in the research, the participants were acquainted with the research topic and the game with its mechanisms. Then, they were presented with the research protocol and any questions were answered. After confirming the understanding of the research methodology, each participant received a link under which the game was posted. The next step was to choose a unique player name to be used in subsequent games and in the survey. After choosing the name, the player was able to play the game in the base version. The condition for the correct ending of the game was pressing a specific button visible after three minutes of the game or the player's defeat. After successfully completing a level, the player then had to wait 24 hours to continue research. This procedure was aimed at reducing the impact of the player's boredom with the game on the results obtained. For the time of research, the game was released on a free platform, so each participant could conduct next test at any time, taking into account the above-mentioned condition. Each player had the opportunity to play all of the modes, starting from the base mode and continuing with the next randomly selected level. After playing all possible levels, the respondent had to complete a questionnaire consisting of demographic questions and questions describing the satisfaction obtained by the player.

5. Discussion

5.1. Participants

A total of 768 players participated in the study. After verifying the correctness and completeness of the obtained data, 747 correct results were obtained. Due to the nature of the research, the research group consisted of people at least partially related to IT or games. The study involved 691 men and 56 women, mostly between the ages of 15 and 29. They were mainly students of IT technical schools, IT studies and people working as programmers of games or other applications. Each of the respondents regularly plays video games and 133 admitted playing at least 5 days a week. In terms of the duration of the game session, most responses focused on 4-6 hours or more. It is not possible to accurately determine the level of difficulty of the game expected by the players, as most of the respondents indicated that it depends on the game. When describing their type of player, almost half of the respondents described themselves as explorers, which, unfortunately, is not an ideal match for the type of game chosen for the research.

5.2. Play time analysis

The analysis of variance (ANOVA) was used to analyze the influence of gamification factors on the player's playing time. Its correct use requires meeting the assumptions of variance and normality. The first of them was checked using the Brown-Forsythe test (p = 0.00), indicating that the assumption of homogeneity of variance is met. Categorized normality plots were used to verify the normal distribution of the dependent variable. After confirming the assumptions, the ANOVA test was performed. The result of p <= 0.00 thus obtained means that the hypothesis of no difference is rejected and additional post-hoc tests can be performed. For this purpose, the Tukey test was selected, the results of which are presented in the figure 2.

Tukey HSD test; Variable: Var2 (timescolumned.sta)
Marked differences are significant at p < .05000

Var1		{1} M=370.15	{2} M=920.16	{3} M=258.36	{4} M=812.71	{5} M=678.19	{6} M=969.90
1	{1}		0.000020	0.007133	0.000020	0.000020	0.000020
2	{2}	0.000020		0.000020	0.011354	0.000020	0.638102
3	{3}	0.007133	0.000020		0.000020	0.000020	0.000020
4	{4}	0.000020	0.011354	0.000020		0.000457	0.000036
5	{5}	0.000020	0.000020	0.000020	0.000457		0.000020
6	{6}	0.000020	0.638102	0.000020	0.000036	0.000020	

Figure 2: Tukey test result for play time.

It clearly shows the existence of differences between the base version and all the others. It is also worth paying attention to the lack of visible differences between the mode with extended graphics and challenges, which indicates a similar effect on the extension of the game time.

5.3. Satisfaction analysis

As in the case of time, the ANOVA test was chosen to analyze the satisfaction obtained by the players. The result of the Brown-Forsythe test (p = 0.00) and the normality of the dependent variable (an example of the normality chart shown in Figure 3) allowed for the correct performance of the test. The ANOVA result (p = 0.00) indicated the presence of differences between the versions, they are presented in figure 4 based on Tukey's test.

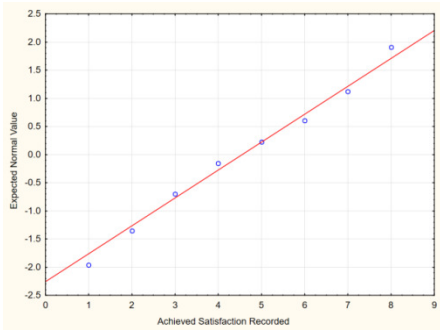


Figure 3: Normality chart for achieved satisfaction recorded in mode with extended graphics.

Tukey HSD test; Variable: Var2 (questionnaire.sta)
Marked differences are significant at p < .05000

Var1		{1} M=4.5060	{2} M=5.8715	{3} M=4.5489	{4} M=6.8046	{5} M=6.3655	{6} M=7.6078
0	{1}		0.000020	0.998947	0.000020	0.000020	0.000020
1	{2}	0.000020		0.000020	0.000020	0.000167	0.000020
2	{3}	0.998947	0.000020		0.000020	0.000020	0.000020
3	{4}	0.000020	0.000020	0.000020		0.001264	0.000020
4	{5}	0.000020	0.000167	0.000020	0.001264		0.000020
5	{6}	0.000020	0.000020	0.000020	0.000020	0.000020	

Figure 4: Tukey test result for player satisfaction.

The noted differences exist between all available game modes except the base version and the version with sound (p = 0.998). Considering the results received in the survey, according to players, these are the two least satisfactory modes.

5.4. Results summary

The answers to the research questions were based on a comparison with the results obtained for the base version of the game. This applies to both the average playing time (M = 370.15) and the average satisfaction obtained by the players (M = 4.5060). Table 1 also includes additional data from the research trials that may clear up some of the possible uncertainties. The number of towers purchased by the player may indicate interest in the game. Sold towers indicate a need for strategic thinking and planning. The number of improved towers allows to analyze the readability of the interface and the player's immersion level.

Table 1: Additional game data

Mode	Towers bought	Towers sold	Towers upgrades
Base	20.6364	3.8757	18.1056
Graphics	31.3917	5.4251	61.3610
Sound	12.7233	2.9198	2.3837
Scoreboard	31.5682	5.4559	53.1658
Player levels	29.7099	5.2313	41.3783
Achievements	34.0615	6.1163	66.5548

RQ1: Does the quality of the graphics extend the player's playing time?

RQ1 Summary: Graphics have a significant impact on the player's playing time (M = 969.90, p = 0.000020). Thus, more than a twofold increase in the obtained time was obtained. In addition, there was also a significant increase in the satisfaction obtained by players (M = 7.6078 p = 0.000020).

RQ2: Does the quality of the soundtrack extend the player's playing time?

RQ2 Summary: According to the test results, the use of sound also affects the player's playing time. In this case, however, it had a negative impact (M = 258.36, p = 0.007133). However, no significant effect on the satisfaction obtained by the players was noticed (M = 4.5489, p = 0.998947). In this case, the results of the additionally obtained data are interesting, in particular the number of upgrades made to the player's towers. Almost nine times lower results compared to the base version could indicate the lack of immersion in the player. One of the most likely reasons for this could be that the soundtrack does not match the player's [14] expectations and the atmosphere of the game.

RQ3: Does the presence of a scoreboard extend a player's playing time?

RQ3 Summary: The presence of the scoreboard had a significant impact on both the playing time (M = 812.71, p = 0.000020) and the satisfaction obtained by the players (M = 6.8046, p = 0.000020). This result may be surprising due to the fact that only 65 respondents described their type of player as Socializer.

RQ4: Does the presence of player levels extend the player's playing time?

RQ4 Summary: Introducing the possibility of player development by gaining new levels had a significant impact on increasing the players' playing time (M = 678.19, p = 0.000020) and on the satisfaction obtained from the game (M = 6.3655, p = 0.000020). When analysing additional data, it can also be noticed that the number of purchased and sold towers achieves results similar to other modes, where the game time was much longer. This may indicate an increase in the player's immersion by encouraging strategic thinking and constant interaction with the game.

RQ5: Does the presence of achievements and badges increase the player's playing time?

RQ5 Summary: The introduction of challenges and badges to the game had a significant impact on the player's playing time (M = 969.90, p = 000020) and on the satisfaction obtained by the players (M = 7.6068, p = 0.000020).

6. Conclusions

Starting with a summary of the results, in terms of their impact on the player's playing time, each of the prepared modes had a positive effect on extending the player's time, with the exception of the extended sound-track mode, which had a negative effect. Considering the satisfaction noted by players, the positive impact of all game modes can again be noticed, with the exception of the mode with the extended soundtrack, where there were no significant differences from the base version.

Due to the assumptions included in the protocol, allowing for the correct completion of the research session after three minutes of gameplay, the average game time in the base version (M = 370.15) turned out to be surprising. The values included in the additional data also allow to state that the players had no problems understanding the rules and the interface presented to them. This is important due to the presence of these factors in each of the other modes.

The use of improved graphics has been a huge success, with one of the highest average play times recorded. The results of the player satisfaction obtained in this way are weaker compared to other presented modes, but it was still an increase compared to the base version. The remaining results for this mode remain normal, showing no unexpected behaviour.

The mode with an extended soundtrack turned out to be a surprise, recording a similar satisfaction obtained by players with the game compared to the base mode, while achieving much worse results in terms of gameplay time. Worse results were also noted in this case for

all additional data, suggesting inappropriate selection of the soundtrack to the game's climate, resulting in problems with causing players to be immersed.

The results of the scoreboard mode also turned out to be unexpected. Despite the fact that the average game time is shorter than the mode with extended graphics or challenges, the statistics on the number of operations on towers, in particular their purchase, are similar. As this was one of the Operations that earned players points, this indicates player engagement as expected for this mode. The mode that allows the player to develop by acquiring levels and unlocking improvements turned out to be underestimated. It did not bring such a significant increase in the player's playing time as the other modes, but it had a large impact on the satisfaction obtained. This is most likely related to the greater difficulty of this mode, resulting from the need for strategic thinking and constant interaction with the game.

Despite the fact that only 155 people identified their player type as achiever, the achievements and badge mode scored the highest for game time, satisfaction, and each category from additional data. The challenges for this mode have been designed in such a way as to encourage the player to interact with the game as much as possible, gradually increasing the difficulty level and requirements.

In conclusion, the answer to the question of which of the gamification factors has the greatest impact on the player's game time and overall satisfaction obtained are the player's achievements and badges. This mode obtained the highest results in all possible categories. In order to extend the game time, it is also worth paying attention to the graphic style presented to the player and providing him with the opportunity to compete with other players based on the results obtained. Competition can also significantly increase the player's satisfaction with the game, and the ability for the player to develop and gain levels can work in a similar way. On the basis of the obtained results, it can be concluded that the soundtrack has the smallest, or even negative impact on the playing time, but it requires additional research due to the possibility of inappropriate matching of the sound to the game climate.

References

[1] J. Hamari, J. Koivisto, H. Sarsa, Does gamification work?--a literature review of empirical studies on gamification, 2014 47th Hawaii international conference on system sciences (2014) 3025-3034.

[2] S. Tobon, J. L. Ruiz-Alba, J. García-Madariaga, Gamification and online consumer decisions: Is the game over?, Decision Support Systems 128 (2020) 113167.

[3] T. Harwood, T. Garry, An investigation into gamification as a customer engagement experience environment, Journal of Services Marketing, 2019.

[4] M. Papoutsoglou, G. M. Kapitsaki, L. Angelis, Modeling the effect of the badges gamification mechanism on personality traits of Stack Overflow users, Simulation Modelling Practice and Theory 105 (2020) 102157.

[5] D. Dicheva, C. Dichev, G. Agre, G. Angelova, Gamification in education: A systematic mapping study, Journal of Educational Technology & Society 18 (2015) 75-88.

[6] M. Aparicio, T. Oliveira, F. Bacao, M. Painho, Gamification: A key determinant of massive open online course (MOOC) success, Information & Management 56 (2019) 39-54.

[7] K. Hicks, K. Gerling, G. Richardson, T. Pike, O. Burman, P. Dickinson, Understanding the effects of gamification and juiciness on players, 2019 IEEE Conference on Games (2019) 1-8.

[8] Y. L. Wu, Gamification design: A comparison of four m-learning courses, Innovations in Education and Teaching International 55 (2018) 470-478.

[9] J. Hamari, D. J. Shernoff, E. Rowe, B. Coller, J. Asbell-Clarke, T. Edwards, Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning, Computers in human behaviour 54 (2016) 170-179.

[10] F. Tomaselli, O. Sanchez, S. Brown, How to engage users through gamification: the prevalent effects of playing and mastering over competing, ICIS, 2015.

[11] J. Filippou, C. Cheong, F. Cheong, A model to investigate preference for use of gamification in a learning activity, 2018.

[12] J. Koivisto, J. Hamari, Demographic differences in perceived benefits from gamification, Computers in Human Behavior 35 (2015) 179-188.

[13] N. Xi, J. Hamari, Does gamification satisfy needs? A study on the relationship between gamification features and intrinsic need satisfaction, International Journal of Information Management 46 (2019) 210-221.

[14] T. Sanders, P. Cairns, Time perception, immersion and music in videogames, Proceedings of HCI 2010 24 (2010) 160-167.

[15] N. Gallacher, Game audio—an investigation into the effect of audio on player immersion, The Computer Games Journal 2 (2013) 52-79.

[16] S. Huiberts, Captivating sound: the role of audio for immersion in games, University of Portsmouth and Utrecht School of the Arts, Portsmouth, United Kingdom, 2010.

[17] S. Roohi, A. Forouzandeh, Regarding color psychology principles in adventure games to enhance the sense of immersion, Entertainment Computing 30 (2019) 100298.

[18] A. M. Connor, T. J. Greig, J. Kruse, Evaluating the impact of procedurally generated content on game immersion, The Computer Games Journal 6 (2017) 209-225.

[19] R. McGloin, K. Farrar, M. Krcmar, Video games, immersion, and cognitive aggression: does the controller matter?, Media psychology 16 (2013) 65-87.

[20] A. S. Bastos, R. F. Gomes, C. C. Dos Santos, J. G. R. Maia, Assessing the experience of immersion in electronic games, 2017 19th Symposium on Virtual and Augmented Reality (SVR) (2017) 146-154.

[21] K. Jin, T. Igarashi, The effects of narcissism and self-esteem on immersion in social network games and massively multiplayer online role-playing games, Shinrigaku Kenkyu: The Japanese Journal of Psychology 87 (2016) 1-11.

[22] S. Kumari, C. Power, P. Cairns, Investigating uncertainty in digital games and its impact on player immersion, Extended abstracts publication of the annual symposium on computer-human interaction in play (2017) 503-509.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 8-12
Received: 4 October 2021
Accepted: 26 October 2021

Comparative analysis of the time performance of database queries in C# language

Analiza porównawcza wydajności czasowej zapytań bazodanowych w języku C#

Tomasz Nowicki*, Sebastian Tomczak*, Grzegorz Kozieł

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

Abstract

There are many computer applications in the world that use databases to store, process, and use data. That translates into many different ways of handling these databases. It is therefore difficult to choose a solution that meets the needs of the user. This article compares three C# solutions in terms of time efficiency: the Entity Framework Core application framework, pure SQL queries, and parameterized Prepared Statement queries. The results obtained in the course of the research has shown that the fastest solution is the use of non-parameterised SQL queries. The use of Entity Framework Core is the slowest of the three tested solutions.

Keywords: C#; Entity Framework Core; SQL Database; Prepared Statement; .NET Framework

Streszczenie

Na świecie istnieje duża liczba aplikacji komputerowych wykorzystujących bazy danych celem utrwalania, przetwarzania i wykorzystania danych, co przekłada się na wiele różnych sposobów obsługi tychże baz. Trudno jest więc wybrać rozwiązanie spełniające potrzeby użytkownika. W niniejszym artykule porówanano pod kątem wydajności trzy rozwiązania dla języka C#: szkielet aplikacji Entity Framework Core, zapytania SQL przesyłane w postaci jednego łańcucha znaków, oraz sparametryzowane zapytania Prepared Statement. Uzyskane w toku badań wyniki pozwoliły określić, że najszybszym rozwiązaniem jest wykorzystanie niesparametryzowanego zapytania SQL. Wykorzystanie Entity Framework Core jest najwolniejszym z trzech badanych rozwiązań.

Słowa kluczowe: C#; Entity Framework Core; SQL Database; Prepared Statement; .NET Framework

*Corresponding author

Email address: tomasz.nowicki1@pollub.edu.pl, sebastian.tomczak@pollub.edu.pl

1. Wstęp

Tworzenie aplikacji, niezależnie od tego czy są to aplikacje mobilne, desktopowe czy internetowe, jest blisko powiązanie z użyciem baz danych aby utrwalić, przetworzyć oraz wykorzystać dane nie tylko uzyskane w czasie działania aplikacji, ale również dane niezbędne do działania tychże aplikacji. Nie jest więc zaskakujące, że wraz w upływem czasu powstały coraz to bardziej zróżnicowane sposoby na ułatwienie i usprawnienie połączenia i komunikacji pomiędzy aplikacją oraz bazą danych [1]. Sytuacja ta stawia programistę przed wyzwaniem, który ze sposobów dostępnych na rynku jest tym, który spełnia założone oczekiwania oraz wymagania.

W niniejszym artykule zdecydowano się zestawić ze sobą trzy popularne rozwiązania dostępne dla języka programistycznego C#. W zestawieniu wzięto pod uwagę czas potrzebny na wykonanie danego zapytania. W efekcie przeprowadzonego badania zidentyfikowano najwydajniejsze rozwiązanie. Tak przeprowadzone badanie oraz wnioski wyciągnięte na jego podstawie pozwolą na wybranie najszybszego rozwiązania do komunikacji i obsługi aplikacji z bazą danych. Systemem bazodanowym, na której zostały przeprowadzone badania jest system Microsoft SQL Server.

Rozwiązania przedstawione w artykule to szkielet aplikacji Entity Framework Core, zapytania SQL przesyłane w postaci łańcucha znaków z aplikacji, oraz sparametryzowane zapytania Prepared Statement. Entity Framework Core [2], jako rozszerzalna część technologii Entity Framework, wykorzystuje nowocześniejsze oraz obiektowe podejście przy pracy z bazami danych jakim jest mapowanie obiektowo-relacyjne (ang. Object-Relational Mapping, ORM) [3]. Pozostałe dwa rozwiązania są rozwiązaniami starszymi, gdzie programista komunikuje się z serwerem bazodanowym bezpośrednio za pomocą zapytań SQL, gdzie w pierwszym sposobie są to zapytania SQL przesyłane do serwera bazodanowego w postaci łańcucha znaków zawierającego komendy SQL oraz parametry i dane, natomiast drugim sposobem są zapytania sparametryzowane tzw. Prepared Statement.

2. Wybrane technologie

2.1. Język C#

Język C# jest zorientowanym obiektowo, nowoczesnym językiem programowania działającym w środowisku .NET [4], umożliwiającym tworzenie typów bezpiecznych. Język ten udostępnia również konstrukcje językowe bezpośrednio obsługujące koncepcje obiektowości. Wbudowane w niego zostały również takie funkcje

jak automatyczne odzyskiwanie pamięci zajętej przez nieosiągalne i nieużywane obiekty, wyrażenia lambda obsługujące techniki programowania funkcjonalnego, ustrukturyzowana i rozszerzalna obsługa wyjątków oraz wykrywania i odzyskiwania błędów. Ponadto, jedną z wbudowanych funkcji jest składnia zapytań LANGUAGE Integrated Query (LINQ) [5], tworzące wzorzec pracy z danymi z dowolnego źródła.

2.2. Entity Framework Core

Entity Framework Core jest lekką, otwartą, międzyplatformową i rozszerzalną wersją technologii Entity Framework [6], w której nacisk położono na sprawny dostęp do danych. Entity Framework Core może służyć jako maper obiektowo-relacyjny (ang. O/RM - Object/Relational Mapper), pozwalający na pracę z bazą danych z użyciem obiektów .NET oraz zmniejszenie zapotrzebowana na większość kodu dostępu do danych, o który zadbać musiał programista. Technologia ta posiada obsługę wielu silników bazodanowych takich jak: MySQL, Oracle DB, PostreSQL, Firebird, oraz wykorzystywanym w badaniach Microsoft SQL Serverem.

2.3. Microsoft SQL Server

System zarządzania relacyjną bazą danych (ang. Relational Database Management System, RDBMS) firmy Microsoft składa się między innymi z silnika bazy danych, przez co cały system jest często postrzegany jako baza danych. SQL Server, jako system RDBMS, jest zaimplementowany i pracuje w architekturze klient/serwer, gdzie każdy komponent jest w stanie pracować samodzielnie i niezależnie od pozostałych. Cechą SQL Servera jest obsługa odbywająca się głównie z wykorzystaniem narzędzi zarządzania z wbudowanym interfejsem graficznym użytkownika, np. z wykorzystaniem narzędzia SQL Server Management Studio. Komunikacja pomiędzy użytkownikiem a silnikiem bazy danych odbywa się poprzez strukturalny język zapytań (ang. Structured Query Language, SQL), który został również rozbudowany o autorskie rozszerzenia firmy Microsoft oraz występuje w dialekcie Transact-SQL.

3. Metodyka badawcza

3.1. Środowisko badawcze

Aby przeprowadzić badania, najpierw przygotowano odpowiednio skonfigurowane środowisko badawcze z lokalnie zainstalowanym potrzebnym oprogramowaniem oraz wymaganymi narzędziami pozwalającymi na przeprowadzenie tychże badań oraz rejestrację ich wyników na maszynie wirtualnej. Wykorzystane oprogramowanie oraz parametry środowiska badawczego przedstawiono w tabelach 1 i 2.

Bazą danych wykorzystaną w badaniach jest udostępniona przez firmę Microsoft na licencji MIT baza danych Adventure Works 2019 [7] przedstawiająca fikcyjny międzynarodowy zakład produkcyjny. Ze względu na rozmiar tejże bazy, zdecydowano się na skorzystanie z zaledwie jej fragmentu, który jest przedstawiony na rysunku 1. Wykorzystane tabele wraz z ich liczbą rekordów pokazane zostały w tabeli 3.

Tabela 1: Wykorzystane oprogramowanie

Nazwa narzędzia	Wersja
SQL Server Management Studio	15.0.18384.0
Microsoft SQL Server 2019	15.0.2000.5
Microsoft Visual Studio Community 2019	16.8.4
.Net 5.0	5.0.202
VirtualBox	6.1.22 r144080 (Qt5.6.2)

Tabela 2: Parametry środowiska badawczego

System operacyjny	Windows 10 Education – Wersja 10.0.18363
System operacyjny maszyny wirtualnej	Windows 10 Education – Wersja 10.0.19041
Pamięć RAM	Pamięć Corsair Vengeance LPX, DDR4, 16 GB, 3200MHz, CL16
Pamięć RAM maszyny wirtualnej	8 GB
Procesor	Procesor AMD Ryzen 5 1600 AF, 3.2GHz, 16 MB
Procesor maszyny wirtualnej	6 rdzeni logicznych, 3.2GHz, 16 MB
Dysk	Dysk SSD GoodRam 240 GB 2.5" SATA III (SSDPR-IRIDPRO-240)
Dysk maszyny wirtualnej	65GB HDD
Pamięć wideo maszyny wirtualnej	128MB

Tabela 3: Wykorzystane tabele wraz z ich liczbą rekordów

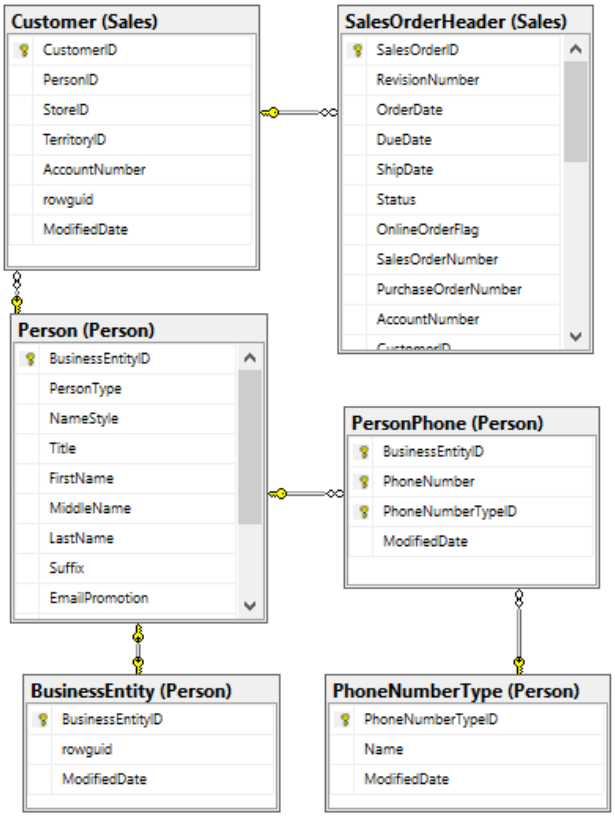
Nazwa tabeli	Liczba rekordów
SalesOrderHeader	31465
Customer	19820
Person	19972
PersonPhone	19972
PhoneNumberType	3
BusinessEntity	20777

3.2. Scenariusze badawcze

Aby przeprowadzić badania i zebrać wyniki potrzebne do porównania wydajności czasowej zapytań bazodanowych przygotowano 5 scenariuszy badawczych, różniących się złożonością zapytań SQL. Przy wyborze zapytań kierowano się przede wszystkim ich wykorzystaniem biznesowym oraz praktycznością. Każdy ze scenariuszy został uruchomiony na bazie danych Adventure Works 2019 działającej na serwerze Microsoft SQL Server 100 razy aby zebrać pulę wyników poddanej następnie analizie. Scenariusze badawcze przedstawione zostały w tabeli 4, podczas gdy ich kod pokazany został na listingach 1-5.

Tabela 4: Scenariusze badawcze

Nr.	Zapytanie
1	Wybranie wszystkich zamówień poszczególnych klientów.
2	Wybranie wszystkich zamówień poszczególnych klientów wraz z przypisanym do nich numerem telefonu i typem numeru telefonu.
3	Wybranie wszystkich zamówień poszczególnych klientów o danej wartości Statusu.
4	Wybranie wszystkich zamówień poszczególnych klientów wraz z przypisanym do nich numerem telefonu i typem numeru telefonu, zawężone do konkretnego typu numeru telefonu.
5	Policzenie liczby zamówień dla klientów osobowych reprezentowanych przez ich numer identyfikacyjny klienta, bądź osoby.



Rysunek 1: Fragment schematu wykorzystywanej bazy danych.

Listing 1: Kod zapytania Nr. 1

```
SELECT SalesOrderID,
SalesOrderNumber, PurchaseOrderNumber,
OrderDate, [Status], AccountNumber,
CustomerID, SubTotal, TotalDue
FROM Sales.SalesOrderHeader
```

Listing 2: Kod zapytania Nr. 2

```
SELECT SOH.SalesOrderID, SOH.CustomerID,
C.AccountNumber, C.PersonID,
P.Title, P.FirstName, P.LastName,
PP.PhoneNumber, PNT.PhoneNumberTypeID, PNT.Name
FROM Sales.SalesOrderHeader SOH
JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
JOIN Person.PersonPhone PP
ON P.BusinessEntityID = PP.BusinessEntityID
JOIN Person.PhoneNumberType PNT
ON PP.PhoneNumberTypeID = PNT.PhoneNumberTypeID
```

Listing 3: Kod zapytania Nr. 3

```
SELECT SalesOrderID, SalesOrderNumber,
PurchaseOrderNumber, OrderDate,
[Status], AccountNumber, CustomerID,
SubTotal, TotalDue
FROM Sales.SalesOrderHeader
WHERE Status = 5
```

Listing 4: Kod zapytania Nr. 4

```
SELECT SOH.SalesOrderID, SOH.CustomerID,
C.AccountNumber, C.PersonID,
P.Title, P.FirstName, P.LastName,
PP.PhoneNumber, PNT.PhoneNumberTypeID, PNT.Name
FROM Sales.SalesOrderHeader SOH
JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
JOIN Person.PersonPhone PP
ON P.BusinessEntityID = PP.BusinessEntityID
JOIN Person.PhoneNumberType PNT
ON PP.PhoneNumberTypeID = PNT.PhoneNumberTypeID
WHERE PNT.PhoneNumberTypeID = 1
```

Listing 5: Kod zapytania Nr. 5

```
SELECT SOH.CustomerID, C.PersonID,
P.Title, P.FirstName, P.LastName,
COUNT(SOH.SalesOrderID) AS OrderCount
FROM Sales.SalesOrderHeader SOH
JOIN Sales.Customer C
ON SOH.CustomerID = C.CustomerID
JOIN Person.Person P
ON C.PersonID = P.BusinessEntityID
GROUP BY SOH.CustomerID, C.PersonID,
P.Title, P.FirstName, P.LastName
```

3.3. Aplikacja testowa

Do przeprowadzenia serii badań z wykorzystaniem przedstawionych w tabeli 3 scenariuszy badawczych przygotowana została aplikacja napisana w języku C# składająca się z 3 projektów, każdy z nich związany odpowiednio z danym badanym rozwiązaniem. W każdym projekcie zawarty jest ten sam mechanizm pomiaru czasu, powielony celem odseparowania bibliotek od siebie, aby potencjalne referencje między nimi nie wpływały negatywnie na wyniki badań.

Aplikacja składa się na zbiór testów oznaczonych uprzednio zaprogramowanym atrybutem “Performance”. Głównym elementem każdego z testów jest część wykonawcza, która zależnie od typu testu zawiera inne zapytanie. W opisywanym rozwiązaniu znajduje się 5 testów - po jednym na każde zapytanie w każdym z projektów. W zależności od danej implementacji, “SQL”, “Prepared Statement” czy “EFC” kod wykonawczy jest różny. W pierwszym z nim znajduje się zapytanie w postaci kodu SQL oraz mapowania zwróconych danych do obiektów.

Kolejne z rozwiązań rozszerza wcześniejsze o dodanie do niektórych zapytań w postaci kodu SQL argumentów, które są w odpowiedni sposób przekazywane do zapytania. Ponownie zwrócony rezultat jest mapowany do pamięci. Ostatni projekt obejmuje wykorzystanie Entity Framework Core, zapytania tym razem realizowane są za pomocą LINQ wykorzystywanego na specjalnym obiekcie Context, całość za sprawą wykorzystania ORM obejmuje pracę na obiektach, wynik mapowany jest po stronie EFC, a jego wynik zapisywany jest do zmiennej analogicznie do poprzednich przypadków.

Atrybut “Performance” odpowiada za programowe zliczanie czasu trwania danych akcji - zapytań, we wcześniej przygotowanej bazie danych, a więc ostatecznie każde wykonanie dowolnego testu z dowolnego projektu skutkuje wykonaniem zapytania za pomocą danej technologii i utrwalenie jego czasu realizacji w celu umożliwienia dalszej analizy.

4. Wyniki badań

Przeprowadzenie serii badań przy wykorzystaniu aplikacji testowej opisanej w rozdziale 3.3 zgodnie ze scenariuszami badawczymi ukazanymi w rozdziale 3.2 poskutkowało następującymi wynikami przedstawionymi w tabelach 5-9, oraz na rysunkach 2-6. Wartościami, na których zdecydowano się skupić porównując je ze sobą, są mediana oraz czas średni wykonania zapytań w milisekundach. Natomiast w celu zobrazowania

uzyskanych wyników posłużono się wykresami pudełkowymi z wąsem.

Tabela 5: Czas wykonania zapytania Nr. 1

Metoda	Mediana [ms]	Czas średni [ms]
SQL	122,19	122,89
Prepared Statement	121,01	125,25
Entity Framework Core	1258,51	1354,19

Tabela 6: Czas wykonania zapytania Nr. 2

Metoda	Mediana [ms]	Czas średni [ms]
SQL	270,83	275,84
Prepared Statement	271,55	752,88
Entity Framework Core	1540,15	2004,88

Tabela 7: Czas wykonania zapytania Nr. 3

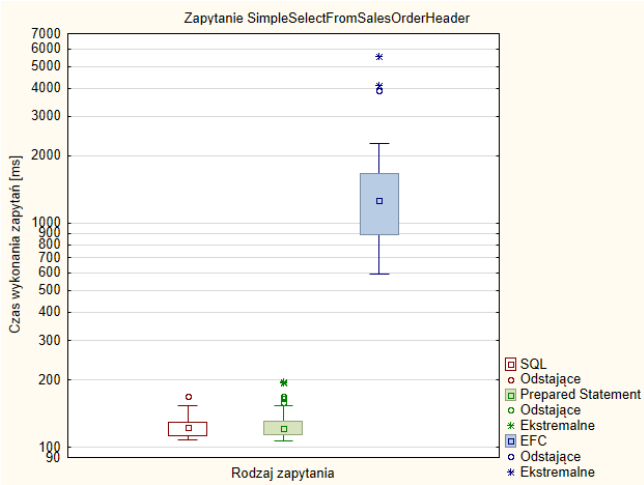
Metoda	Mediana [ms]	Czas średni [ms]
SQL	118,39	120,74
Prepared Statement	141,42	328,32
Entity Framework Core	1252,05	1325,35

Tabela 8: Czas wykonania zapytania Nr. 4

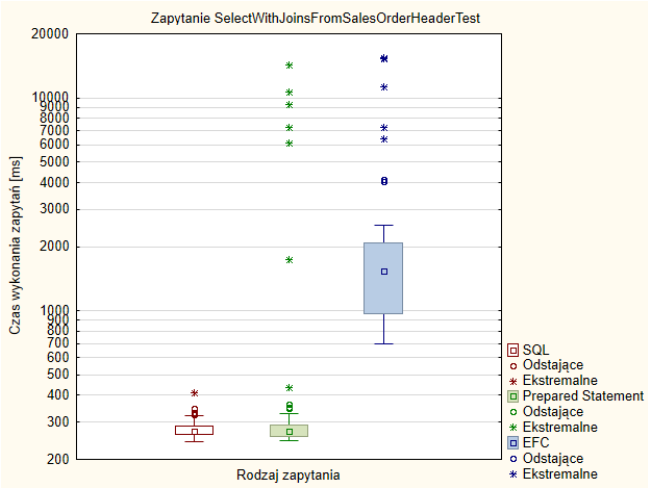
Metoda	Mediana [ms]	Czas średni [ms]
SQL	189,15	192,75
Prepared Statement	191,19	542,15
Entity Framework Core	1588,49	2228,27

Tabela 9: Czas wykonania zapytania Nr. 5

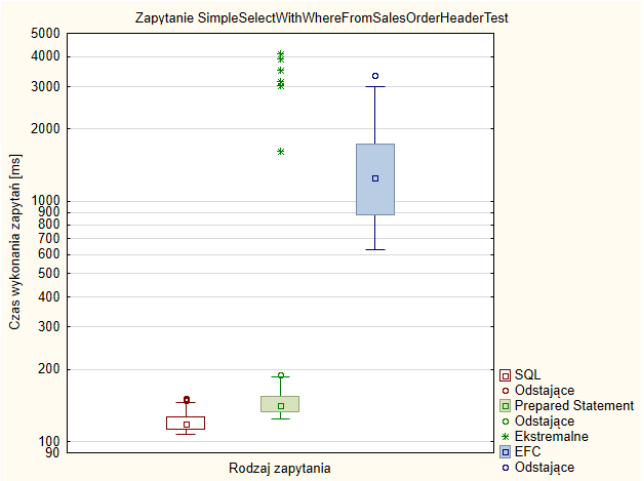
Metoda	Mediana [ms]	Czas średni [ms]
SQL	80,38	83,65
Prepared Statement	84,91	528,24
Entity Framework Core	1440,96	2125,78



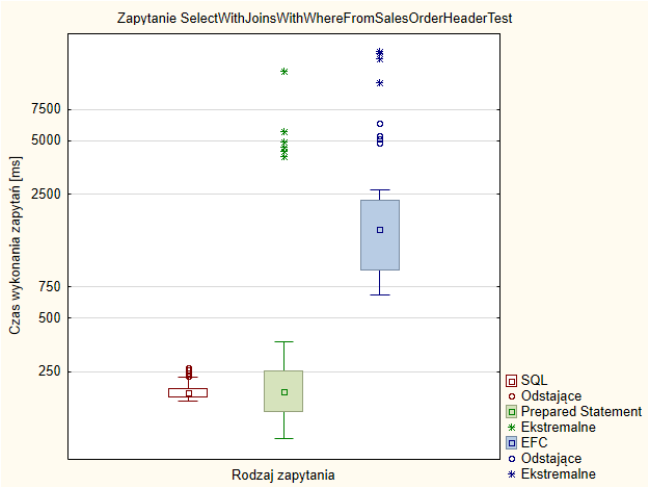
Rysunek 2: Zestawienie wyników badań zapytania Nr. 1.



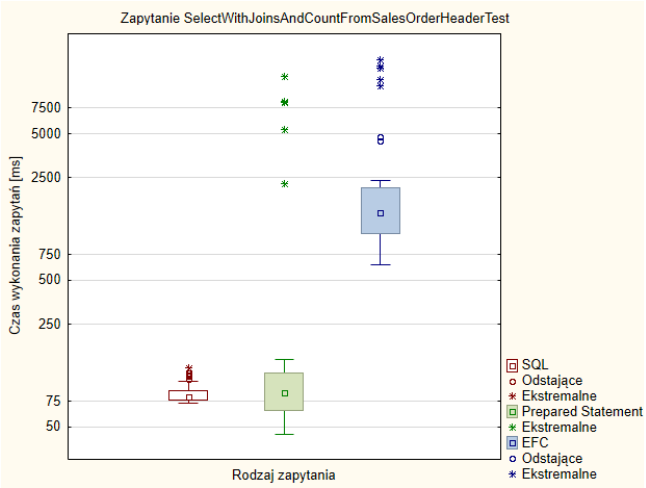
Rysunek 3: Zestawienie wyników badań zapytania Nr. 2.



Rysunek 4: Zestawienie wyników badań zapytania Nr. 3.



Rysunek 5: Zestawienie wyników badań zapytania Nr. 4.



Rysunek 6: Zestawienie wyników badań zapytania Nr. 5.

5. Wnioski

W przeprowadzonych badaniach z wykorzystaniem rozwiązań SQL, Prepared Statement oraz Entity Framework Core realizowanych za pomocą aplikacji opartej na testach napisanej w języku C# można jednoznacznie stwierdzić, że na tle dwóch pozostałych rozwiązań, Entity Framework Core jest rozwiązaniem najwolniejszym. Wykorzystanie tej technologii odciąża programistę od znajomości języka SQL z racji bazowania na obiektowo-relacyjnym mapowaniu. co może okazać się dużą zaletą.

Wartym uwagi jest fakt, że Entity Framework Core wykonując proces translacji zapisu programistycznego (LINQ) do kodu SQL dokonał koniecznych operacji dzięki któremu ostateczne zapytania SQL miały zbliżoną postać do zapytań wykorzystanych w przypadku SQL oraz Prepared Statement.

Drugim w kolejności rozwiązaniem jest Prepared Statement. Warto zauważyć, że w zapytaniach, w których występował warunek “Where”, z racji wykorzystania parametru SQL, czyli Prepared Statement, cały proces trwał dłużej niż w przypadku zwykłych zapytań SQL.

