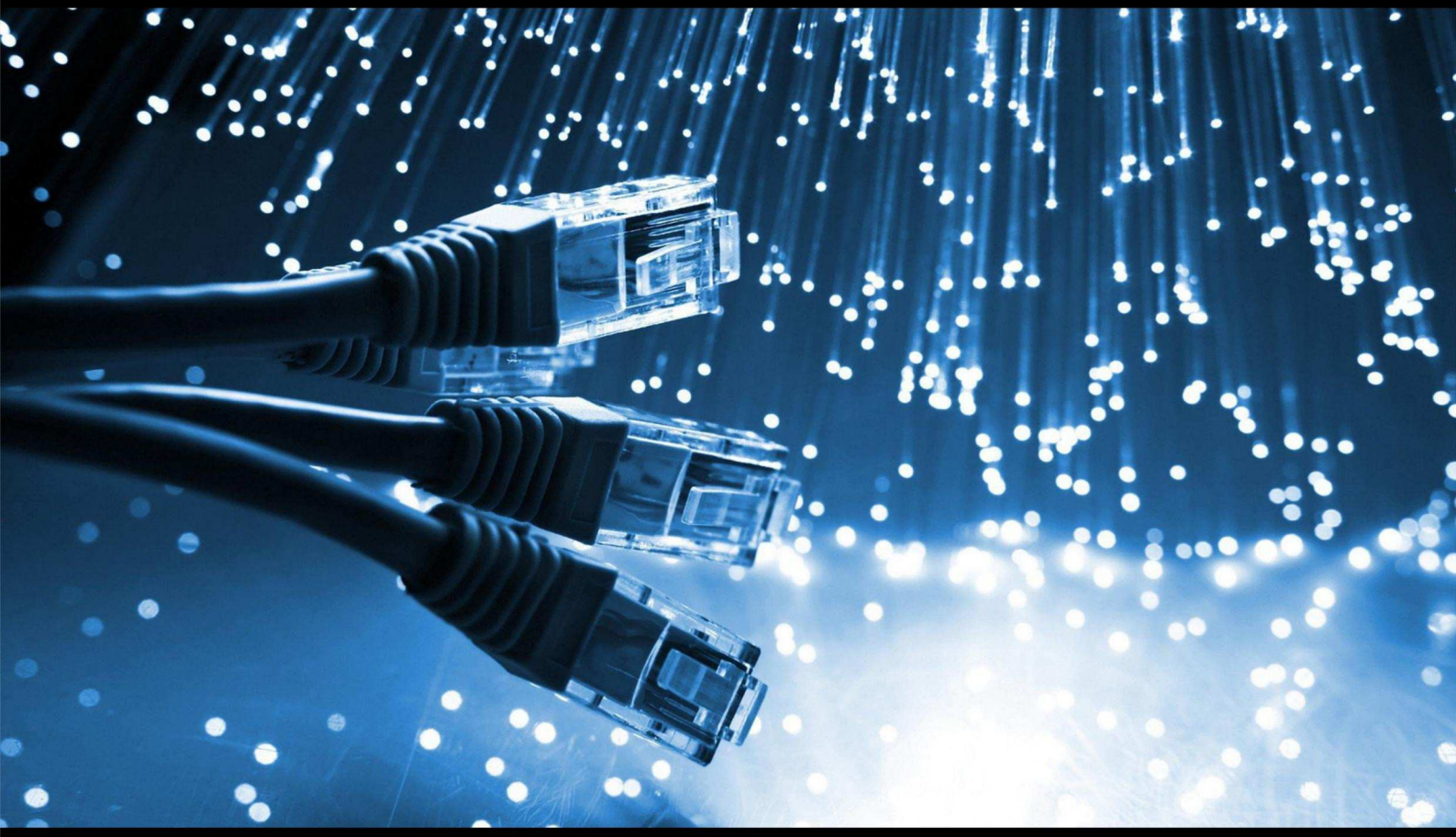
JCSI

Journal of Computer Sciences Institute

Volume 17/2020



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., Tomasz Nowicki
dr inż. Dariusz Gutek
dr inż. Maria Skublewska-Paszkowska
dr inż. Elżbieta Miłosz
dr Mariusz Dzieńkowski
dr inż. Tomasz Szymczyk
dr inż. Sławomir Przyłucki
dr inż. Piotr Muryjas
dr Adam Kiersztyn
dr inż. Jakub Smołka
dr Paweł Powroźnik
dr inż. Jacek Kęsik
dr Marcin Barszcz
dr hab. inż. Dariusz Czerwiński, prof. PL

Skład komputerowy:
Monika Kaczorowska
e-mail: m.kaczorowska@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:

Tomasz Nowicki
Dariusz Gutek
Maria Skublewska-Paszkowska
Elżbieta Miłosz
Mariusz Dzieńkowski
Tomasz Szymczyk
Sławomir Przyłucki
Piotr Muryjas
Adam Kiersztyn
Jakub Smołka
Paweł Powroźnik
Jacek Kęsik
Marcin Barszcz
Dariusz Czerwiński

Computer typesetting:
Monika Kaczorowska
e-mail: m.kaczorowska@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 326–331
Received: 05 July 2020
Accepted: 24 September 2020

Impact of the persistence layer implementation methods on application performance

Wpływ metod implementacji warstwy persystencji na wydajność aplikacji

Kamil Siebyła*, Maria Skublewska-Paszkowska

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

Abstract

There are various methods for creating web applications which have different levels of performance. The way the data access will be programmed at a specific endpoint, therefore, determines the performance of the entire application. There are many programming methods that are often time-consuming to implement. This paper presents a comparison of the available methods of handling the persistence layer in relation to the efficiency of their implementation. There are few methods for Entity Framework enviroment : Linq To Entity, Explicite Loading, Eager Loading, Raw SQL oraz Stored Procedure. While executing particular test scenarios, it was found that working on pure sql code in the case of working with the persistence layer is more efficient than using Object-Relational Mapper.

Keywords: time performance of the queries; entity framework core; sql

Streszczenie

Istnieją różne metody tworzenia aplikacji internetowych. Każda z tych metod charakteryzuje się różnym poziomem wydajności. Sposób, w jaki zostanie zaprogramowany dostęp do danych na konkretnym punkcie końcowym, uwarunkowuje więc wydajność całej aplikacji. Niniejszy artykuł przedstawia porównanie dostępnych sposobów obsługi warstwy persystencji w stosunku do wydajności ich implementacji. Sposobami tymi w środowisku Entity Frameworka są: Linq To Entity, Explicite Loading, Eager Loading, Raw SQL oraz Stored Procedure.. Wykonując poszczególne scenariusze testowe ustalono, że działanie na czystym kodzie sql w przypadku pracy z warstwą persystencji jest wydajniejsze niż korzystanie z maperów obiektowo relacyjnych (ang. *Object-Relational Mapper*).

Słowa kluczowe: wydajność czasowa zapytań; entity framework core; sql

*Corresponding author

Email address: **kamil.siebyla@gmail.com** (K. Siebyła)

1. Wstęp

Przez kilka ostatnich lat oblicze tworzenia aplikacji internetowych przeszło prawdziwą rewolucję. Klienci końcowi rzeczonych aplikacji oczekują wysokiej jakości wizualnej, jak i wydajności. Warstwa persystencji w przypadku tej drugiej właściwości odgrywa kluczową rolę. Sytuacja, w której niepoprawnie zostaną zaimplementowane mechanizmy związane z obsługą bazy danych, może mieć negatywne skutki dla działania aplikacji. Negatywnie może to również wpłynąć na wydajność warstwy wizualnej, bowiem w momencie, kiedy brakuje danych dla kontrolek mogą one zachowywać się w sposób niepożądany. Implementacja określonych rozwiązań i ich rozbudowa od strony programistycznej jest zróżnicowana pod kątem trudności implementacji i zarządzania. Programista często jest odgórnie ograniczony czasowo, przez co należy wybrać metody implementacji, które są wydajne dla danego typu aplikacji oraz które można w łatwy sposób rozbudować przy jednocześnie możliwie najkrótszym czasie tworzenia kodu. Badania dotyczące interfejsów ORM (ang. Object Relational Mapping) zostały poruszone w kilku pracach. Wykorzystywane są przez autorów tych prac różne technologie i podejścia, jednak celem badań poruszanym w tych publikacjach jest wydajność aplikacji. Autor artykułu pod tytułem „Comparsion of performance beetwen Raw SQL and Eloquent ORM in Laravel" [1] wykorzystuje do badań wydajności aplikacji internetowych framework języka PHP (ang. Hypertext Preprocessor), jakim jest Laravel. Bada trzy różne techniki obsługi bazy danych, Eloquent ORM, Query Builder oraz Raw SQL. Autor publikacji stawia tezę w swojej pracy, mówiącą, że Eloquent ORM intuicyjnie powinien być wolniejszy niż kod napisany w czystym SQL – co potwierdza wynikami przeprowadzonych badań. Kolejny artykuł nosi tytuł „Performance evaluation of java object-relational mapping tools" [2]. Autorzy publikacji skupiają się na porównaniu różnych narzędzi ORM w ekosystemie Javy. Porusza on bardzo obszernie wszystkie możliwe narzędzia badając ich wydajność wykorzystując odpowiednie zapytania. Autorzy sugerują, że gdy używane są standardy JPA (ang. Java Persistance API) wydajność konkretnych narzędzi wzrasta. Potwierdza to fakt, iż Hibernate okazał się być najwydajniejszym narzędziem w środowisku Javy. Nie przypadkowo oparty jest on właśnie na JPA. Zdecydowanie najsłabiej wypadł Open JPA w testach zaprezentowanych przez autora publikacji.

Celem niniejszego artykułu jest porównanie metod implementacji warstwy persystencji oraz zbadanie ich wydajności przy użyciu autorskiego narzędzia testującego punkty końcowe API (Application Programing Interface) oraz autorskiej aplikacji internetowej, pełnią-

cej rolę warstwy wizualnej dla bazy danych. Ponadto analizie zostanie poddana trudność implementacji poszczególnych metod dostępu do danych w stosunku do rozmiaru badanej aplikacji.

W artykule postawiono hipotezę badawczą o brzmieniu: *Operacje na czystym kodzie SQL są wydajniejsze od operacji realizowanych przez Entity Frameowrk Core*.

2. Metody dostępu do danych w .NET

2.1. Entity Framework Core

Jest to kompaktowa, rozszerzalna, międzyplatformowa wersja klasycznego Entity Framework, który jest narzędziem pozwalającym na dostęp do warstwy danych aplikacji. Dzięki Entity Framework Core [3] istnieje możliwość pracy z bazą danych przy użyciu klasycznych obiektów. Oznacza to, że chcąc odwoływać się do bazy danych, choćby w klasycznym CRUD (ang. *Create Read Update Delete*), programista nie jest zmuszony dopisania złożonego kodu sql. Dzięki Entity Framework jest w stanie wykonywać operacje na bazie danych w taki sposób jakby pisał kod obiektowy. Deweloper korzysta z abstrakcji jaką oferuje Entity Framework Core. Bazowym pojęciem w tym przypadku jest model, który symuluje encje zawarte w bazie danych. Wraz z kontekstem zawartym w Entity Framework Core stanowi integralną część sesji połączenia z bazą danych.

2.2. Procedury Składowane

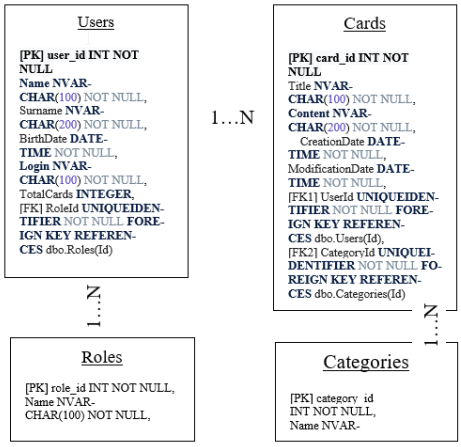
Procedury składowane są znane programistom od kilkudziesięciu lat. Pisanie procedur rozszerza nieco wachlarz możliwości warstwy danych, pod kątem zaawansowania zapytań. Dzięki napisaniu zaawansowanej procedury, programista jest w stanie wyciągnąć skomplikowany zestaw danych. Procedury posiadają parametry, które w praktyce są przekazywane przez kod serwerowy. Wydajność procedur w praktyce powinna prezentować wysoki poziom, gdyż przesyłane są parametry do procedury w jednym żądaniu. Dodatkowo nie jest przesyłany cały kod SQL, a jedynie nazwa procedury, a silnik bazodanowy mając nazwę wykona żądane operacje. Niniejszy artykuł będzie prezentował procedury składowane w środowisku napisane w języku Transact SQL w środowisku MSSQL firmy Microsoft [4].

3. Architektura środowiska pracy

Badania zostały przeprowadzone w środowisku .NET Framework z wykorzystaniem frameworka Entity Framework Core. Interfejs ten został poddany analizie w kontekście realizacji podstawowych operacji na bazie danych w zależności od liczby danych. Dodatkowo został wykorzystany skrypt napisany w Powershell [5], który testuje punkty końcowe serwisu REST (ang *Representational State Tranfer*) [6] mierząc czas wykonania żądania przy wygenerowanych wcześniej danych. Testy zostały przeprowadzone na sprzęcie o podanych poniżej parametrach: MSI GE70, 64 bitowy system operacyjny Windows 10 PRO, Procesor Intel(R) Core(TM) i5-4200M CPU @ 2.50 GHz, karta graficzna NVIDIA GeForce GTX 765M, 8 GB pamięci RAM DDR3 o częstotliwości 1600 MHz oraz dysku SSD o pojemności 128 GB. Baza danych została uruchomiona lokalnie przy użyciu konteneryzacji. W tym celu został wykorzystany popularny kontener o nazwie docker. Uruchomienie bazy danych w kontenerze sprowadzało się instalacji klienta dockera na lokalnej maszynie, pobraniu obrazu rzeczonej bazy danych a następnie wykonaniu kilku komend umożliwiających utworzenie kontenera z działającym pobranym wcześniej obrazem bazy danych.

4. Aplikacja testowa oraz narzędzia

Środowisko testowe składa się ze skonfigurowanej bazy danych, aplikacji internetowej, serwisu typu REST oraz skryptu Powershell. Aplikacja jest środowiskiem, które służy do testowania wydajności warstwy persystencji. Jest interfejsem, który jest wywoływany asynchronicznie przez skrypt Powershell w celu przetestowania wydajności konkretnych metod dostępu do danych. Na rysunku 1 schemat bazy danych aplikacji testowej.



Rysunek 1: Schemat bazy danych aplikacji testowej.

5. Scenariusze testowe

Scenariusze testowe zostały skonstruowane tak, aby jak najwierniej przedstawić wady i zalety konkretnych metod implementacyjnych. Tabele 1-4 przedstawiają scenariusze badawcze. W ostatniej kolumnie podana jest liczba rekordów z bazy danych. Znak „/" w poniższych tabelach oddziela liczbę obsługiwanych rekordów dla danej metody w tysiącach dla kolejnych prób.

Tabela 1: Scenariusze badawcze dla odczytu bez śledzenia zmian

Numer scenariusza badawczego	Rodzaj implementacji	Liczba obsługiwanych rekordów w tysiącach
1	L2E	10/60/160
2	Explicite Loading	10/60/160
3	Eager Loading	10/60/160
4	Raw SQL	10/60/160
5	Procedure	10/60/160

Tabela 2: Scenariusze badawcze dla odczytu z uruchomionym śledzeniem zmian

Numer scenariusza badawczego	Rodzaj implementacji	Liczba obsługiwanych rekordów w tysiącach
6	L2E	10/60/160
7	Explicite Loading	10/60/160
8	Eager Loading	10/60/160
9	Raw SQL	10/60/160
10	Procedure	10/60/160

Tabela 3: Scenariusze badawcze dla zapisu bez śledzenia zmian

Numer scenariusza badawczego	Rodzaj implementacji	Liczba obsługiwanych rekordów w tysiącach
11	Kontekst Globalny	3/10/12
12	Kontekst Lokalny	3/10/12
13	Automatyczna detekcja zmian	3/10/12

Tabela 4: Scenariusze badawcze dla zapisu z uruchomionym śledzeniem zmian

Numer scenariusza badawczego	Rodzaj implementacji	Liczba obsługiwanych rekordów w tysiącach
14	Kontekst Globalny	3/10/12
15	Kontekst Lokalny	3/10/12
16	Automatyczna detekcja zmian	3/10/12

Każdy z powyższych scenariuszy został wykonany jednokrotnie przy użyciu przygotowanej infrastruktury technicznej.

6. Wykonywane operacje podczas realizacji scenariuszy testowych

Podczas realizacji scenariuszy badawczych wyróżnione zostały dwa typy operacji zapisu oraz odczytu.

6.1. Operacje odczytu

Entity Framework Core udostępnia dwie główne grupy metod, które realizują operacje odczytu. Pierwszą grupą rozwiązań są te, które, cechują się znacznym poziomem obiektowości oraz abstrakcji. Dla EF (ang. *Entity Framework*) nazywane są „Linq To Entities". Drugą grupą metod są rozwiązania implementujące język SQL. W środowisku testowym wykorzystywanym w niniejszej pracy jest to odpowiednio „EntitySQL" [7]. Każda z tych metod umożliwia konstruowanie złożonych zapytań do bazy danych. Różni je sposób w jaki dane z bazy są otrzymywane. Obydwie metody zwracają zmaterializowane dane. Oznacza to, że programista w kodzie otrzymuje instancję encji. Z punktu widzenia programisty jest to bardzo wygodne rozwiązanie. Jednakże L2E (ang. Linq To Entity) wprowadza dodatkową warstwę abstrakcji do zbioru wynikowego poprzez typy anonimowe, które w znaczący sposób upraszczają implementację mechanizmów związanych z encjami. Niesie to jednak za sobą również negatywne skutki w postaci niemożności tworzenia silnie dynamicznych oraz złożonych zapytań. EntitySQL z kolei zwraca dane w postaci kolekcji. Takie kolekcje jest trudniej przeszukiwać, a programista nie jest w stanie zbudować warstwy abstrakcji, która ułatwiłaby dostęp do zwróconych przez zapytanie danych.

6.2. Operacje zapisu

Operacje zapisu są realizowane przy udziale kontekstu jakim programista dysponuje w środowisku aplikacji. Każda instancja kontekstu posiada mechanizm śledzący zmiany (ang. *Change Tracker*). Przechowywane są w nim informacje o zmianach jakie zostały dokonane na instancji konkretnej encji. Kontekst śledzi encję i w momencie wywołania metody „SaveChangesAsync" zapisuje informacje do bazy danych. Domyślnie zapytania, które zwracają encje konkretnego typu, są śledzone. Z drugiej strony programista ma do dyspozycji zapytania, pozbawione śledzenia zmian. Kontekst umożliwia ręczne wyłączenie śledzenia zmian. Takie zapytania są szczególnie przydatne, kiedy dane zwracane przez zapytanie znajdują zastosowanie w elementach, w których są wykorzystywane jako dane tylko do odczytu. Istnieją dwa sposoby na wyłączenie tej funkcjonalności. Pierwszym z nich jest wyłączenie kontekstu globalnie z poziomu instalacji kontekstu. Drugim sposobem jest użycie metody „AsNoTracking" przy wykonaniu samego zapytania. Wartą nadmienia kwestią w przypadku mechanizmu śledzenia zmian w Entity Framework Core jest rozwiązywanie identyfikatorów encji (ang. *Identity Resolution*) [8]. Chodzi o to, że dopóki zapytanie jest śledzone, Entity Framework Core potrafi w momencie materializacji encji rozpoznać, czy dana encja jest śledzona i jeżeli tak w istocie jest, zwrócić instancję dokładnie tej śledzonej encji. Jeżeli w zbiorze wynikowym jest wiele wystąpień danej encji, to zostanie zwrócona ta sama instancja, tylko wielokrotnie. W przypadku braku śledzenia zmian za każdym razem będzie tworzona nowa instancja encji dla każdego wystąpienia jej w zbiorze wynikowym. Mechanizm ten jest kluczowy z punktu widzenia wydajności aplikacji. Zostanie zbadane jakie są różnice w wydajności dla konkretnych zbiorów danych z włączonym lub wyłączonym śledzeniem zmian. Dodatkowym pojęciem, jakie wiąże się z zapisem, jest współbieżność dostępu do danych. Sytuacja taka ma miejsce kiedy dwóch użytkowników próbuje zmodyfikować tą samą encję. Jeden użytkownik jest w trakcie modyfikacji, natomiast drugi zdążył zapisać już dane do bazy danych. Popularnym i zarazem najlepszym rozwiązaniem jest wprowadzenie pojęcia właściwości o nazwie „rowversion" [10]. Jest to po prostu dodatkowa kolumna w bazie danych, która przechowuje aktualną wersję wiersza. W środowisku SQL Server stworzono specjalny typ zmiennej o nazwie „rowversion". Jest to zmienna liczbowa, która przy

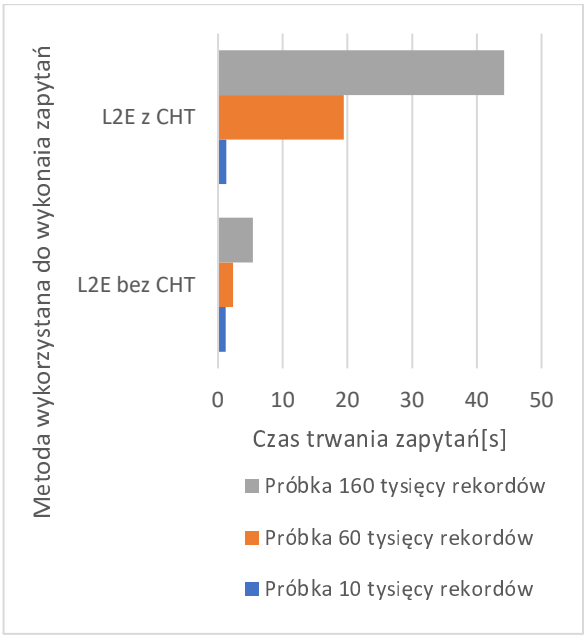
każdej modyfikacji wiersza inkrementuje swoją wartość. Dzięki temu mechanizmowi, kiedy jeden użytkownik zacznie modyfikować wiersz, podczas gdy inny zapisał nową wartość wierszu w czasie kiedy pierwszy wprowadzał modyfikację, użytkownik który modyfikował zawartość wiersza w momencie wywołania metody „SaveChanges" zostanie wyświetlony wyjątek „DbUpdateConcurencyException".

7. Porównanie graficzne wyników

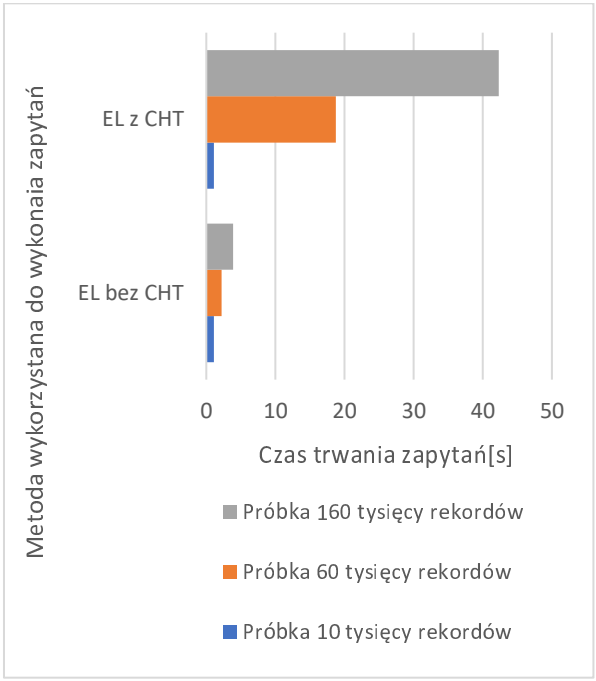
Każdy z powyższych scenariuszy został wykonany jednokrotnie przy użyciu przygotowanej infrastruktury technicznej. Wyniki jakie zostały otrzymane po wykonaniu konkretnych scenariuszy zostały przedstawione w tabelach: 1, 2, 3 oraz 4.

7.1. Wydajność odczytu

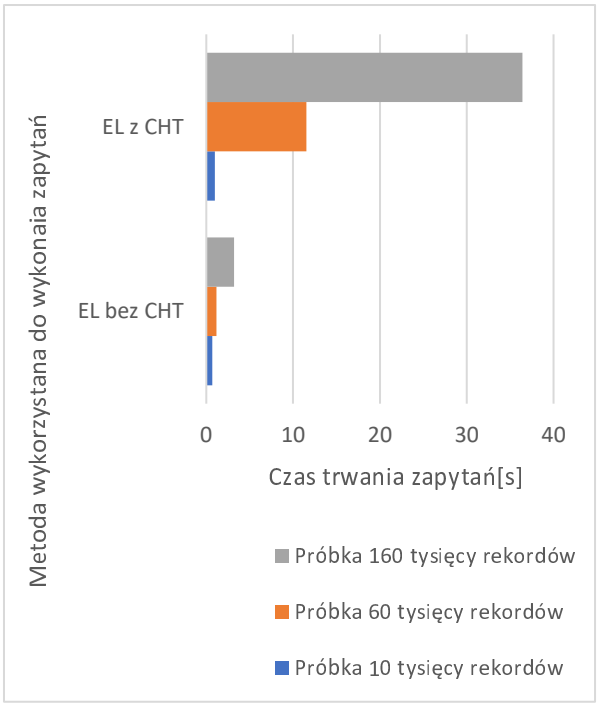
Na rysunkach 2-9 przedstawiono wyniki wykonania wszystkich scenariuszy testowych zawartych w tabelach 1-4. Rysunki te przedstawiają za pomocą wykresów słupkowych wartości czasowe jakie zostały zarejestrowane dla każdej z badanych metod.



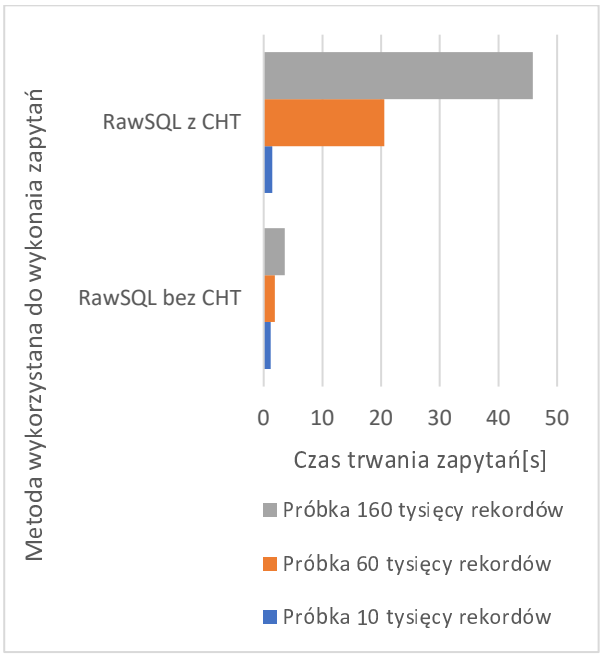
Rysunek 2: Porównanie wydajności odczytu dla L2E z włączonym (L2E z CHT) oraz wyłączonym śledzeniem zmian (L2E bez CHT)



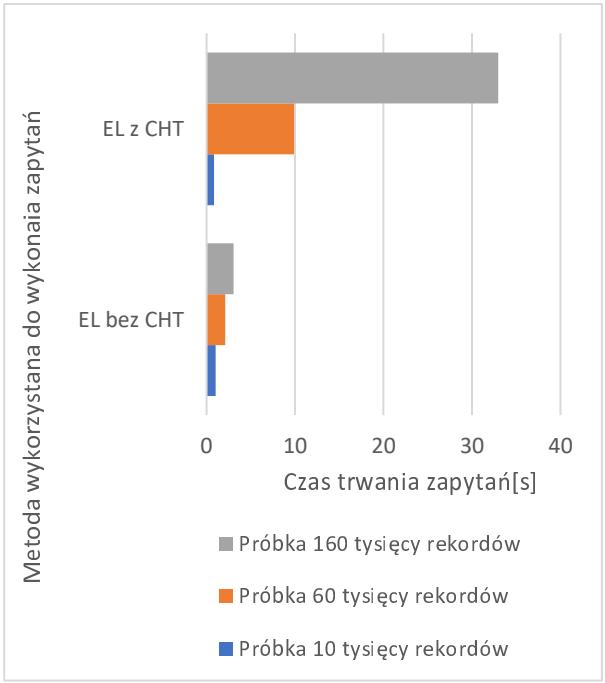
Rysunek 3: Porównanie wydajności odczytu dla Explicite Loading z włączonym oraz wyłączonym śledzeniem zmian



Rysunek 4: Porównanie wydajności odczytu dla Eager Loading z włączonym oraz wyłączonym śledzeniem zmian



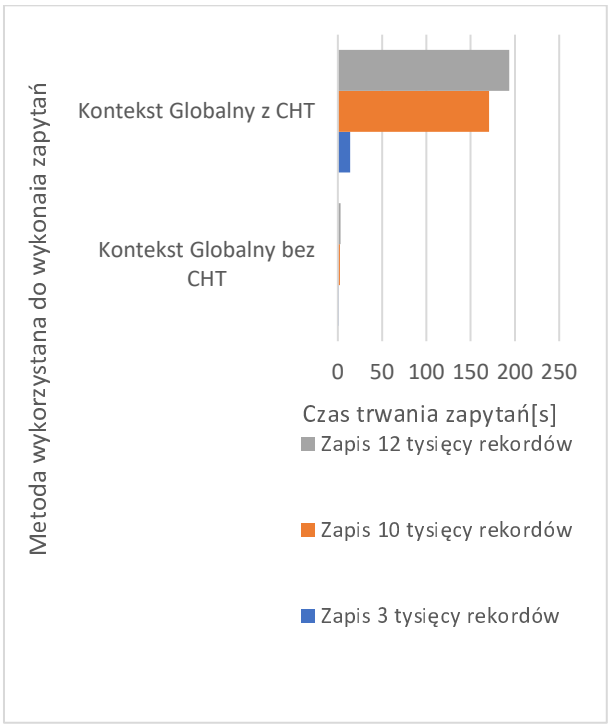
Rysunek 5: Porównanie wydajności odczytu dla Raw SQL z włączonym oraz wyłączonym śledzeniem zmian



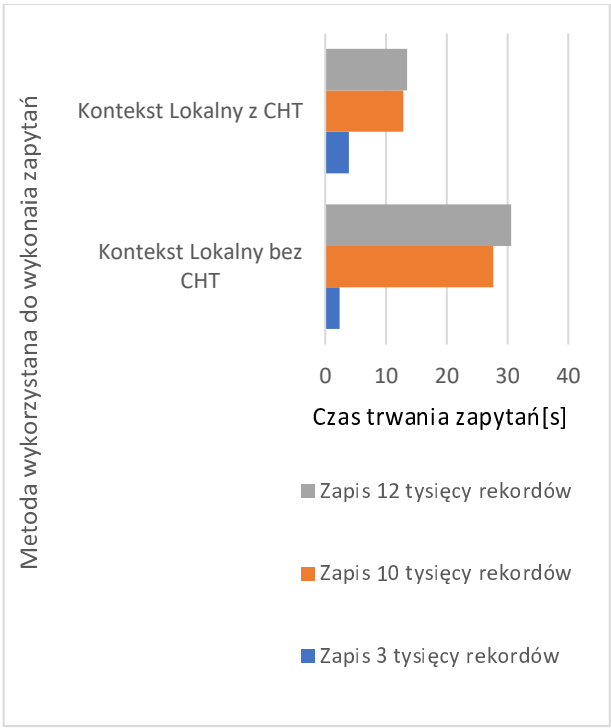
Rysunek 6: Porównanie wydajności odczytu dla procedury składowanej z włączonym oraz wyłączonym śledzeniem zmian

7.2. Wydajność zapisu

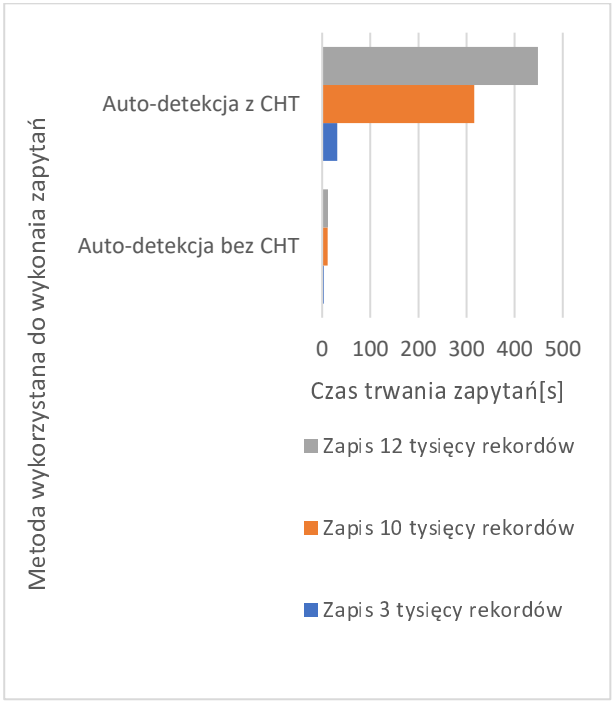
Rysunki 7-9 przedstawiają wydajność związaną z zapisem.



Rysunek 7: Porównanie wydajności zapisu dla kontekstu globalnego z włączonym oraz wyłączonym śledzeniem zmian



Rysunek 8: Porównanie wydajności zapisu dla kontekstu lokalnego z włączonym oraz wyłączonym śledzeniem zmian



Rysunek 9: Porównanie wydajności zapisu dla auto-detekcji zmian z włączonym oraz wyłączonym śledzeniem zmian

8. Wnioski

Przeprowadzone badania związane z monitorowaniem wydajności konkretnych metod implementacji warstwy persystencji pokazały, że czysty kod SQL efektywniej razi sobie z przetwarzaniem zapytań. Wyniki uzyskane w pierwszym eksperymencie (tabele: 1 i 2 oraz rysunki: 2-6) pozwalają zauważyć różnice pomiędzy odpowiednimi metodami implementacji Entity Framework Core, a kodem SQL. Spowodowane jest to tym, że Entity Framework Core jest obudowany warstwą abstrakcji w postaci obiektów klas implementujących dostęp do operacji na bazie danych. Dodatkowo możliwość manipulacji kontekstem oraz mechanizm śledzenia zmian powodują, że czasy operacji zapisu jak i odczytu wzrastają bądź zostają skrócone w zależności od zastosowanej techniki. Można zauważyć, że autodetekcja zmian znacząco wydłuża czas wykonania zapytań – zarówno dla zapisu jak i dla odczytu. Procedura składowana dla zestawu 160 tysięcy rekordów z włączoną detekcją zmian wykonała się w przybliżeniu w 33 sekundy. Jest to najlepszy wynik. Jednak ta sama procedura dla takiej samej porcji danych z wyłączoną detekcją zmian wykonała się w niecałe 10 sekund. Jest to kilka rzędów wielkości. Dla porównania metoda RawSQL użyta z Entity Framework wykonywała się średnio 10 sekund dłużej dla każdego z przypadków – co świadczy o lepszej wydajności procedury składowanej. Można więc powiedzieć, że im bliżej źródła danych programista operuje, tym efektywniej działają operacje związane z warstwą persystencji. Przeprowadzone badania (rys. 2-9) pozwalają potwierdzić, że postawiona teza *Operacje na czystym kodzie SQL są wydajniejsze od operacji realizowanych przez Entity Frameowrk Core została udowodniona.* Operacje na czystym kodzie sql są wydajniejsze niż narzędzia ORM jakimi dysponuje programista w codziennej pracy.

Literatura

[1] I. Jound, H. Halimi, M. Svahnberg, Comparsion of performance beetwen Raw SQL and Eloquent ORM in Laravel, Faculty of Computing Blekinge Institute of Technology SE-371 79 Karlskrona Sweden.

[2] H. Yousaf, Performance evaluation of java object-relational mapping tools, Georgia: University of Georgia, 2012.

[3] P. Anbazhagan, Mastering Entity Framework Core 2.0, Wydawnictwo Helion SA, Gliwice 2017.

[4] Oficjalna dokumentacja MSSQL, https://docs.microsoft.com/en-us/sql, [06.2020].

[5] Oficjalna dokumentacja PowerShell, https://docs.microsoft.com/en-us/powershell, [06.2020].

[6] G. Arorra, T. Dash, Building RESTful Web Services with .NET Core, Wydawnictwo Helion SA, Gliwice 2019.

[7] Dokumentacja EntitySQL, https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/entity-sql-overview, [06.2020].

[8] O. Mehboob, A. Khan, C# 7 i.NET Core 2.0 Programowanie wielowątkowych i współbieżnych aplikacji, Wydawnictwo Helion SA, Gliwice 2019.

[9] Dokumentacja Entity Framework, https://docs.microsoft.com/en-us/dotnet/framework/ data/adonet/ef, [06.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 332–338
Received: 29 July 2020
Accepted: 10 September 2020

Analysis of positioning errors of the GPS navigation receivers

Analiza błędów wyznaczania pozycji odbiorników nawigacyjnych systemu GPS

Łukasz Budzyński[a,*], Eligiusz Pawłowski[b]

[a] *Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland*
[b] *Department of Automation and Metrology, Lublin University of Technology, Nadbystrzycka 38A, 20-618 Lublin, Poland*

Abstract
The article compares the accuracy of receivers related to the geolocation process in the GPS system. To determine the accuracy of the receivers, a series of research experiments were conducted based on test cases. For the needs of the research, a research platform for collecting data from the GPS system was designed and constructed. The research results are presented in the form of tables and graphic charts.

Keywords: GPS; Navigation; Telecommunication

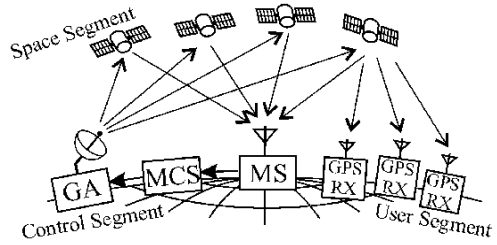
Streszczenie
W artykule porównano dokładność odbiorników związanych z procesem geolokalizacji w systemie GPS. W celu określenia dokładności odbiorników przeprowadzono serię eksperymentów badawczych w oparciu o przypadki testowe. Na potrzeby badań została zaprojektowana oraz skonstruowana platforma badawcza służąca do gromadzenia danych z systemu GPS. Rezultaty badań przedstawiono w postaci tabel oraz wykresów graficznych.

Słowa kluczowe: GPS; Nawigacja; Telekomunikacja

*Corresponding author
Email address: **lukasz.budzynski@pollub.edu.pl** (Ł. Budzyński)

1. Wstęp

Systemy nawigacji satelitarnej GNSS umożliwiają uzyskanie dokładnych informacji na temat pozycji badanego obiektu. Najbardziej popularnym systemem tego typu jest Amerykański system GPS, jest to system o zasięgu globalnym, składający się z 31 satelitów krążących po orbicie ziemskiej [1]. Podstawową zasadą działania systemu GPS jest nieustanny pomiar czasu wysyłanego przez satelitę do odbiornika GPS. W celu prawidłowego funkcjonowania segmentu kosmicznego konieczna jest konstelacja przynajmniej 24 satelitów, poruszających się po orbitach kołowych[1-3]. Satelity GPS rozmieszczone są na 6 orbitach oznaczonymi za pomocą liter A,B,C,D,E,F wzdłuż równika co 60° [1-2]. Rozmieszczenie satelitów na orbitach A,B,C,D,E,F umożliwia uzyskanie widoczności co najmniej 5 satelitów w każdym obszarze kuli ziemskiej [2, 4]. Na rysunku 1 przedstawiono konfigurację systemu GPS.

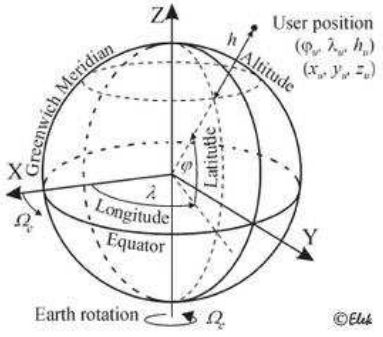


Rysunek 1: Konfiguracja systemu GPS, MMC –Główna stacja kontroli, MS - Stacja monitorująca, GA – Antena naziemna, GPS RX – Odbiornik użytkownika [1].

GPS składa się z trzech segmentów: segmentu kosmicznego, segmentu naziemnego oraz segmentu użytkownika [3]. Głównym zadaniem segmentu naziemnego jest monitorowanie segmentu kosmicznego. Stacje monitorujące przesyłają dane do głównej stacji kontrolnej, na ich podstawie wyznaczane są poprawki do segmentu kosmicznego[1]. Segment użytkownika składa się z odbiorników, jego celem jest odbiór sygnału wysyłanego za pośrednictwem segmentu kosmicznego i określenie pozycji odbiornika na ich podstawie [4-5].

2. Podstawy systemu GPS

Podstawową zasadą działania systemu GPS jest wyznaczenie pozycji satelity oraz odbiornika GPS w określonym układzie współrzędnych [5]. W systemie GPS współrzędne odbiornika oraz satelity przedstawione są w kartezjańskim układzie współrzędnych, znanym inaczej jako układ WGS84 [1, 6].



Rysunek 2: Układ współrzędnych WGS84 [1].

Początek układu współrzędnych znajduję się w centrum Ziemi, oś Z przebiega przez oś obrotu Ziemi, osie X oraz Y pokrywają się z płaszczyzną równikową. Oś X przechodzi przez południk zerowy [1]. Układ WGS84 jest zakotwiczony do osi Ziemi i obraca się razem z nią. W celu wyznaczenia pozycji odbiornika GPS satelita wysyła depeszę nawigacyjną w której zawarta jest informacja na temat orbity, po której porusza się satelita oraz czasu jej nadania [1]. Odbiornik odbiera sygnał pochodzący z satelity z różnicą czasu potrzebną do propagacji fali radiowej na drodze satelita-odbiornik. Mając powyższe na uwadze czas potrzebny do propagacji fali radiowej można wyznaczyć ze wzoru numer 1 [1, 7]:

$$\Delta T = t_1 - t_2 \tag{1}$$

gdzie ΔT jest czasem propagacji sygnału radiowego na drodze satelita-odbiornik, t_1 jest czasem zawartym w depeszy nawigacyjnej, t_2 jest dokładnym czasem odbiornika. Ponieważ prędkość propagacji fali elektromagnetycznej w próżni wynosi c = 299 792 458 m/s odległość pomiędzy odbiornikiem GPS a satelitą można wyznaczyć ze wzoru 2 [1, 7]:

$$d = c\Delta T \tag{2}$$

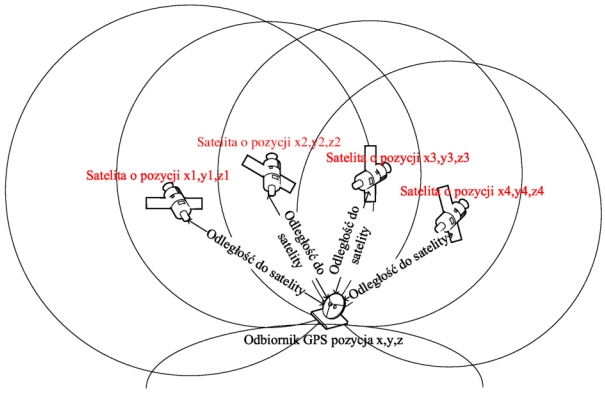
gdzie d jest odległością satelita-odbiornik, c jest prędkością fali elektromagnetycznej. Uwzględniając wzory numer 1 oraz 2 można wyznaczyć strefę o promieniu d, w której centrum obecny jest satelita, a na jej brzegach możliwe aktualne pozycję odbiornika GPS zgodnie z równaniem numer 3 [1, 7, 8]:

$$(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 = (c\Delta T)^2 \tag{3}$$

gdzie x_1, y_1, z_1 jest współrzędną geograficzną satelity, x, y, z jest współrzędną geograficzną odbiornika. Nieznane współrzędne geograficzne odbiornika określane są na podstawie odległości odbiornika do satelity, z którego pochodzi dana depesza nawigacyjna. Aby wyznaczyć pozycję odbiornika w postaci 3D konieczna jest widoczność przez odbiornik przynajmniej 3 satelitów. W praktyce jednak do wyznaczenia pozycji potrzebujemy 4 satelitów, z uwagi na wystąpienie różnic w zegarach atomowych [7]. Czwarty satelita umożliwia wyznaczenie błędu dokładności czasu. Rezultatem powyższego jest układ równań numer 4 [1, 8]:

$$\begin{aligned}
(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 &= c^2(t_1 - t)^2 \\
(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2 &= c^2(t_2 - t)^2 \\
(x_3 - x)^2 + (y_3 - y)^2 + (z_3 - z)^2 &= c^2(t_3 - t)^2 \\
(x_4 - x)^2 + (y_4 - y)^2 + (z_4 - z)^2 &= c^2(t_4 - t)^2
\end{aligned} \tag{4}$$

gdzie $x_{1..4}, y_{1..4}, z_{1..4}$ to współrzędne satelitów, x, y, z to współrzędna odbiornika $t_{1..4,}$ to czas propagacji fali radiowej, t to błąd zegara satelity. Rozwiązanie układu równań numer 4 umożliwia wyznaczenie pozycji odbiornika GPS. Na rysunku numer 3 przedstawiono ideę działania systemu GPS opisaną równaniem numer 4.



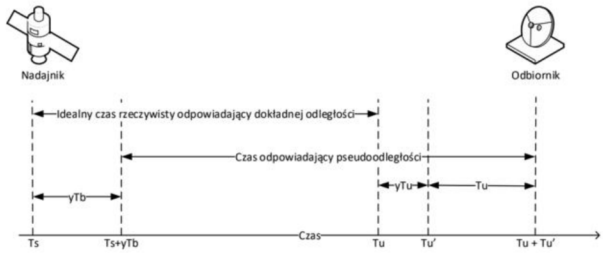
Rysunek 3: Idea działania systemu GPS [6].

3. Błędy systemu GPS

Wyznaczone wartości długości geograficznej, szerokości geograficznej, wysokości nad poziomem morza na podstawie równania numer 4 nie są dokładne i zawierają błąd wyznaczania pozycji. Główne błędy składające się na błąd określenia pozycji odbiornika to: błąd efemeryd, błąd położenie satelitów względem ziemi (DOP), błąd wynikający z niedokładności zegarów satelitarnych, opóźnienie sygnału, wielodrogowość, efekty relatywistyczne, błąd zegara odbiornika, błąd dokładności odbiornika [1]. Błąd obliczania pseudoodległości satelity w stosunku do odbiornika można wyznaczyć ze wzoru:

$$\beta_D = \beta_{at} + \beta_{sz} + \beta_{wd} + \beta_e \tag{5}$$

gdzie β_D jest błędem dokładności, β_{at} jest błędem wynikającym z opóźnienia sygnału w atmosferze, β_{sz} jest błędem szumów i braku dokładnej rozdzielczości odbiornika, β_{wd} jest błędem wielotorowości sygnału GPS, β_e jest błędem układów elektrycznych odbiornika. Sytuację powyżej opisaną przedstawiono na rysunku numer 4 [7].



Rysunek 4: Zależności czasowe odbiornik -nadajnik [7].

W oparciu o rysunek 4 można wyznaczyć odległość odbiornika do satelity zgodnie z równaniem numer 6 [7]:

$$\begin{aligned}
d = c[(Tu' + Tu) - (Ts + \gamma Tb)] = \\
C(Tu' - Ts) + c(Tu - \gamma Tb) = c(Tu + \gamma Tu - \\
Ts) + C(Tu - \gamma Tb) = D_{rz} + c(Tu - \gamma Tb + \\
\gamma Tu)
\end{aligned} \tag{6}$$

gdzie *d* to odległość odbiornika do satelity, *Ts* to czasem w którym sygnał został nadany, *Tu* to czas w którym sygnał powinien dotrzeć do odbiornika, *Tu'* to czas w którym sygnał dociera do odbiornika, *yTb* to błąd zegara satelity, *Tu* to błąd zegara odbiornika, *Drz* to

dokładna rzeczywista odległość do odbiornika. Przy założeniu, że *Drz* jest dokładną rzeczywistą odległością do odbiornika otrzymujemy równanie numer 7 [7].

$$drz = c(Tu - Ts) = c\Delta T \tag{7}$$

Składowa *(Tu – Ts)* reprezentuje błąd rzeczywistej odległości odbiornika od satelity.

4. Sposoby wyznaczania dokładności w systemie GPS

Podstawowym parametrem do pomiaru dokładności wykonanych pomiarów z wykorzystaniem jednej pozycji jest parametr RMS – Root Man Squere. RMS znany inaczej jako estymator odchylenia standardowego wyznaczany jest ze wzoru 8, 10. Należy uwzględnić jednak, że jest on podawany w stopniach szerokości oraz długości geograficznej [1, 9-10].

$$\sigma_\lambda = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{\lambda} - \lambda_k)^2} \tag{8}$$

$$\bar{\lambda} = \frac{1}{n}\sum_{i=1}^{n}\lambda_k \tag{9}$$

$$\sigma_\varphi = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{\varphi} - \varphi_k)^2} \tag{10}$$

$$\bar{\varphi} = \frac{1}{n}\sum_{i=1}^{n}\varphi_k \tag{11}$$

gdzie φ_k , λ_k to wartość pomiaru, $\bar{\varphi}$, $\bar{\lambda}$ to średnia wartość pomiarów, n jest liczbą pomiarów. Ponieważ RMS opisuje błąd wyznaczania pozycji w postaci jednowymiarowej w szerokości oraz długości geograficznej, należy przeliczyć ten błąd na metry wykorzystując równania 12, 13 [1, 9-10]:

$$\sigma_x = \sigma_\lambda \frac{2\pi r_e \cos(\varphi)}{360°} \tag{12}$$

$$\sigma_y = \sigma_\varphi \frac{2\pi r_e}{360°} \tag{13}$$

gdzie σ_x, σ_y to błąd pomiaru w metrach, σ_φ to błąd w szerokości geograficznej, σ_y to błąd w długości geograficznej, r_e to średni promień kuli ziemskiej w postaci idealnej. W praktyce wykorzystuje się błąd DRMS Distance Root Mean Square określający błąd w różnych kierunkach [1, 10]:

$$DRMS = \sqrt{(\sigma_x)^2 + (\sigma_y)^2} \tag{14}$$

gdzie σ_x – średnia niepewność kwadratowa określona na podstawie długości geograficznej, σ_y – średnia niepewność kwadratowa określona na podstawie szerokości geograficznej. W celu zwiększenia dokładności wyznaczanych pozycji zastosowanie ma parametr podwójnej odległościowej średniej niepewności kwadratowej, w języku angielskim Twice the Distance Root Mean Square – 2DRMS określa równanie 15 [9-10].

$$2DRMS = 2\sqrt{(\sigma_x)^2 + (\sigma_y)^2} \tag{15}$$

W zastosowaniu jest również parametr potrójnej odległościowej średniej niepewności kwadratowej [1, 9].

$$3DRMS = 3\sqrt{(\sigma_x)^2 + (\sigma_y)^2} \tag{16}$$

CEP (Circular Error Probability) – błąd kołowy jest miarą wyznaczania dokładności współrzędnych geograficznych określanych na podstawie długości promienia okręgu, w którym znajduje się 50% błędnych pomiarów. Punkt środkowy jest wyznaczany na podstawie współrzędnych rzeczywistych średnich wartości. Błąd kołowy odpowiada za wyznaczenie dokładności współrzędnych dwuwymiarowych. Parametr CEP wyznaczany jest zgodnie z równaniem 17 [1, 9-10].

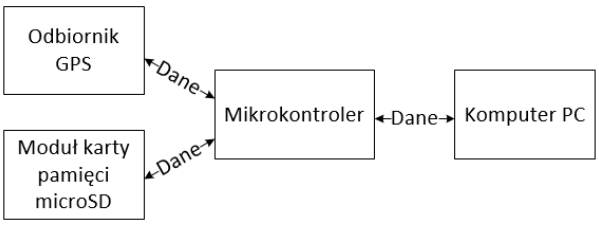
$$\text{CEP} = 0{,}62\sigma_x + 0{,}56\sigma_x \tag{17}$$

Parametr CEP jest prawidłowy dla wartości $\frac{\sigma_y}{\sigma_x} > 0{,}3$. Błąd R95 jest promieniem okręgu w którym znajduję się 95% pomiarów. Na podstawie równania 18 można wyznaczyć wartość R95 uwzględniając parametr CEP [1, 9-10].

$$\text{CEP}(95\%) = R95 = 2.08CEP = 2.08(0{,}62\sigma_x + 0{,}56\sigma_x) \tag{18}$$

5. Platforma badawcza

W celu przeprowadzenia pomiarów pozycji odbiornika GPS zbudowano platformę badawczą przedstawioną na rysunku 5.



Rysunek 5: Platforma badawcza.

Projekt oparto na mikrokontrolerze Arduino UNO oraz odbiorniku L76X GPS module. Dane pochodzące z satelitów w postaci komunikatów NMEA odbierane przez odbiornik, są przesyłane do mikrokontrolera z wykorzystaniem protokołu UART. W kolejnym etapie dane poddano zdekodowaniu w mikrokontrolerze Aruduino za pośrednictwem biblioteki TinyGPSPlus, która umożliwia zdekodowanie danych zawartych w pakietach NMEA [11]. Dane w ten sposób zdekodowane są zapisywane na karcie microSd oraz wysyłane do komputera PC za pośrednictwem portu USB. Wykorzystanie programu CoolTerm umożliwiło zapis danych przesyłanych do komputera w pliku .txt. Uzyskane w ten sposób dane poddano analizie z wykorzystaniem programu

MatLab. Dokładność odbiornika deklarowana przez producenta w karcie katalogowej produktu to 2,5m CEP [12].

6. Badania

Wykorzystując platformę pomiarową opisaną w rozdziale 5 przeprowadzono pomiary trzech przypadków testowych:

1. Badania współczynnika CEP odbiornika L76X w porównaniu do urządzenia wzorcowego przez okres 24 godzin w zmiennych warunkach atmosferycznych.
2. Badanie współczynnika CEP odbiornika L76X w stałym punkcie odniesienia przez okres jednej godziny.
3. Porównanie danych uzyskanych z wykorzystaniem odbiornika L76X do przebytej trasy.

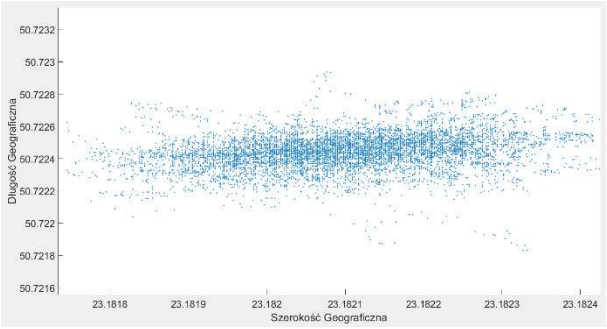
W trakcie przeprowadzania badań dla przypadków testowych określonych w punktach 1, 2, 3 wystąpiły warunki pogodowe przedstawione w tabeli 1.

Tabela 1: Warunki pogodowe

Przypadek 1				
Godzina	**Temp [°C]**	**Zachmurzenie**	**Ciśnienie**	**Opady**
0:00-6:00	2-4	Całkowite	1015 hPa	Tak
6:00-12:00	5-10	Całkowite	1013 hPa	Tak
12:00-18:00	10-13	Częściowe	1012 hPa	Przelotne
18:00-0:00	10-5	Brak	1011hPa	brak
Przypadek 2				
15:00-16:00	18	Brak	1021 hPa	Brak
Przypadek 3				
12:00-13:00	13	Brak	1011 hPa	Brak

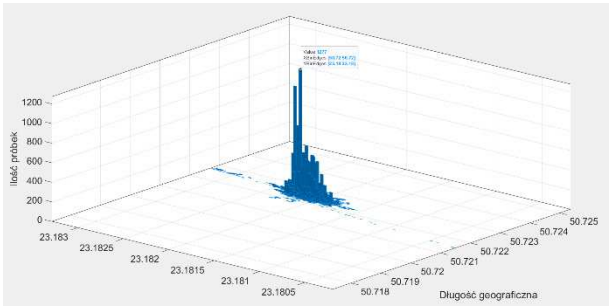
7. Wyniki

Wyznaczenie dokładnej pozycji odbiornika L76X było kluczowym problemem napotkanym podczas przeprowadzania badań. W celu otrzymania wiarygodnych wyników przeprowadzono 24 godzinne badanie z użyciem urządzenia wzorcowego. Na rysunku 6 przedstawiono wyniki z przeprowadzenia 24 godzinnych badań z wykorzystaniem odbiornika L76X, uzyskano 30 tysięcy próbek geolokalizacyjnych pozycji odbiornika L76X.