Podejście związane z wykorzystaniem zwykłych zapytań SQL cechuje najmniejszy rozrzut czasowy całego procesu ze względu na prosty sposób implementacji rozwiązania, dodatkowo bezpośrednio wskazując na fakt, że ta metoda była najszybsza na tle pozostałych badanych.

Literatura

[1] S. Cvetković, D. Janković, A comparative study of the features and performance of orm tools in a. net environment, International Conference on Object and Databases, Springer, Berlin, Heidelberg, (2010) 147-158.

[2] H. Schwichtenberg, Modern Data Access with Entity Framework Core, Apress, Essen, Germany, 2018.

[3] D. Bowers, C. Ireland, M. Newton, K. Waugh, Understanding object-relational mapping: A framework based approach, International lournal On Advances in Software 2.2, (2009) 202-216.

[4] B. Meyer, .NET is coming [Microsoft Web services platform], Computer 34.8 (2001) 92-97.

[5] E. Meijer, The world according to LINQ, Communications of the ACM 54.10 (2011) 45-51.

[6] A. Adya, J. A. Blakeley, S. Melnik, S. Muralidhar, Anatomy of the ado. net entity framework, Proceedings of the 2007 ACM SIGMOD international conference on Management of data (2007) 877-888.

[7] M. Mitri, Teaching Tip: Active Learning via a Sample Database: The Case of Microsoft's Adventure Works, Journal of Information Systems Education 26.3 (2015) 177-185.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 13–17
Received: 7 October 2021
Accepted:21 October 2021

Comparison of selected configuration management systems in Linux

Porównanie wybranych systemów zarządzania konfiguracją w systemie Linux

Szymon Dudziak*, Natalia Barbara Jakubczak, Maciej Pańczyk

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

Abstract

The purpose of this article is to compare selected configuration management systems on Linux. These systems had to meet the condition of integration with Linux. The two of the most popular systems were chosen which are Puppet and Ansible. The comparison was based on a simple system configuration using these two systems in several aspects: installation, file and folder management, package management, user management, configuration of the Apache server and Firewall. Through this comparison, it was possible to determine which system is more suitable for a novice Linux system administrator.

Keywords: Ansible; Puppet; comparison; configuration management systems; Linux

Streszczenie

Celem artykułu jest przeprowadzenie porównania wybranych systemów zarządzania konfiguracją w systemie Linux. Systemy te musiały spełnić warunek integracji z systemem Linux. Zostały wybrane dwa z pośród najpopularniejszych systemów jakimi są Puppet oraz Ansible. Porównanie zostało oparte na prostej konfiguracji systemu z wykorzystaniem tych dwóch systemów w kilku aspektach: instalacji, zarządzania plikami i folderami, zarządzania pakietami, zarządzania użytkownikami, konfiguracji serwera Apache oraz zapory sieciowej. Dzięki takiemu porównaniu można było stwierdzić który system bardziej nadaje się dla początkującego administratora systemów Linux.

Słowa kluczowe: Ansible; Puppet; porównanie; zarządzanie konfiguracją; Linux

*Corresponding author

Email address: szymon.dudziak@pollub.edu.pl (S. Dudziak)

1. Wstęp

Wraz ze wzrostem ilości maszyn zarządzanych przez administratorów była potrzeba utworzenia wydajnych systemów wspomagających ich pracę. Takie oprogramowanie miało w znacznym stopniu skrócić i uprościć pracę administratorów.

Już na samym początku administracji systemami administratorzy próbowali uprościć swoją pracę przez tworzenie skryptów. W tym celu wykorzystywano znane im języki programowania. Przykładem takiego języka jest Python, który do teraz jest jednym z najbardziej popularnych języków programowania.

Zarządzanie konfiguracją jest bardzo rozbudowanym zagadnieniem, na które składa się wiele czynników. W trakcie tego procesu administrator ma za zadanie nadzorować zmiany wprowadzane do systemu, raportować jego stan jak i weryfikować jego kompletność oraz spójność. Ze względu na ciągły rozwój obszaru IT wymagane było utworzenie wydajnych systemów, które wspomagały administratora w codziennej pracy podczas konfiguracji nowych i utrzymywania starych jednostek fizycznych na tym samym poziomie technologicznym oprogramowania.

W tym momencie rynek oferuje wiele rozwiązań wspomagających zarządzanie infrastrukturą IT opartych zarówno na systemie Windows jak i różnych dystrybucjach Linux-a. Systemy te są w dalszym ciągu rozwijane i ulepszane a co za tym idzie oferują coraz to nowsze funkcje. Należałoby sprawdzić, czy systemy te wraz ze wzrostem funkcjonalności są w dalszym ciągu przyjazne dla początkującego użytkownika.

2. Cel i zakres badań

Celem badań zawartych w niniejszym artykule było sprawdzenie łatwości przeprowadzenia konfiguracji systemu Linux z wykorzystaniem Ansible oraz Puppet. Zakres badań obejmował instalację oprogramowania, zarządzania plikami i folderami, zarządzania pakietami na podstawie instalacji pakietu z interfejsem graficznym Xfce oraz zarządzania użytkownikami. Dla każdego z wymienionych oprogramowań utworzono niezbędne pliki konfiguracyjne. Na podstawie odczuć autorów stwierdzono, które oprogramowanie jest lepsze w danym aspekcie.

3. Przegląd literatury

Zapotrzebowanie na zaawansowane systemy zarządzania konfiguracją stale rośnie. Aby zapobiec dublowaniu badań przeprowadzono przegląd literatury.

Grupa badaczy jakimi są Sanjeev Thakur, Chand Gupta, Nishant Singh oraz Soloman Geddam w pracy pod tytułem „Mitigating and Patching System Vulnerabilities Using Ansible: A Comparative Study of Various Configuration Management Tools for IAAS Cloud" [1] podjęli się porównania następujących narzędzi jakimi są Ansible, Chef, Puppet oraz Salt. Wynikiem ich badań było stwierdzenie, że najlepiej rozwijającym się narzę-

dziem do zarządzania konfiguracją jest Ansible. Według autorów wynika to z faktu, że Ansible wspiera oraz oferuje szeroką gamę szybkich wdrożeń oraz napraw.

W pracy pod tytułem „ Software Configuration Management: A comparison of Chef, CFEngine and Puppet" [2] Fredrik Önnberg porównał narzędzia Chef, CFEngine oraz Puppet. Problemem jakim się zajął było sprawdzenie, czy wymienione w pracy oprogramowania wspierają wskazane w pracy systemy operacyjne. W pracy podano ile pakietów jest instalowanych podczas instalacji poszczególnych narzędzi.

„Learning Ansible 2 – Second Edition" [3] autorstwa Fabia Alessandra Locati'ego pozwala na poznanie procesu automatyzacji konfiguracji na podstawie najnowszej (dostępnej w chwili wydania) wersji narzędzia AnsibleGet. W oparciu o wcześniej wymienione oprogramowanie twórca zaznajamia czytelnika z podstawowymi elementami Ansible w skład których wchodzą np. moduły, czy podręczniki.

W trzecim wydaniu książki autorstwa Johna Arundel'a „Puppet 5 Begginer's Guide" [4] zawarto wszystkie niezbędne informacje odnoszące się do oprogramowania jakim jest Puppet. Książka ta zawiera praktyczne przykłady wykorzystania tego narzędzia, które wspomagają użytkownika w procesie konfiguracji oraz instalacji serwera. Dodatkowo dzięki pokazaniu kolejnych kroków postępowania przedstawiane są kluczowe elementy wykorzystywane w pracy administratora systemów. W książce znajdują się także informacje odnośnie instalacji pakietów, konfiguracji plików, tworzenia użytkowników i innych kluczowych elementów pozwalających na bezproblemową pracę podczas konfiguracji systemu.

W artykule pod tytułem „Ansible vs Puppet" [5] Jarek Grzabel przedstawia różnice pomiędzy Ansible a Puppet. Według autora Ansible jest bardziej podatny na przypadkowe pomyłki od Puppet. W Puppet jest ustawiony interwał czasowy, w którym można jeszcze wprowadzić zmiany w kodzie, w Ansible wszystkie zmiany są wykonywane natychmiastowo.

Aayushi Johari w swoim artykule „Chef vs Puppet vs Ansible vs Saltstack: Which Works Best For You?" [6] porównuje podane w tytule systemy pod kątem dostępności, łatwości instalacji, zarządzania konfiguracją czy kosztem utrzymania. Ansible na tle pozostałych produktów odznacza się łatwością instalacji oraz niską ceną. Zarówno wykorzystując SaltStack jak i Ansible w łatwy sposób można przeprowadzić konfigurację systemu. W artykule zaznaczono również popularność poszczególnych systemów. Najbardziej zyskującym popularność produktem jest Ansible.

Jak do tej pory żaden badacz nie odniósł się do przeanalizowania wyboru z punktu widzenia początkujących użytkowników Ansible lub Puppet jako oprogramowania do zarządzania konfiguracją.

4. Obiekty badań

W pracy użyto dwóch systemów jakimi są Ansible, który został utworzony w 2012roku przez Michal'a DeHaan'a [7], oraz Puppet, utworzony w 2005 roku przez Luke Kanies'a [8], natomiast pierwszą wersję wydano w 2011 roku.

W tabeli 1 zawiera podstawowe informacje o wybranych systemach [9].

Tabela 1: Porównanie systemu Ansible oraz Puppet

	System	Ansible	Puppet
	Strona WWW	ansible.com	puppet.com
Języki i formaty	**Implementacja**	Python	Ruby
	Konfiguracja	YAML	Własny
Demony	**Szablon**	Jinja	Embedded Ruby
	Serwer	Nie	Opcjonalnie
	Klient	Nie	Opcjonalnie

4.1. Wstępne porównanie systemów Ansible i Puppet

Ansible jest otwartoźródłowym oprogramowaniem służącym do zarządzania konfiguracją. Można go wykorzystać do konfiguracji systemów Windows, Linux, MacOS i innych systemów UNIXopodobnych. Napisany został w oparciu o język Python.

Do komunikacji z drugim systemem wykorzystuje protokół SSH, PowerShell lub inne API. Nie wymaga instalacji agenta, co powoduje wzrost bezpieczeństwa.

Konfiguracja w Ansible wykorzystuje uprzednio przygotowane pliki oparte o język YAML lub poprzez wprowadzanie poleceń AdHoc. Ansible wykorzystuje szablony Jinja.

Podstawowym elementem konfiguracji użytym w Ansible jest moduł. Istnieje możliwość napisania modułu w oparciu o język skryptowy taki jak Python, czy Bash. Dzięki wykorzystaniu modułu wynik końcowy zawsze zostanie zrównany do jednakowego stanu. Działanie takie powstaje na przykład po przywróceniu działania systemu po awarii. Po wykonaniu modułu wszelkie elementy z nim związane są czyszczone.

Puppet jest otwartoźródłowym oprogramowaniem przeznaczonym do zarządzania konfiguracją. Można go wykorzystać do konfiguracji systemów operacyjnych Windows, Linux, czy MacOS oraz innych. Oparty on został o język Ruby, który wykorzystywany jest przy pisaniu manifestów.

Komunikacja między maszynami zachodzi poprzez agenta. Z tego powodu potrzebna jest głębsza konfiguracja tego oprogramowania do poprawnego działania.

W celu konfiguracji Puppet wykorzystuje własny format plików konfiguracyjnych. Istnieje możliwość wykorzystania wcześniej przygotowanych szablonów napisanych w języku Ruby. Wraz z rozwojem Puppet-a zaistniała możliwość konfiguracji poprzez wprowadzanie poleceń AdHoc.

Język programowania Puppet opiera się na paradygmacie deklaratywnym. Oznacza to, że opisuje stan końcowy konfigurowanej maszyny po wprowadzeniu

zmian. Język ten opisuje stan systemu komputerowego w kategoriach zasobów. Są one reprezentowane przez podstawowe konstrukcje sieci oraz systemu operacyjnego. Administrator łączy zasoby w manifesty, które zawierają finalny stan systemu. Pliki manifestów przechowywane są na maszynie serwerowej. Instrukcje konfiguracyjne dla agentów są wynikiem kompilacji plików manifestowych.

5. Metoda badań

Badanie łatwości wykonywania konfiguracji przeprowadzono na maszynach opartych na tym samym systemie operacyjnym. Końcowym efektem było zbudowanie poprawnie działającego środowiska operacyjnego zawierającego niezbędne podstawowe oprogramowanie.

Porównywana była łatwość konfiguracji z wykorzystaniem plików konfiguracyjnych, proces ich pisania oraz dostęp do dokumentacji. Wynikiem badań była subiektywna ocena tworzona z punktu widzenia niedoświadczonych administratorów. W celu zrealizowania badań nie wykorzystano zewnętrznej grupy badawczej.

6. Platforma testowa

Badania były przeprowadzone z wykorzystaniem maszyn wirtualnych. Poniżej zaprezentowano jednostkę fizyczną na której zainstalowano poszczególne maszyny.

- procesor: AMD Ryzen 5 3600X 3.8GHz, 6 rdzeni, 12 wątków,
- pamięć RAM: 16GB DDR4 3200MHz,
- dysk: Patriot 1TB M.2 PCIe NVMe,
- system operacyjny: Windows 10 Education,
- karta graficzna: NVIDIA GeForce GTX 1660 Ti.

Na każdej z maszyn zainstalowano tę samą wersję systemu operacyjnego CentOS 8 Stream.

7. Obszary testowe

Dla każdego z systemów wyodrębniono wspólne obszary testowe:

- Instalacja oprogramowania,
- Zarządzanie plikami i folderami,
- Zarządzania pakietami,
- Zarządzanie użytkownikami,
- Konfigurowanie serwisów.

Instalacja opierała się na przygotowaniu środowiska do dalszej pracy. W niej skonfigurowano odpowiednio każdy systemów.

Zarządzanie plikami oraz folderami opierało się o wykonanie odpowiednio przygotowanych skryptów które automatycznie tworzyły, edytowały lub usuwały zadany element.

Zarządzanie pakietami wymagało przygotowanie skryptu, który zainstaluje dany pakiet w odpowiedniej wersji. Na potrzeby badań wykorzystano najnowsze wersje wybranych pakietów.

Zarządzanie użytkownikami pozwoliło na tworzenie, edycję oraz usuwanie użytkowników. Pozwalało na wybranie ścieżki do katalogu domowego, przypisaniu użytkownikowi dowolnego hasła oraz wybranych uprawnień.

Konfigurowanie serwisów w głównej mierze opierało się na restartowaniu i uruchamianiu usług. Dodatkowo wykonano konfigurację zapory ogniowej dokonując w niej wpis umożliwiający bezproblemową pracę serwera Apache.

Listing 1: Przykładowy skrypt konfigurujący dla oprogramowania Ansible

```
[root@ansible ~]# cat user.yml
---

- hosts: all
  gather_facts: yes
  vars_files:
          - haslo.yml

  tasks:
          - name: Tworzenie nowej grupy
            group:
                    name: "test-group"
                    gid: 3500
                    state: present

          - name: Tworzenie uzytkownikow
            user:
                    name: "{{ item }}"
                    password: "{{ user_password
                        | password_hash('sha512') }}"
                    update_password: on_create
                    groups: "test-group"
                    shell: /bin/bash
            loop:
                  - test
                  - teststudent
[root@ansible ~]# cat haslo.yml
user_password: test
```

Na Listingu 1 zaprezentowano skrypt napisany dla oprogramowania Ansible. Przedstawia on konfigurację użytkowników oraz nowej grupy. W tym celu wykorzystano mechanizm pętli.

Listing 2: Przykładowy skrypt konfigurujący dla oprogramowania Puppet

```
group { 'test':
 ensure => present,
 gid => 3500,
}

file { '/home/test-user':
 ensure => directory,
 mode => '0750',
 owner => 'test-user',
}

file { '/home/student':
 ensure => directory,
 mode => '0750',
 owner => 'student',
}

user { 'test-user':
 ensure => present,
 uid => 3501,
 home => '/home/test-user',
 shell => '/bin/bash',
 groups => ['test'],
 password => pw_hash('test', 'SHA-512', 'password'),
}

user { 'student':
 ensure => present,
 uid => 3502,
 home => '/home/student',
 shell => '/bin/bash',
 groups => ['test', 'root'],
 password => pw_hash('teststudent', 'SHA-512', 'password'),
}
```

Listing 2 zawiera analogiczną konfigurację dla oprogramowania Puppet. Widoczna jest większa złożo-

ność tego skryptu względem skryptu napisanego dla Ansible.

Całość badań oparto na uprzednio przygotowanych skryptach, dzięki którym w łatwy sposób można było zarządzać konfiguracją systemu.

8. Wyniki

W tabeli 2 zawarto badane w wybranych systemach zarządzania konfiguracją systemu Linux aspekty, które oceniono w skali od 1 (trudne) do 5 (łatwe).

Tabela 2: Przedstawienie wyników łatwości konfiguracji z wykorzystaniem danego oprogramowania

	Ansible	Puppet
Instalacja oprogramowania	5	2
Zarządzanie plikami i folderami	4	4
Zarządzanie pakietami	4	4
Zarządzanie użytkownikami	5	3
Konfigurowanie serwisów	4	4

Wyniki te oparto o badania systemów, na których wykonano podobne czynności. Tylko wtedy można było w rzetelny sposób ocenić poszczególne oprogramowania. Mimo różnic można było poddać ocenie te elementy, które wykonano na obu systemach.

9. Analiza odczuć badaczy

Porównując oba systemy stwierdzono, że instalacja Ansible jest dużo łatwiejsza w stosunku do instalacji Puppet. Do prawidłowego działania Ansible wymagana jest instalacja samego pakietu, dopisania grup w pliku konfiguracyjnym oraz wymiany kluczy SSH. Puppet wymaga instalacji agenta, przez co czas samej instalacji jest znacznie wydłużony. Ponadto ze względu na mnogość elementów, które należy przygotować do poprawnego działania w przypadku Puppet istnieje możliwość uszkodzenia instalacji. Z tego powodu łatwość instalacji określono jako umiarkowanie trudną.

Zarządzanie plikami i folderami nie stanowiło problemu zarówno w przypadku wykorzystania Ansible jak i Puppet. Oba z tych systemów posiadają rozwiązania umożliwiające w prosty sposób obsługę plików i folderów. Dużym plusem dla Ansible była mnogość preinstalowanych modułów. W Puppet, aby wykonać bardziej zaawansowane rzeczy należy ręcznie zainstalować poszczególne rozszerzenia.

Przechodząc do zarządzania pakietami uznano, że zarówno w Ansible jak i Puppet ta funkcjonalność jest łatwa to przyswojenia. Ansible posiada pętlę, która wykona dane zadanie n-krotnie, zależnie od ilości podanych argumentów. W Puppet można utworzyć zmienną, w której przechowana będzie tablica z zawartością. Następnie wykonując dane zadanie opierające je na danej zmiennej wykona ona to zadanie n-krotnie.

Uznano, że tworzenie, edycja jak i usuwanie użytkowników było nieco łatwiejsze w Puppet niż w Ansible. Moduł związany z użytkownikami według autorów lepiej sprawdzał się w tworzeniu użytkowników właśnie w Ansible. Tworząc nowego użytkownika nie jest potrzebne uprzednie tworzenie katalogu domowego, czy ustawianie mu uprawnień. W Ansible wszystko dzieje się automatycznie, co w porównaniu do Puppet jest dużym plusem. W Puppet wszystkie powyższe kroki należy wykonać, aby można było utworzyć użytkownika.

Odnosząc się do zarządzania serwisami stwierdzono, że na bardzo podstawowym poziomie dosyć prosto można obsłużyć wszystkie serwisy. W obu przypadkach istnieją odpowiednie moduły, które wspomagają użytkownika w procesie zarządzania. Podczas badań podjęto się prostej konfiguracji zapory ogniowej w celu poprawnego działania serwera Apache. W obu przypadkach istnieje możliwość pracy z zaporą sieciową. W Ansible istnieje preinstalowany moduł, a w Puppet można taki zainstalować lub wykorzystać moduł *Exec*, który umożliwia wykonanie wprowadzonego do niego polecenia.

10. Wnioski

Oba z porównywanych narzędzi do zarządzania konfiguracją systemów Linux posiadają wiele funkcji wspomagających pracę administratora.

Zdecydowanie łatwiejszym oprogramowaniem do instalacji jest Ansible. Ansible w przeciwieństwie do Puppet-a nie wymaga agenta, a sam proces jego konfiguracji jest prostszy i klarowniejszy dla początkującego użytkownika. W Puppet istnieje wiele zawiłości, które mogą zniechęcić potencjalnego użytkownika.

Zarządzanie plikami i folderami nie stanowiło problemu zarówno przy używaniu Ansible jak i Puppet. Obydwa systemy były w tym aspekcie przyjazne użytkownikowi.

Zastosowanie klasowości znanej z języków programowania w Puppet i pętle zaimplementowane w oprogramowaniu Ansible pozwoliło na bezproblemową pracę z pakietami. Dzięki nim w łatwy sposób można instalować, aktualizować lub usuwać wiele pakietów. Dodatkowo taki zabieg wpływa na optymalizację skryptu, co oznacza, że wystarczy napisać zadanie instalacji jednokrotnie, a dane oprogramowanie powieli jego reprezentację w zależności od ilości elementów w pętli. Uznano, że w jednym i drugim systemie nie ma większych problemów z przeprowadzeniem instalacji jednego lub więcej pakietu.

W przypadku zarządzania użytkownikami według autorów to Ansible jest lepszym wyborem. W Ansible tworzenie użytkowników opierało się na jednym zadaniu. Moduł odpowiedzialny za zarządzanie użytkownikami w tym systemie jest na tyle rozbudowany, że tworzenie bądź usunięcie użytkownika nie sprawia większych trudności.

Ostatnim porównywanym aspektem było zarządzanie serwisami. Sam proces obsługi serwisu w obu systemach jest bardzo podobny. Zarówno w Puppet jak

i Ansible można uruchamiać, restartować, czy zatrzymywać serwisy. W obu przypadkach stwierdzono, że praca z demonami nie jest trudna, jednak wymaga dużego doświadczenia.

Wykorzystując poszczególne systemy można wyjść poza obręb podstawowych funkcji. Ustawienie zapory ogniowej jest jednym z bardziej zaawansowanych elementów i wymaga przemyślenia całego procesu. Oba systemy wspierały pracę z zaporą ogniową i umożliwiały jej konfigurację w dość przejrzysty sposób. W systemie Ansible jest od razu wbudowany moduł umożliwiający wykonywanie poleceń na zaporze. W Puppet istnieje opcjonalny moduł do pracy z Firewall-em.

Według badaczy zaletą Puppet jest możliwość wykonania danego zadania tylko w przypadku, gdy dany pakiet lub serwis istnieje. Oczywiście w Ansible też jest taka możliwość, jednak porównując te dwa systemy to w Puppet ta funkcjonalność jest łatwiejsza do opanowania przez niedoświadczonego użytkownika. W Ansible wymagane jest pisanie długich warunków, co nie jest w dłuższej perspektywie czasu wygodne.

Porównując strukturę pliku konfiguracyjnego Ansible oraz Puppet stwierdzono, że bardziej przyjazną formę mają pliku pisane w języku YAML. Według autorów są one bardziej przejrzyste oraz prostsze w pisaniu. W przypadku funkcjonalności lepsze okazały się pliki własne Puppet, ponieważ język ten został stworzony specjalnie dla tego narzędzia.

Podsumowując, zarówno jeden jak i drugi system w dużym stopniu wspomaga pracę administratora systemów. Dzięki zastosowanych w nich modułach upraszcza pracę i skraca czas potrzebny na wykonanie monotonnych czynności. Nie da się jednoznacznie określić, który z podanych systemów jest lepszy. Według badaczy składnia Ansible oraz proces instalacji są bardziej przyjazne użytkownikowi. Puppet bardziej nadaje się do pracy w dużych korporacjach niż w małych firmach. Oba systemy znajdą swoje grono odbiorców i wspomogą pracę wszystkich administratorów, którzy nie są zamknięci na technologiczne nowości.

Literatura

[1] S. Thakur, S. C. Gupta, N. Singh, S. Geddam, Mitigating and patching system vulnerabilities using ansible: A comparative study of various configuration management tools for iaas cloud. In Information Systems Design and Intelligent Applications, Springer New Delhi (2016) 21-29.

[2] F. Önnberg, Software Configuration Management: A comparison of Chef, CFEngine and Puppet, (2012) http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-6040.

[3] F. Locati, Learning Ansible 2 – Second Edition, Packt Publishing Ltd., 2016.

[4] J. Arundel, Puppet 5 Beginner's Guide – Third Edition. Packt Publishing Ltd., 2017.

[5] J. Grzabel , Puppet Vs Ansible, https://www.devopsgroup.com/blog/puppet-vs-ansible/, [13.07.2021].

[6] A. Johari, Chef vs Puppet vs Ansible vs Saltstack: Which Works Best For You?, https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/, [13.07.2021].

[7] Opis oraz pierwsze wydanie oprogramowania Ansible, https://en.wikipedia.org/wiki/Ansible_(software), [14.06.2021].

[8] Opis oraz pierwsze wydanie oprogramowania Puppet, https://en.wikipedia.org/wiki/Puppet_(software), [14.06.2021].

[9] E. Nemeth, G. Snyder, T. R. Hein, T. Adelstein, B. Lubanovic, T. Limoncelli, UNIX i Linux przewodnik administratora linux. Rozdział 23, Helion, 2018.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 18–25
Received: 15 October 2021
Accepted: 21 October 2021

Comparative analysis of PHP frameworks on the example of Laravel and Symfony

Analiza porównawcza szkieletów PHP na przykładzie Laravel i Symfony

Paulina Garbarz*, Małgorzata Plechawska-Wójcik

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

Abstract

This study aims to determine which of the analyzed PHP-based design patterns – Laravel or Symfony – is a more sufficient solution and which one of them is more complex from the code point of view. For this purpose, a comparative analysis was carried out based on the available documentation, as well as a comparison of the static and dynamic metrics obtained in the research environment of both tested patterns. As a result of a series of experiments and studies, it was established that both design patterns are an optimal and efficient solution, but their best application depends on the developer's individual needs and project requirements.

Keywords: PHP frameworks; Laravel; Symfony; comparative analysis

Streszczenie

Niniejsza praca ma na celu ustalenie, który z analizowanych wzorców projektowych opartych na języku PHP – Laravel czy Symfony – jest wydajniejszym rozwiązaniem, a także który z nich jest bardziej złożony z punktu widzenia kodu. W tym celu przeprowadzono analizę porównawczą opartą na dostępnej dokumentacj, a także porównaniu uzyskanych w środowisku badawczym metryk statycznych i dynamicznych obu badanych wzorców. Wynikiem serii eksperymentów i badań ustalono, że oba wzorce projektowe stanowią optymalne i wydajne rozwiązanie, jednak ich najkorzystniejsze zastosowanie jest zależne od indywidualnych potrzeb dewelopera oraz wymagań projektu.

Słowa kluczowe: szkielety PHP; Laravel; Symfony; analiza porównawcza

*Corresponding author

Email address: **garbarz.paulina@gmail.com** (P. Garbarz)

1. Introduction

Nowadays, working without a design pattern is associated with a long code development process and low efficiency, which is why more and more new solutions appear on the market, and the previously created ones are constantly developed. The most popular are the analyzed Laravel and Symfony - GitHub contains approximately 62,000 and approximately 24,000, respectively, projects based on these patterns (Figure 1), which are marked with stars to provide insight into them in the future.

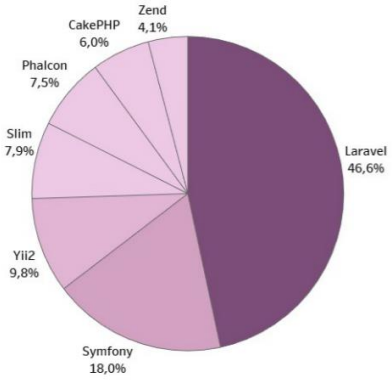


Figure 1: Popularity of particular PHP frameworks – own study based on projects popularity on GitHub.

Frameworks have a similar set of functionalities, but each of them individually has special features that distinguish them from the others. When choosing the right template, it should be taken into account not only the technical aspects and its possibilities, but also the subject of the project and the programmer's skills. Choosing a template can have a significant impact on the future and development of the entire project. That's why choosing the right one is so important.

The main idea behind Symfony is to create and develop long-term projects by default. It is a kind of skeleton for e-commerce platforms such as Magento, Drupal or PrestaShop, but also creates a base for numerous design patterns such as Laravel, Yii or CakePHP. The structure of Symfony is based on the MVC architecture, which provides modularity and complexity to the created projects due to the reusability of the same code in many different places. Its structure consists of a system of bundles, each of which fulfills a specific-task.

Symfony is also independent of the selected database environment – it allows the use of a PDO or ORM module to establish communication with the database, due to which it can get connection with many types of databases, such as MySQL, Oracle or SQLServer. The ORM technique dedicated to Symfony is Doctrine ORM, which allows to improve the performance of queries and improve the security of transmitted data. This framework uses its own template engine, which is

called Twig. It allows the creation of communication between the view and the controler throught the model, making it possible to modify or display relevant data for the user.

The second analyzed framework, Laravel, is based on Symfony components and uses this framework as a kind of skeleton, but it is distinguished by its original logic and the way the code is implemented in PHP. Laravel is characterized by fast application development, a high level of abstraction and a high intuitiveness when writing code. Due to the simplicity of creating a project, it is often assigned to small, quickly implemented projects, but the range of possibilities offered also allows for the creation of extensive solutions. The structure of Laravel, as in the case of Symfony, is based on the MVC architecture. Already at the time of creating a new project, the user receives a fully functional environment with the necessary dependencies and functionalities, which allows to reduce the time spent on appropriate adaptation and configuration of this environment.

Laravel enables to use Eloquent ORM by default and PDO module by optional. Eloquent ORM allows to increase the level of application security, and also streamlines the process of communication with different database environments. The way of processing queries is not only more effective, but also more accessible for the programmer creating the application, which affects the efficiency of the application development process.

Due to the usage of the Blade engine template, it is possible to encrease the performance of communication between the view and the controller. Blade templates also allow direct injection of PHP code into the view, due to which additional modifications are not required unlike Symfony.

The article presents the similarities and differences, strengths and weaknesses, as well as the specific features of the Symfony and Laravel frameworks. Its purpose is to determine which of the analyzed frameworks is a better and more efficient solution among the patterns based on the PHP language. In order to carry out the research, a suitable environment has been created, consisting of two applications that perform basic CRUD functions (Create, Read, Update, Delete) using both frameworks. In addition, the paper presents a comparative analysis between the tested patterns, the results of which includes static metrics (such as the number of lines of written code, the amount of used classes and interfaces, the number of used methods and libraries, the overall size of both developed applications, the ease of introducing changes to them and the overall assessment performance), as well as dynamic metrics (average service response time and server load) [1].

2. Literature Review

Each design pattern offers an extensive set of tools, such as APIs and libraries, the meaning of which is described in article [2]. For the study of overall performance, the authors created an environment implementing the same task, based on three different frameworks and three MySQL databases with an identical structure. Ultimately, it was found that each excelled in specific research areas, with Laravel being considered the most advantageous for file read and write operations and the least optimal for complex data.

The article [3] raises an important issue, which is the selection of an appropriate framework, based on the similarities and differences between them. The model created by the authors compares the performance of Laravel and Symfony on specific levels, comparing their available functionalities and evaluating their performance. Both were rated as effective solutions for PHP applications, with Symfony being a more stable pattern and Laravel as a pioneer in creating dynamic solutions.

Currently, Laravel is the most used PHP-based framework. In the article [4] entirely devoted to this pattern, the authors presented its possibilities by analyzing the functionality based on the creation of an E-Commerce website, as well as comparing it to other patterns. The study showed that Laravel is a pioneer in reading and writing data in files as well as in database migration between different areas. The author of the article [5] compares Laravel with the Slim Framework by measuring the load test performance for three different scenarios. The study shows that Slim Framework is a faster and better solution, but Laravel, due to its size and the availability of numerous libraries and solutions, is better for large projects than the competitor. It is also worth mentioning that Laravel also owes its size to the fact that it contains Symfony components, which constitute its specific skeleton [6].

Symfony is one of the oldest PHP frameworks. Like most PHP frameworks, is based on MVC and uses ORM [7]. Despite the long experience, it is still a willingly chosen model, which was noted in the article [8], which proved that Symfony quickly keeps up with the dynamic changes in standards. It is also popular due to the large community and extensive documentation. During its lifetime, some versions were supported for a longer time than others. In the study [9], the three most popular versions were analyzed, using a research environment based on three applications with different versions of Symfony, which fulfill the same task, as well as comparing the performance of individual versions. Finally, the authors determined that Symfony 4.2 deserved a special distinction due to its advantage, for example, when returning a large amount of data from API.

3. Research Method

For the purpose of the study, it was created a research environment that contains two applications with identical functionalities, but implemented in different design patterns – Laravel and Symfony in the latest stable versions (8.5.7 and 5.1.3 respectively). The applications fulfill the tasks of a simple blog that performs basic CRUD operations and also allows for user authentication. Both versions have been tested for performance in communication with database. For this purpose, two

schemas were created for each project – one in MySQL, one in PostgreSQL. Both applications run on a local server. Table 1 shows the parameters of the equipment on which the research environment was realized.

Table 1: The parameters of the equipment

Parameter	Value
Processor	AMD Ryzen 5 4500U
RAM	16GB
Disc	SSD M.2PCIe 512GB
Graphic Card	AMD Radeon Graphics 2.38GHz
Operating System	Windows 10 Home 20H2

The created environment allows for a comparative analysis of Laravel and Symfony in order to determine their similarities, as well as to highlight significant differences. To complete this task, the author used appropriate methods and metrics, divided into static and dynamic metrics.

To perform the benchmarking using static metrics, a PHP tool called PHPLOC was used. It allows to calculate the exact number of lines of source code, the classes and methods contained in the program, and the overall size of the application (including the number of files and folders).

Two extensions for selected frameworks were used to determine dynamic metrics – Laravel Dusk for Laravel and WebTestCase for Symfony, respectively. They allow to simulate user actions, and thus collect data on SRT, QET and TPT metrics [10]. To measure the execution time of a database query, a series of tests should be carried out that perform a specific number of queries to the database. For each operation supported by the application (create, read, update, delete), tests containing 10, 100 and 1000 queries were performed. Before starting each test, it is necessary to clear the cache to maintain the reliability of the results.

3.1. Static Metrics

The following static metriccs were used in the study:

- **Number of lines of code** – Known as LOC (Lines of Code) or SLOC (Source Lines of Code), this is a type of size metric that allows to identify the lines of source code used in a project. Shows the scale of the software, and indicates classes and methods that are beyond the recommended size.
- **Number of classes and interfaces** – this metric allows to specify the exact number of classes and interfaces used in the software source code, it excludes internal classes.
- **Number of methods** – the metric responsible for indicating the number of methods existing in the application. It applies not only to the entire program, but also to individual classes or interfaces.
- **Program size** – a metric responsible for measuring the number of files and folders present in the application, as well as their size on the disk. Its size may be affected by the number of libraries attached to the project, the number of lines of code or the number of files and folders. This value was expressed in KB.

3.2. Dynamic Metrics

In addition to static metrics, the following dynamic metrics were also included in the study:

- **Total Processing Time (TPT)** – this metric is used to measure the time elapsed from sending the request until the application responds. It provides information on the total processing time of the user's command, from its creation, through processing by the server and database, to obtaining a reply for each CRUD operation. For each operation, the average processing time in milliseconds is calculated.
- **Service Response Time (SRT)** – is used to measure the response time of the application server, starting from sending a request to the network service until obtaining the first byte of the response. Similar to TPT, each CRUD operation is processed by the specified number of query samples, from which the average value for each operation is determined. The metric is expressed in milliseconds.
- **Query Execution Time (QET)** – a metric that measures the time that was needed to process the query sent by the application (in this case, the CRUD operation in the amount of samples specified above).

4. Comparative Analysis

Both frameworks – Symfony and Laravel – were analyzed in terms of static and dynamic metrics. This chapter summarizes their possibilities and the performed analysis.

4.1. Comparative Analysis Based on Static Metrics

The comparison of static metrics of projects implemented in Laravel and Symfony enables the comparison of the quality of the source code of the application being developed. The research environment provides information on the number of lines of code, number of classes and interfaces, number of methods, program size, scalability and overall performance score.

Figure 2 summarizes information on the total number of lines of code (LOC) for both projects that are part of the research environment.

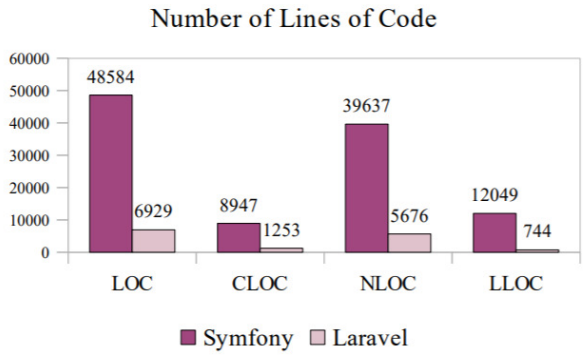


Figure 2: The number of lines of code in Symfony.

It also provides information about the number of comment and non-comment lines of code (CLOC and NLOC metrics respectively), and the number of logical lines of code (LLOC metric) consisting of classes and methods. The numerical advantage of LOC is clearly visible for the benefit of Symfony – the Symfony's LOC is almost 7 times higher than Laravel's. It is also worth noting that Symfony has a greater percentage of CLOC and LLOC (18.42% and 24.80% respectively) than Laravel (12.08% and 10.74% respectively). A higher number of comment lines of code may suggest the presence of more extensive documentation and guidance on the use of Symfony capabilities, while a higher number of lines of code placed in the application logic may result in slower performance as the project grows.

Figure 3 shows a graph showing the number of classes and interfaces available in both frameworks. Symfony, as with the number of lines of code, can boast a considerable number of classes in the project, which exceeds the number of classes in Laravel almost 14 times. Interestingly, none of the patterns offer any interface by default.

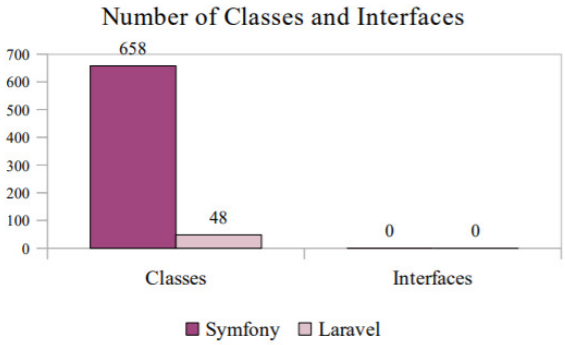


Figure 3: The number of classes and interfaces in Symfony and Laravel.

The above list of the number of classes shows the enormity and complexity of Symfony, but it is worth remembering that it is a fully proprietary framework, while Laravel complements the logic of Symfony with its logic, treating it as its base.

Figure 4 shows the quantification of the different types of method visibility in Symfony.

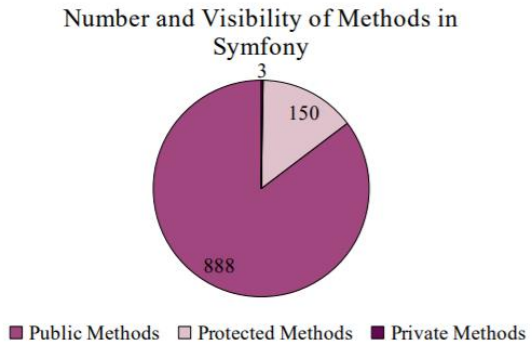


Figure 4: The number and the visibility of methods in Symfony.

It is clearly visible that this framework offers mostly public methods (888), but 15% of all methods (153 exactly) are protected or private. The sheer number of available methods in this framework is considerable, because the project implemented in Symfony in the research environment has 1041 of them. The audience of Symfony methods allows to conclude that the framework by default provides the standard high security of its components by restricting access to them.

The next graph (Figure 5) shows the number and visibility of the methods in the application implemented in Laravel. Their number, as in the case of LOC and the number of classes and interfaces, is several times lower than in the case of Symfony. It is worth noting that most of the methods available in the Laravel (77) application are public, while offering only 4 protected methods and no private methods.

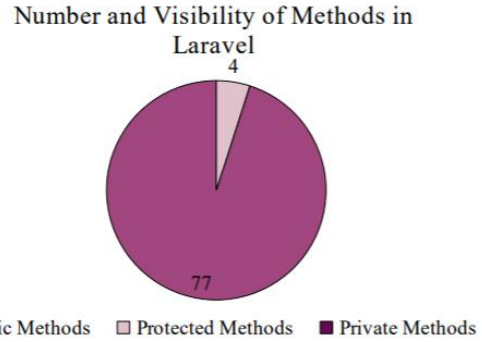


Figure 5: The number and the visibility of methods in Laravel.

Collected static metrics clearly show that Symfony creates larger-sized projects with a much higher number of components than Laravel. This is due to the construction of the entire framework, because Symfony takes full advantage of its proprietary solutions, while Larave relies on Symfony components, complementing them with its own logic.

The volume of the project can have a significant impact on the performance and smoothness of the application to the benefit of Laravel, but Symfony, due to the greater number of protected and private methods, can provide greater security of the designed solutions.

4.2. Comparative Analysis Based on Dynamic Metrics

Both Laravel and Symfony frameworks have been subjected tu dynamic metric analysis summarized in this chapter. It used data collected for 1000 samples, as it turned out to be the most reliable in the context of the entire study. The following therminology is used in the graphs: L for Laravel and S for Symfony.

At the beginning, the cooperation of both patterns with the PostgreSQL database was analyzed, which is visible at the Figure 6 describing the SRT for 1000 post.

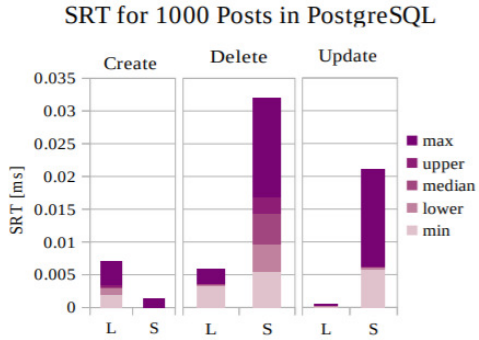


Figure 6: SRT metric for 1000 post sample in PostgreSQL.

It can be seen that when a new post was placed, Symfony obtained better results, but in other cases (delete and update) its result is much worse than Laravel in terms of efficiency – these operations take much longer

The situation is different for the SRT with a sample of 1000 comments presented at Figure 7.

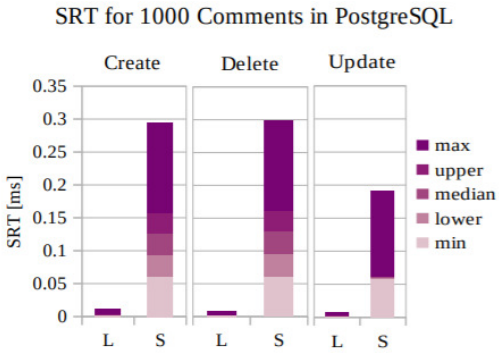


Figure 7: SRT metric for 1000 comments sample in PostgreSQL.

Laravel did better for each operation performed (the waiting time for a response is close to almost 0), while Symfony took definitely more time to deliver the first byte of the response, sometimes up to 0.3ms.

The time of querying the database at the time of creating a new post (Figure 8) was definitely in favor of Symfony.

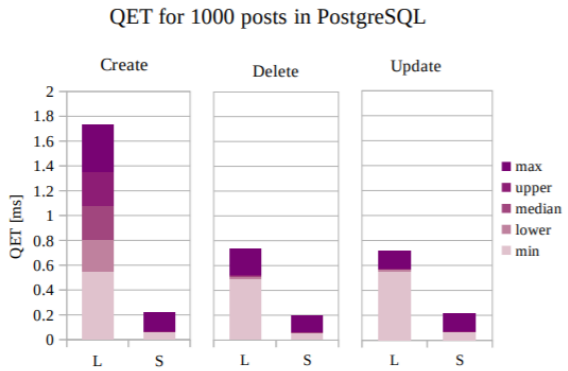


Figure 8: QET metric for 1000 posts sample in PostgreSQL.

With each operation, it is visibly lower than in the case of Laravel (almost 9 times for the create operation itself). The operation of creating a post is the most absorbing among those listed for Laravel, while the results for Symfony are very similar and remain at the level of 0.2 ms.

The situation is similar in the case of posting a comment on the blog (Figure 9). The query processing time by Symfony is much lower than in the case of Laravel, despite the fact that both patterns maintain the values or all the above-mentioned operations at a similar level.

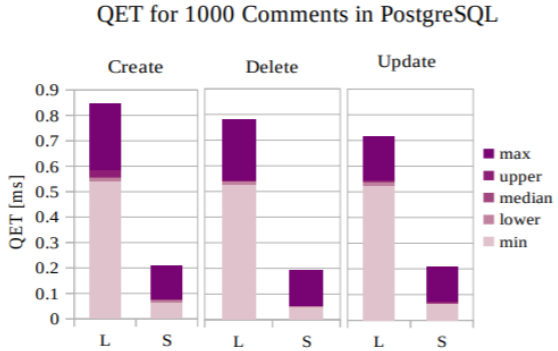


Figure 9: QET metric for 1000 comments sample in PostgreSQL.

It is worth noting, however, that the time for adding a comment by Laravel is significantly lower than the time for adding a post - this is due to the fact that less data lands in the database (a comment is much shorter than a blog entry). Symfony is in the area of 0.2ms in both cases.

The difference in query processing time may be due to the fact that both frameworks, Laravel and Symfony, use separate object-relational mapping – Eloquent ORM and Doctrine ORM respectively. The main difference between them is that Doctrine is entirely based on pure old PHP language, while Eloquent inherits all the ORM persistence logic.

The TPT metric (Figure 10) for the sample of 1000 posts in the case of Laravel was the worst for the Create operation - it clearly surpasses the other values on the chart.

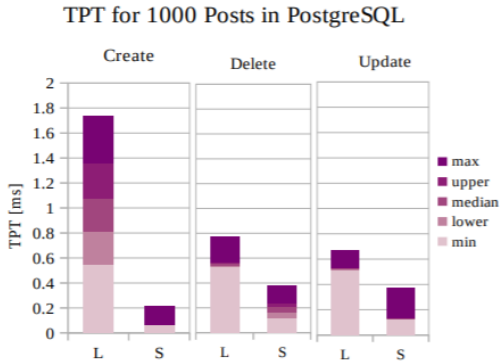


Figure 10: TPT metric for 1000 posts sample in PostgreSQL.

This is due to the fact that the query processing time was much higher than in other operations. A similar tendency was shown by Symfony - a significant part of TPT was spent on query processing, but the result is still more optimal (oscillates between 0.2 – 0.4ms).

In the case of TPT for a sample of 1000 comments (Figure 11), the situation for both patterns is relatively similar - both frameworks show similar values for each operation, but it is worth noting that in the case of Laravel this value is 2 times lower than in the case

of the sample of 1000 posts. There was an exemplary trend in Symfony - each operation requires almost 0.4ms of TPT. This is because Symfony has a higher SRT which automatically increases TPT.

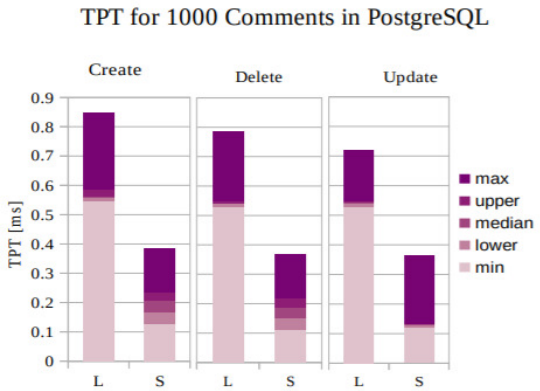


Figure 11: TPT metric for 1000 comments sample in PostgreSQL.

The Read operation for each framework and sample is shown in a separate graph (Figure 12). It clearly shows that the situation is very similar in both cases - both Laravel and Symfony show low SRT, and most TPT is sending and processing a query to the database and getting a response from it. However, it is worth paying attention to the time in which the above-mentioned operation is performed - for Laravel this time is 1.8ms for posts and 1.6ms for comments, while for Symfony - 0.2ms.

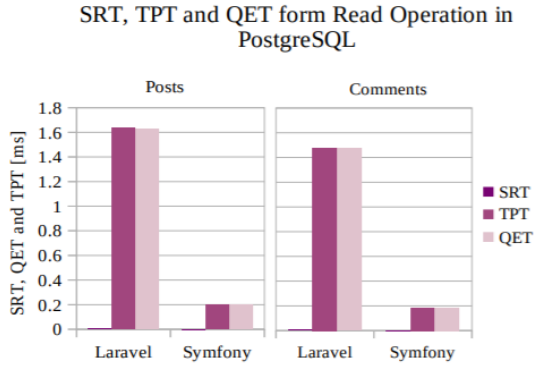


Figure 12: Dynamic metrics for Read operation in PostgreSQL.

The next stage of the analysis was the compilation of dynamic engines for the MySQL database. Figure 13 shows the SRT metric related to a sample of 1000 posts. The most overwhelming operation in this case was the Delete operation - for both Laravel and Symfony.

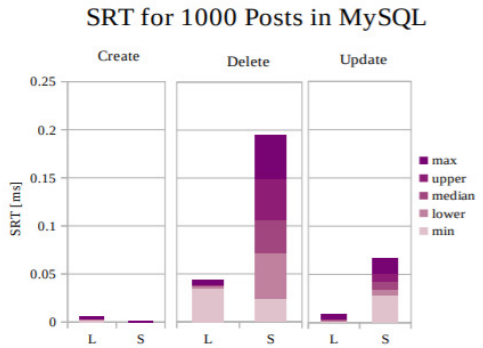


Figure 13: SRT metric for 1000 posts sample in MySQL.

It is worth mentioning that Symfony has a much higher SRT for Update and Delete than Laravel. This may be due to the fact that the size of the post, due to the number of characters, is an aggravating query for the database.

The situation is different for the SRT metric for a sample of 1000 comments (Figure 14).

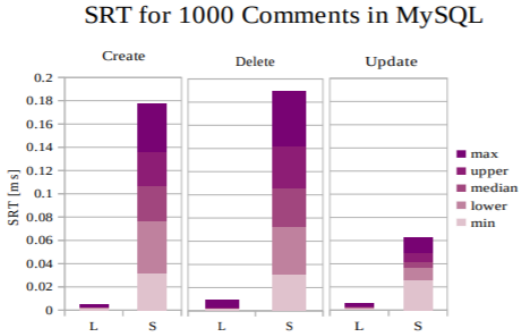


Figure 14: SRT metric for 1000 comments sample in MySQL.

While in the case of creating a post, SRT for both patterns was relatively low and high for the remaining operations, in this case the SRT is clearly (even 10 times) higher for Symfony than Laravel for each performed operation. The graph also clearly shows that the SRT for Laravel comments remains at a similar level of around 0.01ms.

In the case of the query processing time in the MySQL database for a sample of 1000 posts (Figure 15), Laravel did worse in the create and delete operation than Symfony - its execution time was significantly longer. Symfony, however, required much more time to update an existing post (almost 2ms), which is not only a few times higher result in the stocunt to Laravel, but also in relation to the create and delete operation in Symfony itself.

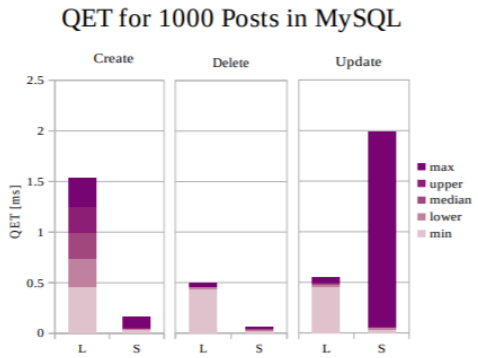


Figure 15: QET metric for 1000 posts sample in MySQL.

The situation looks different for QET metric for comments (Figure 16). In this case, all operations temporarily disadvantage Laravel - the query processing time, from sending it to obtaining a response from the database, is much higher than in the case of Symfony. Contrary to the sample of 10,000 posts, the delete operation was the most absorbing - probably because the comment is additionally burdened with a foreign key.

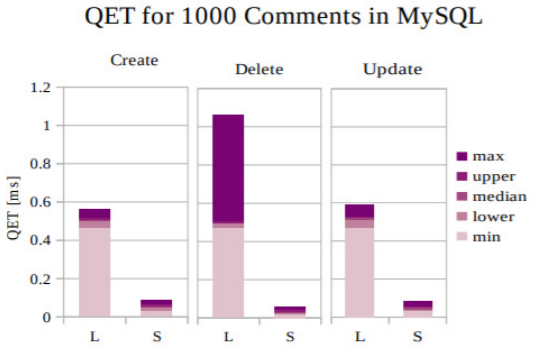


Figure 16: QET metric for 1000 comments sample in MySQL.

TPT for MySQL posts shown at Figure 17 - unlike PostgreSQL - does not unequivocally speak in favor of any of the analyzed patterns. It is true that the operation of adding a new post and removing it takes Laravel more time, but its update is several times lower than in the case of Symfony. This is due to the high QET for a Symfony post update - all this metric is the vast majority of the total processing time in each case.

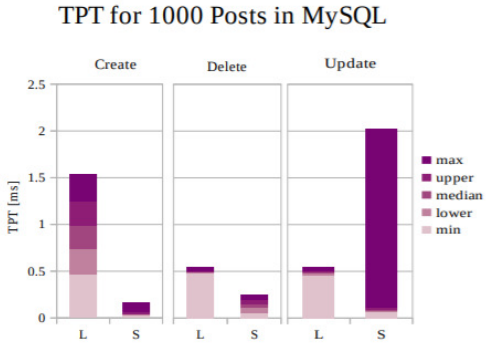


Figure 17: TPT metric for 1000 posts sample in MySQL.

The TPT for the sample of 1000 comments shown in Figure 18 clearly shows that in this case Symfony is again keeping the result at 0.2ms. The thing that stands out is the processing time for the delete operation in Laravel - it is significantly longer than for the other operations and the second pattern. This fact is due to the low SRT and high QET.

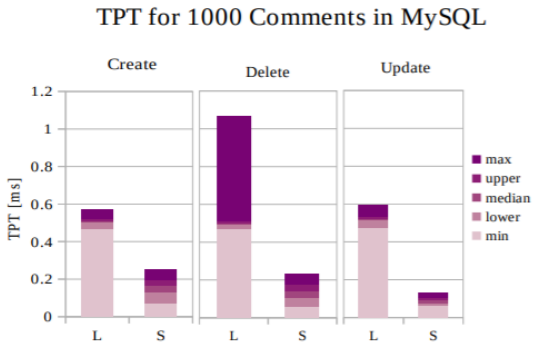


Figure 18: TPT metric for 1000 comments sample in MySQL.

Figure 19 shows the mean values of SRT, QET, and TPT for 1000 comments and 1000 posts read operations. As in the case of other operations, the vast majority of TPT is occupied by QET (approx. 3 ms for Laravel and approx. 0.5 ms for Symfony, respectively). SRT is almost 0 in both cases.

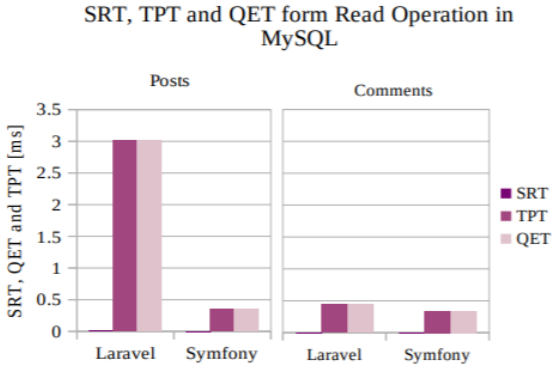


Figure 19: Dynamic metrics for Read operation in MySQL.

The collected values of static and dynamic metrics made it possible to compare the possibilities and quality of the code and the functioning of both analyzed design patterns itself. Dynamic metrics have been compiled in tables to compile the final results.

Table 2 presents dynamic metrics for a sample of 1000 PostgreSQL posts. It is clearly visible the performance of Symfony showing better communication with the database as well as better SRT for half of the CRUD operations.

Table 2: Dynamic metrics for 1000 posts sample in PostgreSQL.

	SRT	QET	TPT
Create	S	S	S
Read	S	S	S
Update	L	S	S
Delete	L	S	S

Table 3 shows a similar trend with an advantage for Symfony - Laravel is a clear favorite for the SRT metric, but Symfony performs better for the rest of the cases.

Table 3: Dynamic metrics for 1000 comments sample in PostgreSQL.

	SRT	QET	TPT
Create	L	S	S
Read	S	S	S
Update	L	S	S
Delete	L	S	S

Table 4 summarizes the dynamic capabilities of both frameworks for a sample of 1000 MySQL posts. In this case, Laravel did better not only for the SRT, but also for each metric for the Delete operation.

Table 4: Dynamic metrics for 1000 posts sample in MySQL.

	SRT	QET	TPT
Create	S	S	S
Read	S	S	S
Update	L	S	S
Delete	L	L	L

The last table (Table 5) shows the dynamic metrics for a sample of 1000 MySQL comments. The situation is the same as for the sample of 1000 PostgreSQL

comments - Laravel has better performance for the SRT metric, except for the Read operation.

Table 5: Dynamic metrics for 1000 comments sample in MySQL.

	SRT	QET	TPT
Create	L	S	S
Read	S	S	S
Update	L	S	S
Delete	L	S	S

While analyzing the above results, it is worth bearing in mind that the main part of the TPT metric is the database query processing time (QET).

5. Conclusion

The aim of the study was to determine which of the analyzed frameworks - Laravel or Symfony - is a better solution for the developer. Based on the collected results, it can be concluded that Laravel as a framework is definitely a less complex solution than Symfony. Its construction based on static metrics shows a lower complexity and volume in relation to the second analyzed model (the values of the metrics sometimes show a several times higher result in relation to the Symfony - Laravel ratio). However, it is worth remembering that Symfony uses only its own proprietary solutions, while Laravel uses Symfony components as a specific base and its backbone, which has a decisive impact on the volume of the code.

Performance was tested based on dynamic metrics. Due to this operation it was possible to test the performance of both frameworks. The analysis revealed that Laravel as a framework shows better results for the SRT metric - this means that compared to Symfony it delivers the first byte of response faster, regardless of the operation performed, but Symfony seems to be a pattern better connected with databases. The query processing time for most operations with samples of 1000 posts and 1000 comments in Symfony most often oscillated around 0.2ms, while in Laravel it was able to achieve the result of less than 2ms.

All CRUD operations showed a similar relationship - in each case, the greater part of TPT took up the processing time of the database query. For both frameworks, this operation was the most absorbing and took a long time to carry out from start to finish.

The above results allow to conclude that both analyzed frameworks are an efficient and optimal solution for a developer, but each of them has its own advantages and disadvantages. Laravel is a better solution for the processing of application content and due to the volume of the code, it allows for easier organization and implementation of changes, but Symfony is better connected with PostgreSQL and MySQL databases. The choice of the best framework should therefore depend on the individual needs of the developer, project assumptions, as well as the requirements set by the client. More extensive research is planned to explore this topic.

References

[1] A. Gdula, M. Plechawska-Wójcik, Porównanie wydajności wybranych technologii tworzenia usług sieciowych w aspekcie zastosowań w aplikacjach internetowych, Journal of Computer Sciences Institute 1 (2016) 14-19, https://doi.org/10.35784/jcsi.61.

[2] M. Jailia, A. Kumar, M. Agarwal, I. Sinha, Behavior of MVC (Model View Controller) based Web Application developed in PHP and .NET framework, International Conference on ICT in Business Industry & Government (ICTBIG) (2016) 1-5, https://doi.org/10.1109/ictbig.2016.7892651.

[3] U. Sa'adah, J. Akhmad, M. Hisyam, Implementing Singleton method in design of MVC-based PHP framework, International Electronics Symposium (IES) (2015) 212-217, https://doi.org/10.1109/elecsym.2015.7380843.

[4] J. Adamu, R. Hamzah, M. M. Rosli, Security issues and framework of electronic medical record: A review, Bulletin of Electrical Engineering and Informatics 9(2) (2020) 565-572, https://doi.org/10.11591/eei.v9i2.2064.

[5] R. F. Olanrewaju, T. Islam, N. A. Ali, An empirical study of the evolution of PHP MVC framework, Advanced Computer and Communication Engineering Technology (2015) 399-410, https://doi.org/10.1007/978-3-319-07674-4_40.

[6] X. Li, S. Karnan, J. A. Chishti, An empirical study of three PHP frameworks, 4th International Conference on Systems and Informatics (ICSAI) (2017) 1636-1640, https://doi.org/10.1109/icsai.2017.8248546.

[7] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi, A. El Yamami, A comparative study of Laravel and Symfony PHP frameworks, International Journal of Electrical and Computer Engineering 9(1) (2019) 704-712, https://doi.org/10.11591/ijece.v9i1.pp704-712.

[8] S. Singh, J. Iyer, Comparative Study of MVC (Model View Controller) Architecture with respect to Struts Framework and PHP, International Journal of Computer Science Engineering 5(3) (2016) 142-150.

[9] X. K. Liu, G. G. Cheng, Analysis and Implementation of ASP. Net and PHP Frameworks Based on MVC Architecture, In Advanced Materials Research 798 (2013) 749-752, https://doi.org/10.4028/www.scientific.net/amr.798-799.749.

[10] M. R. Daraż, P. Kopniak, Analiza możliwości współpracy aplikacji mobilnych z usługami sieciowymi typu REST i Web Service, Journal of Computer Sciences Institute 11 (2019) 155-162, https://doi.org/10.35784/jcsi.181.

JOURNAL COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 26–34
Received: 18 October 2021
Accepted: 7 November 2021

Comparison of frameworks for developing web applications in PHP

Porównanie szkieletów do wytwarzania aplikacji internetowych dla języka PHP

Kamil Pawelec*, Piotr Kopniak

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

Abstract

The article presents a comparative analysis of two frameworks of PHP application development such as Laravel and Symfony. Test results obtained using the implemented implementation problem compare the application code size, code execution time, hardware resources used, as well as the complexity of the code. The purpose of the research was to present the possibilities of given application development skeletons, with the help of which the web application programmer will be able to decide on its use depending on the implemented solution.

Keywords: PHP; Laravel; Symfony

Streszczenie

Artykuł przedstawia analizę porównawczą dwóch szkieletów budowy aplikacji w języku PHP jakimi są Laravel oraz Symfony. Wyniki badań uzyskane za pomocą analizy dokumentacji oraz zaimplementowanych projektów analogicznie dla dwóch szkieletów porównują funkcjonalność, złożoność kodu aplikacji, czas wykonania kodu, oferowane mechanizmy bezpieczeństwa, obsługi baz danych oraz gniazd sieciowych typu WebSocket. Celem badań była prezentacja możliwości danych szkieletów budowy aplikacji, z pomocą której programista aplikacji internetowych w zależności od implementowanego rozwiązania będzie mógł zdecydować o ich użyciu.

Słowa kluczowe: PHP; Laravel; Symfony

*Corresponding author

Email address: **kamil.m.pawelec@gmail.com** (K. M. Pawelec)

1. Wstęp

Język PHP ciągle jest jednym z najpopularniejszych języków wykorzystywanych do implementacji aplikacji internetowych. Wiele istniejących już systemów i aplikacji internetowych wykorzystuje właśnie ten język, co wiąże się z potrzebą jego ciągłego rozwoju. Według badań przeprowadzonych przez Stackoverflow na 90 tys. programistów 26.4% z nich używa języka PHP, co plasuje go na ósmym miejscu w rankingu najczęściej wykorzystywanych języków programowania [1]. W niniejszej pracy porównano możliwości dwóch z najbardziej popularnych szkieletów do budowy aplikacji dla języka PHP tj. Laravel i Symfony. Przeprowadzenie badań mających na celu wskazanie istotnych różnic pomiędzy nimi może ułatwić podjęcie decyzji o wyborze odpowiedniego narzędzia przed rozpoczęciem projektu. W artykule opisano przeprowadzone badania oraz uzyskane w ich rezultacie wyniki. Pierwszą część artykułu stanowi przegląd publikacji dotyczących tego tematu. Następnie przedstawiono proponowaną metodologię badań wraz z opisem przeprowadzonych badań i uzyskanymi wynikami. W ostatniej części artykułu przedstawiono wnioski z przeprowadzonych badań.

2. Przegląd publikacji dotyczących porównania szkieletów budowy aplikacji

Praca zbiorowa [2] opisuje możliwe sposoby podejścia do tematu porównywania szkieletów budowy aplikacji dla języka PHP. Autorzy przeprowadzili badania przy wykorzystaniu Laravel oraz Symfony i zestawili je za pomocą następujących kryteriów: organizacja kodu, architektura techniczna, wymagania systemowe, wielojęzykowość. Kryteria zastosowane w niniejszej pracy mogą być zatem niejako rozszerzeniem badań przeprowadzonych przez autorów.

Sharma P. K [3] oraz Sol T. [4] przedstawili wyniki badań wskazujące na różnice pomiędzy najczęściej wykorzystywanymi wersjami języka PHP. W przypadku Sol badania przeprowadzono uruchamiając dwie witryny o takiej samej konfiguracji i działające przy wykorzystaniu takiej samej pojemności pamięci w środowisku, gdzie zainstalowany został PHP w wersji 5.6 i 7. Witryny uruchomiono przy pomocy systemu zarządzania treścią WordPress. Przeprowadzone badania wskazują na znacznie zwiększoną wydajność wersji 7 w porównaniu do 5, w związku z czym projekt badawczy wykorzystuje wersję 7.

Autor Skaraczyński w swojej publikacji o szkieletach budowy aplikacji w PHP [5] opisuje wykorzystanie Symfony w praktyce krok po kroku. Publikacja jest instrukcją utworzenia projektu aplikacji. Instrukcje zawarte w publikacji sugerują, że Symfony jest szkieletem przeznaczonym do większych projektów.

Autor Gołda w książce [6] szczegółowo opisuje wykorzystanie Laravel krok po kroku. Prezentuje schemat instalacji szkieletu, zasadę działania kontrolerów oraz routingu, a także widoków. Opisuje również efektywne wykorzystywanie mechanizmów bezpieczeństwa

i sesji, a także m.in. Tworzenie i wykorzystywanie pakietów Composer.

Informacje pochodzące z dokumentacji Laravel [7] oraz Symfony [8] były pomocne w organizacji środowiska badawczego, ze względu m.in. na opis wymagań oraz instrukcje instalacji. Zawierały one również niezbędny opis sposobu wykorzystania poszczególnych funkcjonalności szkieletów budowy aplikacji.

Zaprezentowany przegląd literatury wykazał, że prezentowane dotychczas porównania nie opierają się na implementacji tego samego projektu w różnych szkieletach, co stało się podstawą określenia kierunku badawczego i celu niniejszej pracy. Przegląd literatury pozwolił również na dobór metodyki badawczej, w tym, w określeniu kryteriów porównawczych wykorzystanych w pracy, a także wskazał na jakich aspektach w przeprowadzanych badaniach należy się skupić.

3. Metodyka badawcza

Niniejsza praca prezentuje wyniki badań porównawczych szkieletów budowy aplikacji Symfony oraz Laravel, a dokładnie porównania: mechanizmów bezpieczeństwa, obsługi baz danych i możliwości wykorzystania poszczególnych szkieletów na podstawie opisu funkcjonalności oraz implementacji gniazd sieciowych typu WebSocket. Ponadto przeprowadzone zostały badania wydajności aplikacji zaimplementowanej z wykorzystaniem poszczególnych szkieletów oraz złożoność kodu powstałego w wyniku tych implementacji. Podstawą przeprowadzonych badań była aplikacja internetowa pozwalająca na rejestrację użytkowników i zapisywanie ich na utworzone wydarzenia. Aplikacja umożliwiała komunikację z innymi użytkownikami za pośrednictwem czatu. W przyszłości aplikacja zostanie rozwinięta o możliwość tworzenia/edycji wydarzeń, administrację zgłoszeniami, raporty statystyczne.

Do przeprowadzenia badań wykorzystano maszynę z dwurdzeniowym procesorem Intel 2.20GHz x2, 6GB RAM i systemem operacyjnym Windows 10. Wykorzystane oprogramowanie to: Google Chrome 85, PhpStorm 2019.3, pakiet XAMPP 3.2.4 (PHP 7.2.33, MySQL 127.0.0.1). Na strukturę bazy danych MySQL złożyły się tabele: users, events, apps, pozostające w relacjach: users : apps - jeden do wielu, events : apps – jeden do wielu.

4. Porównanie szkieletów budowy aplikacji

4.1. Funkcjonalność

Laravel jest najczęściej wybieranym szkieletem do realizacji niewielkich projektów, które nie wymagają zbyt dużych nakładów finansowych. Wykorzystuje on komponenty z różnych środowisk, które możliwe są do użycia za pośrednictwem narzędzia Composer. Composer dostarcza i standaryzuje format zarządzania zależnościami i bibliotekami i jest wymagany do instalacji samego szkieletu. Funkcje, które oferuje Laravel to przede wszystkim [9]: modułowość – wspomniany Composer pozwala na integrację również z zewnętrznymi bibliotekami, routing - definiowanie ścieżek użytkownika w aplikacji internetowej, łatwe zarządzanie konfiguracją - aplikacja zaprojektowana w Laravel będzie działać w różnych środowiskach - np. zmiana bazy danych (konfiguracja pliku .env), Query Builder i Eloquent ORM - zarządzanie bazą danych za pomocą modeli i łatwe wykonywanie zapytań, silnik szablonów Blade - widoki, gotowe szablony stron, gotowe mechanizmy uwierzytelniania, metody magiczne, relacyjno-obiektowy model bazy danych.

Symfony przeznaczony jest głównie do projektów złożonych i długoterminowych - jest dość skomplikowany i wymaga większych nakładów finansowych. Umożliwia łatwą rozbudowę projektu oraz integrację z innymi bibliotekami. Główne funkcje Symfony to: silnik szablonów Twig, doctrine ORM - zawiera kilka bibliotek PHP ułatwiających przechowywanie baz danych i mapowanie obiektów, wysoka wydajność, kompatybilność i rozszerzalność, elastyczny routing, zarządzanie sesjami, standaryzacja i interoperacyjność aplikacji.

4.2. Bezpieczeństwo

Laravel posiada mechanizmy zwiększające bezpieczeństwo aplikacji internetowej. Posiada ochronę przeciwko atakom SQL injection, o ile w projekcie wykorzystywany jest Query Builder lub Eloquent. Możliwa jest ochrona plików cookie poprzez wykorzystanie wygenerowanego klucza aplikacji. Jeśli klucz aplikacji nie zostanie ustawiony, sesje użytkownika i inne zaszyfrowane dane nie będą bezpieczne. Klucz aplikacji zwiększa jej bezpieczeństwo o ile pozostaje tajny. Możliwa jest tutaj również ochrona CSRF (ang. Cross Site Request Forgery). Mechanizm ochrony przed opisanym atakiem w Laravel polega na generowaniu tokena CSRF dla każdej sesji zalogowanego użytkownika aplikacji. Token służy sprawdzeniu czy uwierzytelniony użytkownik faktycznie wysyła żądania do aplikacji. W celu wykorzystania zabezpieczenia w danym formularzu HTML implementowanej aplikacji powinna znaleźć się dyrektywa @csrf, a cały proces weryfikacji czy dane wejściowe są zgodne z tokenem przechowywanym w sesji odbędzie się automatycznie. W Laravel możliwa jest również ochrona przed atakami typu Cross-site scripting (XSS). Zastosowanie konstrukcji *{{ }}* w kodzie uniemożliwia wykonanie skryptu osadzonego w treści atakowanej strony.

Autoryzacja w Laravel możliwa jest poprzez wykorzystanie middleware'ów, które odpowiadają za filtrowanie żądań HTTP wprowadzanych do aplikacji. Przykładem może być wpisanie w pasku adresu przeglądarki ścieżki do podstrony, która powinna być dostępna tylko dla uwierzytelnionych (zalogowanych) użytkowników, a w przypadku próby otwarcia takiej podstrony nastąpi np. przekierowanie do strony logowania. Wykorzystanie zdefiniowanego za pomocą klasy Authenticate middleware'a odbywa się w sposób widoczny na Listingu 1.

Listing 1: Klasa Authenticate definiująca middleware'a (plik Authenticate.php)

```
class Authenticate extends Middleware {
  protected function redirectTo($request) {
    if (! $request->expectsJson()) { return route('login'); }}}
```

Funkcja redirectTo przyjmuje argument, który jest ścieżką docelową wybraną przez użytkownika. Dla tej ścieżki wymagane jest logowanie użytkownika. Przekierowanie do strony logowania następuje poprzez metodę route('login'). Po zalogowaniu następuje automatyczne przekierowanie do ścieżki docelowej ($request).

Listing 2: Wykorzystanie middleware'a (plik ChatsController.php)

```
  public function __construct() { $this->middleware('auth'); }
```

Funkcja widoczna na Listingu 2 jest konstruktorem kontrolera. Alias auth definiowany jest w tablicy $routeMiddleware zawartej w klasie Kernel. Automatyczne wygenerowanie systemu uwierzytelniania przede wszystkim zwiększa bezpieczeństwo aplikacji, ze względu na wdrożone już mechanizmy bezpieczeństwa, a także poprzez uniknięcie przypadkowych prostych błędów zabezpieczeń.

Symfony posiada bardzo rozbudowany system zabezpieczeń [10], a jego wykorzystanie odbywa się poprzez specjalnie przygotowany pakiet zabezpieczeń pozwalający na kontrolę dostępu, zarządzanie sesjami i zaporami, czy uwierzytelnianie. Wykorzystanie pakietu możliwe jest po jego uruchomieniu poleceniem:

```
composer require symfony/security-bundle
```

W Symfony plikiem konfiguracyjnym dla uwierzytelniania jest plik security.yaml, który zawiera jednocześnie większość ustawień dotyczących bezpieczeństwa. Definicja uwierzytelniania tworzona jest przy pomocy klucza access_control, w którym podawana jest ścieżka do podstrony, dla której ustalane są reguły dostępu. Możliwe jest blokowanie dostępu po adresie IP, porcie, nazwie hosta, metody łączenia lub uprawnień użytkownika. Przykładowe wykorzystanie reguł dostępu do podstron w projekcie wygląda jak na Listingu 3.

Listing 3: Przykładowe wykorzystanie access_control (plik security.yaml)

```
  access_control:
#     - { path: ^/admin, roles: ROLE_ADMIN }
#     - { path: ^/profile, roles: ROLE_USER }
      - { path: ^/login, roles:
IS_AUTHENTICATED_ANONYMOUSLY }
      - { path: ^/random, roles:
IS_AUTHENTICATED_ANONYMOUSLY }
      - { path: ^/*, roles: ROLE_USER }
      - { path: ^/admin, roles: ROLE_ADMIN }
```

Zabezpieczenie przed atakami typu SQL injections wprzypadku Symfony odbywa się podobnie jak w Laravel po stronie mechanizmu obsługującego bazę danych - w tym przypadku biblioteka Doctrine zawiera wszystkie niezbędne zabezpieczenia przed tego typu atakami. W przypadku ataków typu CSRF sytuacja wygląda identycznie jak w przypadku Laravel. Podczas wysyłania danych z formularzy generowany jest i sprawdzany odpowiedni token CSRF (listing 4).

Listing 4: Zastosowanie tokenu CSRF (plik login.html.twig)

```
<input type="hidden" name="_csrf_token"
 value="{{ csrf_token('authenticate') }}" >
```

Ochrona przed atakami typu XSS również odbywa się analogicznie. Wsparciem w sanityzacji danych wejściowych dostarczanych przez użytkowników jest zastosowanie znaków *{{ }}*, które uniemożliwiają wykonanie skryptu osadzonego pomiędzy znakami. Wsparciem w implementacji wszelkich zabezpieczeń związanych z formularzami może być odpowiedni mechanizm generujący formularze, który pozwala na prostą implementację formularzy wykorzystywanych przez użytkowników i udostępnia szereg metod wspierających.

Symfony dostarcza obiekt konstruktora formularzy, który umożliwia opisywanie pól formularza przy użyciu płynnego interfejsu. Kreator ten w następnym kroku tworzy rzeczywisty obiekt formularza używany do renderowania i przetwarzania treści. Utworzenie formularza możliwe jest np. w kontrolerze, który rozszerza klasę AbstractController przy pomocy metody createFormBuiler(). Symfony zaleca umieszczanie jak najmniejszej logiki w kontrolerach, dlatego lepiej jest przenosić złożone formularze do dedykowanych klas, zamiast definiować je w akcjach kontrolera. Ponadto formularze zdefiniowane w klasach mogą być ponownie wykorzystywane w wielu akcjach i usługach. W przypadku projektu badawczego formularz wykorzystany został w funkcjonalności zapisywania na wydarzenie, w której użytkownik do uzupełnienia ma tylko dwa pola, w związku z czym definicja formularza umieszczona została w kontrolerze. W kontrolerze przy pomocy wspomnianej metody createFormBuiler() tworzona jest definicja formularza. Kolejne pola formularza definiowane są za pomocą metody add(), która w argumentach przyjmuje nazwę pola oraz typ. Metoda getForm() zwraca gotowy obiekt formularza.