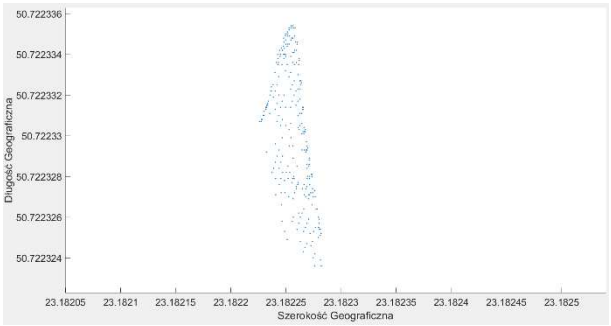
Rysunek 6: Zarejestrowane pozycje odbiornika L76X.

Na podstawie równania 7 oraz 9 wyznaczono punkt centralny zbioru danych jako 50.7224633° długości geograficznej oraz 23.1821494° szerokości geograficznej. W celu wyznaczenia rozkładu empirycznego pozycji utworzono histogram, przedstawiony na rysunku 7 wykres wyznacza najczęściej lokalizowaną pozycję odbiornika GPS. Mając powyższe na uwadze możemy wyznaczyć najczęściej określaną pozycję jako 50.7224651° długości geograficzne oraz 23.1821412° szerokości geograficznej.



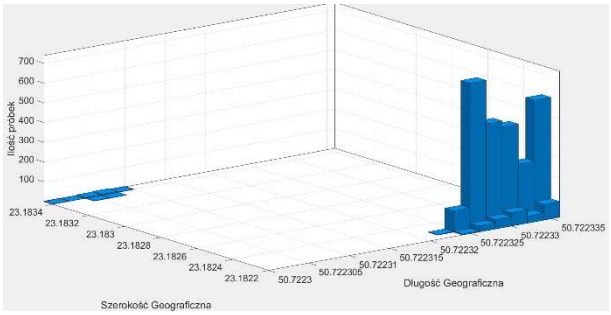
Rysunek 7: Histogram odbiornika L76X.

Na podstawie wzoru 17 wyznaczono parametr CEP = 1.32 m. W czasie badania odbiornika L76X przeprowadzono równoczesne pomiary z wykorzystaniem urządzenia wzorcowego, w ich trakcie uzyskano 10 tysięcy próbek określających pozycję wzorcowego odbiornika GPS. Punkty przedstawione na rysunku 8 są wynikami 24 godzinnych pomiarów z wykorzystaniem odbiornika wzorcowego.



Rysunek 8: Zarejestrowane pozycje odbiornika wzorcowego.

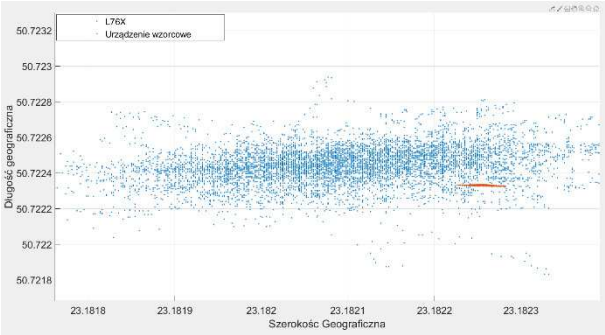
Na rysunku 9 przedstawiono rozkład empiryczny pozycji odbiornika wzorcowego.



Rysunek 9: Histogram urządzenia wzorcowego.

Na podstawie rysunku 9 można określić najczęściej wyznaczaną pozycję określoną jako 50.7223262° długości geograficznej oraz 23.182275° szerokości geograficznej. Obliczając średnią arytmetyczną na podstawie równania 7 oraz 9 z uzyskanych danych pomiarowych wyznaczono punkt centralny o współrzędnych 50.7223303° długości geograficznej oraz 23.1822706° szerokości geograficznej. Wynik parametru CEP otrzymanego z wyliczeń na podstawie wzoru 17 wyniósł

CEP=5.19 m. Porównanie otrzymanych pozycji urządzenia wzorcowego oraz odbiornika L76X przedstawiono na rysunku 10.



Rysunek 10: Porównanie pozycji odbiorników.

Uwzględniając najczęściej występujące pomiary uzyskane z wykorzystaniem histogramów oraz średnich wartości pomiarów można wyznaczyć przybliżoną odległość urządzenia wzorcowego w stosunku do odbiornika L76X. W tabeli 2 przedstawiono porównanie wyznaczonych pozycji.

Tabela 2: Najczęstsze pozycję odbiorników

Badany odbiornik	**Długość Geograficzna średnia**	**Szerokość Geograficzna średnia**
Odbiornik L76X	50.7224633°	23.1821494°
Urządzenie wzorcowe	50.7223303°	23.1822706°
Badany odbiornik	**Długość Geograficzna histogram**	**Szerokość Geograficzna histogram**
Odbiornik L76X	50.72246851°	23.1821412°
Urządzenie wzorcowe	50.72232622°	23.1822752°

Na podstawie tabeli 2 można wyznaczyć różnicę odległości pomiędzy najczęściej występującymi pomiarami.

Tabela 3: Różnica pomiędzy pomiarami

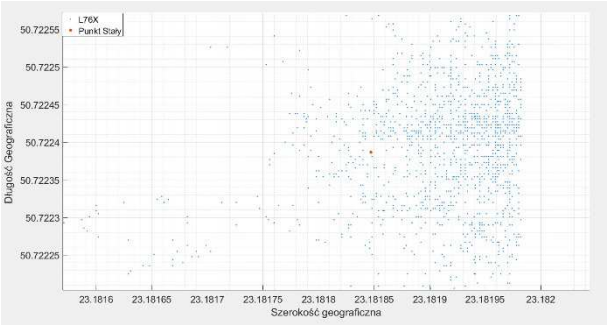
Różnica pomiędzy	**Różnica w długości geograficznej [m]**	**Różnica w szerokości Geograficzna średnia[m]**
Wartość średnia pomiarów	14,80	12,30
Najczęściej występującymi pomiarami	15,82	13,32

Analizując tabelę 3 można dojść do wniosku że uśrednianie rezultatów badań wyznaczania pozycji nawigacyjnych daje większą dokładność w stosunku do najczęściej występujących pomiarów. W tabeli 4 przedstawiono porównanie współczynników CEP dla 24 godzinnego okresu badawczego.

Tabela 4: Porównanie parametrów CEP

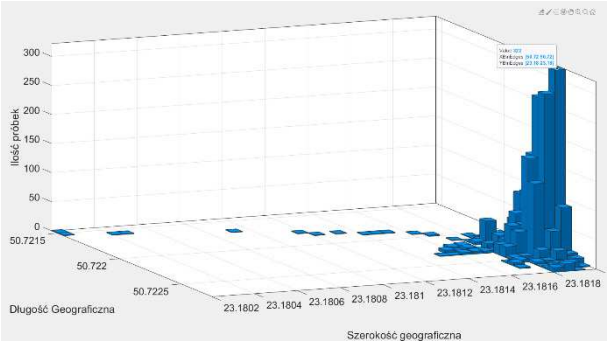
Badany odbiornik	**Wyznaczona wartość parametru CEP**
L76x GPS Module	1.32
Urządzenie wzorcowe	5.18

W drugim przypadku testowym porównano pozycję odbiornika L76X do pozycji stałej o niezmiennych współrzędnych geograficznych. W celu otrzymania wiarygodnych wyników przeprowadzono godzinne badania geolokalizacyjne. Przez ten okres uzyskano 2 tysiące próbek. Na rysunku 11 czerwoną kropką oznaczono stały niezmienny punkt o współrzędnych geograficznych 50.722386° długości geograficznej oraz 23.1818470° szerokości geograficznej, na niebiesko widoczne są współrzędne geograficzne uzyskane w wyniku przeprowadzonych pomiarów.



Rysunek 11: Pozycje odbiornika w zależności do punktu stałego.

Określając centralny punkt na podstawie równania 9 oraz 11 otrzymano wynik 50.722386° długości geograficznej oraz 23.1818470° szerokości geograficznej. Rozkład empiryczny w postaci histogramu przedstawia rysunek 12.



Rysunek 12: Częstotliwość pomiarów współrzędnych geograficznych.

Rezultatem obliczeń parametru CEP na podstawie wzoru 17 jest wynik 1.13 m. W tabeli 5 przedstawiono porównanie uzyskanych współrzędnych geograficznych.

Tabela 5: Zależności pomiędzy punktami

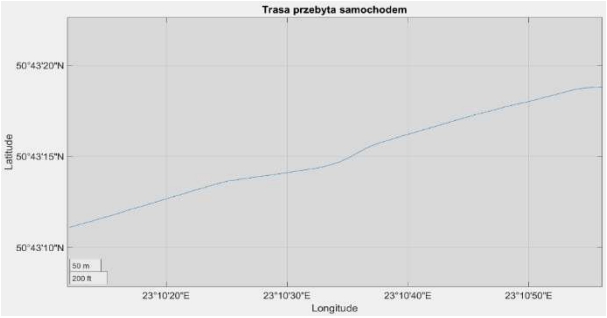
Punkt	**Szerokość Geograficzna**	**Długość Geograficzna**
Punkt średnia	23.181890°	50.722330°
Punkt histogram	23.181920°	50.722330°
Punkt stały	23.181847°	50.722386°

Na podstawie tabeli 5 możemy wyznaczyć odległości punktu stałego w stosunku do punktu średniego oraz punktu najczęściej występującego.

Tabela 6: Różnica odległości

Odległość pomiędzy	**Różnica w długości geograficznej [m]**	**Różnica w szerokości Geograficzna[m]**
Punkt stały – Punkt Średni	4,41	5,74
Punkt stały – Punkt histogram	7,48	5,74

W trzecim przypadku testowym platformę pomiarową umieszczono w samochodzie osobowym i pokonano trasę przedstawioną na rysunku 13.



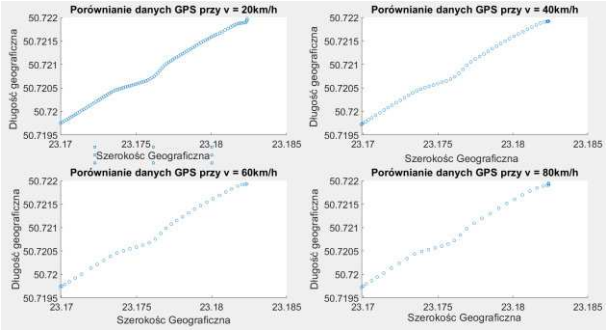
Rysunek 13: Przebyta trasa.

W celu zbadania dokładności pomiarów w zależność od prędkości samochodu przeprowadzono cztery próby badawcze z prędkościami zgodnymi z tabelą 7.

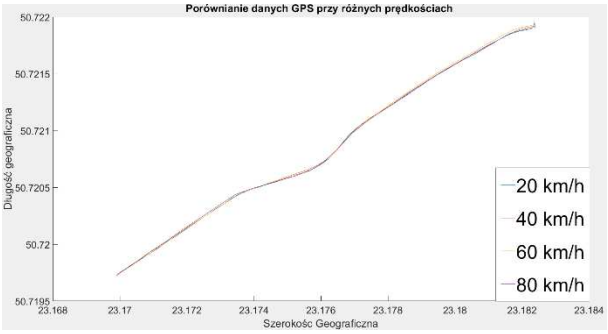
Tabela 7: Zależność prędkości od liczby próbek

Prędkość	Czas Przebycia Drogi	Liczba Uzyskanych Próbek
v1 = 20 km/h	3:20	98
v2 = 40 km/h	1:56	56
v3 = 60 km/h	1:12	39
v4 = 80 km/h	0:50	28

Wyniki pomiarów przedstawiono na rysunku 14.



Rysunek 14: Wyniki pomiarów.



Rysunek 15: Porównanie przebytych tras.

Konfrontując ze sobą wyniki pomiarów przeprowadzonych przy prędkościach 20km/h, 40 km/h, 60 km/h, 80km/h dokonano graficznej prezentacji widocznej na rysunku 15 przebytych tras zgodnych z rysunkiem 13. Poddając analizie rysunek 14,15 możemy stwierdzić, że dokładność wyznaczania punktów geolokalizacyjnych zależy od ilości próbek pobranych z przemieszczającego się obiektu. W przypadku pomiaru z prędkością 80 km/h widoczny jest znaczący spadek dokładności odwzorowania przebytej trasy.

8. Wnioski

Na podstawie przeprowadzonych badań można sformułować następujące wnioski:

1. Parametr CEP odbiornika L76X GPS Module podczas pierwszego przypadku testowego wyniósł 1,32 m. Porównując do wartości parametru CEP wynoszącego 2,5 m podawanego przez producenta modułu można stwierdzić że moduł spełnia kryterium określone przez producenta,
2. Parametr CEP urządzenia wzorcowego podczas badań wyniósł 5,19, mając powyższe na uwadze można stwierdzić że odbiornik L76X jest 3,5 razy dokładniejszy,
3. Różnica w długości geograficznej według średnich wartości pomiaru pomiędzy urządzeniem wzorcowym a modułem L76X wynosiła 14,80 m natomiast w szerokości geograficznej 12,30 m,
4. Różnica w długości geograficznej według najczęściej występujących wartości pomiaru pomiędzy urządzeniem wzorcowym a modułem L76X wynosiła 15,82 m natomiast w szerokości geograficznej 13,32 m,
5. Parametr CEP odbiornika L76X GPS Module podczas drugiego przypadku testowego wyniósł 1,13 m. Porównując do wartości parametru 2,5 m podawanego przez producenta modułu możemy stwierdzić że moduł spełnia kryterium określone przez producenta,
6. Różnica w długości geograficznej według średnich wartości pomiaru pomiędzy miejscem wzorcowym (drugi przypadek badawczy) a modułem L76X wynosiła 4,40 m natomiast w szerokości geograficznej 5,73 m,
7. Różnica w długości geograficznej według najczęściej występujących wartości pomiaru pomiędzy urządzeniem wzorcowym (drugi przypadek badawczy) a modułem L76X wynosiła 7,48 m natomiast w szerokości geograficznej 5,73 m,
8. Porównując wyniki badań przeprowadzonych podczas ruchu z wykorzystaniem samochodu osobowego (trzeci przypadek badawczy), możemy stwierdzić że w przypadku większej ilości próbek otrzymujemy dokładniejszy pomiar geolokalizacyjny.

Literatura

[1] E. Pawłowski, Experimental study of a positioning accuracy with GPS receiver, the 12th Conference on Selected Problems of Electrical Engineering and Electronics, WZEZ 2015.

[2] Dr. Bernhard Hofmann-Wellenhof, GNSS Global Navigation Satellite Systems GPS, GLONASS, Galileo & more., SpringerWien NewYork 9783211730126.

[3] D. Doberstein, Fundamentals of GPS Receivers. A Hardware Approach, Springer Science+Business Media, 2012.

[4] E. Dziadczyk, Satellite Navigation System GPS, Conference: CAD Systems in Microelectronics, 2007.

[5] J. Lamparski, Navstar GPS od teorii do praktyki, Wydawnictwo uniwersytetu Warmińsko-Mazurskiego, Olsztyn 2001.

[6] NMEA Reference Manual, rev. 1, SiRF Technology Inc. January 2005.

[7] E. D. Kaplan, Understanding GPS Principles and Applications Second edition, Artech House, London.

[8] J. Narkiewicz, Budowa działanie, zastosowanie", WKŁ, Warszawa 2003.

[9] E. Pawłowski, An experimental investigation into the positioning accuracy of low-cost gps receivers in labview environment, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, Gdańsk 2015.

[10] Novatel, GPS Position Accuracy Measures, December, 2003.

[11] S. Ashrtaf, A low-cost solution for unmanned aerial vehicle navigation in a global positioning system–denied environment, International Journal of Distributed Sensor Networks, June 18, 2018.

[12] Dokumentacja techniczna odbiornika L76X GPS Module.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 339–344
Received: 29 July 2020
Accepted: 24 September 2020

Analysis and comparison of programming frameworks used for automated tests

Analiza i porównanie szkieletów programistycznych używanych do testów automatycznych

Damian Gromek*, Dariusz Gutek

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

Abstract
This article compares and discusses the programming skeletons for automatic tests. Two test skeletons have been selected for testing purposes and then a test environment has been installed and configured, in which appropriate test scenarios have been prepared. Once the test environment has been properly prepared, measurements of the time to launch both frameworks were performed. The results obtained were shown in the form of tables and charts for later analysis. The frameworks were also analyzed in terms of syntax and functionality. At the end, a summary was presented, containing more information and conclusions that resulted from the analysis.

Keywords: Testing; JUnit; TestNG; Cucumber

Streszczenie
W niniejszym artykule zostały porównane i omówione szkielety programistyczne służące do testów automatycznych. Do celów badawczych zostały wybrane dwa szkielety a następnie zostało zainstalowane i skonfigurowane środowisko badawcze, w którym przygotowano odpowiednie scenariusze testowe. Po odpowienim przygotwaniu środowiska badawczego zostały przeprowadzone pomiary czasu uruchomienia obydwu frameworków. Wyniki jakie otrzymano pokazano w postaci tabel i wykresów umożliwiających ich późniejszą analizę. Badane frameworki zostały również przeanlizowane pod kątem składni i funkcjonalności. Na koniec zostało przedstawione podsumowanie zawierające najistotniejsze informację i wnioski jakie wynikły z przeprowadzonej analizy.

Słowa kluczowe: Testowanie; JUnit; TestNG; Cucumber

*Corresponding author
Email address: **damian.gromek@pollub.edu.pl** (D. Gromek)

1. Wstęp

W dzisiejszych czasach zaobserwować można bardzo dynamiczny rozwój nowych technologii a także bardzo szybkie tempo życia co implikuję potrzebę kreowania bardziej wydajnych i bardziej niezawodnych systemów informatycznych mogących sprostać coraz to nowszym wymaganiom. Klienci oczekują niezawodności tworzonego dla nich oprogramowania, gdyż często w przypadku awarii systemów informatycznych może to ich doprowadzić do ogromnych strat finansowych wynikających z zaprzestania prowadzenia przez nich działalności a nawet do strat wizerunkowych wynikających z błędnie działającego oprogramowania.

Wzrost złożoności tworzonych systemów informatycznych a także zmiany w samych metodach zarządzania projektem zwiększają ryzyko wystąpienia potencjalnych błędów. Aby nabrać zaufania do tworzonego oprogramowania niezbędny jest proces testowy, którego celem jest zapewnienie jakości rozwijanemu oprogramowaniu.

Coraz to nowsze wymagania stawiane deweloperom wymuszają pośrednio wymyślanie i tworzenie bardziej skutecznych technik i narzędzi służących do testowania oprogramowania. Jednym z działań usprawniających proces testowy jest jego automatyzacja. Mnogość narzędzi i frameworków służących do automatyzacji testowania często sprawia istotny problem z doborem odpowiednich narzędzi na początku prac nad zautomatyzowaniem testów w projektach komercyjnych. Dokonanie błędnego lub nieoptymalnego wyboru narzędzi często powoduję późniejszą frustracje wynikającą z pracy z mało wydajnymi narzędziami jak i straty finansowe na poczet nowych licencji w przypadku wymiany narzędzi na inne w późniejszej fazie projektu.

W niniejszym artykule zostały omówione i porównane ze sobą dwa szkielety programistyczne służące do automatyzacji testów jednostkowych: JUnit oraz TestNG. Analiza porównawcza tych dwóch frameworków jest niezwykle istotna z racji tego, że oba te szkielety programistyczne ukierunkowane są na wspieranie testów jednostkowych dla aplikacji pisanych w języku Java. Język Java jest bowiem bardzo popularnym i wszechstronnym językiem programowania co przekłada się na to, że obecnie wiele firm pracuję w tej technologii szukając przy tym wydajnych i funkcjonalnych narzędzi wspierających proces wytwarzania oprogramowania, którego istotną częścią jest faza testów. Analiza tych dwóch szkieletów programistycznych miała na celu wskazanie bardziej wydajnej technologii i została oparta na porównaniu ich szybkości a także zakresu funkcjonalności.

2. Wprowadzenie do testowania oprogramowania

Istnieje wiele różnych podejść i metod testowania oprogramowania. W tym rozdziale zostało opisanych kilka z nich [1].

- Testy modułowe polegają na testowaniu najmniejszych komponentów możliwych do wyizolowania – mogą to być moduły, klasy, metody. Testy modułowe możemy podzielić na testy jednostkowe, testy pokrycia instrukcji, testy klas równoważności bądź testy wartości brzegowych.
- Testy integracyjne przeprowadzane są po testach modułowych i polegają na sprawdzeniu interakcji pomiędzy poszczególnymi modułami.
- Testy systemowe mają za zadanie przetestować zachowanie oprogramowania w ujęciu całościowym – testy na tym poziomie wykonywane są w momencie, gdy poszczególne komponenty i funkcje są ze sobą zintegrowane i tworzą spójny system.
- Testy akceptacyjne są najczęściej wykonywane przez użytkowników a celem tych testów nie jest już wykrywanie błędów a jedynie formalne potwierdzenie jakości i spójności gotowego już oprogramowania.
- Testy alfa mogą być rzeczywiste bądź symulowane i są zwykle przeprowadzane przez potencjalnych przyszłych użytkowników danego oprogramowania bądź przez niezależny zespół testerów. Przeprowadzane są najczęściej na miejscu w siedzibie producenta.
- Testy beta przeprowadzane są już na rzeczywistym środowisku produkcyjnym przez potencjalnych bądź też realnych użytkowników danego oprogramowania. Testy te nie odbywają się już w siedzibie producenta a poza nim. Celem tych testów jest ocena przez użytkowników czy system spełnia ich potrzeby i wymagania.
- Testy regresji są często mylone z retestami. Jest to duży błąd, gdyż te terminy nie są ze sobą jednoznaczne. Retest jest to sprawdzenie czy pojedynczy defekt został naprawiony przez dewelopera i czy już więcej nie występuje, natomiast testy regresji polegają na sprawdzeniu większego obszaru aplikacji w celu weryfikacji czy np. wprowadzone zmiany w implementacji wynikające z dodania nowej funkcjonalności bądź przy okazji poprawy defektu nie spowodowały błędów w innych częściach oprogramowania nawet pozornie nie związanego z wprowadzonymi zmianami.
- Testy pielęgnacyjne dokonywane są już na wdrożonym i użytkowanym systemie informatycznym. Testy te zwykle dokonywane są po wdrożeniu nowej wersji oprogramowania.

3. Automatyzacja testowania

Proces automatyzacji testów wnosi wiele wartości w obszarze zapewnienia jakości w projektach komercyjnych. Automatyzacja testów pozwala zaoszczędzić wiele czasu, który byłby niezbędny na wykonanie tych testów w sposób manualny. Mimo wielu zalet automatyzacji posiada on też pewne ograniczenia przez, które zwykle nie jest możliwe zautomatyzowanie wszystkich testów w projekcie. Niniejszy rozdział ma na celu omówić najważniejsze plusy automatyzacji testów jak i jej ograniczenia [2].

3.1. Korzyści ze stosowania zautomatyzowanych testów

Głównym celem automatyzacji jest zautomatyzowanie najczęściej powtarzanych procesów wykonywanych do tej pory w sposób manualny. W wyniku tego procesu następuje szereg korzyści, które zostały omówione poniżej [3].

- Wzrost produktywności. Dobrze zaimplementowany zestaw testów automatycznych powinien wykonywać się bez jakichkolwiek ingerencji ze strony testera – nie powinien on w trakcie działania wymagać od testera podejmowania żadnych dodatkowych działań ani pytać go np. o napotkany wyjątek. Obsługa takich przypadków powinna być z góry zaimplementowana w kodzie testów automatycznych. Dobrze napisany i skonfigurowany test automatyczny powinien przechodzić przez wszystkie kroki bez zatrzymywania się, nawet gdy po drodze napotka defekt – chyba, że defekt ten uniemożliwia dalsze wykonywanie testu. W związku z tym test automatyczny może być wykonywany poza godzinami pracy testera. W ciągu dnia tester automatyzujący może pracować nad kodem testu automatycznego a tuż przed wyjściem z pracy uruchomić go – wtedy następnego dnia wygenerowany raport z przeprowadzonych testów może zostać przekazany do zespołu deweloperskiego odpowiedzialnego za naprawę znalezionych defektów. Jako, że gotowe testy automatyczne mogą być przeprowadzane po opuszczeniu przez testera stanowiska pracy oznacza to zaoszczędzenie czasu potrzebnego na manualne wykonanie tychże testów na stanowisku pracy w firmie.
- Wzrost niezawodności. Dobry test powinien być łatwy do powtórzenia zarówno przez testera jak i przez dewelopera odpowiedzialnego za naprawę potencjalnego defektu znalezionego w czasie takiego testu. Podczas wykonywania testu manualnego możliwe jest pomimo wykonywania jednego i tego samego scenariusza przeprowadzenie go nieco inaczej np. używając za każdym razem innych danych testowych, bądź w przypadku manualnych testów frontendu możliwe jest kliknięcie nieco innego obszaru aplikacji co może skutkować zupełnie innym zachowaniem testowanej aplikacji. Test automatyczny z kolei uruchamiany jest zawsze w ten sam, przewidywalny sposób, który został zaprogramowany przez testera automatyzującego co oznacza, że w przypadku testów frontendu, obszar klikany na frontendzie aplikacji jest zawsze ten sam. Dzięki temu ewentualne rozbieżności przy ponownych uruchomieniach testów automatycznych są wykluczane. Wykonując po raz n-ty ten sam scenariusz testowy, można przestać zwracać uwagę na szczegóły i nie

zauważać np. literówek na stronie internetowej albo nawet i poważniejszych usterek. W przypadku testów automatycznych za wykonanie tych testów jest odpowiedzialny automat, którego zadaniem jest przetestowanie aplikacji dokładnie w taki sposób w jaki został on zaprogramowany przez testera. Takie zachowanie gwarantuje brak przypadkowego pominięcia sprawdzenia jakiejś funkcjonalności bądź jakiegoś obszaru aplikacji. Każdy niewykonany test np. poprzez złą implementacje jest wyraźnie komunikowany poprzez wyrzucenie kodu błędu wraz z informacją co poszło nie tak, zatem nie istnieje możliwość przypadkowego pominięcia testów w przypadku testowania automatycznego.

- Wzrost pokrycia testami. Testy automatyczne wykonywane są zdecydowanie szybciej niż testy manualne a dodatkowo dzięki nim możliwe jest symulowanie obciążeń bądź nawet przeciążeń danej aplikacji poprzez np. uruchomienie testu na kilku przeglądarkach naraz. Test manualny przeprowadzony może być tylko jeden w jednym czasie. Nie ma możliwości manualnego kliknięcia na dwa elementu naraz w dodatku na dwóch różnych przeglądarkach. Testowanie manualne nie jest zatem w stanie objąć testami pewnych specyficznych przypadków bądź wymagań, które z kolei mogą zostać bez problemu przetestowane dzięki użyciu testów automatycznych.

3.2. Ograniczenia testowania automatycznego

Opracowanie testów automatycznych niesie ze sobą liczne korzyści zarówno dla zespołu testerów – ograniczanie wykonywania monotonnych czynności wiele razy, jak i dla całego projektu – szybsze testowanie to szybsze informację o jakości oprogramowania. Zwykle jednak testy automatyczne nie mogą w pełni zastąpić testów manualnych w projekcie. Wynika to głównie z faktu, że część potencjalnych problemów może zostać znaleziona jedynie poprzez wykonanie manualnych testów przez człowieka.

Często spotykanym zabezpieczeniem na stronach internetowych jest tzw. CAPTCHA. Zabezpieczenie to ma na celu dopuszczenie przesyłania danych bądź formularzy tylko przez człowieka. W momencie, gdy test automatyczny natrafi na to zabezpieczenie to prawdopodobnie wynik tego testu będzie negatywny, co wcale nie musi być prawdą, gdyż zabezpieczenie typu CAPTCHA jest często używanym i świadomym zabezpieczeniem więc samo jej występowanie nie świadczy o błędzie w implementacji. W tym przypadku zatem, wynik z przeprowadzonego testu może wprowadzać testera w błąd i generować nieprawdziwy raport o poziomie jakości danego oprogramowania.

Żeby testy automatyczne zaczęły wnosić wartość w projekcie niezbędne jest spędzenie znacznej ilości czasu nad ich zaplanowaniem i przygotowaniem. W związku z tym w bardzo małych i krótkich projektach testy automatyczne mogą nie być najlepszym pomysłem ze względu na ograniczone zasoby czasowe. Testy manualne na poziomie frontendu aplikacji zwykle w takich przypadkach sprawdzają się lepiej, gdyż mogą one być wykonywane w każdym momencie po uprzednim przygotowaniu scenariuszy testowych.

Opis porównywanych szkieletów programistycznych

W niniejszym rozdziale zostały zaprezentowane i omówione badane szkielety programistyczne. Zostały również zestawione ze sobą poszczególne adnotacje jakie są używane w każdym z omawianych frameworków służące do wywoływania określonych funkcjonalności.

TestNG

Framework TestNG został opracowany przez Cedrica Beusta w celu automatyzacji testów dla języka Java. Szkielet ten został stworzony w czasie, gdy na rynku dostępny był framework JUnit w wersji 3 po to aby wyeliminować jego liczne wówczas ograniczenia. Opracowanie frameworka TestNG nie tylko wyeliminowało większość ograniczeń ówczesnego frameworka JUnit ale także dodało wiele nowych funkcjonalności obejmujących nie tylko testy jednostkowe ale także testy funkcjonalne a nawet testy end-to-end. Została wprowadzona obsługa adnotacji, która była niedostępna we frameworku JUnit w wersji 3 [4].

3.3. JUnit

Kent Beck, Erich Gamma, David Saff oraz Kris Vasudevan są autorami frameworka JUnit, którego celem jest tworzenie testów jednostkowych dla oprogramowania napisanego w języku Java. Najnowsza wersja frameworka – JUnit 5 – wprowadziła szereg zmian oraz aktualizacji wcześniej wykorzystywanych funkcjonalności. Zmiany te dotyczyły m.in. zastąpienia dotychczas używanych nazw poszczególnych adnotacji na bardziej samo opisujące się a przez to bardziej zrozumiałe dla użytkowników [5, 6]. Dokonano zmian w nazwach adnotacji odpowiedzialnych za uruchamianie konkretnego testu bądź grup testów. Dodatkowo w JUnit w wersji 5 zostały wprowadzone dodatkowe asercje odpowiadające za testowanie i sprawdzanie czy czas uruchomienia określonego fragmentu kodu został wykonany w czasie szybszym niż z góry zadeklarowany limit [7].

3.4. Porównanie składni frameworka TestNG i JUnit

Pomimo tego, że zarówno framework TestNG jak i framework JUnit służą do przeprowadzania testów jednostkowych w języku Java oraz że posiadają wiele cech wspólnych istnieją pomiędzy nimi pewne różnice zarówno w składni jak i w zakresie funkcjonalności. W Tabeli 1 zamieszczonej poniżej przedstawione zostało porównanie adnotacji używanych w każdym z omawianych frameworków [8].

Na podstawie zestawień przedstawionych w Tabeli 1 wynika, że framework TestNG posiada więcej funkcji możliwych do zastosowania podczas implementowania testów automatycznych.

Tabela 1: Porównanie składni i funkcji frameworka TestNG i JUnit

Opis adnotacji	Adnotacja	
	TestNG	JUnit 5
Uruchomienie testu	@Test	@Test
Uruchomienie kodu przed testem	@BeforeClass	@BeforeAll
Uruchomienie kodu po wszystkich metodach	@AfterClass	@AfterAll
Uruchomienie kodu przed każdą metodą testującą	@BeforeMethod	@BeforeEach
Uruchomienie kodu po każdej metodzie testującej	@AfterMethod	@AfterEach
Ignorowanie określonego testu	@Test (enabled=false)	@ignore
Określenie spodziewanego wyjątku	@Test(expectedException = Arithmetic Exception.class)	@Test(expected = Arithmetic Exception.class)
Określenie maksymalnego czasu na wykonanie testu	@Test (timeOut = 1000)	@Test (timeout = 1000)
Uruchomienie kodu przed wszystkimi testami w scenariuszu	@BeforeSuite	Brak
Uruchomienie kodu po wszystkich testach w scenariuszu	@AfterSuite	Brak
Uruchomienie kodu przed konkretnym testem	@BeforeTest	Brak
Uruchomienie kodu po konkretnym teście	@AfterTest	Brak
Uruchomienie kodu przed pierwszym testem w grupie	@BeforeGroups	Brak
Uruchomienie kodu po ostatnim teście w grupie	@AfterGroups	Brak

4. Przeprowadzenie badań pomiarowych

Przeprowadzenie badania miało na celu ustalić, który z porównywanych frameworków potrzebuje mniej czasu na wykonanie testów dla wszystkich przygotowanych scenariuszy testowych.

4.1. Opis badania

Badanie zostało przeprowadzone na laptopie z procesorem Intel Core i7-9750H z pamięcią RAM 32GB przy użyciu narzędzia IntelliJ IDEA w środowisku Windows. Użyta została przeglądarka Google Chrome w wersji 83. Posłużono się narzędziem Cucumber w celu przygotowania i uruchamiania scenariuszy testowych. Każdy scenariusz testowy został wykonany trzykrotnie dla obu badanych szkieletów programistycznych. Po dokonaniu pomiarów wyniki zostały ze sobą zestawione i przeanalizowane.

4.2. Przebieg badania

Na początku badań przygotowano dwa odrębne projekty – jeden dla testów z użyciem frameworka TestNG a drugi dla testów z użyciem frameworka JUnit. W następnym kroku przy pomocy narzędzia Cucumber zostało opracowanych siedem następujących scenariuszy testowych:

- TC 1 – Scenariusz sprawdzający logowanie się do serwisu przy użyciu poprawnego loginu i hasła.
- TC 2 – Scenariusz sprawdzający logowanie do serwisu przy użyciu poprawnego loginu i niepoprawnego hasła.
- TC 3 – Scenariusz sprawdzający wyszukiwanie produktów na stronie.
- TC 4 – Scenariusz sprawdzający formularz rejestracyjny.
- TC 5 – Scenariusz sprawdzający funkcjonalność dodawania produktu do koszyka.
- TC 6 – Scenariusz sprawdzający kolor czcionki napisu na stronie głównej serwisu.
- TC 7 – Scenariusz, który za każdym razem ma dać negatywny wynik.

Jeden z przykładowych scenariuszy testowych wykonany w narzędziu Cucumber został zamieszczony na Listingu 1 zamieszczonym poniżej.

Listing 1: Przykładowy scenariusz testowy opracowany w narzędziu Cucumber

```
@grupaScenariuszy
Feature: TestNG + Cucumber

  @test1_poprawneLogowanie
  Scenario: Użytkownik wprowadza poprawne dane logowania
    Given Użytkownik uruchamia strone serwisu
    Then Strona główna serwisu Media Expert jest wyświetlona
    And Użytkownik klika na przycisk Twoje konto
    And Użytkownik czeka na załadowanie się strony
    And użytkownik wprowadza w pole login swój login
    And użytkownik wprowadza w pole hasło swoje hasło
    And użytkownik klika na przycisk zaloguj
    And Strona powitalna jest wyświetlona
    And Test nr jeden zakonczony
```

Skrypt zamieszczony na Listingu 1 przedstawia wysokopoziomową implementację testu sprawdzającego funkcjonalność logowania się do serwisu internetowego. Po opracowaniu scenariuszy testowych w narzędziu Cucumber dla każdego kroku zostały zaimplementowane odpowiednie metody testujące. Do badań opracowano takie same scenariusze testowe dla każdego z analizowanych frameworków.

Po każdym przeprowadzonym pomiarze generowany był raport z wykonanych testów. Przykład raportu otrzymanego dla przypadku testowego sprawdzającego logowanie się do serwisu internetowego został ukazany na Rysunku 1 zamieszczonym poniżej.

Rysunek 1: Raport po wykonaniu testu

Raport z przeprowadzonego testu ukazuje dwie istotne informację. Pierwszą z nich jest status każdego poszczególnego kroku informujący o tym czy krok ten zakończył się pomyślnie czy nie. Drugą istotną informacją jest czas wykonania każdego kroku jak i łączny czas wykonania testów dla uruchomionego przypadku testowego. Czasy jakie zostały zawarte w raportach z przeprowadzonych testów wykorzystane zostały w późniejszej analizie porównawczej.

4.3. Wyniki przeprowadzonego badania

Po pomyślnym uruchomieniu i zakończeniu wszystkich testów, za każdym razem wygenerowany został raport z wynikami testów poszczególnych scenariuszy i kroków w nim zawartych a także czasy ich wykonania.
Na potrzeby badań zostały zebrane wszystkie zmierzone cząstkowe czasy a ich wyniki zostały przedstawione w postaci tabel.
Tabela 2 zamieszczona poniżej przedstawia zestawienie czasów dla frameworka TestNG.

Tabela 2: Pomiary dla frameworka TestNG

Przypadek testowy	Pomiar 1 (s)	Pomiar 2 (s)	Pomiar 3 (s)
TC 1	18,91	22,53	22,21
TC 2	21,45	21,71	21,89
TC 3	27,66	27,94	29,47
TC 4	18,74	17,21	17,73
TC 5	50,41	58,82	50,25
TC 6	9,96	12,45	13,38
TC 7	11,83	10,49	10,85
Suma	158,96	171,15	165,78

Z kolei Tabela 3 zamieszczona poniżej przedstawia zestawienie czasów wykonania testów dla frameworka JUnit.

Tabela 3: Pomiary dla frameworka JUnit

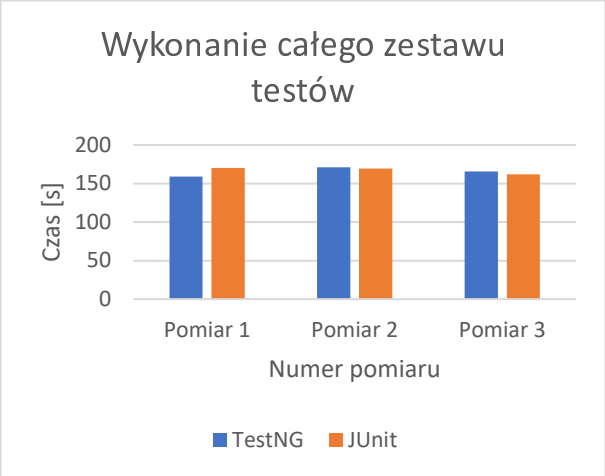
Przypadek testowy	Pomiar 1 (s)	Pomiar 2 (s)	Pomiar 3 (s)
TC 1	24,25	24,78	21,95
TC 2	24,64	20,72	20,54
TC 3	31,99	29,39	29,44
TC 4	18,56	17,57	16,98
TC 5	49,75	57,82	49,48
TC 6	9,68	8,39	12,37
TC 7	11,33	10,56	11,23
Suma	170,19	169,24	161,99

4.4. Porównanie wyników przeprowadzonego badania

Sumy wszystkich otrzymanych wyników cząstkowych dla poszczególnych pomiarów zostały przedstawione w formie wykresów kolumnowych. Każdego pomiaru dokonywano w 10-cio minutowym przedziale czasu a dodatkowo pomiędzy kolejnymi pomiarami został zachowany pewien odstęp czasowy. Ramy czasowe poszczególnych pomiarów zostały zaprezentowane poniżej:

- Pierwszego pomiaru dokonano w godzinach: 21:15 – 21:25.
- Drugiego pomiaru dokonano w godzinach 23:20 – 23:30.
- Trzeciego pomiaru dokonano w godzinach 00:40 – 00:50.

Na Rysunku 2 zamieszczonym poniżej zostały zestawione ze sobą łączne czasy wykonania wszystkich scenariuszy testowych wykonanych przy użyciu zarówno frameworka JUnit jak i frameworka TestNG.



Rysunek 2: Zestawienie wyników trzech pomiarów

Pierwszy pomiar ukazał, że czas potrzebny na uruchomienie wszystkich przypadków testowych był o 11,23s krótszy dla testów z wykorzystaniem frameworka TestNG. Drugi pomiar ukazał z kolei przewagę frameworka JUnit dla, którego czas uruchomienia wszystkich testów był o 1,91s krótszy niż dla frameworka TestNG. Podobną zależność wykazał pomiar trzeci, w którym to różnica wyniosła 3,79s na korzyść frameworka JUnit.

Przeprowadzony eksperyment miał za zadanie wykazać, który z badanych frameworków potrzebuje mniej czasu na wykonanie całego scenariusza testowego. Czasy wykonania testów dla poszczególnych scenariuszy były do siebie bardzo zbliżone i nie odbiegały znacząco od siebie. Pomiar pierwszy wykazał szybsze wykonanie testów opartych o framework TestNG, natomiast pomiary: drugi i trzeci wykazały odwrotną zależność. W czasie pomiaru drugiego i trzeciego to framework JUnit osiągnął nieznacznie lepszy wynik. Wpływ na te wartości mogły mieć różnice w obciążeniu testowanego serwisu internetowego co mogło się przełożyć na późniejsze wczytywanie się elementów strony, których pełne załadowanie było niezbędne do kontynuowania testów.

5. Wnioski

W świetle przeprowadzonej analizy czasów wykonania poszczególnych scenariuszy testowych z użyciem analizowanych szkieletów programistycznych wykazano, że czasy wykonania testów dla obu frameworków były bardzo do siebie zbliżone. Wyniki pierwszego pomiaru pokazały, że łączny czas uruchomienia wszystkich przypadków testowych był o 11,23s krótszy dla frame-

worka TestNG w porównaniu z frameworkiem JUnit, natomiast pomiary: drugi i trzeci pokazały odwrotne zależności i tak łączne czasy uruchomienia testów dla frameworka JUnit były krótsze o 1,91s w drugim pomiarze i o 3,79s w trzecim pomiarze.

Analiza funkcjonalności wykazała, że framework TestNG posiada więcej funkcji możliwych do wykorzystania przez testera bądź przez programistę. Framework TestNG w porównaniu do frameworka JUnit wspiera obsługę adnotacji wykorzystywanych do zarządzania testami w odniesieniu do całych grup jak i do całych scenariuszy testów.

Technologią częściej wybieraną przez użytkowników okazała się być technologia JUnit [9].

Pomimo trzykrotnego uruchomienia testów dla każdego z badanych frameworków nie było możliwe jednoznaczne wskazanie szybszego frameworka. Czasy wykonania testów dla oby szkieletów programistycznych były bardzo do siebie zbliżone i różnice pomiędzy nimi nie były znaczące. To pokazuje, że chcąc wybrać najlepszy framework służący do testów automatycznych spośród TestNG i JUnit czynnik czasu nie powinien być najważniejszym kryterium przy jego wyborze. Użytkownicy powinni skupić się raczej na zakresie funkcjonalności każdego z frameworków, których szerszy zakres posiada framework TestNG. Kolejnym istotnym czynnikiem dla wyboru najlepszego frameworka może też być jego popularność. W tym aspekcie to framework JUnit ma przewagę o czym świadczą liczne tematy na specjalistycznych forach programistycznych dotyczące tego frameworka poprzez co prawdopodobieństwo otrzymania pomocy od innych użytkowników jest większe w przypadku napotkania jakichkolwiek problemów w pracy z tą technologią.

Literatura

[1] ISTQB: Certified Tester - Foundation Level Sylabus, ISTQB, 2018

[2] ISTQB: Advanced Level – Test Automation Engineer Sylabus. ISTQB, 2016

[3] Zbiór informacji o testowaniu, https://pwicherski.gitbook.io/, [28.06.2020]

[4] Informacje o TestNG, https://testerzy.pl/baza-wiedzy/akcja-automatyzacja-czesc-2-junit-testng-porownanie , [30.06.2020]

[5] Informacje o JUnit, https://en.wikipedia.org/wiki/JUnit, [30.06.2020]

[6] B. Garcia, Mastering Software Testing with JUnit 5: Comprehensive guide to develop high quality Java applications, Packt Publishing Ltd, 2017

[7] Dokumentacja frameworka JUnit, https://junit.org/junit5/docs/current/user-guide/, [10.07.2020]

[8] Składnia frameworków JUnit oraz TestNG, https://www.toolsqa.com/testng/testng-vs-junit/, [28.06.2020]

[9] Popularność JUnit oraz TestNG, https://blog.overops.com/junit-vs-testng-which-testing-framework-should-you-choose/, [10.07.2020]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 345–350
Received: 11 August 2020
Accepted: 10 September 2020

An Examination of Selected Websites Availability For People With Various Types of Disabilities

Badanie dostępności wybranych serwisów internetowych dla osób z różnymi rodzajami niepełnosprawności

Sylwia Podkościelna*, Mateusz Proskura*, Grzegorz Kozieł

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

Abstract

The Internet is transforming into the main source of information. Standards are being developed so that people with different types of disabilities are not to feel excluded. Tools based on certain standards automate tasks, but they are also important to work correctly. The study consisted of checking the pages with selected tools discussed in the literature dealing with website accessibility. The validators are identically configured each time the website verifies compliance with the WCAG 2.1 standard. The number of errors detected per page varies significantly when using individual tools. The error rate calculated for the number of all HTML elements of the page ranges between less than 1% and 80%. It is not possible to unambiguously select the worst service on the basis of analyzing all results obtained, it can be a different service for each tool. The tools automate the activities of verifying availability. Each of the validators tested found errors, some of the same. The best solution is not to rely on just one tool, because the results obtained can relate to completely different elements of the page. It all depends on the care of the creators in the preparation of the tool and their care for compliance with standards.

Keywords: availability of websites; website; disabled people; WCAG 2.1

Streszczenie

Internet przekształca się w główne źródło informacji. Opracowywane są standardy, dzięki którym osoby z różnymi rodzajami niepełnosprawnościami nie mają czuć się wykluczone. Narzędzia oparte na standardach automatyzują czynności, ale ważne jest też ich poprawne działanie. Badanie polegało na sprawdzeniu stron wybranymi narzędziami omawianymi w literaturze zajmującej się dostępnością stron internetowych. Walidatory są identycznie skonfigurowane przy każdorazowej weryfikacji zgodności strony ze standardem WCAG 2.1. Liczba błędów wykrywanych na stronie znacząco różni się przy wykorzystaniu poszczególnych narzędzi. Stosunek błędów obliczany do liczby wszystkich elementów HTML strony waha się od wartości poniżej jednego procenta do 80%. Nie jest możliwe jednoznaczne wybranie najgorszego serwisu na podstawie analizy wszystkich otrzymanych wyników, dla każdego narzędzia może być to inny serwis. Narzędzia automatyzują czynności polegające na weryfikacji dostępności. Każdy z badanych walidatorów wykrył błędy, częściowo te same. Najlepszym rozwiązaniem jest nie polegać tylko na jednym narzędziu, gdyż otrzymane wyniki mogą dotyczyć całkowicie innych elementów strony. Wszystko zależy od staranności twórców przy przygotowaniu narzędzia oraz ich dbałości o zgodność ze standardami.

Słowa kluczowe: dostępność stron internetowych; strony WWW; osoby niepełnosprawne; WCAG 2.1

*Corresponding author

Email address: **sylwia.podkoscielna@pollub.edu.pl** (S. Podkościelna), **mateusz.proskura@pollub.edu.pl** (M. Proskura)

1. Wstęp

Internet jest jednym z wielu wynalazków, który ma ogromny wpływ na funkcjonowanie całego świata. Sieć zdominowała niemal wszystkie dziedziny życia. Dlatego nie może dziwić fakt, że w dzisiejszych czasach nikt nie wyobraża sobie życia bez tego medium. Oddziałuje ono na każdą dziedzinę życia. Niektórzy użytkownicy serwisów internetowych mogą mieć problem z korzystaniem z nich ze względów zdrowotnych. Nie wszyscy zdają sobie z tego sprawę. Polskie prawo nakazuje przystosowanie stron do potrzeb odbiorców z niepełnosprawnościami, jednak dotyczy to tylko serwisów publicznych.

Słowo "dostępność" w zakresie serwisów internetowych oznacza, że każdy z użytkowników, który korzysta z powszechnie stosowanych technologii jest w stanie w pełni samodzielnie uzyskać dostęp do treści i funkcjonalności zamieszczonych na stronie. Większość serwisów jest stworzonych po to, aby docierać do jak najszerszego kręgu odbiorców. Pomijając użytkowników specyficznych pod względem swoich potrzeb ograniczamy ten zasięg. Problemy z dostępnością mogą również zniechęcić takie osoby do dalszego korzystania z serwisu i obniżyć jego oglądalność. Serwis stworzony według standardów dostępności ma większe szanse na korzystniejsze pozycjonowanie w wynikach wyszukiwarki, co może zaowocować większą bazą odwiedzających. Plusem dobrze zaimplementowanego serwisu jest łatwość konserwacji i modyfikacji.

2. Przegląd literatury

Twórcy stron internetowych zdając sobie sprawę z wielu problemów osób z niepełnosprawnościami w podstawowych czynnościach przystosowują swoje produkty do ich potrzeb. Zakres tych dostosowań dla podmiotów publicznych jest regulowany w Polsce przez akty prawne. Wymagania zostały określone w Rozporządzeniu Rady Ministrów w Ramach Interoperacyjności z 12 kwietnia 2012 r. Na mocy rozporządzenia wszystkie strony publiczne być dostosowane do ustalonych wymagań. Akt normatywny uznaje WCAG 2.0 (Web Content Accessibility Guidelines). na poziomie AA za najważniejszy dokument określający zasady dostępności. 23 maja 2020 roku weszła w życie nowa ustawa uchwalona przez Sejm. Wskazuje ona WCAG 2.1, jako minimalny standard dostępności cyfrowej. Rozszerza on wcześniej przejęte wytyczne na poziomie AA. Większość narzędzi dostępnych na rynku jest oparte o standard WCAG. Jest to zbiór dokumentów opublikowany przez Web Accessibility Initiative z inicjatywy organizacji World Wide Web Consortium. Fundacja Widzialni w książce „WCAG 2.0 Podręcznik Dobrych Praktyk" [1] zajmuje się zagadnieniem wykluczenia niektórych grup ludzi. Autorzy wyjaśniają poszczególne podpunkty zawarte w standardzie oraz ich sens. Ministerstwo Cyfryzacji odsyła także do podręcznika przygotowanego przez Fundację Integracja pod tytułem „Dostępność Serwisów Internetowych" [2].

Fundacja Instytut Rozwoju Regionalnego w książce „Dostępne WWW" [3] zajmuje się zagadnieniem dostępności serwisów dla osób z różnymi potrzebami. Autorzy objaśniają, co oznacza dostępność, po co to nam potrzebne i do kogo jest skierowane. W kilku rozdziałach omówiona jest budowa statystycznej strony internetowej. Począwszy analizę od najczęściej używanych technologii. Autorzy kładą nacisk na strukturę, jak poprawnie przystosować ją do potrzeb programów wspomagających. W kolejnych rozdziałach można znaleźć informacje pomocne przy tworzeniu elementów tekstowych, multimedialnych i formularzy. Omawiane są potrzeby dostosowania opisywanych elementów nie tylko do potrzeb wizualnych. Także nie mające wizualnego znaczenia fragmenty, mogą mieć ogromne znaczenie w dostępie do treści. Na zakończenie podkreślona jest istota spójnego systemu nawigacji w obrębie całej strony. Pomocne mogą być także skróty klawiszowe przypisane do konkretnych elementów. „Dostępne multimedia" [4] Moniki Szczygielskiej z fundacji Widzialni dotyczą osób z dysfunkcjami słuchu oraz wzroku. Szczegółowo omawiane jest zagadnienie tłumaczenia multimediów na język migowy, w tym wybór odpowiedniej wersji tego języka. W kolejnym rozdziale autorka skupia się na zastosowaniu napisów w materiałach wideo. Według polskiego prawa wszystkie filmy instytucji publicznych muszą być dostosowane do różnych potrzeb. Ważne też jest zastosowanie odpowiedniej formy napisów. W filmach i serialach mamy zazwyczaj dostępne napisy, jako zamiennik lektora i dubbingu. Są to napisy dialogowe, zalecane jest używanie napisów rozszerzonych. Zawierają one dodatkowe informacje, pomagające namierzyć mówcę. Kolejnym omawianym zagadnieniem jest audiodeskrypcja. Podobnie, jak w poprzednich pomocach, autorka wyjaśnia cel i odbiorców. Przygotowane zostały także testy sprawdzające wiedzę czytelników oraz praktyczne aspekty zastosowania przekładu audiowizualnego.

Łukasz Krawiec i Helena Dudycz w artykule „Porównywanie walidatorów do badania dostępności badania stron WWW" [5] zajmują się oceną wybranych walidatorów. Przed użyciem opisywana jest potrzeba ich używania oraz charakterystyka badanych narzędzi. W badaniu między innymi zostały ustalone kryteria zgodności ze standardem WCAG, pozostałymi standardami, formę prezentacji wyniku oraz liczbę równoczesnego badania podstron. Autorzy za każde kryterium przyznawali od zera do dwóch punktów. Na podstawie tego obliczono procentowy wskaźnik jakości narzędzia w stosunku do maksymalnej możliwej oceny. Z zaprezentowanych wyników można wywnioskować, że najwięcej punktów uzyskał walidator Functional Accessibility Evaluator. W książce „Narzędzia do badania dostępności i tworzenia dostępnych treści" [6] autorstwa Grzegorza Kozłowskiego, Mikołaja Rotnickiego, Mariusza Trzeciakiewicza, Piotra Witka oraz Jacka Zadrożnego przedstawione są narzędzia pomocne tworzeniu stron internetowych. Autorzy przedstawiają najpopularniejsze walidatory sprawdzające dostępność. Sprawdzana jest kolorystyka, materiały audiowizualne oraz zgodność ze standardem WCAG. Po przebadaniu autorzy opisują słabe oraz mocne strony z użytych narzędzi.

3. Rodzaje niepełnosprawności

Przykłady grup użytkowników mających najczęściej problemy podczas interakcji z komputerem:

- osoby niewidome i ociemniałe,
- osoby niedowidzące,
- osoby z zaburzeniami widzenia barw,
- osoby niesłyszące,
- osoby niepełnosprawne ruchowo,
- osoby z padaczką fotogenną,
- osoby niepełnosprawne intelektualnie,
- osoby z zaburzeniami poznawczymi.

4. Wymogi i standardy dostępności

Ustanawianiem standardów pisania i przesyłania treści internetowych zajmuje się organizacja World Wide Web Consortium (W3C), założona w 1994 r. W jej skład wchodzi w chwili obecnej ponad 400 organizacji, firm i instytucji naukowych. W 1997 r. organizacja W3C utworzyła grupę nazwaną „Inicjatywa dostępności do sieci" (w skrócie WAI - ang. Web Accessibility Initiative). Głównym celem stowarzyszenia jest zwiększenie szeroko rozumianej dostępności stron WWW dla osób niepełnosprawnych i cyfrowo wykluczonych.

W Polsce wewnętrznym aktem prawnym wskazującym na obowiązek zapewnienia dostępności systemów i serwisów należących do instytucji realizujących zadania publiczne, dla osób niepełnosprawnych jest ustawa z dnia 4 kwietnia 2019 r. Ustawa odnosi się do dostęp-

ności cyfrowej stron internetowych i aplikacji mobilnych podmiotów publicznych. Obowiązuje ona od 23 maja 2012 roku [7]. Jest ona rozszerzeniem Rozporządzenia Rady Ministrów w sprawie Krajowych Ram Interoperacyjności (Dz.U. 2012 nr 0 poz. 526), które obowiązywało od 30 maja 2012 roku.

Celem rozporządzenia jest doprowadzenie do wystandaryzowania i interoperacyjności systemów należących do instytucji publicznych. Ustawa mówi o przyjętym przez Polskę standardzie projektowania serwisów internetowych i stron tak, aby korzystać z nich mogły również osoby niepełnosprawne (nie używając przy tym dodatkowych urządzeń i aplikacji). Są w niej zawarte międzynarodowe wytyczne dostępności WCAG 2.1.

4.1. WCAG 2.1

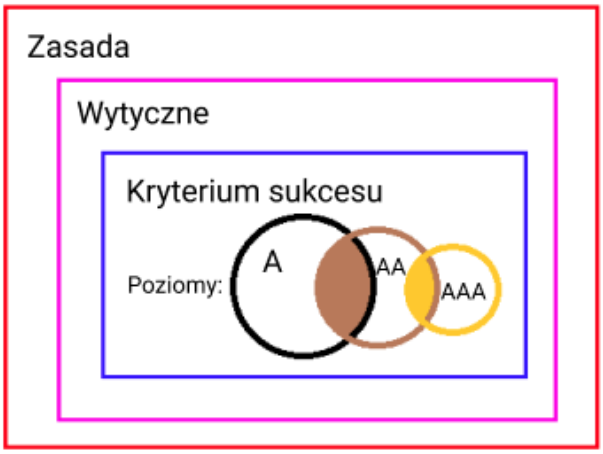
WCAG 2.1 to standard, który definiuje wytyczne dotyczące tworzenia dostępnych serwisów WWW. Dokument ten można znaleźć pod adresem https://www.w3.org/TR/WCAG21/ [8].
Rekomendacje zawarte w standardzie WCAG 2.1 są podzielone na cztery główne zasady:
Postrzegalność - informacje oraz elementy interfejsu użytkownika muszą być pokazane w sposób dostępny dla zmysłów użytkowników.
Funkcjonalność - nawigacja oraz elementy interfejsu użytkownika muszą być możliwe do użycia.
Zrozumiałość - interfejs użytkownika oraz informacje na stronie muszą być zrozumiałe.
Kompatybilność - strona musi być zbudowana poprawnie, aby mogła być skutecznie interpretowana przez inne technologie - na przykład przez robota czytającego artykuły osobie niewidomej.
Każda z wyżej wymienionych zasad podzielona jest na wytyczne, te z kolei zawierają kryteria sukcesu z różnymi poziomami zgodności (rysunek 1):

- Poziom A - obejmuje rekomendacje, które twórcy stron muszą spełnić,
- Poziom AA - uwzględnia kryteria, które powinny zostać spełnione, inaczej niektóre grupy użytkowników mogą napotkać trudności przy dostępie do treści strony,
- Poziom AAA - zbiór rekomendacji, które mogą zostać spełnione, aby ułatwić niektórym użytkownikom dostęp do stron.

Niższe poziomy zgodności zawierają się w wyższych, czyli strona zgodna z wytycznymi na poziomie AA musi spełniać wszystkie kryteria z poziomów A oraz AA. Zaś strona zgodna z WCAG 2.1 na poziomie AAA musi spełniać kryteria z poziomów A, AA oraz AAA.

5. Technologie wykorzystywane w badaniu dostępności

HTML Validator [9] **-** narzędzie stworzone przez konsorcjum W3C wykorzystywane do weryfikowania poprawności znaczników używanych w języku HTML, HTML 5 oraz XHTML.
Utilitia [10] - narzędzie to pozwala zweryfikować tylko te kryteria standardu WCAG 2.1, które są możliwe do zbadania w sposób zautomatyzowany i zaprogramowany. Walidator automatycznie analizuje kod źródłowy wskazanej podstrony lub grupy podstron pod kątem wybranych kryteriów.



Rysunek 1: Składowe WCAG 2.1

Wave Toolbar [11] - narzędzie opracowane przez organizację WebAIM (ang. Web Accessibility in Mind). umożliwiaja ocenę zawartości stron internetowych pod kątem problemów z dostępnością. Pozwala ocenić lokalnie wyświetlane style i dynamicznie generowaną zawartość ze skryptów lub AJAX.
aChecker [12] - został opracowany w 2011 roku przez IDI (Inclusive Design Institute), kanadyjski ośrodek badawczy zajmujący się teleinformatyką (Information and Communication Technologies - ICT), głównie w kontekście optymalizacji szeroko rozumianej dostępności informacji.
Functional Accessibility Evaluator [13] – walidator został stworzony w 2014 roku na Uniwersytecie w Illinois (Champaign) i jest nadal rozwijany jako open source. Functional Accessibility Evaluator jako jedyny w tym zestawieniu nie sprawdza zgodności ze standardem WCAG. Stanowi za to doskonałe uzupełnienie analiz, sprawdza stronę pod kątem standardów technicznych takich jak HTML4, HTML5 i ARIA.
Check My Colours [14] - walidator umożliwia przetestowanie kontrastu pomiędzy kolorami zastosowanymi na stronie internetowej, znajdującej się pod konkretnym adresem URL.

6. Badanie dostępności wybranych serwisów publicznych

Jedną z metod badania dostępności jest zastosowanie walidatorów, automatycznych testów sprawdzających poprawność składni dokumentów umieszczonych w Internecie. Do oceny dostępności niżej wymienionych stron posłużyły walidatory opisane w rozdziale 4. Strony WWW użyteczności publicznej wybrane do badań to:

- https://lublin.eu/ [15]
- http://investin.zamosc.pl/pl/ [16]
- https://wupwarszawa.praca.gov.pl/ [17]
- https://gorzkow.eu/ [18]

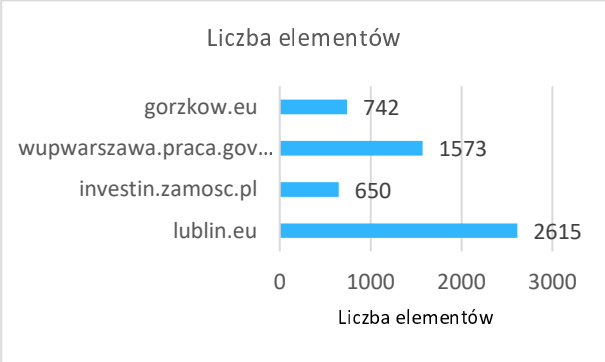
6.1. Metodyka badań

Każda strona została sprawdzona przez wszystkie przygotowane narzędzia. W celu jak najbardziej wiarygodnego porównania dostępności badanych stron, ustalono spójną metodykę oraz jednakową konfigurację początkową walidatorów. Badane strony różnią się rozmiarem. Porównywanie otrzymanej liczby wykrytych błędów dla poszczególnych stron byłoby mało miarodajne, dlatego zastosowano zliczanie elementów za pomocą kodu JavaScript. Przedstawiony kod na Listingu 1 zlicza znaczniki HTML według nazwy tagu.

Listing 1: Kod JavaScript zwracający liczbę elementów strony

```
document.getElementsByTagName('*').length;
```

Rozmiar badanych stron wyrażony w postaci liczby elementów widoczny jest na wykresie 1. Wynika z niego, że najbardziej rozbudowaną stroną jest lublin.eu. Nieco mniej rozbudowany jest wortal prowadzony przez Wojewódzki Urząd Pracy w Warszawie. Strona gminy Gorzków oraz serwis gospodarczy miasta Zamość składają się z porównywalnej liczby elementów i jednocześnie są najmniej rozbudowanymi stronami.



Wykres 1: Liczba elementów badanych stron

6.2. Badanie dostępności

Walidator aChecker nie mógł sprawdzić składni strony gminy Gorzków. Strona zawiera trzy krytyczne błędy w składni. Narzędzie nie poradziło sobie z jej wczytaniem oraz wykryło liczne braki w tekstach alternatywnych grafik umieszczonych na stronie. W strukturze stron znaczniki h1, h2, h3 są używane w niewłaściwy sposób. Autorzy za ich pomocą w niektórych miejscach po prostu wyróżniają tekst w sposób graficzny. Według organizacji W3Schools prawidłowym rozwiązaniem jest używanie tych znaczników w kolejności h1, h2, h3. W serwisie Zamościa są użyte tylko znaczniki h2.

Odpowiednia kolejność i hierarchia tych znaczników jest niezbędna do prawidłowego działania programów wspomagających przeglądania stron internetowych.
Innym często wykrywanym problemem jest dobór kolorów ze zbyt małym kontrastem. Dla niektórych użytkowników może stanowić to barierę, która uniemożliwi odczytanie treści.

Wynik badania przeprowadzonego narzędziem Wave Toolbar strony Wojewódzkiego Urzędu Pracy zaprezentowany jest na rysunku 2. Wszystkie sześć wykrytych błędów dotyczy braku opisów alternatywnych dla obrazków umieszczonych na stronie. Dwadzieścia pięć błędów dotyczy samego kontrastu.

Rysunek 2: Badanie Wave Tolbar

Najwięcej błędów w strukturze strony zawiera serwis gminy Gorzków. Także brakuje tekstów alternatywnych, ale w niektórych przypadkach są one dodane wielokrotnie w różnych miejscach. Może to powodować dezorientację użytkowników oraz sprawiać problem z poruszeniem się po stronie użytkowników korzystających z programów wspomagających.

Narzędzia także wykryły pozytywne aspekty w budowie stron. Dostępne są różne wersje językowe i kolorystyczne. Każdy może dostosować strony do swoich potrzeb. Po zastosowaniu semantyki ARIA dostarczane są informacje strukturalne strony, które ułatwiają identyfikację elementów przez programy asystujące.

Wyniki narzędzia Utilitia zaprezentowane na rysunku 3 są przedstawione w kolejności miasta Lublin, Zamość, Urzędu Pracy oraz gminy Gorzków. Badanie składa się z dwudziestu dwóch testów. Strona miasta Lublin uzyskała łączny wynik 6,8. Strona ta wypadła nieznacznie gorzej od strony dla inwestorów miasta Zamość.

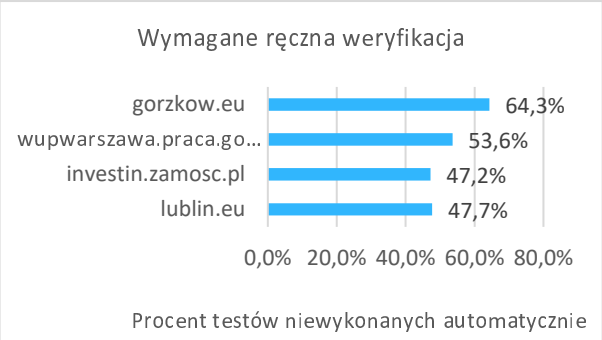
Rysunek 3: Wyniki stron w badaniu narzędziem Utilitia

Walidator kolorem zielonym oznaczył pozostałe dwie strony. Strony Urzędu Pracy i gminy uzyskały odpowiednio 7,9 oraz 7,7 punktów na dziesięć możliwych.

6.3. Podsumowanie badań

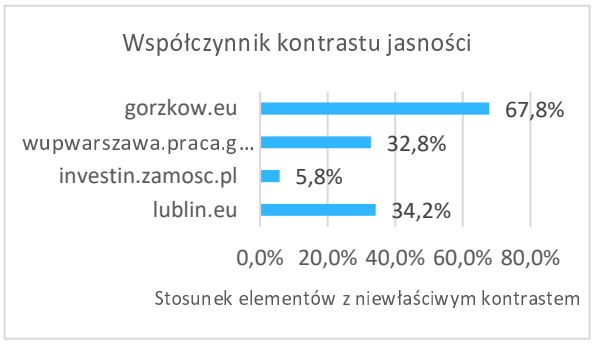
W badaniu wykryto niską skuteczność narzędzia Functional Accessibility Evaluator. W przypadku storn Lublina i Zamościa blisko połowa testów nie wykonała się automatycznie (wykres 2). Walidator jedynie informuje o konieczności manualnego wykonania tych testów.

Najgorzej narzędzie wypadło w przypadku strony gminy Gorzków. Prawie dwie trzecie testów wymaga powtórzenia. Problem ten dotyczy także wortalu Urzędu Pracy, nie udało się wykonać 54% testów.



Wykres 2: Testy wymagające ręcznej weryfikacji

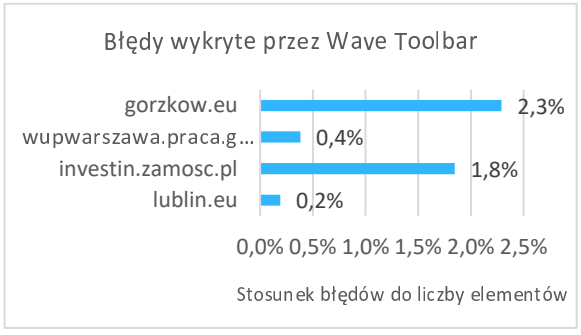
Na wykresie 3 przedstawione zostało porównanie współczynnika kontrastu badanych stron. Ponad dwie trzecie elementów strony gminy Gorzków ma zbyt niski współczynnik kontrastu jasności. Osoby mające problem z rozpoznawaniem barw mogą nie rozróżnić poszczególnych elementów.



Wykres 3: Współczynnik błędnego kontrastu

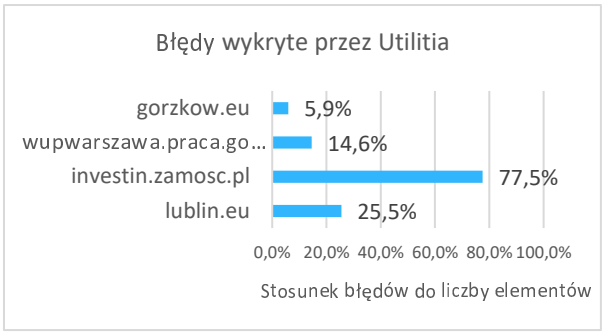
W zestawieniu współczynnika kontrastu najlepszy wynik uzyskała strona gospodarcza miasta Zamość. Omawiany problem dotyczył jedynie około 6% struktury strony. Wyniki osiągnięte przez dwie pozostałe badane strony są zbliżone do 33%.

Wykres 4 przedstawia stosunek błędów wykrytych przez narzędzie Wave Toolbar dla badanych stron. Można z niego wywnioskować, że wyniki strony gminy Gorzków oraz miasta Zamość są zbliżone do 2%. Natomiast pozostałe dwie strony osiągnęły wynik poniżej 0,5%.



Wykres 4: Błędy wykryte przez Wave Toolbar

Przeprowadzone badanie z użyciem narzędzia Utilitia dostarcza całkowicie różniące się wyniki od walidatora Wave Toolbar. Najgorzej wypadła storna miasta Zamość, uzyskując ponad trzy czwarte błędów w przeliczeniu do rozmiaru strony. Wynik otrzymany jest znacznie większy od wyniku uzyskanego przy weryfikacji za pomocą poprzedniego narzędzia.



Wykres 5: Błędy wykryte przez Utilitia

Strona miasta Lublin zawiera 25,5% błędnych elementów, natomiast strona Urzędu Pracy niecałe 15%. Najlepszy stosunek uzyskała strona gminy Gorzków. Jest to bardzo zaskakujące, ponieważ w wynikach walidatora Wave Toolbar ta strona osiągnęła najgorszy wynik (patrz wykres 5).

7. Wnioski

Pomimo wprowadzonych standardów dostępności i zastosowaniu dobrych praktyk użyteczności, nie jest możliwe wskazanie takiej strony WWW, która w całości spełnia wymogi prawne i standardy WCAG 2.1 Nie jest możliwe stworzenie strony, która odpowiada na oczekiwania i potrzeby wszystkich użytkowników, natomiast jest możliwe, aby była przystosowana dla możliwie jak największej liczby użytkowników, niezależnie od ich wieku, niepełnosprawności, oprogramowania czy używanego sprzętu. W wielu testach za najgorszą można uznać stronę gminy Gorzków, jednak przeprowadzając analizę na podstawie innych kryteriów uzyskuje ona znacznie lepsze wyniki niż pozostałe badane strony. Każdy z walidatorów skupia się na innych parametrach strony. Stąd wynikają rozbieżności w otrzymanych wynikach. Walidatory te nie powinny być jedynym źródłem informacji o dostępności strony. Narzędzia te wykryją jedynie najpopularniejsze problemy, przewidziane przez ich twórców. Nie oznacza to, że strona, która przejdzie testy wzorowo, będzie dostępna dla wszystkich. Strona może być atrakcyjna wizualnie dla zdrowego odbiorcy jednak przez zastosowanie nieodpowiednich kolorów część osób może nie zobaczyć tego, co najważniejsze – treści. Najlepszym rozwiązaniem jest przygotowanie kilku wersji kolorystycznych strony. Dostępne w Internecie narzędzia znacząco ułatwiają weryfikację stron. Należy mieć na uwadze, że najprawdopodobniej zostały one stworzone przez osoby nie mające trudności z odbiorem treści. Oznacza to, że mogą oni nie dostrzegać niektórych problemów przy projektowaniu tych narzędzi. Walidatory są pomocne, jednak ostateczna decyzja o wyglądzie i dostępności

treści powinna zostać podjęta na podstawie docelowej grupy odbiorców.

Literatura

[1] A. Marcinkowski, P. Marcinkowski, WCAG 2.0 Podręcznik Dobrych Praktyk, Fundacja Widzialni, 2012.

[2] J. Dębski, D. Paszkiewicz, Dostępność Serwisów Internetowych, Fundacja Integracja, 2013.

[3] J. Zadrożny, Dostępne WWW, Fundacja Instytut Rozwoju Regionalnego, 2009.

[4] M. Szczygielska, Dostępne multimedia, Fundacja Widzialni, 2016.

[5] H. Dudycz, Ł. Krawiec, Porównywanie walidatorów do badania dostępności badania stron WWW, Studia Informatica Pomerania nr 3/2016 (41).

[6] G. Kozłowski, M. Rotnicki, M. Trzeciakiewicz, P. Witek, J. Zadrożny, Narzędzia do badania dostępności i tworzenia dostępnych treści, Fundacja Instytut Rozwoju Regionalnego, 2014.

[7] Rozporządzenie Rady Ministrów z dnia 12 kwietnia 2012 r. w sprawie Krajowych Ram Interoperacyjności, minimalnych wymagań dla rejestrów publicznych i wymiany informacji w postaci elektronicznej oraz minimalnych wymagań dla systemów teleinformatycznych, isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=wdu20120000526, [03.06.2020].

[8] Dokumentacja WCAG 2.1, http://www.w3.org/TR/WCAG21, [03.06.2020].

[9] Html Validator, validator.w3.org, [03.06.2020].

[10] Dokumentacja Utilitia, utilitia.pl/mozliwosci/opis-dzialania-wszystkich-testow-uslugi-utilitia, [03.06.2020].

[11] Walidator Wave Toolbar, wave.webaim.org, [03.06.2020].

[12] Walidator aChecker, achecker.ca, [03.06.2020].

[13] Walidator Functional Accessibility Evaluator, fae.disability.illinois.edu, [03.06.2020].

[14] Walidator Check My Colours, www.checkmycolours.com, [03.06.2020].

[15] Badana strona urzędu miasta Lublin, lublin.eu, [03.06.2020].

[16] Badana strona urzędu miasta Zamość, investin.zamosc.pl, [03.06.2020].

[17] Badana strona urzędu pracy miasta Warszawa, wupwarszawa.praca.gov.pl, [03.06.2020].

[18] Badana strona gminy Gorzków, gorzkow.eu, [03.06.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 351–357
Received: 2 September 2020
Accepted: 21 September 2020

Comparative analysis of web application performance testing tools

Analiza porównawcza narzędzi do badania wydajności aplikacji internetowych

Agata Kołtun*, Beata Pańczyk

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

Abstract

Recent years have brought the rise of importance of quality of developed software. Web applications should be functional, user friendly as also efficient. There are many tools available on the market for testing the performance of web applications. To help you choose the right tool, the article compares three of them: Apache JMeter, LoadNinja and Gatling. They were analyzed qualitatively in terms of a user-friendly interface, parameterization of the requests and creation of own testing scripts. The research was carried out using a specially prepared application. The summary indicates the most important advantages and disadvantages of the selected tools.

Keywords: performance; Apache JMeter; LoadNinja; Gatling

Streszczenie

Ostatnie lata pokazują, jak ważna jest jakość wytwarzanego oprogramowania. Aplikacje internetowe, oprócz funkcjonalności przyjaznej użytkownikowi, muszą być również wydajne. Na rynku oferowanych jest wiele narzędzi do badania wydajności aplikacji internetowych. Aby ułatwić wybór odpowiedniego narzędzia, w artykule porównano trzy z nich: Apache JMeter, LoadNinja oraz Gatling. Przeanalizowano je jakościowo pod kątem przyjaznego interfejsu użytkownika, możliwości parametryzacji oraz tworzenia własnych skryptów. Do badań wykorzystano autorską aplikację, a w podsumowaniu wskazano najważniejsze zalety i wady wybranych narzędzi.

Słowa kluczowe: wydajność; Apache JMeter; LoadNinja; Gatling

*Corresponding author

Email address: **agata.koltun95@gmail.com** (A. Kołtun)

1. Wstęp

W związku z kluczową rolą, jaką pełni odporność aplikacji na zadane obciążenie, istotnym aspektem pracy nad rozwojem aplikacji jest wybór właściwego narzędzia wspomagającego testowanie wydajności aplikacji. Wybór ten nie jest prosty, zważywszy na to, że dostępne w sieci artykuły nie dają jednoznacznej odpowiedzi, które z narzędzi powinno zostać wybrane do przeprowadzenia takich testów.

Na rynku oferowanych jest wiele narzędzi (w tym płatnych), za pomocą których developerzy są w stanie napisać symulacje odpowiadające działaniom wielu użytkowników – na przykład tysiąca kupujących przeglądających asortyment i dokonujących zakupów w sklepie internetowym. Rozwiązania bazujące na odpowiednio przygotowanym, sparametryzowanym skrypcie wydają się znacznie wygodniejszym i tańszym rozwiązaniem (zwłaszcza w kontekście powtarzania testów w wielu iteracjach). Koszt pracy developera niezbędny do przygotowania i utrzymania takich skryptów należy również uwzględnić podczas wyboru odpowiedniego narzędzia. Przygotowanie skryptu nie powinno wymagać zaawansowanych umiejętności technicznych, a ewentualne zmiany w testowanej aplikacji powinny być raczej kosmetyczne. Inspekcja kodu (ang. *code review*) dokonywana przez innego programistę nie powinna być zbyt czasochłonna.

Większość publikacji odnosi się do JMeter [1, 2]. Dodatkowo pozycje internetowe, porównujące takie narzędzia dotyczą z reguły zestawienia rozwiązania firmy Apache z innymi narzędziami [3, 4, 5].

2. Testy wydajnościowe

„Testy wydajnościowe to rodzaj testów, które mają na celu scharakteryzowanie responsywności, niezawodności, przepustowości, interoperacyjności i skalowalności systemu oraz/lub aplikacji po zadaniu danego obciążenia. Może również określać prędkość lub efektywność komputera, sieci, aplikacji lub urządzenia” [1].

Testy obciążeniowe skupione są na trzech aspektach określających jakość systemu:

- **prędkości** (ang. *speed*) – jak szybko odpowiada aplikacja,
- **stabilności** (ang. *stability*) - czy aplikacja działa stabilnie przy dużym obciążeniu,
- **skalowalności** (ang. *scalability*) – cecha określająca maksymalną liczbę użytkowników, którą aplikacja jest w stanie obsłużyć.

Jedną z podstawowych pobudek do przeprowadzenia testów wydajności jest *Service Level Agreement*, czyli umowa o gwarantowanym poziomie świadczenia usług, którą wprowadza się w celu określenia czy spełnione są wymagania funkcjonalne określone przez usługobiorcę.

Niewątpliwą zaletą, którą mogą ukazać takie testy, jest wskazanie pojedynczych punktów błędów w przypadku szczytowego obciążenia oraz określenia

tak zwanego wąskiego gardła, które definiuje się jako element procesu, który ma ograniczoną wydajność, a zatem ogranicza przepustowość systemu jako całości [6].

W przypadku aplikacji internetowych test odnosi się zwykle do pojedynczego modułu, a nawet zapytania (np. przeprowadzanej skomplikowanej transakcji bazodanowej), które powoduje, że cały system staje się mniej efektywny.

Najczęściej spotykanymi **wąskimi gardłami** są [7]:

- wysokie obciążenie procesora,
- wykorzystanie pamięci,
- wysokie obciążenie sieci,
- ograniczenia systemu operacyjnego,
- wykorzystanie dysku.

3. Proces przeprowadzania testów

Liczne źródła internetowe, a także pozycje literaturowe opisują następujący proces testowania wydajności [8]:

1. Wybranie środowiska, które będzie testowane (np. środowisko testowe, które odpowiada parametrom środowiska produkcyjnego).
2. Określenie wymaganej, akceptowanej wydajności.
3. Zaplanowanie testów – opracowanie scenariuszy testowych, wybranie liczby użytkowników końcowych.
4. Konfiguracja środowiska testowego – przygotowanie zasobów przed uruchomieniem testów.
5. Implementacja testów.
6. Przeprowadzenie testów – uruchomienie i monitorowanie przebiegu testów.
7. Analiza wyników i ponowne przeprowadzenie testów po poprawkach w przypadku zdiagnozowania źródeł wąskiego gardła lub innych problemów.

Zaznaczyć należy, że każdy krok jest jednakowo istotny. Właściwe zaplanowanie pracy jest bardzo ważnym działaniem, które determinuje jakość przeprowadzonych badań.

4. Przegląd literatury

Autor pozycji [8] opisuje proces zrozumienia charakterystyk wydajności aplikacji oraz jej modułów jako bezcenny. W swojej pracy dokonuje porównania rozwiązań wspierających wspomniane badanie. Pierwszym z opisywanym narzędzi jest Apache Benchmark, program dedykowany do testowania obciążeniowego serwerów HTTP, w szczególności serwera Apache. Narzędzie określone jest tam jako mało elastyczne, lecz cechujące się satysfakcjonującą prostotą działania. Jako alternatywne, wyróżniające się rozwiązanie, zaproponowany jest Gatling, który oferuje wiele sposobów konfiguracji interakcji wirtualnych użytkowników z witryną oraz współbieżnością działań. W pozycji [8] Gatling określony jest jako potężne narzędzie ze względu na język dziedzinowy DSL (ang. *domain specific language*) [9].

DSL umożliwia:

- określenie asercji (dotyczących np. statusów HTTP),
- zawartości fragmentu przesyłanych danych (ang. *payload*),
- konfigurację dopuszczalnych opóźnień.

Autor skrótowo podsumowuje również Apache JMeter, określając go jako narzędzie dobre, lecz mniej elastyczne od Gatling. Stosowanie Apache JMeter wskazane jest w przypadku, gdy programista nie chce uczyć się nowego DSL.

Pozycja [1] również poświęcona jest testowaniu wydajności za pomocą JMeter. Wspomniano tu także o innych rozwiązaniach, np. LoadRunner, LoadUI, Gatling, OpenSTA oraz Apache Benchmark. Jednak podstawową ich wadą jest komercyjność, brak dojrzałości rozwiązania, niewystarczającą przenośność i rozszerzalność rozwiązania w stosunku do Apache JMeter. HP LoadRunner jest kosztowny, lecz oferuje lepszy interfejs użytkownika oraz ciekawsze możliwości monitorowania w stosunku do JMeter. Gatling z kolei jest opisany jako projekt nowy, „wciąż w powijakach", choć obiecujący. Główną zaletą Gatling jest łatwiejsze testowanie języka specyficznego dla domeny (w porównaniu do dużych plików XML JMeter) oraz atrakcyjne wizualizacje rezultatów. Za wadę podano niewielką liczbę użytkowników narzędzia oraz konieczność znajomości języka Scala.

Większość porównań dotyczących poszczególnych narzędzi stanowią jednak publikacje dostępne w Internecie. W [5] jako najlepsze podano płatne rozwiązanie WebLOAD, oparte na elastycznej platformie obsługującej setki technologii i umożliwiającej integrację z większością najważniejszych narzędzi. LoadNinja zajmuje trzecie miejsce w rankingu. Opisane jest jako rozwiązanie umożliwiające tworzenie zaawansowanych testów bez użycia skryptów, skracające czas testowania o 50% i zastępujące emulatory obciążenia prawdziwymi przeglądarkami. Dzięki LoadNinja zespoły projektowe mogą bardziej skupić się na tworzeniu i rozwijaniu aplikacji, niż na tworzeniu skryptów je testujących. W [5] Apache JMeter wskazano jako szóste z najlepszych narzędzi, które oferuje przyjazny interfejs użytkownika, wysoką przenośność i generujące proste wykresy (wystarczające do analizy kluczowych statystyk monitorowania obciążenia aplikacji). Pozycja [5] nie wspomina o Gatling.

5. Wybór narzędzi do porównania

Narzędzia można podzielić na dwie podstawowe grupy: bazujące na płatnej licencji oraz te typu open source [10]. Pierwsze z nich zwykle oferują wersję próbną, umożliwiającą sprawdzenie możliwości narzędzia przed podjęciem decyzji, a poniesiony koszt zależy od liczby wirtualnych użytkowników dostępnych dla scenariusza testu. Liczba testów, które należy przeprowadzić, rodzaj testu i możliwości raportowania często nie wpływają na wycenę procesu.

W niniejszym artykule do porównania jakościowego wybrano trzy najpopularniejsze rozwiązania służące do badania wydajności aplikacji internetowych, które reprezentują odmienne podejście twórców narzędzi do interfejsu użytkownika:

- Apache JMeter - rozwiązanie darmowe, bardzo popularne i dobrze opisane w wielu publikacjach,

posiada interfejs użytkownika w formie aplikacji okienkowej;
- Gatling - narzędzie darmowe, coraz bardziej popularne [3] i ciekawe ze względu na brak interfejsu użytkownika;
- LoadNinja - płatne rozwiązanie platformowe.

5.1. Apache JMeter

Apache JMeter [11] jest narzędziem stworzonym przez Apache Software Foundation, które może zostać wykorzystane do analizy obciążenia różnych serwisów, w szczególności aplikacji internetowych. Na dzień 19 stycznia 2020 roku najnowszą, stabilną, wydaną wersją jest 5.2 na licencji Apache License 2.0.

JMeter jest narzędziem darmowym bazującym na licencji typu open source, co oznacza, że można nie tylko korzystać z niego za darmo (również w przypadku zastosowań komercyjnych), ale też kopiować, analizować, modyfikować i rozbudowywać dostępne moduły do własnego użytku.

Najważniejszymi komponentami, które służą do budowania testu są [12]:
- kontrolery typu Samplers (generowanie prostych zapytań HTTP, FTP, zapytań do bazy danych, wywołanie implementacji JavaSamplerClient lub wysyłanie/odczyt wiadomości z kolejki typu Java Message Service),
- kontrolery Logic Controllers (generowanie zapytań po spełnieniu określonych warunków),
- asercje (weryfikacja poprawności odpowiedzi na zapytanie),
- timers (umożliwiają opóźnienia lub synchronizację zapytań, szczególnie przydatne podczas symulacji czasu, jaki użytkownik spędza przeglądając stronę internetową, bez wykonywania interakcji z nią),
- listeners (elementy umożliwiające wygenerowanie reprezentacji m.in. graficznych wykresów przedstawiających wyniki testów, zapisu odpowiedzi na zapytanie lub spełnienia asercji).

Od wersji 3, domyślnym językiem pisania skryptów jest Groovy. Narzędzie posiada prosty graficzny interfejs, który może być przydatny dla mniej zaawansowanych użytkowników. HTTP Proxy umożliwia nagranie działań użytkownika na stronie internetowej – przez co utworzenie podstawowych testów może sprowadzić się do podania parametrów dla gotowych, wygenerowanych komponentów oraz przygotowania liczby wątków wykonujących dany test.

5.2. LoadNinja

LoadNinja [13] jest rozwiązaniem chmurowym, które zostało opracowane przez SmartBear Software. Umożliwia testy symulujące działania użytkownika aplikacji (ang. *UI Testing*) oraz testy API, symulując żądania klienta przesyłającego zapytania HTTP do serwera.

Wersja trial jest edycją, która nie nadaje się do użytku komercyjnego. Na dzień 10 lipca 2020 roku dostępne są trzy rodzaje licencji, każda oferująca 8 godzin testów dla wariantu miesięcznego oraz 100 godzin dla rocznego. Najtańsza subskrypcja, która umożliwia przeprowadzenie testu na 1000 wirtualnych użytkownikach to koszt 739€.

Najważniejsze komponenty tego narzędzia to:
- Recorder InstaPlay (tworzenie testu i jego późniejsze odtwarzanie),
- VU Debugger (umożliwia debugowanie testu w czasie rzeczywistym),
- VU Inspector (monitorowanie aktywności wirtualnych użytkowników w czasie rzeczywistym).

Narzędzie nie wymaga instalowania żadnego oprogramowania na urządzeniu – nie jest wymagana nawet instalacja Javy, co wyróżnia je na tle pozostałych porównywanych narzędzi. Potrzebne jest jedynie konto użytkownika oraz jedna z przeglądarek internetowych: Chrome, Firefox, Safari lub Microsoft. Czyni to LoadNinja narzędziem polecanym do przeprowadzania testów przez analityków biznesowych, bez wsparcia technicznego.

5.3. Gatling

Gatling [14] jest narzędziem do badania wydajności i testowania obciążenia, wyprodukowanym przez firmę Gatling (Gatling FrontLine – w przypadku wersji Enterprise). Na dzień 14 kwietnia 2020 roku ostatnią, stabilną wersją był Gatling 3.1.3, wydany w 2019 roku.

Narzędzie opiera się o licencję typu open source. Oprócz oficjalnie wydanych dodatków, wokół narzędzia skoncentrowana jest spora społeczność, której rezultatem pracy jest m.in. darmowy dodatek integrujący Gatling z systemem kolejkowym RabbitMQ.

Podstawowymi komponentami są: Recorder oraz Domain Specific Language [10]. Język dziedzinowy sprawia, że testy są łatwiejsze w odbiorze, proste do utrzymania oraz zrozumiałe biznesowo.

Skrypty pisane są w języku Scala, który działa na wirtualnej maszynie Javy i ma cechy języków obiektowych i funkcyjnych.

Narzędzie nie udostępnia graficznego interfejsu użytkownika. Nagrywarka wyzwalana jest z aplikacji desktopowej. Testy modyfikowane są za pomocą edytora tekstowego, w szczególności polecane jest IDE, które można rozszerzyć o podpowiadanie składni języka Scala np. IntelliJ. Uruchomienie testów należy przeprowadzić z poziomu konsoli, raport generowany jest samoistnie.

6. Porównanie wybranych narzędzi

Do badań wykorzystano specjalnie przygotowaną aplikację internetową. W każdym z narzędzi opracowano skrypty do przeprowadzenia testów zgodnie z zadanym scenariuszem.

Scenariusz testowy zakłada wykonanie przez wirtualnego użytkownika następujących akcji:
- rejestrację konta użytkownika,
- zalogowanie się do aplikacji,
- wypełnienie anonimowej ankiety,
- wypełnienie formularza,
- wylogowanie się z systemu.

6.1. Aplikacja testowa

Na potrzeby badań utworzono prostą aplikację webową, której część serwerowa napisana została w języku Java 8. Jako narzędzie automatyzujące budowanie projektu wybrano Apache Maven. Wykorzystano Spring Framework, a także Spring Boot – w celu szybkiego uruchomienia gotowej aplikacji, niewymagającej dodatkowej konfiguracji. Warstwa wizualna aplikacji oparta jest o formularze JSP. Pracę z danymi obsługuje PostgreSQL v42.2.6.

Baza danych przechowuje informacje o użytkownikach systemu, czyli wirtualnych użytkownikach, których akcje symulowane są w ramach scenariusza oraz wynikach formularza osobowego i ankiety badającej poziom satysfakcji użytkownika.

Realizacja scenariusza, którego przebieg opisano w punkcie 6, wymagała utworzenia formularza JSP oraz odpowiedniej akcji HTTP dla każdego z opisywanych kroków.

6.2. Realizacja badań

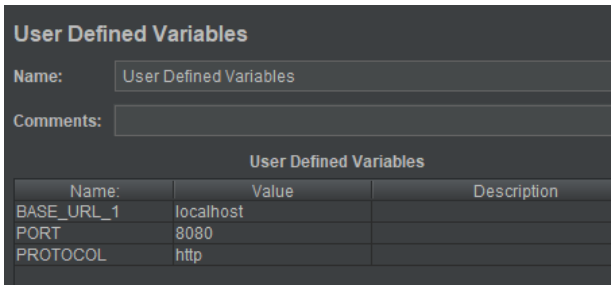
Testy wydajnościowe przygotowano i przeprowadzono za pomocą trzech, wcześniej opisanych narzędzi.

Apache JMeter

Test można utworzyć „ręcznie”, wyklikując kolejne komponenty w graficznym interfejsie użytkownika lub tworząc własny plik w formacie JMX. O wiele efektywniejsze jest jednak nagranie testu za pomocą nagrywarki, a następnie parametryzacja zapytań.

Jedną z możliwości nagrania scenariusza jest wykorzystanie szablonu *Recording with Think Time*, które wymaga wygenerowania certyfikatu i zaimportowania go do przeglądarki. Łatwiejszym rozwiązaniem jest skorzystanie z narzędzia BlazeMeter [12], które jest dodatkiem do przeglądarki Chrome.

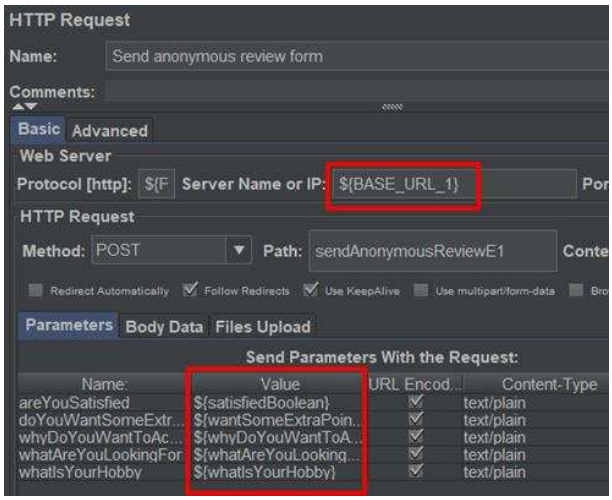
Są trzy podstawowe sposoby parametryzacji scenariusza. Blok User Definded Variables (Rys. 1) umożliwia definiowanie parametrów globalnych np. IP serwera i rodzaju protokołu sieciowego.



Rysunek 1: Parametryzacja za pomocą bloku User Defined Variables

Obsługa plików płaskich (CSV oraz TSV) jest przydatna do przeprowadzania testów wymagających unikalnych danych, które są wykorzystywane przez każdego wirtualnego użytkownika. Przekazywanie parametrów z poziomu konsoli może zostać wykorzystane np. do ustalenia liczby wirtualnych użytkowników wskazanych przy przeprowadzeniu konkretnej iteracji testów. Niezależnie od typu parametru i zastosowanej metody, sposób użycia parametru (Rys. 2) jest taki sam, co czyni testy elastycznymi, a parametry można łatwo nadpisać.

Narzędzie oferuje różnego typu wizualizacje rezultatów testów, zarówno w formie wykresów, jak i tabel. Wykresy nie są atrakcyjne wizualnie (Rys. 3). Na szczególną uwagę zasługuje View Results Tree, które umożliwia weryfikację każdego wykonanego zapytania osobno (wartości parametrów oraz odpowiedzi na zadane żądanie) w trakcie i po przeprowadzeniu testu.



Rysunek 2: Przykład parametryzacji zapytań Apache JMeter

LoadNinja

Aby rozpocząć pracę z narzędziem LoadNinja [13] konieczne jest utworzenie konta użytkownika. W przypadku testowania aplikacji na hoście lokalnym (localhost) należy lokalnie uruchomić aplikację LoadNinja Tunnel.

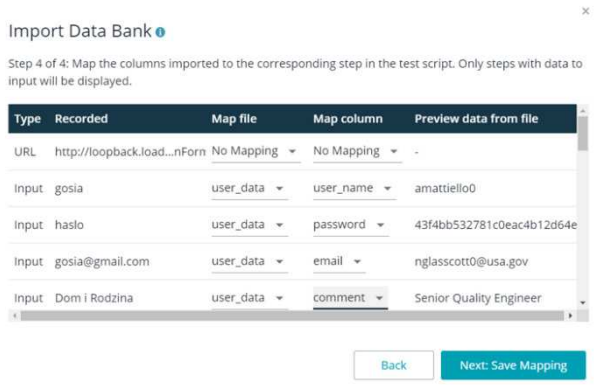
Utworzenie projektu i uruchomienie nagrywarki powoduje, że na ekranie wyświetlany jest podgląd testowanej aplikacji. Każde wysłanie formularza prezentowane jest jako osobny krok, na który składają się akcje użytkownika takie jak: bezczynność, kliknięcie pola tekstowego, przewijanie strony, użycie przycisku *backspace* lub wprowadzenie tekstu.

Narzędzie umożliwia parametryzację zapytań poprzez pliki płaskie CSV lub TSV. Podgląd (Rys. 3) oryginalnej wartości wprowadzonej podczas nagrywania scenariusza oraz przykładowa wartość z wybranej kolumny pliku płaskiego czyni parametryzację testu dosyć przystępną.

Podczas przeprowadzania bardziej skomplikowanych testów, brak możliwości tworzenia wartości wyliczeniowych parametrów może okazać się problematyczny.

Wersja próbna oferuje ograniczoną liczbę wizualizacji rezultatów testów, choć są nowoczesne i interaktywne (Rys. 4).

LoadNinja oferuje wizualizację statystyk czasowych pojedynczych kroków wyłącznie w formie tabelarycznej (Apache JMeter i Gatling umożliwiały wygenerowanie ich również w postaci diagramów graficznych) (Rys. 5).



Rysunek 3: Parametryzacja formularzy LoadNinja.

Gatling

Utworzenie testu Gatling możliwe jest przez napisanie własnego skryptu lub wykorzystanie nagrywarki, a następnie dokonanie modyfikacji (np. w celu parametryzacji zapytań).

Nagrywarka wymaga konfiguracji proxy, jednak prostszym rozwiązaniem jest zaimportowanie pliku HAR, wygenerowanego podczas nagrywania logów sieciowych (dostępne w zakładce *Network* w przeglądarce Chrome).

Skrypt może zostać sparametryzowany w dowolnej formie, nie ma konieczności korzystania z rozwiązań oferowanych przez bibliotekę, podnosi to jednak koszty czasowe realizacji testu. Przykładowo biblioteka oferuje możliwość zdefiniowania opóźnień pomiędzy wykonaniem kolejnych kroków, jednak wprowadzenie bardziej losowych wartości wymaga dodatkowej implementacji (Listing 1).

Listing 1: Funkcja generująca losowy czas z przedziału

```
def randomTimeInSeconds(from: Int, to: Int): Int = {
    val random = new Random()
    random.nextInt(to - from + 1) + from
}
val between30and40 = randomTimeInSeconds(from = 30, to = 40)
```

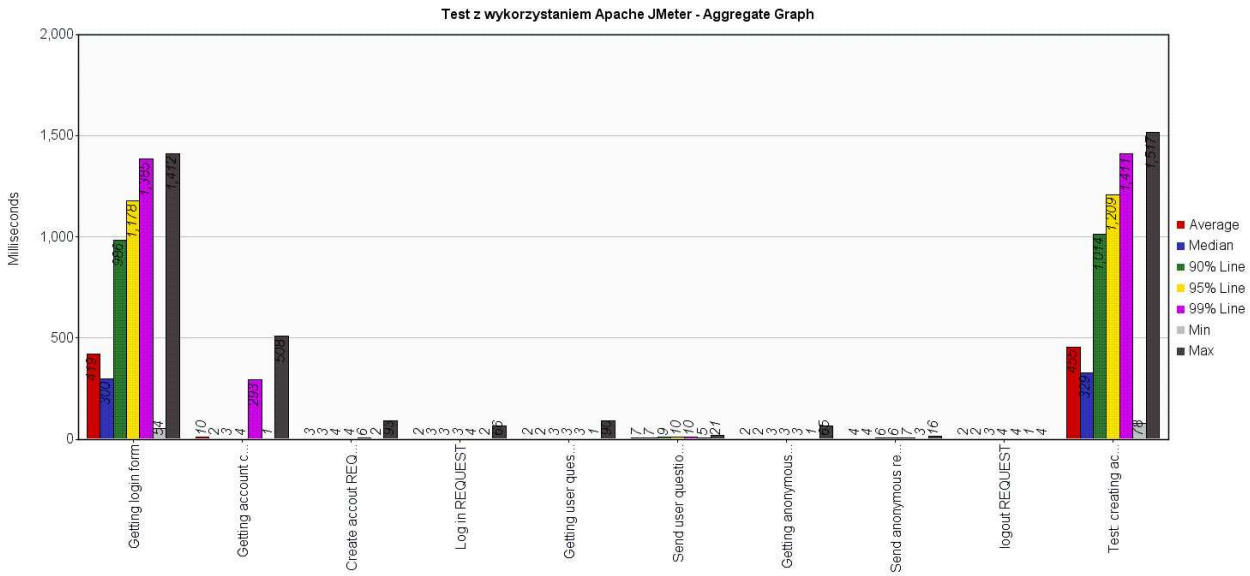
Podstawowym sposobem parametryzacji testu jest przekazywanie zmiennej uruchomieniowej programu. Jest to odpowiedni sposób do przekazywania podstawowych parametrów takich jak liczba wirtualnych użytkowników czy tryb uruchomienia testu. Komponent Feeders [15] (Listing 2) umożliwia parametryzację zapytań m.in. przez pliki płaskie, JSON lub źródło bazodanowe.

Na zakończenie testu generowane są nie tylko globalne statystyki, ale również te dotyczące poszczególnych kroków. Wizualizacje graficzne są nowoczesne i atrakcyjne (Rys. 6).

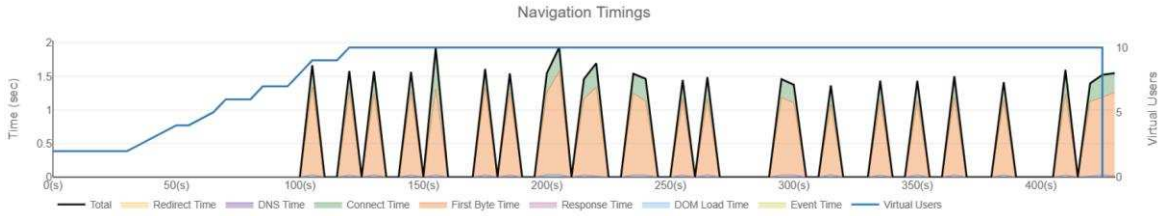
Brak graficznego interfejsu użytkownika jest największą wadą Gatling.

Listing 2: Parametryzacja scenariusza za pomocą Feeders.

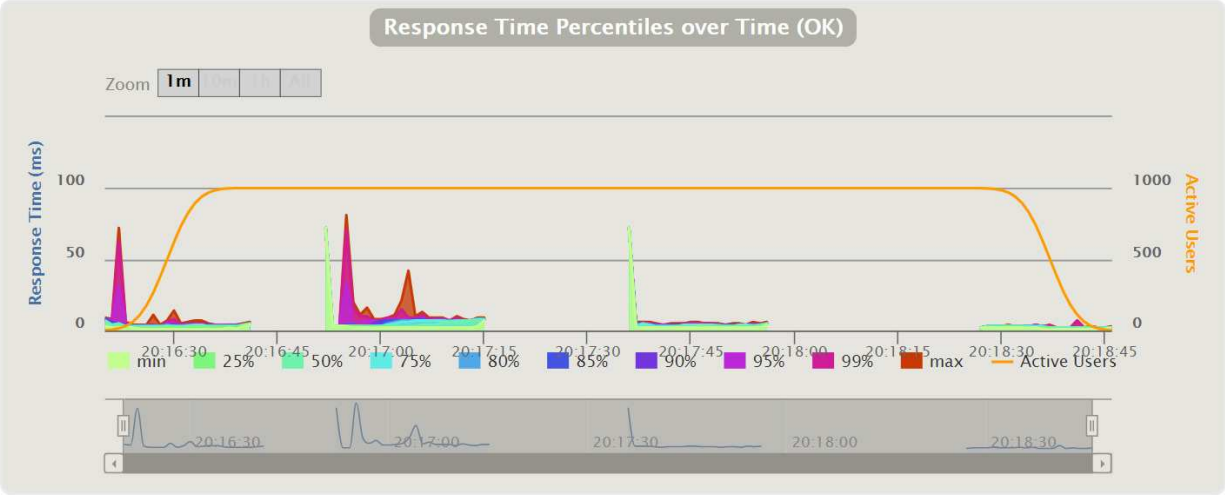
```
val usersCsv = csv(resourcePath + "user_data.csv")

val scn = scenario("GatlingTest")
    .feed(usersCsv)
    .exec(http("create_account_request")
              .post("/createAccountA")
              .headers(headers_1)
              .formParam("login", "${user_name}")
              .formParam("password", "${password}")
              .formParam("email", "${email}")
              .formParam("comment", "${comment}"))
    .pause(randomTimeInSeconds(20, 30))
```



Rysunek 4: Wynik wizualizacji Aggregate Graph eksperymentu Apache JMeter.



Rysunek 5: Wykres statystyk czasowych realizowanych żądań na przestrzeni czasu eksperymentu LoadNinja.



Rysunek 6: Liczba realizowanych żądań na przestrzeni czasu eksperymentu Gatling.

7. Wyniki

Wybrane narzędzia były znacząco różne, każde z nich wydaje się skierowane do grupy docelowej, o różnych oczekiwaniach. Przeprowadzone testy pozwoliły na opracowanie wyników, zestawionych w tabeli 1.

Tabela 1: Wyniki eksperymentu

Apache JMeter	LoadNinja	Gatling
Tworzenie nowych testów		
Trudne. Narzędzie posiada nagrywarkę, wymaga instalacji oraz importu certyfikatów.	Łatwe. Narzędzie posiada nagrywarkę, nie wymaga żadnej konfiguracji.	Trudne. Narzędzie posiada nagrywarkę, wymaga instalacji oraz importu certyfikatów.
Modyfikacja istniejących testów		
Łatwe. W przypadku podziału testu na fragmenty wymieniane są całe bloki testu (możliwe jest nagranie pojedynczego fragmentu).	Trudne. Możliwa jest tylko podstawowa modyfikacja fragmentów kroków np. zmiana wartości parametru w formularzu.	Umiarkowanie łatwe. Kod czytelny i samo-opisujący się dzięki DSL biblioteki pod warunkiem znajomości zmian np. REST API.
Inspekcja zmian (ang. *Code review*)		
Umiarkowanie trudne. Kod ma znaczne rozmiary, format JMX jest mało popularny.	Umiarkowanie łatwe. Brak kodu do weryfikacji, każda zmiana wymaga sprawdzenia spełnienia założeń biznesowych.	Umiarkowanie łatwe. Skrypt ma małe rozmiary, jest samo-opisujący się, pisany w języku obiektowym.
Wersjonowanie testu		
Wymaga systemu kontroli wersji np. Git.	Nie wymaga systemu kontroli wersji.	Wymaga systemu kontroli wersji np. Git.
Intuicyjność interfejsu		
Interfejs nie przyjazny użytkownikowi.	Interfejs przejrzysty i nowoczesny.	Brak interfejsu użytkownika.
Język pisania skryptów		
Groovy i Bash.	Brak.	Scala.
Możliwości parametryzacji		
Satysfakcjonujące. Możliwa jest parametryzacja adresu i treści zapytania poprzez m.in. pliki płaskie, wartości pobrane z bazy danych, zmienne wyliczeniowe poprzez gotowe komponenty.	Niesatysfakcjonujące. Narzędzie obsługuje tylko pliki płaskie, brak możliwości pobrania danych z bazy danych, brak możliwości wprowadzenia zmiennych wyliczeniowych.	Bardzo duże. Możliwa jest parametryzacja testu poprzez gotowe komponenty, które obsługują liczne formaty danych lub przygotowanie własnej implementacji zasilania danymi.
Metryki rezultatów		
Umiarkowanie satysfakcjonujące. Zawiera szereg gotowych komponentów generujących gotowe zestawienia. Wykresy graficzne są mało atrakcyjne wizualnie. Możliwy eksport wyników do wybranej formy. Przykładowymi metrykami są statystyki czasowe konkretnych żądań HTTP w formie tabelarycznej (View Results in Table) i graficznej (Aggregate Graph).	Umiarkowanie satysfakcjonujące. Wersja próbna zawiera ograniczoną liczbę metryk. Są dość atrakcyjne wizualnie. Dostępny eksport rezultatów do niskiej jakości pliku PDF. Przykładowymi metrykami są statystyki czasowe konkretnych żądań http w formie tabelarycznej (Statistics) i graficznej (Response Time Percentiles over Time) oraz statystyki zbiorcze (np. Number of responses per second).	Satysfakcjonujące. Wizualizacje są nowoczesne i interaktywne, generowane w formacie HTML. Statystyki dostępne również w formie pliku płaskiego, co umożliwia np. import wyników do bazy danych. Przykładowymi metrykami są statystyki czasowe konkretnych żądań HTTP wyłącznie w formie tabelarycznej (Durations and Errors).

8. Wnioski

Przeprowadzenie eksperymentu umożliwiło sformułowanie oceny każdego z badanych narzędzi.

Apache JMeter jest rozwiązaniem bardzo kompleksowym, co stanowi o jego dużej popularności. Posiada interfejs użytkownika, który w przeciwieństwie do opinii z pozycji [5], w niniejszym artykule określono jako nieatrakcyjny. Umożliwia pisanie własnych skryptów,

jednak jego złożoność oraz wymogi do konfiguracji stacji roboczej, pozwalają twierdzić, że przygotowanie testów jest łatwe dla programisty, ale bardzo trudne dla analityka biznesowego.

LoadNinja jest narzędziem bardzo prostym, chmurowym. Zaspokaja podstawowe potrzeby biznesowe. Zgodnie z opisem z pozycji [5], utworzenie testu jest bardzo szybkie, jednak nie da się tu utworzyć bardzo skomplikowanych testów, opartych o własne skrypty. Badania potwierdziły opinię przytoczoną w pozycji [1] - LoadNinja jest płatne, ale posiada bardzo atrakcyjny interfejs użytkownika, co wyróżnia go znacząco spośród pozostałych badanych narzędzi.

Gatling nie posiada interfejsu użytkownika, czym wyróżnia się spośród pozostałych analizowanych narzędzi. Przygotowanie w nim testów było najszybsze, dzięki DSL, którego rolę podkreślono w pozycji [8] - utworzony kod był bardzo czytelny, a potencjalne modyfikacje, które należałoby poddać ocenie przez innego dewelopera zrozumiałe. To darmowe rozwiązanie ma dodatkowo niezwykle rozwiniętą społeczność. Żadne z pozostałych narzędzi nie posiada takiego wachlarza rozszerzeń do wykorzystania bezpłatnie. Dla autorów nie było problemu z nieznajomością języka Scala (na co zwracano uwagę w pozycji [1]), gdyż składnia języka jest podobna do Javy.

Podsumowując:

- Apache JMeter najbardziej nadaje się dla dużych projektów, w których zachodzi konieczność wykorzystania najpopularniejszych, sprawdzonych rozwiązań posiadających bogatą dokumentację i wsparcie techniczne.
- Gatling można polecić nowym projektom, w których nacisk kładziony jest na dalsze rozwijanie czystego kodu, a przygotowanie testów wydajności leży w domenie zespołu programistów.
- LoadNinja oferuje przyjazny interfejs, godny polecenia dla np. analityków biznesowych. Rozwiązanie sprawdzi się w dużych, rentownych projektach, w których wymagania biznesowe są jasno zdefiniowane.

Literatura

[1] B. Erinle, Performance Testing with JMeter, Packt Publishing (2015) 7-14.

[2] A. Rodrigues, B. Demion, P. Mouawad, Master Apache JMeter - From Load Testing to DevOps: Master performance testing with JMeter, Packt Publishing (2019) 23-29.

[3] When Should I use Gatling vs. JMeter? A Tale of Two Tools, https://www.flood.io/blog/when-should-i-use-gatling-vs-jmeter-a-tale-of-two-tools, [29.07.2020].

[4] LoadNinja vs. JMeter: When to Use Each of Them, https://loadninja.com/resources/articles/performance-testing/loadninja-vs-jmeter-when-to-use-each-of-them/, [30.07.2020].

[5] 13 BEST Performance Testing Tools. Load Testing Tool (2020), https://www.guru99.com/performance-testing-tools.html, [23.04.2020].

[6] N. Hastings, Physical Asset Management, Springer (2010) 239.

[7] Performance Testing Tutorial: What is, Types, Metrics & Example. Common Performance Problems, https://www.guru99.com/performance-testing.html#3, [03.04.2020].

[8] D. Bryant, A. Marín-Pérez, Continuos Delivery in Java: Essentials Tools and Best Practises for Deploying Code to Production, O'Reilly Media (2018), 340-345.

[9] M. Bernardino, A.F. Zorzo, E. Rodrigues, FM de Oliveira, A Domain-Specific Language for Modeling Performance Testing (2014).

[10] L. Molyneaux, The Art of Application Performance Testing, O'Reilly Media (2009) 11-50.

[11] Apache JMeter Documentation, https://jmeter.apache.org/usermanual/, [05.04.2020].

[12] Getting started with JMeter – A basic Tutorial, https://www.blazemeter.com/blog/getting-started-jmeter-basic-tutorial, [05.04.2020].

[13] LoadNinja Documentation, https://support.smartbear.com/loadninja/docs/, [21.07.2020].

[14] Gatling Documentation, https://gatling.io/docs/current/, [03.05.2020].

[15] Gatling Feeders, https://gatling.io/docs/current/session/feeder, [03.05.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 358–364
Received: 19 September 2020
Accepted: 18 October 2020

Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework

Porównanie wydajności relacyjnych baz danych SQL Server, MySQL oraz PostgreSQL z zastosowaniem aplikacji webowej i frameworku Laravel

Rafał Wodyk*, Maria Skublewska-Paszkowska

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

Abstract

Many database implementations are supported by application frameworks that can affect their performance. The paper presents a comparison of the performance of SQL Server, MySQL and PostgreSQL relational databases based on an application written in PHP using the Laravel framework. The time of performance for various types of queries, both simple and using column and table concatenation was evaluated. The obtained results for the same database structures differed depending on the operations performed on the databases. Looking at the entirety of the research conducted, it can be concluded that in the case of databases in which the number of records is not too large (up to 1000 records) and the technical parameters of the device on which the database is running are of low or medium class, MySQL performs very well.