Przekazanie formularza do widoku odbywa się przy użyciu metody createView(). Formularz przekazany może być do widoku signup/index.html.twig poprzez parametr applicationForm, w którym jest renderowany za pomocą funkcji pomocniczych form_start, form_widget, form_end. Przycisk zatwierdzający znajduje się bezpośrednio w widoku. Przekazanie danych z formularza odbywa się poprzez metodę handleRequest() oraz obiekt uzupełnionego formularza ($form), dla którego sprawdzane są warunki isSubmitted() i isValid(). Po spełnieniu warunków na obiekcie encji Application przypisywane są wartości jej pól. Obiekt encji poprzez menadżer encji jest następnie kolejkowany (metoda persist()) i zapisywany w tabeli bazy danych (metoda flush()).

4.3. Obsługa baz danych

Laravel w wersji 7 obecnie wspiera następujące systemy zarządzania bazami danych: MySQL, PostgreSQL, SQLite, SQL Server.

Konfiguracja bazy danych odbywa się w pliku config/database.php, jednak dla ułatwienia parametry dostępu do bazy danych definiowane mogą być również w pliku .env, który zawiera zmienne globalne projektu. Na Listingu 5 zaprezentowane są parametry połączenia z bazą danych, która została wykorzystana w projekcie badawczym.

Listing 5: Plik .env z konfiguracją bazy danych

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel1
DB_USERNAME=root
DB_PASSWORD=
```

Korzystanie z bazy danych w projekcie Laravela wspierane jest przez konstruktor zapytań Database Query Builder, a także Eloquent ORM. Query Builder pozwala na generowanie zapytań do bazy danych w wygodny i płynny sposób nawet bez dogłębnej znajomości języka SQL. Wykorzystywane są tutaj metody wywoływane na wbudowanej fasadzie DB. Na ich podstawie możliwe jest skonstruowanie dowolnych zapytań SQL. Przykładowo metoda index() kontrolera ApplicationController zawiera zapytanie zwracające informacje o wydarzeniu, którego identyfikator zawarty jest w zmiennej $event_id. Metoda zwraca wynik zapytania poprzez parametr $event do widoku signup.create (listing 6).

Listing 6: Zapytanie zwracające informacje o wydarzeniach (plik AppController.php)

```
public function index($event_id){
  $event = DB::table('events')
    ->select('events.id', 'events.name', 'events.price',
      'events.start_date', 'events.end_date',
      'events.city', 'events.city')
    ->where('events.id', '=', $event_id)
    →first();
    return view('signup.create')->with('event', $event); }
```

Stosowane w Laravel narzędzie Eloquent ORM pozwala natomiast na prostą interakcję z bazą danych, która polega na przechowywaniu w pamięci obiektu, którego odpowiednikiem jest tabela w bazie danych. Obiekt taki, może posiadać zestaw metod, które pozwalają na interakcję z tą tabelą. Z pomocą modeli możliwe jest wyszukiwanie danych w tabelach, a także dodawanie nowych rekordów. Zastosowana konwencja nazw przyjmuje, że model Event przechowuje rekordy w tabeli events, a więc nazwa tabeli jest nazwą modelu z dopisaną literą ‘s’. Możliwe jest jednak wskazanie niestandardowej tabeli definiując w klasie modelu wartość chronioną $table, której wartością będzie nazwa tabeli, np.

protected $table = 'my_events';

Utworzony model znajduje się w katalogu app i jest on rozszerzeniem wbudowanej klasy Model (Listing 7). Każdy model pozwala na używanie metod Query Builder’a.

Listing 7: Definicja modelu Event (plik Event.php)

```
namespace App;
use Illuminate\Database\Eloquent\Model;
class Event extends Model{ }
```

Migracje w Laravel to mechanizm pozwalający na zautomatyzowane tworzenie tabel w bazie danych. Umożliwiają zachowywanie informacji o tabelach w repozytorium dzięki czemu ułatwia ich odtworzenie w innym środowisku. Migracje znajdują się w katalogu database/migrations. Klasa migracji, która rozszerza klasę Migration, zawiera publiczną funkcję tworzącą tabelę up() oraz funkcję usuwającą tabelę down().

Zastosowanie seed’ów (ang. seeds) pozwala z kolei na automatyczne generowanie danych i uzupełnianie nimi tabel w bazie danych. Wykorzystanie tej funkcjonalności odbywa się poprzez utworzenie klasy rozszerzającej klasę Seeder i implementację metody run(), w której znajdzie się przypisanie wartości do kolumn. Listing 8 prezentuje automatyczne generowanie wartości dla tabeli wydarzeń.

Listing 8: Klasa EventsTableSeeder (plik EventsTableSeeder.php)

```
class EventSeeder extends Seeder{
  public function run() {
    $event = new Event();
    $event->name = 'Oaza Nowego Życia I';
    $event->city = 'Lublin';
    $event->start_date = '2020-09-20';
    $event->end_date = '2020-09-30';
    $event->app_deadline = '2020-09-19';
    $event->description = 'Twoje pierwsze rekolekcje';
    $event->max_app_count = 50;
    $event->price = 550;
    $event→save(); }}
```

Symfony udostępnia wszystkie narzędzia potrzebne do korzystania z baz danych dzięki Doctrine, zestawowi bibliotek PHP do pracy z bazami danych. Narzędzia te obsługują silniki relacyjnych baz danych, takie jak MySQL i PostgreSQL, a także bazy danych NoSQL, takie jak MongoDB [16]. Konfiguracja bazy danych odbywa się tutaj w pliku .env projektu, w którym to w zmiennej środowiskowej o nazwie DATABASE_URL przechowywane są takie dane jak: typ silnika bazodanowego (mysql), nazwa użytkownika serwera MySQL (root), hasło użytkownika (w tym przypadku brak hasła), adres i numer portu, pod którym dostępny jest serwer (127.0.0.1:3306), nazwa bazy danych (symfony) oraz wersja serwera (mariadb-10.4.14).

Podobnie jak w przypadku Laravela, możliwe jest utworzenie klasy obiektu odpowiadającego rekordowi w tabeli. Klasa ta nazywana jest encją. W Doctrine możliwe jest użycie narzędzia make:entity, które w przypadku konieczności utworzenia nowej encji (tabeli) zdecydowanie ułatwia dodawanie, usuwanie czy aktualizację kolejnych jej pól przed wykonaniem migracji na bazie. Utworzenie encji odbywa się za pomocą polecenia:

php bin/console make:entity

Po wywołaniu powyższego polecenia w konsoli, pojawią się odpowiednie zapytania co do nazwy encji oraz

nazw pól, które będzie ona posiadać. W ten sposób utworzona została klasa Event ze zdefiniowanymi polami i zestawem metod dostępowych typu get i set. Klasa ta ma postać jak na listingu 9.

Listing 9: Fragment klasy encji Event (plik Event.php)

```
class Event{
  private $id;
  ...
  private $description;
  public function getId(): ?int{
            return $this->id; }
  public function getName(): ?string{
            return $this->name; }
  public function setName(string $name): self{
     $this->name = $name;
     return $this; }
  public function getCity(): ?string{
     return $this->city; }
  public function setCity(string $city): self{
     $this->city = $city;
     return $this; }  … }
```

Kolejno, podobnie jak w Laravel generowana jest klasa migracji. Wygenerowanie klas migracji dla wszystkich istniejących encji możliwe jest za pomocą polecenia:

php bin/console make:migration

W ten sposób dla encji Event tworzona utworzona zostanie klasa migracji jak na listingu 10.

Listing 10: Klasa migracji (plik Version20201105193925.php)

```
final class Version20201105193925 extends AbstractMigration{
   public function getDescription() : string{
      return ''; }
   public function up(Schema $schema) : void {
      $this->addSql('CREATE TABLE event (id INT AU-
TO_INCREMENT NOT NULL, name VARCHAR(255) NOT
NULL, city VARCHAR(255) NOT NULL, start_date DATE NOT
NULL, end_date DATE NOT NULL, app_deadline DATE NOT
NULL, max_app_count INT NOT NULL, price INT NOT NULL,
description VARCHAR(255) NOT NULL, PRIMARY KEY(id))
DEFAULT CHARACTER SET utf8mb4 COLLATE
`utf8mb4_unicode_ci` ENGINE = InnoDB'); }
   public function down(Schema $schema) : void {
      $this->addSql('DROP TABLE event'); }}
```

Klasa ta w funkcji up() zawiera wygenerowane zapytanie SQL tworzące odpowiednią tabelę w bazie danych. Wywołanie funkcji down() natomiast spowoduje usunięcie utworzonej w bazie danych tabeli. Wywołanie migracji na bazie danych odbywa się przy użyciu polecenia:

php bin/console doctrine:migrations:migrate

W Symfony analogicznie jak w przypadku Laravela możliwe jest wykorzystanie mechanizmu uzupełniania tabel danymi. Tutaj jednak do czynienia mamy z wbudowaną klasą o nazwie Fixture i metodą load(). Wykorzystanie jej możliwe jest po wywołaniu następującego polecenia:

composer require --dev orm-fixtures

Po implementacji klasa definicji Fixture'a dla encji Event przyjmuje postać jak na listingu 11.

Listing 11: Klasa EventFixtures (plik EventFixtures.php)

```
namespace App\DataFixtures;
use App\Entity\Event;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;

class EventFixtures extends Fixture{
   public function load(ObjectManager $manager){
      $event = new Event();
      $event->setName('Oaza Nowego Życia Ist.');
      $event->setAppDeadline(new \DateTime(2021-03-31));
      $event->setCity('Lublin');
      $event->setDescription('To jest pierwsza oaza');
      $event->setStartDate(new \DateTime('2021-04-01'));
      $event->setEndDate(new \DateTime('2021-04-10'));
      $event->setMaxAppCount(50);
      $event->setPrice(550);
      $manager->persist($event);
      $manager→flush(); }}
```

W powyższej definicji kluczowe jest wskazanie klasy encji, ponieważ obiekt tej klasy tworzony jest w funkcji load(). Jako, że nazwa encji odpowiada nazwie tabeli w bazie danych, w późniejszym kroku dany obiekt zapisany zostanie w odpowiadającej mu tabeli. W funkcji load() wywoływane są metody typu set dla poszczególnych pól obiektu klasy Event. Po ustawieniu wartości pól konieczne jest zapisanie obiektu poprzez wykorzystanie menadżera obiektów (ObjectManager) i metod persist() oraz flush(). Zapisanie utworzonego obiektu w tabeli bazy danych możliwe jest poprzez polecenie:

php bin/console doctrine:fixtures:load

Po jego wywołaniu dane wszystkich zdefiniowanych Fixtures zostaną zapisane w odpowiednich tabelach bazy danych. Doctrine podobnie jak Eloquent wykorzystuje mapowanie relacyjno-obiektowe (ORM) w związku z czym ułatwione zostaje wykonywanie zapytań na bazie danych. Przykładowo metoda new() widoczna na Listingu 12 zwróci tablicę z wartościami kolumn z tabeli event (klasa encji Event), gdzie wartość w kolumnie id tablicy będzie równa wartości zapisanej w zmiennej $event_id. Wynik zapytania w dalszych krokach przekazany zostanie za pomocą zmiennej $events do widoku home.html.twig.

Listing 12: Metoda zwracająca rekordy z tabeli event (plik HomeController.php)

```
public function new(Request $request, $event_id){
   $events = $this->getDoctrine()
      ->getRepository(Event::class)
      ->findBy(array('id'=> $event_id)); …
```

4.4. Obsługa gniazd sieciowych - WebSocket

Gniazda sieciowe typu Websocket pozwalają na połączenie dwukierunkowe serwera aplikacji z przeglądarką. Laravel umożliwia ich obsługę np. za pomocą usługi Pusher [12]. Kolejno, wykorzystanie pakietu laravel-websockets znacząco ułatwia implementację połączeń typu WebSocket, ponieważ zawiera on odpowiednią konfigurację, panel debugowania oraz migracje tabel zawierających statystyki przesyłanych danych. Użycie tego pakietu w pierwszej kolejności odbywa się

poprzez ustawienie jego wymagalności w narzędziu Composer za pomocą polecenia:

```
composer require beyondcode/laravel-websockets
```

Kolejnym krokiem jest wykonanie migracji i publikacja pliku konfiguracyjnego za pomocą polecenia:

```
php artisan vendor:publish –
provider="BeyondCode\LaravelWebSockets
WebSocketsServiceProvider" --tag="config".
```

Konfiguracja połączeń typu WebSocket znajduje się w pliku websockets.php. Standardowe ustawienia zawarte w pliku w przypadku realizacji projektu badawczego są wystarczające. Następnym krokiem jest wypełnienie zmiennych środowiskowych tj. parametrów:

```
APP_NAME, PUSHER_APP_ID, PUSHER_APP_KEY,
PUSHER_APP_SECRET
```

w pliku .env. Po uruchomieniu serwera istnieje możliwość monitorowania zdarzeń. Parametr path w pliku konfiguracyjnym określa podstronę, na której dostępny jest panel debugowania. Panel prezentuje statystyki przepływu danych przez zestawione połączenie, umożliwia podgląd przesyłanych danych, a także ich testowe przesyłanie. W pliku broadcasting.php zdefiniowana jest konfiguracja usługi Pusher poprzez podanie adresu hosta i port, na którym ma działać usługa. Przesyłanie wiadomości w czasie rzeczywistym za pomocą czatu, który jest przykładem wykorzystania połączeń typu WebSocket możliwe jest dzięki implementacji zdarzeń (ang. Event).

Listing 13: Przykład klasy Eventu (plik WebSocket Demo Event.php)

```
class WebsocketDemoEvent implements ShouldBroadcast{
use Dispatchable, InteractsWithSockets, SerializesModels;
public $somedata;
public function __construct($somedata){
 $this->somedata = $somedata; }
  public function broadcastOn(){
      return new Channel('DemoChannel'); }}
```

Klasa zdarzenia widoczna na listingu 13 jest implementacją wbudowanego interfejsu ShouldBroadcast. Interfejs ten wymaga zdefiniowania metody broadcastOn(), która odpowiada za zwrócenie kanałów, na których wydarzenie powinno być transmitowane. W parametrze konstruktora klasy przekazywana jest wartość, która ma zostać przesłana za pomocą połączenia Websocket. Metoda sendMessage wysyłająca wiadomości czatu wygląda jak na listingu 14. Zmienna message przechowuje treść przesyłanej wiadomości. Funkcja broadcast przyjmuje jako argument obiekt klasy MessageSent, a w nim zmienną message, którą następnie przesyła (rozgłasza) na wszystkich użytkowników.

Listing 14: Metoda sendMessage (plik ChatsController.php)

```
public function sendMessage(Request $request){
  $message = auth()->user()->messages()->create([
     'message' => $request→message ]);
  broadcast(new MessageSent($message->load('user')))->toOthers();
  return ['status' => 'success'];}
```

W celu implementacji połączeń typu WebSocket w Symfony konieczne jest wykorzystanie biblioteki Ratchet. Biblioteka ta udostępnia narzędzia potrzebne to tworzenia dwukierunkowych aplikacji działających w czasie rzeczywistym pomiędzy klientami, a serwerem poprzez połączenia typu WebSocket. W projekcie wykorzystanie Ratchet rozpoczyna się od modyfikacji pliku *composer.json*, w którym należy dopisać odpowiednią klauzulę:

```
"require":{
"cboden/ratchet": "^0.4.1", …}
```

Kolejnym krokiem jest stworzenie klasy WebSocket. W projekcie klasa ta ma postać jak na listingu 15.

Listing 15: Klasa WebSocket (plik Chat.php)

```
class Chat implements MessageComponentInterface{
  protected $connections = [];
  public function __construct(){}
  function onOpen(ConnectionInterface $conn): void{
    $this->connections[] = $conn;
    $conn->send('Witaj!' . PHP_EOL);}
  function onClose(ConnectionInterface $conn): void{
     foreach ($this->connections as $key => $connection){
        if($connection === $conn){
           unset($this->connections[$key]);
           break; }}
    $conn->send('Do zobaczenia!' . PHP_EOL);
    $conn→close(); }
  function onError(ConnectionInterface $conn, \Exception $e): void{
    $conn->send('Error ' . $e->getMessage() . PHP_EOL);
    $conn→close(); }
  function onMessage(ConnectionInterface $from, $msg): void{
    $messageData = json_decode(trim($msg));
    foreach ($this->connections as $connection){
       $connection→send($messageData); }}}
```

Klasa MessageComponentInterface jest klasą abstrakcyjną, dlatego wymaga użycia wszystkich czterech metod: onOpen, onClose, onError i onMessage. Metoda onOpen odpowiada za działania podczas nawiązania nowego połączenia z serwerem, w tym przypadku wyświetlenie komunikatu: *Do zobaczenia!* Metoda onClose zostaje uruchomiona w momencie zakończenia połączenia przez klienta. Metoda onError jest uruchamiana w momencie zgłoszenia błędu przez połączenie. Metoda onMessage odpowiada za obsługę wiadomości przesyłanych na serwer. W projekcie badawczym każda wiadomość przesyłana jest do każdego klienta w czasie rzeczywistym.

Po implementacji logiki służącej do przetwarzania połączeń przychodzących na serwer należy zaimplementować nasłuchiwanie portu, w tym przypadku będzie to port 8080. W klasie ChatServer tworzony jest obiekt $chatServer (listing 16), który zawiera serwer HTTP wygenerowany za pomocą obiektu klasy WsServer (wbudowana klasa biblioteki Ratchet) oraz obiektu klasy Chat.

Listing 16: Funkcja execute klasy ChatServer (plik ChatServer.php)

```
protected function execute(InputInterface $input, OutputInterface
$output){
$chatServer = IoServer::factory(new HttpServer(new WsServer(new
```

```
Chat())), 8080);
$chatServer→run(); }
```

Wyświetlanie wiadomości jak i otwarcie połączenia typu WebSocket odbywa się z wykorzystaniem JavaScript jak na listingu poniżej. Zmienna socket przechowuje obiekt gniazda sieciowego, na którym wywoływane są metody onopen, onclose, onmessage, onerror. W dalszej części kodu, generowany jest widok, wyświetlający odpowiednie pola czatu (listing 17).

Listing 17: Obsługa połączenia (plik main.js)