Keywords: framework Laravel; SQL Server; MySQL; PostgreSQL

Streszczenie

Wiele implementacji baz danych jest wspieranych przez szkielety aplikacji, które mogą rzutować na ich wydajność. Artykuł przedstawia porównanie wydajności relacyjnych baz danych SQL Server, MySQL oraz PostgreSQL na podstawie aplikacji napisanej w języku PHP z wykorzystaniem frameworku Laravel. Pod uwagę wzięty został czas wykonania różnego typu zapytań, zarówno prostych, jak również z użyciem konkatenacji kolumn oraz tabel. Otrzymane wyniki zostały poddane wielowymiarowej analizie. Otrzymywane wyniki dla takich samych struktur baz danych różniły się w zależności od wykonywanych operacji na bazach. Patrząc na całokształt przeprowadzonych badań można dojść do wniosku, że w przypadku baz danych, w których liczba rekordów nie jest zbyt duża (do 1000 rekordów) oraz parametry techniczne urządzenia, na którym uruchomiona jest baza są klasy niskiej bądź średniej bardzo dobrze wypada MySQL.

Słowa kluczowe: framework Laravel; SQL Server; MySQL; PostgreSQL

*Corresponding author

Email address: **rafal.wodyk@pollub.edu.pl** (R. Wodyk), **maria.paszkowska@pollub.pl** (M. Skublewska-Paszkowska)

1. Wstęp

W obecnych czasach, gdzie z roku na rok coraz więcej spraw można załatwić bez wychodzenia z domu, kluczową rolę odgrywają rozwiązania bazodanowe, z których korzysta znaczna większość aplikacji zarówno internetowych, mobilnych, czy desktopowych. Wykorzystywane są one zarówno do przechowywania danych, które użytkownicy mogą zaobserwować na ekranach swoich urządzeń, jak i do rejestrowania zdarzeń, magazynowania wyników złożonych analiz oraz badań rynku.

Podczas tworzenia różnego typu projektów informatycznych najczęściej wybór pada na relacyjne mechanizmy bazodanowe [1]. Niejednokrotnie obecne systemy informatyczne przechowują bardzo duże ilości danych, co zarazem wymusza na administratorach jak najlepszą optymalizację szybkości ich działania m.in. przez optymalizację zapytań. Dlatego też bardzo ważna jest przemyślana struktura bazy, która w przypadku relacyjnych baz danych może mieć duży wpływ na jej wydajność i szybkość wykonywania zapytań, szczególnie tych rozbudowanych.

W przeszłości wiele osób skupiało się na przeprowadzaniu badań wydajności poszczególnych rozwiązań bazodanowych, zarówno SQL jak i NoSQL.

W artykule [2] autorzy porównują wydajności systemów RDMBS o licencji open source PostgreSQL oraz MySQL z komercyjnym rozwiązaniem Microsoft SQL Server. Badania polegały na porównaniu średnich czasów wykonania takich operacji jak SELECT, INSERT, UPDATE oraz DELETE dla każdego silnika bazodanowego. Każda operacja wykonywana była 30-40 razy zarówno synchronicznie jak i asynchronicznie. Do badań wykorzystano system operacyjny Ubuntu oraz SQL Server 2012, MySQL 5.5.31 i PostgreSQL 9.1.

W zapisie danych, zarówno dla małej liczby rekordów 1000 jak i dużej 100000, najlepiej radził sobie PostgreSQL, natomiast najwolniejszy okazał się MySQL. W przypadku aktualizacji danych najszybszym systemem bazodanowym okazał się SQL Server, a następnie po nim MySQL. Dla operacji kasowania rekordów, podobnie jak dla operacji aktualizacji najbardziej wydajnym rozwiązaniem okazał się SQL Server, lecz w tym przypadku najsłabszy był MySQL. Dla odczytu

danych najszybszą bazą okazał się PostgreSQL, a następnie po nim uplasował się MySQL. Autorzy tego artykułu na podstawie przeprowadzonych badań zaobserwowali, że najbardziej wydajnym systemem RDMBS spośród badanych jest PostgreSQL.

W tym artykule [3] autorzy analizują wydajności baz danych takich jak: Microsoft SQL Server, PostgreSQL, SQLite oraz MySQL na systemach mobilnych. Do badań wykorzystane zostały systemy mobilne Android w wersji 5.0.1 oraz Windows Mobile oraz serwer PHP wykorzystany do wystawiania odpowiedniego REST API. Badaniom poddane zostały operacje zapisu oraz odczytu danych. Badania zostały przeprowadzone z wykorzystaniem liczby rekordów od 10 do 1000000. W przypadku badań operacji zapisu rekordów o różnych typach najbardziej wydajnym systemem bazodanowym dla systemu mobilnego Windows Mobile okazał się PostgreSQL, a nieznacznie gorszy był SQL Server. Natomiast w przypadku systemu Android najwydajniejszym rozwiązaniem był SQL Server. Biorąc pod uwagę operację odczytu danych z użycie systemu Windows Mobile najlepszą wydajność zaprezentował SQL Server, lecz nieznacznie gorszy od niego był w tym badanym przypadku MySQL. Na systemie Android wyjątkowo najszybszym rozwiązaniem okazał się MySQL. Autorzy analizowanego artykułu doszli do wniosku, że najszybszą formą wybierania oraz przechowywania danych jest SQL Server.

Analiza powyższych artykułów pokazała, że wyniki otrzymywane przez konkretne systemy bazodanowe są w dużej mierze zależne od technologii w jakiej powstała aplikacja oraz na jakim systemie operacyjnym działa. Niniejsza praca ma na celu przedstawienie wyników porównania wydajności najpopularniejszych relacyjnych baz danych takich jak MySQL, PostgreSQL oraz SQL Server. Są one wspierane przez obecnie najpopularniejszy szkielet aplikacji (ang. framework) dla języka PHP służący do tworzenia aplikacji internetowych, a mianowicie przez framework Laravel [4]. W dużej mierze dzięki swojej uniwersalnej składni interakcja z bazami danych jest niezwykle prosta, zarówno przy użyciu „surowego" języka SQL, konstruktora zapytań jak i Eloquent ORM (ang. object-relational mapping). Framework ten umożliwia generowanie takich samych struktur baz danych dla różnych wspieranych silników bazodanowych, co jest ułatwieniem zarówno w ewentualnej migracji z jednego rozwiązania na inne, jak i też do przeprowadzenia miarodajnych badań wydajnościowych. W niniejszym artykule otrzymane wyniki badań zostaną poddane wielowymiarowej analizie, dzięki czemu będzie można przekonać się, czy tak duża popularność silnika bazodanowego MySQL jest uzasadniona.

2. Relacyjne bazy danych

2.1. MS SQL

SQL Server jest to system do zarządzania relacyjnymi bazami danych, stworzony i licencjonowany przez firmę Microsoft [5]. Product SQL Server jest w większości przypadków rozwiązaniem płatnym wersje Standard oraz Enterprise. Istnieje również darmowa wersja Express do zastosowania dla małych aplikacji. Głównym wykorzystywanym językiem zapytań w tej bazie danych jest Transact-SQL, który pozwala na używanie takich rozwiązań jak instrukcje warunkowe, czy pętle [5,6].

Pierwsza wersja produktu została wydana w 1989 roku jako SQL Server 1.0. Rozwiązanie to przez cały czas było rozwijane, a obecnie najnowsza wersja opatrzona numerem 15 pochodzi z 2019 roku. Obecnie SQL Server dostępny jest na systemy: Windows oraz Linux, a także na maszyny wirtualne jak np. Docker i Azure. Samo środowisko SQL Server jest skalowalne i może składać się z różnych elementów w zależności od potrzeb konkretnej aplikacji [7].

2.2. MySQL

MySQL jest darmowym - na licencji open source, systemem do zarządzania relacyjnymi bazami danych. Należy on do pakietu oprogramowania LAMP (Linux, Apache, MySQL, PHP) [8]. MySQL powstał w 1995 roku, a obecnie rozwijany jest przez firmę Oracle. System ten cechuje się elastycznością oraz dużą wydajnością (potrafi obsłużyć do miliona zapytań na sekundę). Nie bez przyczyny z tego rozwiązania korzystają takie serwisy jak YouTube, czy Facebook [9].

Jednym z mechanizmów poprawiający znacząco szybkość wykonywania zapytań jest zapisywanie do pamięci podręcznej ostatnich zapytań wraz z ich wynikami. W znaczącej mierze zmniejsza to czas zwrócenia żądanego wyniku.

Baza danych MySQL wspierana jest z wykorzystaniem API przez popularne języki programowania takie jak: PHP, Java, C#, C++, a bazę danych można uruchomić na wszystkich popularnych systemach operacyjnych jak: Linux, Windows, macOS [10].

2.3. PostgreSQL

PostgreSQL jest obiektowym system do zarządzania bazami danych (ORDBMS) na licencji open source. Podobnie jak MySQL, wspiera najpopularniejsze systemy operacyjne, m.in. Windows, Linux, macOS [11]. Początki rozwoju PostgreSQL sięgają 1973 roku, a od roku 1996 za sprawą społeczności open source był rozwijany i popularyzowany. Obecnie z omawianego systemu do zarządzania bazami danych korzysta m.in.: Apple, Cisco, Instagram, Spotify, czy Skype [9].

Różnego typu aplikacje wykorzystujące bazę danych PostgreSQL mogą być tworzone z wykorzystaniem popularnych języków programowani jak np. Java, Python, Ruby, .NET czy PHP i framework Laravel [12].

PostgreSQL jest zgodny z ACID (ang. atomicity, consistency, isolation, durability), wspiera transakcyjność oraz zawiera wbudowaną obsługę zwykłych indeksów B-drzewa i skrótów. Posiada aktualizowalne i zmaterializowane widoki, wyzwalacze i klucze obce. Obsługuje również funkcje (również niektóre NoSQL) oraz procedury składowane [13].

3. Metoda badań

W badanym przypadku pod uwagę brane były czasy wykonania zapytania dla trzech silników bazodanowych, tj. SQL Server, PostgreSQL oraz MySQL, w zależności od liczby rekordów w tabeli z ogłoszeniami. Liczba rekordów w tej tabeli mieściła się w przedziale od 1 do 20000, w kolejnych eksperymentach. Za każdym razem zwracane były wszystkie rekordy aktualnie znajdujące się w tabeli.

Do testów wykorzystany został komputer o następującej specyfikacji:

- procesor Intel i5-5200U 2,2GHz;
- 8 GB RAM DDR 3 1600MHz;
- dysk SSD Goodram IRDM 256GB SATA 3;
- system operacyjny Windows 10 64 bit.

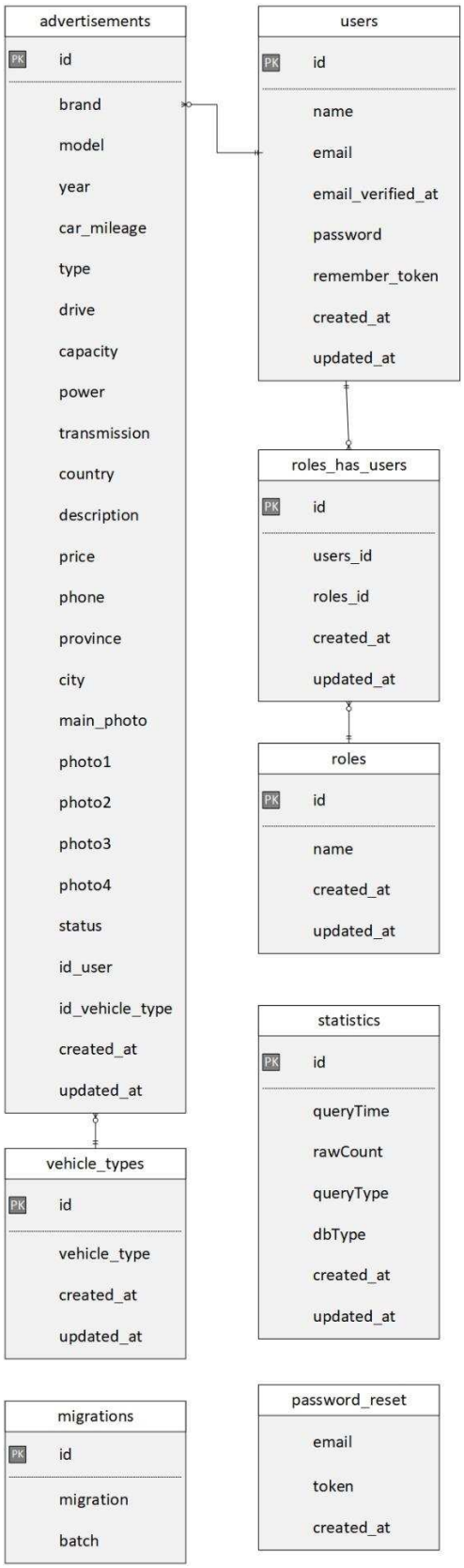
Testom poddane zostały następujące bazy danych:

- SQL Server 2017;
- PostgreSQL 12;
- MySQL 8.0.

Bazy danych dla poszczególnych silników zostały wygenerowane z wykorzystaniem mechanizmu migracji wpieranego przez framework Laravel. Proces migracji polegał na stworzeniu struktury bazy danych na podstawie modelu bazy napisanego w języku PHP. Model każdej tabeli składa się ze scharakteryzowania każdej kolumny poprzez podanie jest nazwy, typu danych oraz nadania ewentualnych indeksów w postaci klucza podstawowego i obcego. Wykorzystując ten mechanizm utworzono takie same struktury baz dla badanych silników bazodanowych, jak przedstawiono na rysunku 1.

W badaniach skupiono się na najpopularniejszych operacjach bazodanowych, które można spotkać w większości aplikacji internetowych korzystających z baz danych czyli m.in. SELECT, INSERT, DELETE. Po każdym badanym zapytaniu następowało czyszczenie pamięci podręcznej cache w celu uzyskania jak najbardziej miarodajnych wyników. Czas wykonania instrukcji na bazie był mierzony z poziomu kodu PHP obliczając różnicę czasu po i przed wykonania instrukcji. Wyniki przeprowadzonych eksperymentów zostały zebrane i przedstawione w postaci wykresów, żeby w jak najlepszym stopniu zobrazować wydajność badanych systemów baz danych dla poszczególnych operacji. Całość badań została przeprowadzona z wykorzystaniem aplikacji internetowej w postaci portalu z ogłoszeniami motoryzacyjnymi, przechowującej dane dotyczące ogłoszeń i użytkowników w bazie danych.

Konfiguracja połączenia z bazą danych w projektach napisanych z wykorzystaniem frameworka Laravel znajduje się w pliku .env. Domyślną wykorzystywaną nazwą bazy danych współpracującej z aplikacją internetową wykorzystaną do badań było „web_db". Jednym z najistotniejszych parametrów konfiguracyjnych baz danych wpływających na ich wydajność jest wielkość podstawowego bufora pamięciowego. Wykorzystywany jest on do buforowania danych podczas wykonywania operacji odczytu i zapisu na bazie danych. W przypadku wszystkich trzech badanych silników bazodanowych jego wielkość była ustawiona na 128MB.



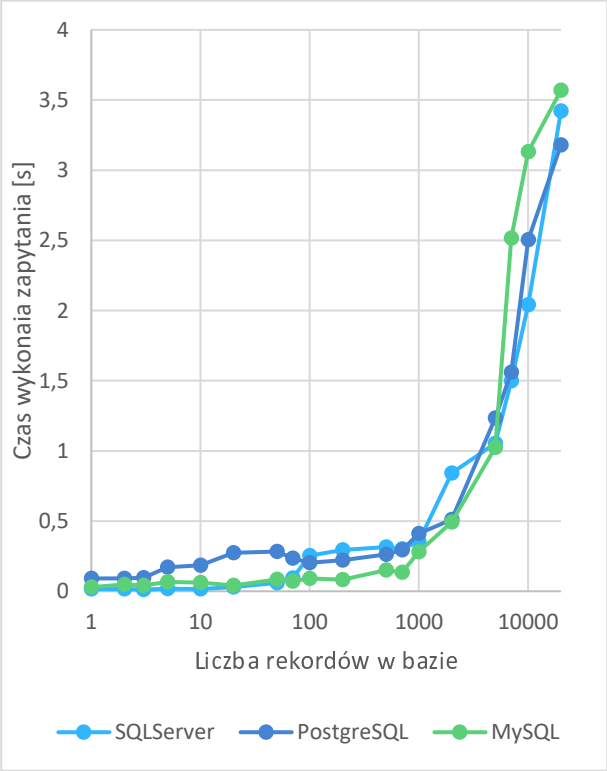
Rysunek 1: Schemat ERD bazy danych wykorzystywanej przez portal ogłoszeniowy

4. Wyniki badań

Najpopularniejszym typem zapytań bazodanowych jest instrukcja SELECT służąca do wybierania rekordów z tabel baz danych.

Na wykresie (Rys. 2) przedstawiono zebrane wyniki dla wszystkich badanych baz. W przypadku małej liczby rekordów (tj. do 1000 rekordów) w bazie danych, najwolniejszym rozwiązaniem okazywał się PostgreSQL, natomiast SQL Server i MySQL uzyskiwali czasy na zbliżonym poziomie. Większe dysproporcje można zauważyć w przypadku większej liczby rekordów w bazie (powyżej 5000), gdyż wtedy zdecydowanie najwolniejszym rozwiązaniem staje się popularny MySQL. Dla 20000 rekordów najszybsza okazała się baza danych PostgreSQL, będąc o prawie 0,2 s szybsza od SQL Server. W pewnym stopniu przewagę PostgreSQL można tłumaczyć tym, że SQL Server został stworzony z myślą o średnich i dużych aplikacjach, a co za tym idzie działania na sprzęcie bardziej wydajnym niż środowisko testowe użyte w tych badaniach.

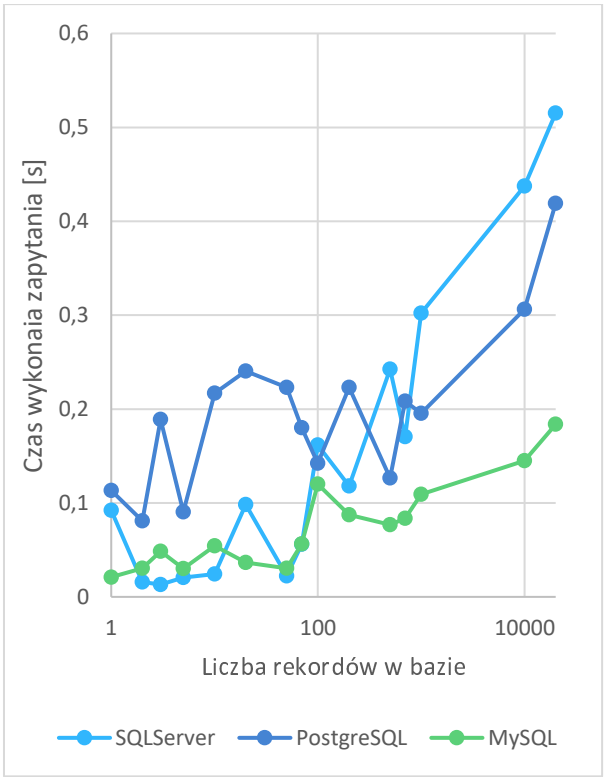
W przypadku portali ogłoszeniowych bardzo często użytkownicy dostają możliwość skorzystania z wyszukiwarki ograniczającej wyniki do określonych fraz. W tym przypadku często wykorzystywana jest konkatencja dwóch lub więcej kolumn, po których odbywa się wyszukiwanie. Dodatkowo niejednokrotnie wykorzystywane jest dodatkowe wyszukiwanie warunkowe z użyciem klauzuli WHERE, w celu dokładniejszego wyboru pasujących danych.



Rysunek 2: Czasy wykonania instrukcji Select w zależności od liczby zwróconych rekordów

W przypadku wykorzystywanej aplikacji również występuje wyszukiwarka, z wykorzystaniem której przeprowadzano badania. Zapytania również jak poprzednio, wykonywane były na tabeli advertisements. Zwracane wyniki były zawężane do marki i modelu pojazdu oraz województwa, z którego pochodzi ogłoszenie. Brane były pod uwagę czasy wykonania instrukcji do wyżej opisanego wybierania danych. Otrzymane wyniki zostały przedstawione na rysunku 3.

Analizując otrzymane wyniki (Rys. 3) można zauważyć zależność podobną jak, w przypadku wybierania danych. Dla małej liczby rekordów (tj. do 500 rekordów) czasy wykonywania zapytań były zbliżone, choć na podstawie wykresu można zauważyć, że to jednak silnik bazodanowy MySQL był najszybszy. Jego przewaga w szybkości wykonywania zapytań jest tym bardziej zauważalna w przypadku większej liczby rekordów w bazie danych, gdzie wyniki, które osiąga są lepsze o ok. 0,2s w porównaniu z PostgreSQL oraz o ok. 0.3s w porównaniu z SQL Server.



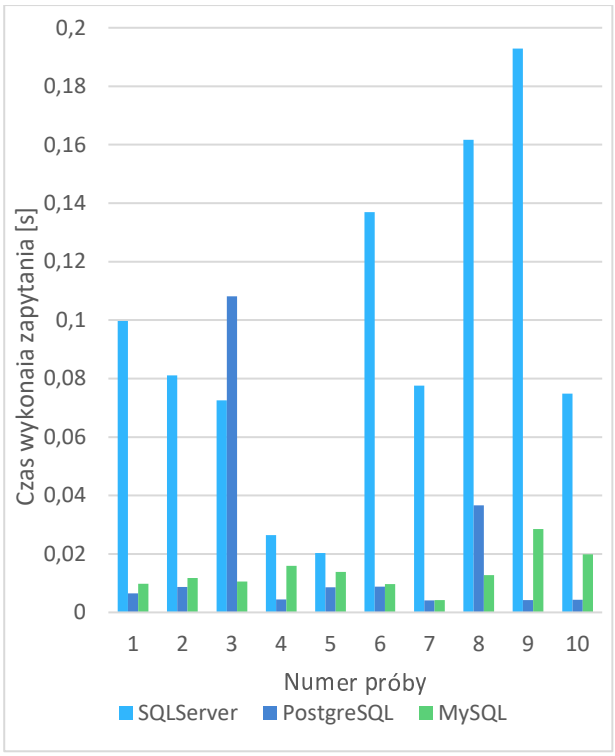
Rysunek 3: Czasy wykonania instrukcji Select w zależności od liczby rekordów w bazie

Kolejną z operacji poddanej badaniom było wstawianie nowych ogłoszeń motoryzacyjnych z poziomu interfejsu aplikacji. Badana operacja polegała na dodaniu jednego rekordu do tabeli advertisements oraz w przypadku braku danego typu pojazdu, jednego rekordu do tabeli vehicle_types. Dla każdego silnika bazodanowego przeprowadzono po 10 prób. Otrzymane wyniki zostały przedstawione na rysunku 4 i 5.

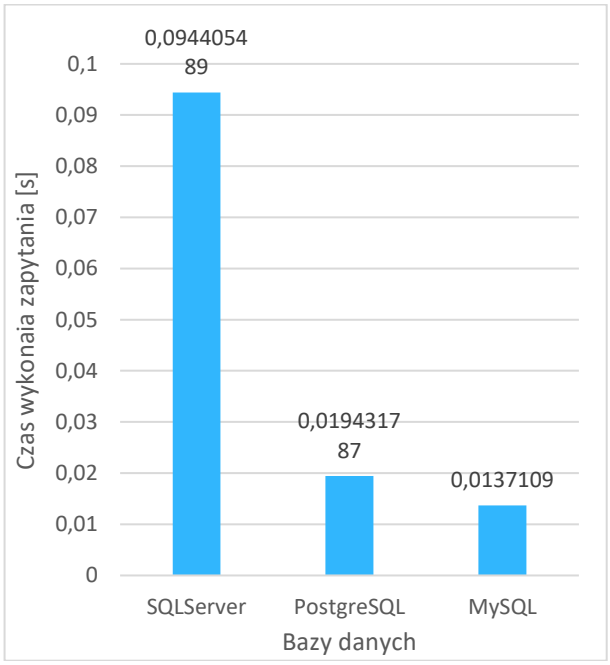
Na podstawie otrzymanych wyników można stwierdzić, że zdecydowanie najwolniejszym rozwiązaniem jest SQL Server, który jest średnio wolniejszy od swoich konkurentów o 0,08 s (Rys. 5). Jak duża jest to różnica świetnie prezentuje wykres (Rys. 4) na podstawie, którego od razu można zauważyć, że czasy SQL Server

są kilkukrotnie gorsze, niż pozostałych baz danych. Najszybszym rozwiązaniem w tym badanym przypadku okazał się PostgreSQL, który okazywał się nieznacznie szybszy od MySQL.

Następną operacją, na której się skupiono podczas badań, była aktualizacja danych związanych z ogłoszeniami motoryzacyjnymi. Edycji zostało poddanych po 10 (dla każdej bazy danych) losowo wybranych ogłoszeń znajdujących się w tabeli advertisements.

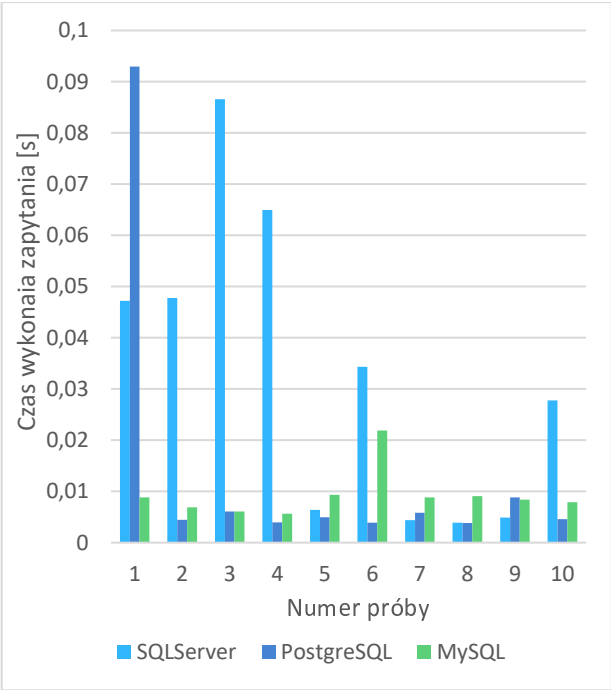


Rysunek 4: Czasy wykonania instrukcji Insert

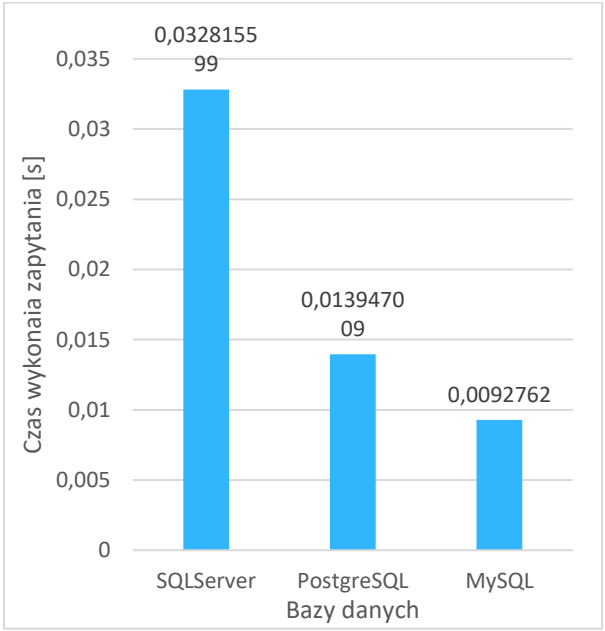


Rysunek 5: Średni czas wykonywania operacji dodawania danych

Porównując wyniki przestawione na rysunku 6, z wynikami dotyczącymi operacji wstawiania danych, można zauważyć podobieństwo w kwestii różnicy czasów osiąganych przez poszczególne systemy bazodanowe. Podobnie jak poprzednio, zdecydowanie najwolniejszym rozwiązaniem jest SQL Server, którego czasy zdecydowanie odstają od reszty co dobrze widać na rysunku 6. Również tak samo jak poprzednio, najszybszym rozwiązaniem okazał się PostgreSQL. Porównując średni czas na podstawie 10 prób można zauważyć, że różnica między SQL Server, a pozostałą dwójką jest mniejsza niż poprzednio i wynosi 0,02s (Rys. 7).



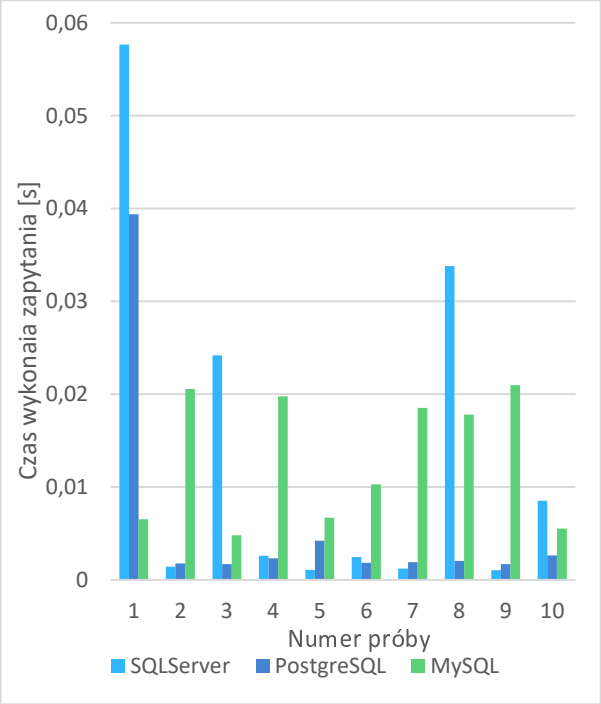
Rysunek 6: Czasy wykonania instrukcji Update



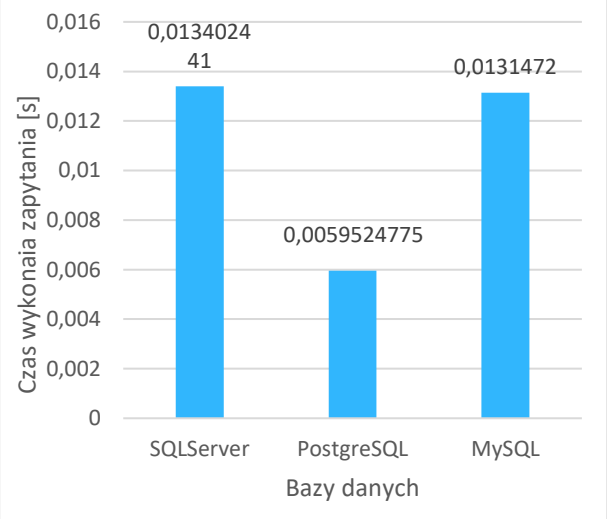
Rysunek 7: Średni czas wykonywania operacji aktualizacji danych

Ostatnią z operacji poddanych badaniom wydajności jest kasowanie danych. W tym przypadku pomiar czasu wykonania instrukcji DELETE był również oparty na danych znajdujących się w tabeli advertisements. Dokonano po 10 usunięć ogłoszeń motoryzacyjnych dla każdej z testowanych baz danych za pośrednictwem aplikacji internetowej.

Analizując otrzymane wyniki (Rys. 8) można zauważyć, że w tym przypadku najszybszym rozwiązaniem okazuje PostgreSQL. Jego średni czas wykonania operacji usunięcia jednego rekordu wynosi 0,00595s (Rys. 9) i jest lepszy od pozostałych dwóch rozwiązań o ok. 0,007s. SQL Server oraz MySQL w tym badaniu biorąc pod uwagę średnią arytmetyczną czasów na podstawie 10 prób osiągnęły bardzo zbliżone wyniki.



Rysunek 8: Czasy wykonania instrukcji Update



Rysunek 9: Średni czas wykonywania operacji kasowania danych

5. Wnioski

Otrzymywane wyniki dla takich samych struktur baz danych różniły się w zależności od wykonywanych operacji na bazach. Patrząc na całokształt przeprowadzonych badań można dojść do wniosku, że w przypadku baz danych, w których liczba rekordów nie jest zbyt duża (do 1000 rekordów) oraz parametry techniczne urządzenia, na którym uruchomiona jest baza są klasy niskiej bądź średniej bardzo dobrze wypada MySQL. Fakt ten potwierdzałby dużą popularność tego rozwiązania w przypadku małych konsumenckich aplikacji. Nieznacznie gorsze wyniki dla niezbyt dużej liczby danych (do 1000 rekordów) w bazie osiągał PostgreSQL, co daje mu drugie miejsce spośród trzech badanych rozwiązań.

Zaskoczeniem natomiast mogą być wyniki osiągnięte przez SQL Server, który w większości badanych operacji osiągał najgorsze czasy znacząco różniące się od konkurencji. Na plus natomiast można zapisać fakt, że w przypadku większej liczby rekordów w bazie danych (powyżej 1000 rekordów), jego czasy stawały się zbliżone do konkurencji. Może to potwierdzać założenie, że to rozwiązanie bazodanowe jest lepszym wyborem w przypadku średnich i dużych aplikacji, które operują na pokaźnych zasobach danych, a co za tym idzie wykorzystują urządzenia o wiele wydajniejsze niż testowe środowisko. Można się domyślać, że mając do dyspozycji wydajniejszy sprzęt SQL Server uzyskiwałby wyniki bardziej zbliżone do konkurencji.

Skupiając się natomiast na otrzymywanych wynikach w przypadku większej liczby rekordów (powyżej 1000) znajdujących się w bazie danych można zaobserwować, że MySQL traci pozycję lidera na rzecz PostgreSQL. Na podstawie tej obserwacji można stwierdzić, że najlepszym wyborem rozwiązania silnika bazy danych dla średniej bądź też dużej aplikacji jest właśnie PostgreSQL. Na jego korzyść z pewnością świadczy fakt, że jest oparty na licencji open source, a w dodatku jest cały czas wspierany i rozwijany, co z pewnością przemawia do sporej części twórców oprogramowania, żeby właśnie to rozwiązanie wybrać.

Literatura

[1] Relacyjne bazy danych – dlaczego warto je znać?, https://teamquest.pl/blog/461_relacyjne-bazy-danych-dlaczego-warto-je-znac, [15.09.2020]

[2] C. Truica, A. Boicea, Asynchronous Replication in Microsoft SQL Server, PostgreSQL and MySQL, International Conference on Cyber Science and Engineering, Guangzhou, 2013.

[3] M. Grudzień, K. Korgol, D. Gutek, Porównanie możliwości wykorzystania oraz analiza wydajności baz danych na systemach mobilnych, Journal of Computer Sciences Institute 2 (2016) 133-139.

[4] 8 Best PHP Frameworks in 2020, https://athemes.com/collections/best-php-frameworks/, [15.09.2020]

[5] B. Masood-Al-Farooq, SQL Server 2014 Development Essentials, Packt Publishing, Birmingham, 2014.

[6] SQL Server Tutorial, https://www.sqlservertutorial.net, [01.05.2020]

[7] Środowisko MS SQL Server, https://www.sqlpedia.pl/srodowisko-ms-sql-server/, [01.05.2020]

[8] MYSQL, https://vavatech.pl/technologie/bazy-danych/mysql, [26.04.2020]

[9] PostgreSQL vs. MySQL: Which One Is Better for Your Use Case, https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/, [26.04.2020]

[10] Supported Platforms: MySQL Database, https://www.mysql.com/support/supportedplatforms/database.html, [16.10.2020]

[11] S. Juba, A. Volkov, Learning PostgreSQL 11, Packt Publishing, Birmingham, 2019.

[12] S. Riggs, G. Ciolli: PostgreSQL 9 Administration Cookbook, Second Edition, Packt Publishing, Birmingham, 2015.

[13] PostgreSQL Vs. MySQL: Differences In Performance, Syntax, And Features, https://blog.panoply.io/postgresql-vs.-mysql, [26.04.2020]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 365–372
Received: 22 September 2020
Accepted: 5 October 2020

Comparative analysis of microcontrollers from the Arduino family and other compatible ones

Analiza porównawcza mikrokontrolerów z rodziny Arduino oraz innych kompatybilnych

Przemysław Suszek*, Tomasz Szymczyk

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

Abstract
This work concerns the design, creation and testing of popular microcontrollers from the Arduino family, and other selected ones, compatible with the Arduino Software (IDE). The purpose of the tests was to conduct a comparative analysis of selected devices in terms of various aspects of their operation. The following features were thoroughly analyzed: the speed of calculations, heat generation by the device, quality of the received Wi-Fi signal (for those devices that have this module), and the general evaluation of the microcontroller, resulting from selected features of its specifications.

Keywords: microcontrollers; AVR; Arduino; ESP8266

Streszczenie
Niniejsza praca dotyczy zaprojektowania, stworzenia oraz przeprowadzenia testów popularnych mikrokontrolerów z rodziny Arduino, oraz innych wybranych, kompatybilnych ze zintegrowanym środowiskiem programistycznym Arduino Software. Zadaniem przeprowadzonych testów było przeprowadzenie analizy porównawczej wybranych urządzeń pod kątem zbadania różnych, wybranych aspektów ich działania. Dokładnie przeanalizowano takie cechy jak:szybkość wykonywania obliczeń, wydzielanie ciepła przez urządzenie, jakość odbieranego sygnału Wi-Fi (dla tych spośród badanych urządzeń, które ten moduł posiadają), oraz ogólną ocenę mikrokontrolera, wynikającą z wybranych cech jego specyfikacji.

Słowa kluczowe: mikrokontrolery; AVR; Arduino; ESP8266

*Corresponding author
Email address: **przerek@wp.pl** (P. Suszek)

1. Wstęp

Elektronika jako dziedzina rozwija się już od ponad kilkudziesięciu lat. Mimo tego, tylko nieliczne z technologii oraz wynalazków stanowią kamienie milowe dorobku tej dziedziny. Wyparcie lamp elektronowych poprzez zastosowanie diod półprzewodnikowych oraz tranzystorów stało się początkiem wielkiego przeskoku technologicznego. Przeskok ten poskutkował miniaturyzacją, oraz zwiększeniem możliwości układów elektronicznych, w konsekwencji doprowadzając do wprowadzenia układów scalonych, które początkowo były analogowe, potem natomiast przyszła pora na cyfrowe układy scalone [1, 2]. Z układów cyfrowych wyłoniły się w końcu mikrokontrolery, stanowiące obecnie podstawę zdecydowanej większości urządzeń spotykanych w codziennym życiu i pracy [3].

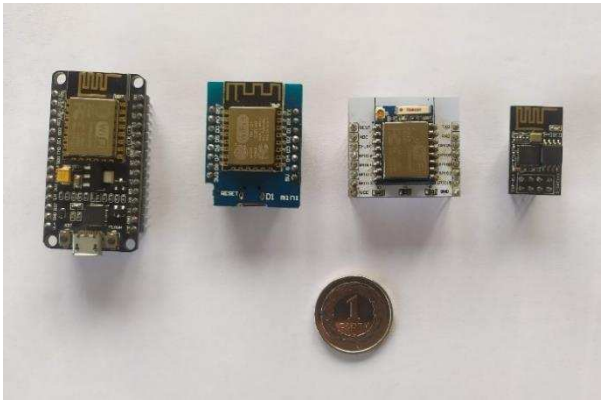
Arduino to płytka drukowana, mieszcząca się na dłoni, wyposażona w złącza typu goldpin. Sercem tej płytki jest właśnie mikrokontroler z rodziny AVR. Jednak Arduino to nie tylko płytka, lecz również odrębna dziedzina, jedynie bazująca na oryginalnych płytkach produkcji włoskiej, jak i niezliczonej ilości ich klonów. O bogactwie tej platformy świadczy między innymi rozległa oferta dedykowanych nakładek do Arduino, tzw. Arduino Shield, oraz prawidłowo opracowane środowisko programistyczne (Arduino IDE), pozwalające również po zaimportowaniu odpowiednich bibliotek, na programowanie płytek kompatybilnych z Arduino [4].

Szerokie rozpowszechnienie platformy Arduino w ostatnich latach doprowadziło w ostatnich latach do znacznego wzrostu zainteresowania programowanymi płytkami. Arduino, jak i jego liczne klony oraz inne mikrokontrolery pozwoliły wielu pasjonatom oraz entuzjastom takich dziedzin, jak informatyka, elektronika czy mechatronika na rozwinięcie swoich pasji oraz zainteresowań. W Internecie można znaleźć wiele kreatywnych projektów, które jako bazę wykorzystują właśnie mikrokontrolery, co jeszcze przydaje im popularności. Myślę, że uniwersalność oraz różnorodność zastosowań mikrokontrolerów sprawią, iż będą one jeszcze przez wiele lat uczyć i bawić kolejne pokolenia swoich entuzjastów [4, 5].

2. Badane urządzenia

Do przeprowadzenia badań zostały wytypowane wybrane rodzaje płytek z rodziny ESP8266, widoczne na poniższym rysunku (Rysunek 1).

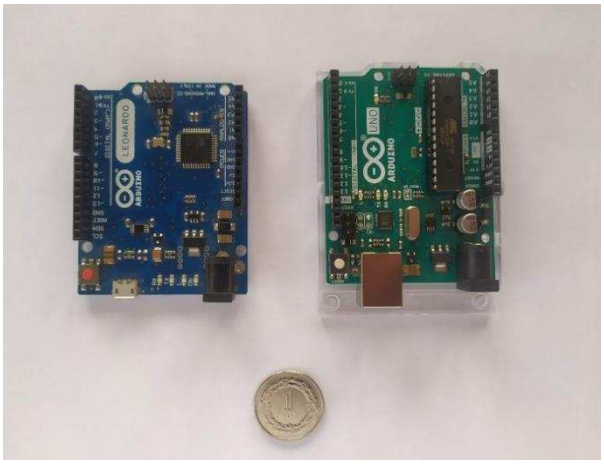
Każda z wymienionych płytek z rodziny ESP8266 wyposażona jest w mikrokontroler ATmega328.



Rysunek 1: Widok badanych mikrokontrolerów, od lewej: NodeMCU, Wemos D1 mini V2, ESP8266-01, ESP8266-07. Dla zobrazowania ich rzeczywistej wielkości – moneta 1 PLN

Na powyższym rysunku (Rys.1.) można dostrzec również, iż trzy z czterech płytek mają wbudowaną antenę w postaci nadrukowanej ścieżki PCB, natomiast jedna z nich posiada antenę ceramiczną. Dodatkowo, każde urządzenie jest znamionowo zasilane napięciem 5V [3, 6-8].

Do badań użyto również płytki z rodziny Arduino (Rysunek 2).



Rysunek 2: Widok badanych mikrokontrolerów, od lewej: Arduino Leonardo, Arduino UNO. Dla zobrazowania ich rzeczywistej wielkości - moneta 1PLN

Każda z przedstawionych płytek z rodziny Arduino posiada mikrokontroler z rodziny Atmega. W przypadku Arduino Leonardo jest to ATmega32u4, natomiast dla Arduino UNO jest to ATmega328. Każda z nich posiada wbudowany programator, dzięki czemu możliwe jest zaprogramowanie każdej z nich bez używania zewnętrznego programatora.

Dodatkowo warto zauważyć, iż obie płytki posiadają złącze zasilające 5V, dające możliwość połączenia z komputerem, w celu wgrania programu na mikrokontroler czy połączenia z monitorem portu szeregowego, dostępnym w Arduino IDE [3, 6-8].

3. Analiza porównawcza

3.1. Porównanie szybkości wykonywania obliczeń

Nie wszystkie z badanych tu urządzeń charakteryzują się jednakową szybkością działania. Wynika to pośrednio z częstotliwości taktowania. Zademonstrowano je w tabeli Tab.1

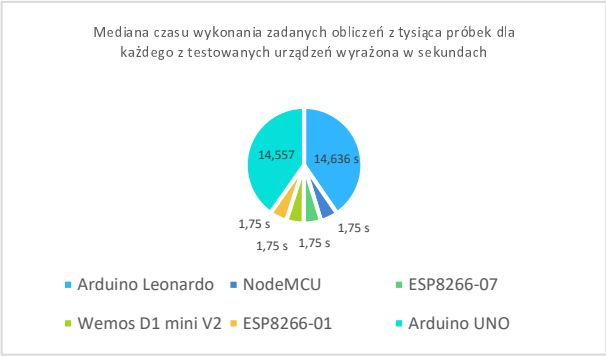
Tabela 1: Częstotliwości pracy poszczególnych procesorów zadeklarowane przez producenta urządzenia

	Arduino Leonardo	NodeMCU	ESP8266-07	Wemos D1 mini V2	ESP8266-01	Arduino UNO
Częstotliwość pracy procesora [MHz]	16	16	80	80	80	80

Do badania szybkości wykonywania obliczeń przez dany mikrokontroler posłużył autorski, prosty program napisany w środowisku Arduino. Zadaniem tego programu jest wielokrotne wykonywanie miliona przebiegów pętli, która wykonuje operacje potęgowania, a następnie zmierzenie czasu, w jakim wykonane zostały opisane wyżej operacje.

Dla każdego z testowanych mikrokontrolerów wykonano po tysiąc wykonań pętli, a co za tym idzie, zebrano po tysiącu wyników pomiarów czasu wykonania działań zawartych w wewnętrznej pętli programu, dla każdego testowanego urządzenia.

Pozyskane dane, będące wynikami badań Autora, zostały opracowane i przedstawione w formie wykresu kołowego (Rysunek 3) który prezentuje medianę czasu wykonywania obliczeń dla każdego z badanych urządzeń. Jako miarę zastosowano tu medianę, gdyż przeważająca większość otrzymanych wyników pomiaru dla danego urządzenia była jednakowa.



Rysunek 3: Mediana czasu wykonania zadanego zadania dla każdego

Dodatkowo, dane zaprezentowano również w tabeli (Tabela 2), w której zawarto dodatkowo informacje o wartości odchylenia standardowego jak i średniej arytmetycznej czasu wykonywania zadanych obliczeń przez dany mikrokontroler, na podstawie tysiąca prób przeprowadzonych na danym module.

Na podstawie Tabeli 2 można stwierdzić, iż najwolniejsze są mikrokontrolery Arduino Leonardo oraz Arduino UNO. Czasy te wynoszą kolejno 14,557 sekund oraz 14,636 sekund. Można zatem uznać, iż wzajemna klasa tych urządzeń pod względem szybkości wykonywania obliczeń jest bardzo zbliżona. Czas wykonywania zadanego podprogramu w przypadku Arduino Leonardo jest większy o zaledwie 0,54% w stosunku do czasu wykonywania tych samych działań na Arduino UNO.

Dodatkowo można zauważyć zależność między czasami osiągniętymi przez mikrokontrolery NodeMCU, ESP8266-07, Wemos D1 mini V2 oraz ESP8266-01. Mediana czasów osiąganych przez wyżej wymienione mikrokontrolery jest identyczna dla każdego z nich, i wynosi 1,75 sekundy.

Tabela 2: Wyniki pomiarów czasu wykonania obliczeń oraz wartości odchylenia standardowego, dla każdego badanego urządzenia

	Arduino Leonardo	NodeMCU	ESP8266-07	Wemos D1 mini V2	ESP8266-01	Arduino UNO
Mediana czasu wykonania obliczeń [s]	14,63	1,75	1,75	1,75	1,75	14,56
Średnia czasu wykonania obliczeń [s]	14,636028	1,750123123	1,750126873	1,75012987	1,750144144	14,55686
Odchylenie standardowe z tysiąca pomiarów	0,00019	0,0001561	0,00024475	0,0002987	0,000260137	0,000596992

Można zatem uznać, że szybkość działania mikrokontrolerów z rodziny ESP8266, analizowana na podstawie czasu wykonywania wcześniej przedstawionych działań, jest o 88,04% lepsza w porównaniu z Arduino Leonardo, oraz o 87,98% lepsza w porównaniu z Arduino UNO.

Dodatkowo, na podstawie informacji zawartych w tabeli (Tabela 1), można zauważyć, iż odchylenie standardowe dla każdego zestawu badanych próbek jest znikome – w żadnym z badanych przypadków nie przekracza ono tysięcznej części jedności.

3.2. Badanie mocy odbieranego sygnału

RSSI (ang. *Received Signal Strength Indicator*) jest wskaźnikiem, który określa moc odbieranego sygnału radiowego. Wartości wskazywane przez ten parametr informują o tym, w jakim stopniu urządzenie odbiorcze „słyszy" urządzenie nadawcze. Znajomość tego parametru pozwala określić, czy sygnał radiowy jest wystarczający do ustanowienia połączenia bezprzewodowego. Zależność siły sygnału i (autorskiej) klasyfikacji jakości tego sygnału przedstawia poniższa tabela [9, 10] (Tabela 3).

Tabela 3: Klasyfikacja poziomu sygnału do jego użyteczności dla transmisji danych

Siła sygnału [dBm]	Klasyfikacja
-30	Niesamowity
-67	Bardzo dobry
-70	Dobry
-80	Niedobry
-90	Nieużyteczny

Wartość siły odbieranego sygnału mierzalna jest w jednostce dBm. Jest to logarytmiczna jednostka mocy odniesiona do mocy 1mW [10].

Przeprowadzone badanie polegało na zmierzeniu parametru RSSI dla każdego z badanych mikrokontrolerów, które posiadają moduł Wi-Fi. Układ pomiarowy składał się z nadajnika sygnału radiowego w postaci telefonu typu smartphone, który miał uruchomioną funkcją hotspot, oraz odbiornika w postaci badanego mikrokontrolera. Pomiary przeprowadzono dla czterech spośród badanych urządzeń. Warto nadmienić, iż przeprowadzone pomiary zostały wykonane w dwóch różnych warunkach otoczenia. Wykonano dwie serie pomiarów. Pierwsza z nich została przeprowadzona w przestrzeni całkowicie otwartej, gdzie pomiędzy źródłem sygnału a jego odbiornikiem nie było żadnych stałych przeszkód. Jest to pomiar referencyjny możliwości komunikacji tych urządzeń. Druga z serii pomiarowych przeprowadzona została natomiast w przestrzeni, gdzie przeszkodę między urządzeniami transmitującymi sygnał radiowy stanowiła jednolita ściana wykonana z cegły o grubości 12 cm. Pomiary następowały w linii prostej między urządzeniami nadawczym i odbiorczym. Kolejne dane pomiarowe były pobierane z częstotliwością 10 odczytów pomiaru wartości parametru RSSI co jeden metr, na łącznej odległości czterdziestu trzech metrów. Pomiary te miały za zadanie pokazać właściwe możliwości użytkowe urządzeń.

Wyniki badanych parametrów zostały zaprezentowane na poniższym rysunku w formie graficznej, za pomocą wykresu (Rysunek 4). Na osi rzędnych wykresu zaznaczono wartości parametru RSSI wyrażone w dBm, natomiast na osi odciętych - odległość między urządzeniem odbiorczym oraz nadawczym.

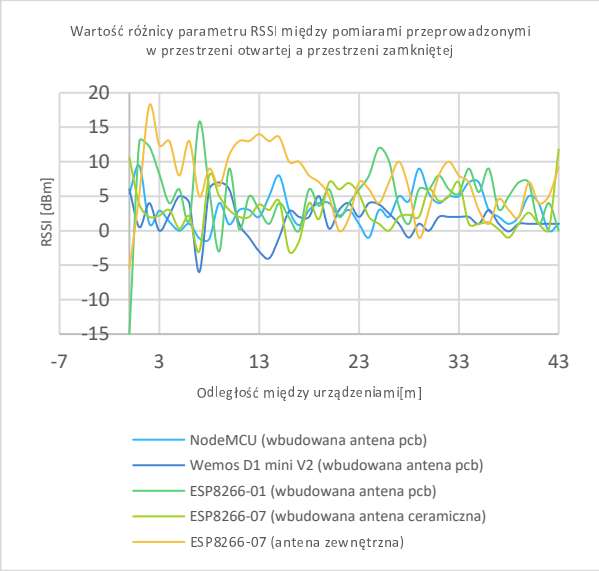
Rysunek 4: Zależność parametru RSSI od odległości między urządzeniami dla różnych konfiguracji płytek

Każde urządzenie oznaczono według schematu: nazwa urządzenia/przeszkoda/ typ anteny, gdzie nazwa urządzenia oznacza nazwę badanego mikrokontrolera, przeszkoda oznacza, czy badanie zostało przeprowadzone w przestrzeni otwartej (PO), czy w przestrzeni zamkniętej (PZ). Typ anteny oznacza typ anteny użytej do wykonania pomiaru: antena PCB nadrukowana na płytkę, antena ceramiczna, czy antena zewnętrzna.

Dokonując analizy spadków mocy sygnału odbieranego dla konkretnych układów pomiarowych, można zauważyć zależność pomiędzy spadkiem mocy sygnału w dwóch różnych warunkach otoczenia, w których zostały przeprowadzone pomiary. W przypadku każde-

go z badanych urządzeń można zauważyć, iż siła sygnału odbieranego była większa w przypadku badań przeprowadzanych w przestrzeni otwartej, niźli tych, które zostały przeprowadzone w przestrzeni zamkniętej.

Na poniższym wykresie (Rysunek 5) zaprezentowano względną wartość wyrażoną w dBm [9], o jaką lepsze były sygnały badanych urządzeń w przypadku pomiarów przeprowadzanych w przestrzeni otwartej, niż tych, które zostały przeprowadzone w przestrzeni zamkniętej.



Rysunek 5: Różnica RSSI między pomiarami przeprowadzonymi w przestrzeni otwartej oraz przestrzeni zamkniętej

Utworzono również tabelę (Tabela 3) przedstawiającą zależność, pomiędzy średnią różnicą sygnału, którego wartość została zmierzona w przestrzeni otwartej, a tym, którego wartość zmierzona została w przestrzeni zamkniętej. Dodatkowo, w tabeli zawarto wiersz, którego poszczególne kolumny odpowiadają wartościom odchylenia standardowego dla wszystkich badanych próbek badanego urządzenia.

Dodatkowo, w tabeli przedstawiono również wartości współczynnika zmienności - parametru używanego w statystyce, który określa miarę zróżnicowania cechy. Pozwala on ocenić siłę zróżnicowania danej zbiorowości statystycznej, wykazując siłę zmiennej, a także ocenia średnią arytmetyczną. Duża wartość tego współczynnika ukazuje silne zróżnicowanie danej cechy [4].

Tabela 4: Porównanie zmian jakości sygnału w różnych warunkach dla różnych urządzeń

	NodeMCU	Wemos D1 mini V2	ESP8266-01	ESP8266-07 (antena wbudowana)	ESP8266-07 (antena zewnętrzna)
Średnia wartość wzrostu jakości sygnału [dBm]	3,17	1,75	4,93	2,93	7,23
Odchylenie standardowe	2,5	2,57	4,84	3,08	4,6
Współczynnik zmienności [%]	79	147	98	105	64

Na podstawie powyższej tabeli (Tabela 3) można stwierdzić, iż najlepszy przyrost jakości odbieranego sygnału Wi-Fi, będącego różnicą pomiędzy średnią różnicą sygnału, którego wartość została zmierzona w przestrzeni otwartej, a tym, którego wartość zmierzona została w przestrzeni zamkniętej, uzyskano w przypadku modułu ESP8266-07 oraz zastosowania anteny zewnętrznej do tego modułu. Wartość współczynnika zmienności dla wyżej wymienionego mikrokontrolera jest najmniejsza spośród prezentowanych wartości, i wynosi 64, co świadczy o najlepszym rozkładzie ba

danej cechy spośród badanych tu urządzeń. Dodatkowo warto wspomnieć, iż średnia wartość przyrostu jakości sygnału jest również najlepsza (największa) dla tego modułu. Podobnie sytuacja ma się również z odchyleniem standardowym.

Natomiast najmniej stabilnym przyrostem jakości sygnału charakteryzuje się natomiast NodeMCU, którego współczynnik zmienności wyniósł 79. Zauważono również, iż modułem, którego średnia wartość wzrostu jakości sygnału okazała się najmniejsza, było urządzenie Wemos D1 mini V2.

3.3. Pomiar temperatury

Każde urządzenie elektryczne wydziela ciepło. Dotyczy to także niewielkich, mikroprocesorowych płytek. Jest ono najczęściej niepożądanym efektem ubocznym w bardzo wielu przypadkach – zbyt duże nagromadzenie ciepła potrafi wpłynąć negatywnie na szybkość działania samego układu, lub w ekstremalnych sytuacjach trwale uszkodzić taki układ elektroniczny. Dlatego też zbadano to, jak mocno nagrzewają się poszczególne układy [1, 3, 6].

Badanie zostało przeprowadzone w celu poznania zmian temperatury zachodzących w okolicach badanych płytek. Z uwagi na fakt, iż badania były przeprowadzane przez wiele godzin w pomieszczeniu zamkniętym, to na temperaturę układu elektronicznego miała także wpływ temperatura otoczenia. W związku z tym, na wykresach przedstawionych w dalszej części pracy uwzględniono także zmiany temperatury otoczenia, wynikające z naturalnych zmian temperatury o różnych porach dnia. Znając temperaturę otoczenia (odniesienia), możliwe było określenie zmian temperatury zachodzących w bezpośrednim otoczeniu badanych urządzeń, gdyż zmiany te były określane względem temperatury odniesienia.

Do wykonania serii pomiarów temperatury każdego z badanych mikrokontrolerów wykorzystano czujniki temperatury DS18B20. Są to czujniki przede wszystkim dobrze skalibrowane, tanie, oraz cieszące się ogromną popularnością [8].

W celu zaobserwowania zmian temperatury zachodzących w bezpośrednich okolicach badanych mikrokontrolerów, w odniesieniu do temperatury otoczenia, został utworzony układ pomiarowy o wymiarach 105 mm x 70 mm x 30 mm Układ składa się z ze szczelnego pojemnika, do którego dołączone są czujniki temperatury. Czujniki są rozmieszczone w strategicznych miejscach pojemnika w następujących ilościach: pięć czuj-

ników wewnątrz komory pomiarowej przyrządu pomiarowego, oraz pięć czujników temperatury na zewnątrz przyrządu pomiarowego. Czujniki wewnątrz przyrządu pomiarowego odpowiadają za odczyt temperatury bezpośrednio w okolicy badanej płytki. Czujniki na zewnątrz przyrządu pomiarowego odpowiadają za odczyt temperatury otoczenia, po to, aby możliwe było między innymi zbadanie zależności zachodzących pomiędzy tymi dwiema wymienionymi temperaturami, a w konsekwencji wyznaczenie względnej temperatury generowanej przez mikrokontroler.

Program użyty do odczytu kolejnych pomiarów bazuje na programie służącym do pomiaru szybkości działania procesorów. Kluczową modyfikacją w stosunku do tamtego programu, jest zapis temperatury z czujników temperatury, który wykonywany jest po wykonani zawartości pętli obciążającej. Dodatkowo warto wspomnieć, iż pomiary temperatury dla każdego z badanych urządzeń były prowadzone przez okres sześciu godzin. Ze względu na skończoną dokładność czujników temperatury, wynoszącą ± 0.5 °C, wprowadzono pojęcie wartości poprawnej. Jeżeli jako wartość prawdziwą mierzonej wartości wprowadzi się pojęcie wartości poprawnej, czyli tej wyznaczonej lub obliczonej w najdokładniejszy możliwy sposób, wówczas można założyć, iż wyznaczenie wartości średniej jest takim sposobem na uzyskanie wartości poprawnej [5].

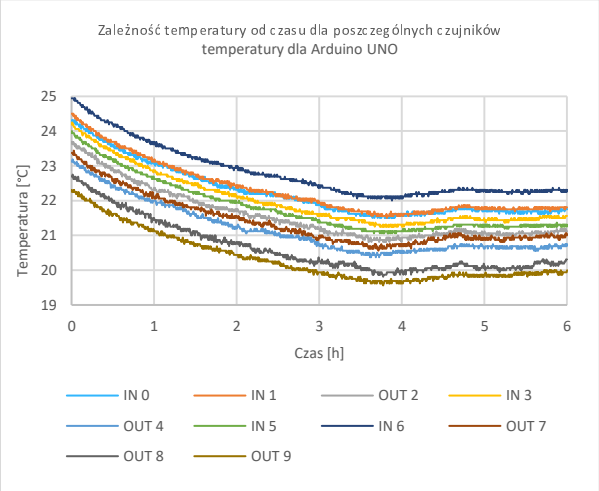
Dla każdego z badanych mikrokontrolerów wyznaczono wartość poprawną temperatury panującej wewnątrz urządzenia pomiarowego (na podstawie pięciu czujników temperatury wewnątrz urządzenia pomiarowego), jak i wartość poprawną temperatury panującej na zewnątrz urządzenia pomiarowego (na podstawie pięciu czujników temperatury na zewnątrz urządzenia pomiarowego). Wyznaczono również wartości odchylenia standardowego dla wartości poprawnej temperatury wewnętrznej, jak również zewnętrznej (temperatura odniesienia).

Wyznaczono wreszcie względną temperaturę generowaną przez badany mikrokontroler, za którą przyjęto różnicę między wartością poprawną temperatury wewnętrznej, a wartością prawdziwą temperatury zewnętrznej (Wzór 1).

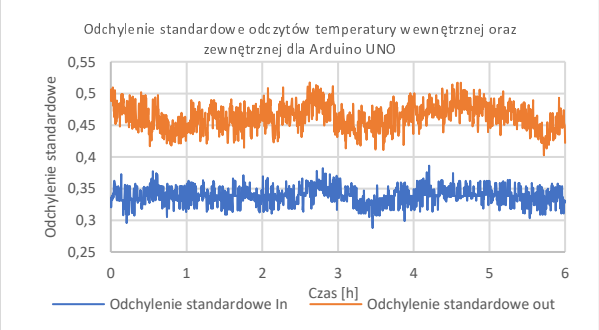
$$T_w = T_1 - T_0 \tag{1}$$

gdzie:
T_w - temperatura względna
T_1 - temperatura poprawna zmierzona w pobliżu urządzenia
T_0 -temperatura poprawna otoczenia

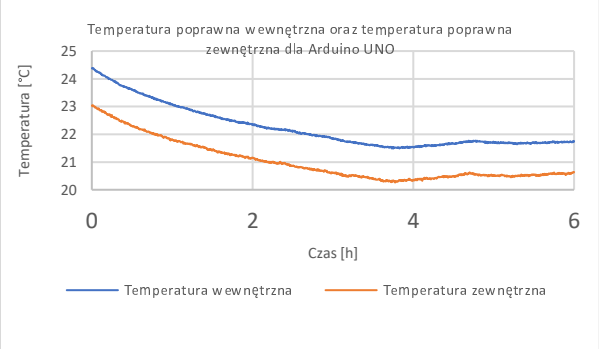
Przykładowe wyniki pomiarów, dla mikrokontrolera Arduino UNO, zostały przedstawione na poniższych rysunkach (Rysunki 6-9).



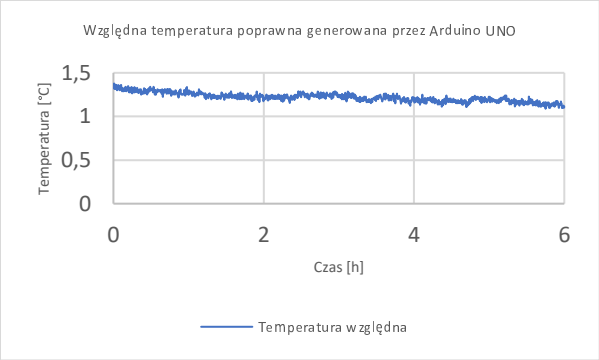
Rysunek 6: Temperatura zmierzona w okolicach badanego mikrokontrolera



Rysunek 7: Wartości odchylenia standardowego dla badanych temperatur

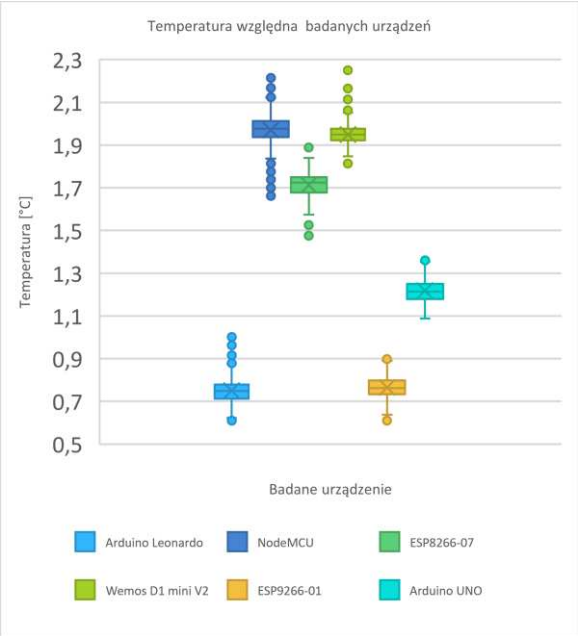


Rysunek 8: Zależność pomiędzy temperaturą wewnętrzną a zewnętrzną badanego mikrokontrolera



Rysunek 9: Względna temperatura generowana przez dany mikrokontroler

Na kolejnym rysunku (Rysunek 10) również zbiorcze zestawienie temperatury względnej, generowanych przez każde z testowanych urządzeń.



Rysunek 10: Porównanie zbiorcze temperatury generowanej przez badane urządzenia

Na powyższym rysunku można zauważyć trzy zasadnicze grupy generowanych temperatur. Grupę, która generowała największą temperaturę tworzą urządzenia NodeMCU, ESP9266-07, oraz Wemos D1 mini V2. Generowały one średnio kolejne temperatury: 1.97 °C, 1.71 °C, 1,95 °C. Kolejny, jednoelementowy podzbiór stanowi Arduino UNO, które generowało subiektywnie średnią temperaturę, której średnia wyniosła 1.22 °C. Zbiór, który stanowią urządzenia, które zostały zaliczone do urządzeń o najmniejszej wydzielonej temperaturze, stanowią urządzenia Arduino Leonardo oraz ESP8266-01. Wydzielają one kolejno średnie temperatury o wartości 0.75 °C oraz 0.76 °C.

Na podstawie otrzymanych wyników można zauważyć, iż żadna z temperatur generowanych przez badane urządzenia nie jest znaczna, gdyż największa z nich nie przekracza 2 °C. Bazując na dokumentacji [11-15] dołączonej do każdego z badanych mikrokontrolerów, która wskazuje górny zakres temperatury na 85 °C dla płytek z rodziny Arduino, oraz 125 °C dla płytek z rodziny ESP8266, oraz na otrzymanych wynikach zmian temperatury, można wysnuć wniosek, iż po stosunkowo długim czasie użytkowania badanych mikrokontrolerów, temperatura ich pracy nie zbliżyła się w znacznym stopniu do maksymalnej temperatury pracy podanej w dokumentacji. Dzięki temu, badane mikrokontrolery mogą być z łatwością i bez żadnych obaw stosowane w miejscach, gdzie temperatura odgrywa kluczową rolę, bez obaw, iż ich zastosowanie w znacznym stopniu podniesie temperaturę sąsiadujących z nimi elementów. Dodatkowo, można przypuszczać, iż urządzenia takie, z uwagi na niewielkie generowanie ciepło, mogą działać nieprzerwanie w szczelnie zamkniętej obudowie przez długi czas.

3.4. Analiza badanych urządzeń pod względem budowy, funkcji i potencjalnych zastosowań

Wszystkie informacje na temat parametrów mikrokontrolerów w tym podrozdziale, zostały zaczerpnięte z dokumentacji danych mikrokontrolerów oraz z zasobów Internetu [11-15]. Na podstawie określonych wartości danych parametrów, każdemu z badanych mikrokontrolerowi została przyznana pewna liczba punktów, w zakresie od zera do stu. Liczba punktów informuje o tym, jak dobrze radzi sobie dany mikrokontroler w badanym zakresie możliwości – im większa liczba punktów, tym lepszymi cechami odznacza się mikrokontroler. Warto nadmienić, iż w przypadku wartości liczbowych, gdzie ich wzrost jest proporcjonalny do poprawy parametrów urządzenia, punkty zostały przyznane w taki sposób, aby 100% możliwych do uzyskania punktów stanowiło najlepsze wskazanie wartości danego mikrokontrolera analizowanego w danej dziedzinie, przy czym kolejne(gorsze) badane urządzenia w danej dziedzinie otrzymują punkty względem przytoczonych stu punktów, które dostał najlepszy z nich (wzór 2).

$$liczba\ punktów = \frac{f(i)}{\max(i = 1 \dots n)} * 100 \qquad (2)$$

gdzie:
i – kolejne urządzenie
f(i) – wartość liczbowa cechy danego urządzenia
max(i=1…n) – największa wartość liczbowa spośród wszystkich urządzeń dla danej cechy

Dodatkowo, do każdej badanej cechy przypisana została waga, która została dobrana przez Autora tak, aby jak najlepiej odzwierciedla to, jak istotna jest badana cecha w ujęciu całościowym.

- *Rozmiar mikrokontrolera*

Rozmiary badanych mikrokontrolerów (wartości zostały przybliżone do pełnych milimetrów) zostały przedstawione w poniższej tabeli (Tabela 4). Wszystkie wymiary zostały zapisane w milimetrach. Warto nadmienić, iż ostatni wiersz tabeli zawiera dodatkową informację, w postaci sumy długości wszystkich boków danego mikrokontrolera, będącej jednocześnie obiektywnym wyznacznikiem służącym do porównania względnego rozmiaru badanych mikrokontrolerów.

Tabela 5: Rozmiary mikrokontrolerów

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Wymiary [mm]	65x53x10	65x53x10	14x25x3	16x21x3	35x25x12	47x25x12
Suma długości wymiarów [mm]	128	128	42	40	72	84
Punkty	31,25	31,25	98,44	100	75	65,63

- *Liczba wyprowadzonych pinów GPIO*

Liczba wyprowadzeń pinów GPIO (ang. *general-purpose input/output*) w każdym z badanych mikrokontrolerów test różna. Im większą liczbą pinów GPIO dysponuje mikrokontroler, tym większa jest liczba potencjalnych jego zastosowań dla układów wymagających wielu wyjść mikrokontrolera [3, 6]. Dokładną liczbę przedstawia poniższa tabela (Tabela 5).

Tabela 6: Liczba wyprowadzonych pinów GPIO

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Liczba wyprowadzonych pinów GPIO	20	20	2	11	11	13
Punkty	100	100	10	55	55	65

- *Częstotliwość taktowania mikroprocesora*

Częstotliwość taktowania każdego z badanych urządzeń, wraz z liczbą przyznanych punktów przedstawia poniższa tabela (Tabela 6). Różnica między najmniejszą a największą wartością nie jest stosunkowo duża, gdyż wynosi 64 MHz.

Tabela 7: Częstotliwość taktowania mikroprocesora

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Częstotliwość taktowania mikrokontrolera [MHz]	16	16	80	80	80	80
Punkty	20	20	100	100	100	100

- *Pamięć flash*

Przeanalizowano również rozmiar pamięci flash, której rozmiar dla poszczególnych mikrokontrolerów, wraz z liczbą przyznanych im z tego względu punktów przedstawia poniższa tabela (Tabela 7)

Tabela 8: Rozmiar pamięci flash

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Pamięć flash [kB]	32	32	128	128	4096	4096
Punkty	0,78	0,78	3,13	3,13	100	100

- *Pamięci RAM*

Rozmiar pamięci o swobodnym dostępnie dla każdego z badanych mikrokontrolerów, wraz z przyznanymi im punktami, zaprezentowano na poniższej tabeli (Tabela 8).

Tabela 9: Rozmiar pamięci RAM

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Pamięć RAM [kB]	2,5	2	36	36	64	64
Punkty	3,91	3,13	56,25	56,25	100	100

- *Komunikacja bezprzewodowa Wi-Fi*

Istotną cechą wpływającą na znaczne zwiększenie liczby zastosowań danego mikrokontrolera, jest obecność modułu Wi-Fi. Obecność (lub jej brak), dla każdego z badanych urządzeń przedstawiono na poniższej tabeli (Tabela 9). Przedstawiono również liczbę przyznanych punktów.

Tabela 10: Obecność modułu Wi-Fi

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Obecność modułu Wi-Fi	Nie	Nie	Tak	Tak	Tak	Tak
Punkty	0	0	100	100	100	100

- *Wbudowany programator oraz micro USB*

Istotną cechą przy wyborze mikrokontrolera jest prostota jego użytkowania. Aby zaprogramować taki mikrokontroler, należy użyć dedykowanego programatora. Na szczęście niektóre z mikrokontrolerów mają już taki programator wbudowany. Obecność (lub jej brak), dla każdego z testowanych urządzeń przedstawiono na poniższej tabeli (Tabela 10). Przedstawiono również liczbę przyznanych punktów.

Tabela 11: Obecność złącza micro USB oraz wbudowanego programatora

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU
Obecność wbudowanego programatora oraz złącza micro usb	Tak	Tak	Nie	Nie	Tak	Tak
Punkty	100	100	0	0	100	100

Jak zostało opisane w podrozdziale 4.4.1, każdemu badanemu mikrokontrolerowi przyznano pewną liczbę punktów, które odzwierciedlają sprawność mikrokontrolera w danej dziedzinie. W tym rozdziale przedstawiono zbiorcze porównanie badanych mikrokontrolerów, wraz z sumą punktów, jakie uzyskał badany mikrokontroler. Warto nadmienić, iż dla każdej z badanych kategorii została przypisana odpowiednia waga, z zakresu 1-5, odzwierciedlająca „ważność" badanej cechy na tle wszystkich badanych cech w tym rozdziale, dla danego urządzenia. Wyniki analizy w postaci punktów cząstkowych, jak i ostatecznej sumy punktów, jaką zdobył w rankingu dany mikrokontroler, wynikają z sumy iloczynów punktów cząstkowych i wagi, dla konkretnego mikrokontrolera przedstawia poniższa tabela (Tabela 10).

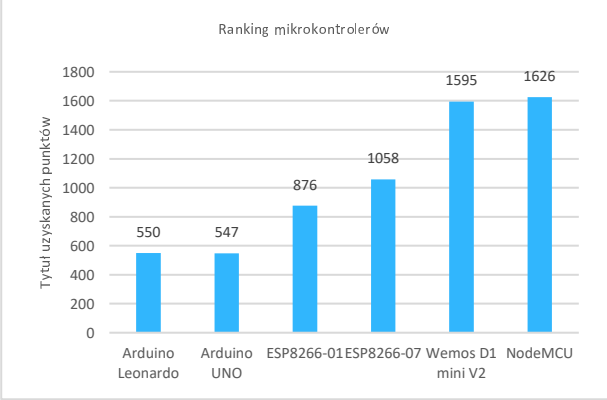
Na podstawie rysunku 11, prezentującego zależność użyteczności badanych mikrokontrolerów pod względem badanych w tym rozdziale cech, można zauważyć trzy wyróżniające się grupy, które odpowiadają sumarycznej ocenie badanych cech przedstawionych w tym podrozdziale. W pierwszej (najlepszej) grupie, na pierwszym miejscu, z najwyższą liczbą punktów znalazł się mikrokontroler NodeMCU, który zebrał blisko 1625 punktów. Bardzo bliski wynik, dający mu drugie miej-

sce, uzyskała płytka Wemos D1 mini V2, której liczba punktów to 1595.

Tabela 12: Analiza sumy punktów uzyskanych przez dany mikrokontroler

	Arduino Leonardo	Arduino UNO	ESP8266-01	ESP8266-07	Wemos D1 mini V2	NodeMCU	Waga
Suma długości wymiarów	31,25	31,25	98,44	100	75	65,63	1
Liczba wyprowadzonych pinów GPIO	100	100	10	55	55	65	4
Częstotliwość taktowania mikrokontrolera	20	20	100	100	100	100	5
Pamięć flash	0,78	0,78	3,13	3,13	100	100	4
Pamięć RAM	3,91	3,125	56,25	56,25	100	100	4
Suma uzyskanych punktów	550,1	547,88	876	1057,52	1595	1625,63	

Informacje o ostatecznej sumie punktów, które zdobyły poszczególne mikrokontrolery, przedstawiono również w formie graficznej, na poniższym wykresie.



Rysunek 11: Graficzne porównanie mikrokontrolerów

W kolejnej wyróżniającej się grupie można zaobserwować dwa wyróżniające się mikrokontrolery. Są to ESP8266-01 oraz ESP8266-07. Uzyskały one kolejno 876 oraz 1058 punktów.

Ostatnią, najsłabiej ocenianą grupą stanowią pozostałe dwa mikrokontrolery- Arduino Leonardo oraz Arduino UNO. Zebrały one kolejno 550 oraz 547 punktów.

4. Podsumowanie

Przeprowadzona analiza porównawcza pozwoliła na wzajemne porównanie badanych mikrokontrolerów pod wieloma względami, takimi jak temperatura generowana przez dany mikrokontroler, czas, w jakim wykonają się zadane obliczenia, wartości parametru RSSI w zależności od wzajemnej odległości urządzenia nadawczego oraz urządzenia odbiorczego, czy wreszcie pod względami ogólnymi badanych mikrokontrolerów, również parametrów hardware'u.

Można stwierdzić, iż testowane mikrokontrolery nie wykazywały żadnych niestabilności w swoim działaniu, nawet podczas stosunkowo długim czasie działania, przy badaniu temperatur generowanych przez mikrokontrolery, którego czas trwania oscylował w okolicach sześciu godzin.

Literatura

[1] T. Francuz, AVR: praktyczne projekty, Helion, 2013.

[2] B. Danowski-Ździebło, Wi-Fi: domowe sieci bezprzewodowe, Helion, 2010.

[3] W. Golde, Układy elektroniczne. T.1. Wydawnictwa Naukowo-Techniczne, 1970.

[4] P. Horowitz, H. Winfield, Sztuka elektroniki, cz.1 i 2, WKŁ, 1995.

[5] G. Holden, Sieci domowe i bezprzewodowe, NAKOM, 2010.

[6] J. Bielecki, Od C do C++: programowanie obiektowe w języku C, Wydawnictwa Naukowo-Techniczne, 1990.

[7] D. Mendrala, M. Szeliga, Praktyczny kurs SQL, Wydanie III Helion, 2015.

[8] J. Doliński, Mikrokontrolery AVR w praktyce, BTC, 2003.

[9] R. Baranowski, Mikrokontrolery AVR AT-mega w praktyce, BTC, 2005.

[10] T. Francuz, AVR: układy peryferyjne, Helion, 2014.

[11] Dokumentacja mikrokontrolera Wemos D1 mini V2, https://docs.wemos.cc/en/latest/d1/d1_mini.html, [04.2020].

[12] Dokumentacja mikrokontrolera ESP8266-01, http://www.microchip.ua/wireless/esp01.pdf, [03.2020].

[13] Dokumentacja mikrokontrolera ESP8266-07, https://www.mikrocontroller.net/attachment/338570/Ai-thinker_ESP-07_WIFI_Module-EN.pdf, [02.2020].

[14] Dokumentacja mikrokontrolera Arduino Leonardo, https://store.arduino.cc/arduino-leonardo-with-headers, [03.2020].

[15] Dokumentacja mikrokontrolera Arduino UNO, https://botland.com.pl/pl/arduino-moduly-glowne/1060-arduino-uno-rev3-a000066-8058333490090.html?fbclid=IwAR1IL1vK-1QqX9wde4o-ER4dOfbqbIoecgkKMTicCS8XaCAh-CXXcnotdbo&gclid=Cj0KCQjw3Nv3BRC8ARIsAPh8hgJ1nwUhQmXotmqvJ53--dxpB8ceqn5ygNmOzgnGQ6ZPYM4x64BStjcaAqV-EALw_wcB%20, [04.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 373–378
Received: 20 September 2020
Accepted: 7 November 2020

Oracle 19c, SQL Server 2019, Postgresql 12 and MySQL 8 database systems comparison

Porównanie systemów bazodanowych Oracle 19c, SQL Server 2019, PostgreSQL 12 oraz MySQL 8

Arkadiusz Solarz*, Tomasz Szymczyk

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

Abstract
This article presents a comparative analysis of four popular database technologies. Commercial Oracle Database and SQL Server systems have been compared with open source database management systems: PostgreSQL and MySQL. These systems have been available on the market for over a dozen years. Versions released in 2019 were selected for testing and comparison. For the purposes of the comparative analysis, a database schema was developed and instantiated. Then, test scenarios have been developed. They have been prepared on the basis of the most popular operations performed with the use of database systems.

Keywords: relational database; performance; SQL

Streszczenie
W artykule przedstawiona została analiza porównawcza czterech technologii bazodanowych. Komercyjne systemy Oracle Database i SQL Server porównane są z darmowymi systemami do zarządzania bazą danych: PostgreSQL i MySQL. Systemy te dostępne są na rynku od kilkunastu lat, do testów i porównania wybrane zostały wersje wydane w 2019 roku. Na potrzeby analizy porównawczej zaprojektowano oraz utworzono schemat bazy danych. Następnie opracowano scenariusze testowe. Przygotowane zostały one w oparciu o najpopularniejsze operacje wykonywane z wykorzystaniem systemów bazodanowych.

Słowa kluczowe: relacyjne bazy danych; wydajność; SQL

*Corresponding author
Email address: **arkadiusz.solarz@pollub.edu.pl** (A. Solarz)

1. Wstęp

W obecnych czasach szeroko rozwinięte są aplikacje biznesowe, programy komputerowe oraz aplikacje internetowe. Rozwiązania te w dużym stopniu bazują na danych oraz zarządzaniu nimi. Istotnym elementem takich aplikacji jest zarówno przetwarzanie nowych danych jak i zapewnienie dostępu do danych zapisanych wcześniej [1,2].

Przechowywanie i odpowiednie zabezpieczenie danych stanowi dla twórców oprogramowania niemałe wyzwanie. Istnieje wiele sposobów przechowywania danych, a wybór odpowiedniej metody powinien być świadomy i przemyślany [1,3]. Zastosowanie arkusza kalkulacyjnego sprawdzi się w przypadku małej grupy użytkowników, która nie potrzebuje wielu skomplikowanych funkcji. Systemy bazodanowe wykorzystywane są do zarządzania znacznie większymi zbiorami danych, zapewniając jednocześnie dostęp wielu użytkownikom. Istnieje wiele różnych typów baz danych takich jak: relacyjne, obiektowe, relacyjno-obiektowe, NoSQL, grafowe czy też hurtownie danych [4-7]. Rozwój w dziedzinie sprzętu oraz programowania sprawił że powstają nowoczesne rozwiązania takie jak bazy danych chmurowe, wielomodelowe bazy danych, samoczynne bazy danych lub dokumentowe bazy danych [1,5,7,8].

Na rynku dostępnych jest wiele systemów zarządzania bazą danych, występuje zarówno oprogramowanie komercyjne jak i programy na licencji open source, różniące się pod względem obsługiwanego modelu danych, oraz stopnia zaawansowania i dostępnych funkcjonalnościach [9]. Na rysunku 1 przedstawiony jest ranking systemów bazodanowych bazujący na ich popularności [10]. W artykule przedstawiono porównanie czterech systemów: Oracle Database 19c, MySQL 8, SQL Server 2019 oraz PostgreSQL 12.

Rank Aug 2020	DBMS	Database Model	Score Aug 2020
1.	Oracle	Relational, Multi-model	1355.16
2.	MySQL	Relational, Multi-model	1261.57
3.	Microsoft SQL Server	Relational, Multi-model	1075.87
4.	PostgreSQL	Relational, Multi-model	536.77
5.	MongoDB	Document, Multi-model	443.56
6.	IBM Db2	Relational, Multi-model	162.45
7.	Redis	Key-value, Multi-model	152.87
8.	Elasticsearch	Search engine, Multi-model	152.32
9.	SQLite	Relational	126.82
10.	Microsoft Access	Relational	119.86

Rysunek 1: Ranking popularności systemów bazodanowych [10]

2. Środowisko testowe

W celu przetestowania wydajności silników bazodanowych zostało utworzone środowisko testowe, na które

składa się zarówno hardware i software. Wykorzystane oprogramowanie uruchomiono w środowisku Windows 10. Zainstalowane oprogramowanie to: PostgreSQL 12, MySQL 8, SQL Server 2019, Oracle Database 19c

2.1. Baza danych

Omawiane systemy zarządzania bazą danych są narzędziami do obsługi baz danych wykorzystujących model relacyjny [6,10-12]. W tym celu do przeprowadzenia testów utworzona została relacyjna baza danych, której schemat przedstawiony jest na rysunku 2. Przedstawiona baza danych została zaprojektowana do obsługi systemu wypożyczalni samochodów. Celem zastosowania takiej bazy danych jest przetwarzanie transakcyjne (OLTP Online Transaction Processing). Tego typu baz używa się głównie w różnego rodzaju systemach ewidencyjnych takich jak: wypożyczalnie, systemy rezerwacji, sklepy internetowe, biblioteki, bankowość elektroniczna i inne [1,3,4].

Schemat bazy został uproszczony i zmodyfikowany głównie z myślą o przeprowadzeniu testów. Na stworzonym schemacie umieszczono jedynie niezbędne struktury, które pozwolą na traktowanie modelu bazy danych jako jedną spójną całość i wykonywanie zapytań SQL które będą zbliżone do rzeczywistych operacji [13,14].

Na podstawie przedstawionego poniżej schematu zostały przygotowane cztery skrypty SQL tworzące w bazie danych fizyczne tabele oraz powiązania między nimi.

2.2. Metoda testowania

Porównanie wydajności silników bazodanowych polegać będzie na przeprowadzeniu serii testów i zmierzeniu czasu ich wykonania. W celu zmniejszenia niepewności pomiarów, każdy test został wykonany pięciokrotnie, a jako wynik końcowy posłuży średnia arytmetyczna, obliczona z sumy wszystkich cząstkowych czasów wykonania zapytania - t_i podzielonej przez liczbę pomiarów – n.

$$t = \frac{\sum_n^1 t_i}{n} \quad (1)$$

Rozpoczęcie testowania następuje od jednego rekordu, w każdej kolejnej próbie liczba rekordów w tabeli jest odpowiednio zwiększana poprzez wstawienie danych testowych.

Podczas testów sprawdzone zostaną najczęściej wykonywane operacje związane z relacyjnymi bazami danych: wyszukiwanie danych w bazie, grupowanie oraz wstawienie danych. Przetestowane zostanie również wykonanie funkcji typowo administracyjnych takich jak utworzenie kopii zapasowej oraz jej przywrócenie [15].

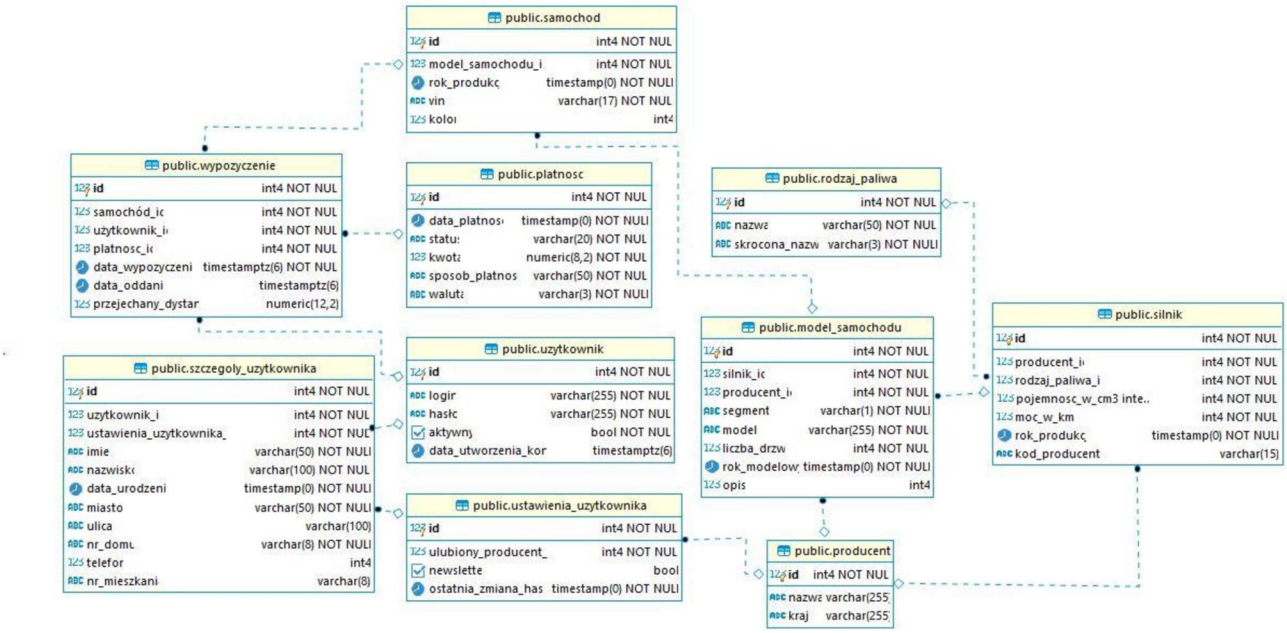
2.3. Technologie oraz narzędzia

Na czas wykonywania zapytania oraz wydajność systemu bazodanowego wpływ ma sprzęt (hardware), na którym jest on uruchomiony. Aby wyniki były niezależne od sprzętu, wszystkie testy zostały wykonane z wykorzystaniem laptopa o następującej specyfikacji:

- System operacyjny: Windows 10
- Procesor: Intel Core i3 6006U
- RAM: 8GB DDR4
- Dysk: SSD 120GB + HDD 500 GB

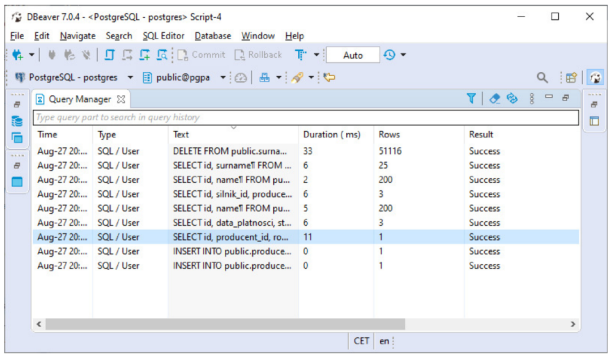
System operacyjny zainstalowany był na dysku SSD, systemy do zarządzania bazą danych na dysku HDD. Podczas testu urządzenie pracowało z włączonym zasilaniem i miało ustawiony tryb „Najwyższa wydajność”. W trakcie wykonywania testów programy, które mogłyby mieć wpływ na zużywanie zasobów laptopa zostały wyłączone.

Narzędziem wykorzystanym do połączenia z bazą danych i przeprowadzenia testów wydajnościowych był klient Dbeaver w wersji 7.0.4. Program ten jest darmo-



Rysunek 2: Schemat autorskiej bazy danych

wy i obsługuje połączenie z badanymi systemami zarządzania bazą danych PostgreSQL, SQL Server, Oracle Database, MySQL oraz wieloma innymi bazami w tym nierelacyjnymi (NoSQL). Narzędzie to zostało wybrane ze względu na dużą funkcjonalność, łatwość konfiguracji i połączenia z bazą danych. Zastosowanie takiego podejścia sprawia, że testy będą wykonywane w sposób jednolity. Do pomiaru czasu wykorzystany zostanie menedżer zapytań, okno programu przedstawione jest na rysunku 3. Widoczne są w nim najważniejsze informacje dotyczące zapytania przetworzonego przez bazę: czas jego wykonania w milisekundach, liczba wierszy, która została zwrócona lub zmodyfikowana oraz status informujący o poprawności wykonania [16].



Rysunek 3: Widok okna programu do pomiaru czasu wykonania zapytania

Dodatkowe narzędzia wykorzystane w testach wydajnościowych to programy dołączone lub zintegrowane z systemem bazodanowym. Wykorzystane zostały narzędzia do tworzenia i przywracania kopii zapasowej bazy danych. Dla systemu PostgreSQL są to pg_dump i pg_restore. W przypadku Oracle będą to narzędzia IMP oraz EXP. Do eksportu i importu danych w SQL Server wykorzystane zostanie składania języka Transact SQL która umożliwia wykonanie tych operacji poprzez odpowiednie zapytanie [17,18].

3. Testy wydajnościowe

Podczas testów wydajnościowych wykonane zostały popularne operacje związane z relacyjnymi bazami danych: wyszukiwanie, grupowanie oraz wstawienie danych. Zmierzone zostały także czasy utworzenia kopii zapasowej oraz jej przywrócenia.

3.1. Wyszukiwanie danych

Scenariusz testowy polegał na zmierzeniu czasu odpowiedzi bazy danych, który potrzebny jest na wyszukanie określonej informacji. W podanym przykładzie wybierani są użytkownicy, którzy na imię mają JAN. (Podczas wstawiania danych wszystkie imiona i nazwiska wpisane zostały w całości z dużych liter, również bazy danych podczas tworzenia zostały skonfigurowane tak aby wielkość liter była rozróżniana.) Do wybierania danych posłużyło zapytanie przedstawione na listingu 1.

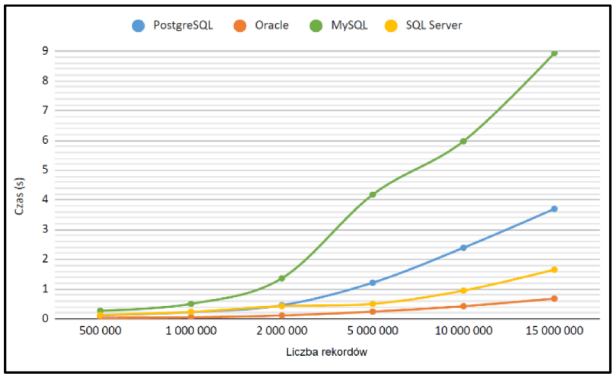
Listing 1: Zapytanie do wyszukiwania danych

```
SELECT imie, nazwisko, miasto, ulica, nr_domu,
nr_mieszkania, FROM szczegoly_uzytkownika
WHERE IMIE = 'JAN';
```

W tabeli 1 przedstawione są wyniki testów przeprowadzonych dla różnej liczby rekordów w tabeli z użytkownikami. Na rysunku 4 znajduje się wykres, na którym przedstawiono czasy wykonania zapytań, gdy liczba rekordów w tabeli wynosiła 500 000 lub więcej. Wraz ze wzrostem ilości danych różnice uwypuklają się. PostgreSQL potrzebował średnio dwa razy mniej czasu na odpowiedź, natomiast najwydajniejszy okazał się Oracle, który w ostatnim teście był o około sekundę szybszy od SQL Server 2019.

Tabela 1: Pomiary czasu wyszukiwania danych

Liczba rekordów	Czas wykonania (ms)			
	Postgres	Oracle	MySQL	SQL Server
1	3,0	2,4	3,0	3,0
1 000	3,2	3,8	6,4	6,0
5 000	3,0	3,4	6,2	9,0
10 000	6,4	3,0	13,8	9,4
20 000	9,4	3,0	13,4	12,4
50 000	18,6	6,2	32,2	22,0
100 000	28,0	9,4	66,0	28,2
200 000	45,2	12,6	124,8	53,0
500 000	118,6	31,2	274,2	128,2
1 000 000	225,8	49,8	503,0	234,2
2 000 000	456,6	112,2	1364,4	428,0
5 000 000	1215,6	243,2	4180,2	502,8
10 000 000	2391,8	424,2	5977,6	949,8
15 000 000	3700,0	681,8	8941,6	1654,6



Rysunek 4: Wykres czasu wyszukiwania w zależności od liczby rekordów

3.2. Grupowanie danych

Do grupowania danych wykorzystany został skrypt przedstawiony na listingu 2. Wykonywana operacja służy do wyświetlenia liczby klientów wypożyczalni z poszczególnych miast w zaczynając od miast z naj-

Listing 2: Zapytanie wykorzystane do grupowania danych

```
SELECT miasto, count(*) AS liczba_klientow
FROM SZCZEGOLY_UZYTKOWNIKA GROUP
BY miasto ORDER BY liczba_klientow desc;
```

większą liczbą klientów. Otrzymane rezultaty przedstawione są w tabeli 2. Podobnie jak podczas operacji wyszukiwania danych, najwolniejszy okazał się darmowy system zarządzania bazą danych MySQL 8. Czasy odpowiedzi są kilkanaście razy dłuższe niż w przypadku czołowych systemów: Oracle Database 19c i SQL Server 2019.

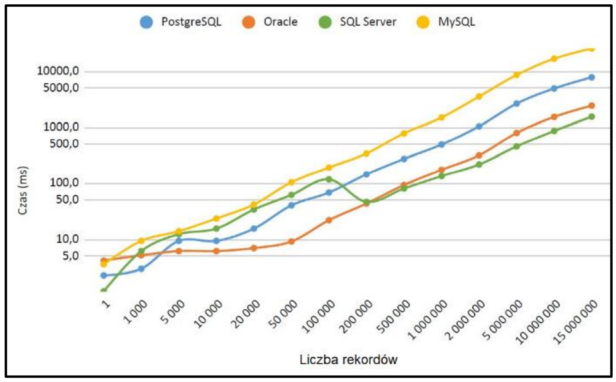
Tabela 2: Pomiary czasu grupowania danych

Liczba rekordów	Czas wykonania (ms)			
	Postgres	Oracle	MySQL	SQL Server
1	2,3	4,2	3,6	1,2
1 000	3,0	5,2	9,4	6,2
5 000	9,4	6,2	14,0	12,4
10 000	9,4	6,2	23,6	15,6
20 000	15,6	7,0	42,0	34,2
50 000	40,6	9,2	104,8	62,4
100 000	68,8	22,0	193,2	118,6
200 000	146,0	43,8	343,8	46,8
500 000	275,4	93,6	789,6	81,2
1 000 000	496,2	175,8	1505,8	135,6
2 000 000	1052,6	318,8	3573,8	219,6
5 000 000	2666,6	801,8	8654,2	458,4
10 000 000	4960,0	1544,8	16880,8	870,4
15 000 000	7861,0	2451,8	25716,2	1562,0

Trzy bazy danych osiągają czasy odpowiedzi poniżej 500 ms w sytuacji, gdy w tabeli znajduje się mniej niż milion rekordów. Najwolniejszy czas osiągnął MySQL, grupowanie danych gdy w tabeli znajduje się 1 000 000 rekordów zajmuje 1,5 sekundy. Czas ten jest dłuższy od czasów pozostałych systemów, lecz jest on akceptowalny w kontekście wykorzystania bazy w systemie OLTP. Powyżej tej liczby rekordów czasy wykonania zapytania wydłużają się proporcjonalnie do liczby rekordów w tabeli. Aby zobrazować różnicę pomiędzy czasami wykonania zapytania, warto spojrzeć na wykres (rysunek 5). Wynika z niego, że w czasie 1,5 sekundy, MySQL jest w stanie wykonać zapytanie dla 1 000 000 rekordów, Oracle potrafi wykonać to samo zapytanie w tym samy czasie dla 10 000 000 rekordów, natomiast SQL Server dla 15 000 000 rekordów.

3.3. Odtwarzanie bazy danych

Odtwarzanie bazy danych jest operacją wykonywaną zdecydowanie rzadziej niż tworzenie kopii zapasowej, jednak niezwykle ważne jest, aby w przypadku awarii lub problemów przywrócenie danych i ponowne uruchomienie usługi odbyło się w jak najkrótszym czasie.
W omawianym scenariuszu wykorzystane zostały pliki kopii zapasowej przygotowane według następującej metody: w każdym z plików znajduje się wszystkie tabele, w których początkowo znajduje się 71 300 rekordów. Wraz z kolejnymi testami do tabeli z wypożyczeniami wstawiane są kolejne rekordy. Dodawanie danych do tej tabeli ma symulować działanie rzeczywistego systemu, w którym istniejący użytkownicy dokonują wypożyczeni które zapisywane są właśnie w tej tabeli bazodanowej.

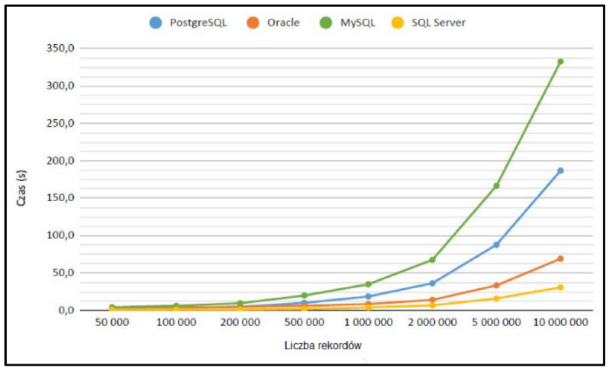


Rysunek 6: Porównanie czasu wykonania zapytania grupującego dane dla różnej liczby rekordów

Tabela 3: Pomiary czasu importu danych

Liczba rekordów	Czas wykonania (ms)			
	Postgres	Oracle	MySQL	SQL Server
1	1386,2	3227,8	2689,4	1018,4
500	1292,8	3417,0	2680,2	985,4
1 000	1240,0	3837,6	2721,0	1017,8
5 000	1238,0	3447,2	3036,4	1026,8
10 000	1464,6	3621,2	3065,6	1062,2
50 000	2125,6	3629,0	4245,4	899,0
100 000	2892,2	4245,4	6147,0	1064,2
200 000	4785,6	4656,2	9618,8	1120,8
500 000	10098,0	6255,4	19922,2	2128,0
1 000 000	18633,0	8470,4	35017,0	4028,2
2 000 000	36384,6	14118,4	67672,4	6760,2
5 000 000	87881,8	33458,4	166689,4	15987,4
10 000 000	186928,0	69167,4	332796,0	30759,8

Testy zostały przeprowadzone w zakresie od 1 do 10 000 000 rekordów, analizując otrzymane wyniki można jednoznacznie wskazać system, który najlepiej poradził sobie z odtwarzaniem bazy danych z kopii zapasowej. W każdym z przypadków czas importu bazy danych najkrótszy był dla SQL Server 2019. Warto zwrócić uwagę, że system ten nie korzysta z oddzielnego narzędzia do importu, ale wykonuje go poprzez składnię wbudowaną w język Transact-SQL rozwijany wraz z systemem bazodanowym. Drugi pod względem szybkości importowania jest system Oracle. Analizując rezultaty należy zwrócić uwagę na względne różnice pomiędzy systemami, w przypadku systemu Oracle czas odtworzenia bazy danych jest średnio dwa razy dłuższy



Rysunek 5: Porównanie czasu importu danych

niż czas analogicznej operacji wykonanej z wykorzystaniem SQL Server.

W tabeli 4 tabeli porównany został rozmiar wyeksportowanego pliku z daną liczbą rekordów. Na podstawie przedstawionych danych można stwierdzić, że najwyższy współczynnik kompresji danych występuje w przypadku systemu MySQL. Rozmiar pliku wygenerowanego z bazy Oracle miał dwukrotnie większy rozmiar niż analogiczny plik utworzony z wykorzystaniem systemu bazodanowego MySQL. Wysoki stopień kompresji może być też powodem wolnego odtwarzania bazy danych, w przypadku MySQL potwierdza to wykres na rysunku 6.

Tabela 4: Rozmiar odtwarzanej bazy danych

Liczba rekordów	Rozmiar (kB)			
	Postgres	Oracle	MySQL	SQL Server
1	1266	1335	542	7325
500	1294	1366	557	7325
1 000	1320	1398	571	7451
5 000	1540	1646	687	7581
10 000	1812	2273	831	7837
50 000	4000	4488	1987	11357
100 000	6734	7651	3431	14429
200 000	12202	14086	6328	19549
500 000	28604	33347	15138	35937
1 000 000	54958	65467	29498	62569
2 000 000	109636	130684	58559	116853
5 000 000	273668	326336	145741	278658
10 000 000	547056	652422	291035	547985

3.4. Wyszukiwanie danych na podstawie wzorca

Scenariusz testowy polegał na selekcji rekordów z tabeli na podstawie podanego wzorca. Wyszukiwanie odbywa się z wykorzystaniem trzyliterowej frazy, która może wystąpić na dowolnej pozycji. Test został przeprowadzony poprzez wyszukiwanie osób, które w nazwisku mają frazę PRO. Wykonanie takiego zapytania pozwala wyszukać osoby bez konieczności wprowadzenia pełnego nazwiska. Zapytanie wykorzystane do wyszukania osób z podaną frazą widoczne jest na listingu 3

Listing 3: Zapytanie wykorzystane do wyszukiwania danych na podstawie wzorca

```
SELECT imie, nazwisko, miasto, ulica, nr_domu,
nr_mieszkania, FROM szczegoly_uzytkownika su
WHERE nazwisko LIKE '%PRO%';
```

Szczegółowe wyniki testów przedstawione są w tabeli 5. Test wykonano dla czternastu różnych przypadków. W każdym z nich w tabeli z której wybierane są dane znajduję się określona liczba rekordów. Pierwszy scenariusz wykonano, gdy w tabeli znajdował się jeden rekord, w kolejnych testach ich liczba była odpowiednio zwiększana. W ostatnim przypadku test wykonany został dla 15 000 000 rekordów. Najkrótszy czas odpowiedzi uzyskany został gdy zapytanie było przetwarzane z wykorzystaniem systemu Oracle, drugi z komercyjnych systemów (SQL Server) osiągnął znacznie gorsze wyniki przegrywając w niektórych próbach z MySQL.

Tabela 5: Wyszukiwanie na podstawie wzorca

Liczba rekordów	Czas wykonania (ms)			
	Postgres	Oracle	MySQL	SQL Server
1	6,0	1,8	3,0	6,2
1 000	1,0	4,0	5,4	9,4
5 000	3,0	4,2	6,4	12,4
10 000	6,2	9,4	13,4	18,8
20 000	9,4	6,2	20,2	43,8
50 000	21,8	9,4	46,4	96,8
100 000	37,4	24,8	73,4	165,6
200 000	75,4	43,6	144,6	284,4
500 000	161,6	112,4	320,8	690,6
1 000 000	306,4	202,6	639,2	1302,8
2 000 000	594,8	384,2	1614,6	1507,6
5 000 000	1496,0	914,8	5034,4	3215,2
10 000 000	3057,4	1880,0	7686,8	6710,0
15 000 000	4337,6	2856,4	10966,4	9490,8

Czas wykonania operacji wyszukiwania na podstawie wzorca jest dłuższy niż czas analogicznej operacji wykonanej z użyciem operatora równości. Procentowy przyrost czasu wykonania zapytania przedstawiony jest w tabeli 6. Obliczony został on dla każdego systemu zarządzania bazą danych ze wzoru

$$x_i = \left(\frac{t1_i}{t2_i} * 100\%\right) - 100\% \quad (2)$$

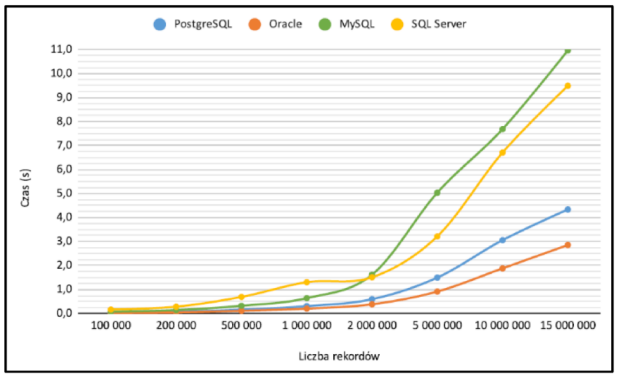
gdzie:
i – liczba rekordów w tabeli
x_i – procentowy przyrost czasu dla i-tej liczby rekordów w tabeli
$t1i$ – czas wykonania zapytania wyszukującego dane na podstawie wzorca dla i-tej liczby rekordów
$t2_i$ – czas wykonania zapytania wyszukującego dane dla i-tej liczby rekordów

Tabela 6: Przyrost czasu wykonania zapytania wyszukiwania na podstawie wzorca w stosunku do wyszukiwania z wykorzystaniem operatora równości

Liczba rekordów	Procentowy przyrost czasu			
	Postgres	Oracle	MySQL	SQL Server
1	100,00%	-25,00%	0,00%	106,67%
1 000	-71,88%	5,26%	-15,63%	56,67%
5 000	0,00%	23,53%	3,23%	37,78%
10 000	-3,13%	213,33%	-2,90%	100,00%
20 000	0,00%	106,67%	50,75%	253,23%
50 000	17,20%	51,61%	44,10%	340,00%
100 000	33,57%	163,83%	11,21%	487,23%
200 000	66,81%	246,03%	15,87%	436,60%
500 000	36,26%	260,26%	16,99%	438,69%
1 000 000	35,70%	306,83%	27,08%	456,28%
2 000 000	30,27%	242,42%	18,34%	252,24%
5 000 000	23,07%	276,15%	20,43%	539,46%
10 000 000	27,83%	343,19%	28,59%	606,46%
15 000 000	17,23%	318,95%	22,64%	473,60%

Na podstawie danych z tabeli 6 można wywnioskować, że zmiana typu wyszukiwania wywołuje duży przyrost czasu wykonania zapytania w dwóch systemach komercyjnych: SQL Server i Oracle Database. Waha się on w zależności od liczby rekordów w tabeli. Jednak patrząc całościowo jest to wzrost o kilkaset procent. Odmienna sytuacja wystąpiła w przypadku systemów na licencji open source. Przyrost czasu wykonania zapytania wynosił w nich od kilkunastu do kilkudziesięciu procent.

Analizując bezpośrednie wyniki scenariusza testowego (rysunek 7), najlepsze czasy wyszukiwania na podstawie wzorca osiągnięte zostały przez system Oracle Database 19c. Duży przyrost czasu wykonania zapytania dla bazy danych SQL Server spowodował, że podczas tego scenariusza wydajnościowo uplasowała się ona na trzecim miejscu, znacząco odstając od Oracle i PostgreSQL.



Rysunek 7: Wyszukiwanie na podstawie wzorca - wykres

4. Wnioski

Przeprowadzone badania wykazały różnice w wydajności obsługi relacyjnych baz danych pomiędzy badanymi systemami. W testach najlepsze rezultaty osiągnęły systemy komercyjne czyli Oracle Database 19c który okazał się najlepszy w większości testów oraz SQL Server 2019. Dwa ostatnie miejsca przypadły dla PostgreSQL 12 oraz MySQL 8, który w zdecydowanej większości prób odnotował ostatni rezultat. Pomimo uzyskania niekonkurencyjnych rezultatów w odniesieniu do systemów komercyjnych, pojawiły się również pozytywne akcenty. MySQL uzyskał najlepszy wynik w teście dotyczącym utworzenia kopii zapasowej całej bazy danych, która zajmowała najmniej miejsca, natomiast PostgreSQL okazał się lepszy od SQL Server w teście wyszukiwania na podstawie wzorca.

Literatura

[1] A. Pelikant, Bazy danych. Pierwsze starcie, Helion, 2012.

[2] C.J. Date, Database Design and Relational Theory, O'Reilly, 2012.

[3] B. Pękala, Bazy danych. Teoria i praktyka, Wydawnictwo Uniwersytetu Rzeszowskiego, 2015.

[4] L. Rockoff, Język SQL. Przyjazny podręcznik, Helion 2017.

[5] S. Feuerstein, B. Pribyl, Oracle PL/SQL Programming, O'Reilly, 2014.

[6] Oficjalna dokumentacja SQL Server 2019 https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15, [05.2020].

[7] B. Peasland, Oracle DBA Mentor: Succeeding as an Oracle Database Administrator, Apress, 2019.

[8] L. Davidson, J. Moss, Pro SQL Server Relational Database Design and Implementation, Apress, 2016.

[9] C. Mehta, A. Bhavsar, H. Oza, S. Shah, MySQL 8 Administrator's Guide: Effective guide to administering high-performance MySQL 8 solutions, Packt Publishing, 2018.

[10] Ranking systemów zarządzania bazą danych https://db-engines.com/en/ranking ,[08.2020].

[11] Oficjalna dokumentacja bazy danych PostgreSQL https://www.postgresql.org/files/documentation/pdf/12/postgresql-12-A4.pdf, [02.2020].

[12] R. West, SQL Server 2019 Administration Inside Out, 2020.

[13] A. Silberschatz, H.F. Korth, S. Sudarshan, Database System Concepts, McGraw-Hill Higher Education, 2019.

[14] W. Khadzhynov, P. Ratuszniak, Wprowadzenie do systemów baz danych (Wydanie VII), Helion, 2016.

[15] M. Winand, SQL Performance Explained Everything Developers Need to Know about SQL Performance, 2012.

[16] Dokumentacja narzędzia wykorzystanego do pomiarów https://github.com/dbeaver/dbeaver/wiki , [07.2020].

[17] S. Riggs, G. Ciolli, S.K. Meesala, PostgreSQL 11 Administration Cookbook, Packt Publishing, 2019.

[18] M. Leach, T. Lahdenmäki, Database Systems: Design, Implementation & Management 12th Edition, Cengage Learning, 2016.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 379–383
Received: 26 October 2020
Accepted: 5 November 2020

Comparative analysis of medical images watermarking methods

Analiza porównawcza metod znakowania wodnego obrazów medycznych

Sylwia Duda*, Dominik Fijałek*, Grzegorz Kozieł

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

Abstract

The article is devoted to the analysis of watermarking algorithms in terms of their use in marking medical images. The algorithms based on the Integer Wavelet Transform (IWT), Discrete Cosine Transform (DCT), and Singular Value Decomposition (SVD) were compared. The algorithms were implemented using the combinations: IWT, IWT-DCT, and IWT-SVD. As part of the research, the level of disturbances caused by embedding the watermark was checked using subjective and objective methods. The attack resistance of the watermarked images was tested and the steganographic capacity was measured. All algorithms are based on IWT, however, each has different advantages. The algorithm based on the IWT showed the highest capacity. The most resistant to attacks is IWT-SVD, and the lowest level of interference was obtained for the IWT-DCT algorithm.