```
var socket = new WebSocket("ws://127.0.0.1:8080");
socket.onopen = function () {
  console.log('Connection successful'); };
socket.onclose = function (event) {
  if (event.wasClean) {
    console.log('Connection closed.'); }
else {
    console.log('Connection killed:('); }
  console.log(event.code + event.reason); };
socket.onmessage = function (event) {
  var list = document.getElementById('list');
  var message = document.createElement("div");
  var node = document.createTextNode(event.data);
  message.appendChild(node);
  message.classList.add('card');
  list.appendChild(message); };
socket.onerror = function (error) {
  console.log(error.message); };
var button = document.getElementById('send');
var textarea = document.getElementById('message-box');
function sendText() {
  var text = textarea.value;
  if (text.length > 0) {
    socket.send(JSON.stringify(text));
    textarea.value = '';
    return true; }
  return false; }
button.onclick = sendText;
textarea.onkeypress = function (ev) {
  if (ev.charCode === 13 && ev.shiftKey) { sendText(); } };
```

4.5. Złożoność kodu

W większości przypadków złożoność kodu przekłada się wprost proporcjonalnie do czasu poświęconego na implementację. W przypadku szkieletu Laravel średnia liczba linii kodu przypadająca na implementację funkcjonalności projektu badawczego wynosi: 96,5 (tabela 1).

Tabela 1: Złożoność kodu dla Laravel

Funkcjonalność	Liczba plików	Liczba linii kodu
Migracja tabeli wydarzeń	1	39
Wyświetlenie wydarzeń	3	89
Zapisanie na wydarzenie	4	114
Czat	7	144

W przypadku szkieletu Symfony średnia liczba linii kodu przypadająca na implementację funkcjonalności projektu badawczego wynosi: 150,75 (tabela 2).

Wynik badania może dowodzić większemu skomplikowaniu kodu implementowanego w Symfony.

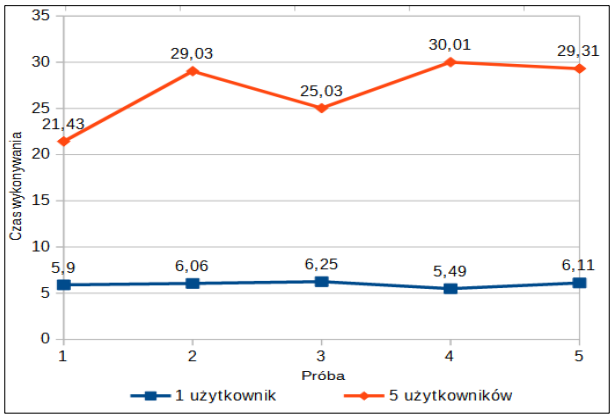
Funkcjonalność	Liczba plików	Liczba linii kodu
Migracja tabeli wydarzeń	1	31
Wyświetlenie wydarzeń	3	220
Zapisanie na wydarzenie	4	179
Czat	5	173

Tabela 2: Złożoność kodu dla Symfony

4.6. Wydajność

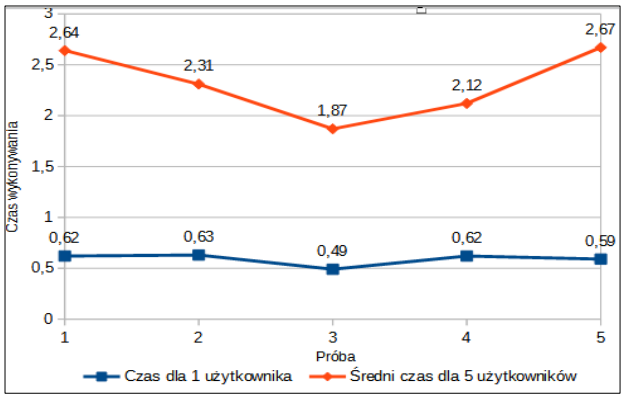
Wydajność projektu docelowego jest najczęstszym kryterium wyboru narzędzi programistycznych przez deweloperów. Przeprowadzono badanie z wykorzystaniem funkcjonalności wyświetlenia 100 rekordów znajdujących się w bazie danych, które dla poszczególnych szkieletów zawierały te same dane. Zaimplementowano metodę zwracającą informacje dotyczące wydarzeń i wyświetlająca te informacje w widoku oraz przeprowadzono zapisywanie na wydarzenia – zatwierdzono formularz zawierający uzupełnione przez użytkownika dane, który wywołał metodę zapisującą nowy rekord w bazie danych. Wykonano po 5 prób badań. Każdą próbę przeprowadzono w obrębie tego samego systemu, w którym uruchomiony został serwer PHP.

Średni czas oczekiwania na wyświetlenie wyników wyniósł odpowiednio 5,96 sekund w przypadku obsługi jednego użytkownika i 26,96 sekund w przypadku obsługi równoległej pięciu użytkowników w obrębie jednego systemu operacyjnego. Szczegółowe wyniki dla wykonanych prób przedstawia wykres widoczny na rysunku 1.



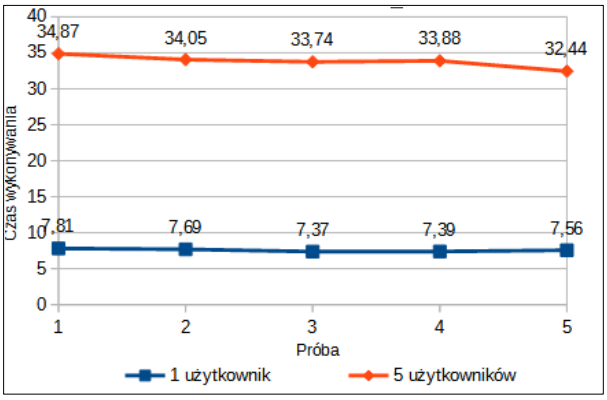
Rysunek 1: Czas wyświetlenia listy wydarzeń w Laravel.

Zapisanie danych w bazie i przeładowanie strony w Laravel odbyło się średnio w czasie odpowiednio 0,59 sekundy w przypadku obsługi jednego użytkownika i 2,32 sekundy w przypadku obsługi równoległej pięciu użytkowników w obrębie jednego systemu operacyjnego. Szczegółowe wyniki dla wykonanych prób przedstawia wykres widoczny na rysunku 2.



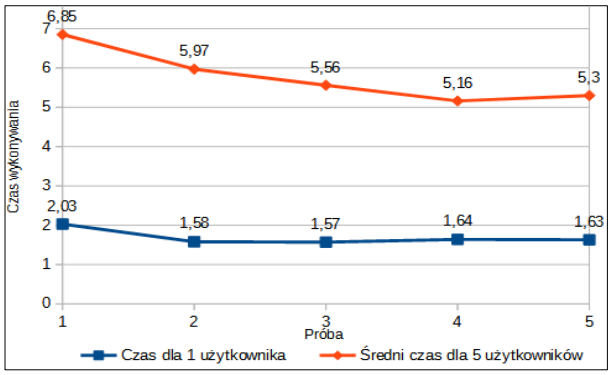
Rysunek 2: Czas zapisania zgłoszenia i przeładowania strony w Laravel.

Średni czas oczekiwania na wyświetlenie wyników wyniósł dla Symfony odpowiednio 7,56 sekund w przypadku obsługi jednego użytkownika i 33,79 sekund w przypadku obsługi równoległej pięciu użytkowników w obrębie jednego systemu operacyjnego. Szczegółowe wyniki dla wykonanych prób przedstawia wykres widoczny na rysunku 3.



Rysunek 3: Czas wyświetlenia listy wydarzeń w Symfony.

Zapisanie danych w bazie i przeładowanie strony w Symfony odbyło się średnio w czasie odpowiednio 1,69 sekundy w przypadku obsługi jednego użytkownika i 5,76 sekundy w przypadku obsługi równoległej pięciu użytkowników w obrębie jednego systemu operacyjnego. Szczegółowe wyniki dla wykonanych prób przedstawia wykres widoczny na rysunku 4.



Rysunek 4: Czas zapisania zgłoszenia i przeładowania strony w Symfony.

5. Podsumowanie

Przeprowadzone badania informują o możliwościach, które daje każdy ze szkieletów. Laravel jest szkieletem programistycznym dość prostym do nauczenia, a jednocześnie posiadającym wszystkie niezbędne mechanizmy do implementacji projektu. Dostępność dokumentacji i aktywna społeczność sprawia, że jest to szkielet bardzo często wybierany przez deweloperów. Podobnie wygląda sytuacja dla Symfony. Tutaj jednak możliwe jest osiągnięcie dogodnych rozwiązań dla projektów rozbudowanych i wymagających ciągłego rozwoju. Tego typu projekty wymagane są w wielu przedsiębiorstwach, gdzie ważne jest ich nieustanne dostosowywanie do zmieniających się wymagań. Informacje zawarte w rozdziale dotyczącym funkcjonalności szkieletów budowy aplikacji znajdują potwierdzenie w kolejnych punktach niniejszej pracy.

W przypadku bezpieczeństwa obydwa szkielety oferują podobne rozwiązania, przy czym implementacja np. systemu uwierzytelniania w Laravel jest znacznie przyspieszona, ponieważ udostępnia on gotowy system uwierzytelniania. Szkielety jednak mają swoje społeczności, które często udostępniają rozwiązania gotowe do implementacji. Obsługa baz danych w przypadku omawianych szkieletów wygląda analogicznie. Implementują mechanizmy usprawniające działania na bazach typu Query Builder, wzorce Active Record lub Data Mapper. Symfony wymaga bardziej szczegółowego określenia reguł dostępu do bazy.

Oba szkielety umożliwiają implementację połączeń typu WebSocket. W przypadku Laravela wydaje się ona szybsza, np. ze względu na udostępniany panel debugowania ułatwiający analizę przepływu danych na serwerze, co pozwala na łatwiejsze wyszukanie błędów działania aplikacji.

Laravel i Symfony są ciągle rozwijane i dostosowywane do zmian wprowadzanych na bieżąco w nowych wersjach języka PHP. Wprowadzane usprawnienia zastosowane w czystym języku przekładają się na usprawnienia wprowadzane w szkieletach budowy aplikacji. W realizacji małych projektów jakim był zaprezentowany projekt badawczy, lepiej sprawdził się Laravel. Wydajność projektu badawczego w przypadku Laravel była większa niż Symfony, natomiast złożoność kodu w Laravel była mniejsza.

Otrzymane wyniki mogą posłużyć jako wskazówki, które mogą być wykorzystane podczas podejmowania decyzji o wyborze docelowego szkieletu programistycznego przez deweloperów.

Literatura

[1] Informacje z ApacheFriends o XAMPP, https://www.apachefriends.org/pl/index.html, [28.12.2020].

[2] M. Laaziri, K. Benmoussa, S. Khoulji, M. L. Kerkeb, A. E. Yamami, A comparative study of Laravel and Symfony PHP frameworks, IJECE, 2019.

[3] Artykuł zawierający porównanie frameworków, https://www.quora.com/What-are-the-major-difference between-PHP-5-and-PHP-7, [28.12.2020].

[4] Porównanie PHP 5.6 i PHP 7, https://gbksoft.com/blog/php-5-vs-php-7-performance-comparison/, [02.01.2020].

[5] T. Skaraczyński, A. Zioła, PHP5. Programowanie z wykorzystaniem Symfony, CakePHP, Zend Framework, Helion, Gliwice, 2009.

[6] M. Gołda, Laravel Tworzenie aplikacji Receptury, Helion, Gliwice, 2015.

[7] P. G. de Gennes, Scaling Concepts in Polymer Physics, Cornell University Press, London, 1979.

[8] Dokumentacja szkieletu Laravel, https://laravel.com/, [28.12.2021].

[9] Dokumentacja szkieletu Symfony, https://symfony.com/, [28.12.2021].

[10] Konfiguracja Doctrine, https://symfony.com/doc/current/reference/configuration/doctrine.html, [28.12.2020].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 35–39
Received: 20 October 2021
Accepted: 26 October 2021

Comparison of web application performance on the example of Laravel and Vaadin frameworks

Porównanie wydajności aplikacji internetowych na przykładzie szkieletów programistycznych Laravel i Vaadin

Jakub Radomski*

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

Abstract

The article presents the result of comparing the performance of web applications written with the use of Laravel and Vaadin frameworks. The aim of the study was to identify which framework offers better time efficiency and to verify the thesis: "A web application created using the Vaadin framework is more efficient than an application created using the Laravel framework". A research method based on a comparative analysis was used to conduct the study and with the help of two applications was tested the speed of resource generation, or the times obtained in work under load. Tests were carried out according to 13 research scenarios and the obtained results allowed to clearly verify the thesis was proven.

Keywords: Laravel; Vaadin; Web application; Performance

Streszczenie

Artykuł przedstawia wyniki porównania wydajności aplikacji internetowych napisanych przy użyciu szkieletów aplikacji Laravel oraz Vaadin. Celem badania było wskazanie, który szkielet oferuje lepszą wydajność czasową oraz zweryfikowanie postawionej tezy brzmiącej "Aplikacja internetowa stworzona przy użyciu platformy programistycznej Vaadin jest bardziej wydajna niż aplikacja stworzona za pomocą platformy Laravel". Do przeprowadzenia badania wykorzystano metodę badawczą opartą na analizie porównawczej i przy pomocy dwóch aplikacji zbadano między innymi prędkość generowania zasobów, czy czasy uzyskane w pracy pod obciążeniem. Przeprowadzono próby według 13 scenariuszy badań a otrzymane wyniki pozwoliły jednoznacznie zweryfikować, że postawiona teza została udowodniona.

Słowa kluczowe: Laravel; Vaadin; Aplikacja internetowa; Wydajność

*Corresponding author

Email address: jakub.radomski@pollub.edu.pl (J. Radomski)

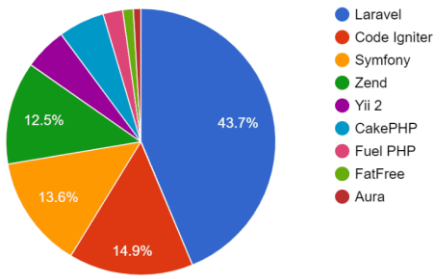
1. Wstęp

W dzisiejszych czasach programiści mają do wyboru mnogość narzędzi i języków a dodatkowo każdy język oferuje różne szkielety do budowy aplikacji mające na celu ułatwić ich tworzenie.

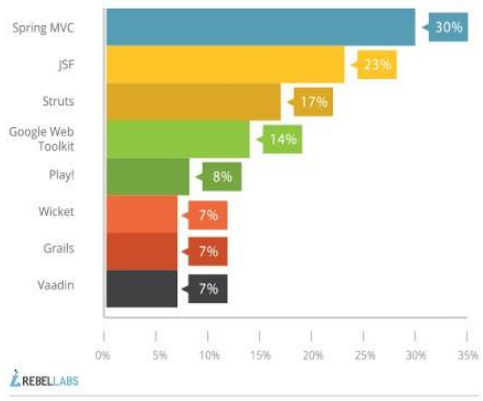
Aplikacja internetowa może zostać napisana przy użyciu różnych języków programowania. Ta sama aplikacja może powstać z użyciem języka PHP lub, na przykład wykorzystując język Java i będzie oferowała te same funkcje oraz ten sam interfejs użytkownika. Jednak nie każdy język, czy szkielet do budowy aplikacji cechuje się tą samą prostotą użycia, mnogością dostępnych gotowych rozwiązań, czy wydajnością działa aplikacji. Na tym ostatnim aspekcie skupia się ten artykuł, porównując wydajność popularnego wśród programistów PHP szkieletu Laravel (Rysunek 1) oraz zyskującego powoli na popularności szkieletu Vaadin (Rysunek 2).

2. Przegląd literatury

Wybrane szkielety do tworzenia aplikacji internetowych nie miały bezpośredniego porównania wydajności i jest to jeden z powodów dla, którego zdecydowano się przeprowadzić to badanie. Laravel jako rozwiązanie, które zyskało sporą popularność na przestrzeni ostatnich lat, doczekało się wielu artykułów naukowych. Poruszano w nich tematykę wydajności względem innych szkieletów do tworzenia aplikacji, subiektywną ocenę łatwości używania, czy liczby dostępnych gotowych rozwiązań jakie oferuje. Co jednak jest ciekawe, w kwestii wydajności można natrafić na prace, które są ze sobą sprzeczne.



Rysunek 1: Popularność szkieletów PHP w roku 2018 [1].



Rysunek 2: Popularność szkieletów Java w roku 2020 [2].

Artykuł napisany przez Przemysława Mincewicza [3] bada aplikacje napisane z użyciem szkieletów Laravel, Symfony oraz Zend pod kątem wydajności i możliwości. Porównana zostaje wydajność czasowa 3 aplikacji. Autor po wykonaniu 23 zaplanowanych scenariuszy badań dochodzi do wniosku, że Laravel "wypadł, zdecydowanie najsłabiej ze wszystkich porównywanych frameworków". Zostały zbadane takie aspekty jak czas dostępu do bazy danych, czy czas ładowania plików graficznych. W artykule zostało przedstawionych 9 wykresów dotyczących badań wydajności, gdzie w 4 z nich szkielet Laravel wcale nie przegrał z Symfony i Zend, więc z powyższym stwierdzeniem autora można się nie zgodzić. Z kolei artykuł zatytułowany "A Comparative study of PHP frameworks performance" [4] pokazuje szkielet Laravel z innej strony. Autorzy również porównywali szkielet Laravel, tym razem jednak z Symfony i Codeigniter. Doszli do wniosku, że jest szkielet Laravel jest najwydajniejszy względem dwóch pozostałych. Wniosek ten opracowali na podstawie zrealizowanych 4 scenariuszy badań. Zbadali aplikacje nie tylko pod względem wydajności czasowej ale również wykorzystanie pamięci wszystkich 3 aplikacji. Powstał również artykuł porównujący wydajność Laravel z platformą programistyczną Spring [5]. Jest to więc porównanie już dwóch różnych języków. Autorzy porównywali takie aspekty jak bezpieczeństwo czy wydajność czasową. Analizę wydajności przeprowadzili przy pomocy narzędzia JMeter oraz Chrome DevTools, dzięki któremu zbadali czasy odpowiedzi dla obydwu platform. Zbadali między innymi czas odpowiedzi strony głównej aplikacji, czy czas odpowiedzi dla wybranej podstrony. Prawie pięciokrotnie lepsze rezultaty uzyskał Spring. Ciekawe porównanie zrealizowano w pracy pod tytułem "Performance Comparison of QEC Network based JAVA Application and Web based PHP Application", ponieważ badanie zrealizowano bez użycia szkieletów do budowy aplikacji, a więc tylko z wykorzystaniem języków Java i PHP [6]. Jest to kolejna praca, w której zbadano wydajność czasową przy użyciu narzędzia JMeter. Również w tym badaniu Java okazała się szybsza względem PHP. Platformę Spring porównał również w swoim artykule Łukasz Tomczyk [7]. Jednak w jego pracy platforma ta została zestawiona z platformą Vaadin. Do testów autor przygotował 9 scenariuszy. Autor skupił się na wydajności czasowej i podczas realizacji zaplanowanych scenariuszy zbadał efektywność pracy z bazą danych oraz efektywność wczytywania zasobów w przeglądarce internetowej. W jego badaniu zarówno Vaadin, jak i Spring wykazywały miejsca w, których okazywały się lepsze niż druga badana platforma. Należy odnotować, że wszystkie badania przeprowadzone w wyżej wymienionych artykułach, każde zestawienie platform, czy języków zostało zrealizowane na aplikacjach, które posiadała możliwie identyczny wygląd, mechanikę działania oraz identyczną bazę danych w celu zapewnienia możliwie jak najbardziej miarodajnych wyników. Platformy programistyczne, jak również i języki z wykorzystaniem których zostały zbudowane, ciągle się zmieniają, otrzymują następne wersje. Należy mieć więc na uwadze, że ich wydajność będzie się stale zmieniać i rezultaty niektórych badań mogłyby być teraz inne.

3. Cel, teza i metoda badawcza

Celem badania było porównanie wydajności czasowej obydwu platform programistycznych poprzez sprawdzenie jakie wyniki uzyskują w testach sprawdzających, między innymi prędkość generowania zasobów, czy czasy uzyskane w pracy pod obciążeniem. Postawiono również tezę brzmiącą: "Aplikacja internetowa stworzona przy użyciu platformy programistycznej Vaadin jest bardziej wydajna niż aplikacja stworzona za pomocą platformy Laravel". Dla potwierdzenia przedstawionej tezy wykorzystano metodę badań opartą na analizie porównawczej obu rozwiązań i w tym celu stworzono dwie aplikacje różniące się jedynie narzędziem użytym do ich stworzenia, natomiast baza danych, czy funkcjonalności zostały zachowane te same.

4. Aplikacje testowe

Aplikacje Laravel oraz Vaadin:

- stworzono w środowisku programistycznym IntelliJ z wykorzystaniem narzędzia Maven oraz przy użyciu środowiska PhpStorm z wykorzystaniem narzędzia Composer [8-11]. Zarówno Maven, jak i Composer pozwalają na automatyzację budowy projektów.
- współpracują z serwerem bazy danych MySQL [12].

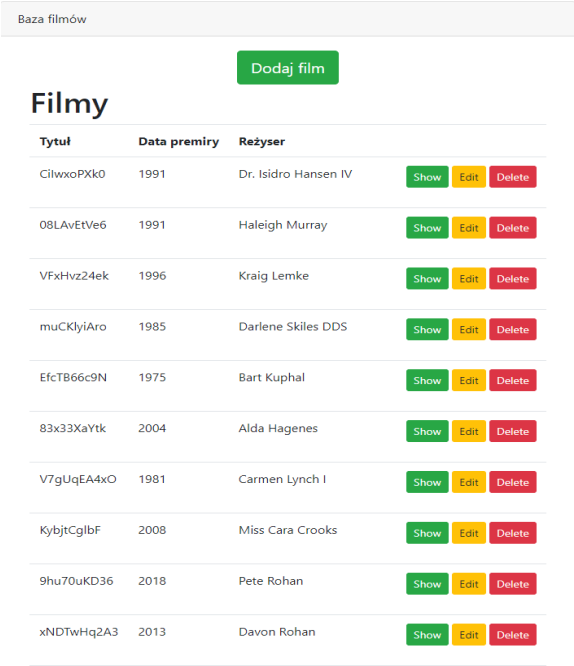
Obie aplikacje testowe posiadają typowe funkcjonalności CRUD (ang. Create, Read, Update, Delete), czyli tworzenie, edycję oraz usuwanie informacji o filmach bazy danych. Interfejs obu tych aplikacji jest bardzo podobny i przykładowa strona aplikacji została przedstawiona na Rysunku 3.

5. Analiza wydajności

W celu zbadania wydajności stworzonych aplikacji skorzystano z Chrome DevTools, narzędzie to jest wbudowane w przeglądarkę Google Chrome oraz programu ApacheBench, który pozwala symulować obciążenie aplikacji [13-14]. Analizę wydajnościową przedstawiono w postaci wykresów opisujących badane aspekty. Każdy z nich zawiera zestawienie wszystkich badanych narzędzi na tle dwóch zestawów danych. W badaniach

zwrócono uwagę na czas, opisany na wykresach w milisekundach (ms).

Założono 13 scenariuszy badań, których opisy przedstawiono w Tabela 1. Każdy badany scenariusz składa się z 10 powtórzeń oraz wyliczonych średnich czasów dla każdego z badanych szkieletów.



Rysunek 3: Przykładowa strona aplikacji testowej.

Tabela 1: Scenariusze badania wydajności

Scenariusz	Opis
1	Generowanie strony z 10 rekordami
2	Generowanie strony ze 100 rekordami
3	Generowanie strony z 1000 rekordów
4	Generowanie strony z 10000 rekordów
5	Zapis 1 rekordu
6	Zapis 10 rekordów
7	Zapis 100 rekordów
8	Usunięcie rekordu z bazy
9	Generowanie strony startowej
10	10 pobrań strony startowej symulowanej przez 1 użytkownika w jednej chwili
11	1000 pobrań strony startowej symulowanej przez 50 użytkowników w jednej chwili
12	10 pobrań listy 10 filmów symulowanej przez 1 użytkownika w jednej chwili
13	1000 pobrań listy 10 filmów symulowanej przez 50 użytkowników w jednej chwili

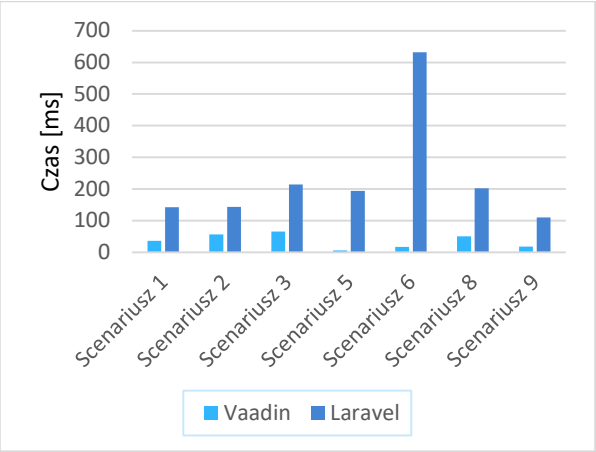
Badania zostały przeprowadzone na jednej jednostce pomiarowej, której parametry wraz z wersjami wykorzystanych technologii przedstawia Tabela 2.

Tabela 2: Parametry sprzętowe jednostki pomiarowej oraz wersje wykorzystanych technologii

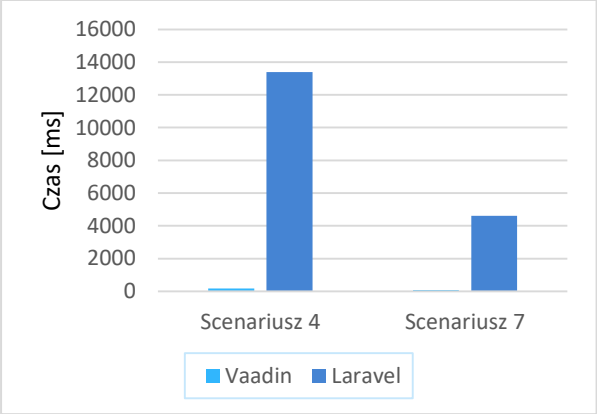
System operacyjny	Windows 10 Pro 64-bitowy
Procesor	Intel Core i5-9400F
Pamięć RAM	16 GB
PHP	Wersja 7.4
Java	Wersja 8
Laravel	Wersja 8.5
Vaadin	Wersja 14

5.1. Analiza z wykorzystaniem Chrome DevTools

Badanie z wykorzystaniem narzędzia Chrome DevTools dotyczy pierwszych 9 scenariuszy. Ich wyniki są przedstawione na Rysunkach 4 i 5. Zostały one podzielone na 2 oddzielne rysunki w celu zachowania lepszej czytelności wyników.



Rysunek 4: Średnie czasy operacji scenariuszy testowych.



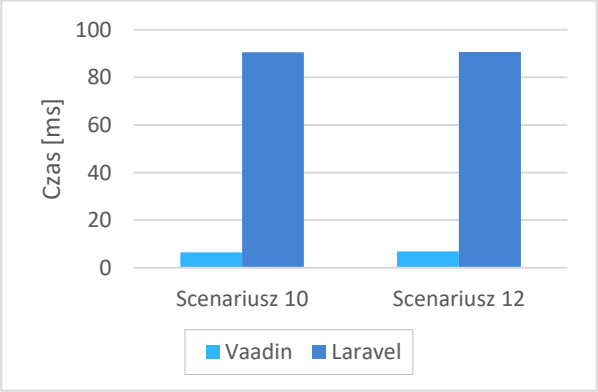
Rysunek 5: Średnie czasy operacji dla scenariusz 4 i 7.

Szkielet Vaadin osiągnął lepsze wyniki od szkieletu Laravel w każdym z 9 scenariuszy. Stopniowe zwiększanie liczby rekordów w 2 i 3 scenariuszu ma minimalny wpływ na otrzymywane czasy. Przewagę szkieletu Vaadin szczególnie widać w przypadkach zakładają-

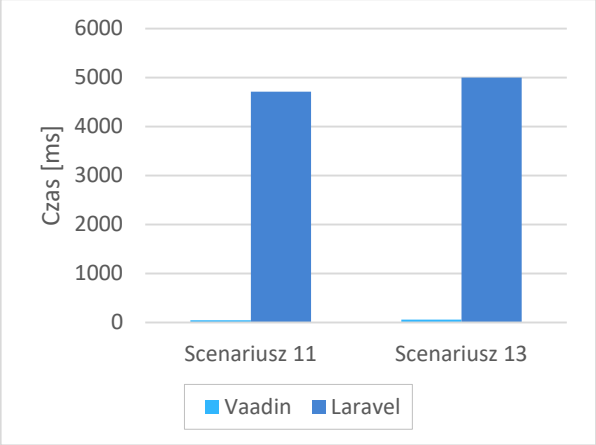
cych operacje na co raz większej liczbie rekordów tak jak podczas realizacji scenariusza 4 i 7.

5.2. Analiza z wykorzystaniem ApacheBench

Badanie z wykorzystaniem narzędzia ApacheBench dotyczy ostatnich 4 scenariuszy. Ich wyniki są przedstawione na Rysunkach 6 oraz 7. Zostały one podzielone na 2 oddzielne rysunki w celu zachowania lepszej czytelności wyników.



Rysunek 6: Średnie czasy operacji dla scenariusz 10 i 12.



Rysunek 7: Średnie czasy operacji dla scenariusz 11 i 13.

Również dla scenariuszy z wykorzystaniem narzędzia ApacheBench szkielet Vaadin osiągnął lepsze wyniki od szkieletu Laravel, wygrywając w każdym z pozostałych 4 scenariuszy.

6. Wnioski

Zaprezentowane wyniki badań dla podanych 13 scenariuszy jasno pokazują, że Laravel jest szkieletem mniej wydajnym. Vaadin uzyskał lepsze wyniki czasowe dla każdego prezentowanego scenariusza co jednoznacznie potwierdza postawioną na początku tezę mówiącą, że szkielet ten będzie szkieletem wydajniejszym.

Vaadin największą przewagę pokazuje w przypadku, gdy zaczęto zwiększać liczbę rekordów do 1000, 10000 w kwestii odczytu oraz do 100 podczas zapisu. Takie sytuacje, gdy trzeba było wygenerować stronę z co raz większą liczbą danych z bazy, nie stanowiły dla tego szkieletu problemu. Jest to spowodowane tym, że szkielet ten powstał głównie do zastosowań frontendowych, więc operacje związane ze stroną klienta są mocną stroną tego szablonu.

Laravel pomimo tego, że oferuje obsługę strony klienta, między innymi z użyciem szablonów Blade, to jest to szkielet głównie do obsługi części serwerowej, a więc generowanie wizualne co raz większej liczby rekordów, jak to miało miejsce podczas scenariusza 3 oraz 4, może już sprawiać temu szkieletowi problemy. Również zapis nawet 100 rekordów do bazy nie był problemem dla szkieletu Vaadin. Wpływ na to może mieć fakt, że Laravel nie jest szkieletem, który stawia przede wszystkim na wydajność, ale skupia się w dużej mierze na tym by programiście udostępnić możliwie jak najwięcej gotowych rozwiązań, czy modułów, które ułatwią mu pracę nad aplikacją ale jednocześnie mogą mieć mniej pozytywny wpływ na prędkość jej działania.

W przypadku testów z wykorzystaniem narzędzia ApacheBench Vaadin również okazał się nieporównywalnie wydajniejszym szkieletem. Nawet testy symulujące pobieranie zasobów przez 50 użytkowników w tym samym momencie nie okazały się wyzwaniem dla tego szkieletu. Wartym odnotowania jest również fakt, że im większa liczba użytkowników jest symulowana w danej chwili tym wyniki bardziej się różnią między pomiarami.

Literatura

[1] Popularność szkieletów PHP z roku 2018, https://www.excellentwebworld.com/, [20.10.2021].

[2] Popularność szkieletów Java z 2017, https://www.jrebel.com/blog/java-web-framework-usage-stats, [20.10.2021].

[3] P. Mincewicz, M. Plechawska-Wójcik, Performance and possibility analysis of Laravel tool dedicated to create modern web applications, Journal of Computer Sciences Institute 7 (2018) 108-115, https://doi.org/10.35784/jcsi.655.

[4] M. Laaziri, K. Benmoussa, S. Khoulji, M. L. Kerkeb, A Comparative study of PHP frameworks performance, Procedia Manufacturing 32 (2019) 864–871, https://doi.org/10.1016/j.promfg.2019.02.295.

[5] S. Jędrych, B. Jędruszak, B. Pańczyk, Comparative analysis of web applications development using JEE and PHP, Journal of Computer Sciences Institute 11 (2019) 86-90, https://doi.org/10.35784/jcsi.145.

[6] S. Memon, R. B. Palh, M. Memon, H. Memon, Performance Comparison of QEC Network based JAVA Application and Web based PHP Application, International Journal of Advanced Computer Science and Applications 9 (8) (2018), http://dx.doi.org/10.14569/IJACSA.2018.090870.

[7] Ł. Tomczyk, B. Pańczyk, Comparison of web applications development using Spring MVC and Vaadin, Journal of Computer Sciences Institute 6 (2018) 1-5, https://doi.org/10.35784/jcsi.631.

[8] Oficjalna dokumentacja narzędzie IntelliJ, https://www.jetbrains.com/idea/, [23.10.2021].

[9] Oficjalna dokumentacja narzędzia Maven, https://maven.apache.org/guides/, [23.10.2021].

[10] Oficjalna dokumentacja narzędzia PhpStorm, https://www.jetbrains.com/phpstorm/, [23.10.2021].

[11] Oficjalna dokumentacja narzędzia Composer, https://getcomposer.org/doc/, [23.10.2021].

[12] Oficjalna dokumentacja narzędzia MySQL, https://dev.mysql.com/doc/, [23.10.2021].

[13] Oficjalna dokumentacja narzędzie ApacheBench, https://httpd.apache.org/docs/2.4/programs/ab.html, [23.10.2021].

[14] Oficjalna dokumentacja narzędzia Chrome DevTools, https://developer.chrome.com/docs/devtools/, [23.10.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 40–45
Received: 24 October 2021
Accepted: 27 October 2021

Comparison of ASP.NET Core and Spring Boot ecosystems

Porównawnanie ekosystemów ASP.NET Core i Spring Boot

Petro Kopyl*, Teofil Rozaliuk* , Jakub Smołka

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

Abstract

The article describes a comparative analysis of the ASP.NET Core and Spring Boot framework ecosystems. The research was carried out on the basis of implemented two applications with identical functionality, which use the PostgreSQL database engine. In the implementation of the application, appropriate ORM (Object-Relational Mapping) tools were used to perform database operations, ie Spring Data and Entity Framework Core, technologies enabling the implementation of authentication and authorization (Spring Security and ASP.NET Core Identity) and several additional libraries that simplify the entire process of building the application. The criteria of comparison were the ease and intuitiveness of a given tool in the implementation of the application, the offered possibilities of the tools implementing authentication and authorization mechanisms and the efficiency of database operations. Based on the research, it was found that Spring Data technology is a faster tool than Entity Framework Core, while Spring Security, unlike Asp.Net Core Identity, is less integrated into the framework ecosystem, because it has a smaller set of ready-made solutions for database integration.

Keywords: Spring Boot; ASP.NET Core; comparative analysis.

Streszczenie

Artykuł opisuje analizę porównawczą ekosystemów szkieletów aplikacji ASP.NET Core oraz Spring Boot. Badania przeprowadzono w oparciu o dwie autorskie aplikacje o identycznej funkcjonalności, wykorzystujące silnik bazodanowy PostgreSQL. W implementacji aplikacji wykorzystano odpowiednie narzędzia ORM (ang. Object-Relational Mapping) do wykonywania operacji bazodanowych tj. Spring Data i Entity Framework Core, technologie umożliwiające implementację uwierzytelniania i autoryzacji (Spring Security i ASP.NETCore Identity) oraz kilka dodatkowych bibliotek, które upraszczają cały proces budowania aplikacji. Kryteriami porównania były łatwość i intuicyjność danych technologii w implementacji aplikacji, oferowane możliwości narzędzi implementujących mechanizmy uwierzytelniania i autoryzacji oraz wydajność operacji bazodanowych. Na podstawie wyników przeprowadzonych badań stwierdzono, że technologia Spring Data jest szybszym narzędziem od Entity Framework Core, natomiast Spring Security w odróżnieniu od Asp.Net Core Identity jest mniej zintegrowany w ekosystem frameworku, ponieważ posiada mniejszy zestaw gotowych rozwiązań do integracji z bazą danych.

Słowa kluczowe: Spring Boot; ASP.NET Core; analiza porównawcza.

*Corresponding author

Email address: **petro.kopyl@pollub.edu.pl** (P. Kopyl), **teofil.rozaliuk@pollub.edu.pl** (T. Rozaliuk)

1. Wstęp

W dzisiejszych czasach istnieje wiele różnych frameworków do budowy aplikacji serwerowych. Każdy z nich posiada swoje wady oraz zalety. W zależności od wymagań projektowanych systemów oraz innych czynników, takich jak doświadczenie programisty, koszty wsparcia projektu itd., wybór właściwego frameworku odgrywa istotną rolę przy implementacji systemu i pozwoli na stworzenie wydajnych i niezawodnych aplikacji webowych. Dodatkowo różne szkielety aplikacji zawierają swoje rozwiązania do implementacji poszczególnych funkcjonalności, wykorzystując gotowe biblioteki, które składają się na ekosystem tego frameworku. W danej pracy do badania wzięto dwa najbardziej popularne frameworki: Spring Boot [1] oraz ASP.NET Core [2].

W 2004 roku została wydana pierwsza wersja Spring Framework - framework z otwartym kodem źródłowym dla platformy Java. Dane narzędzie powstało jako alternatywa Enterprise JavaBeans. Spring nadaje programistom Java większą swobodę projektowania, ponadto zapewnia dobrze udokumentowane i łatwe w użyciu narzędzia do rozwiązywania problemów pojawiających się podczas tworzenia aplikacji. Spring można traktować jako zbiór mniejszych frameworków. Większość z nich może działać niezależnie od siebie, jednak razem zapewniają większą funkcjonalność. W 2016 roku dużą popularność osiągnął Spring Boot, ponieważ wprowadził automatyczną konfigurację projektów opartych o szkielet programistyczny Spring Framework, co znacznie przyspieszyło i uprościło pracę programistom.

ASP.NET Core to technologia tworzenia aplikacji webowych o otwartym kodzie wspierana przez Microsoft. Założona w 2016 roku jako następca technologii ASP.NET. Jest to modularna technologia wieloplatformowa (działa pod Windows, Linux oraz MacOS). Razem z językiem C# jest rozbudowanym, ciągle rozwijającym narzędziem dla programistów, które umożliwia łatwe wsparcie oraz migrację projektów do najnowszych wersji. Jedną

z głównych zalet jest duży zbiór narzędzi, ułatwiających implementację dla często się powtarzających scenariuszy biznesowych dla typowych aplikacji internetowych, takich jak np. uwierzytelnienie oraz praca z systemem bazodanowym.

2. Cel, teza i metody badań

Celem pracy jest przeprowadzenie analizy porównawczej ekosystemów dwóch szkieletów aplikacji Spring Boot oraz ASP.NET Core pod kątem wydajności wykonywania operacji bazodanowych, oraz wygody użycia narzędzi do implementacji systemów uwierzytelnienia i autoryzacji na podstawie prostej aplikacji internetowej zaimplementowanej w obu technologiach.

W niniejszym artykule zdefiniowano trzy tezy:

T1: Spring Data jest bardziej wydajnym narzędziem od Entity Framework Core przy współpracy z bazą danych PostgreSql.

T2: ASP.NET Core Identity jest bardziej elastycznym i wygodnym w użyciu narzędziem w porównywaniu do Spring Security.

Dla potwierdzenia wyżej wymienionych tez wykorzystano metodę badań opartą na analizie porównawczej obu frameworków. W tym celu stworzono dwie, funkcjonalnie identyczne, testowe aplikacje internetowe wykorzystując następujące narzędzia i technologie:

- szkielety programistyczne Spring Boot oraz ASP.NET Core;
- środowisko programistyczne IntelliJ IDEA oraz Visual Studio;
- serwer bazodanowy PostgreSQL [3];
- Maven - narzędzie do automatyzacji budowy projektów Java;
- Spring Security oraz ASP.NET Core Identity – narzędzia, które zapewnia mechanizmy budowania systemów uwierzytelniania i autoryzacji;
  - Spring Data [4] oraz Entity Framework Core [5] – technologie służące do realizacji warstwy dostępu do danych.

3. Przegląd literatury

Istnieje wiele prac, w których przeprowadzono analizę porównawczą różnych narzędzi do tworzenia aplikacji internetowych. W niniejszym artykule wybrano do porównania dwa najbardziej znaczące w dzisiejszych czasach frameworki do budowania aplikacji webowych Spring Boot oraz ASP.NET Core. Technologie często można spotkać w pracach badawczych w zestawie z innymi frameworkami, natomiast w literaturze nie znaleziono bezpośredniego ich porównania.

Przy budowaniu każdej nowoczesnej aplikacji internetowej bardzo ważnym elementem jest bezpieczeństwo. Ciekawym artykułem jest [6], w którym autor opisuje, jak ważne jest bezpieczeństwo, prezentuje przykłady przeprowadzonych różnego rodzaju ataków oraz porównuje rozwiązania, aby chronić swoje własne aplikacje przed nimi. Dodatkowo na podstawie prostej utworzonej przez autora aplikacji zostały przeprowadzone badania, w których wykonano ataki różnego rodzaju. Celem badań było wykluczenie lub potwierdzenie podatności aplikacji na ataki. W rezultacie okazało się, że nawet w prostej aplikacji internetowej, mogą być wykryte podatności na znane ataki oraz typowe błędy bezpieczeństwa tj. niewłaściwy sposób przechowywania hasła, lub błędy przy zarządzaniu sesją. Dla każdego z omówionych przypadków zostało przedstawione rozwiązanie mające na celu poprawę bezpieczeństwa.

Nie mniej ważnym czynnikiem podczas napisania aplikacji webowych jest poprawne wybranie i implementacje wzorcu architektonicznego. W większości aplikacji webowych używano wzorca MVC lub jego odmian. W artykule [7] autorzy porównują realizację tego wzorca w technologiach PHP oraz .Net Framework. Testowanie przeprowadzono na podstawie dwóch aplikacji o identycznych funkcjonalnościach, realizujących operacje dodawania, edycji oraz usuwania. Do napisania aplikacji w .Net Framework wykorzystano szkielet aplikacji webowych ASP.NET MVC oraz lokalną bazę danych SQL Server. Dla aplikacji napisanej w PHP użyto CakePHP oraz MySQL. Podczas testów okazało się, że .Net Framework w większości wskaźników zachowuje się lepiej od PHP, chociaż strony internetowe napisane w PHP są szybsze przy odbieraniu żądań i wysyłaniu odpowiedzi. W następnej pracy [8] wykonano porównanie kolejnych dwóch frameworków Spring Boot oraz JavaServer Faces na podstawie utworzonych w obu technologiach dwóch identycznych aplikacji internetowych. Jako kryteria dla analizy porównawczej brano pod uwagę ogólną strukturę aplikacji, podstawowe metryki kodu, efektywność wykonania operacji z danymi oraz efektywność ładowania zasobów przez przeglądarkę internetową. Struktura projektów aplikacji w obu technologiach jest podobna, natomiast bardziej wyraźne różnice widać przy porównaniu metryk kodu. Na podstawie pomiarów zauważono, że dla Spring Boot liczba linii kodu jest mniejsza niż w przypadku JSF, dlatego że Spring Boot zawiera wiele wbudowanych narzędzi, co z drugiej strony powoduje to, że rozmiar projektu jest znacznie większy.

W celu badania efektywności wykonywania operacji bazodanowych w obu aplikacjach zaimplementowano identyczne funkcjonalności: dodawanie, edycja oraz usuwanie rekordów. Do wykonywania pomiarów przygotowano kilka scenariuszy, gdzie każdy powtarzano 10 razy, a jako wynik pomiaru brano ich średni czas. Okazało się, że w przypadku pobierania, wyszukiwania oraz dodawania małej liczby rekordów Spring Boot jest znacznie szybszy od JSF, natomiast w przypadku dodawania większej liczby rekordów JSF jest nieznacznie bardziej efektywny. Jeśli chodzi o efektywność ładowania wszystkich niezbędnych zasobów aplikacji w przeglądarce internetowej, to tutaj też znacząco wygrywa Spring Boot. W rezultacie, na podstawie przeprowadzonych badań autorzy pracy, sformułowali wniosek, że

Spring Boot jest jednak korzystniejszym wyborem niż JavaServer Faces przy tworzeniu aplikacji internetowych JEE. Bardzo ciekawą pracą też jest praca [9] w której porównano od razu 4 różne szkielety programistyczne: Laravel, Rails, Django oraz Spring. Tutaj autorzy skupiają się na ważnych dla programisty kryteriach takich jak skalowalność aplikacji napisanych w wybranych technologiach, wsparcie wielojęzyczności, wsparcie urządzeń mobilnych oraz trendy biznesowe. W rezultacie okazało się, że wszystkie frameworki, które brały udział w porównaniu, spełniają wybrane kryteria, a różnice między nimi są nieznaczące. Liderem z nieznaczną przewagą został wybrany Spring Framework. Aplikacje napisane przy użyciu tego narzędzia są bardziej skalowalne. Spring ma też lepszą i bardziej rozbudowaną dokumentację, która wynika z dojrzałości technologii. Aby lepiej ocenić wybrane frameworki, autorzy również wprowadzili dodatkowe kryteria porównawcze, które głównie skupiają się na ocenie popularności tych frameworków wśród społeczności deweloperów oraz łatwość napisania stron internetowych w wybranych technologiach. Sporą przewagę uzyskał tutaj framework Django. Ma on niski próg wejścia i doskonale nadaje się do pisania dużych programów w małych ramach czasowych. Ważnym czynnikiem przemawiającym za tą technologią jest aktywne wsparcie przez społeczność programistów na platformach reddit, GitHub oraz Stack Overflow.

Oczywiście, należy pamiętać, że w epoce internetu, kiedy ciągle powstają nowe technologie, rozwijają się stare albo odwrotnie przestają być wpierane i używane, wszystko bardzo szybko się zmienia. Dlatego niektóre wyniki testów przeprowadzonych w poszczególnych wyżej opisanych pracach są już niestety nieaktualne.

4. Aplikacje testowe

Do porównania ekosystemów obu szkieletów aplikacji webowych została stworzona aplikacja testowa — messanger webowy. Do realizacji aplikacji wykorzystano wszystkie wymienione w punkcie 2. narzędzia. Dzięki nim zaimplementowano takie funkcjonalności jak typowe operacje CRUD (ang. Create, Read, Update, Delete) [10], uwierzytelnianie oraz autoryzacja użytkowników.

W obu rozwiązaniach wykorzystano serwer bazodanowy PostgreSQL. Schemat samej bazy danych jest przedstawiony na Rysunku 1.

5. Analiza porównawcza

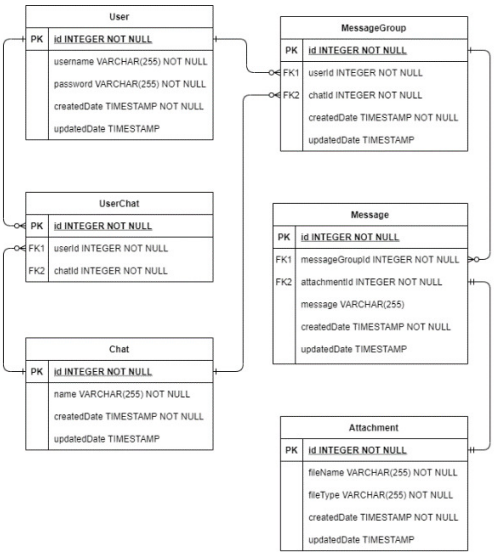
Aplikacje testowe porównano według następujących kryteriów:

- wydajność operacji bazodanowych,
- łatwość implementacji uwierzytelnienia oraz autoryzacji.

Wszystkie testy zostały przeprowadzone na komputerze o parametrach przedstawionych w Tabeli 1.

Tabela 1: Parametry sprzętowe komputera

Parametr	Wartość
System operacyjny	Windows 10
Procesor	Intel Core i7
Pamięć RAM	8 GB
Wersja PostgreSQL	13
Wersja SpringBoot	2.5.2
Wersja ASP.NET Core	5.0.7



Rysunek 1: Schemat bazy danych dla testowej aplikacji.

5.1. Wydajność operacji bazodanowych

W celu porównania wydajności wykonania operacji bazodanowych dla obu aplikacji zmierzono ich czasy. W badaniach brano pod uwagę trzy podstawowe operacje: dodawanie rekordów do bazy, ich pobieranie z bazy oraz usuwanie. Aby obiektywnie ocenić wydajność, dla każdej z nich przygotowano po 6 scenariuszy, które przedstawione są w Tabeli 2. Pomiary dla każdego scenariusza powtarzano 1000 razy i jako wynik przyjęto wartość średnią.

Operacje bazodanowe są dość szybkie, dla tego ich czasy wykonywania mierzono w milisekundach. Samo wywoływanie tych operacji zostało zrealizowane przy pomocy bibliotek ORM: w przypadku aplikacji w Spring Boot jest to Spring Data oraz Entity Framework Core dla aplikacji w ASP.NET Core. Czas mierzono bezpośrednio w kodzie. Brano pod uwagę wyłącznie czas wykonywania operacji bazodanowej, żadna logika biznesowa nie była uwzględniana. Przykład wykonywania pomiaru czasu dla operacji zapisu grupy powiadomień w aplikacji w Spring Boot przedstawiono na Listingu 1.

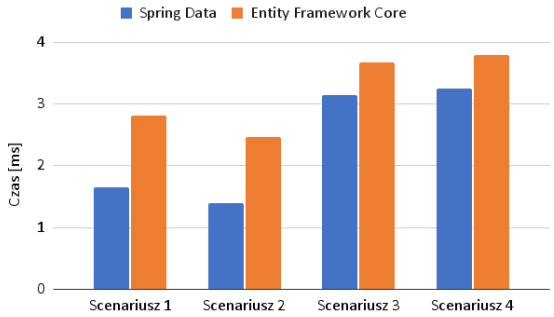
Listing 1: Przykład mierzenie czasu operacji bazodanowej

```
Long start = System.currentTimeMillis();
MessageGroup savedMessageGroup = messageGroupRepository.save(newMessageGroup);
Long end = System.currentTimeMillis();
operationService.create( timespan: end - start, operationType);
```

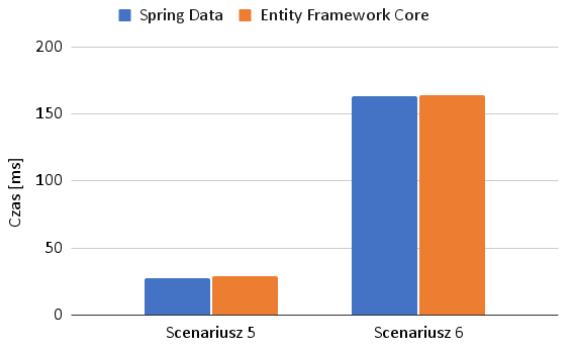
Tabela 2: Scenariusze testowe

Scenariusz	Opis
1	Zapis grupy powiadomień z jedną krótką (<10 znaków) wiadomością tekstową bez załącznika
2	Zapis grupy powiadomień z jedną długą (>200 znaków) wiadomością tekstową bez załącznika
3	Zapis grupy powiadomień z jednym załącznikiem bez wiadomości tekstowej
4	Zapis grupy powiadomień z pięcioma krótkimi wiadomościami tekstowymi bez załącznika
5	Zapis grupy powiadomień z pięcioma długimi wiadomościami tekstowymi bez załącznika
6	Zapis grupy powiadomień z pięcioma załącznikami bez wiadomości tekstowych
7	Pobranie grupy powiadomień z jedną krótką (<10 znaków) wiadomością tekstową bez załącznika
8	Pobranie grupy powiadomień z jedną długą (>200 znaków) wiadomością tekstową bez załącznika
9	Pobranie grupy powiadomień z jednym załącznikiem bez wiadomości tekstowej
10	Pobranie grupy powiadomień z pięcioma krótkimi wiadomościami tekstowymi bez załącznika
11	Pobranie grupy powiadomień z pięcioma długimi wiadomościami tekstowymi bez załącznika
12	Pobranie grupy powiadomień z pięcioma załącznikami bez wiadomości tekstowych
13	Usunięcie grupy powiadomień z jedną krótką (<10 znaków) wiadomością tekstową bez załącznika
14	Usunięcie grupy powiadomień z jedną długą (>200 znaków) wiadomością tekstową bez załącznika
15	Usunięcie grupy powiadomień z jednym załącznikiem bez wiadomości tekstowej
16	Usunięcie grupy powiadomień z pięcioma krótkimi wiadomościami tekstowymi bez załącznika
17	Usunięcie grupy powiadomień z pięcioma długimi wiadomościami tekstowymi bez załącznika
18	Usunięcie grupy powiadomień z pięcioma załącznikami bez wiadomości tekstowych

Na Rysunkach 2 oraz 3 przedstawiono wyniki dla obu aplikacji w postaci diagramów dla pierwszych sześciu scenariuszy, które dotyczyły operacji zapisu rekordów do bazy danych. Tutaj wyraźnie widać, jak w zależności od liczby przesyłanych wiadomości zmienia się czas wykonania operacji, im więcej przesyłanych danych, tym dłuższy czas ich zapisu. Ciekawe wyniki, na które należy zwrócić uwagę, są w przypadku scenariuszy o numerach 3 i 6, w których mierzono czas wykonania zapisu rekordów w postaci plików. W przypadku przesyłania pojedynczego pliku średni czas jego zapisu wynosi mniej niż 4 milisekundy, natomiast w przypadku przesyłania pięciu plików na raz ten czas wielokrotnie się zwiększa (ponad 150 milisekund). Natomiast ważny jest ten fakt, że dla wszystkich zdefiniowanych w artykule scenariuszy dotyczące operacji zapisu lepsze wyniki uzyskuje aplikacja zrealizowana w Spring Boot, w której użyto technologii Spring Data.



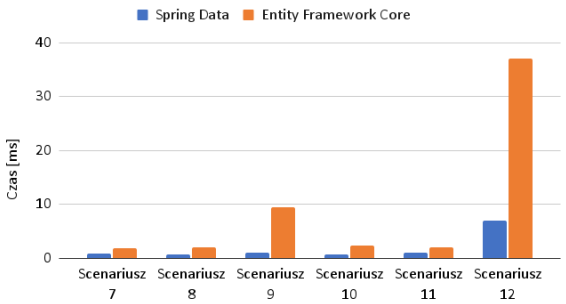
Rysunek 2: Średnie czasy pomiarów dla scenariuszy 1-4.



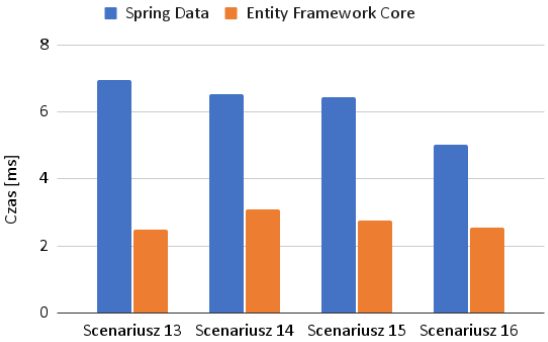
Rysunek 3: Średnie czasy pomiarów dla scenariuszy 5-6.

Kolejne 6 scenariuszy dotyczy operacji pobierania danych (Rysunek 4). Tutaj, tak samo, jak i w przypadku operacji zapisu, czas ich wykonania zależy od rozmiaru pobierających danych. W scenariuszach o numerach 9 oraz 12 dane są pobierane w postaci pliku. Widać, że czasy ich wykonania są znacznie dłuższe niż w przypadku innych scenariuszy, gdzie pobierane są dane w postaci wiadomości tekstowej. Jest to potwierdzeniem tego, że pliki są znacznie większe od samej treści tekstowej.

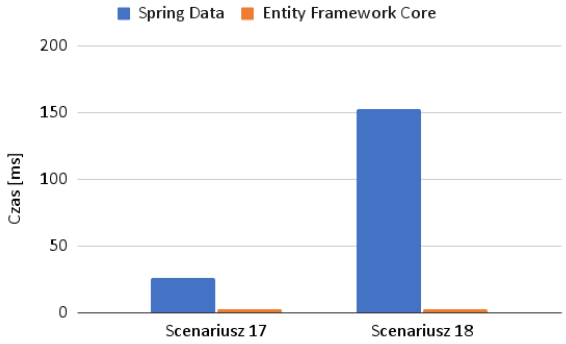
Ponadto, w przypadku technologii Entity Framework Core, operacje pobierania plików są wykonywane znacznie dłużej w porównaniu do technologii Spring Data. We wszystkich pozostałych scenariuszach, dotyczących pobierania danych, Spring Data też pokazuje lepsze rezultaty.



Rysunek 4: Średnie czasy pomiarów dla scenariuszy 7-12.



Rysunek 5: Średnie czasy pomiarów dla scenariuszy 13-16.



Rysunek 6: Średnie czasy pomiarów dla scenariuszy 17-18.

Ciekawe wyniki przedstawione są na dwóch ostatnich diagramach (Rysunek 5 oraz Rysunek 6). Prezentują one wyniki pomiarów dla scenariuszy, w których wykonywano operacje usuwania rekordów z bazy danych. W przypadku technologii Spring Data można zauważyć identyczną do poprzednich diagramów zależność czasu wykonania operacji od rozmiaru danych. Natomiast w przypadku technologii Entity Framework Core okazuje się, że przy zwiększeniu rozmiaru danych czas ten znacząco nie zmienia się, dla poszczególnych scenariuszy czas jest podobny, przy czym jest krótszy niż w przypadku technologii Spring Data. W tej części badań technologia Entity Framework Core uzyskuje lepsze wyniki pod kontem wydajnościowym.

5.2. Uwierzytelnianie oraz autoryzacja

Porównanie systemów autoryzacji, uwierzytelnienia oraz zarządzania użytkownikami zostało ocenione według następujących kryteriów:

- rozszerzalność narzędzia,
- wygoda pracy z tokenem JWT,
- współpraca z innymi narzędziami w obrębie wybranej technologii.

Do oceny rozszerzalności narzędzi Spring Security oraz Entity Framework Core wprowadzono 5 warunków:

- Czy dane narzędzie oferuje możliwość implementacji uwierzytelnienia i autoryzacji za pośrednictwem firm trzecich (Google lub Facebook)?
- Czy dane narzędzie pozwala na własną implementację mechanizmu hashowania haseł?
- Czy dane narzędzie oferuje możliwość przechowywania informacji o zalogowanym użytkowniku na różne sposoby (Cookies, JWT Token, baza danych)?
- Czy dane narzędzie pozwala na łatwy sposób rozbudowywania polityki autoryzacji, takie jak roli użytkownika?
- Czy pozwala na łatwą rozbudowę modelu użytkownika?

Za każdy spełniający warunek technologia uzyskiwała jeden punkt. W wyniku oceny wybranych technologii okazało się, że zarówno ASP.NET Core Identity, jak i Spring Security spełniają każdy z wyżej wymienionych warunków i dostają maksymalną liczbę punktów - 5. Wygodę pracy z tokenem JWT oceniono według skali od 1 do 5, gdzie:

- 1 - nie nadaje się do implementacji JWT,
- 2 - wymaga własnej implementacji JWT,
- 3 - pozwala na implementację JWT za pośrednictwem firm trzecich,
- 4 - oferuje własne dodatkowe narzędzia do implementacji oraz walidacji JWT,
- 5 - oferuje własne narzędzia do implementacji JWT oraz bierze na siebie realizację autoryzacji użytkownika za pomocą JWT.

Każdy z ocenianych frameworków oferuje własne narzędzia do pracy z JWT, ale biorąc pod uwagę to, że ASP.NET Core Identity posiada dodatkowe narzędzia, ułatwiające implementację uwierzytelnienia i autoryzacji za pomocą JWT, oceny tych frameworków są następujące: ASP.NET Core Identity uzyskuje 5 punktów, natomiast Spring Security – 4 punkty.

Ostatnie kryterium dotyczyło współpracy testowanych systemów do implementacji uwierzytelnienia oraz autoryzacji z innymi narzędziami w obrębie wybranej technologii. To kryterium zostało ocenione według skali od 1 do 5, gdzie:

- 1 - wymaga własnej implementacji integracji pomiędzy systemem uwierzytelnienia i autoryzacji oraz szkieletem aplikacji webowej,
- 2 - współpracuje ze szkieletem aplikacji webowej przy pomocy rozwiązań firm trzecich,
- 3 - współpracuje ze szkieletem aplikacji webowej za pomocą gotowych narzędzi,
- 4 - współpracuje ze szkieletem aplikacji webowej za pomocą gotowych narzędzi oraz z rozwiązaniem ORM za pośrednictwem firm trzecich lub za pomocą własnej implementacji takiej integracji,

- 5 - współpracuje ze szkieletem aplikacji webowej oraz z rozwiązaniem ORM za pomocą gotowych narzędzi.

Według tej skali Spring Security uzyskał 4 punkty, natomiast ASP.NET Core Identity uzyskał 5 punktów. W Tabeli 3 przedstawiono wyniki oceniania wybranych narzędzi.

Tabela 3: Wyniki oceniania framewroków uwierzytelnienia i autoryzacji Spring Security oraz ASP.NET Core Identity

Kryterium	Spring Security	ASP.NET Core Identity
Rozszerzalność narzędzia	5	5
Wygodność pracy z tokenem JWT	4	5
Współpraca z innymi narzędziami w obrębie wybranej technologii	4	5

6. Wnioski

W artykule przeprowadzono analizę porównawczą ekosystemów dwóch najbardziej popularnych szkieletów aplikacji ASP.NET Core oraz Spring Boot. Do przeprowadzenia badań stworzono dwie aplikacje testowe typu REST o identycznej funkcjonalności messengera webowego. Aplikacje testowe zostały stworzone w różnych stosach technologicznych. Pierwsza aplikacja napisana została w języku Java. Jako szkielet wybrano Spring Boot. Do komunikacji z bazą danych użyto narzędzia Spring Data, do realizacji uwierzytelnienia oraz autoryzacji użytkownika — Spring Security. Drugą aplikację testową napisano w języku C# za pomocą frameworku ASP.NET Core. Tutaj do komunikacji z bazą danych służy Entity Framework Core, natomiast za realizację mechanizmów uwierzytelnienia oraz autoryzacji odpowiada ASP.NET Core Identity.

Podczas analizy badawczej narzędzi do komunikacji z bazą danych brano pod uwagę czas wykonywania operacji CRUD. W celu obiektywnej oceny poszczególnych operacji zdefiniowano kilka scenariuszy. Po przetestowaniu każdego ustalono, że Spring Data w operacjach zapisu oraz odczytu danych jest szybszy od Enitity Framework Core. Natomiast Entity Framework Core jest szybszy przy usuwaniu rekordów. Uwzględniając to, że Spring Data w większości scenariuszach uzyskał lepsze rezultaty, można stwierdzić, że jest bardziej wydajnym narzędziem od Entity Framework. Teza *T1* została potwierdzona.

Systemy uwierzytelnienia oraz autoryzacji subiektywnie oceniono według następujących kryteriów: rozszerzalność narzędzia, wygoda pracy z tokenem JWT oraz współpraca z innymi narzędziami w obrębie wybranej technologii. Każdy ze szkieletów aplikacji dostał wysokie oceny i w całości spełnia wybrane kryteria. Warto zaznaczyć, że ASP.NET Core Identity jest bardziej zintegrowany w ekosystem szkieletu, w obrębie którego pracuje. ASP.NET Core Identity posiada dodatkowe mechanizmy do integracji z Entity Framework Core, a także dodatkowe już gotowe klasy do pracy z tokenem JWT. Z tego powodu ASP.NET Core Identity dostał wyższą ocenę w takich kryteriach jak wygoda pracy z tokenem JWT oraz współpraca z innymi narzędziami w obrębie wybranej technologii. Na podstawie tego można stwierdzić, że teza *T2* została potwierdzona — Asp.Net Core Identity jest bardziej elastycznym i wygodnym w użyciu narzędziem w porównywaniu do Spring Security.

Literatura

[1] Szkielet programistyczny Spring Boot, https://spring.io/projects/spring-boot , [26.01.2021].

[2] J. Ciliberti, ASP. NET Core Recipes: A Problem-Solution Approach, Apress, New York, 2017.

[3] Serwer bazodanowy PostgreSQL, https://www.postgresql.org/ , [22.10.2021].

[4] Narzędzie Spring Data, https://spring.io/projects/spring-boot , [22.10.2021].

[5] J. Lerman, R. Miller, Programming Entity Framework: Code First, O'Reilly Media, Sebastopol, 2011.

[6] M. Furtak, Bezpieczeństwo aplikacji internetowych, JCSI 3 (2017) 1-5.

[7] M. Jailia, A. Kumar, M. Agarwal and I. Sinha, Behavior of MVC (Model View Controller) based Web Application developed in PHP and .NET framework, 2016 International Conference on ICT in Business Industry & Government (ICTBIG) (2016) 1-5.

[8] M. M. Kizeweter, B. Pańczyk, Porównanie technologii tworzenia aplikacji internetowych JEE na przykładzie JavaServer Faces i Spring Boot, JCSI 3 (2017) 28-32.

[9] M. Kaluža, M. Kalanj, A comparison of back-end frameworks for web application development, Computer Science 7 (2019) 317-332.

[10] Operacje bazodanowe typu CRUD, https://www.codecademy.com/articles/what-is-crud , [19.10.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 46–50
Received: 26 October 2021
Accepted: 17 December 2021

Tools for modeling and simulating business processes - a comparative analysis

Analiza porównawcza narzędzi do modelowania i symulacji procesów biznesowych

Dominik Lipski* , Radosław Lipski*

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

Abstract

The purpose of this article is to compare selected tools for modeling and simulating business processes. The comparative study concerned Bizagi, Adonis, Bonita Studio and Visual Paradigm tools. The analysis has the aim to identify the best tool according to the criteria. The article explains the concepts related to business processes and analyzes the existing literature. Research scenarios containing evaluation criteria will be created for the analysis. The conducted research indicated that the best tool among the considered: Bizagi, Adonis, Bonita Studio, Visual Paradigm is Bizagi.

Keywords: business process; process modeling; process simulation

Streszczenie

Celem artykułu jest porównanie wybranych narzędzi do modelowania i symulacji procesów biznesowych. Badanie porównawcze dotyczyło narzędzi Bizagi, Adonis, Bonita Studio i Visual Paradigm. Analiza ma na celu wskazanie najlepszego narzędzia według określonych kryteriów. W artykule wyjaśniono pojęcia związane z procesami biznesowymi i przeanalizowano dotychczasową literaturę. Do analizy zostaną stworzone scenariusze badawcze zawierające kryteria oceny. Przeprowadzone badania wskazały, że najlepszym narzędziem spośród rozpatrywanych: Bizagi, Adonis, Bonita Studio, Visual Paradigm jest Bizagi.

Słowa kluczowe: proces biznesowy; modelowanie procesów; symulacja procesów

*Corresponding author

Email address: dominik.lipski@pollub.edu.pl (D. Lipski), radoslaw.lipski@pollub.edu.pl (R.Lipski)

1. Wstęp

Modelowanie i symulacja procesów biznesowych w dzisiejszych czasach jest najważniejszym działaniem związanym z optymalizacją pracy oraz zarządzaniem każdym z przedsiębiorstw, niezależnie od branży, w której działa.

Obecnie prawie we wszystkich firmach systemy wspomagające kwestie organizacyjne oraz systemy informatyczne pomagają w obsłudze procesów biznesowych. Pracownicy firmy odpowiedzialni za usprawnienie jej działalności wewnętrznej oraz zewnętrznej muszą przeprowadzać szereg różnych analiz i symulacji. Do przeprowadzenia tych analiz i symulacji konieczna jest znajomość odpowiednich metod i narzędzi, których celem jest ulepszenie działalności firmy. Liczba tych narzędzi ciągle rośnie, a wybranie najlepszej aplikacji, która spełniałaby oczekiwania danej firmy jest niemałym wyzwaniem. Na rynku jest spore zróżnicowanie aplikacji począwszy od prostych opierających się na tworzeniu podstawowych diagramów po te bardziej zaawansowane, zawierające funkcje takie jak analiza czy optymalizacja.

Przedmiotem naszych badań są narzędzia wykorzystywane do analizy i symulacji procesów biznesowych.

2. Zarządzanie procesami biznesowymi

2.1. Pojęcia

Proces biznesowy to seria zadań lub działań, które są ze sobą powiązane i rozwiązują określony problem lub wiodą do osiągnięcia określonego efektu.

Symulacja procesów biznesowych to sposób analizy i testowania procesów w wirtualnym środowisku. Zastosowanie narzędzi, które umożliwiają takie symulacje oszczędza wielu problemów, które mogłyby się pojawić podczas testowania na rzeczywistym środowisku. Korzystanie z środowiska wirtualnego pozwala na uniknięcie kosztów obniżenia produktywności lub przerw w pracy, a procesy można badać z wykorzystaniem wielu zmiennych bez ograniczeń [1].

Modelowanie procesów biznesowych to graficzne przedstawienie procesów biznesowych przedsiębiorstwa w sposób możliwy do analizy, ulepszenia i zautomatyzowania. Modele te są przedstawione jako diagramy lub schematy blokowe. Dzięki modelowaniu mamy możliwość poznania procesów od początku do końca co gwarantuje lepsze zrozumienie konkretnych etapów. Głównym celem modelowania procesów jest ich doskonalenie poprzez na przykład zwiększenie szybkości wykonywania procesów lub obniżenie

kosztów, które poszczególne procesy generują. Istnieje wiele technik modelowania procesów biznesowych. Najpopularniejsze to: BPMN (Business Process Model and Notation), schematy blokowe, diagramy przepływu danych DFD [2].

3. Przegląd literatury

Pierwszy artykuł [3] porównuje wybrane narzędzia do symulacji procesów biznesowych. Analiza została dokonana na podstawie kryteriów takich jak: wydajność aplikacji, testowalność, aspekt wizualny czy możliwości eksperymentalne. W pracy tej zostały wykorzystane 2 metodologie DEX (Decision EXpert) i QQ (Qualitative to Quantitative). Ten artykuł przedstawia w jaki sposób dokonać analizy w przypadku złożonych kryteriów. Wyniki zostały dobrze opisane i przedstawione na wielowymiarowym wykresie.

Artykuł [4] przedstawia analizę the Business Process Simulation Interchange Standard (BPSim). Jest to standard wykorzystywany do wymiany modeli pomiędzy różnymi narzędziami do modelowania i symulacji procesów biznesowych. BPSim jest rozszerzeniem BPMN stosowanym do symulacji procesów. Artykuł przedstawia możliwości i ograniczenia tego standardu.

W artykule [5] zostały porównane dwie metody modelowania procesów biznesowych. Zaprezentowane zostało porównanie dwóch metod modelowania: sieci Petriego oraz BPMN. Metody porównano za pomocą czterech kryteriów: wizji rozwoju, wsparcia narzędzi, popularności, dostępności. Na podstawie tych kryteriów analiza wskazała metodę BPMN jako tę lepszą. Metoda BPMN uzyskała ocenę pozytywną w każdym z czterech kryteriów. Sieci Petriego uzyskało ocenę negatywną w kryteriach rozwoju i dostępności.

Publikacja [6] zawiera analizę porównawczą stosunkowo popularnych modeli procesów biznesowych skupiając się na narzędziach. W artykule porównano wiele różnych narzędzi. Analiza tych narzędzi oparta była o kryteria takie jak na przykład zdolność przepływu danych, możliwość symulacji czy łatwość obsługi.

Kolejny artykuł [7] opiera się na narzędziach, które oprócz funkcji modelowania rozwinęły też możliwość symulacji stworzonych procesów biznesowych. Poddano analizie możliwości obsługi symulacji przez te narzędzia. Dodatkowo przedstawiono podstawowe funkcjonalności każdego z poddanych analizie narzędzi. W tej publikacji znajdziemy pięć aktualnie najpopularniejszych narzędzi do symulacji procesów dostępnych na rynku.

Autorzy publikacji "Comparative Analysis of Business Process Modelling Tools for Compliance Management Support" [8] z 2017 roku dokonali porównania narzędzi modelowania procesów biznesowych wspierających zautomatyzowane zarządzanie zgodnością w organizacjach na przykład zgodność z prawem czy zewnętrzne regulacje. W pracy tej została przeprowadzona analiza literatury związanej z zarządzaniem zgodnością. Autorzy zdecydowali się na wybór kilkunastu narzędzi w wersjach darmowych, dla uczniów lub wersji trial. Kryteria oceny zostały czytelnie przedstawione w formie tabeli razem z przyczynami dlaczego dane kryterium jest ważne dla przeprowadzonych badań. Twórcy tego artykułu podsumowali wyniki przeprowadzonej analizy w kilku punktach. Według nich narzędzia open source spełniają mniej kryteriów niż narzędzia komercyjne. Najlepsze okazały się CPN Tool oraz Intalio BPMS. Większość kryteriów jest obsługiwana przez Enterprise Architect, Adonis i Visual Paradigm. Jednak Enterprise Architect nie ma funkcji w zakresie zarządzania zgodnością procesów biznesowych, ale rozwiązuje problem zgodności architektury korporacyjnej. Narzędziu Adonis brakuje jedynie wsparcia zakresu notacji i łączenia obiektów zewnętrznych. Visual Paradigm nie obsługuje zarządzania zgodnością, obsługuje też mniej notacji.

Wsparcie symulowania procesów biznesowych przez narzędzia BPM zostało przeanalizowane w artykule "Process Simulation Support in BPM Tools: The Case of BPMN" [9] z 2015 roku. Już we wstępie autorzy zauważają, że procesy które zostały opisane za pomocą BPMN można zasymulować aby znaleźć ich optymalną formę. W dalszej części artykułu przedstawiono pojęcie zarządzania procesami biznesowymi, po którym następuje bardzo zwięzłe wprowadzenie do języka powszechnie używanego do graficznego przedstawiania procesów - BPMN. Autorzy zebrali ważne elementy procesu biznesowego, dzięki którym możliwe będzie przeprowadzenie symulacji. Według nich należy uwzględnić to, że proces może być uruchamiany wiele razy w określonym czasie, dlatego należy znać sposób jego wyzwalania, co może być przedstawione za pomocą rozkładu probabilistycznego. Każda czynność posiada zasoby do jej wykonania, takie jak ludzie czy maszyny, istotna jest ich ilość użyta w każdym działaniu danego procesu. Zauważono również, że czas na wykonanie zadania nie zawsze jest przeznaczony w 100%. Szczególnie to widać przy zasobach ludzkich, które mają tendencję do dzielenia uwagi pomiędzy zadania, dlatego należy to uwzględnić podczas symulowania. Zasoby poza ilością czasu da się również scharakteryzować określając ich dostępność i priorytet. Nie zawsze ludzie mają pełną dostępność, a maszynom czasami potrzebna jest przerwa na serwis lub dostosowanie ich do kolejnego zadania. W części badawczej artykułu przeanalizowano pięć narzędzi skupiając się głównie na symulowaniu.

4. Analizowane narzędzia

Analizowane programy zostały wyszukane za pomocą internetu według opisów i przeglądu narzędzi umożliwiających zarządzanie procesami. Przy wyborze programów znaczenie miały oceny i doświadczenia użytkowników popularność.

Bizagi Modeler i Bizagi Studio to darmowe narzędzia do tworzenia diagramów i symulacji procesów w standardowym formacie BPMN. Umożliwiają publikowanie utworzonych diagramów w wielu formatach oraz ich współdzielenie w grupie. Dają możliwość eliminacji wąskich gardeł i optymalizacji procesu poprzez zastosowanie symulacji. Narzędzia Bizagi Modeler i Bizagi Studio oferują przechowywanie procesów w chmurze. Zapewniają również importowanie istniejących diagramów z takich narzędzi jak Microsoft Vision, IBM Bluewoks, XPDL oraz tworzenie dokumentacji w formacie PDF, Microsoft Word, Microsoft Excel czy Wiki. Praca w środowisku modelowania polega na przeciąganiu i upuszczaniu poszczególnych elementów co sprawia, że diagramy są proste w budowaniu [10]. Dostępne podczas instalacji dostępnych jest kilka wersji językowych, jednak brakuje języka polskiego.

Adonis to darmowe narzędzie BPM oparte na chmurze ADONIS:CE. Umożliwia tworzenie diagramów BPMN, budowanie procesów biznesowych oraz ich analizę i optymalizację. Zapewnia eksport do najpopularniejszych formatów – PDF, PNG, Excel. Dodawanie elementów odbywa się za pomocą funkcji przeciągnij i upuść. Dodatkowym ułatwieniem dla mniej zaawansowanych osób jest automatyczna walidacja BPMN [11]. Dostępna jest polska wersja językowa wraz z funkcją pomocy napisaną w języku polskim.

Bonita Studio to program, który pozwala użytkownikowi tworzyć graficzne modele procesów biznesowych z wykorzystaniem standardu BPMN. Po utworzeniu procesu na tablicy w postaci graficznej reprezentacji można go łatwo przetestować, uruchamiając sekwencję symulacji. Składa się z dwóch komponentów - Bonita Studio i Bonita Runtime. Bonita Studio zawiera wszystkie elementy niezbędne do utworzenia określonych procesów, modeli danych lub innych widoków. Bonita Runtime umożliwia wdrożenie i testowanie poszczególnych procesów utworzonych w Bonita Studio [12]. Podczas instalacji do wyboru jest pięć wersji językowych - bez języka polskiego.

Visual Paradigm to narzędzie obsługujące standard UML 2 i standardową notację modelowania procesów biznesowych BPMN. Zapewnia generowanie raportów i daje możliwość odtwarzania diagramów z kodu. Zapewnia duży zbiór komponentów do tworzenia procesów ale też i daje możliwość utworzenia własnych. Umożliwia organizacjom poprawę biznesową i informatyczną, a z ich usług korzysta ponad 320 tysięcy użytkowników - od małych firm do dużych organizacji na całym świecie [13]. Dostępny jest wyłącznie język angielski.

5. Cel i założenia badań

Głównym celem badań jest przeprowadzenie analizy porównawczej narzędzi do modelowania i symulacji procesów biznesowych. Wynikiem badań jest wybór najlepszego i najkorzystniejszego rozwiązania dla użytkownika. Ocena polega na zbadaniu, które z narzędzi najlepiej spełnia określone kryteria.

5.1. Hipotezy i problemy badawcze

Przygotowano następujące pytania badawcze:

- Które z narzędzi zapewnia najwięcej funkcjonalności dostępnych w darmowej wersji?
- Które z narzędzi posiada najbardziej przyjazny i intuicyjny interfejs użytkownika?
- Które z narzędzi zapewnia najlepszy sposób modelowania i symulacji procesów?
- Które z narzędzi jest najprostsze w instalacji i obsłudze?

Na podstawie postawionych pytań badawczych sformułowano następującą hipotezę badawczą:
Narzędzia Bizagi Modeler i Bizagi Studio są najlepszym rozwiązaniem dla użytkowników firmowych i studentów.

6. Kryteria oceny i metody badań

W celu oceny, porównania i próby wyselekcjonowania najlepszego z wcześniej wybranych narzędzi do analizy i symulacji procesów biznesowych w pracy wykorzystano dwie metody badawcze - wielokryterialna analiza porównawcza oraz eksperyment naukowy. Analiza porównawcza była przeprowadzona według następujących kryteriów:

1. Modelowanie
   1.1. Dostępność elementów i zgodność z BPMN - 0-5 pkt
   1.2. Intuicyjność i podpowiedzi kolejnych kroków - 0-3 pkt
   1.3. Formatowanie i układ diagramów - 0-3 pkt
   1.4. Walidacja poprawności diagramów - 0-3 pkt
2. Symulowanie
   2.1. Dostępność parametrów - 0-5 pkt
   2.2. Prezentacja wyników - 0-3 pkt
   2.3. Walidacja danych - 0-3 pkt
3. Inne kryteria
   3.1. Eksport i import danych - 0-3 pkt
   3.2. Ergonomia i stabilność aplikacji - 0-3 pkt
   3.3. Darmowa wersja i jej funkcjonalności - 0-3 pkt
   3.4. Dostępność pomocy - 0-3 pkt
   3.5. Łatwość rejestracji lub instalacji - 0-3 pkt

W badaniu tym zostały uwzględnione kryteria dotyczące nie tylko modelowania i symulowania procesów biznesowych ale również uwzględniające dostępność wybranych narzędzi, ich ergonomię oraz możliwości wymiany między nimi danych. Najbardziej znaczącym kryterium jest modelowanie procesów. Tworzenie modeli pozwala w ustaleniu sposobu działania danego przedsiębiorstwa i przedstawienie tego w sposób graficzny. Istotne jest wykorzystanie pewnych standardów do opisania całego procesu, dlatego w ocenie narzędzi należy zweryfikować czy oferują one zrozumiałą dla osoby modelującej notację BPMN.

Poza wcześniej wymienionymi kryteriami również ważna jest dostępność narzędzi. Dla celów edukacyjnych czy zapoznawczych z modelowaniem

i symulowaniem procesów biznesowych warto wziąć pod uwagę darmowe wersje narzędzi. Często mogą one mieć ograniczone funkcje lub być dostępne tylko przez określony czas. Wpływ na ocenę końcową narzędzi mają również dostępność pomocy, możliwości wyboru języka i łatwość rejestracji lub instalacji.

Eksperyment naukowy miał na celu stworzenie przykładowego modelu procesów. Został on wykorzystany przez autorów do przeanalizowania możliwości symulacji procesów poszczególnych narzędzi. Sprawdzono sposób przedstawiania wyników i ilość dostępnych parametrów. Uwagę zwrócono też na dostępność jednostek do wyboru dla parametrów.

Zespołem oceniającym byli autorzy pracy. Są to osoby w wieku 24 lat o zaawansowanej wiedzy na temat obsługi komputera i jego oprogramowania. Narzędzia, które oceniali nie były przez nich wcześniej wykorzystywane.

7. Wyniki badań

Pierwszą z czynności był wybór poszczególnych narzędzi na podstawie rankingów internetowych i analizowanej literatury. Wybrane narzędzia i ich wersje przedstawiono w tabeli 1. Następnie pobranie, instalacja i wstępna konfiguracja poszczególnych narzędzi. Gdy instalacja i konfiguracja narzędzi przebiegła pomyślnie nastąpił etap analizy porównawczej wybranych programów do modelowania i symulacji procesów biznesowych.

Tabela 1: Wersje wybranych narzędzi

	Nazwa	Wersja	Producent
1	Bizagi Modeler, Bizagi Studio	3.8.0. 11.2.5	Bizagi
2	Adonis	12	Adonis
3	Bonita Studio, Bonita Runtime	2021.1	BonitaSoft
4	Visual Paradigm	16.4	Visual Paradigm

Jak widać w tabeli 2. najlepszym rozwiązaniem według autorów pracy okazała się aplikacja Bizagi uzyskując najwięcej punktów. Narzędzie to uzyskało najwięcej punktów w kategoriach modelowania i symulacji procesów, w innych kryteriach okazało się słabsze o 1 punkt od Visual Paradigm i Adonis. Drugie miejsce zajęło narzędzie Adonis, rozwiązanie dostępne online nie wymagające instalacji idealne dla małych projektów. Kolejne miejsce to Visual Paradigm, który okazał się słabszym rozwiązaniem od dwóch poprzednich głównie w kwestii modelowania. Najgorszym rozwiązaniem został program Bonita Studio, który oferuje bardzo ograniczone możliwości. Symulacja w tym narzędziu nie była możliwa do przeprowadzenia, a w pozostałych dwóch kategoriach narzędzie znacznie odbiega w ocenie od konkurencji.

Tabela 2: Kryteria oceny wybranych narzędzi

Kryterium	Bizagi (pkt)	Adonis (pkt)	Bonita Studio (pkt)	Visual Paradigm (pkt)
1.1.	4	4	2	3
1.2.	3	3	2	2
1.3.	2	3	1	2
1.4.	3	1	2	1
2.1.	4	2	-	2
2.2.	2	1	-	1
2.3.	2	2	-	2
3.1.	2	2	3	3
3.2.	3	3	1	2
3.3.	3	2	1	3
3.4.	3	3	2	3
3.5.	1	3	2	2
Suma pkt.	**32**	29	16	26

8. Wnioski

Analiza wyników pokazuje, że oceniane rozwiązania mają mniejsze lub większe braki w funkcjonalności. Postawiona w pracy hipoteza: *Narzędzia Bizagi Modeler i Bizagi Studio są najlepszym rozwiązaniem dla użytkowników firmowych i studentów* potwierdziła się. Na podstawie przeprowadzonego eksperymentu naukowego i kryteriów oceniania Bizagi posiada w bezpłatnej wersji najbardziej przydatne funkcje oraz zasoby elementów. Osoba, która nie ma doświadczenia w modelowaniu procesów biznesowych z pewnością odnajdzie się w narzędziach Bizagi, dzięki przyjaznemu interfejsowi oraz pomocy dostępnej zarówno w aplikacjach jak i platformach online udostępnianych przez producenta.

Na wyróżnienie zasługuje narzędzie chmurowe Adonis, które przy małych projektach dobrze by się sprawowało, między innymi dzięki łatwości rejestracji oraz dostępności przykładowych modeli. Przy większych aplikacjach brakowałoby jednak mu funkcjonalności i płynności w działaniu.

Lista narzędzi oferujących modelowanie i symulację procesów cały czas zwiększa się o nowe rozwiązania. Aplikacje zaprezentowane w tej pracy należały do najpopularniejszych rozwiązań dostępnych na rynku.

Literatura

[1] Symulacja procesów biznesowych - opis, https://kissflow.com/workflow/bpm/business-process-simulation , [23.09.2021].

[2] Modelowanie procesów biznesowych – zastosowanie oraz opis technik, https://kissflow.com/bpm/business-process-modeling/, [23.09.2021].

[3] N. Damij, P. Boškosk, M. Bohanec, B. M. Boshkoska, Ranking of Business Process simulation software Tools with DEX/qq hierarchical Decision Model, PLOS ONE, (2016), https://doi.org/10.1371/journal.pone.0148391.

[4] R. Laue, C. Müller, The business process simulation standard (BPSim): chances and limits, Germany, (2016), 413-418, http://dx.doi.org/10.7148/2016-0413.

[5] I. Chomiak-Orsa, A. Kołtonowska, Modelowanie procesów biznesowych z wykorzystaniem sieci Petriego i BPMN - Próba oceny metod, Informatyka Ekonomiczna, Wrocław, (2018), 9-18, http://dx.doi.org/10.15611/ie.2018.2.01.

[6] R. C. Papademetriou, D. A. Karras, Towards a Thorough Evaluation Framework of Software. Tools Suitable for Small and Medium Size Enterprises. Focusing on Modelling and Simulating Business Processes, Rodos, (2017), 161-182, https://doi.org/10.1007/978-3-319-57222-2_8.

[7] J. L. Pereira, A. P. Freitas, Simulation of BPMN Process Models: Current BPM. Tools Capabilities, International Journal for Quality Research 13(4), Portugal, (2016), 783-796, http://dx.doi.org/10.24874/IJQR13.04-02.

[8] R. Koncevičs, L. Peņicina, A. Gaidukovs, Comparative Analysis of Business Process Modelling Tools for Compliance Management Support, Latvia, (2017), 22-27, https://doi.org/10.1515/acss-2017-0003.

[9] A. Freitas, P. Pereira, J. L. Mota, Process Simulation Support in BPM Tools: The Case of BPMN, Proceedings of 2100 Projects Association Joint Conferences, (2015).

[10] Bizagi - opis narzędzia, https://en.wikipedia.org/wiki/Bizagi, [26.09.2021].

[11] Bonita Studio - opis narzędzia, https://en.wikipedia.org/wiki/Bonita/BPM, [26.09.2021].

[12] Adonis - opis narzędzia, https://en.wikipedia.org/wiki/ADONIS/(software), [26.09.2021].

[13] Visual Paradigm - opis narzędzia, https://www.visual-paradigm.com/aboutus/, [26.09.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 51–58
Received: 28 October 2021
Accepted: 1 November 2021

Comparative analysis of interface sketch design tools in the context of User Experience

Analiza porównawcza narzędzi do projektowania szkiców interfejsów w kontekście User Experience

Agnieszka Glinka*, Edyta Kowalczyk*, Tomasz Szymczyk

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

Abstract

The design stage plays a significant role in software development. Making skeleton models helps to understand the specification and requirements of the product. Design tools, as well as any other tools, should focus on User Experience. The aim of the publication is to compare tools for sketching interfaces taking into account User Experience. The main criterion for the evaluation of the tools was a test consisting in measuring the time and distance of a computer mouse during the execution of a given application sketch in 8 selected tools. Then, the respondents assessed the possibilities and experiences resulting from working in a given tool. This was supplemented by the assessment of the technical criteria. Ultimately, the best and the worst tools were selected according to the research subgroups. The level of education has been proven to influence the evaluation of interface sketch design tools.

Keywords: sketches of interfaces; design tools; designing; User Experience

Streszczenie

Etap projektowania odgrywa znaczącą rolę w wytwarzaniu oprogramowania. Wykonanie modeli szkieletowych pomaga zrozumieć specyfikację oraz wymagania produktu. Narzędzia przeznaczone do projektowania, jak również każde inne, powinno skupiać się na doświadczeniach użytkownika, czyli User Experience. Celem publikacji jest porównanie narzędzi do szkicowania interfejsów z uwzględnieniem User Experience. Głównym kryterium oceny narzędzi było badanie polegające na pomiarze czasu oraz przebytej drogi myszy komputerowej podczas wykonania zadanego szkicu aplikacji w 8 wybranych narzędziach. Następnie badani oceniali możliwości oraz doświadczenia wynikające z pracy w danym narzędziu. Uzupełnieniem była ocena kryteriów technicznych. Ostatecznie wyłoniono narzędzia najlepsze oraz najgorsze według podgrup badawczych. Udowodniono, że poziom wykształcenia wpływa na ocenę narzędzi do projektowania szkiców interfejsów.

Słowa kluczowe: szkice interfejsów; narzędzia do projektowania; projektowanie; User Experience

*Corresponding author

Email address: **agnieszkasia.glinka@gmail.com** (A. Glinka), **edytaxkowalczyk@gmail.com** (E. Kowalczyk)

1. Wstęp

Projektowanie to znaczący etap wytwarzania oprogramowania. Ułatwia zrozumienie wymagań oraz wizji klientów dzięki czemu możliwe jest uniknięcie wielu nieporozumień i trudności w ukończeniu produktu. Właściwie wykonany plan projektu często zawiera szkic budowanego systemu. Na rynku istnieje wiele narzędzi służących do ich tworzenia, niezależnie czy są to modele szkieletowe, makiety czy prototypy. Jest wiele cech, które warto wziąć pod uwagę podczas wybierania odpowiedniego oprogramowania, jednakże wybór często uzależniony jest od osobistych preferencji lub wymagań, które ma spełniać gotowy produkt.

W niniejszej pracy przedstawiono globalne spojrzenie na narzędzia do tworzenia szkiców interfejsów. Badaniu poddano 8 narzędzi. Po przedstawieniu wyników badań pracę zakończono ogólnym wysunięciem wniosków.

2. Cel i zakres pracy

Celem pracy jest wykonanie analizy porównawczej obecnie istniejących narzędzi do projektowania szkiców interfejsów w kontekście User Experience. Na podstawie wyników badań na grupie badawczej oraz specyfikacji technicznej zostaną wyłonione narzędzia o najlepszych parametrach zależnie od funkcjonalności oraz ocen użytkowników.

Zakres pracy obejmuje dobór badanych narzędzi, analizę narzędzi, określenie metody badań, wyłonienie grupy badawczej, ocenę narzędzi oraz sformułowanie wniosków.

3. Projektowanie w wytwarzaniu oprogramowania

Cykl życia oprogramowania zwykle dzieli się na pięć do ośmiu etapów. Mogą być one łączone, dzielone lub pomijane, w zależności od zakresu projektu. Najpopularniejszymi fazami podczas wytwarzania oprogramowania są: analiza wymagań oraz wykonalności, projektowanie, rozwój oprogramowania i implementacja, testowanie i integracja, wdrożenie i pielęgnacja.

Podczas etapu planowania i projektowania interfejsów często wykonuje się ich szkice. Należy pamiętać, że projektowanie to nie sam wygląd, ale również sposób komunikacji z systemem [1, 2]. Projektowanie syste-

mów z uwzględnieniem User Experience związane jest z poziomem satysfakcji oraz zadowolenia użytkowników korzystających z programu.

3.1. Makietowanie

Makietowanie jest częścią procesu projektowania. Polega na utworzeniu w dowolnej formie szkicu wykonywanego oprogramowania.

Rozróżniamy trzy podstawowe terminy produktów projektowych. Modele szkieletowe ukazują główne grupy treści, strukturę i hierarchię informacji oraz opis interakcji poszczególnych elementów. Reprezentują początkową koncepcję produktu. Mogą być rysowane ręcznie jak i cyfrowo. Makiety od modeli szkieletowych odróżnia fakt, że muszą być one tworzone cyfrowo. Reprezentują końcowy wygląd produktu na rzeczywistym urządzeniu z uwzględnieniem kolorów, typografii czy ikonografii. Są statyczną reprezentacją produktu. Prototypy natomiast charakteryzują się największą szczegółowością, w dużym stopniu odzwierciedlają produkt, który ma zostać zaimplementowany. Skupiają się na wyglądzie oraz na demonstracji funkcjonalności. Zawierają wgląd w interakcję z produktem czyli gotową symulację pracy użytkownika z interfejsem [3].

3.2. Doświadczenia użytkownika

Doświadczenia użytkownika (UX - User Experience) to całokształt odczuć użytkownika związanych z korzystania z aplikacji. Pojęcie kojarzone jest z użytecznością połączoną z projektowaniem zorientowanym na użytkowników. Według normy ISO 9241-210 użyteczność opisana jest za pomocą wydajności, efektywności oraz satysfakcji. UX uznaje się za część procesu projektowania, w którym najważniejszym czynnikiem jest badanie wrażeń oraz odczuć użytkowników [4]. Ważnym aspektem jest przestrzeganie ogólnie ustalonych zasad oraz procedur, które ułatwiają korzystanie z produktu. Podczas projektowania brane są pod uwagę m.in. takie aspekty jak: estetyka, intuicyjność, łatwość obsługi czy użyteczność. Tworzony produkt powinien spełniać konkretne wymagania, które umożliwią użytkownikowi w przejrzysty i prosty sposób wykonać dane operacje, dotyczące funkcjonalności systemu bądź aplikacji [5, 6].

3.3. Przegląd dostępnej literatury

Analizując dostępną literaturę o danej tematyce, można znaleźć artykuł pod tytułem „User interface prototyping. Techniques, methods and tools" autorstwa Pawła Weichbrotha i Marcina Sikorskiego. Głównym celem autorów było dokonanie przeglądu, analizy oraz oceny technik, metod i narzędzi do prototypowania interfejsów. Ukazano podejście do prototypowania za pomocą utworzenia interaktywnego modelu prototypu systemu, który został przetestowany i oceniony przez użytkowników. Zwrócono uwagę na główne cechy, które musi spełniać każde oprogramowanie w celu uzyskania pozytywnych doświadczeń użytkownika. Ponadto ukazano podejście, które pokazuje tworzenie prototypów interfejsów przy wykorzystaniu specyfikacji wymagań użytkownika [7].

Artykuł „User experience prototyping - a literature review" autorstwa Tuomasa Nissinen'a skupia się na przeglądzie literatury. Głównym celem postawionym przez autora było określenie korzyści tworzenia prototypów interfejsów oraz przedstawienie w jakich fazach projektowania najpotrzebniejsze jest tworzenie prototypów aplikacji. Skupiono się również na rodzajach prototypowania oraz narzędziach do tego przeznaczonych. Zagłębiono się w rozumienie potrzeb użytkownika oraz opisano makiety wysokiej i niskiej wierności. Według autora przegląd literatury mógłby zostać rozszerzony o zaprojektowanie całego doświadczenia użytkownika i dodania na przykład studium przypadku, gdzie będą badane emocjonalne aspekty prototypowania [8].

Temat prototypowania w projektowaniu oprogramowania poruszyli również Sánchez-Villarín Alejandro, Alejandro Santos-Montaño oraz José Gonzalez Enríquez - autorzy artykułu „Automatic Reuse of Prototypes in Software Engineering: A Survey of Available Tools". Autorzy zauważyli, że występuje problem w prototypowaniu poprzez szybkie go wykonywanie i bez kontaktów z użytkownikami. W artykule przeanalizowano narzędzia do projektowania interfejsów, które mają pomagać projektantom w rozpoczęciu i przyspieszeniu pracy nad wytwarzaniem oprogramowania. Na podstawie wyników stwierdzono, że istnieje luka w inżynierii oprogramowania w aspekcie prototypowania [9].

Wartym uwagi artykułem jest „Analiza porównawcza narzędzi do budowy prototypów interfejsów" autorstwa Stanisława Lipskiego i Marka Miłosza. Głównym celem jest porównanie narzędzi służących do tworzenia prototypów interfejsów. Metodą badań użytą w pracy jest analiza wielokryterialna. Badanie polegało na utworzeniu przez grupę badawczą modelu systemu zgodnego ze zdefiniowaną dokumentacją, a następnie wypełnieniu ankiet odnośnie danego narzędzia. Przeprowadzono analizę wyników, wraz z normalizacją danych z ankiet. Każdemu programowi została przypisana ocena, w oparciu o obserwację autorów, jak i dane uzyskane z ankiet. W pracy stwierdzono, że każde z narzędzi wyróżnia się innymi cechami, oraz posiada różną efektywność, a wybór odpowiedniego narzędzia, zależny jest od rodzaju i wymagań tworzonego projektu [10].

Kolejnym istotnym artykułem jest „Analiza porównawcza systemów wspomagających prototypowanie interfejsów" Artura Łasochy oraz Marka Miłosza. W pracy użyto analizy wielokryterialnej, wzięto pod uwagę kryteria merytoryczne oraz formalne. Celem pracy było zbadanie narzędzi do projektowania pod względem przydatności, zależnie od potrzeb konkretnego projektu. Każdy z programów otrzymał odpowiednią liczbę punktów zależnie od stopnia spełnienia ustalonych kryterium. Na podstawie wyników stwierdzono, że odpowiedni wybór narzędzia, zależy od rodzaju oraz potrzeb danego projektu, a każdy z badanych programów posiada cechy wyróżniające spośród reszty, dzięki czemu możliwe jest wybranie odpowiedniego programu, które sprosta oczekiwaniom projektanta [11].

Temat wyboru narzędzi do projektowania interfejsów został również poruszony w artykule „Narzędzia

wspomagające projektowanie interfejsu użytkownika aplikacji webowych - analiza porównawcza" autorstwa Pawła Serafina oraz Marka Miłosza. Analizie zostały poddane narzędzia Adobe XD oraz Figma, oceniane za pomocą wielowymiarowej analizy porównawczej. W badaniu została użyta Skala Użyteczności Systemu. Wyniki zostały poddane analizie, która umożliwiła porównanie wybranych narzędzi pod względem wydajności, jednak według autorów wybranie odpowiedniego narzędzia, zależne jest od wymagań projektu [12].

Dokonując analizy artykułów warto wspomnieć o artykule Michał Cioczek, Tomasz Czarnota, Tomasz Szymczyk „Analiza współczesnych interfejsów człowiek-komputer". Badacze skupiają się na porównaniu interfejsów takich jak mysz komputerowa, touchpad, trackpoint, joystick, bezprzewodowy kontroler Xbox, pilot Magic, ekran dotykowy, kontroler Leap Motion czy cyfrowe rękawice. W celu analizy porównawczej stworzono specjalne środowisko badawcze z funkcjami, które miał wykonać użytkownik. Wynikiem jaki uzyskano był między innymi czas wykonania zadania i długość ścieżki jaką przebywa kursor sterowany badanymi interfejsami. Kryterium długości ścieżki przebytej przez kursor został wprowadzony do niniejszego badania. Jest ona mierzalnym potwierdzeniem ergonomii, łatwości pracy oraz intuicyjności danego programu [13].

4. Narzędzia do projektowania szkiców interfejsów

4.1. Dobór oprogramowania

Narzędzia wzięte pod uwagę w niniejszej analizie porównawczej zostały wybrane po ustaleniu kilku cech niezbędnych do wykorzystania ich podczas badania.

Głównym czynnikiem wyboru była cena narzędzia. Wybrano narzędzia wyłącznie darmowe lub posiadające darmowy okres próbny. Kolejną cechą była dostępność na system operacyjny Windows lub narzędzia działające za pomocą przeglądarki internetowej. Istotna podczas badania była również możliwość tworzenia prototypów zarówno na aplikacje mobilne jak i webowe.

Po określeniu niezbędnych wymagań dokonano przeglądu artykułów odnoszących się do obecnej popularności i jakości narzędzi do projektowania szkiców interfejsów. Wszystkie wymienione aspekty przyczyniły się do wyboru ośmiu najbardziej polecanych narzędzi, które poddano badaniu.

4.2. Charakterystyka wybranych narzędzi