Keywords: watermarking; IWT; DCT; SVD

Streszczenie

Artykuł poświęcono analizie algorytmów znakowania wodnego pod kątem wykorzystania w znakowaniu obrazów medycznych. Porównano algorytmy oparte o całkowitą transformatę falkową (IWT), dyskretną transformatę kosinusową (DCT) i rozkład według wartości osobliwych (SVD). Zaimplementowano algorytmy stosując kombinacje: IWT, IWT-DCT i IWT-SVD. W ramach badań sprawdzono poziom zakłóceń spowodowanych osadzaniem znaku wodnego przy pomocy metod subiektywnych i obiektywnych. Przeprowadzono badania odporności oznakowanych obrazów na ataki i zmierzono pojemność steganograficzną. Wszystkie algorytmy bazują na IWT, jednakże każdy z nich ma inne zalety. Największą pojemność wykazał algorytm oparty o IWT. Najodporniejszy na ataki jest IWT-SVD, a najmniejszy poziom zakłóceń uzyskano dla algorytmu IWT-DCT.

Słowa kluczowe: znakowanie wodne; IWT; DCT; SVD

*Corresponding author

Email address: **sylwia.duda1@pollub.edu.pl** (S. Duda), **dominik.fijalek@pollub.edu.pl** (D. Fijałek)

1. Wstęp

Zapewnienie ochrony i bezpieczeństwa danych stanowi obecnie jedno z największych wyzwań. Ze względu na swoją wartość, często informacja musi zostać ukryta. Wraz z powstaniem komputerów zaczęto ukrywać informacje w obiektach cyfrowych. W tym celu wykorzystywana jest steganografia [1]. Postępująca cyfryzacja przyczyniła się do powstania cyfrowego znaku wodnego [2], który wywodzi się ze steganografii.

Znakowanie jest także techniką ukrywania informacji, ale istnieje kilka wytycznych, które rozróżniają pojęcie znakowania wodnego od steganografii [3].

Znaki wodne są stosowane przede wszystkim w celu potwierdzenia autentyczności i praw autorskich. Są stosowane także do wykrywania zmian w oznakowanych plikach. Znakowanie wodne znalazło zastosowanie w wielu dziedzinach, m.in. w medycynie.

W związku z oszustwami dotyczącymi dokumentacji cyfrowy znak wodny jest używany do ukrywania adnotacji medycznych. Pozwala to na potwierdzenie przynależności oraz na wykrycie ewentualnych zmian w oznakowanym dokumencie medycznym.

Obrazy medyczne należą do grupy najbardziej wymagających. W celu ukrycia adnotacji medycznych stosuje się techniki osadzania niewidocznego znaku wodnego, a więc działanie algorytmu nie powinno zniekształcać obrazu ze względu na charakter zdjęć medycznych oraz ważność detali. Oczekując wiernej rekonstrukcji oznakowanego obrazu oraz niezauważalności znaku wodnego, powstają coraz nowsze metody, ale niekoniecznie doskonalsze od istniejących.

W przeciągu ostatnich lat powstało wiele technik znakowania w dziedzinie przestrzennej oraz dziedzinie transformaty. Jednakże ostatnimi czasy popularność zyskały metody oparte o przekształcenie falkowe.

2. Wybrane algorytmy znakowania wodnego

2.1. Metoda ukrywania danych w oparciu o IWT

Pierwszym z testowanych algorytmów był zaproponowany w artykule pod tytułem “Distortionless data hiding based on integer wavelet transform” przez Guorong Xuan, Jiang Zhu, Jidong Chen, Shi Y.Q., Zhicheng Ni oraz Wei Su [4]. Jest to metoda wykorzystująca do rozkładu obrazu na podpasma całkowitą transformatę falkową. Dane są ukrywane w podpasmach o średniej i wysokiej częstotliwości. Osadzanie odbywa się zgodnie z następującymi krokami:

1. Obraz-nośnik jest wstępnie przetwarzany do całkowitej transformaty falkowej w celu podziału obrazu na podpasma.

2. Osadzenie znaku wodnego w podpasmach LH, skali szarości obrazów ośmiobitowych.
3. Zastosowanie LL i HL
4. Zastosowanie odwrotnej całkowitej transformaty falkowej w celu uzyskania oznakowanego obrazu.

Algorytm używa IWT w celu uzyskania jak największej pojemności steganograficznej oraz uniknięcia wystąpienia błędu zaokrąglenia, co może wystąpić w przypadku użycia dyskretnej transformaty falkowej.

2.2. Metoda znakowania wodnego oparta na IWT i DCT

Kolejna badana metoda wykorzystuje połączenie całkowitej transformaty falkowej (IWT) i dyskretnej transformaty kosinusowej (DCT). Została ona zaproponowana w artykule "Text-image watermarking based on integer wavelet transform" przez Reem A. Alotaibi i Lamiaa A. Elrefaei [5]. Osadzanie znaku wodnego w danych odbywa się według następującego algorytmu:

1. Rozłożenie obrazu nośnika przy pomocy całkowitej transformaty falkowej na cztery podpasma: LL1, LH1, HL1, HH1.
2. Użycie dyskretnej transformaty kosinusowej (DCT) na podpasmie LL1 z podziałem na bloki o rozmiarze 8x8 pikseli.
3. Wybór współrzędnych w bloku, w których będzie osadzany znak wodny.
4. Dobranie współczynnika skalowalności α zależnego od współrzędnych, na których osadzany zostanie znak wodny.
5. Osadzenie znaku wodnego w wybranych współczynnikach.
6. Użycie odwrotnej dyskretnej transformaty kosinusowej na poszczególnych blokach zmodyfikowanego podpasma LL1.
7. Uzyskanie oznakowanego obrazu poprzez użycie odwrotnej całkowitej transformaty falkowej na wszystkich podpasmach uzyskanych w pierwszym punkcie oraz zmodyfikowanym LL1.

W powyższym procesie osadzania do modyfikacji współczynników DCT został użyty wzór:

$$AC'_{b,i} = AC_{b,i} + (\alpha * V(i + (b-1) * N)), \quad (1)$$

gdzie:
b - numer bloku,
i - numer modyfikowanego współczynnika,
N - liczba modyfikowanych współczynników w bloku,
α - współczynnik skalowalności,
AC - współczynnik DCT,
V - wektor obrazu znaku wodnego.

2.3. Metoda znakowania wodnego oparta na IWT i SVD

Ostatni z badanych algorytmów zakłada wykorzystanie w procesie osadzania znaku wodnego całkowitej transformaty falkowej (IWT) oraz rozkład macierzy na wartości osobliwe (SVD). Metodę zaproponowali P. Gupta i G. Parmar w artykule pod tytułem "Image watermarking using IWT-SVD and its comparative analysis with DWT-SVD" [6]. Spośród dwóch porównywanych w tej pracy algorytmów w badaniach użyto pierwszego z nich. Osadzanie znaku wodnego odbywa się w następujących krokach:

1. Obraz oryginalny i obraz znaku wodnego są rozkładane za pomocą IWT na podpasma LL1, LH1, HL1 oraz HH1.
2. Podpasma LL1 nośnika i znaku wodnego rozkładane są na wartości osobliwe(SVD).
3. Macierz sigma jest obliczana jako fuzja obu macierzy powstałych w wyniku rozkładu na wartości osobliwe.
4. Nowe wartości dla podpasma LL1 są obliczane przy użyciu otrzymanej macierzy sigma oraz odwrotnego rozkładu na wartości osobliwe.
5. Obraz oznakowany jest otrzymywany poprzez użycie odwrotnego IWT na otrzymanym podpasmie LL1 oraz pozostałych z punktu 1.

Aby odzyskać osadzony znak wodny potrzebny jest osadzony znak wodny, obraz oryginalny oraz obraz oznakowany. Wyodrębnianie odbywa się w następujących krokach:

1. Podział obrazu oznakowanego, oryginalnego oraz znaku wodnego na podpasma za pomocą IWT.
2. Użycie SVD do rozkładu na wartości osobliwe obrazów i znaku wodnego.
3. Obliczenie nowej macierzy sigma poprzez zastosowanie fuzji macierzy sigma uzyskanych w kroku nr. 3 oraz współczynnika skalowania.
4. Zastosowanie odwrotnego rozkładu na wartości osobliwe do obliczenia wartości podpasma LL1.
5. Uzyskanie znaku wodnego poprzez zastosowanie odwrotnego IWT na nowo uzyskanym podpasmie LL1 i pozostałych.

3. Metodyka badawcza

Jak wspomniano na wstępie, obrazy medyczne należą do jednych z najbardziej wymagających grup obrazów. Ze względu na to, że oznakowane obrazy niekiedy odbiegają w mniejszym lub większym stopniu od oryginału, należy poddać badaniom jakość rekonstruowanych obrazów. Pojęcie obrazu dobrej jakości zależy od kontekstu. W przypadku medycyny jest to nie tylko obraz jasny, ostry i o wysokim kontraście, ale także jest to uzyskanie jak największego podobieństwa obrazu oznakowanego do oryginalnego. W związku z przedmiotem badań jakim są obrazy medyczne, nie istnieją metody, które w jednoznaczny sposób wyłonią jedną technikę znakowania wodnego jako najdoskonalszą spośród badanych. Do oceny oznakowanych zdjęć używane są metody subiektywne i obiektywne. Ze względu na istotność wizualną oraz ważność zachowania detali zdjęć, ocena subiektywna będzie ważniejsza niż obiektywna. Ocena subiektywna to nic innego jak ocena wizualna dokonana ludzkim okiem. Natomiast do obiektywnych zalicza się [7] m.in.:

- MSE - błąd średniokwadratowy,
- SNR - stosunek sygnału do szumu,

- PSNR - szczytowy stosunek sygnału do szumu,
- SSIM - podobieństwo strukturalne.

Dla poszczególnych metod znakowania wodnego zostały przeprowadzone badania jakości rekonstruowanych obrazów przy pomocy metod obiektywnych. Badania dla każdego z algorytmów zostały przeprowadzone na zbiorze 50 obrazów medycznych o wymiarach 512x512 pikseli. Natomiast jako znak wodny był używany tekst o długości 104 znaków.

Następnym z przeprowadzonych badań było określenie pojemości steganograficznej, czyli ilości danych jakie mogą zostać ukryte w obrazie o wymiarach 512x512 pikseli. Badanie polegało na stopniowym zwiększaniu ilości ukrywanych danych w obrazie. Oznakowane obrazy poddawano ocenie jakości oraz sprawdzano czy odzyskany znak wodny jest identyczny z pierwotnym. Dla algorytmu IWT-DWT dodatkowo zwiększano liczbę bitów (w bloku), na których osadzano dane.

Kolejnym z badań jakie przeprowadzono była analiza odporności na ataki. Oznakowane obrazy modyfikowano za pomocą następujących ataków:

- przesunięcie bitowe,
- rozmycie,
- dodanie szumu,
- usuwanie szumu,
- kompresja JPG,
- korekcja histogramu,
- dostrojenie kontrastu,
- korekcja gamma.

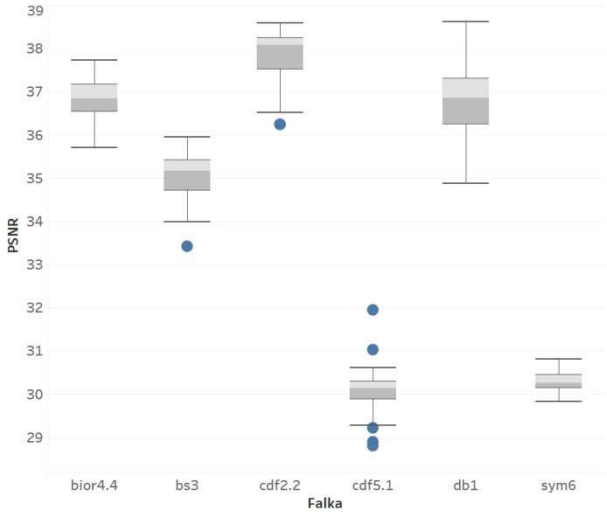
Następnie ze zmodyfikowanego obrazu próbowano wyodrębnić osadzony znak wodny. W badaniu jako znak wodny używany był tekst o długości 104 znaków lub obraz o wymiarach 32x32 pikseli z umieszczoną na nim informacją. W przypadku udanej próby odzyskania danych ze zmodyfikowanego obrazu dokonywano analizy podobieństwa wyodrębnionego znaku wodnego z oryginalnym. Gdy osadzany był obraz oceniano możliwość przeczytania tekstu umieszczonego na obrazie. Natomiast w przypadku ukrywania tekstu sprawdzana była liczba identycznych znaków.

4. Wyniki badań

Wszystkie algorytmy zostały zaimplementowane w środowisku Matlab a następnie przeprowadzone zostały badania. w testach użyto 50 ogólnodostępnych zdjęć medycznych, natomiast do testów odporności na ataki użyto jednego zdjęcia medycznego, zaś znak wodny stanowił tekst oraz obraz.

4.1. Metoda ukrywania danych w oparciu o IWT

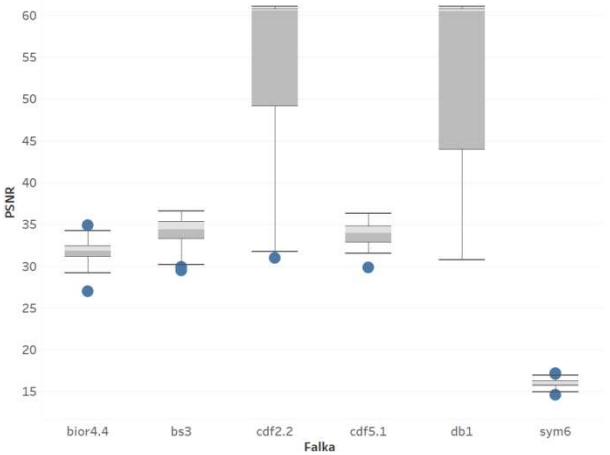
Na rysunku 1 przedstawiono wykres zależności wartości PSNR od użytej falki. Analizując otrzymane wyniki najlepsze wartości PSNR uzyskano dla falki z rodziny Cohena-Daubechies-Feauveau - cdf2.2. Niewiele gorsze wyniki otrzymano dla falek db1 oraz bior4.4, jednakże wartości PSNR uzyskane dla pierwszej z nich są bardziej rozproszone. Najmniej zadowalające wartości uzyskano dla falki cdf5.1 i sym6, a dla pierwszej z nich uzyskano wiele wyników odstających. Rozbieżność wyników oscyluje w granicach 28 – 38,5 dB.



Rysunek 1: Zależność wartości PSNR od rodzaju falki zastosowanej w algorytmie opartym o IWT

4.2. Metoda znakowania wodnego oparta na IWT i DCT

Na rysunku 2 przedstawiono wykres zależności wartości PSNR od użytej falki w całkowitej transformacie falkowej dla algorytmu opartego o IWT-DCT. Wyniki badań pokazują, że ponownie najlepsze wartości PSNR uzyskano dla falek cdf2.2 (32-62 dB) i db1 (31-61 dB). Natomiast rozstęp ćwiartkowy dla falki cdf2.2 jest mniejszy niż w przypadku falki db1. Pozostałe wyniki są znacznie niższe, a najgorzej wypadła falka sym6 uzyskując wartości PSNR w zakresie 14-16 dB. Natomiast falka bior4.4 w przypadku tego algorytmu uzyskała wyniki porównywalne z falkami bs3 oraz cdf5.1.

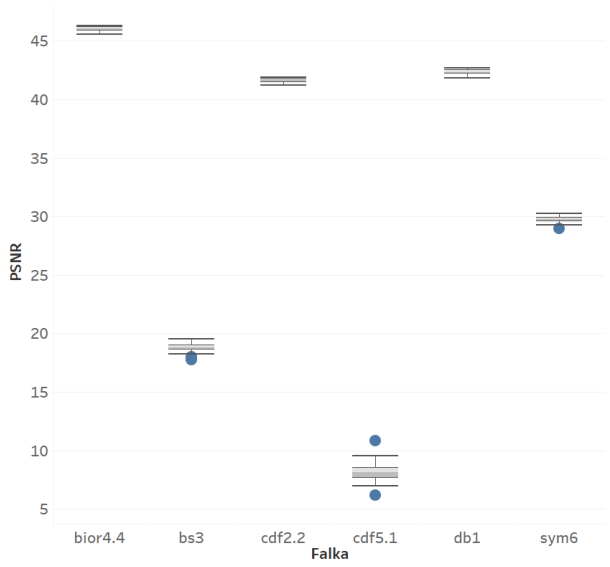


Rysunek 2: Zależność wartości PSNR od rodzaju falki zastosowanej w algorytmie opartym o IWT-DCT

4.3. Metoda znakowania wodnego oparta na IWT i SVD

Na rysunku 3 przedstawiono wykres zależności wartości PSNR od użytej falki dla algorytmu opartego o IWT-SVD. Najlepsze wyniki PSNR uzyskano dla falki bior4.4 – 45,5-47 dB. Następnie najlepsze wyniki otrzymano dla falek db1 – około 43 dB oraz cdf2.2 – około 42 dB. Zdecydowanie najgorsze wartości oscylu-

jące w zakresie 6,19-10,84 dB uzyskano dla falki cdf5.1 Algorytm IWT-SVD charakteryzuje się małym rozproszeniem wyników w obrębie danej falki.



Rysunek 3: Zależność wartości PSNR od rodzaju falki zastosowanej w algorytmie opartym o IWT-SVD

4.4. Porównanie metod

Zestawienie otrzymanych średnich wartości dla miar obiektywnych zostało przedstawiony w tabeli 1. Biorąc pod uwagę wyniki otrzymane dla miary sprawdzającej stosunek sygnału do szumu (PSNR) najlepsze wyniki uzyskano dla algorytmu wykorzystującego IWT-DCT(im wyższa wartość tym lepsza jakość). Pozostałe algorytmy uzyskały wartości średnie gorsze o ponad 10 dB, natomiast nieco lepiej wypadła w tym przypadku metoda oparta o IWT-SVD. Najgorsze wartości średnie miar PSNR oraz SNR otrzymano dla metody IWT. Wyniki średnie uzyskane dla miar MSE oraz SSIM otrzymane dla metody ukrywania danych opartej o IWT odbiegają znacznie od pozostałych algorytmów (im niższa wartość tym lepsza jakość), natomiast w przypadku pozostałych metod ukrywania danych wartości te są bardzo do siebie zbliżone.

Tabela 1: Porównanie wartości PSNR, SNR, MSE oraz SSIM dla poszczególnych algorytmów

Algorytm	SNR [dB]	PSNR [dB]	MSE	SSIM
IWT	32,27	37,86	10,36	0,9217
IWT-DCT	48,72	54,32	4	0,9991
IWT-SVD	36,18	41,64	4	0,9981

Wyniki badań pojemności steganograficznej przedstawione w tabeli 2 wskazują, że najwięcej danych można ukryć przy pomocy algorytmu opartego o IWT, natomiast algorytm wykorzystujący IWT-DCT pozwala na ukrycie jedynie 1024 bitów danych, taki wynik jest efektem ukrywania danych jedynie w jednym bicie podpasma LL1 przekształconego za pomocą DCT.

Tabela 2: Zestawienie wartości pojemności steganograficznej

Algorytm	Pojemność steganograficzna [b]
IWT	195000
IWT-DCT	1024
IWT-SVD	16384

Tabela 3 przedstawia wyniki badań związanych z atakami mającymi na celu uniemożliwienie odzyskania znaku wodnego, wykazały one brak odporności algorytmu opartego o IWT-DCT na wszystkie przeprowadzone ataki. Największą odporność uzyskano dla algorytmu wykorzystującego IWT-SVD, gdzie dane udało się odzyskać po przeprowadzeniu pięciu z dziewięciu ataków. w przypadku algorytmu opartego o IWT odporność wykazano w przypadku trzech prób usunięcia znaku wodnego, natomiast tylko w jednym przypadku algorytm pozwolił odzyskać w pełni czytelny znak wodny.

Tabela 3: Wyniki badań odporności na ataki dla poszczególnych algorytmów

Typ ataku	IWT	IWT-DCT	IWT-SVD
Przesunięcie bitowe	Brak	Brak	Duża
Obrót	Brak	Brak	Brak
Rozmycie	Brak	Brak	Duża
Dodanie szumu	Brak	Brak	Brak
Usuwanie szumu	Brak	Brak	Brak
Kompresja JPG	Brak	Brak	Brak
Korekcja histogramu	Średnia	Brak	Średnia
Dostrojenie kontrastu	Duża	Brak	Średnia
Korekcja gamma	Niska	Brak	Duża

5. Wnioski

Celem badań było wyłonienie algorytmu znakowania wodnego, który będzie najodpowiedniejszy dla znakowania obrazów medycznych. Na podstawie przeprowadzonych badań nie można wyłonić algorytmu, który byłby najdoskonalszym spośród badanych.

Metoda ukrywania danych w oparciu o IWT charakteryzuje się największą pojemnością steganograficzną, co umożliwi osadzanie dosyć obszernych adnotacji medycznych. Warto jednak mieć na uwadze, że im większa ilość osadzonych danych tym niższa jakość rekonstruowanego obrazu a ta metoda uzyskała najmniejsze wartości miar obiektywnych spośród badanych technik znakowania.

Metoda znakowania wodnego oparta na IWT i DCT charakteryzuje się dosyć małą pojemnością steganograficzną w porównaniu do pozostałych badanych algorytmów oraz brakiem odporności. Definitywny brak odporności na ataki sprawia, że nie ma możliwości odtworzenia osadzonego znaku. Jedyną zaletą jest osadzenie

niewielkiej ilości adnotacji medycznych przy jednoczesnym zachowaniu bardzo dużego podobieństwa obrazu oznakowanego do oryginalnego.

Metoda znakowania wodnego oparta na IWT i SVD uzyskała najlepsze wyniki pod względem badań odporności na ataki. Algorytm cechuje się również dobrą jakością oznakowanego obrazu jak i wystarczającą pojemnością steganograficzną.

Zaobserwowano, że połączenie metody IWT z DCT czy SVD wpływa na zmniejszenie pojemności steganograficznej, jednakże połączenie metod zapewnia nowe właściwości, takie jak zwiększenie jakości rekonstruowanego obrazu lub odporności na ataki. Natomiast nie istnieje algorytm, który jednocześnie charakteryzowałby się dużą pojemnością steganograficzną, dobrej jakości zrekonstruowanym obrazem i odpornością na ataki.

Literatura

[1] G. Kozieł, Zmodyfikowane metody cyfrowego przetwarzania sygnałów dźwiękowych w steganografii komputerowej, Politechnika Lubelska, Lublin 2010.

[2] D. Bogumił, Cyfrowe znaki wodne odporne na kompresję JPEG, Politechnika Warszawska, Instytut Informatyki, wrzesień 2001.

[3] Różnice między steganografią a znakowaniem wodnym, https://absta.pl/zakad-ochrony-informacji.html?page=8, [02.10.2020].

[4] G. Xuan, J. Chen, J. Zhu, Y.Q. Shi, Z. Ni, W. Su, Lossless data hiding based on integer wavelet transform, 10.1109/MMSP.2002.1203308 (2003) 312-315.

[5] A.R. Alotaibi, A.L. Elrefaei: Text-image watermarking based on integer wavelet transform (IWT) and discrete cosine transform (DCT), Applied Computing and Informatics, 2019.

[6] P. Gupta, G. Parmar, Image watermarking using IWT-SVD and its comparative analysis with DWT-SVD, 10.1109/COMPTELIX.2017.8004026 (2017) 527-561.

[7] A. Horé, D. Ziou, Image quality metrics: PSNR vs. SSIM. 10.1109/ICPR.2010.579 (2010) 2366-2369.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 384–389
Received: 25 October 2020
Accepted: 8 November 2020

Performance comparison of web services using Symfony, Spring, and Rails examples

Porównanie wydajności serwisów webowych na przykładzie Symfony, Spring i Rails

Patryk Lubartowicz*, Beata Pańczyk

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

Abstract
The article presents the results of a comparative analysis of web application frameworks for Java, PHP and Ruby. The most popular programming frameworks for each language were used for the research: Spring, Symfony and Ruby on Rails. In each of the frameworks the REST and SOAP web services were prepared and used to measure the request execution time. Measurements were made using Postman and SoapUI tools. The tests results showed that Spring is the fastest way to handle requests.

Keywords: Spring; Symfony; Ruby on Rails; performance

Streszczenie
W artykule przedstawiono rezultaty analizy porównawczej szkieletów aplikacji internetowych dla języków Java, PHP oraz Ruby. Do badań zastosowano, najbardziej popularne dla każdego języka, szkielety programistyczne: Spring, Symfony i Ruby on Rails. W każdym z frameworków przygotowano aplikacje testowe typu REST i SOAP, wykorzystane do testów pomiaru czasu wykonywania żądań. Pomiary wykonano za pomocą narzędzi Postman i SoapUI. Wyniki badań wykazały, że najszybciej żądania obsługiwane są w aplikacjach Spring.

Słowa kluczowe: Spring; Symfony; Ruby on Rails; wydajność

*Corresponding author
Email address: **patryk.lubartowicz@pollub.edu.pl** (P. Lubartowicz)

1. Wstęp

Możliwość uruchomienia tej samej aplikacji z poziomu przeglądarki z dowolnego urządzenia stała się codziennością. Wymiana danych i integracja z zewnętrznymi systemami to obecnie element wszystkich złożonych aplikacji. Aby taka wymiana była możliwa, stosowane są konkretne reguły i formaty komunikacji. Zasady wymiany danych określają protokoły komunikacji, czy odpowiednio usystematyzowane struktury danych. Najbardziej rozpowszechnione jest podejście oparte na SOAP (ang. Simple Object Access Protocol) lub REST (ang. REpresentational State Transfer) [1]. Protokół SOAP umożliwia komunikację pomiędzy aplikacjami w języku XML. Ciągle jest szeroko używany, z uwagi na standaryzację W3C, bezpieczeństwo i prostą kontrolę nad zawartością przekazanych danych. REST jest stylem architektonicznym, który definiuje format przesyłanych danych, jako element standaryzacji protokołu HTTP. Używany jest ze względu na elastyczność, szybkość i prostotę. Nie jest to protokół, ale usługa działająca w oparciu o protokół HTTP.

Dostępność wielu formatów danych zdecydowanie przemawia na korzyść REST. SOAP umożliwia komunikację jedynie z użyciem XML. Ograniczeniem dla usługi REST jest jeden protokół – HTTP, podczas gdy SOAP może działać z protokołami HTTP, SMTP, UDP itp.

Na rynku znajduje się wiele rozwiązań, które usprawniają wytwarzanie usług sieciowych w oparciu o oba wspomniane rozwiązania. Językami najczęściej wybieranymi są tu Java, C#, PHP i Ruby [2]. Każdy z języków posiada narzędzia przyspieszające wytwarzanie aplikacji.

2. Cel i obiekt badań

Celem badań jest analiza wydajności usług sieciowych wykonanych za pomocą trzech popularnych szkieletów programistycznych Spring (Java), Symfony (PHP) i Rails (Ruby) [3].
W artykule podjęto próbę odpowiedzi na pytanie:
W jakim stopniu na czas obsługi żądania wpływa zastosowany język programowania i szkielet programistyczny, w którym usługa została zaimplementowana?

Obiektem badań były trzy aplikacje przygotowane w wybranych frameworkach. Każda aplikacja posiadała dwa typy serwisów: jeden oparty o protokół SOAP [4] i drugi wykorzystujący architekturę REST [5]. Każdy z serwisów został odpowiednio zabezpieczony: protokół SOAP za pomocą rozszerzenia WSS (ang. Web Services Security), a REST przy pomocy JWT (ang. JSON Web Token). Wprowadzone zabezpieczenia miały na celu odwzorowanie rzeczywistych warunków, w których pracują serwisy webowe w Internecie.

3. Przegląd literatury

Po przeanalizowaniu dostępnych źródeł, można wnioskować, że większość prac skupia się na formach wytwarzania serwisów webowych, na dalszym miejscu pozostawiając szybkość działania, która uzależniona jest od obranych metod i stopnia komplikacji wykonywanych zadań.

Artykułem, który najbardziej koncentruje się na szybkości przetwarzania żądań jest *Performacje Evaluation of RESTfull Web Services and SOAP/WSDL Web Services* [6]. Opisuje on badania, które polegały na obliczeniu czasu wykonania żądań za pomocą aplikacji zainstalowanej na smartfonie. Brano tu pod uwagę wielkość oraz czas odpowiedzi z serwera na urządzeniu mobilnym. Wnioski na podstawie przeprowadzonych badań jasno wskazują na przewagę serwisu webowego opartego o REST.

Książka *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services* [7] koncentruje się na opisaniu metod wytwarzania serwisów webowych i podaje rozwiązania znanych problemów. Autor wskazuje kiedy warto korzystać z danego typu serwisu webowego.

Artykuł *Performance comparison of selected web service development technology in web applications* [8] przedstawia problem podobny do przedstawianego w niniejszej pracy. Autor skupia się tam jednak tylko na narzędziach dostosowanych do wytwarzania usług webowych związanych z jednym językiem programowania. Podane tam wnioski również potwierdzają przewagę usługi REST w tworzeniu serwisów webowych.

Praca *Performance and usage comparison between REST and SOAP web services [9]* przedstawia proste badanie wykonane w oparciu o zebranie czasów żądań z API dostarczanego przez Sitatech. Uruchomiony lokalnie serwis webowy był odpytywany przez wcześniej przygotowaną aplikację napisaną w C#, która służyła do zbierania danych o czasie wykonania polecenia. Zliczała ona czas od momentu tworzenia żądania do otrzymania odpowiedzi, bez jej obróbki. Analiza wyników wykonana przez autora pracy wykazała, że wykorzystanie architektury REST jest wydajniejsze w porównaniu do SOAP i zaleca się jej wykorzystanie.

W pracy *Comparing Performance of Web Service Interaction Styles: SOAP vs. REST* [10] skupiono się na zbadaniu wydajności serwisu webowego z uwzględnieniem napływu wielu zapytań jednocześnie. Pod uwagę wzięto czas odpowiedzi żądania oraz przepustowość dla sieci połączonej kablem i bezprzewodowo. Utworzone zostały dwa oddzielne serwisy – REST i SOAP – pobierające dane z tej samej przygotowanej bazy. Dodatkowo utworzono program kliencki komunikujący się obydwoma sposobami, który służył do zbierania czasów z wykonanych żądań. Czasy zliczane były od momentu wywołania metody wykonującej żądanie, do czasu przetworzenia odebranych danych. Wykonana przez autora analiza wyników podkreśla przewagę REST dla szybkości wykonywania żądań i łatwość jego przetwarzania, oraz ma większą przepustowość przesyłania danych.

Artykuł *Analiza możliwości współpracy aplikacji mobilnych z usługami sieciowymi typu REST i Web Service* [11] skupia się na porównaniu szybkości działania bibliotek dla aplikacji mobilnych z systemem Android. Obsługuje ona komunikację z serwisem webowym napisanym przez autora w języku Java, który obsługuje dwa typy komunikacji: REST i SOAP. Przeprowadzone badanie koncentruje się na czasie pracy samych bibliotek. Czas ten obliczono poprzez odjęcie czasu całkowitego wykonania żądania od sumy czasu odpowiedzi usługi oraz czasu odpowiedzi sieci. Wnioski wyciągnięte przez autora skupiają się na wyliczonym przez niego parametrze, jednak dane na temat czasu przetwarzania żądania po przechwyceniu go w ruchu sieciowym przedstawiają też istotne informacje.

4. Metoda badań

Badania zostały przeprowadzone poprzez wykonywanie 100 powtórzeń dla czterech scenariuszy testowych przedstawionych w tabeli 1. Jednakowe scenariusze wykonano dla metody wymiany danych SOAP i REST.

Tabela 1: Scenariusze testowe

Oznaczenie	Scenariusz
1	Pobranie pojedynczego obiektu reprezentującgo dane przetworzone w zapytaniu
2	Pobranie tablicy zawierającej zbiór 100 obiektów
3	Przesłanie obiektu do zapisu
4	Przesłanie obiektu do aktualizacji

Pojedynczą pobieraną w żądaniu daną jest wartość rocznego obrotu. Otrzymano ją z zapytania sumującego wartości faktur z kolumny *Total* w tabeli *Invoice* w podanym w żądaniu roku. Obiektem przesyłanym w żądaniach do zapisu i aktualizacji była encja *Track* o strukturze pokazanej na Rysunku 1.

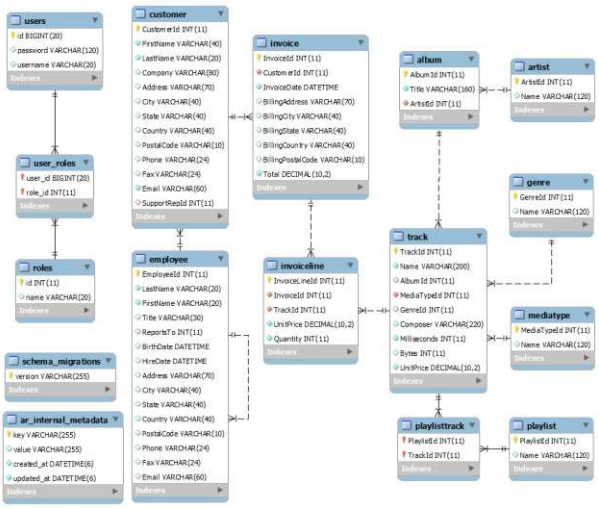
Czasy wykonania 100 żądań dla każdego ze scenariuszy mierzono programami SoapUI dla SOAP i Postman dla REST. Prezentowane wyniki uśredniono (średnia arytmetyczna). Ilość zajmowanej pamięci operacyjnej mierzono za pomocą monitora zasobów systemu Windows. Każda z aplikacji była uruchamiana oddzielnie. Badania prowadzone były na maszynie o następujących parametrach:

- płyta główna: M5A97 R2.0,
- procesor: AMD FX™-6300 4.21GHz,
- RAM: 16GB DDR3 1866MHz,
- system: Windows 10 x64 Pro.

5. Aplikacje testowe

Każda z aplikacji testuje po 4 żądania dla dwóch serwisów webowych: REST i SOAP. Wykonywane są one po uprzedniej autentykacji użytkownika za pomocą tokena JWT (ang. JSON Web Token) w przypadku REST, lub danych logowania przesłanych w sposób jawny w nagłówku WSS (ang. Web Socket Security) według specyfikacji opublikowanych przez OASIS dla SOAP. Aplikacje wykorzystują dane z pojedynczej kopii bazy danych Chinook [13], która jest dostępna na

licencji MIT i reprezentuje sklep z utworami muzycznymi w wersji cyfrowej (Rysunek 1). Na potrzeby projektu dodane zostały do niej tabele wymagane w procesie autentykacji użytkownika i inne niezbędne do pracy z danym frameworkiem.



Rysunek 1: Model bazy danych

5.1. Aplikacja Spring

Aplikacja wykonana została w języku Java v8. Podstawą projektu była paczka Spring Boot w wersji 2.2.4. Dodatkowo wykorzystano wsparcie bibliotek WSDL4J i JAXB2 oraz JJWT. Pierwsze dwie biblioteki wspomagają proces wytwarzania serwisu webowego w oparciu o protokół SOAP, natomiast trzecia służy implementacji zabezpieczeń dla serwisu REST. Aplikacja była uruchamiana z wiersza poleceń ze skompilowanego pliku .jar. Przykładowe fragmenty kodu do realizacji scenariusza 1 dla tej aplikacji w technologii REST i SOAP przedstawiają listingi 1 i 2.

Listing 1. Pobranie pojedynczej danej - REST w Spring

```
Controllers/InvoiceController.java:
@GetMapping("/total-sales-in-year")
@PreAuthorize("hasRole('USER')
 or hasRole('MODERATOR') or hasRole('ADMIN')")
@ResponseBody
public Float getYearSales(
@RequestParam
(value = "year", required = true)
Integer year
) { return invoicesService.getTotalSalesPerYear(year); }
```

Listing 2. Pobranie pojedynczej danej - SOAP w Spring

```
Controllers/Endpoints/InvoicEndpoint.java:
@PayloadRoot(namespace = NAMESPACE_URI, localPart =
"getYearSalesRequest")
@ResponsePayload public GetYearSalesResponse getYearSale(
@RequestPayload GetYearSalesRequest request
){
GetYearSalesResponse response =
new GetYearSalesResponse();
  Float amount =
invoicesService.getTotalSalesPerYear(
request.getYear().intValue()
);
  response.setAmount(amount);
  return response;
}
```

5.2. Aplikacja Symfony

Aplikacja napisana została w PHP v7.2. Projekt opierał się o szkielet projektu Symfony w wersji 5.0.8. W tym przypadku brakuje bibliotek wspomagających tworzenie serwisów webowych SOAP oraz REST. Jedynym dodatkowym pakietem był LexikJWTAuthenticationBundle, wykorzystany do zabezpieczenia wymiany danych metodą REST. Testy wykonywane były z serwera z wersją PHP 7.2.30 x64. Przykładowe fragmenty kodu do realizacji scenariusza 1 dla tej aplikacji przedstawiają listingi 3 i 4.

Listing 3. Pobranie pojedynczej danej - REST w Symfony

```
src/Controller/InvoiceController.php:
/**
   * @param InvoiceRepository $repository
   * @param $year
   * @return JsonResponse
 *                                    @Route("/sales/{year}",
name="invoice_year_sales_get",
methods={"GET"})
   */
public function getTotalSalesPerYear(InvoiceRepository $re-
pository, $year){
$amount = $repository ->
getTotalYearSales($year);
if (!$amount){
$data = [
'status' => 404,
'errors' => "No amounts in this year",
];
return $this->response($data, 404);
}
return new JsonResponse(
$amount,
Response::HTTP_OK
);
}
```

Listing 4. Pobranie pojedynczej danej – SOAP w Symfony

```
src/Service/Soap/TestService.php:
public function getYearSales($year){
     return $this->invoiceRepository
->getTotalYearSales($year);
}
```

5.3. Aplikacja Rails

Ta aplikacja utworzona została w języku Ruby v2.6. Wykorzystano szkielet projektu podstawowej aplikacji Ruby on Rails w wersji 6.0.3.1.

Do wsparcia implementacji dodano biblioteki: JWT, Wash Out oraz Nokogiri. Pierwsza służy zabezpieczeniu komunikacji poprzez REST, dwie następne wspierają wytwarzanie serwisu webowego korzystając z protokołu SOAP. Aplikacja była uruchamiana za pomocą komendy z wiersza poleceń.

Przykładowe fragmenty kodu do realizacji scenariusza 1 dla tej aplikacji w technologii REST i SOAP przedstawiają listingi 5 i 6.

Listing 5. Pobranie pojedynczej danej - REST w Ruby on Rails

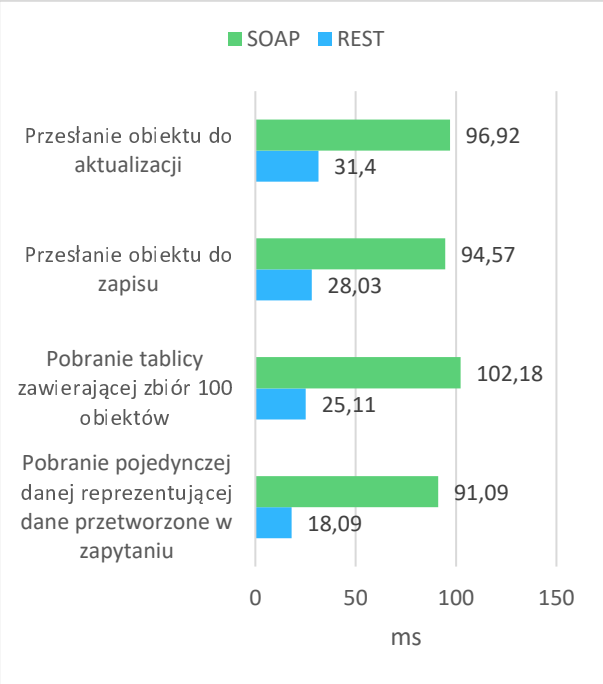
```
app/controllers/invoices_controller.rb:
def totalSalesInYear
  sales =
          Invoice.getYearSales(params[:year])[0]
  render json: sales["amount"].to_f
end
```

Listing 6. Pobranie pojedynczej danej - SOAP w Ruby on Rails

```
app/controllers/soap_controller.rb:
soap_action "getYearSales",
  :args => {:year => :integer},
  :return => :double
def getYearSales
  sales = Invoice.getYearSales(params[:year])[0]
  raise SOAPError, "No sales in this year"
          if sales["amount"] == nil
  render :soap => sales["amount"].to_f
end
```

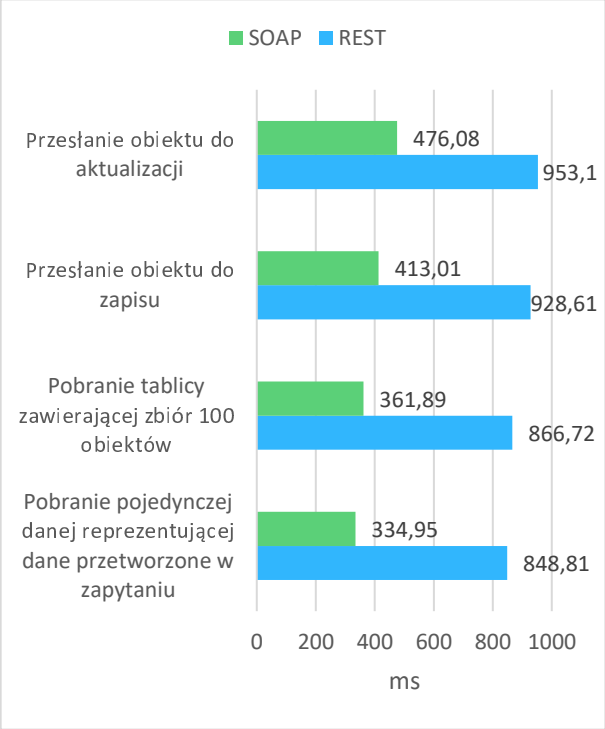
6. Wyniki badań

Na rysunku 2 przedstawiono czasy żądań dla obu serwisów webowych w aplikacji Spring. Widoczna jest duża różnica pomiędzy czasem wykonania żądań w przypadku REST i SOAP - dla SOAP czasy są nawet 3-4 razy dłuższe. Pomimo różnic, oba wyniki są akceptowalne i nie przekraczają 100ms (poza jednym wyjątkiem).



Rysunek 2: Zestawienie średnich czasów wykonania 100 żądań w aplikacji Spring dla REST i SOAP.

Rysunek 3 przedstawia porównanie czasów obsługi żądań dla aplikacji Symfony. W tym przypadku to protokół SOAP ma lepsze rezultaty w odniesieniu do czasu wykonywania żądań. Różnica pomiędzy czasami wynosi 2-krotność w przypadku przesłania obiektu do aktualizacji, w przypadku jego zapisu jest to wartość 2,25. Pobranie tablicy zawierającej 100 obiektów, 2,4 razy szybciej realizuje SOAP, a dla pojedynczej danej wykonuje to aż 2,54 razy szybciej niż REST. Może to być spowodowane brakiem wsparcia bibliotek ze względu na początkowe stadium rozwoju najnowszej wersji frameworka.

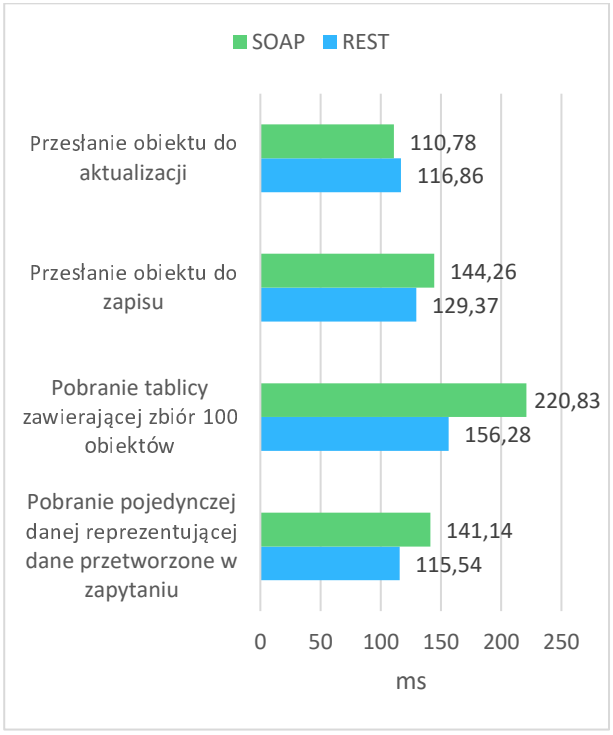


Rysunek 3: Zestawienie średnich czasów wykonania 100 żądań w aplikacji Symfony dla REST i SOAP.

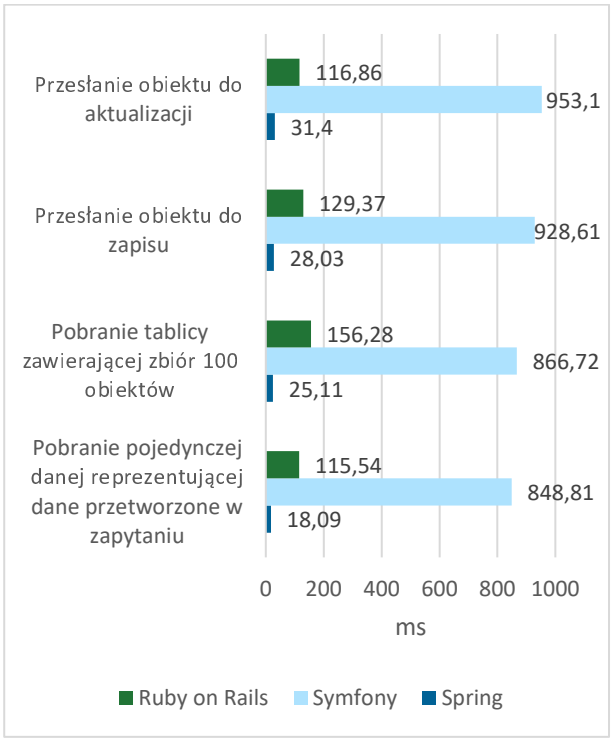
Rysunek 4 pokazuje średnie czasy dla obu serwisów webowych w Ruby on Rails. W tym przypadku czasy żądań (oprócz pobrania 100 obiektów) są zbliżone, ale generalnie wykorzystanie REST daje lepsze rezultaty. Należy zwrócić też uwagę na czas aktualizacji obiektu, który w przypadku REST prawie się nie różni od pobierania pojedynczej danej, a w przypadku SOAP jest znacznie mniejszy. Dzieje się tak z powodu zachowania w aplikacji informacji o wykorzystaniu danego rekordu. Wykonanie po raz kolejny tego samego zapytania o dany rekord, powoduje, że dane nie są pobierane ponownie, a odczytywane z wcześniej przygotowanych ciasteczek.

Rysunki 5 i 6 przedstawiają porównanie czasów wykonania żądań dla każdego scenariusza dla wszystkich aplikacji, odpowiednio dla REST i SOAP.

W przypadku REST wykresy pokazują wyraźną przewagę czasową dla aplikacji Spring. Czas pomiędzy najszybszym, a najwolniejszym wykonaniem żądań jest nawet 50 razy krótszy w porównaniu do aplikacji Symfony. W przypadku Rails wyniki są akceptowalne, na poziomie 115-156ms, odbiegając od rywala od około 4 do 6-krotności różnicy czasu.

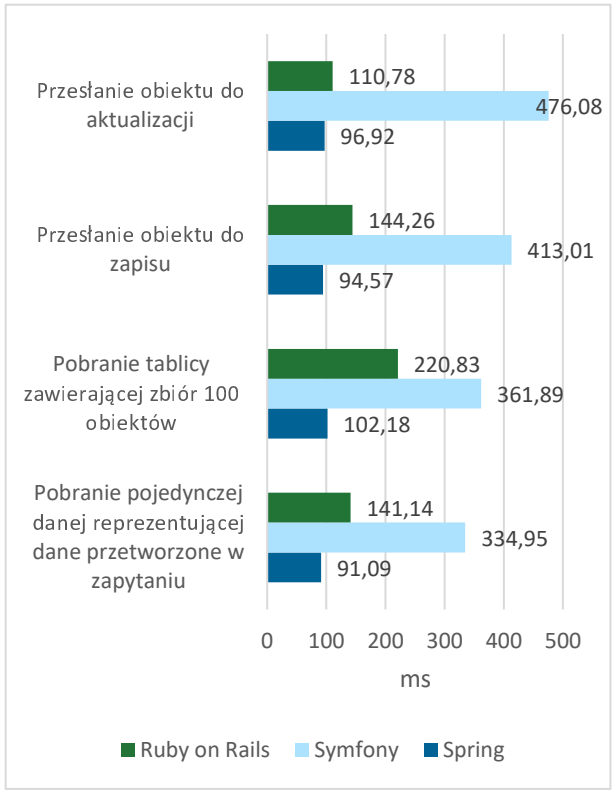


Rysunek 4: Zestawienie średnich czasów wykonania 100 żądań w aplikacji Ruby on Rails dla REST i SOAP.



Rysunek 5: Zestawienie wszystkich scenariuszy testowych we wszystkich badanych aplikacjach dla REST.

W porównaniu do różnic widocznych w technologii REST, w przypadku SOAP różnice nie są już tak bardzo duże. Najniższy czas jest około 5 razy mniejszy od najdłuższego średniego czasu wykonania żądania. Warto zwrócić szczególną uwagę na niski czas wykonania operacji przesłania obiektu do aktualizacji w przypadku Ruby on Rails. Jest on porównywalny z czasem dla aplikacji Spring.



Rysunek 6: Zestawienie wszystkich scenariuszy testowych we wszystkich badanych aplikacjach dla SOAP

Warto też zwrócić uwagę na ilość pamięci zajmowanej przez programy po wykonaniu testów (tabela 2).

Tabela 2: Ilość zajmowanej pamięci operacyjnej przez aplikacje

Aplikacja	Ilość zajmowanej pamięci
Spring	778 160 KB
Symfony	18 844 KB
Ruby on Rails	84 464 KB

Najmniej pamięci zajmuje aplikacja Symfony (niewiele ponad 18MB) co jest bardzo dobrym wynikiem, biorąc pod uwagę ilość zasobów w nowoczesnych serwerach. Podstawowa konfiguracja serwera PHP może zawierać limity dotyczące wykorzystywanej pamięci, choć w tym przypadku wynik nie osiągnął bariery aktualnie ustawianego 128MB. Drugie miejsce zajmuje Rails (około 84MB) a najgorzej wypada aplikacja Spring (aż 778MB, czyli około 43-razy więcej niż w przypadku Symfony). Wybór aplikacji Spring, pomimo najszybszej obsługi żądań, wymaga jednak rozważenia pod kątem dopasowania do posiadanych zasobów sprzętowych.

7. Wnioski

Wykonana analiza wykazała, że wydajność usług sieciowych, na przykładzie REST i SOAP, zależy od wybranego języka programowania i szkieletu programistycznego.

Wykorzystanie aplikacji Spring do utworzenia serwisu webowego pozwala uzyskać największą przepu-

stowość wymiany danych. Niestety, w tym przypadku trzeba liczyć się z większym zapotrzebowaniem na zasoby sprzętowe, potrzebne do uruchomienia i utrzymania serwisu.

W przypadku, gdy zasoby sprzętowe są ograniczone, a trochę dłuższy czas odpowiedzi jest akceptowalny – to zastosowanie Ruby on Rail jest dobrym wyborem. Aplikacja w Ruby potrzebuje 9 razy mniej pamięci operacyjnej w porównaniu do Spring.

Wybór najbardziej optymalnego rozwiązania wiąże się też z wyborem odpowiedniego serwisu webowego. W analizowanych przypadkach, rozwiązanie oparte o REST dało lepsze wyniki. Wynik jest zgodny z oczekiwanym. W publikacji [12] uzyskano analogiczne rezultaty, jednak bez analizy aplikacji Symfony.

Nie zawsze na pierwszym miejscu stawia się szybkość działania, a wykorzystanie SOAP jako ustandaryzowanego rozwiązania nie oznacza, że system będzie dużo wolniejszy, co widać na przykładzie Ruby on Rails.

Literatura

[1] T. Zientarski, M. Miłosz, M. Kamiński, M. Kołodziej: Applicability analysis of REST and SOAP web services, Informatyka, Automatyka, Pomiary W Gospodarce i Ochronie Środowiska, 7 (2017) 28-31.

[2] S. Cass, The 2017 top programming languages, IEEE Spectrum 2018.

[3] Statystyki dotyczące wykonywanych zawodów i wybieranych technologii oraz narzędzi StackOverflow, https://insights.stackoverflow.com/survey/2020, [15.09.2020].

[4] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S.R. Thatte, D. Winer, Simple object access protocol (SOAP) 1.1., W3C Note, 2000.

[5] M. Masse, REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces, O'Reilly Media, Inc. 2011.

[6] R. Digvijaysinh, PERFORMANCE EVALUATION OF RESTFUL WEB SERVICES AND SOAP / WSDL WEB SERVICES, International Journal of Advanced Research in Computer Science, 8 (2017) 415-420.

[7] R. Daigneau, Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services, Addison-Wesley, 2012.

[8] A. Gdula, M. Plechawska-Wójcik, Performance comparison of selected web service development technology in web applications, Journal of Computer Sciences Institute, 1 (2016) 14-19.

[9] J. Makkonen, Performance and usage comparison between REST and SOAP web services, 2017

[10] P. K. Potti, S. Ahuja, K. Umapathy, Z. Prodanoff, Comparing Performance of Web Service Interaction Styles: SOAP vs. REST, Proceedings of the conference on information systems applied research, (2012) 2167.

[11] M. R. Daraż, P. Kopniak, Analysis of the possibilities of cooperation of mobile applications with network services of the type REST and Web Service, Journal of Computer Sciences Institute, 11 (2019) 155-162.

[12] A. Soni, V. Ranga, API Features Individualizing of Web Services: REST and SOAP, International Journal of Innovative Technology and Exploring Engineering, 8 (2019) 664-671.

[13] Baza danych, https://github.com/lerocha/chinook-database, [03.10.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 390–395
Received: 24 October 2020
Accepted:9 November 2020

Modeling of covid-19 cases of selected states in Nigeria using linear and non-linear prediction models

Modelowanie przypadków COVID-19 w wybranych stanach Nigerii przy użyciu liniowych i nieliniowych modeli predykcyjnych

Babatunde Abdulrauph Olarenwaju [a], Igboeli Uchenna Harrison[b,*]

[a] *Department of Computer Sciences, University of Ilorin, Nigeria*
[b] *Department of Computer Science, University of Abuja, Nigeria*

Abstract
COVID-19 has stamped an indelible mark in the history of humanity as one of the recorded deadly virus that has wiped out millions of lives on planet earth many whose exact cause of death cannot be account for due to lack of knowledge. It has become a household name in every nook and cranny from developed to the underdeveloped nations of the world. Most of the prominent signs of COVID-19 like fever, cough, difficulty in breathing and accessional muscle pain can also resemble those of many other notable diseases thereby making it highly necessary to undergo a diagnostic test to be able to categorically identify COVID-19 patients. The use of medical diagnostic tests can also help determine patients who have recovered from COVID-19. Various studies abound with researchers trying to predict and even forecast the level of damage and disruption of economic activities this may have brought to almost every nation of the world. This research attempts to find out the nature of the spread of the virus using Autoregressive Integrated Moving Average (ARIMA) and Artificial Neural Networks (ANN). The essence is to ascertain the exact model to use in forecasting the future occurrence of the pandemic especially at this stage where the second wave of the pandemic is in view. The study found that both linear and nonlinear predictions models can fit the trend of the virus in Nigeria with ARIMA producing results of over 97% on a 120-day period while ANN produced results of about 98.01% in some states. We conclude that future waves of the virus in addition to other epidemics of this nature can be predicted with high degree of accuracy with ARIMA or ANN.