Badaniu poddano narzędzia do tworzenia prototypów takie jak Adobe XD, Axure RP, Wireframe.cc, Pencil Project, Justinmind, Mockplus, UXPin oraz ProtoPie. Pozwalają one na tworzenie i udostępnianie makiet stron internetowych oraz aplikacji mobilnych. Ponadto:

- **Adobe XD** jest narzędziem bezpłatnym, może pracować na komputerach Mac oraz z systemem Windows. W przypadku funkcji mobilnych jest przystosowane dla systemu operacyjnego Android oraz iOS.
- **Pencil Project** to otwarte i bezpłatne oprogramowanie. Udostępnia wbudowane kolekcje kształtów, które można umieszczać za pomocą przeciągania.
- **Axure RP** jest dostępne jest dla systemów operacyjnych Windows i Mac. Narzędzie jest płatne, udostępniające darmowy okres próbny.
- **Wireframe.cc** to darmowa, minimalistyczna aplikacja internetowa oparta na chmurze.
- **Justinmind** to narzędzie dostępne dla użytkowników Windows oraz MacOS. Narzędzie udostępnia darmową wersję próbną.
- **Mockplus** jest narzędziem, z którego można korzystać poprzez instalację na komputerze jak i przeglądarkowo. Dostępne jest na system operacyjny Windows oraz MacOS. Posiada plan darmowy.
- **ProtoPie** jest dostępne na komputery z systemem Mac oraz Windows. Narzędzie jest zaawansowane technologicznie. Jest narzędziem płatnym, oferującym darmowy okres próbny.
- **UXPin** to aplikacja internetowa, która oferuje darmową wersję próbną, a także bezpłatną wersję studencką.

5. Badania

5.1. Problem badawczy

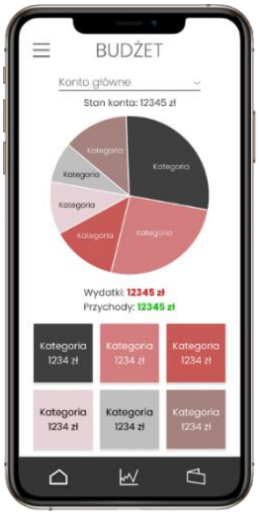
Badanie może pomóc wybrać oprogramowanie przyjazne dla użytkowników oraz z największą liczbą możliwości. Doświadczenia użytkownika podczas korzystania z oprogramowania mogą być równie ważne jak jego możliwości funkcjonalne. Korzystanie z oprogramowania łatwego do opanowania dla ludzi z różnych środowisk wpływa na szybkość tworzonego modelu projektu.

Postawiona teza badawcza brzmi: Wykształcenie znacznie wpływa na ocenę narzędzi do projektowania szkiców interfejsów.

5.2. Metoda badań

Głównym celem badania było wyłonienie najlepszego narzędzia spośród wybranych do tworzenia szkiców interfejsów w kontekście User Experience.

Badana była praca na danym narzędziu, doświadczenia wynikające z korzystania z nich, oraz kryteria techniczne. Na potrzeby badania zaprojektowano model szkieletowy aplikacji mobilnej (Rysunek 1).



Rysunek 1: Szablon zadanego szkicu interfejsu dla aplikacji mobilnej

Szkic następnie miał być odwzorowany przez grupę badawczą w każdym z narzędzi. Podczas tego etapu mierzony był czas wykonania szkiców w sekundach oraz droga kursora wyrażona w pikselach. Z uzyskanych wyników wyciągane były średnie arytmetyczne dla każdego z narzędzi, a następnie przydzielano odpowiednią liczbę punktów zgodnie z punktacją z Tabeli 1, uwzględniając podział na podgrupy badawcze.

Tabela 1: Punktacja dla czasów utworzenia szkiców i drogi kursora

Kryterium i waga	Podgrupa badawcza	Wskaźnik czasu realizacji	Liczba punktów
Średni czas wykonania modelu szkieletowego w minutach (60%)	Doświadczona	< 20 min	1
		20-22 min	0,5
		> 22min	0
	Niedoświadczona	< 23min 30s	1
		23min 30s -25min 30s	0,5
		> 25 min 30s	0
	Pokolenie X	< 29 min 30s	1
		29min 30s -31min 30s	0,5
		> 31 min 30 s	0
Średnia droga kursora w pikselach (40%)	Doświadczona	< 300000 px	1
		300000 px -335000 px	0,5
		> 335000 px	0
	Niedoświadczona	< 365000 px	1
		365000 px -410000 px	0,5
		> 410000 px	0
	Pokolenie X	< 295000 px	1
		295000 px -325000 px	0,5
		> 325000 px)	0

Po wykonaniu modeli, osoby badane otrzymały ankietę, w której oceniały pracę i odczucia towarzyszące podczas badania w każdym narzędziu. Analogicznie, z uzyskanych wyników wyciągane były średnie dla każdego z narzędzi, przydzielając odpowiednią liczbę punktów zgodnie z punktacją. Dla pytań z możliwością oceny w skali od 1 do 5 maksymalna liczba punktów (1) przydzielana była za średnią ocen powyżej 4,0. 0,5 punktu przydzielano za średnią pomiędzy 3,0 – 4,0, natomiast 0 pkt otrzymywało narzędzie, jeśli średnia ocen wynosiła poniżej 3,0. Dla pytań wielokrotnego wyboru punktacja została przedstawiona w Tabeli 2.

Tabela 2: Punktacja dla pytań ankietowych wielokrotnego wyboru

Kryterium	Podgrupa badawcza	Wskaźnik liczby wskazań	Liczba punktów
Które z narzędzi wywołało u Ciebie pozytywne odczucia?	doświadczona, niedoświadczona	4-6 wskazań	1
		2-3 wskazania	0,5
		0-1 wskazań	0
	generacja X	4 wskazania	1
		2-3 wskazania	0,5
		0-1 wskazań	0
Które z narzędzi byś wybrał do stworzenia szkiców aplikacji?	doświadczona, niedoświadczona	4-6 wskazań	1
		2-3 wskazania	0,5
		0-1 wskazań	0
	generacja X	4 wskazania	1
		2-3 wskazania	0,5
		0-1 wskazań	0
Które z narzędzi wywołało u Ciebie negatywne odczucia?	doświadczona, niedoświadczona	0-1 wskazań	1
		2-3 wskazania	0,5
		4-6 wskazań	0
	generacja X	0 wskazań	1
		1-2 wskazania	0,5
		3-4 wskazania	0
Które z narzędzi sprawiło Ci najwięcej trudności i nie wybrałabyś go do tworzenia szkiców aplikacji?	doświadczona, niedoświadczona	0-1 wskazań	1
		2-3 wskazania	0,5
		4-6 wskazań	0
	generacja X	0 wskazań	1
		1-2 wskazania	0,5
		3-4 wskazania	0

Uzupełnieniem była ocena kryteriów technicznych bez udziału grupy badawczej. Każde narzędzie za określone kryteria otrzymywało właściwą liczbę punktów. Ustalona punktacja została przedstawiona w Tabeli 3.

Tabela 3: Punktacja dla kryteriów technicznych narzędzi

Kryterium	Wskaźnik	Liczba punktów
Dostępność na różne systemy operacyjne	Wszystkie	1
	Windows	0,5
	Mac OS	0,5
Możliwość projektowania na różne platformy	Wszystkie	1
	Aplikacje mobilne	0,5
	Aplikacje webowe	0,5
Rodzaje eksportu projektów	Wszystkie	1
	PNG, JPG, PDF	0,4
	HTML	0,3
	SVG	0,3
Współpraca pomiędzy zespołem projektowym	Tak	1
	Nie	0
Cena narzędzia (miesięczna subskrypcja)	Niska	1
	Średnia	0.5
	Wysoka	0
Dostęp do dokumentacji	Tak	1
	Nie	0
Wsparcie techniczne oraz społeczność	Wszystkie	1
	Wsparcie techniczne	0.5
	Społeczność	0.5
	Brak	0
Możliwość korzystania z rozszerzeń	Możliwość zainstalowania	1
	Brak	0
Ilość potrzebnego miejsca na dysku twardym	Do 1GB	1
	> 1GB	0

Na ogólną ocenę narzędzia składały się trzy wymienione wcześniej aspekty. Największą wagę wynoszącą 50% miał czas wykonania szkicu wraz z drogą myszy, gdyż było to głównym czynnikiem wskazującym na ergonomiczność narzędzia. Pytania ankietowe dostały wagę 30%, ponieważ były drugim ważnym kryterium oceny narzędzi. Techniczne kryteria oprogramowania dostały najniższą wagę, która wynosiła 20%, gdyż były uzupełnieniem porównania narzędzi.

W badaniu wprowadzono funkcję kary. Funkcja kary wyrażona jest poprzez dodanie czasu potrzebnego na wykonanie elementu oraz średniej drogi myszki niezbędnej do narysowania pominiętego elementu do całościowego wyniku wykonania makiety przez daną osobę. Do ostatecznej oceny narzędzi brane były wyniki wraz z uwzględnioną funkcją kary. Kary zostały określone poprzez wykonanie przez ekspertów zadanego szkicu w każdym z programów mierząc czasy wykonania i drogę kursora dla każdego elementu szkicu. Otrzymane wyniki zostały uśrednione i na tej podstawie mogły posłużyć do dodawania kar poszczególnych szkiców wykonywanych przez grupę badawczą. Jako, że ergonomia każdego z narzędzi jest inna, zrobiono to dla każdego programu osobno. Nie zrobienie minimum sześciu elementów przez osoby badane było niedopuszczalne. Kary zostały przedstawione w Tabeli 4.

Tabela 4: Kary czasowe oraz kary drogi kursora dla badanych programów

Element	Części składowe	Adobe XD	Axure RP	Justinmind	Mockplus	Pencil Project	ProtoPie	UxPin	Wireframe.cc
Kara czasowa [s]									
Napis „BUDŻET”	Całkowity brak	20	20	20	20	15	20	15	10
Ikona hamburger menu	Całkowity brak	40	20	40	20	25	25	20	15
Lista rozwijana	Całkowity brak	35	30	40	25	25	80	45	40
Napis „Stan konta: 12345 zł”	Całkowity brak	20	30	15	20	15	20	20	20
Diagram kołowy podzielony na 6 części oznaczony kolorami i podpisami	Całkowity brak	280	240	360	280	130	250	270	340
	Wstawiony zamiennik	260	220	340	260	120	230	240	320
	Brak kolorów	190	110	270	150	40	120	165	310
	Brak podpisów	35	20	35	20	15	20	35	30
	Brak linii	15	15	15	15	10	15	40	20
Napis „Wydatki: ” z pogrubionym i pokolorowanym „12345 zł”	Całkowity brak	20	30	20	25	40	30	25	30
	Brak wyróżnienia kwoty	10	10	10	10	15	10	10	15
Napis „Przychody: ” z pogrubionym i pokolorowanym „12345 zł”	Całkowity brak	20	30	20	25	40	30	25	30
	Brak wyróżnienia kwoty	10	10	10	10	15	10	10	15
6 kafelek, każdy z napisem „Kategoria 1234 zł”, tego samego rozmiaru, różnego koloru	Całkowity brak	60	120	70	110	90	100	90	100
	Brak kolorów	10	30	30	20	20	20	35	25
	Brak podpisów	20	30	20	30	30	30	30	30
Dolny pasek nawigacyjny	Całkowity brak	10	20	15	20	20	20	15	15
Ikona na pasku nawigacyjnym - dom (rysowana lub gotowa)	Całkowity brak	20	20	15	20	120	125	15	20
	Wstawiony zamiennik	15	10	5	10	100	105	5	5
Ikona na pasku nawigacyjnym - wykres (rysowana lub gotowa)	Całkowity brak	20	20	50	20	110	110	15	75
	Wstawiony zamiennik	15	10	30	10	90	90	5	55
Ikona na pasku nawigacyjnym - portfel (rysowana lub gotowa)	Całkowity brak	30	80	50	20	100	90	15	20
	Wstawiony zamiennik	20	60	30	10	80	70	5	5
Kara drogi kursora [px]									
Napis „BUDŻET”	Całkowity brak	4300	4160	5950	4100	3700	5000	3000	1500
Ikona hamburger menu	Całkowity brak	7550	4200	10000	2500	4000	2800	9000	1600
Lista rozwijana	Całkowity brak	6650	5550	12700	5500	4100	16500	15000	4900
Napis „Stan konta: 12345 zł”	Całkowity brak	3950	5100	2150	4500	2400	5000	2600	2200
Diagram kołowy podzielony na 6 części oznaczony kolorami i podpisami	Całkowity brak	61200	53300	77660	79500	27000	73500	60600	54600
	Wstawiony zamiennik	54000	49900	69600	74900	26000	69000	55500	54000
	Brak kolorów	40750	38000	54900	50000	9000	45000	36100	49500
	Brak podpisów	7800	3000	9200	2000	3400	4000	7100	5100
	Brak linii	1500	2000	3000	3000	2000	3000	12300	2000
Napis „Wydatki: ” z pogrubionym i pokolorowanym „12345 zł”	Całkowity brak	2500	5500	3050	5500	10000	4500	4000	3800
	Brak wyróżnienia kwoty	1200	1800	2700	3000	4500	2000	2000	2500
Napis „Przychody: ” z pogrubionym i pokolorowanym „12345 zł”	Całkowity brak	2500	5500	3050	5500	10000	4500	4000	3800
	Brak wyróżnienia kwoty	1200	1800	2700	3000	4500	2000	2000	2500
6 kafelek, każdy z napisem „Kategoria 1234 zł”, tego samego rozmiaru, różnego koloru	Całkowity brak	19950	25900	23500	30000	25000	24000	22000	15500
	Brak kolorów	3850	8000	18300	8000	8000	4800	12300	2600
	Brak podpisów	3000	6500	3300	5000	5000	8000	5100	3600
Dolny pasek nawigacyjny	Całkowity brak	3750	4500	5000	5000	6800	5000	2800	2000
Ikona na pasku nawigacyjnym - dom (rysowana lub gotowa)	Całkowity brak	6300	4000	5900	3500	22500	22300	5900	2500
	Wstawiony zamiennik	2000	2000	3900	1000	18500	18300	3900	500
Ikona na pasku nawigacyjnym - wykres (rysowana lub gotowa)	Całkowity brak	3000	4000	7900	3500	23500	22600	5900	6000
	Wstawiony zamiennik	2000	2000	5900	1000	19500	18600	3900	4000
Ikona na pasku nawigacyjnym - portfel (rysowana lub gotowa)	Całkowity brak	3900	11500	13900	3500	17800	17500	5900	2500
	Wstawiony zamiennik	2400	9000	11900	1000	13800	13500	3900	500

5.3. Grupa badawcza

Grupa badawcza składała się z 16 osób i została podzielona na trzy podgrupy. Pierwszą sześcioosobową podgrupą byli studenci ostatniego roku studiów magisterskich na kierunku informatyka. Średni wiek tej podgrupy wynosi 24 lata o odchyleniu standardowym 0,69. Drugą sześcioosobową podgrupą, były osoby innych kierunków niż informatyczne, gdzie średnia wieku to 24 lata, a odchylenie standardowe wynosiło 2,81. Ostatnią podgrupą były osoby z pokolenia X o średniej wieku 52 lat i odchyleniu standardowemu 2,86.

Zdecydowano się na dołączenie do badania osób z generacji X, aby sprawdzić czy osoby starsze również poradzą sobie z badaniem i jak ich oceny będą różniły się od osób młodszych.

5.4. Przebieg badania na grupie badawczej

Pierwszym etapem badania była wstępna ankieta, dzięki której możliwe było poznanie odpowiedzi na pytanie

o znajomość wybranych narzędzi. Aż 11 osób badanych nie znało żadnego z narzędzi. Najbardziej znanym programem okazało się Adobe XD, które otrzymało cztery głosy, po dwa głosy otrzymały narzędzia Wireframe.cc, Axure RP oraz Justinmid, natomiast po jednym głosie otrzymały Pencil Project oraz Mockplus.

Następnie każda z osób miała za zadanie wykonać makietę aplikacji mobilnej na podstawie wcześniej ustalonego szablonu w każdym z programów. Kolejność badanych narzędzi była losowa w przypadku każdej osoby, aby otrzymane wyniki były rzetelne. Podczas tego etapu mierzony był czas utworzenia szkicu oraz długość drogi kursora podana w pikselach. Po wykonaniu zadania grupa badawcza otrzymała drugą ankietę do oceny programów pod względem odczuć oraz doświadczeń związanych z pracą w danym narzędziu. Kryteria jakie zostały wzięte pod uwagę to: przejrzystość, dostępne pomoce w narzędziu, możliwość utworzenia każdego elementu zadanego szkicu, możliwość tworzenia i edycji wielu elementów jednocześnie czy szybkość opanowania obsługi narzędzia. Nie zabrakło również pytań o kryteria związane z User Experience.

6. Wyniki badań

6.1. Czasy wykonania szkicu i droga kursora

W wynikach badań uwzględniono wyniki zarówno zmierzone bezpośrednio po wykonaniu szkiców oraz wyniki z uwzględnieniem funkcji kary. Wyniki średnich czasów utworzenia szkiców oraz drogi kursora w wybranych narzędziach zostały przedstawione w Tabeli 5.

Tabela 5: Wyniki średnich oraz odchyleń standardowych dla czasów utworzenia szkiców oraz drogi kursora w wybranych narzędziach

Nazwa narzędzia	Podgrupa doświadczona				Podgrupa niedoświadczona				Podgrupa pokolenie X			
	Średni czas utworzenia szkiców [mm:ss:ms]	Odchylenie standardowe czasów makietowania [mm:ss:ms]	Średnia droga kursora [px]	Odchylenie standardowe drogi kursora [px]	Średni czas utworzenia szkiców [mm:ss:ms]	Odchylenie standardowe czasów makietowania [mm:ss:ms]	Średnia droga kursora [px]	Odchylenie standardowe drogi kursora [px]	Średni czas utworzenia szkiców [mm:ss:ms]	Odchylenie standardowe czasów makietowania [mm:ss:ms]	Średnia droga kursora [px]	Odchylenie standardowe drogi kursora [px]
Wyniki bez uwzględnienia funkcji kary												
Adobe XD	17:14:40	5:17:34	301155	118747	19:07:40	6:05:04	353293	151067	31:29:15	4:52:59	323573	54874
Pencil Project	20:30:10	1:40:36	318436	36497	26:26:00	7:03:45	460827	159281	28:30:30	10:18:19	378079	177742
Axure RP	16:31:10	3:37:30	286336	73531	19:13:20	2:40:24	317898	32914	25:42:30	5:52:38	290747	150195
Wireframe.cc	16:44:40	6:03:44	192110	130442	16:07:30	5:20:31	166666	54510	23:39:15	5:19:50	105635	11912
Justinmind	15:38:40	1:33:54	229327	27154	19:16:50	6:56:13	281973	134099	26:42:30	2:32:29	223989	50649
Mockplus	16:06:20	3:38:07	258142	93832	21:02:40	3:51:41	397795	61813	22:45:15	3:30:04	268069	112564
ProtoPie	19:04:00	7:51:28	294518	131992	21:57:10	3:05:36	374051	87509	29:00:00	8:38:12	249397	84623
UXPin	14:53:10	3:39:09	244994	66784	21:12:40	6:57:55	321055	121103	24:19:15	4:23:29	204295	61114
Wyniki z uwzględnieniem funkcji kary												
Adobe XD	20:14:40	5:04:21	337555	122168	22:16:50	5:14:52	394151	138904	35:19:15	5:03:32	370961	59069
Pencil Project	23:35:10	3:14:06	357436	58358	29:11:50	6:23:15	494210	148071	31:00:30	11:40:05	410029	200342
Axure RP	19:02:50	3:37:54	330319	75499	22:14:10	2:57:06	364998	37209	28:17:30	5:33:49	334472	147563
Wireframe.cc	23:02:10	5:54:10	248327	128619	22:14:10	5:15:12	220950	56308	30:16:45	4:13:50	164585	7908
Justinmind	21:21:10	1:18:42	302793	32017	24:53:30	7:18:21	353156	136442	32:20:00	2:10:40	296164	41666
Mockplus	18:31:20	2:38:15	302309	80171	23:41:00	3:54:57	449129	62186	25:57:45	2:27:57	328194	97640
ProtoPie	24:18:20	6:09:16	377685	123596	26:52:10	4:44:05	452751	103683	32:52:30	8:03:40	316872	80527
UXPin	17:47:20	2:36:51	286627	54692	23:53:30	7:44:15	364921	118570	27:56:45	3:58:14	257520	54332

Średni czas wykonania szkicu bez uwzględnienia kar w podgrupie doświadczonej najlepiej wypadł dla narzędzia UXPin. Pencil Project uzyskało najgorsze wyniki zarówno dla podgrupy doświadczonej jak i niedoświadczonej. Najlepszy średni czas wykonania szkicu przez podgrupę niedoświadczoną otrzymano dla programu Wireframe.cc. Podgrupa pokolenia X uzyskała najlepszy średni czas wykonania szkicu w programie Mockplus, natomiast najgorszy dla narzędzia Adobe XD.

Największe wartości średnie drogi kursora w wybranych narzędziach bez uwzględnienia funkcji kary uzyskała podgrupa niedoświadczona. Najdłuższą średnią drogę kursora wyrażoną w pikselach dla wszystkich podgrup uzyskało narzędzie Pencil Project, natomiast najkrótszą średnią drogę uzyskało narzędzie Wireframe.cc, również dla całej grupy badawczej.

W przypadku wyników z uwzględnieniem funkcji kary dla podgrupy doświadczonej najgorszy czas uzyskało narzędzie ProtoPie, pozostałe najlepiej oraz najgorzej wypadające narzędzia pozostały bez zmian dla każdej podgrupy.

Największą średnią drogę kursora podczas wykonania szkiców z uwzględnieniem funkcji kary dla podgrupy doświadczonej uzyskało narzędzie ProtoPie, pozostałe największe oraz najmniejsze średnie pozostały bez zmian dla każdej podgrupy.

Na podstawie uzyskanych średnich wyników czasów oraz dróg myszy z uwzględnieniem funkcji kary, zostały przydzielone punkty zgodnie z punktacją ustaloną w Tabeli 1. W Tabeli 6 zestawiono punktacje czasów wykonania szkicu oraz drogi myszki dla każdego narzędzia z uwzględnieniem podziału na podgrupy, jak również łączny wynik. Największą liczbę punktów za średni

czas wykonania szkicu uzyskało narzędzie Axure RP, natomiast za najlepszą średnią drogę myszy najwięcej punktów uzyskało narzędzie Wireframe.cc.

Tabela 6: Przyznane punkty dla czasów utworzenia szkiców oraz drogi kursora z uwzględnieniem funkcji kar

Nazwa narzędzia	Podgrupa doświadczona	Podgrupa niedoświadczona	Podgrupa pokolenie X	Łącznie
Punkty dla czasów utworzenia szkiców				
Adobe XD	0.5	1	0	1,5
Pencil Project	0	0	0.5	0,5
Axure RP	1	1	1	3,0
Wireframe.cc	0	1	0.5	1,5
Justinmind	0.5	0.5	0	1,0
Mockplus	1	0.5	1	2,5
ProtoPie	0	0	0	0,0
UXPin	1	0.5	1	2,5
Punkty dla drogi kursora				
Adobe XD	0	0.5	0	0,5
Pencil Project	0	0	0	0,0
Axure RP	0.5	0.5	0	1,0
Wireframe.cc	1	1	1	3,0
Justinmind	0.5	1	0.5	2,0
Mockplus	0.5	0	0.5	1,0
ProtoPie	0	0	0.5	0,5
UXPin	1	0.5	1	2,5

6.2. Ankieta doświadczeń użytkownika

W Tabeli 7 zestawiono łączną punktacje za pytania z ankiety, którą badani otrzymali po wykonaniu szkiców, dla każdego narzędzia z uwzględnieniem podziału na podgrupy, jak również łączny wynik.

Tabela 7: Łączna liczba punktów z ankiety.

Nazwa narzędzia	Podgrupa doświadczona	Podgrupa niedoświadczona	Podgrupa pokolenie X	Łącznie
Adobe XD	10,0	9,0	7,5	26,5
Pencil Project	1,5	0,5	1,0	3,0
Axure RP	14,0	12,0	9,0	35,0
Wireframe.cc	6,5	0,0	1,5	8,0
Justinmind	14,0	12,0	13,5	39,5
Mockplus	11,0	12,5	14,5	38,0
ProtoPie	4,5	1,0	4,0	9,5
UXPin	14,0	14,0	13,0	41,0

Punkty były przyznawane na podstawie średnich ocen odpowiedzi na pytania, gdzie przedział ocen wynosił 1-5. W przypadku średniej znajdującej się w przedziale 3-4 było przyznawane 0,5 pkt, dla niższej średniej nie przydzielano punktów, natomiast dla średniej powyżej 4 przyznawano 1 punkt. Pytania z ankiety odnosiły się do oceny przejrzystości, łatwości obsługi, ilości dostępnych podpowiedzi i pomocy, liczby dostępnych komponentów, możliwości utworzenia wszystkich elementów szkicu oraz edycji wielu elementów jednocześnie, a także oceny cech oprogramowania pod kątem UX takich jak: przyjazność, użyteczność, zapotrzebowanie, pożądanie, dostępność, intuicyjność. W ankiecie uwzględniono również cztery pytania odnoszące się do tego które z narzędzi badana osoba wybrałaby do własnego użytku oraz którego nie wybrałaby. Pytania te były punktowane na podstawie Tabeli 2. Największą łączną liczbę punktów za oceny z ankiet uwzględniając całą grupę badawczą uzyskało narzędzie UXPin z wynikiem 41 punktów.

6.3. Ocena kryteriów technicznych

W Tabeli 8 przedstawiono punktację dla każdego narzędzia za kryteria techniczne (szczegółowe kryteria punktacji przedstawiono w Tabeli 3). Najwięcej punktów zdobyło narzędzie Mockplus z wynikiem 8,5 punktu.

Tabela 8: Punkty przyznane za kryteria techniczne

Kryterium	Adobe XD	Pencil Project	Axure RP	Wireframe.cc	Justinmind	Mockplus	ProtoPie	UXPin
Dostępność na systemy operacyjne	1	1	1	1	1	1	1	1
Projektowanie na różne platformy	1	1	1	1	1	1	1	1
Rodzaje eksportu projektów	0,7	1	0,7	0,4	1	1	0,3	0,7
Współpraca między ludźmi w projekcie	1	0	1	1	1	1	1	1
Cena narzędzia	0,5	1	0	0,5	0	0,5	1	0
Dostęp do dokumentacji	1	1	1	1	1	1	1	1
Wsparcie techniczne oraz społeczność	1	0	1	1	1	1	1	1
Możliwość korzystania z rozszerzeń	1	1	1	0	1	1	1	1
SUMA	7,2	7	6,7	6,9	7	8,5	8,3	7,7

6.4. Podsumowanie wyników badań

Zsumowano zdobyte punkty w każdym z kryteriów uwzględniając wagę każdego z nich. Wyniki podzielono analogicznie dla wszystkich podgrup badawczych. Podsumowanie przedstawiono w Tabeli 9.

Tabela 9: Liczba punktów z każdego etapu dla wszystkich podgrup po uwzględnieniu wag kryteriów

	Podgrupa doświadczona				Podgrupa niedoświadczona				Podgrupa pokolenie X			
Nazwa narzędzia	Czasy wykonania szkiców i droga kursora	Ankiety	Kryteria techniczne	Łącznie	Czasy wykonania szkiców i droga kursora	Ankiety	Kryteria techniczne	Łącznie	Czasy wykonania szkiców i droga kursora	Ankiety	Kryteria techniczne	Łącznie
Adobe XD	0,30	0,48	0,80	0,45	0,80	0,43	0,80	0,69	0,00	0,36	0,80	0,27
Pencil Project	0,00	0,07	0,78	0,18	0,00	0,02	0,78	0,16	0,30	0,05	0,78	0,32
Axure RP	0,80	0,67	0,74	0,75	0,80	0,57	0,74	0,72	0,60	0,43	0,74	0,58
Wireframe.cc	0,40	0,31	0,77	0,45	1,00	0,00	0,77	0,65	0,70	0,07	0,77	0,52
Justinmind	0,50	0,67	0,78	0,61	0,70	0,57	0,78	0,68	0,20	0,64	0,78	0,45
Mockplus	0,80	0,52	0,94	0,75	0,30	0,60	0,94	0,52	0,80	0,69	0,94	0,80
ProtoPie	0,00	0,21	0,92	0,25	0,00	0,05	0,92	0,20	0,20	0,19	0,92	0,34
UXPin	1,00	0,67	0,86	0,87	0,50	0,67	0,86	0,62	1,00	0,62	0,86	0,86

Analizując wyniki, można odczytać, że dla podgrupy doświadczonej największą liczbę punktów otrzymało narzędzie UXPin, natomiast najgorszy wynik otrzymało narzędzie Pencil Project. W przypadku podgrupy niedoświadczonej narzędziem, które otrzymało największą liczbę punktów jest Axure RP, natomiast tak jak w przypadku podgrupy doświadczonej najgorszy wynik otrzymało narzędzie Pencil Project. Dla podgrupy pokolenia X najwięcej punktów uzyskało narzędzie UXPin, dokładnie jak w przypadku grupy doświadczonej. Najgorzej pod względem liczby punktów wypadło Adobe XD.

Tabela 10 przedstawia łączną liczbę punktów dla każdego z narzędzi, bez uwzględnienia podziału na grupy. W ogólnym zestawieniu najwięcej punktów uzyskało narzędzie UXPin z wynikiem 0,78 pkt. Najgorzej natomiast wypadło narzędzie Pencil Project, który uzyskał 0,22 punktu.

Tabela 10: Łączna liczba punktów z każdego etapu dla wszystkich podgrup po uwzględnieniu wag kryteriów

Nazwa narzędzia	Łącznie
Adobe XD	0,47
Pencil Project	0,22
Axure RP	0,68
Wireframe.cc	0,54
Justinmind	0,58
Mockplus	0,69
ProtoPie	0,26
UXPin	0,78

7. Wnioski

Narzędzia wybrane do badania różnią się od siebie w wielu aspektach. Oprogramowanie Wireframe.cc wyróżnia się m.in. obecnością menu kontekstowego, przez co średnie długości drogi kursora były najniższe wśród wszystkich podgrup badawczych. Pencil Project jest programem o dużej liczbie komponentów, jednak większość z nich jest przestarzała, w związku z tym mało przydatna. ProtoPie to narzędzie o bardzo wysokim stopniu zaawansowania, a większość funkcji, które posiada nie została uwzględniona w badaniu z powodu jego specyfiki, co mogło wpłynąć na ocenę.

W każdym z programów najszybciej szkice były wykonywane przez osoby doświadczone. Najdłuższe czasy dla każdego programu zaobserwowano dla podgrupy generacji X. Największe średnie drogi myszy uzyskano wśród podgrupy niedoświadczonej, natomiast najmniejsze uzyskała podgrupa pokolenia X. Można wnioskować, iż osoby starsze, reprezentujące generacje X, spędzają dużo więcej czasu na szukaniu potrzebnego elementu na ekranie nie pracując przy tym myszą.

UXPin to narzędzie, które dostało najwięcej punktów w ogólnej ocenie, biorąc pod uwagę wszystkie kryteria z uwzględnieniem ich wag, dla podgrupy doświadczonej oraz podgrupy pokolenia X. W podgrupie niedoświadczonej narzędzie, które otrzymało największą liczbę punktów w ogólnej klasyfikacji to Axure RP. Najgorszy całościowy wynik w grupie doświadczonej oraz niedoświadczonej otrzymało narzędzie Pencil Project. Dla pokolenia X najgorzej wypadło narzędzie Adobe XD. Mogło tak być ze względu na minimalistyczny interfejs, zawierający mało elementów, podczas gdy osoby bardziej doświadczone wiedziały o możliwości doinstalowania rozszerzeń ułatwiających pracę. Narzędzia, które otrzymały najmniejszą oraz największą liczbę punktów w zestawieniu ogólnym bez uwzględnienia podziału na podgrupy pokrywają się z narzędziami o najmniejszej oraz największej liczbie zdobytych punktów w zestawieniu dla podgrupy doświadczonej. Analizując otrzymane wyniki, można uznać, że postawiona w pracy teza badawcza jest prawdziwa.

Przeprowadzone badania i uzyskane wyniki mogą jedynie wspomóc projektantów przy wyborze odpowiedniego narzędzia. W dużej mierze ostateczny wybór powinien opierać się na specyfikacji projektu.

Literatura

[1] A. Molga, A. Hamela, D. Pawłowski, Projektowanie aplikacji i interakcja z użytkownikiem, Dydaktyka informatyki 12 (2017) 233-241.

[2] S. Zawadzki, Jak stworzyć intuicyjny interfejs użytkownika (user interface)?, https://smartbees.pl/blog/jak-stworzyc-intuicyjny-interfejs-uzytkownika-user-interface, [15.05.2021].

[3] Modele szkieletowe, prototypy, makiety: jaka jest różnica?, https://ichi.pro/pl/modele-szkieletowe-prototypy-makiety-jaka-jest-roznica-6771688105039, [15.05.2021].

[4] J. Górowski, N. Iwaszczuk, User Experience w procesie tworzenia oprogramowania, 3MICT 3 (2020) 97.

[5] Ch. Badura, UXUI. Design Zoptymalizowany. Manual Book, Helion, 2019.

[6] A. Jesmond, J. Chudley, Projektowanie witryn internetowych User eXperience, Helion, 2013.

[7] P. Weichbroth, M. Sikorski, User interface prototyping. Techniques, methods and tools, Studia Ekonomiczne. Zeszyty Naukowe Uniwersytetu Ekonomicznego w Katowicach 234 (2015) 184-198.

[8] T. Nissinen, User experience prototyping–a literature review, University of Oulu (2015).

[9] A. Sánchez-Villarín, A. Santos-Montaño, J. Gonzalez Enríquez, Automatic Reuse of Prototypes in Software Engineering: A Survey of Available Tools, WEBIST (2019) 144-150.

[10] S. Lipski, M. Miłosz, Analiza porównawcza narzędzi do budowy prototypów interfejsów, Journal of Computer Sciences Institute 1 (2016) 38-43.

[11] A. Łasocha, M. Miłosz, Analiza porównawcza systemów wspomagających prototypowanie interfejsów, Journal of Computer Sciences Institute 4 (2017) 122-127.

[12] P. Serafin, M. Miłosz, Narzędzia wspomagające projektowanie interfejsu użytkownika aplikacji webowych – analiza porównawcza, Journal of Computer Sciences Institute 12 (2019) 172-178.

[13] M. Cioczek, T. Czarnota, T. Szymczyk, Analysis of modern human-computer interfaces, Journal of Computer Sciences Institute 18 (2021) 22-29.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 59–65
Received: 4 November 2021
Accepted: 25 November 2021

Comparing the performance of the object-relational mapping programming frameworks available in Java

Porównanie wydajności szkieletów programistycznych mapowania obiektowo-relacyjnego dostępnych w języku Java

Jakub Benedykt Pitera* , Mateusz Połeć*, Grzegorz Kozieł

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

Abstract

The paper concerns a performance evaluation of selected object-relational mapping tools. This work is intended to assist software architects in determining which of the analyzed libraries will be the optimal choice for use in a specific project. The work includes the comparison of Hibernate ORM, EclipseLink, Apache OpenJPA and DataNucleus libraries from the theoretical and practical point of view. Each of the examined tools has been described according to criteria allowing to distinguish the most important features influencing communication with relational databases. These features will then be compared on a practical level by examining the behavior of the test applications. In terms of performance, the Apache OpenJPA library turned out to be the best, but in terms of configuration and availability it significantly differs from other libraries. This was caused by an unintuitive configuration and poor documentation of the technology. In this respect, the Hibernate library definitely dominated thanks to full compatibility with the Spring programming framework, intuitive configuration as well as rich documentation and support by the extensive community of programmers using it.

Keywords: Java ORM; Java Persistence API; performance evaluation

Streszczenie

Artykuł poświęcono wykonaniu analizy porównawczej wybranych narzędzi mapowania obiektowo-relacyjnego. Jego celem jest pomoc architektom oprogramowania w określeniu, która z analizowanych bibliotek będzie optymalnym wyborem do użycia w określonym projekcie. Celem artykułu jest ocena bibliotek Hibernate ORM, EclipseLink, Apache OpenJPA oraz DataNucleus pod kątem teoretycznym oraz praktycznym. Każde z badanych narzędzi opisane zostało według kryteriów pozwalających na wyodrębnienie najważniejszych cech mających wpływ na komunikację z relacyjnymi bazami danych. Cechy te następnie zostały porównane na poziomie praktycznym poprzez zbadanie zachowania aplikacji testowych. Pod względem wydajnościowym, najlepsza okazała się biblioteka Apache OpenJPA, jednak pod względem konfiguracji i dostępności znacznie odstępuje od innych bibliotek. Spowodowane było to nieintuicyjną konfiguracją oraz ubogą dokumentacją technologii. Pod tym względem zdecydowanie górowała biblioteka Hibernate dzięki pełnej kompatybilności ze szkieletem programistycznym Spring, intuicyjnej konfiguracji oraz bogatej dokumentacji i wsparcia przez obszerną społeczność korzystających z niej programistów.

Słowa kluczowe: Java ORM; Java Persistence API; ocena wydajności

*Corresponding author

Email address: jakub.pitera@pollub.edu.pl (J. B. Pitera), mateusz.polec1@pollub.edu.pl (M. Połeć)

1. Wstęp

W dzisiejszych czasach możliwość gromadzenia danych przez systemy informatyczne jest rzeczą powszechnie spotykaną, niezależnie czy mówimy tu o prostych stronach-wizytówkach czy też o skomplikowanych systemach zarządzających całymi korporacjami – wszystkie wykorzystują relacyjne bazy danych. Zdecydowana większość projektowanych aplikacji tworzona jest za pomocą programowania obiektowego [1]. Jednoczesne wykorzystywanie relacyjnych baz danych i podejścia obiektowego w trakcie tworzenia systemów informatycznych spowodowało utworzenie nowej, abstrakcyjnej warstwy, umożliwiającej przedstawianie tabel bazy danych w postaci klas i obiektów w kodzie źródłowym aplikacji. Mechanizm ten został określony mapowaniem obiektowo-relacyjnym (ORM – ang. *object-relational mapping*) [2].

Język programowania Java posiada całą paletę dostępnych bibliotek ORM [3]. Mnogość dostępnych rozwiązań stawia jedno pytanie – której z bibliotek użyć do zrealizowania projektu? Na rynku dostępnych jest wiele rozwiązań: Hibernate ORM, czyli najpopularniejsza biblioteka ORM, która od 2001 jest pionierem stosowanych dzisiaj rozwiązań, przyczyniając się do istotnych zmian w specyfikacji EJB (ang. Enterprise JavaBeans) oraz JPA (ang. Java Persistence API), doprowadzając je do stanu, w którym są obecnie znane; EclipseLink, główna konkurencja technologii Hibernate, wspierająca standardy JPA, JAXB (ang. Java Architecture for XML Binding), JCA (Java Connector Architecture) i SDO (Service Data Objects); biblioteka Apache OpenJPA, która może być używana jako samodzielna warstwa trwałości POJO (ang. Plain Old Java Object) lub zintegrowana z dowolnymi platformami zgodnymi z Java EE (ang. Enterprise Edition); oraz technologia DataNucleus zapewniająca dostęp do rozproszonych systemów baz danych takich jak HBase i Cassandra, graficznych systemów baz danych jak Neo4j, arkuszów kalkulacyjnych w formatach Excel i OpenDocument, danych zapisywa-

nych w formacie JSON dla Amazon i Google Storage czy baz danych w formacie JSON pokroju MongoDB [4]. Biblioteki te zostały wybrane ze względu na popularność, badaną poprzez liczbę powiązanych tematów w wyszukiwarkach Google i Google Scholar oraz serwisów Github i Stack Overflow. Uzyskane wyniki obrazują częstotliwość wykorzystywania badanych technologii, co przekłada się na ich popularność i łatwość w dostępie do informacji oraz wsparcia technicznego.

2. Analiza literatury

Niniejszy rozdział zawiera przegląd renomowanych i wiarygodnych źródeł, w oparciu, o które powstała treść merytoryczna artykułu. Zebrane publikacje zawierają informacje dotyczące dokładnego opisu badanych technologii wraz ze sposobami oraz wynikami ich porównania. Pozwoliły one na zebranie niezbędnej wiedzy wykorzystanej podczas przeprowadzanych badań.

Artykuł „Object-Relational Mapping as a Persistence Mechanism for Object-Oriented Applications" [1] opisuje proces tworzenia aplikacji obiektowych poprzez zwrócenie uwagi na jego kluczowy element – mapowanie obiektowo-relacyjne. Publikacja ta zawiera kompleksowy opis całego mechanizmu i zwraca uwagę na najczęściej pojawiające się problemy, na które każdy programista powinien zwrócić szczególną uwagę. Artykuł ten pozwolił określić jakie elementy powinny znaleźć się w pracy dotyczącej badania bibliotek mapowania obiektowo-relacyjnego, co przełożyło się na sposób organizacji i wykonania części teoretycznej i praktycznej niniejszej pracy.

Publikacja naukowa pod tytułem „Performance Evaluation of Java Based Object Relational Mapping Tool" [2] porusza tematykę badania wydajności wykonywania zapytań wykorzystując szkielety programistyczne mapowania obiektowo-relacyjnego dla języka programowania Java. Autor publikacji poddał badaniom trzy najpopularniejsze narzędzia ORM takie jak Hibernate, Ebean oraz Toplink. Badanie polegało na wykonaniu szeregu podzapytań do bazy danych MySQL oraz zmierzeniu czasu ich wykonania. Badania dla wszystkich szkieletów programistycznych zostały wykonane na tej samej platformie. Informacje dotyczące sposobu przeprowadzenia badań oraz wykorzystania bazy danych MySQL stanowiły wzór testów przeprowadzonych w niniejszej pracy.

Następna pozycja literaturowa to artykuł naukowy pod tytułem „Analiza porównawcza technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java" [3]. Celem wyżej wymienionego artykułu jest porównanie cech, możliwości oraz wydajności technologii mapowania obiektowo-relacyjnego takich jak Hibernate i Oracle TopLink. Wszystkie testy wydajnościowe zostały przeprowadzone na wyszczególnionej platformie, a wykorzystana baza danych to Oracle 10g. Każdy pomiar powtarzany był czterokrotnie, a końcowy wynik został uśredniony. Na potrzeby testów baza danych została wypełniona przykładowymi danymi. Zostało wyszczególnione dziesięć operacji na bazie danych, które modyfikują, zapisują, odczytują oraz usuwają dane. Szczegółowe opisy przeprowadzonych testów zawierają informacje pozwalające na wydajny i efektywny sposób przeprowadzenia badań wydajności bibliotek ORM.

Publikacja o tytule „Object-Relational Mapping Tools and Hibernate" [4] zajmuje się tematyką narzędzi mapowania obiektowo-relacyjnego, głównie skupiając się na technologii Hibernate. Kompleksowo opisuje sposoby zastosowania opisywanych narzędzi, punktuje ich zalety oraz wady. Szkielet programistyczny Hibernate opisuje jako jedno z najważniejszych narzędzi tego typu. Wywiera on również ogromny wpływ na paradygmaty oraz konwencje JPA. Ponieważ istnieje od wielu lat, Hibernate jest według autorów najlepszym wyborem dla narzędzia ORM do użycia z językiem programowania Java dostępnym na rynku. Kompleksowe informacje zawarte w artykule stanowią wzór dla teoretycznego opisu badanych bibliotek i dostarczają bogaty zasób wiedzy dotyczący technologii Hibernate.

„Analysis of ORM Based JPA Implementations" [5] obejmuje przegląd najczęściej używanych dostawców JPA, w szczególności szkieletów programistycznych zapewniających obsługę JPA, takich jak Hibernate, EclipseLink, OpenJPA i DataNucleus. Analiza wydajności powyższych czterech implementacji JPA jest oparta na strukturze ORM, która w największym stopniu przyczyniła się do odkrycia wyzwań i zweryfikowania zagadnień programowania w języku Java. Publikacja doskonale wpasowuje się w tematykę niniejszej pracy, ze względu na informacje dotyczące badanych przez nią bibliotek.

3. Badane biblioteki mapowania obiektowo-relacyjnego

Technologie ORM dostarczane są jako samodzielne biblioteki, możliwe do wykorzystania w każdym projekcie. Narzędzia mapowania obiektowo-relacyjnego stały się niezbędnym elementem szkieletów programistycznych dla języka Java [5].

W tym rozdziale przedstawione zostaną cztery biblioteki ORM, które wykorzystywane są w języku programowania Java.

3.1. Hibernate ORM

Hibernate to biblioteka utworzona w 2001 roku przez przedsiębiorstwo Red Hat, oparta na licencji GNU Lesser General Public License (LGPL). Hibernate odwzorowuje klasy języka programowania Java na tabele bazy danych oraz typy danych Java na typy danych SQL. Warstwa biblioteki znajduje się pomiędzy kodem źródłowym aplikacji, a serwerem bazy danych, dzięki czemu obsługuje wszystkie prace związane z utrwalaniem danych w oparciu o odpowiednie mechanizmy i wzorce ORM [6].

Hasło „Hibernate ORM" zostało odnalezione w 1 790 000 wynikach wyszukiwarki Google. Google Scholar zawiera 8 980 publikacji związanych z tą technologią. Serwis Github przechowuje 64 987 repozytoria powiązane z biblioteką Hibernate ORM. Na forum

Stack Overflow zadano 88 211 pytania dotyczące tej biblioteki.

3.2. EclipseLink

EclipseLink to oparty o licencję Eclipse Public License projekt o otwartym kodzie źródłowym, opracowany przez firmę Eclipse Foundation. Biblioteka ta zapewnia rozszerzalną strukturę, która pozwala na interakcję z takimi usługami danych, jak bazy danych, usługi sieciowe, mapowanie Object XML (OXM) czy systemy informacji korporacyjnej (EIS). EclipseLink jest oparty na bibliotece TopLink [7].

Hasło „EclipseLink" w wyszukiwarce Google dało 503 000 wyników. W Google Scholar pojawiło się 1 160 publikacji związanych z tą technologią. Serwis Github zawiera 553 repozytoria powiązane z badaną technologią. Na forum Stack Overflow zadano 4914 pytań dotyczących biblioteki EclipseLink.

3.3. Apache OpenJpa

Apache OpenJPA to biblioteka o otwartym kodzie źródłowym, oparta na licencji Apache License 2.0. Narzędzie to opiera się na projekcie Kodo, który pierwotnie opracowany został przez firmę Solar Metric Inc w 2001 roku. Firma ta została następnie przejęta przez BEA Systems, gdzie biblioteka Kodo został rozszerzona, aby móc implementować specyfikację JDO (Java Data Object) oraz JPA (Jakarta Persistence). W roku 2006 BEA Systems przekazał większą część kodu narzędzia Kodo firmie Apache Software Foundation pod nazwą OpenJPA. W maju 2007 r. OpenJPA przeszło z inkubatora do projektu najwyższego poziomu [8].

Hasło „Apache OpenJPA" zostało odnalezione w 103 000 wynikach wyszukiwarki Google. Google Scholar zawiera 6 110 publikacji związanych z tą technologią. Serwis Github przechowuje 124 repozytoria powiązane z biblioteką Apache OpenJPA. Na forum Stack Overflow zadano 1082 pytania dotyczące tej biblioteki.

3.4. DataNucleus

DataNucleus (wcześniej znany jako Java Persistent Objects JPOX) to projekt na licencji Apache 2.0, o otwartym kodzie źródłowym. Projekt JPOX rozpoczął się w 2003 r. i został ponownie uruchomiony jako DataNucleus w 2008 r., oferując przy tym o wiele większy zakres obsługiwanych magazynów danych [9].

Wyszukanie hasła „DataNucleus" dało 91 100 wyników w wyszukiwarce Google i 701 publikacji w usłudze Google Schoolar. Biblioteka DataNucleus jest częścią 146 repozytoriów w serwisie Github. Na forum Stack Overflow zadano 891 pytań z wiązanych z tą technologią.

4. Implementacja aplikacji testowych

Podczas implementacji aplikacji testowych wykorzystano system zarządzania relacyjnymi bazami danych MySQL 8.0 [10]. Baza danych zawiera 14 tabel powiązanych relacjami. W bazie danych występują relacje jeden do jednego, jeden do wielu oraz wiele do wielu z tabelami łączącymi. Tabele zawierają dane w postaci tekstowej, liczbowej oraz pliki w formacie BLOB.

Aplikacja testowa została zaimplementowana w języku programowania Java 11 [11] przy wykorzystaniu narzędzia automatyzującego budowanie projektu Maven 3.6.3 oraz szkielet programistyczny Spring Boot 2.5.4 [12]. Dla każdej z badanych technologii został utworzony moduł zawierający zależności do bibliotek badanych szkieletów programistycznych mapowania obiektowo-relacyjnego. Moduły zostały zintegrowane z projektem Spring Boot Starter Data JPA. Dla każdej aplikacji zdefiniowano obiekty domenowe odzwierciedlające tabele z bazy danych. Utworzono 14 klas zawierających adnotację @Table, @Entity oraz adnotacje umożliwiające tworzenie relacji pomiędzy encjami. Dla relacji wiele do wielu wykorzystano adnotację @ManyToMany oraz @JoinTable, które pozwalają na utworzenie dodatkowej tabeli łączącej zrelacjonowane encje. Zaimplementowane aplikacje wykorzystują pojedynczy wątek oraz zachowują transakcyjność wykonywanych operacji.

Implementacja każdej z bibliotek pozwoliła na analizę poziomu skomplikowania integracji badanych technologii ze szkieletem programistycznym Spring.

Biblioteka Hibernate w bardzo prosty sposób integruje się ze technologią Spring. Bogata dokumentacja pozwoliła znaleźć odpowiedź na każdy pojawiający się problem, a intuicyjność kodu przyspieszyła proces tworzenia modułu przeznaczonego dla tej technologii.

Szkielet programistyczny Spring wspiera integrację z biblioteką EclipseLink, przez co konfiguracja tej technologii jest względnie prosta. Warto natomiast zwrócić uwagę na nieaktualną oraz niekompletną dokumentację. Podczas implementacji, na stronie producenta nie znaleziono przykładów oraz wymagań stawianych przez technologię EclipseLink podczas integracji ze szkieletem programistycznym Spring. Głównymi źródłami wiedzy, wykorzystanymi podczas konfigurowania biblioteki oraz projektowania w niej operacji bazodanowych, były niezależne serwisy dostępne w Internecie oraz dokumentacja szkieletu Spring.

Integracja ze szkieletem programistycznym Spring oraz samo wykorzystanie biblioteki DataNucleus okazało się problematyczne. Głównym problemem była uboga dokumentacja oraz brak przykładów dostępnych w Internecie. Podczas konfiguracji oraz tworzenia zapytań wymagana była analiza bibliotek technologii DataNucleus oraz dostępnych klas implementujących standard JPA.

Również w przypadku technologii OpenJPA występuje trudność ze znalezieniem aktualnej dokumentacji pozwalającej na sprawną integrację środowisk. Dodatkowa utrata wsparcia ze strony Spring znacząco utrudnia implementację.

5. Metodyka badań

Dla każdej z testowanych technologii zaprojektowano aplikacje w języku Java z wykorzystaniem szkieletu programistycznego Spring Boot. Każda z utworzonych

aplikacji miała na celu wykonywanie operacji na tej samej bazie danych z wykorzystaniem ocenianych bibliotek. Aplikacje są do siebie jak najbardziej zbliżone i oddzielne dla każdej z technologii. W ramach porównania przygotowano zestaw operacji odpowiedzialnych za odczyt, tworzenie, aktualizację oraz usuwanie rekordów z bazy danych z wykorzystaniem badanych bibliotek mapowania obiektowo-relacyjnego. Dla każdej operacji wykonano serię dziesięciu prób, a następnie uśredniono otrzymane rezultaty. Następnie rezultaty dotyczące konkretnej metody i wykorzystanej biblioteki zostały porównane z pozostałymi wynikami.

Do zmierzenia czasu wykorzystano podejście programowania zorientowanego aspektowo (AOP), umożliwiającego przechwycenie działań wskazanych metod, poprzez uruchomienie określonego fragmentu kodu przed oraz po logice wykonującej się w przechwytywanych metodach. Implementację tego rozwiązania umożliwił wykorzystywany szkielet programistyczny Spring Boot, dający dostęp do adnotacji @Aspect oraz @Around. Do badań wykorzystano urządzenia o specyfikacji przedstawionej w tabeli 1.

Tabela 1: Dane środowiska testowego

Nazwa	Wartość
System operacyjny	Windows 10
Procesor	Intel Core i5 6300HQ
Pamięć operacyjna	16GB DDR4
Dysk	SSD 500GB

Dla operacji odczytu przygotowano trzy zapytania. Każde z zapytań wykorzystuje inne funkcje pozwalające na wyszukiwanie danych. Pierwsze zapytanie skupia się na wyszukaniu danych bazując na pojedynczej tabeli. W drugim zapytaniu skupiono się na wyszukaniu danych z tabel powiązanych relacją. Trzecie zapytanie wykorzystuje funkcję agregujące. Na listingu 1 – 3 przedstawiono wykorzystane zapytania JPQL podczas badania operacji odczytu.

Listing 1: Operacja odczytu danych z pojedynczej tabeli

```
@Query("select v from Vote v " +
        "where v.post is not null")
Set<Vote> findAllPostVotes();
```

Listing 2: Operacja odczytu danych wykorzysujące zewnętrzne złączenie dwóch tabel

```
@Query("select p from Post p " +
        "left join p.answers a " +
        "where a.id is null")
Set<Post> findAllWithoutAnswers();
```

W celu wykonania badania wykonania operacji zapisu nowych rekordów przygotowano zestaw tysiąca, 5 tysięcy oraz 10 tysięcy danych. Badanie polegało na utworzeniu oraz zapisaniu przygotowanej serii danych do tabeli.

Dla operacji aktualizacji danych przygotowano zestaw tysiąca, 5 tysięcy oraz 10 tysięcy danych. Operacja ma na celu zaktualizowanie danych w dwóch tabelach powiązanych relacją jeden-do-jednego. Przykładowa operacja, umożliwiająca zarówno aktualizację jak i zapis danych przedstawiono na listingu 4.

Listing 3: Operacja odczytu danych wykorzystująca funkcję agregującą

```
@Query("select u from User u " +
        "inner join u.userDetails ud " +
        "inner join u.votes v " +
        "where ud.gender = :gender " +
        "group by v.user " +
        "having count(v.user) >= 10")
Set<User> findAllByGenderWithMin10Vote(String gender);
```

Listing 4: Operacja umożliwiająca zapis oraz aktualizację danych

```
@Transactional
@Override
public void createOrUpdateAll(Set<User> users) {
    userRepository.saveAll(users);
}
```

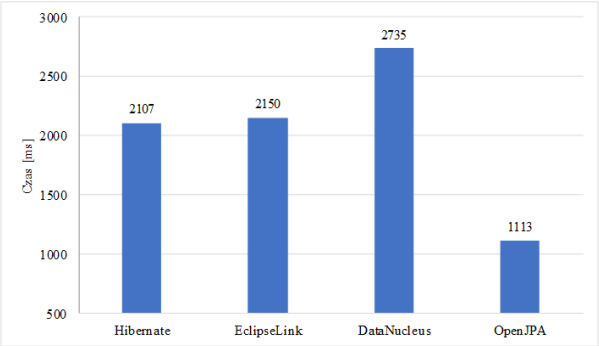
Dla operacji usuwania zaplanowano zestaw instrukcji umożliwiający usunięcie rekordu oraz relacyjnie powiązanych z nią rekordów z innych tabel. Przygotowano trzy instrukcje mające na celu usunięcie 100, 500 oraz tysiąca rekordów. Na listingu 5 przedstawiono instrukcję wykorzystaną podczas usuwania danych.

Listing 5: Operacja usunięcia rekordów wraz z relacyjnie powiązanymi danymi z innych tabel

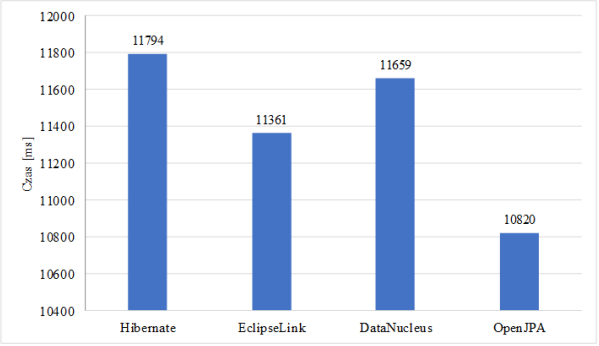
```
@Transactional
@Override
public void deleteAll(Set<User> users) {
    final var userIds = users.stream()
            .map(User::getId)
            .collect(Collectors.toSet());
    postService.deleteAllPostsByUsersIds(userIds);
    voteService.deleteVotesByUsersIds(userIds);
    answerService.deleteAnswersByUsersIds(userIds);
    commentService.deleteCommentsByUsersIds(userIds);
    userDetailsService.deleteAllByIds(users);
    userRepository.deleteAll(users);
}
```

6. Wyniki badań

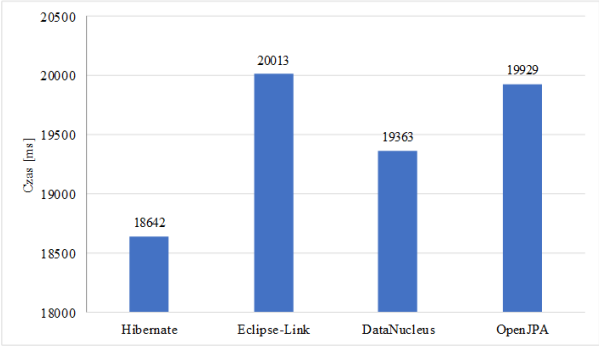
Tabela 2 przedstawia zbiorcze zestawienie otrzymanych rezultatów dla zapytań wyszukiwania rekordów. Czasy odczytu danych przedstawiono na rysunkach 1 – 3. Analizując otrzymane rezultaty można stwierdzić, że najbardziej optymalną technologią jest OpenJPA. Technologia ta wykazuje ponad dwukrotnie szybsze działanie wykonania operacji na pojedynczej tabeli względem innych technologii. Również dla operacji, które wymagają odczytu danych z tabel połączonych relacją wykazuje nieznaczną przewagę. Dla operacji wykorzystującej funkcję agregującą uzyskała nieznacznie wolniejszy czas, jednak otrzymany rezultat nie odbiega od uzyskanego przez pozostałe szkielety programistyczne. Warto wyróżnić również technologię Hibernate, która względem wyszukiwania na pojedynczej tabeli nie ustępuje technologii EclipseLink oraz DataNucleus, natomiast wykazuje szybsze działanie dla operacji wykorzysującej funkcje agregujące.



Rysunek 1: Średni czas odczytu danych z pojedynczej tabeli ze zbioru 200 tyś. rekordów.



Rysunek 2: Średni czas odczytu danych z wykorzystaniem relacji ze zbioru 500 tyś. rekordów.

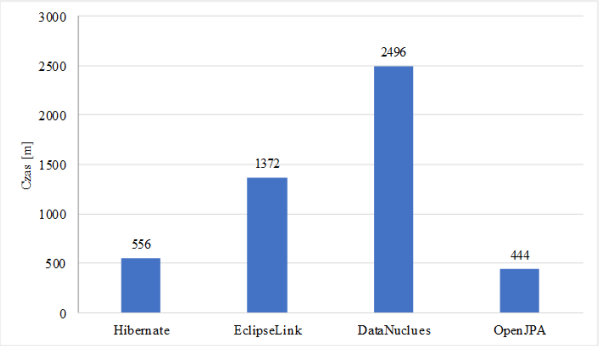


Rysunek 3: Średni czas odczytu danych z wykorzystaniem funkcji agregującej ze zbioru 500 tyś. rekordów.

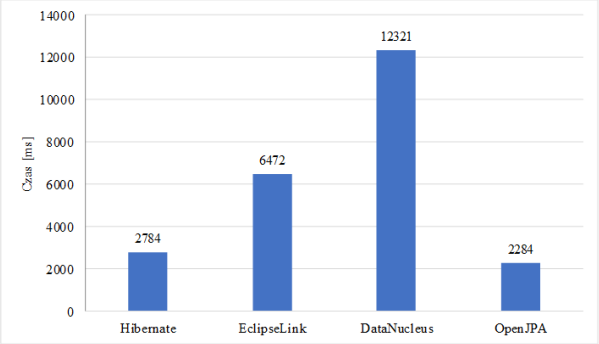
Tabela 2: Średni czas odczytu danych w milisekundach

Seria / Tech.	Zapytanie nr 1	Zapytanie nr 2	Zapytanie nr 3
Hibernate	2107 ms	11794 ms	**18642 ms**
EclipseLink	2150 ms	11361 ms	20013 ms
DataNucleus	2735 ms	11659 ms	19363 ms
OpenJPA	**1113 ms**	**10820 ms**	19929 ms

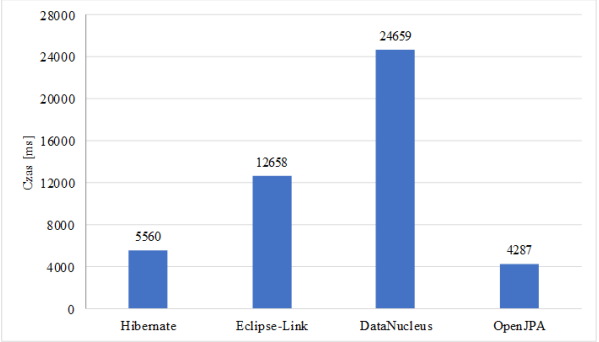
Analogiczne badanie przeprowadzono dla operacji zapisu danych. Rysunki 4-6 prezentują czasy wykonania badanych operacji, porównanie uzyskanych wyników przedstawiono w tabeli 3. Analizując otrzymane rezultaty można stwierdzić, że najwydajniejszą technologią w zakresie zapisu jest OpenJPA. Warto również zwrócić uwagę na technologię Hibernate, która dla małego zbioru otrzymuje czas porównywalny względem OpenJPA, natomiast różnice zwiększają się wraz ze wzrostem liczby operacji. Technologie EclipseLink oraz DataNucleus znacząco odbiegają od wcześniej wspomnianych technologii wykazując kilkukrotnie większy czas realizacji zadania zapisu.



Rysunek 4: Średni czas zapisu 1 tyś. rekordów.



Rysunek 5: Średni czas zapisu 5 tyś. rekordów.



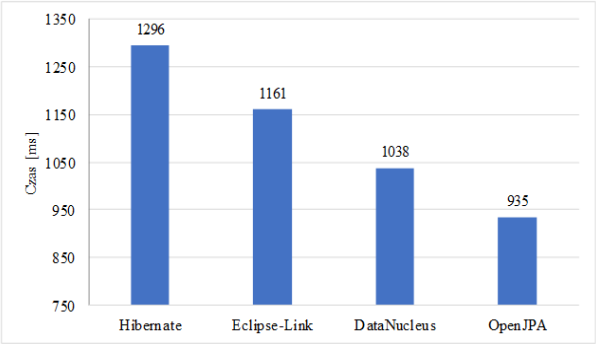
Rysunek 6: Średni czas zapisu 10 tyś. rekordów.

Tabela 3: Średni czas zapisu danych w milisekundach

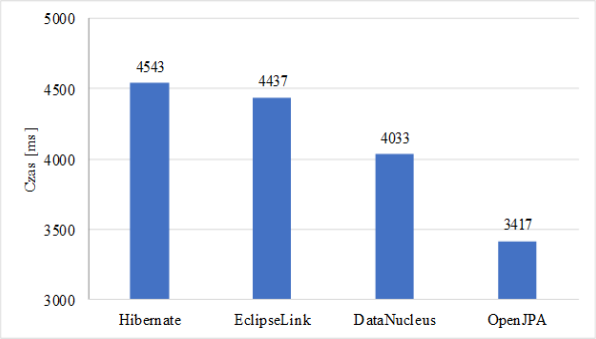
Seria / Tech.	1 tyś. rekordów	5 tyś. rekordów	10 tyś. rekordów
Hibernate	556 ms	2784 ms	5560 ms
EclipseLink	1372 ms	6472 ms	12658 ms
DataNucleus	2496 ms	12321 ms	24659 ms
OpenJPA	**444 ms**	**2284 ms**	**4287 ms**

Wyniki uzyskane podczas badania wydajności aktualizacji danych w bazie przedstawiono na rysunkach 7- 10. Tabela 4 przedstawia średni czas wykonania operacji aktualizacji danych. Analizując otrzymane

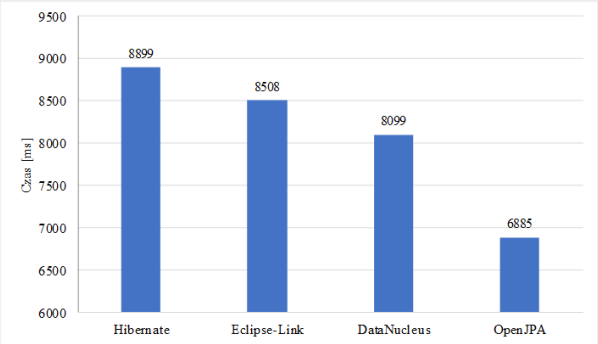
rezultaty można stwierdzić, że najwydajniejszą technologią w zakresie aktualizacji danych jest OpenJPA. Dla mniejszego zbioru danych różnice pomiędzy technologiami są nieznaczne, natomiast rosną one wraz ze wzrostem liczby wykonywanych operacji.



Rysunek 7: Średni czas aktualizacji 1 tyś. rekordów.



Rysunek 8: Średni czas aktualizacji 5 tyś. rekordów.

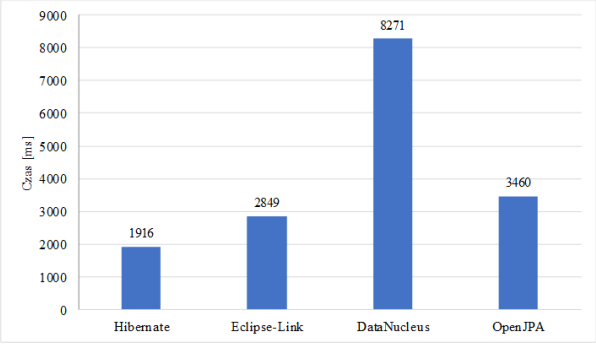


Rysunek 9: Średni czas aktualizacji 10 tyś. rekordów.

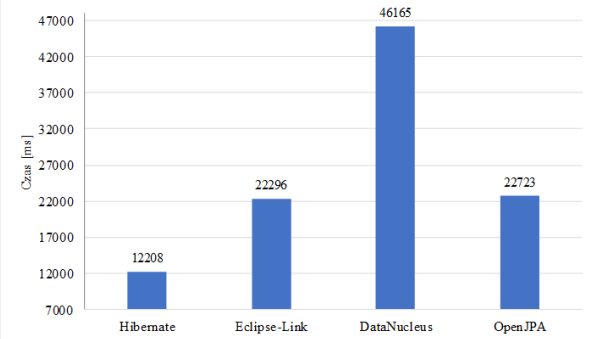
Tabela 4: Średni czas aktualizacji danych w milisekundach

Seria / Tech.	1 tyś. Rekordów	5 tyś. rekordów	10 tyś. rekordów
Hibernate	1296 ms	4543 ms	8899 ms
EclipseLink	1161 ms	4437 ms	8508 ms
DataNucleus	1038 ms	4033 ms	8099 ms
OpenJPA	**935 ms**	**3417 ms**	**6885 ms**

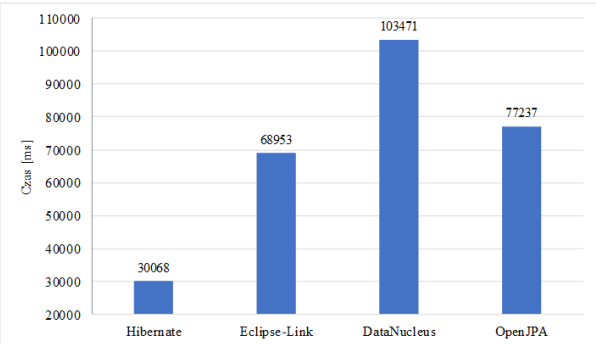
Badanie czasu wykonania zapytania usuwania rekordów wykazało, że najwydajniejszą technologią w zakresie usuwania danych jest Hibernate uzyskując minimum dwukrotnie lepszy czas względem innych technologii. Wyniki uzyskane przez poszczególne technologie zostały zaprezentowane na rysunkach 10 – 12. Dodatkowo Hibernate uzyskuje najmniejszy wzrost czasu przy wzroście liczby operacji do wykonania podczas usuwania. Tabel 5 zawiera zbiorcze zestawienie otrzymanych rezultatów.



Rysunek 10: Średni czas usunięcia 100 rekordów.



Rysunek 11: Średni czas usunięcia 500 rekordów.



Rysunek 12: Średni czas usunięcia 1 tyś. rekordów.

Tabela 5: Średni czas usunięcia danych w milisekundach

Seria / Tech.	1 tyś. Rekordów	5 tyś. rekordów	10 tyś. rekordów
Hibernate	**1916 ms**	**12208 ms**	**30068 ms**
EclipseLink	2849 ms	22296 ms	68953 ms
DataNucleus	8271 ms	46165 ms	103471 ms
OpenJPA	3460 ms	22723 ms	77237 ms

7. Wnioski

Wykorzystując wybrane materiały oraz publikacje, dokonano przeglądu czterech wykorzystywanych w szkielecie programistycznym Spring technologii ORM:

Hibernate ORM, EclipseLink, Apache OpenJPA oraz DataNucleus. Następnie wykonano praktyczne porównanie wydajności wybranych technologii.

Każda z badanych bibliotek wykorzystuje adnotacje JPA, przez co sam kod reprezentujący encje, serwisy i repozytoria jest do siebie bardzo podobny w każdym z omawianych przypadków. Z tego też powodu na poziom skomplikowania korzystania z danej biblioteki najbardziej wpływał proces konfiguracji narzędzia. Zdecydowanie najlepiej pod tym względem wypadła biblioteka Hibernate, która ze wszystkich narzędzi najlepiej integruje się ze szkieletem programistycznym Spring, oraz posiada najbogatszą dokumentację techniczną wraz z najliczniejszą, wspierającą się społecznością programistów wykorzystujących to narzędzie. Najgorzej natomiast wypadła biblioteka DataNucleus oraz Apache OpenJPA. Obie biblioteki nie posiadają wsparcia ze strony szkieletu programistycznego Spring, nie posiadają aktualnej dokumentacji, która jest również zbyt uboga, by umożliwić sprawną integrację środowisk.

W przypadku operacji wyszukiwania przeprowadzanej na jednej tabeli oraz na tabeli połączonej relacjami, najlepsza okazała się biblioteka Apache OpenJPA. Podczas wykorzystania funkcji agregujących prym wiodła technologia Hibernate ORM. Ogólnie podczas operacji wyszukiwania najlepiej wypadła biblioteka Apache OpenJPA, natomiast najgorzej biblioteka DataNucleus.

Badania operacji zapisu zaowocowały następującym wnioskiem – narzędzie Hibernate ORM jest najefektywniejsze w przypadku małych zbiorów danych, natomiast im większy jest ten zbiór, tym wydajniejsza okazuje się biblioteka Apache OpenJPA. Najgorzej podczas przeprowadzania zapisu danych wypadała technologia DataNucleus.

Test aktualizacji wykonany na małym i dużym zbiorze danych wykazał, że najwydajniejsza w tym przypadku jest biblioteka Apache OpenJPA, jednak biblioteka DataNucleus niewiele jej ustępuje. Najgorzej podczas testów wypadło narzędzie Hibernate ORM.

Podczas usuwania danych prym wiódł Hibernate ORM, a zdecydowanie najgorzej wypadła biblioteka DataNucleus.

Najlepiej w części praktycznej wypadła biblioteka Apache OpenJPA, na drugim miejscu natomiast znajduje się technologia Hibernate ORM.

Na podstawie przeprowadzonych badań oraz zdobytych informacji można wysunąć następujący wniosek – w przypadku, gdy w tworzonym systemie najważniejszą rolę gra wydajność, najlepszą opcją jest wykorzystanie technologii Apache OpenJPA, ponieważ uzyskała ona najlepsze wyniki w przypadku operacji wyszukiwania, zapisu i aktualizacji danych, a w przypadku operacji usuwania danych niewiele ustępowała ona technologii Hibernate. Jeżeli jednak wydajność nie jest kluczowa dla działania tworzonej aplikacji, lepszym wyborem okazuje się być technologia Hibernate ORM, która przewyższa OpenJPA pod względem prostej konfiguracji, popularności oraz dostępu do bogatych zasobów wiedzy i doświadczonej społeczności.

Literatura

[1] J. M. Barnes, Object-relational mapping as a persistence mechanism for object-oriented applications, Macalester College, 2007.

[2] S. N. Bhatti, Z. H. Abro, F. Rufabro, Performance evaluation of java based object relational mapping tool, Mehran University Research Journal of Engineering and Technology, 32(2) (2013) 159-166.

[3] P. Błoch, M. Wojciechowski, Analiza porównawcza technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java. XIII Konferencja PLOUG: Systemy informatyczne. Projektowanie, implementowanie, eksploatowanie, Zakopane, 2007.

[4] B. B. Correa, Y. Wang, E. Zimanyi, Object-relational mapping tools and Hibernate, Universite libre de Bruxelles, 2017.

[5] N. Dhingra, Analysis of ORM based JPA Implementations, University of Ottawa, 2017.

[6] C. Bauer, K. Gavin, G. Gary, Java Persistence. Programowanie aplikacji bazodanowych w Hibernate. Wydanie II, Helion, 2016.

[7] Dokumentacja techniczna biblioteki ORM Eclipselink https://www.eclipse.org/eclipselink/documentation, [22.10.2021].

[8] Dokumentacja techniczna biblioteki ORM Apache OpenJPA, http://openjpa.apache.org/documentation.html, [22.10.2021].

[9] Dokumentacja techniczna biblioteki ORM DataNucleus, https://www.datanucleus.org/products/accessplatform_6_0/, [22.10.2021]

[10] K. Appigatla, MySQL 8 Cookbook, Packt Publishing, 2018.

[11] J. Bloch, Java. Efektywne programowanie. Wydanie III, Helion, 2018.

[12] S. Raemaekers, A. Van Deursen, J. Visser, The maven repository dataset of metrics, changes, and dependencies, 10th Working Conference on Mining Software Repositories (2013) 221-224.