Keywords: ARIMA; ANN; Prediction; Pandemic

*Corresponding author
Email address: **uchenna.igboeli@uniabuja.edu.ng** (I. U. Harrison)

1. Introduction

Coronavirus 2019 is an infectious and transmittable viral disease caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), which emerged in Wuhan, China [1]. The epidemic which has negatively impacted almost every nation of the world, began in Guangdong province, China, in the late 2002. It spread to Hong Kong on February 21, 2003 [2], and from there to what has been described as a pandemic today. The rate at which the disease is spreading is alarming as it has being reported in almost every continent from Asia to America, Australia, Europe and Africa. With the spread of the dreaded virus into various parts of the world, the Nigeria government took some pro-active measures to prevent it from getting into the country. This mainly took the form of taking steps to enforce strict security checks at the airport as early as in January 2020. These measures yielded insufficient results as the country recorded her index case late February from a victim that imported the virus from Europe. The announcement of the presence of the virus in Nigeria immediately brought great fear and concern to all Nigerians as many began to doubt the effectiveness of the surveillance operations deployed in our airports and the country's general preparedness as the dreaded virus continued to spread around Europe, Asia and United States in an astronomical rate. The index case of the virus here in Nigeria, had prior to testing positive, visited some other states of the apart from Lagos state. This is a proof of the inadequate level of readiness to combat the virus especially considering the level of publicity given to the pandemia worldwide. Just after the index case was detected, the NCDC launched a National Emergency Operations Centre (EOC) to oversee the national response to COVID-19. As a follow-up, the Presidential Task Force (PTF) for coronavirus control was also inaugurated on March 9, 2020 [3].

As at August 13, 2020, Nigeria has recorded about 47,743 confirmed cases of COVID-19 with 12,844 active and 33,943 cases already discharged while having about 956 deaths. This result is coming out of a total of 338,084 samples tested throughout the country [4]. The country has a total of 62 NCDC COVID-19 Laboratories scattered in all the states of the federation as at 13th of August 2020 for a population of about 200,000,000. Andres et. al., (2020) found that there exists a relationship between COVID-19 behavior and population in the region being considered. One can now argue that the few number of cases recorded in Nigeria may not adequately reflect the true position of the spread of COVID-19. Jester, Uyeki & Jernigan (2018)

[5] traced the outbreak of epidemic from 1918, 1957, 1968 and 2009 till date and noted that all the four pandemics in last 100 years have had some genes that originated from avian influenza viruses. These viruses are constantly changing and therefore require ongoing surveillance and frequent vaccine virus changes. Improvements in healthcare, scientific researches, vaccines and effective communications have however improved pandemic response. There exists various works and studies on the prediction of cases and mortality rates related to the COVID-19 dreaded disease. This research is aimed at examining the effectiveness of both the ANN and ARIMA models in predicting the rate of spread of the virus in Nigeria. Its major significance is to compare a linear (ARIMA) and a non-linear method (ANN) of time series forecasting when applied to the confirmed cases of covid-19 in Nigeria as it will help the concerned agencies in tackling the prevalence and future nature of the disease. Oladipo & Babatunde (2016) [6] used Decision Tree, Logistic Regression and ANN for the prediction of diabetes among two sets of people from china and arrived at the conclusion that machine learning algorithms can efficiently predict such an ailment that is regarded as among the top five causes of death found in the United States [7].

2. Methodology

In this study, daily COVID-19 situation reports as presented by the Nigeria Centre for Disease Control (https://covid19.ncdc.gov. ng/) was used. The data covers a sample of 10 states for the period of 120days between April 2020 and July, 2020. The data is normalized and for the ANN, a unipolar sigmoid function is applied to produce values between 1 and 0. For the states selected, only the observed cases, number of patients already discharged and number of deaths were used as input variables to the neural network, while the output is a single attribute representing the predicted number of cases. The implementation of the simulation was carried out using the R programming language with the predicted values written to various spreadsheets. These worksheets were exported to C# interface where the relative error values where computed and the output written to MySQL data tables.

2.1 General Form Of ARIMA(P,D,Q)

An AR(p) model is a discrete time linear equation with noise of the form:

$$X_t = \alpha_1 X_{t-1} + \ldots\ldots + \alpha_p X_{t-p} + \varepsilon_t \quad (1)$$

Here, p is the order, $\alpha_1 \ldots \alpha_p$ are the parameters or coefficients (real numbers), ε_t is an error term which is usually a white noise.

Where p = 1 then AR(1) becomes:

$$X_t = \alpha X_{t-1} + \varepsilon_t \quad (2)$$

With Abs(α) < 1 and $Var(X_t) = \frac{\alpha^2}{1-\alpha^2}$, it is a wide sense stationary process.

Since AR is a time series model, when we introduce the time lag operator (L) the AR becomes $Lx_t = x_{t-1}$, for all t Є Z (set of real numbers). Since the time lag operator is a linear operator, the powers, positive or negative can be denoted as:

$$L^k L^k x_t = x_{t-k}, \text{for all } t \in Z$$

With this lag operation, the AR model becomes:

$X_t = \alpha X_{t-1} + \varepsilon_t$

$X_t - \alpha X_{t-1} = \varepsilon_t$

The MA(q) model in ARIMA which denotes the moving average with orders (p and q) is an explicit formula for X_t in terms of noise of the form

$$X_t = c + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots .. + \beta_q \varepsilon_{t-q} \quad (3)$$

Where:

c = *a contant from the regression modelled*

ε_t = *the white noise in the past forcast errors*

$\beta_1 \varepsilon_{t-1}$ = *error in the AR values of the prediction*

$\beta_q \varepsilon_{t-q}$ = *last lag error in the AR used in the prediction*

Combining the AR and the MA, we have:

$$X_t = c + \alpha_1 X_{t-1} + .. + \alpha_p X_{t-p} + \beta_1 \varepsilon_{t-1} + .. + \beta_p \varepsilon_{t-p} + \varepsilon_t \quad (4)$$

This becomes the general form of ARIMA model as a discrete time linear equation.

2.2 Data Preparation for ARIMA

To process the data we used ARIMA(p,d,q) model. The steps involved in the use of ARIMA model are identification, parameter estimation and forecasting. These steps were repeated continually until an appropriate model is identified. The model to be selected is chosen based on the Akaike Information Criterion (AIC) [8] which is the model with the least AIC value. Based on the dataset, the data can also be made stationary by differencing before the training and forecasting.

2.3 Data Preparation for ANN

The general equation model for ANN is given as:

$$y_t = w_0 + \sum_{j=1}^{q} w_j f\left(w_{0,j} + \sum_{i=1}^{p} w_{i,j} y_{t-i}\right) + \varepsilon_t \quad (5)$$

Where:

y_t is the predicted value,
w_j are the output layer weights,
$w_{i,j}$ are input layer weights,
f is a transfer function,
q is the number of hidden nodes,
p is the number of input nodes,
ε_t is a random error at time t.

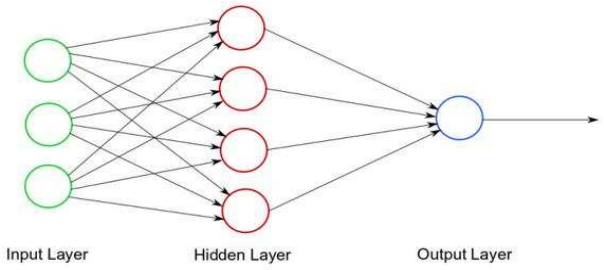


Figure 1: Overview of ANN (https://learn.g2.com/ artificial-neural-network)

Out of the 120 days data collected for the selected states, we adopted 70% for training the algorithm, while the remaining 30% is used for testing. The 70% mapped out for training is further divided into two. While 40% is reserved for actual training, 30% is used for cross-validation. Before uploading the data into the ANN, the data is normalized. The need to normalize the data arises from the fact that the features to be used as input do not have a uniform scale. This experiment uses the min-max normalization function.

The unipolar sigmoid function is adopted and is defined as:

$$y = f(x) = \frac{1}{1 + e^{-\lambda x}} \quad (6)$$

It was used to normalize input vectors of the training data prior to processing and returns values between 0 and 1.

2.4 Validation and evaluation

The daily time series reports of confirmed cases of COVID19 from April 1, 2020 to July 30, 2020, were extracted from Nigeria Centre for Disease Control website and analyzed for the purpose of the research. This sample is made up of states that had confirmed cases from April after the index case was announced in March. For the ARIMA prediction, the times-series data for all the states were checked for stationarity using the Augmented Dickey Fuller adf.test() function. Each state data is differenced accordingly to arrive at a stationary set with differentials. It is evident that while some states became stationary at differences=1, others attained stationarity at difference = 2 while none was stationary at the initial stage. Ensuring that the data became stationary became necessary as a precondition for the use of use of ARIMA since the data must not have any form of trend.

Shown in Table 1 are the various models for the individual states along with their error values. The data series from the states were tested for ARIMA (0,1,1), ARIMA (2,1,1), ARIMA (1,2,1), ARIMA (2,2,1), ARIMA (0,1,2), ARIMA (0,2,2) and, ARIMA (0,2,1) models. For each model, the AIC value is computed while the model which produces the lowest AIC value is selected as the optimum model which was finally used for the prediction.

Table 1: ARIMA Models and their errors

States					
Model	**Edo**	**River**	**Lagos**	**Osun**	**Ogun**
0,1,1	1034.20	1039.70	1294.47	790.64	966.74
2,1,1	1028.16	1039.30	1292.90	792.75	965.98
1,2,1	1028.77	1037.30	1290.90	791.10	968.28
2,2,1	1028.16	1039.30	1292.90	792.75	965.96
0,1,2	1027.20	1037.39	1291.23	790.87	968.44
0,2,2	1027.19	1037.39	1291.23	790.87	968.44
0,2,1	1034.20	1039.70	1294.47	790.62	966.74
States					
Model	**Ekiti**	**Kaduna**	**FCT**	**Oyo**	**Enugu**
0,1,1	599.07	882.88	1083.40	1105.40	930.04
2,1,1	601.18	881.94	1085.80	1108.10	931.22
1,2,1	599.19	884.74	1085.40	1106.30	930.21
2,2,1	601.18	881.97	1085.80	1108.10	931.22
0,1,2	599.23	884.63	1085.40	1106.20	929.81
0,2,2	599.23	884.63	1085.40	1106.20	929.83
0,2,1	599.05	882.88	1083.40	1105.70	930.04

Table 2: ANN Models for various periods

States	**120 days**			
	3:2	**3:1**	**4:1**	**4:2**
River	0.0038	0.0055	0.0054	0.0057
Enugu	0.0025	0.0052	0.0053	0.0047
Ogun	0.0029	0.0025	0.013	0.0018
Kaduna	0.0036	0.0027	0.0072	0.0017
Lagos	0.0015	0.0012	0.0039	0.0038
FCT	0.0024	0.0038	0.0012	0.0038
Oyo	0.0024	0.0064	0.0047	0.0015
Osun	0.0068	0.0034	0.0039	0.0007
Ekiti	0.0027	0.0061	0.0016	0.0014
Edo	0.0026	0.0022	0.003	0.0029
States	**90 days**			
	3:2	**3:1**	**4:1**	**4:2**
River	0.0044	0.0036	0.0012	0.0023
Enugu	0.005	0.0055	0.0092	0.0012
Ogun	0.0125	0.0041	0.0008	0.0035
Kaduna	0.0023	0.0039	0.0036	0.0073
Lagos	0.0048	0.006	0.0189	0.0037
FCT	0.0069	0.0043	0.0104	0.0022
Oyo	0.0031	0.0021	0.0014	0.0009
Osun	0.0055	0.0127	0.0088	0.0054
Ekiti	0.0087	0.0088	0.0051	0.0042
Edo	0.0072	0.0103	0.0123	0.0058
States	**60 days**			
	3:2	**3:1**	**4:1**	**4:2**
River	0.0027	0.0054	0.0233	0.0010
Enugu	0.0058	0.0074	0.0149	0.0116
Ogun	0.0049	0.0100	0.0061	0.0030
Kaduna	0.0104	0.0043	0.0056	0.0020
Lagos	0.0036	0.0035	0.0011	0.0009
FCT	0.0067	0.0076	0.0217	0.0072
Oyo	0.0008	0.0045	0.0025	0.0013
Osun	0.0089	0.0239	0.0384	0.0099
Ekiti	0.0064	0.0080	0.0089	0.0044
Edo	0.0137	0.0010	0.0026	0.0079
States	**30 days**			
	3:2	**3:1**	**4:1**	**4:2**
River	0.0106	0.0081	0.0111	0.0078
Enugu	0.0010	0.4778	0.4534	0.4742

Ogun	0.0025	0.0081	0.0094	0.0017
Kaduna	0.0031	0.0064	0.0153	0.0021
Lagos	0.0051	0.0030	0.0047	0.0027
FCT	0.0060	0.0063	0.0061	0.0014
Oyo	0.0084	0.0202	0.0308	0.0095
Osun	0.0027	0.0010	0.0174	0.0107
Ekiti	0.0200	0.0167	0.0170	0.0146
Edo	0.0037	0.0063	0.0042	0.0045

Based on the ANN models (3,1), (3,2), (4,1), (4,2) and their errors as shown on Table 2, the model with the minimum error is selected for prediction. ANN(4:2) showed dominance over other models however. The unipolar sigmoid activation function was used for the mapping while min-max normalization was used to normalize the data.

The prediction performance is evaluated using various error valuation techniques. These include the Mean Absolute Percentage Error (MAPE) defined as follows:

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{(A_t - F_t)}{A_t}\right| \quad (7)$$

Where:

A_t = *Actual Value*

F_t=*Forcasted Value*

T=*Number of times Summations.*

3. Results

The charts in Figure 2 shows the plotting of the FCT & Lagos State 60-Days plots of the prediction errors of both ARIMA and ANN.

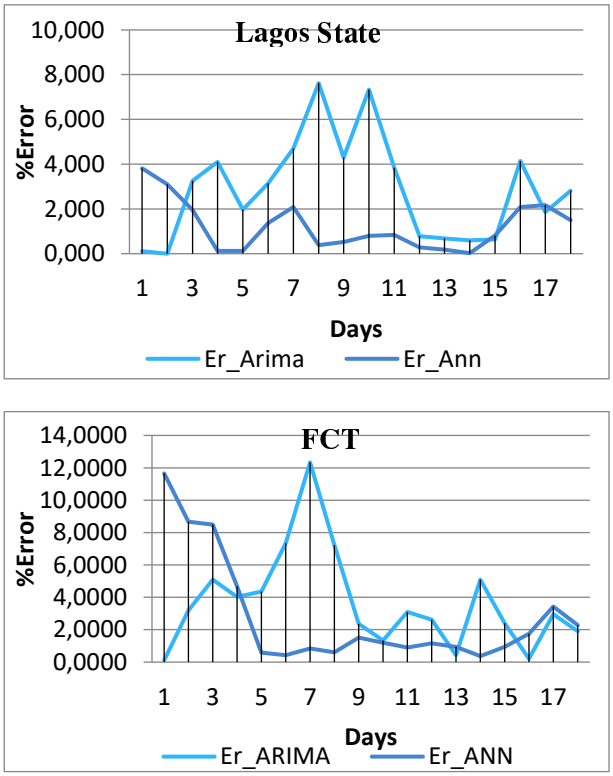


Figure 2: 60-Days Error Plots for ARIMA and ANN for FCT and Lagos States

For easy comparison, the 30% data used for the testing of ANN were also extracted from the predictions of ARIMA via the use of a C# programming module. This led to the presentation of 18-periods rather than the entire 30-days duration. The charts on figure 2 are the FCT and Lagos State 60-days error plots for both ARIMA and ANN. Presented in Table 3 is the summary of ARIMA versus ANN error values for all states modeled.

4. Discussion

For the prediction using ANN, considering each state and prediction period, the data was split into training and testing datasets consisting of 70% and 30% respectively. The input is made up of the number of patients on admission and the number already discharged.

Table 3: Summary of ARIMA & ANN error values for all states modeled

Days	State	Er_ARIMA	Er_ANN
30	Lagos	5.5082	2.3249
60		3.8885	2.9689
90		2.6091	3.6301
120		2.0348	9.6285
30	FCT	2.8343	5.6592
60		3.0526	1.3037
90		3.7146	2.6181
120		2.9871	1.9868
30	Kaduna	6.9396	13.9458
60		9.2587	16.5799
90		4.5329	2.5237
120		4.3819	3.9552
30	Enugu	0	4.5
60		10.7065	2.383
90		8.465	6.9796
120		5.8304	4.2215
30	Ogun	9.0758	15.8903
60		6.493	8.5536
90		6.1392	4.0318
120		5.0028	23.5906
30	Osun	0	25.4033
60		4.1779	1.8558
90		2.309	1.272
120		3.5205	3.6437
30	River	3.9375	20.6042
60		7.7425	14.2968
90		7.4306	30.2168
120		4.5729	27.7891
30	Oyo	6.9384	5.8372
60		3.7193	9.7848
90		5.4633	2.5101
120		2.8541	4.8947
30	Edo	5.8698	10.2801
60		7.2401	4.623
90		3.9818	5.1882
120		3.8242	7.2988
30	Ekiti	8.3333	6.8542
60		11.3526	7.999
90		13.0728	5.2975
120		5.1626	5.5611

From all the series collected it is evident that there is an upward trend in the number of confirmed cases of

COVID19 in Nigeria despite the fact that several safety measures have been put in place by the government. Lagos state has been on top of the list having recorded a total of over 19,600 cases followed by FCT with 5,758 cases. Comparing these figures with states like Zamfara, Yobe and Kogi with 79, 76 and 5 cases, one can easily be convinced that lack of enough testing facilities and experts must have influenced the low figures. Since different ARIMA(p,d,q) values were used for individual states depending on the nature of their data, it is difficult to state a universal model that fits all the states. The two states with the highest number of cases i.e Lagos State and FCT however performed incredibly well with ARIMA(1,2,1) and ARIMA(0,1,1) models respectively. On the 120-day period, the two models posted an accuracy level of 97.96% and 97.01% with is in line with the results obtained in the study by [9-11. Though no particular ANN model produced the optimal error value across all the data modeled, it is seen that ANN(4:2) was predominant over other ANN models tested. On the 120-days period tested for Lagos State and FCT, the model produced an accuracy level of 90.37% and 98.01% respectively.

5. Conclusion

There is no known universal antiviral treatment for COVID-19 at the moment, and no vaccine has been certified by the WHO though there are various claims by different scientists of having one kind of cure or vaccine or the other available. The results from the study is a true representation of the trend in all the states of Nigeria including the Federal Capital Territory (FCT), considering the fact that that almost all geopolitical areas in Nigeria were represented with the exception of areas that do not have testing facilities during the early days of COVID-19. This shows that either of the two models (ARIMA & ANN) can be used to optimally predict the COVID-19 confirmed cases in Nigeria. We are of the opinion that ANN would have produced a better all-round result if additional input neurons were made up of more of the factors influencing positive confirmations. This might include number of laboratories in each state, number of people tested on daily basis which are not available at real time.

As the spread of the virus have continued to disrupt economic activities in the world over, the risks posed, such as restriction of traffic, closure of cities and continued closure of schools, will create a huge economic downturn and slowdown the welfare of the people. To ease the tension and the suffering of the masses, the government must intensity its effort in providing necessary financial incentives to the people during this period [12]. With the gradual relaxation of the lockdown in most cities, leading to opening up of markets, churches and schools, it becomes imperative that safety measures have to be put in place. We therefore advice that more proactive measures be taken by all stakeholders to forestall an outbreak which the country cannot contain considering the level of health facilities and experts available in the country. As the lifting of stay-at-home and other restrictions continues to spread all through the states of Nigeria, it is important that the populace still continues to observe some degrees of safety measures. Adams et. al., (2020) [13] in a study found that there is a positive correlation between the number of deaths as a result to COVID-19 and the number of active and critical cases. This simply implies that it is not yet time for Nigerians to celebrate and off their guards as more cases are still being confirmed on a daily basis. It has been observed that people have abandoned the masks and continued to flock together in market places, religious houses, schools while social gatherings have all commenced even without strict adherence to COVID-19 protocols. In the future, other variables that could contribute to the forecasting accuracy of our models such as humidity, temperature, hygiene, sentiments among other variables can be included in the models.

Funding

The authors certify that there is no conflict of interest with any financial organization regarding the material discussed in the manuscript.

Conflicts of interest

The authors declare no conflict of interest.

References

[1] A. Muhammad, K. Suliman, K. Abeer, B. Nadia, S. Rabeea, COVID-19 infection: Origin, transmission, and characteristics of human coronaviruses, Journal of Advanced Research 24 (2020) 91–98.

[2] T. Tong Severe Acute Respiratory Syndrome Coronavirus (SARS-CoV). Perspectives in Medical Virology (2006) 16:43-95, DOI:10.1016/s0168-7069(06)16004-8.

[3] J. Amzata, K. Aminub, V. Kolob, A. Akinyeleb, J. Ogundairob, M. Dnjibo, Coronavirus outbreak in Nigeria: Burden and socio-medical response during the first 100 days, International Journal of Infectious Diseases 98 (2020) 218-224.

[4] NCDC, Nigeria Centre for Disease Control. COVID-19 Outbreak in Nigeria Situation Report; NCDC: Abuja, Nigeria, 2020.

[5] B. Jester, T. Uyeki, D. Jernigan, Readiness for Responding to a Severe Pandemic 100 Years After 1918, American Journal of Epidemiology 187(12) 2596-2602.

[6] I. Oladipo, A. Babatunde, Data Mining Classification Techniques for the Diagnosis, Treatment and Management of Diabetes Mellitus: A Review., Proceedings of the 1St International Conference of IEEE Nigeria Computer Chapter In collaboration with Department of Computer Science, University of Ilorin, Ilorin, Nigeria – 2016.

[7] B. Kaplan CDC Reveals Top 5 Causes of Death in the U.S., https://www.orlando health.com/content-hub/cdc-reveals-top-5-causes-of-death-in-the-us (accessed 13 August 2020).

[8] H. Akaike, Information Theory and an Extension of the Maximum Likelihood Principle. In: B.N. Petrov and F.

Csaki (eds.) 2nd International Symposium on Information Theory: (1793) 267-81 Budapest:Akademiai Kiado.

[9] S. Roy, G.S. Bhunia, P.K. Shit, Spatial prediction of COVID-19 epidemic using ARIMA techniques in India. Model. Earth Syst. Environ (2020), doi.org/10.1007/s40808-020-00890-y .

[10] Q. Yang, J. Wang, H. Ma, Research on COVID-19 based on ARIMA model—Taking Hubei, China as an example to see the epidemic in Italy, Journal of Infection and Public Health, (2020), Volume 13, Issue 10 Pages 1415-1418.

[11] D. Benvenuto, M. Giovanetti, L. Vassallo, Angeletti, M. Ciccozzi, Application of the ARIMA model on the COVID-2019 (2020), Epidemic Dataset, Data in Brief, 29 (2020) 105340.

[12] A. Oyelola, I. Adeshina, E. Gayawan Early Transmission Dynamics of Novel Coronavirus (COVID-19) in Nigeria, (2020). International Journal of Environmental Research and Public Health, 17 (2020) 3054 .

[13] O.S. Adams, A.M. Bamanga, U.H. Yahaya, O. R. Akano, Modeling COVID-19 Cases in Nigeria Using Some Selected Count Data Regression Models, (2020), International Journal of Healthcare and Medical Sciences, ISSN(e): 2414-2999, ISSN(p): 2415-5233, Vol. 6, Issue. 4 (2020) 64-73.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 396–400
Received: 2 November 2020
Accepted: 16 November 2020

Comparative analysis of the technology used to create multi-platform applications on the example of NW.js and Electron

Analiza porównawcza technologii tworzenia aplikacji wieloplatformowych na przykładzie NW.js i Electron

Maciej Hołowiński*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

Electron and NW.js are technologies that allow you to create multi-platform desktop applications. This article discusses the performance comparison of these frameworks. The research was undertaken in order to find an alternative to the Electron, which, despite its considerable popularity, is criticized in terms of its performance. For the purposes of the research, a test application with social media functionalities was designed and implemented. An identical application was created on both compared platforms and packaged for Windows 10 and Ubuntu 12.04 operating systems. Applications were tested in terms of the speed of data rendering, cache occupation and the size of the packed application. In each of the tests performed, NW.js obtained a better result than Electron, especially in tests consisting in downloading a relatively small amount of data. Based on the research carried out and the results obtained, it was found that in terms of performance, the NW.js framework is a much better solution than the Electron framework.

Keywords: NW.js; Electron; cross-platform development; comparative analysis

Streszczenie

Electron i NW.js to technologie pozwalające na tworzenie wieloplatformowych aplikacji desktopowych. Niniejszy artykuł omawia porównanie tych frameworków pod względem wydajności. Badania podjęto w celu znalezienia alternatywy dla Electron, który mimo swojej popularności, krytykowany jest pod względem oferowanej wydajności. Na potrzeby badań zaprojektowano i zaimplementowano aplikację testową z funkcjonalnościami mediów społecznościowych. Identyczna aplikacja została stworzona na obydwu porównywanych platformach oraz spakowana na systemy operacyjne Windows 10 i Ubuntu 12.04. Aplikacje badane były pod kątem szybkości renderowania danych, zajętości w pamięci podręcznej oraz rozmiaru spakowanej aplikacji. W każdym przeprowadzonym teście NW.js uzyskał lepszy wynik od Electron, szczególnie w testach polegających na pobraniu stosunkowo małej ilości danych. Na podstawie zrealizowanych badań i uzyskanych wyników stwierdzono, że pod względem wydajności framework NW.js jest znacznie lepszym rozwiązaniem od frameworka Electron.

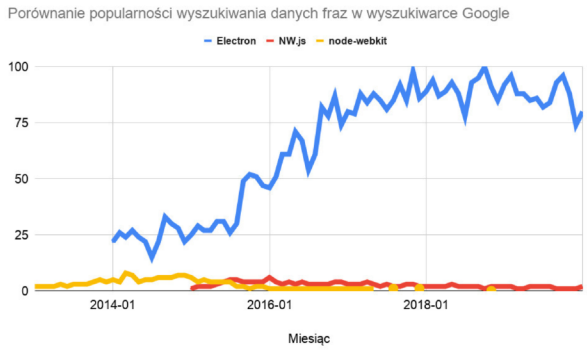
Słowa kluczowe: NW.js; Electron; tworzenie aplikacji wieloplatformowych; analiza porównawcza

1. Wstęp

Przed rozpoczęciem budowania aplikacji desktopowej programista musi zastanowić się nad wybraniem odpowiedniego systemu operacyjnego. Każdy z nich ma swoje własne API i środowiska programistyczne. Poznanie wszystkich, w przypadku chęci wydania aplikacji na wiele systemów, jest bardzo czasochłonne. Programista jednak ma również możliwość napisanie jednej aplikacji, która będzie mogła być rozprowadzana na wiele systemów operacyjnych (Windows, Linux, MacOS). Taki wybór pozwala oszczędzić czas oraz pomaga w obniżeniu kosztów produkcji [1]. Tworzenie aplikacji na różne platformy nie jest szczególnie nową koncepcją - przez ostatnie dziesięciolecie było to osiągalne za pomocą m.in.: Flash Air, JavaFX i Silverlight [2].

Jednak według corocznej ankiety prowadzonej przez stronę StackOverflow, technologie te z roku na rok tracą popularność [3]. Bazując na tej samej ankiecie najpopularniejszą technologią w roku 2019 jest JavaScript, a w kategorii „Frameworki, biblioteki i narzędzia" na pierwszym miejscu stoi Node.js. Zatem w omawianiu technologii pozwalających na tworzenie aplikacji wieloplatformowych warto przytoczyć Electron oraz NW.js, u fundamentów których leży Node.js, co za tym idzie – JavaScript. Porównując Electron i NW.js (dawniej node-

*Corresponding author
Email address: holowinski.maciej@pollub.edu.pl (M. Hołowiński)



Rysunek 1: Porównanie popularności Electron – NW.js (node-webkit) [4]

webkit) pod względem popularności, Electron jest zdecydowanym faworytem (Rysunek 1). W artykule postawiono następującą tezę: "Framework NW.js, jest pod względem wydajnościowym lepszym rozwiązaniem od frameworka Electron".

2. Przegląd literatury

W wyniku przeglądu literatury stwierdzono niewielki udział artykułów poruszających tematykę wydajności frameworków Electron i NW.js. Publikacje na temat NW.js omawiają jedynie sposób działania technologii oraz etapy implementacji aplikacji [5]. Pod względem dostępnych artykułów i książek znacznie większą popularnością cieszy się Electron. Przegląd serwisów oferujących dostęp do artykułów naukowych takich jak Google Scholar, Research Gate oraz BazTech nie wykazał żadnej pracy porównującej omawiane technologie pod względem wydajnościowym. Dostępne są jedynie badania porównujące framework Electron z innymi technologiami pozwalającymi na budowanie aplikacji desktopowych.

Artykuł [6] omawia różnice pomiędzy Electron, a NW.js pod względem czysto teoretycznym. Nie przedstawione są w nim żadne badania, prezentowane są natomiast różnice w funkcjonalnościach omawianych technologii. Technologie są porównywane pod względem architektury, możliwości pozwalających na zmianę interfejsu użytkownika, integracji z systemem operacyjnym, dostępem do systemu plików i obsługą multimediów. Autorzy wywnioskowali, że Electron posiada o wiele większą funkcjonalność pod względem budowania aplikacji, przy czym NW.js jest o wiele łatwiejszą technologią do opanowania.

Artykuł [7] porównuje implementacje oraz wydajność aplikacji wieloplatformowej tworzonej w Electron oraz JavaFX. Praca skupia się na porównywaniu dostępnych funkcjonalności oraz różnic w implementacji. Badania polegały na zbudowaniu podobnych aplikacji w obydwu technologiach i porównanie ich pod względem zajętości pamięci podręcznej oraz na mierzeniu czasu odpowiedzi z bazy danych. Wyniki zaprezentowały, że aplikacja w Electron zapewniła krótszy czas wykonania operacji i mniejsze zużycie pamięci niż aplikacja JavaFX. Jednak autor zaznaczył, że implementacja koncepcji OOP w Electron przy użyciu JavaScript wiąże się z pewnymi obawami dotyczącymi hermetyzacji i dziedziczenia.

Po przeanalizowaniu dostępnych publikacji naukowych można stwierdzić, że brak jest badań porównujących obydwie omawiane technologie, co stało się podstawą do ich przeprowadzenia.

3. Obiekty badań

3.1. Electron

Wypuszczony pierwotnie pod nazwą Atom Shell w roku 2013, Electron to framework open-source rozwijany oraz wspierany przez firmę GitHub. Electron jest biblioteką używaną do tworzenia desktopowych aplikacji za pomocą technologii internetowych JavaScript, HTML oraz CSS [4]. Aplikacje tworzone w Electron mogą być pakowane i uruchamiane na systemach Mac, Windows i Linux. Wszystko za pomocą połączenia Chromium, Node.js oraz natywnych API dla każdego systemu, które pozwalają na obsługę plików, ikon oraz powiadomień.

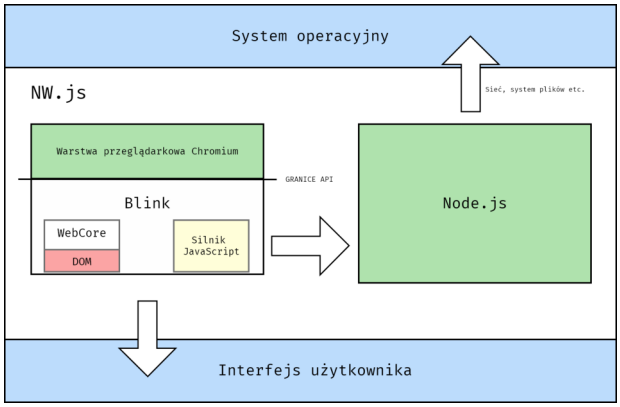
Aplikacje tworzone za pomocą Electron składają się z dwóch typów procesów: Main oraz Renderer. Pierwszy z nich jest procesem głównym, który określany jest w skrypcie package.json i odpowiedzialny jest za wyświetlanie graficznego interfejsu użytkownika poprzez tworzenie stron internetowych [8]. Każda ze stron uruchamia swój własny, drugi typ procesów - proces Renderer.

Proces główny tworzy strony za pomocą instancji okna BrowserWindow. Każde okno posiada swój własny, niezależny od innych, proces Renderer, który kończony jest po zamknięciu okna. Komunikacja między procesem Main, a procesami Renderer jest możliwa dzięki wbudowanym interfejsom API HTML5 np.: storage API, localStorage, sessionStorage oraz IndexedDB. Electron oferuje swoje własne rozwiązanie - system IPC. Przechowuje on obiekty w procesie głównym jako globalne zmienne oraz pozwala na dostęp do nich poszczególnym instancjom BrowserWindow.

3.2. NW.js

Projekt NW.js był wprowadzony w 2011 roku. początkowo pod nazwą node-webkit, przez Rogera Wanga ówcześnie pracującego w Intel's Open Source Technology Center (Shanghai). NW.js jest środowiskiem uruchomieniowym aplikacji internetowych stworzonym przy pomocy Node.js oraz Chromium [9]. Pozwala na wywoływanie modułów Node.js bezpośrednio z modelu obiektowego dokumentu (DOM). Oprócz Node.js u podstaw NW.js leży Blink - silnik przeglądarki internetowej rozwijany przez Google od 2013 roku, stworzony na podstawie silnika WebKit [10]. Blink rozwijany jest jako część projektu Chromium.

Diagram (Rysunek 2) przedstawia połączenie Node.js oraz Blink w celu uzyskania dostępu do graficznego interfejsu użytkownika oraz systemu operacyjnego.



Rysunek 2: Diagram połączenia Node.js oraz Blink [10]

4. Metoda badań

W celu przeprowadzenia badań zaprojektowano oraz stworzono aplikację mediów społecznościowych pozwalającą użytkownikom m.in. na udostępnianie zdjęć innym użytkownikom systemu, ich komentowanie oraz obserwowanie użytkowników. Implementacja została wykonana w NW.js oraz Electron. Aplikacja została podzielona na następujące moduły:

- **Rejestracja** - moduł pozwalający na stworzenie nowego użytkownika używając loginu, adresu e-mail oraz hasła,
- **Logowanie** - moduł sprawdzający poprawność danych oraz autoryzowanie użytkownika,
- **Profil** - moduł pozwalający użytkownikowi na wyświetlanie zdjęć, zmianę opisu profilu, zmianę zdjęcia profilowego, wyszukiwanie innych użytkowników, udostępnianie zdjęć oraz wyświetlanie listy użytkowników obserwowanych i obserwujących,
- **Interakcje** - moduł obsługujący obserwowanie użytkowników, usuwanie użytkowników z listy obserwowanych oraz komentowanie zdjęć,
- **Tablica** - moduł odpowiedzialny za wyświetlanie wszystkich użytkowników obserwowanych przez użytkownika zalogowanego.

Do celów badawczych wykorzystano moduł Tablica oraz moduł Profil. Mierzono czas w jakim aplikacja wyrenderuje odpowiednie elementy. Dla modułu Tablica jest to odpowiednio 10, 50 oraz 150 zdjęć, a dla modułu Profil jest to czas odświeżenia strony profilu użytkownika oraz czas wyświetlenia odpowiednio 10, 50 oraz 250 wierszy zawierających nazwy obserwatorów. Każde ze zdjęć wyświetlanych na stronie Tablica jest o rozdzielczości 1920x1080 i rozmiarze 350kb.

Testy zostały wykonane na komputerze o następujących specyfikacjach:

- procesor - Intel Core i5 - 4690k;
- pamięć RAM - 8 GB;
- dysk - 256GB SSD;
- karta graficzna – NVIDIA GeForce GTX 970;
- system operacyjny – Windows 10 / Ubuntu 12.04;

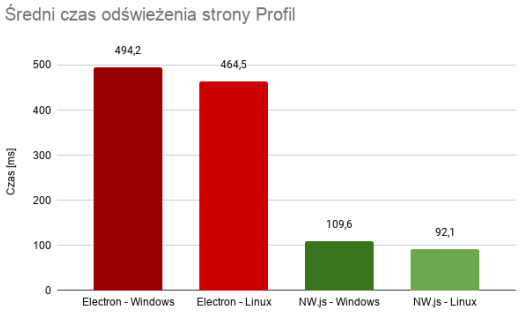
5. Wyniki badań

Testy wydajnościowe zostały przeprowadzone według następujących scenariuszy:

- S1 - odświeżanie strony, czas mierzony od kliknięcia w przycisk "Profile" do wyrenderowania wszystkich elementów profilu zalogowanego użytkownika;
- S2 - wyświetlanie obserwatorów, czas mierzony od kliknięcia w przycisk "Followers" do wyświetlenia 10, 50, 250 wierszy z nazwami obserwatorów;
- S3 - wyświetlanie zdjęć, czas mierzony od kliknięcia przycisk "Home" do wyrenderowania 10, 50, 150 zdjeć;

5.1. Scenariusz testowy S1

Rysunek 3 przedstawia wykres obrazujący średnie czasy odświeżania strony Profil omawianych technologii na poszczególnych systemach operacyjnych. Electron uzyskał w tym przypadku zaskakująco słaby wynik. W NW.js strona zawierająca podstawowe informacje o użytkowniku ładuje się prawie 5 razy szybciej niezależnie od systemu operacyjnego. Nawet bez badań, do takiego samego wniosku może dojść użytkownik używający aplikacji na porządku dziennym – w odświeżaniu widoku, czy też w przechodzeniu między widokami, zauważalne jest opóźnienie. W porównaniu z NW.js ładowanie strony jest niemal natychmiastowe.

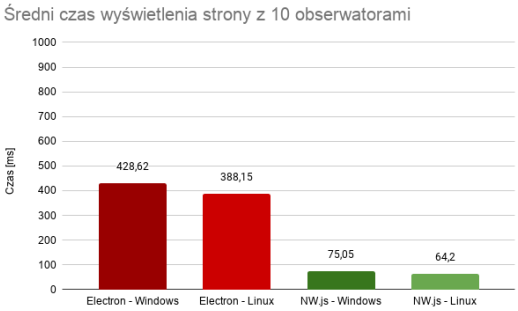


Rysunek 3: Diagram przedstawiający średni czas odświeżania strony Profil

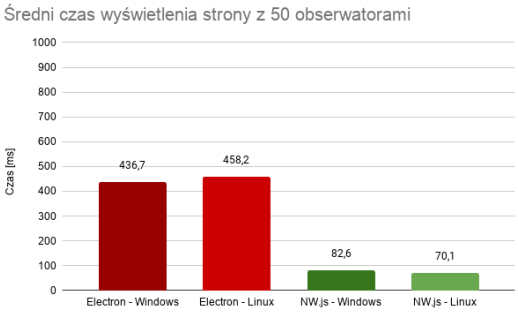
5.2. Scenariusz testowy S2

Rysunki 4, 5 i 6 przedstawiają wyniki badań przeprowadzone według scenariusza testowego S2.

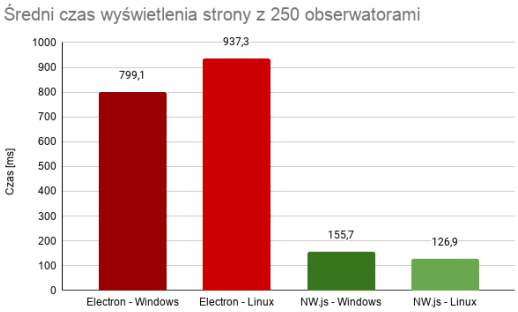
Scenariusz S2 miał na celu sprawdzenie w jakim czasie badane frameworki wyświetlą odpowiednio 10, 50 oraz 250 wierszy zawierających nazwę użytkownika oraz jego zdjęcie profilowe. W tym przypadku testy zdecydowanie umocniły pozycję NW.js. W przypadku testów dla 10 i 50 wierzsy, NW.js jest około 4 razy szybszy od konkurenta. Warto również zauważyć, że przy średniej prędkości ładowania 70ms, ładowanie jest dalej płynne i natychmiastowe. Dopiero przy renderowaniu 250 wierszy NW.js przy średniej prędkości ładowania 130ms, zauważalne jest opóźnienie w ładowaniu. Ciekawą rzeczą, którą pokazało ładowanie większej liczby wierszy jest fakt, że system Windows wyświetla treści 15% szybciej od



Rysunek 4: Średni czas wyświetlenia strony z 10 obserwatorami



Rysunek 5: Średni czas wyświetlenia strony z 50 obserwatorami



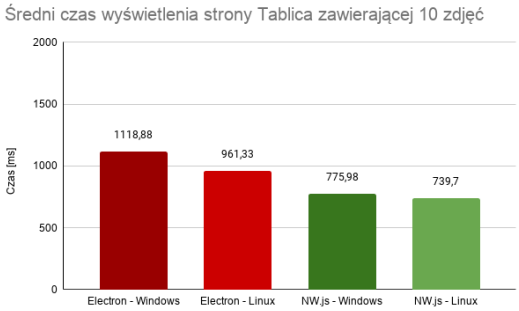
Rysunek 6: Średni czas wyświetlenia strony z 250 obserwatorami

systemu Linux. Jest to jedynie ciekawostka, która wymagałaby przeprowadzenia kolejnych badań, szczególnie zwracając uwagę na stosunkowo mały wpływ systemu operacyjnego na szybkość renderowania w poprzednich wynikach testów.

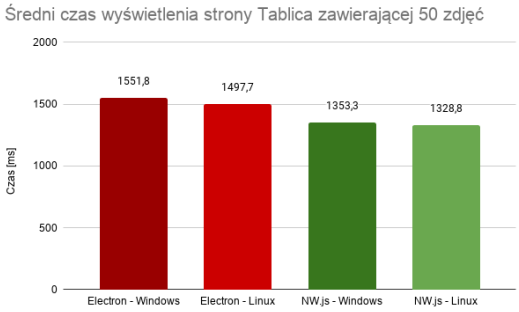
5.3. Scenariusz testowy S3

Rysunki 7, 8 i 9 przedstawiają wyniki nadań przeprowadzonych według scenariusza testowego S3.

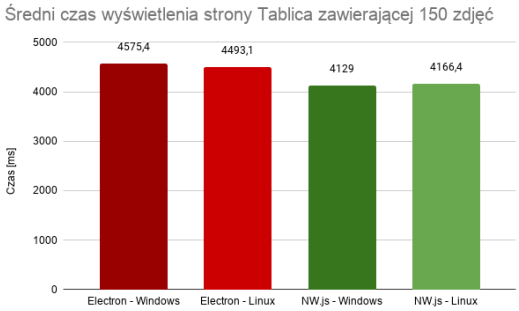
Scenariusz S3 sprawdzał szybkość wyświetlania odpowiednio 10, 50 i 150 zdjęć w wysokiej rozdzielczości. W tym przypadku wyniki pomiędzy obydwoma frameworkami były bardziej zbliżone. W ładowaniu 10 zdjęć NW.js był o ok. 31% szybszy, w 25 zdjeć o ok. 22%, a w przypadku ładowania 150 zdjęć – ok. 11%. Ten test uwidocznia fakt, że wraz ze zwiększoną ilością danych potrzebnych do pobrania, wydajność NW.js w porównaniu do Electrona spada. Test pozolił również na zaobser-



Rysunek 7: Średni czas wyświetlenia strony Tablica zawierającej 10 zdjęć



Rysunek 8: Średni czas wyświetlenia strony Tablica zawierającej 50 zdjęć



Rysunek 9: Średni czas wyświetlenia strony Tablica zawierającej 150 zdjęć

wowanie wpływu systemu operacyjnego na wydajność. W przypadku ładowania 10 zdjęć Electron działający na systemie Linux jest szybszy od Electron działającego na Windows o ok. 20%. W przypadku ładowania 50 i 150 zdjęć wyniki pomiędzy systemami oraz omawianymi technologiami nie są duże. Przy większej ilości danych na wynik mogą wpływać czynniki zewnętrzne takie jak ograniczenia sprzętowe czy też sama wydajność Node.js.

6. Wnioski

W tym artykule przedstawiono porównanie wydajności wieloplatformowej aplikacji Electron i NW.js. Wyniki przeprowadzonych badań pokazują różnicę wydajności pomiędzy porównywanymi technologiami. NW.js był technologią szybszą w każdym analizowanym przypad-

ku. Electron, nawet w operacjach pobierających małą ilość danych wykazywał znaczne opóźnienia wpływające na płynność obsługi aplikacji. Wpływ systemu operacyjnego na wydajność aplikacji jest trudny do oceny. O ile w większości przypadków nie wpływa on znacznie na szybkość działania aplikacji, tylko niektóre testy wykazały wyraźną przewagę jednego z systemów, co nie pozwala na wydanie jednoznacznej opinii w tej kwestii. Na podstawie uzyskanych wyników, można stwierdić, że teza "Framework NW.js, jest pod względem wydajnościowym lepszym rozwiązaniem od frameworka Electron" jest prawdziwa.

Literatura

[1] P. Lindhol, Web technologies for cross-platform desktop applications–a feasible option?, 2017.

[2] D. Alymkulov, Desktop Application Development Using Electron Framework: Native vs. Cross-Platform, 2019.

[3] Developer Surver Results 2019, https://insights.stackoverflow.com/survey/2019, [10.10.2020].

[4] Porównanie popularności Electron – NW.js, https://trends.google.com/trends/explore?date=all&q=%2Fg%2F11bw˙559wr,nw.js,node-webkit,,[10.10.2020].

[5] D. Sheiko, Cross-platform Desktop Application Development: Electron, Node, NW.js, and React: Build desktop applications with web technologies, Packt Publishing, 2017.

[6] Z. Hussein, An In-Depth Comparison of Software Framework for Developing Desktop Applications Using Web Technologies, 2019.

[7] A. Alkhars, Cross-Platform Desktop Development (JavaFX vs. Electron), 2017.

[8] S. Kinney, Electron in Action, Manning Publications, 2018.

[9] NW.js, https://nwjs.readthedocs.io/en/latest/, [10.10.2020],

[10] A. Benoit, NW.js Essentials, Packt Publishing, 2015.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 401–404
Received: 22 September 2020
Accepted: 5 October 2020

Faster R-CNN model learning on synthetic images

Model Faster R-CNN uczony na syntetycznych obrazach

Błażej Łach*, Edyta Łukasik

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

Machine learning requires a human description of the data. The manual dataset description is very time consuming. In this article was examined how the model learns from artificially created images, with the least human participation in describing the data. It was checked the effect of augmentation and progressive image size when training model on a synthetic dataset. The model has achieve up to 3.35% higher mean average precision on syntetic dataset in the training with increasing images resolution. Augmentations improved the quality of detection on real photos. The production of artificially generated training data has a great impact on the acceleration of prepare training, because it does not require as much human resources as normal learning process.

Keywords: computer vision; synthetic images; Faster R-CNN; deep learning

Streszczenie

Uczenie maszynowe wymaga opisu danych przez człowieka. Opisywanie zbioru danych ręcznie jest bardzo czasochłonne. W artykule zbadano jak model uczył się na zdjęciach sztucznie wytworzonych, z jak najmniejszym udziałem człowieka przy opisywaniu danych. Sprawdzono jaki wpływ miało zastosowanie augmentacji i progresywnego rozmiaru zdjęcia przy treningu modelu na syntetycznym zbiorze. Model osiągnął nawet o 3,35% wyższą średnią precyzję na syntetycznym zbiorze danych przy zastosowaniu treningów z rosnącą rozdzielczością. Augmentacje poprawiły jakość detekcji na rzeczywistych zdjęciach. Wytwarzanie sztucznie danych treningowych ma duży wpływ na przyśpieszenie przygotowania treningów, ponieważ nie wymaga tak dużych nakładów ludzkich, jak klasyczne uczenie modeli z danymi opisanymi przez człowieka.

Słowa kluczowe: computer vision; sztuczne obrazy; Faster R-CNN; głębokie uczenie

1. Wprowadzneie

Trenując modele sieci neuronowych do nauki potrzeba danych z opisem. Zbiór danych oznaczją zwykle ludzie, wykorzystując swoje zmysły poznawcze, zaznaczają obiekty widoczne na zdjęciach. Deskrypcji danych zwykle trzeba poświęcić wiele godzin, aby dokładnie scharakteryzować większą liczbę obrazów. Wykorzystując komputery w prosty sposób można wygenerować zbiór danych syntetycznych, poprzez nanoszenie wzorców na zdjęcia tła i zapisywanie cech obiektów.

Oznaczając jakość modelu przy detekcji obiektów stosuje się metrykę mean average precision (mAP). Metryka ta ocenia predykowanie położenia i klasyfikacji obiektów na zadanym zbiorze. Podaczas wyliczania mAP stosowana jest skala od 0 do 1. Jednak warto zaznaczyć, że w pracach z dziedziny rozpoznawania obiektów, przedmiotowa metryka jest najczęściej podawana w procentach, co zostało zastosowane również w ninejszej pracy.

*Corresponding author
Email address: blazej.lach@pollub.edu.pl (B. Łach)

Powszechnie wiadomo, że przy danych opisywanych przez człowieka, modele są w stanie nauczyć się rozpoznawać obiekty na zdjęciach. Twórcy Faster R-CNN nauczyli wykrywać obiekty na podstawie zdjęć ze zbioru COCO i Pascal VOC, poprawili wynik osiągnięty przez Fast R-CNN o 2,2% mAP na zbiorze COCO [1, 2]. Wykorzystując Faster R-CNN nauczono model wykrywać twarze ludzi z dość dużą dokładnością [3].

Augmentacje są to przekształcenia obrazu takie jak: skalowanie, obracanie, nadawanie perspektywy, rozmazanie, zmiana kolorów, wykorzystywane do modyfikacji danych treningowych. Dzięki przekształceniom obrazów treningowych można powiększać zbiór danych lub zwiększać różnorodność cech obiektów występujących w zbiorze. Bezsprzecznie można wykorzystać augmentacje do poprawienia wyników detekcji na zdjęciach rzeczywistych. Stosując grupy przekształceń zdjęć do prawdziwych obrazów, można zwiększyć wykrywalność obiektów na zbiorze COCO o 2,3% mAP i o 2,7% mAP na zbiorze Pascal VOC [4]. Ponadto wykorzystując grupy augmentacji można uzyskać większy przyrost mAP na mniejszych zbiorach danych [4]. Augmentacje

wspomagają również klasyfikatory obrazów. Dzięki stosowaniu przekształceń zdjęć poprawiono wyniki precyzji na zbiorach ImageNet o 0,4% i CIFAR-10 o 0,6% przy klasyfikacji obiektów [5]. Mając na uwadze powyższe, można zauważyć jak bardzo istotne są augmentacje w procesie uczenia modeli.

Przeglądając prace innych autorów dotyczące tego tematu można zauważyć, że większości sytuacji syntetyki wykonywane są pod ściśle określony przypadek, niewykorzystywany do detekcji obiektów. Udało się nauczyć model określać pozycje ciała człowieka na podstawie nacisku na materac. Model ten trenowany był na sztucznie wytwarzanych obrazach nacisku generowanych z obiektów 3D, oddziaływujących ze sobą [6]. Wytrenowano również model rozpoznający głębię przedmiotów z wykorzystaniem masek obiektów, stosując Mask R-CNN. Do treningu wykorzystano sztucznie wygenerowany zbiór danych, stworzony z obiektów 3D umieszczonych w pudełku [7]. Nauczono również model sterować ramieniem robota na podstawie sztucznego środowiska 3D. Wykorzystując augmentacje obrazów syntetycznego środowiska zmniejszono błąd położenia kostki, którą robot miał przenosić. Model sprawdził się przy sterowaniu rzeczywistym ramieniem dzięki nauce na sztucznym środowisku [8].

Na podstawie przeprowadzonego przeglądu dokonań innych autorów można, zauważyć, że nikt nie przeprowadzał eksperymentów uczenia modelu do rozpoznawania syntetyków bez wykorzystywania obiektów 3D. Według tych informacji wyznaczono następujące cele badawcze. Sprawdzono, czy model jest w stanie nauczyć się rozpoznawać obiekty na zdjęciu przy treningu na obrazach sztucznie generowanych. Dodatkowo zweryfikowano, jakie znaczenie ma uczenie z rosnącym rozmiarem zdjęcia treningowego na detekcję obiektów. W badaniu sprawdzano również, jaki wpływ mają augmentacje syntetycznego zbioru treningowego na predykcje modelu.

Kolejna sekcja skupia się na wykorzystanych technologiach do przeprowadzenia badania. Zamieszczona będzie również charakterystyka maszyny wykorzystywanej do treningu i sposób prowadzenia treningów. W tym dziale opisana będzie również metoda badawcza. Następna sekcja przedstawiać będzie wyniki badania i ich interpretację. Ostatni dział będzie podsumowaniem przeprowadzonego badania.

2. Materiały i metody

Python był głównym narzędziem stosowanym przy przeprowadzaniu eksperymentu. Język ten jest szeroko wykorzystywany przy uczeniu maszynowym. Wybrano go głównie ze względu na dużą dostępność bibliotek wspomagających programowanie. Do badań korzystano z Python'a w wersji 3.7.9. Jako bibliotekę do prowadzenia treningów wykorzystano TensorFlow w wersji 1.15.2. Biblioteka ta współpracuje z wyżej wymienionym językiem. TenosrFlow zapewnia kod umożliwiający treningi, ewaluację i serwowanie predykcji modelu, przy użyciu do obliczeń CPU lub GPU [9]. Faster R-CNN jest szkieletem do detekcji obiektów. Został stworzony poprzez dodanie sieci proponującej regiony (RPN) do Fast R-CNN [1]. Szkielet ten jest zaimplementowany w TensorFlow. W badaniu korzystywano z Faster R-CNN, we współpracy z ResNet50, jako siecią ekstraktującą cechy. Albumentations jest szybką biblioteką stosowaną do nakładania augmentacji na zbiór danych [10]. Bibliotekę tą wykorzystano w wersji 0.4.6. Jest ona dedykowana do języka Python. Dzięki Albumentations możliwe jest nakładanie wybranych przekształceń na obrazy. Przy modyfikacji obrazów metamorfozie ulegają również opisy zdjęć. Tworząc zdjęcia syntetyczne wykorzystywano bibliotekę OpenCV. Przy jej pomocy wczytywano zdjęcia, nanoszono wzorce na zdjęcia tła i zapisywano zamiany do pliku. Biblioteka ta zaimplementowana w języku C++ współpracuje dzięki nakładce z Python'em i Albumentations. Wykorzystywano ją w wersji 4.4.0.42.

Ucząc sieci neuronowe ważna jest fizyczna maszyna wykorzystywana do treningów. Aby zapewnić szybkie obliczenia, wskazane jest wykorzystanie wydajnych podzespołów. Badania przeprowadzano na komputerze wyposażonym w 8 rdzeniowy procesor Intel i7 7700K z taktowaniem maksymalnym 4,5 GHz. Podczas badania modele miały do dyspozycji 16 GB pamięci RAM o częstotliwości 3600 MHz w dwóch modułach po 8 GB, pracujących w trybie dual channel. Wszystkie obliczenia były wykonywane na karcie graficznej Nvidia GTX1080Ti, posiadającej 11 GB pamięci VRAM. Taktowanie pamięci GPU wynosiło 11124 MHz, natomiast taktowanie rdzenia wynosiło 1683 MHz.

Trenując modele potrzeba dwóch zbiorów danych: treningowego i ewaluacyjnego. Sieć neuronowa wykorzystując syntetyki uczy się informacji o wykrywanych obiektach. Zbiór danych testowych wykorzystywany jest do sprawdzania, czy model nauczył się generalizować informacje o obiektach. Zbiory te przy rzeczywistych danych powinny pochodzić z różnych dystrybucji, aby w inny sposób ukazywać obiekty.

Obiektami do rozpoznania przez model były wybrane polskie znaki drogowe. Generując syntetyki nanoszono zdjęcia wzorców na obrazy tła. Obrazy tła zostały selekcjonowane w taki sposób, aby przedstawiały naturalny kontekst występowania obiektów. Na zdjęciach występowały głównie ulice, zadbano o to, by nie było na nich obiektów, które model miał rozpoznawać. Do przygotowania zbiorów wyselekcjonowano 40 zdjęć dla zbioru danych treningowych i 10 innych zdjęć dla zbioru danych testowych. Wszystkie obrazy tła przycięto lub przeskalowano do rozmiarów 1536 pikseli szerokości i 1536 pikseli wysokości. Łącząc wzorce i tła, nanoszono w losowej ilości i wielkości zdjęcie obiektu na obraz tła, zapisując położenie i nazwę klasy. Według powyższego sposobu wygenerowano 1000 obrazów dla zbioru treningowego i 100 obrazów dla zbioru testowego. W danych treningowych i ewaluacyjnych występują wszystkie obiekty które model miał rozpoznawać. Liczba obiektów dla klas

w zbiorze testowym wynosi od 20 do 34 wystąpień. Natomiast w danych treningowych jest to od 238 do 318 obiektów dla każdej klasy. Zbiór ewaluacyjny przygotowano stosując inne tła i skalowanie wzorców niż występujące w zbiorze treningowym. Wygenerowany zbiór treningowy będzie określany, jako zbiór pierwszy.

Stosując Albumentations stworzono przekształcone zbiory danych treningowych. Drugi zbiór wytworzono przez nałożenie przekształcenia IAAPerspective, uzyskując obiekty posiadające perspektywę. Trzeci zbiór stworzono wykorzystując przekształcenie ShiftScaleRotate, zmniejszając i obracając całe zdjęcia. Czwarty zbiór augmentowany był przy pomocy obu przekształceń, otrzymując wszystkie powyższe efekty zdjęć. Każde z przekształceń nakładano na zdjęcia treningowe z rozkładem równomiernym.

Do weryfikacji celów zaplanowano 8 treningów. Wykorzystując 4 zbiory przeprowadzono po jednym treningu na zbiór, w standardowy sposób. Po jednym zbiorze na trening zastosowano również do kolejnych 4 treningów, które wykorzystywały zwiększający się w trakcie rozmiar zdjęcia. Wszystkie treningi rozpoczynały naukę z tego samego punktu kontrolnego. Treningowy punkt kontrolny przechowuje dokładną wartością wszystkich patametrów używanych przez model. Rosnąca rozdzielczość oznacza uruchomienie treningu przy przeskalowanym zdjęciu wejściowym do 384x384 pikseli. Z takiego treningu wybierano najlepszy punkt kontrolny i kontynuowano go ze zwiększoną rozdzielczością do 768x786 pikseli. Ponownie wybierano najlepszy punkt kontrolny i kontynuowano go z rozdzielczością docelową 1536x1536 pikseli. Przy analizie wyników dla każdego z modeli będzie podane mAP na sztucznym zbiorze testowym. W celu sprawdzenia, czy modele potrafią rozpoznawać rzeczywiste obiekty, każdy model zostanie odpytany o predykcję na tym samym zdjęciu rzeczywistym.

3. Rezultaty i dyskusja

Wyniki mAP wszystkich modeli sprawdzane na sztucznym zbiorze testowym zestawiono w tabeli 1. Można zauważyć, że stosowanie wybranych w tym badaniu augmentacji pogarsza mAP na syntetycznym zbiorze testowym. Prawdopodobnie jest to spowodowanie zbyt małą różnorodnością cech obiektów w zestawie danych ewaluacyjnych. Zastosowanie treningu z rosnącą rozdzielczością przeważnie zwiększało mAP na sztucznym zbiorze testowym. Stosowanie rosnącej rozdzielczości poprawiło wynik mAP o 1,08% dla pierwszego zbioru danych. Rosnąca rozdzielczość wykorzystana podczas treningu poprawiła wynik o 1,98% mAP dla zbioru z augmentacjami ShiftScaleRotate. Jedynie użycie zestawu danych przekształcanych za pomocą IAAPerspective pogorszyło o 0,62% mAP przy zastosowaniu treningu z rosnącą rozdzielczością. Największy przyrost mAP odnotowano dla rosnącej rozdzielczości trenowanej na zbiorze łączonych augmentacji IAAPerspective, ShiftScaleRotate i wynosił 3,35% mAP. Zastosowanie zwiększającej się rozdzielczości prawdopodobnie nauczyło model lepiej generalizować informacje o obiektach ze zbioru syntetycznego.

Tabela 1: Wyniki modeli testowanych na zbiorze syntetycznym

Zastosowany trening	Wykorzystane augmentacje	mAP (%)
Standardowy	Brak	72,50
	IAAPerspective	70,04
	ShiftScaleRotate	69,21
	Obie	67,36
Rosnąca rozdzielczość	Brak	73,58
	IAAPerspective	69,42
	ShiftScaleRotate	71,19
	Obie	70,71

Rysunek 1: Predykcje na zdjęciu rzeczywistym modelu trenowanego standardowo na zbiorze bez augmentacji

Przeglądając predykcje modeli na prawdziwych zdjęciach można zauważyć, że treningi bez augmentacji wskazywały koło samochodu, jako znak ruch okrężny (Rysunek 1, 2). Warto zauważyć, że model trenowany ze zwiększającą rozdzielczością, z mniejszą pewnością wskazał koło samochodu, jako znak (Rysunek 2). Oba modele oznaczyły poprawnie znaki przejście dla pieszych i zakaz zatrzymywania się. Model uczony z zastosowaniem narastającego rozmiaru zdjęcia wskazał prawidłowo znak zakaz postoju, którego model trenowany w standardowy sposób nie oznaczył. Sieć trenowana z rosnącą rozdzielczością oznaczyła dodatkowo tabliczkę z nazwą ulicy (zasłoniętą przez opis), jako znak teren zabudowany z dużą pewnością.

Predykcja na zdjęciu rzeczywistym modelu wytrenowanego bez rosnącego rozmiaru zdjęcia na zbiorze z augmentacjami ShiftScaleRotate, jest jedną z najlepszych predykcji spośród wszystkich modeli (Rysunek 3). Model poprawnie i z wysoką pewnością oznaczył dobrze

Rysunek 2: Predykcje na zdjęciu rzeczywistym modelu trenowanego z rosnącą rozdzielczością na zbiorze bez augmentacji

widoczne znaki, dokładnie zaznaczając obwiednie obiektów. Sieć nie wskazała jedynie małych znaków przejście dla pieszych w centralnej części zdjęcia.

Rysunek 3: Predykcje na zdjęciu rzeczywistym modelu trenowanego standardowo na zbiorze z augmentacją ShiftScaleRotate

Poprawnie oznaczone obiekty na zdjęciach rzeczywistych, sugerują że można wytrenować model na zdjęciach syntetycznych, do rozpoznawania obiektów wystepujących na prawdziwych zdjęć. Przy zastosowaniu augmentacji można poprawić jakość detekcji na rzeczywistych zdjęciach. Błędnie wskazane obiekty na zdjęciach mogą oznaczać, że model nie wyuczył się generalizować wystarczająco dobrze informacji. Rozwiązaniem tego problemu mogło by być dodanie do zbioru syntetycznego obrazów podobnych do obiektów, które model ma rozpoznawać, przykładowo koła samochodów lub tablice z nazwami miast i ulic. Brak rozpoznawania małych obiektów na zdjęciach rzeczywistych sugeruje, brak występowania tak małych wielkości obiektów w zbiorze treningowym. Aby zwiększyć wykrywalność małych obiektów, należało by wygenerować zbiór syntetyczny zawierający mniejsze zdjęcia wzorców nanoszonych na obrazy treningowe.

4. Podsumowanie

Dzięki komputerom nie ma przszkód, aby wygenerować sztuczne zbiory danych wraz z opisem. Na takim zbiorze można wytrenować model do rozpoznawania obiektów na zdjęciach rzeczywistych. Wykorzystując augmentacje na sztucznym zbiorze jest możliwe poprawienie jakości predykcji. Do przebadania pozostaje wpływ na mAP zastosowania treningu z rosnącą rozdzielczością przy wykorzystaniu zbioru ze zdjęciami rzeczywistymi.

Literatura

[1] S. Ren et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, t. 39, nr 6, s. 1137–1149, cze. 2017, https://doi.org/10.1109/TPAMI.2016.2577031.

[2] R. Girshick, Fast R-CNN, [W:] 2015 IEEE International Conference on Computer Vision (ICCV), 2015, https://doi.org/10.1109/ICCV.2015.169.

[3] H. Jiang, E. Learned-Miller, Face Detection with the Faster R-CNN, [W:] 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), 2017, https://doi.org/10.1109/FG.2017.82.

[4] B. Zoph et al., Learning Data Augmentation Strategies for Object Detection, 2019, https://arxiv.org/pdf/1906.11172.pdf.

[5] E. Cubuk et al., AutoAugment: Learning Augmentation Strategies from Data, [W:] 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, https://doi.org/10.1109/CVPR.2019.00020.

[6] H. Clever et al., Bodies at Rest: 3D Human Pose and Shape Estimation from a Pressure Image using Synthetic Data, [W:] 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, https://doi.org/10.1109/CVPR42600.2020.00625.

[7] M. Danielczuk et al., Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data, [W:] 2019 International Conference on Robotics and Automation (ICRA), 2019, https://doi.org/10.1109/ICRA.2019.8793744.

[8] A. Pashevich et al., Learning to Augment Synthetic Images for Sim2Real Policy Transfer, [W:] 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, https://doi.org/10.1109/IROS40897.2019.8967622.

[9] M. Abadi et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2016, https://arxiv.org/pdf/1603.04467.pdf.

[10] A. Buslaev et al., Albumentations: fast and flexible image augmentations, Information, t. 11, nr 2, s. 125, luty 2020, https://doi.org/10.3390/info11020125.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 405–411
Received: 23 October 2020
Accepted: 13 November 2020

Enviromental data visualisation using Delaunay triangulation

Wizualizacja danych środowiskowych z wykorzystaniem triangulacji Delauney.

Mateusz Nowosad*

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

Graphical data representation is very helpful when analyzing environmental data. It allows for discovering trends in data and analysis of phenomena occurring in the area. There are many possibilities to represent such values graphically. This article contains visualizations generated using Delauney triangulation to represent data on a map. Strengths and weaknesses, comparative analysis with another solution, performance, and usage suggestions will be presented.

Keywords: enviromental data; delaunay; statistical maps

Streszczenie

Graficzna reprezentacja danych jest bardzo pomocna przy analizie danych środowiskowych. Pozwala na dostrzeżenie trendów w danych i analizę zjawisk zachodzących na określonym terenie. Jest wiele możliwości przedstawienia takich wartości. W tym artykule przedstawiono sposób wykorzystujący triangulacje Delauney do reprezentacji danych na mapie. Przedstawione będą mocne i słabe strony, analiza porównawcza z innym rozwiązaniem, wydajność oraz sugestie dotyczące użycia.

Słowa kluczowe: dane środowiskowe; delaunay; mapy statystyczne

1. Wstęp

Stan powietrza ma bardzo duży wpływ na ludzkie zdrowie i środowisko. Według ostatnich szacunków choroby spowodowane jakością powietrza zabijają ok. milion Europejczyków rocznie [1]. Przykładem istotnego parametru badania powietrza jest PM10, który pojawia się czasami w literaturze pod nazwą “pył gruby”. Udowodniono silny związek przyczynowo skutkowy pomiędzy wzrostem stężenia tej mieszaniny a chorobami dróg oddechowych i układu krążenia [2]. Wraz ze wzrastającą świadomością społeczną wzrasta zapotrzebowanie na rozwiązania pozwalające na analizę i kalkulacje ryzyka związanego z poziomem zanieczyszczeń pochodzenia antropogenicznego. Udostępnienie przystępnych narzędzi wizualizujących dane środowiskowe pozwala na świadome decyzje i kalkulacje ryzyka przez zainteresowanych ludzi, a także może być wykorzystywane do kolejnych prac naukowych związanych z tym tematem.

W niniejszym artykule poddano analizie wykorzystanie triangulacji Delaunay. Bardzo często w literaturze pojawiają się wizualizacje oparte na kolorowych punktach albo mapach ciepła, ale rzadko widuje się graficzne reprezentacje oparte o wielokąty. W tym artykule starano się udowodnić przydatności takiego podejścia. Nacisk położony zostanie na dostępne dane ogólnokrajowe. Wizualizacje oraz badanie wydajności zostaną oparte na przygotowanej aplikacji webowej.

2. Uogólniony opis algorytmu

Triangulacja Delaunay jest popularnym narzędziem z dziedziny geometrii obliczeniowej. Jest ona grafem dualnym diagramu Woronoja. Opracowana została przez matematyka Borysa Delone w Rosji 1934 r. Do generowania wizualizacji w tym artykule użyty został algorytm typu „dziel i rządź” opracowany przez Leonidasa J Guibasa i Jorge’a Stolfisa w pracy o tytule ”Primitives for the manipulation of general subdivisions and the computation of Voronoi” [3]. Jego złożoności obliczeniowa wynosi O(nlogn) dla najgorszego przypadku. Pozwala on na stworzenie siatki trójkątów na podstawie punktów o dowolnym rozmieszczeniu i dowolnej ilości na danej płaszczyźnie.

Opis algorytmu opisany na podstawie wcześniej wymienionej pracy. Wykonanie tego algorytmu rozpoczynane jest podzieleniem punktów na dwie połowy, lewą(L) i prawą(R), oddzielone danym koordynatem x. Następnie rekurencyjnie obliczana jest triangulacja Delauneya dla obu wyżej wymienionych podzbiorów. Końcowym krokiem jest połączenie połówek triangulacji w jedną triangulację całego oryginalnego zbioru. Warto tutaj zaznaczyć, że taka rekurencyjna dekompozycja nie

*Corresponding author
Email address: mateusz.nowosad@pollub.edu.pl (M. Nowosad)

może być użyta, gdy liczba punktów jest mniejsza niż cztery, ze względu na to, że jedna ze stron L albo R skończyłaby tylko z jednym punktem (każdy diagram Delaunay musi mieć co najmniej jedną krawędź więc minimum punktów na grupę to dwa). Przypadki z mniejszą ilością punktów niż cztery rozpatrywane są osobno.

Dodatkowo przy wykonywaniu takiego algorytmu warto najpierw posortować rozpatrywane punkty po koordynacie x, przypadki równości rozstrzygać sortując po koordynacie y i odrzucaniu punktów, które się pokrywają. Dzięki temu każda kolejna operacja dzielenia jest o stałym czasie. Rozpatrując krok łączenia dwóch obliczonych triangulacji, może zdarzyć się potrzeba usunięcia kilku krawędzi L - L i R - R lub dodania krawędzi L - R, jednak nigdy nie zostaną dodane nowe krawędzi L - L lub R - R, na co dowód umieszczony jest w artykule, z którego pochodzi ten algorytm.

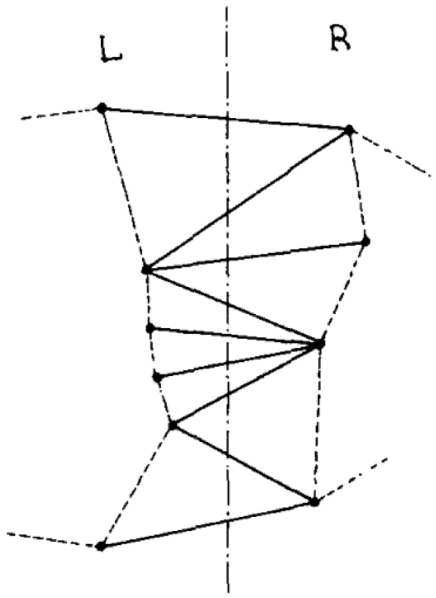
Niezbędnie ważnym dla działania algorytmu jest metoda *inCircle*. Posiada cztery argumenty, będące czterema unikalnymi punktami na badanej płaszczyźnie. Zwraca ona prawdę, gdy tylko punkt D jest we wnętrzu regionu płaszczyzny, który jest ograniczony przez okrąg zawierający ABC i leżąc na lewo od niego. Implikuje to, że D powinno leżeć wewnątrz takiego okręgu, gdy A, B i C tworzą trójkąt zorientowany przeciwnie do ruchu wskazówek zegara, a na zewnątrz, gdy tworzą trójkąt o orientacji zgodnej ze wskazówkami zegara. W przypadku gdzie A, B, C i D leżą na tym samym okręgu funkcja zwraca fałsz.

Dodatkowo potrzebny jest także predykat oznaczony jako $CCW(A, B, C)$ używany często do sprawdzenia, czy punkt X leży po lewej bądź prawej linii krawędzi e. Zwraca on prawdę, wtedy kiedy A, B i C tworzą trójkąt zorientowany przeciwnie do ruchu wskazówek zegara. Jego definicja widnieje poniżej na równaniu 1.

$$\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} > 0 \tag{1}$$

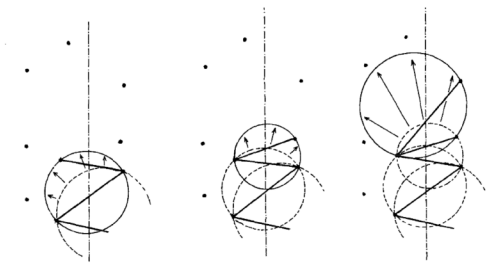
Zgodnie z przyjętą definicją krawędzie L - R muszą przecinać linie równoległą do osi y umieszczoną na określonym x. Trzeba tutaj założyć, że dowolne dwie krawędzie sąsiadujące w kolejności Y mają wspólny wierzchołek. Trzeci bok definiowanego przez nich trójkąta to krawędź L - L lub R - R. W literaturze spotykane jest oznaczenie bazowej krawędzi przecinającej ustalony x mianem *basel*, pamiętając przy tym, że jest ona skierowana od prawej do lewej. Kolejną krawędzią wytworzoną przy działaniu algorytmu będzie krawędź idąca od lewego końca *basel* do któregoś z punktów z grupy R leżącego nad nim lub analogicznie od prawego końca do punktu z grupy L ponad nim, mogą być oznaczone odpowiednio *lcand* i *rcand*.

Aby zobrazować działanie tego algorytmu, autorzy opisują, że należy wyobrazić sobie okrąg, który pnie się w górę, przeplatając pomiędzy L i R. W efekcie otrzymujemy krawędzie przecinające ustalony X. Co za tym



Rysunek 1: Poglądowa struktura krawędzi L - R [3].

idzie, gdy mamy koło opisujące trójkąt określony przez basel i poprzednią przecinającą krawędź. Należy kontynuować przekształcanie tego okręgu w inne okręgi mające *basel* jako cięciwę, ale leżące dalej w półpłaszczyźnie powyżej podstawy. Jest to korzystne, bo istnieje tylko jeden stopień swobody, ponieważ środek okręgu jest ograniczony do położenia na symetralnej cięciwy *basel* widoczne na rysunku 2. Okręgi będą się rozszerzały, aż napotkany zostanie kolejny punkt należący do L albo R, z wyjątkiem sytujacji gdzie *basel* jest już najwyższą możliwą krawędzią dla zbiorów L i R, co da początek nowemu trójkątowi z okręgiem opisanym, który też będzie transformowany aż do znalezienia następnego punktu. Nowa krawędź L - R tego trójkąta jest następną poprzeczną krawędzią wyznaczoną przez ciało głównej pętli algorytmu. Kandydat *lcand* jest obliczany tak, aby jako miejsce docelowe miał pierwszy punkt L napotkany w tym procesie a *rcand* pierwszy punkt R. Ostatecznie test wybiera punkt spośród tych dwóch, który był napotkany jako pierwszy. Iterację rozpoczyna się od znalezienia pierwszej możliwej krawędzi przecinającej L i R.



Rysunek 2: Poglądowa wizualizacja działania tworzenia krawędzi L - R [3].

3. Pseudokod algorytmu

Pseudokod algorytmu triangulacji Delauney przystosowany do zastosowania w różnych językach programowania jest przedstawiony na rysunku 3. Warto zwrócić uwagę na fakt, że jego złożoność obliczeniowa, wynoszącą $O(nlogn)$ to kolejna wskazówka, że jest on w grupie algorytmów typu "dziel i rządź" takich jak sortowanie przez kopcowanie, scalanie i sortowanie szybkie. Sprawia to, ze sam trzon jego struktury wydaje się znajomy dla programistów.

```
PROCEDURE Delaunay [S] RETURNS [le, re]:
  IF |S| = 2 THEN
    |Let s1, s2 be the two sites, in sorted order. Create an edge a from s1 to s2:|
    a ← MakeEdge[ ]; a.Org ← s1; a.Dest ← s2; RETURN [a, a.Sym]
  ELSIF |S| = 3 THEN
    |Let s1, s2, s3 be the three sites, in sorted order.|
    |Create edges a connecting s1 to s2 and b connecting s2 to s3:|
    a ← MakeEdge[ ]; b ← MakeEdge; Splice[a.Sym, b];
    a.Org ← s1; a.Dest ← b.Org ← s2; b.Dest ← s3;
    |Now close the triangle:|
    IF CCW[s1, s2, s3] THEN c ← Connect[b, a]; RETURN [a, b.Sym]
    ELSIF CCW[s1, s3, s2] THEN c ← Connect [b, a]; RETURN [c.Sym, c]
    ELSE |The three points are collinear:| RETURN [a, b.Sym] FI
  ELSE ||S| ≥ 4. Let L and R be the left and right halves of S.|
    [ldo, ldi] ← Delaunay[L]; [rdi, rdo] ← Delaunay[R];
    |Compute the lower common tangent of L and R:|
    DO
      IF LeftOf[rdi.Org, ldi] THEN ldi ← ldi.Lnext
      ELSIF RightOf[ldi.Org, rdi] THEN rdi ← rdi.Rprev
      ELSE EXIT FI
    OD;
    |Create a first cross edge basel from rdi.Org to ldi.Org:|
    basel ← Connect[rdi.Sym, ldi];
    IF ldi.Org = ldo.Org THEN ldo ← basel.Sym FI;
    IF rdi.Org = rdo.Org THEN rdo ← basel FI;
    DO |This is the merge loop.|
      |Locate the first L point (lcand.Dest) to be encountered by the rising bubble,|
      |and delete L edges out of basel.Dest that fail the circle test.|
      lcand ← basel.Sym.Onext;
      IF Valid[lcand] THEN
        WHILE InCircle
          [basel.Dest, basel.Org, lcand.Dest, lcand.Onext.Dest]
          DO t ← lcand.Onext; DeleteEdge[lcand]; lcand ← t OD
      FI;
      |Symmetrically, locate the first R point to be hit, and delete R edges:|
      rcand ← basel.Oprev;
      IF Valid[rcand] THEN
        WHILE InCircle
          [basel.Dest, basel.Org, rcand.Dest, rcand.Oprev.Dest]
          DO t ← rcand.Oprev; DeleteEdge[rcand]; rcand ← t OD
      FI;
      |If both lcand and rcand are invalid, then basel is the upper common tangent:|
      IF NOT Valid[lcand] AND NOT Valid[rcand] THEN EXIT FI;
      |The next cross edge is to be connected to either lcand.Dest or rcand.Dest.|
      |If both are valid, then choose the appropriate one using the InCircle test:|
      IF NOT Valid[lcand] OR
        (Valid[rcand] AND
          InCircle[lcand.Dest, lcand.Org, rcand.Org, rcand.Dest])
        THEN |Add cross edge basel from rcand.Dest to basel.Dest:|
          basel ← Connect[rcand, basel.Sym]
        ELSE |Add cross edge basel from basel.Org to lcand.Dest:|
          basel ← Connect[basel.Sym, lcand.Sym]
      FI
    OD;
    RETURN [ldo, rdo]
  FI
END Delaunay.
```

Rysunek 3: Pseudokod algorytmu obliczającego triangulacje Delaunay [3].

Samo działanie opisane jest w poprzednim rozdziale, a także jako przypisy w tym kodzie. Poniżej umieszczono także definicje wybranych pomocniczych funkcji $RightOf$, $LeftOf$ i $Valid$.

Algorytm 1: Definicje funkcji RightOf i LeftOf

input : X - współrzędne badanego punktu, e - zorientowana krawędź e w formie obiektu zawierającego punkty Org i $Dest$ odpowiadające odpowiednio puntkowi poczatkowemu i końcowemu
output: prawda lub fałsz

RightOf(X, e) **return** CCW$(X, e.Dest, e.Org)$
LeftOf(X, e): **return** CCW$(X, e.Org, e.Dest)$

Reszta definicji używanych funkcji zostanie pominięta ze względu na ich obszerność. Dostępnę są one w pracy "Primitives for the manipulation of general subdivisions and the computation of Voronoi" wraz z dowodami ich poprawności [3].

Algorytm 2: Definicja funkcji Valid

input : e, $basel$ - badana krawędź bazowa w takim samym formacie co e
output: prawda lub fałsz

Valid$(e, basel)$: **return** RightOf$(e.Dest, basel)$

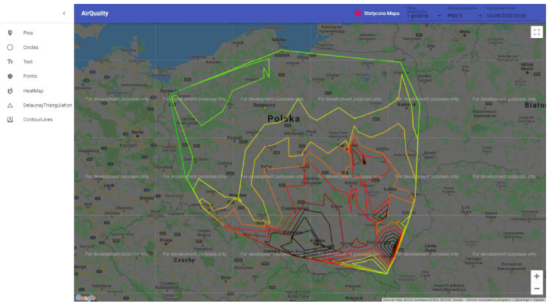
4. Program testowy

Na potrzeby przeprowadzenia testów przygotowana została aplikacja webowa typu "mashup" [4]. Wykorzystany stos technologiczny, często określany skrótem MERN [5] opiera się na technologiach:

- MongoDB – rozwiązanie bazodanowe,
- Express.js – serwerowy szkielet programistyczny,
- React – biblioteka służąca do tworzenia interfejsów użytkownika,
- Nodejs - środowisko uruchomieniowe oparte na silniku V8 firmy Google.

Uzasadnieniem takiego wyboru może być popularność tych technologii. Warunki eksperymentów starano się doprowadzić do jak najbliższych rzeczywistości. Zgodnie z ankietą przeprowadzoną na stronie StackOverflow [7], JavaScript ósmy rok z rzędu okazał się najpopularniej używanym językiem wśród respondentów, otrzymując 69.7%. React.js uplasował się na drugim miejscu z 36.8% spośród wszystkich szkieletów webowych. W kategorii „różnych" najczęściej używanym narzędziem okazał się Node.js z udziałem 51.9%. Zapewniło to środowisko bliskie docelowej implementacji.

Ważnym elementem aplikacji jest też oczywiście baza danych NoSQL MongoDB. Nierelacyjne systemy bazy danych coraz częściej znajdują wykorzystanie w SPA (ang. Single Page Application). Według powyższej ankiety używa go 26.7% badanych, więc skłania to do sprawdzenia wydajności takiego rozwiązania oraz poznania wyzwań, jakie towarzyszą implementacjom przy użyciu takiej technologii.



Rysunek 4: Główny widok programu testowego

Główny widok programu jest zaprezentowany na rysunku 4. Dane po pobraniu z bazy przekazywane są do algorytmu w formię współrzędnych i wartości wybra-

nego zanieczyszczenia. Decyzja o okresie agregacji nie wpływa na ilość danych przekazanych do algorytmu, co za tym idzie, nie zwiększa czasu jego wykonania. Wynika to z tego, że agregacja danych i wybór danego związku chemicznego wykonywane są już na poziomie bazy danych, dzięki temu każdy punkt na mapie ma przypisaną jedną uśrednioną wartość danego zanieczyszczenia a ilość stacji pomiarowych to ilość punktów używanych do stworzenia siatki na mapie. Do algorytmu trafiają współrzędne każdej stacji. Zgodnie z opisem algorytmu każda ze stacji staje się wtedy wierzchołkiem trójkąta. Kolory wypełnienia widoczne na mapie obliczane są na podstawie średniej arytmetycznej z zanieczyszczenia w każdym wierzchołku na zasadzie Zielony-Czerwony-Czarny, gdzie od zielonego do czerwonego mamy zakres wartości do limitu ustawionego przez WHO albo inny podmiot, a przyciemnianie czerwonego aż do czarnego zarezerwowane jest do wartości przekraczających limit.

5. Implementacja wizualizacji

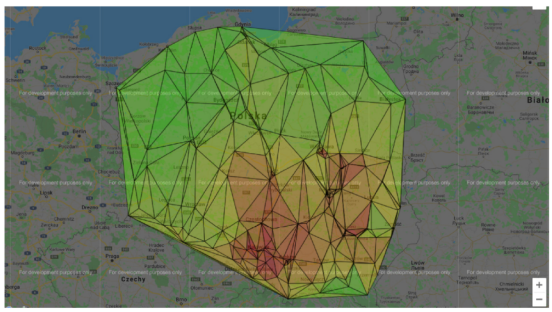
Dla wizualizacji z triangulacją Delauney wykorzystana została dodatkowa biblioteka „faster-delaunay". Została ona oparta na algorytmie opisanym w poprzedniej części tej pracy. Jest to także najszybsza dostępna biblioteka, jaka została sprawdzona. Ze względu na format argumentu, jaki przyjmuje funkcja dostarczana przez tę bibliotekę, dane wymagały przetworzenia.

Najważniejszym krokiem przy implementacji wizualizacji na mapie było prze-mapowanie danych pomiarowych na tablice zawierające współrzędne punktów pomiarowych. Wartości współrzędnych pomnożone zostały przez milion, aby uniknąć, problemów związanych z operacjami na liczbach zmiennoprzecinkowych dla tej biblioteki. Przed wykorzystaniem gotowe współrzędne wierzchołków nowo powstałych trójkątów zostaną podzielone ponownie przez milion, aby można było je nałożyć na mapę zgodnie z ich prawdziwym położeniem. Następnie ponownie przyporządkowano wartości do punktów pomiarowych o konkretnych współrzędnych. Nastąpiło także wyliczenie średniej z danych z wierzchołków trójkąta, aby można było określić wypełniający kolor. Dzięki tak przygotowanym danym samo wyświetlenie ich na mapie nie stanowi większego problemu. Wysyłane są przy użyciu biblioteki do obsługi map i zwracana zostaje finalna mapa ze zdefiniowanymi figurami.

6. Wizualizacje

Oceniając tą wizualizacje warto pamiętać o celu wykorzystywania map statystycznych. W pracy pod tytułem "Graficzna prezentacja danych statystycznych Wykresy, mapy, GIS" umieszczona została bardzo trafna definicja: "Mapy statystyczne, w odróżnieniu od tabel statystycznych, czy wykresów statystycznych, pozwalają na jednoczesne zaprezentowanie przestrzennego rozmieszczenia opisywanego zjawiska (jego występowania) i jego wartości. Ich zadaniem nie jest pokazywanie precyzyjnej wartości zjawiska dla danego obiektu (choć jest to często możliwe do odczytania z mapy), tylko jego przybliżonej wartości wraz z relacjami przestrzennymi pomiędzy poszczególnymi obiektami" [6].

Mając na uwadze cel, który powinna taka reprezentacja osiągnąć po wykonaniu kilku eksperymentów nakładka wydaje się ciekawą możliwością do wykorzystania przy obrazowaniu trendów na większym terenie np. w skali krajowej. Nie wyklucza to możliwości wykorzystania jej na obszarze np. miastowym, co przedstawione jest poniżej (rysunek 6). Na rysunku 5 umieszczona została mapa, wygenerowana na podstawie algorytmu, oznaczona kolorami zgodnie z opisem w rozdziale 5. Na pierwszy rzut oka mapa przypomina kartogram, i rzeczywiście techniki wykonania są podobne, przedstawia także uśrednione wartości dla zanieczyszczenia z każdego wierzchołka danego trójkąta co dalej utwierdza to podobieństwo. Dokładność mapy ściśle zależy od ilości dostępnych punktów pomiarowych. Im mniejszy rozmiar każdego trójkąta, tym więcej szczegółów i trendów na mniejszym terenie możemy zobaczyć.



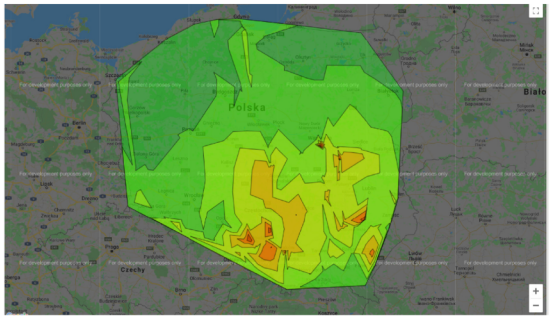
Rysunek 5: Wizualizacja oparta o Triangulacje Delaunay.

Na teren testowy wybrane zostało miasto Kraków i okolice. Zachowano kolorowanie na podstawie limitów, aby nie wprowadzać nieścisłości przy interpretacji.

Rysunek 6: Triangulacja Delaunay na małym obszarze.

Widać jednak, że są lepsze alternatywy dla przekazywania takich informacji. Ilość punktów pomiarowych, na obszarze miasta, nie była wystarczająca do uzyskania tak dobrego efektu co na skalę krajową. Mapa pozostała czytelna, lecz nie widać na niej trendów tak jasno, jak na poprzedniej. Warto tutaj zaznaczyć, że nie ma wizualizacji idealnej, która będzie perfekcyjna dla każdego przypadku. Trzeba iść na kompromisy i dobierać je dla konkretnych sytuacji.

Nie można jednak rozpatrywać takiej wizualizacji nie poruszając innych. W ramach porównania opisana zostanie także wizualizacja oparta o izolinie. Jest to znana i sprawdzona forma reprezentacji danych na mapie.

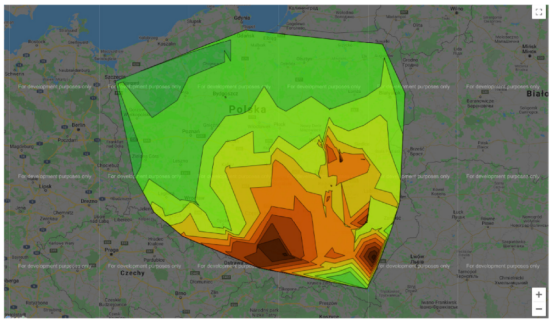


Rysunek 7: Przykładowy widok mapy z izoliniami.

Powyżej umieszczony został przykład wygenerowanej mapy (rysunek 7). Aby oddać wartości, skorzystano z kolorów wypełniających pole w wielokątach utworzonych przez izolinie. Tak samo, jak w poprzednich nakładkach kolory odpowiadają limitom zanieczyszczeń, a nie skali od najmniejszej wartości pomiaru na widocznym terenie do największej. Nie skorzystano też z żadnego wygładzania krawędzi, aby zachować jak najdokładniejsze oddanie rzeczywistości.

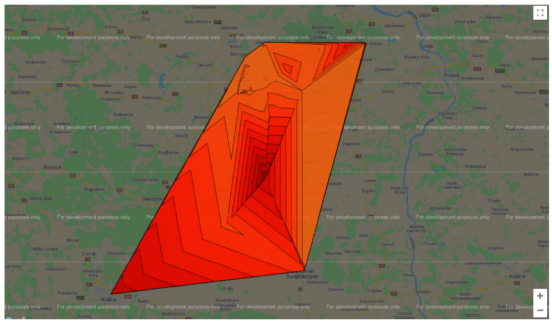
Ulepszenie wyglądu takiej mapy wymagałoby więcej punktów pomiarowych. Jest to cecha wspólna z poprzednią wizualizacją. Pewien poziom wygładzania mógłby ulepszyć wizualne aspekty tej mapy, jednak tą, decyzje najlepiej podjąć już na etapie konkretnej implementacji.

Przedstawione na tej mapie dane są spójne z poprzednimi. Można jednak przedstawić jeszcze dodatkowy obraz wygenerowanej mapy na podstawie poziomów innego zanieczyszczenia. Ta nakładka wygląda zdecydowanie lepiej wizualnie niż poprzednia. Jest to swoistego rodzaju najlepszy przypadek z dostępnych danych. Pokazuje to potencjał tego podejścia (rysunek 8).



Rysunek 8: Przykładowy widok mapy z izoliniami.

W następnej części analizowana jest przydatność tej mapy przy obszarach miejskich. Mniejsza ilość czujników i fakt, że wartości zanieczyszczeń są na podobnych poziomach, utrudnia wygenerowanie przydanej mapy. Tak jak w poprzednim eksperymencie zadecydowano zachować te same kolory ze względu na spójność danych.



Rysunek 9: Przykładowy widok mapy z izoliniami.

Jak widać poniżej ciężko odczytać jakieś wartościowe informacje z tej mapy. Z prezentowanych jak do tej pory wizualizacji ta prezentuje się najgorzej (rysunek 9). Nie nadaje się w tym stanie do przestawiania danych miejskich, chyba że mamy wystarczającą liczbę punktów pomiarowych oraz kolory zostaną wyskalowane zgodnie z najmniejsza i największą badaną wartością. Przy takiej reprezentacji trzeba umieścić legendę dotyczącą skali, aby nie wprowadzać użytkowników w błąd. Pozwoli to na znalezienie punktów problematycznych na danym obszarze.

7. Testy wydajnościowe

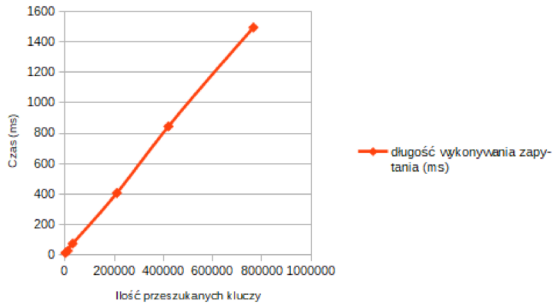
Wykorzystanie rozwiązań zaprezentowanych w tej pracy będzie wiązało się ściśle z szybkością działania. Wymagania współczesnego użytkownika co do responsywności aplikacji internetowych. Platforma testowa wyposażona była w procesor Intel i7 4720HQ o taktowaniu 3,60 GHz oraz 16 GB pamięci ram DDR3L CL9 o prędkości 1600MHz. Pierwszy aspekt aplikacji poddany analizie to, jak sprawdza się model bazodanowy, stworzony na potrzeby tego badania. Aktualny stan kolekcji zawiera 1373039 obiektów pomiarów i zajmuje, po wyeksportowaniu, do pliku BSON 195 MB. Czasy prezentują się następująco:

Tabela 1: Czasy agregacji dla różnych okresów uśredniania

Okres uśredniania (h)	Średnia czasu wykonania zapytania z pięciu prób (ms)	Liczba dokumentów zebranych do uśredniania (ms)	Ilość przeszukanych kluczy (n)
1	14.22	250	1238
8	28.39	2239	11160
24	75.82	6239	31064
168	408.33	42274	210501
336	845.62	84365	420144
672	1495.47	157155	765429

Wybrano takie okresy, ponieważ wydawały się zbliżone do tego, jak mogłyby być wykorzystane te dane i wizualizacje. Czasy te są akceptowalne i pozwalają na wygodne korzystanie z aplikacji bez większych okresów

oczekiwania. Warto jeszcze zauważyć, że użyto unikalnych indeksów zawierających pole daty, aby maksymalnie przyspieszyć te operacje.



Rysunek 10: Triangulacja Delaunay na małym obszarze.

Kolejny aspektem poddanym analizie wydajnościowej będzie przygotowanie danych do generowania samych wizualizacji. Przygotowanie będzie oznaczać czas potrzebny do wykonania potrzebnych algorytmów albo obróbki danych, jeżeli nie można ich wykorzystać bezpośrednio z bazy. Z tego powodu dane będą dotyczyły dwóch wizualizacji mapy z izoliniami i opartej o triangulacje Delauney. Sam czas odpowiedzi Google Maps API będzie wyłączony z tego badania ze względu na zbyt wiele zmiennych wpływających na długość oczekiwania takich jak lokalizacja, prędkość sieci, obciążenie serwerów w danym momencie itp. na które badający nie ma wpływu.

Tabela 2: Czasy agregacji dla różnych okresów uśredniania

Wizualizacja	Czas wykonywania algorymów (ms)	Całkowity czas przygotowania danych (ms)
Triangulacja Delauney	9	24
Izolinie	16	17

8. Mocne i słabe strony

Rozpatrując mocne i słabe strony przedstawionego rozwiązania możemy zidentyfikować:

Mocne strony:

- Ciekawa alternatywa dla innych map statystycznych przy odpowiednio dużej ilości czujników.
- Nowatorskość rozwiązani może zachęcać do zapoznania się z przekazywanymi informacjami.
- Blisko stuprocentowe pokrycie mapy Polski uśrednionymi danymi, mało pustych obszarów bez żadnych danych.

Słabe strony:

- Silna zależność dokładności od ilości punktów pomiarowych.
- Duże uśrednienie powoduje niedokładność danych przy obszarach z małym zagęszczeniem stacji pomiarowych.

W ramach porównania dla wizualizacji opartych o izolinie można wymienić:

Mocne strony:

- Znajomy format wizualizacji, duże prawdopodobieństwo, że odbiorca będzie mógł korzystać z dotychczasowej wiedzy przy analizie.
- Dobrze widoczne uogólnione trendy na obszarach z dużą ilością punktów pomiarowych.
- Tak jak poprzednia wizualizacja, prawie stuprocentowe pokrycie Polski uśrednionymi danymi, mało pustych obszarów bez żadnych danych.

Słabe strony:

- Najgorsza z przedstawionych w reprezentacji obszarów o niewielkiej ilości stacji pomiarowych.
- Nakładające się kolory zasłaniają kluczowe elementy mapy. Ten efekt można zniwelować kolorując same izolinie.

9. Wnioski

Temat zanieczyszczeń powietrza będzie towarzyszył społeczeństwu akademickiemu przez dłuższy czas. Jego obszerność i bezprecedensowe znaczenie daje bardzo duże możliwości dopasowania go do różnych dyscyplin naukowych. Przedstawione w tym artykule problemy i ich analiza dostarczają narzędzi do lepszego zrozumienia problemu stanu środowiska, jak i zwiększaniu świadomości społecznej. Wykorzystywanie wizualizacji przy użyciu niespotykanych technik bądź algorytmów może przynieść nowe ciekawe metody przekazywania informacji. Wizualizacja będąca tematem tej pracy zdecydowanie spełnia oczekiwania, jeżeli chodzi o prezentowanie trendów w danych i przydatność. Ciekawym rozszerzeniem tych badań mogłoby być zbadanie innych algorytmów opartych na diagramach Vornoia do wygenerowania map przedstawiających zanieczyszczenia powietrza.

Literatura

[1] J. Lelieveld,Cardiovascular disease burden from ambient air pollution in Europe reassessed using novel hazard ratio functions, Eur Heart J, t. 40, nr 20, s. 1590–1596, maj 2019, doi: 10.1093/eurheartj/ehz135.

[2] D.W. Dockery, C.A. Pope, Acute respiratory effects of particulate air pollution, Annual review of public health, t. 15, nr 1, s. 107–132, 1994.

[3] L. Guibas, J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi, ACM Trans. Graph., t. 4, nr 2, s. 74–123, kwi. 1985, doi: 10.1145/282918.282923.

[4] D. Fichter, What is a mashup, Library Mashups. Exploring new ways to deliver library data. Medford, NJ: Information Today, Inc, 2009.

[5] V. Subramanian, Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. Apress, 2017.

[6] M. Pieniążek, B. Szejgiec, M. Zych, A. Ajdyn, G. Nowakowska, Graficzna prezentacja danych statystycznych Wykresy, mapy, GIS. Warszawa: Główny

Urząd Statystyczny Departament Badań Regionalnych i Środowiska, 2014.

[7] „Stack Overflow Developer Survey 2020", Stack Overflow. https://insights.stackoverflow.com/survey/2020/?utm˙source=social-share&utm˙medium=social&utm˙campaign=dev-survey-2020, (udostępniono cze. 25, 2020).

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 412–416
Received: 21 September 2020
Accepted: 20 October 2020

Comparison of methods and tools for generating levels of details of 3D models for popular game engines

Porównanie metod i narzędzi generowania poziomów szczegółowości modeli 3D dla popularnych silników gier

Tomecki Michał*

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

Abstract

Computer games have come a long way since their inception using increasingly advanced graphics. With so many models present on the screen at the same time, simplified models are needed that will be replaced when they are further away from the camera in the game. The aim of the study was to compare the tool for automatic generation of model detail levels in Unity and Unreal Engine. The article presents the results of the equipment load test and the time needed for generation, as well as a survey checking the opinion of people.

Keywords: OD; 3D models

Streszczenie

Gry komputerowe przeszły długą drogę od momentu swojego powstania korzystając z coraz bardziej zaawansowanej grafiki. Przy tak dużej ilości modeli obecnych jednocześnie na ekranie potrzebne są uproszczone modele, które będą podmieniane gdy będą dalej od kamery w grze. Celem badania było porównanie narzędzia służącego do automatycznego generowania poziomów szczegółowości modeli w Unity oraz Unreal Engine. W artykule zostały zaprezentowane wyniki testu obciążenia sprzętu i czasu potrzebnego do generacji oraz ankiety sprawdzającej opinię ludzi.

Słowa kluczowe: poziomy szczegółowości modeli; modele 3D

*Corresponding author

Email address: **michal.tomecki@pollub.edu.pl** (M.Tomecki)

1. Wstęp

Niniejszy dokument opracowano dla autorów Gry komputerowe to obecnie jeden z najpopularniejszych sposobów wykorzystania komputerów przez młodych ludzi na całym świecie. Przeszły one długą drogę od momentu swego powstania. Zaczynając od gry Pong po raz pierwszy uruchomionej w rok 1972 przez firmę Atari poprzez gry typu Minecraft do gier takich jak Wiedźmin 3 czy wszystkim znane gry z serii Grand Theft Auto [1]. Z prostych gier w dwóch wymiarach mających kilku bitową grafikę dotarliśmy do gier w trzech wymiarach z grafiką tak realistyczną, że można ją pomylić z rzeczywistością..

Na samym początku swojej historii gry były pisane przez bardzo wąskie grono osób. Wraz z rozwojem języków programowania pisanie gier stawało się coraz łatwiejsze. Obecnie na rynku dostępne są darmowe silniki gier takie jak Unity Game Engine oraz Unreal Engine. Upraszczają one znacząco pracę osoby tworzącej daną grę poprzez udostępnianie wiele różnych potrzebnych funkcjonalności [2]. W środowisku Unreal Engine dostępne są tzw. schematy(blueprints) [3]. Pozwalają one na programowanie gier używając graficznego interfejsu wykorzystującego prostokąty połączone ze sobą liniami. W ten sposób potencjalny twórca gry nie musi znać nawet języka programowania i dalej być w stanie stworzyć swoją wymarzoną grę. Z powodu tak łatwego dostępu do możliwości tworzenia swojej własnej gry, twórcy muszą sprawić aby ich gra się w jakiś sposób wyróżniała od konkurencji na rynku. Wielu z nich wykorzystuje do tego wspaniale wyglądające modele graficzne. Niestety obciążają one znacząco system w trakcie wyświetlania ich na ekranie. W przypadku, w którym będzie ich dużo jednocześnie obecnych na ekranie gra może nie działać płynnie, a tego osoby, które w nią grają nie chcą widzieć. Dlatego większość twórców używa algorytmów teselacji. Służą one do dynamicznego podmieniania modeli obiektów obecnych w świecie gry [7]. Wraz z oddalaniem się kamery od obiektu jego model zostaje podmieniony na bardziej uproszczony. Pozwala to na zmniejszenie obciążenia systemu, co umożliwia płynniejsze działanie gry na danym systemie.

Większość obecnych na rynku silników gier posiada w mniejszym lub większym stopniu dostęp do narzędzia służącego do automatycznego generowania poziomów szczegółowości modeli 3D. Jedyne co twórca gry musi zrobić to wskazanie wybranego obiektu, dla którego mają zostać wygenerowane poziomy szczegółowości oraz ile ich ma być, jeśli mu nie pasuje domyślna ilość. Narzędzia te same przypiszą wygenerowane modele do wybranego modelu oraz wybiorą odległość, przy której mają one zostać podmienione. Nasuwa się jednak pytanie czy te modele wygenerowane przez różne środowiska będą na takim samym poziomie jakości czy może jednak będą się od siebie znacznie różnić.

2. Siliniki gier

Obecnie na rynku znajduje się wiele różnych silników gier. Są bardzo rozbudowanymi środowiskami służącymi do pisania głównie gier. Mają one zestaw narzędzi znacznie ułatwiający pisanie nowym w tej dziedzinie użytkownikom. Jedne z najpopularniejszych silników to Unity Game Engine oraz Unreal Engine.

Unity Game Engine jest to silnik służący do tworzenia gier zarówno w 2D jak i w 3D. Powstał on w 2005 roku jako silnik do pisania gier na system OS X. Następnie został rozbudowany i obecnie wspiera pisanie gier na ponad 25 różnych platform. Skrypty używane do kontrolowania rzeczy, które dzieją się wewnątrz niego na początku jego istnienia były pisane zarówno w stworzonym przez firmę Microsoft języku C# jak i w zmodyfikowanym języku JavaScript zwanym przez użytkowników UnityScript [4]. W roku 2017 UnityScript stracił wsparcie i od tego czasu jednym językiem używanym wewnątrz Unity jest C#.

Elementy używane wewnątrz silnika, takie jak skrypty, obiekty, pliki graficzne itd są umieszczone wewnątrz folderu "Assets" [5]. Programista może sam je tam ręcznie umieścić lub pobrać z Unity Asset Store. Jest to wbudowany w Unity sklep gdzie można znaleźć zarówno płatne jak i bezpłatne elementy wykonane przez innych użytkowników.

Unreal Engine jest to silnik gier wyprodukowany przez przedsiębiorstwo Epic Games [6]. Podobnie jak Unity Game Engine jest to duże, rozbudowane środowisko do tworzenia gier. Po raz pierwszy środowisko na rynku pojawiło się w 1998 roku. Wstępnie był on tylko wykorzystywany w grach typu FPS(First Person Shooter). Wraz z rozbudową silnika zwiększono jego zastosowanie, nie był już on ograniczony tylko do jednego typu gier. Podobnie jak Unity Game Engine umożliwia on tworzenie gier na wiele różnych platform. Unreal Engine został napisany w języku C++, ale skrypty kontrolujące co się dzieje wewnątrz gry mogą być pisane nie tylko w tym języku. Dostępne są również tak zwane schematy(blueprints). Pisanie w nich odbywa się w sposób graficzny poprzez połączenie liniami prostokątów, które odpowiadają wcześniej automatycznie wygenerowanym funkcjom takim jak np "Skok". Jest to bardzo duże ułatwienie dla nowych w tej dziedzinie użytkowników. Przez brak konieczności wcześniejszego znania jakiegokolwiek języka programowania jest on dość popularnym rozwiązaniem .

3. Narzędzia służące do automatycznego generowaniapoziomów szczegółowości modeli 3D

Praktycznie każdy obecny na rynku silnik gier posiada mniej lub bardziej rozbudowane narzędzia służące do automatycznego generowania poziomów szczegółowości modeli 3D. Wyjątkami nie są Unity Game Engine ani Unreal Engine. Oba silniki są jednymi z najpopularniejszych na rynku, więc bardzo dziwny by był brak takiej funkcjonalności. W pierwszym z nich jest to realizowane przez społeczność użytkowników tego środowiska. We wbudowanym w nie Unity Asset Store można znaleźć wiele płatnych jak i bezpłatnych narzędzi, które w różnym stopniu dokładności i jakości wygenerują odpowiednie mniej szczegółowe modele dla wybranych obiektów. Nie jest to jedyne miejsce gdzie można takie narzędzia znaleźć, innym miejscem, w którym jest ich dość sporo jest platforma GitHub. W przeciwieństwie do swojego poprzednika twórcy Unreal Engine sami napisali narzędzie do automatycznego generowania poziomów szczegółowości modeli 3D. Jest ono wbudowane w silnik i dostępne w oknie ze szczegółami wybranego modelu.

Jednym z popularniejszych bezpłatnych narzędzi dostępnych w środowisku Unity jest AutoLOD. Dostępne z platformy GitHub narzędzie jest bardzo rozbudowane jak na projekt niekomercyjny. Funkcja generowania poziomów szczegółowości modelu jest dostępna z trzech różnych miejsc w programie. Dzięki obecnej w silniku Unity możliwości dodawania opcji do pasków narzędzi, AutoLOD wygląda jak opcja dostępna domyślnie w środowisku, a nie jak dodatek. Wartą wspomnienia funkcją jest też możliwość usuwania wcześniej wygenerowanych poziomów szczegółowości dla wybranego modelu. Więcej szczegółowych ustawień znajduje się w menu dostępnym z paska narzędzi. Umieszczone są tam między innymi takie opcje jak wybór ilości wygenerowanych poziomów szczegółowości jak i maksymalna ilość poligonów.

Jako, że w silniku Unreal Engine jest domyślnie wbudowane narzędzie służące do generowania poziomów szczegółowości użytkownik nie musi nic szukać. W momencie otworzenia okna ze szczegółami wybranego obiektu po prawej stronie ekranu będą widoczne dostępne ustawienia. W sekcji dotyczącej poziomów szczegółowości jest dostępne kilka gotowych ustawień przygotowanych przez twórców, jeśli użytkownik nie chce nic zmieniać, to wystarczy tylko wybranie gotowego zestawu, a narzędzie automatycznie wygeneruje odpowiednią ilość poziomów szczegółowości i wybierze pozostałe potrzebne ustawienia. Są one łatwo dostępne w tej samej sekcji na wypadek gdyby domyślne wartości nie były wystarczające.

4. Testy i ich metodologia

W celu dokonania prawidłowej oceny obu wcześniej wymienionych narzędzi zostały wybrane dwa sposoby testowania każdego z nich. Przygotowano dziesięć różnych obiektów 3D. Każdy z nich miał inną ilość werteksów, trójkątów i inne elementy charakterystyczne, na których było można zauważyć zmiany wywołane przez zmianę poziomu szczegółowości tego modelu. Dla każdego obiektu zostały wygenerowane dodatkowe trzy uproszczone modele. W poniższych tabelach (Tabela 1 oraz 2) przedstawiono ilość werteksów każdego modelu w obu środowiskach. Jak widać modele w środowisku Unreal Engine mają ich więcej niż te w Unity oraz każdy następny poziom szczegółowości ma prawie dokładnie o połowę mniej werteksów. W przypadku uproszczonych modeli w Unity Game Engine stosunek ilości werteksów uproszczonego modelu do bardziej szczegółowego jest różny dla każdego poziomu. Dla konsekwencji w obu środowiskach zostały one ustawione do

zmieniania się kiedy są w takiej samej odległości od kamery.

Tabela 1: Ilość werteksów modeli w środowisku Unreal Engine

Model	LOD0	LOD1	LOD2	LOD3
M1	2053134	1026564	513282	256638
M2	321088	160540	80268	40134
M3	418653	209322	104658	52326
M4	569505	284751	142374	71187
M5	691357	345675	172836	86415
M6	175896	87948	43974	21984
M7	1697952	848976	424488	212244
M8	445219	222608	111302	55647
M9	1289151	644592	322302	161148
M10	248610	124302	62148	31074

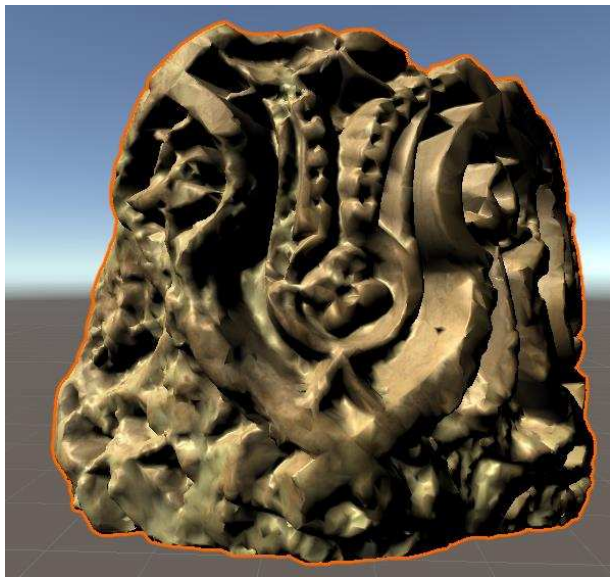
Tabela 2: Ilość werteksów modeli w środowisku Unity

Model	LOD0	LOD1	LOD2	LOD3
M1	356320	187377	100575	54585
M2	55039	28668	15158	8101
M3	73935	39818	21979	12262
M4	107261	60470	35080	20461
M5	123449	67193	37643	21339
M6	30762	16449	8949	4935
M7	288443	148270	77256	40480
M8	78990	42798	23861	13308
M9	218740	112210	58505	30883
M10	43234	23005	12550	6903

Pierwszy z nich polegał na porównania czasu potrzebnego na wygenerowanie takiej samej liczby poziomów szczegółowości tego samego modelu w obu środowiskach. Poza wspomnianymi wcześniej mierzonymi aspektami została sprawdzona i porównana liczba werteksów oraz trójkątów we wszystkich wygenerowanych modelach w obu środowiskach.

Drugi sposób służący do porównania obu narzędzi polegał na wykonaniu ankiety zawierającej zrzuty ekranu z wygenerowanymi poziomami szczegółowości modeli z obu środowisk. Następnie ankieta ta została wykorzystana do zbadania nią grupy ludzi i poznania ich opinii na temat aspektu wizualnego każdego modelu.

W ankiecie były umieszczone wszystkie dziesięć modeli w kolejności. Najpierw zostały pokazane wszystkie poziomy szczegółowości danego modelu obecne tuż przy kamerze w celu dokładnego pokazania wygenerowanego modelu (Rysunek 1). Następnie te same modele umieszczono w odległości, w której by były widoczne w grze (Rysunek 2). Na koniec z każdej serii zrzutów ekranów były umieszczone animacje pokazujące oddalanie się i przybliżanie do danego modelu. Pytania, które zadawano ankietowanym składały się z ogólnego pytania o wybranie środowiska, w którym ich zdaniem modele wyszły lepiej oraz z pytania o widoczność przez nich przejść między poszczególnymi poziomami szczegółowości każdego modelu na dołączonych animacjach.



Rysunek 1: Jeden z wykorzystanych w ankiecie modeli LOD3 tuż przy kamerze w Unity Game Engine

Rysunek 2: Jeden z wykorzystanych w ankiecie modeli LOD0 w odpowiedniej odległości od kamery w Unreal Engine

5. Wyniki testów

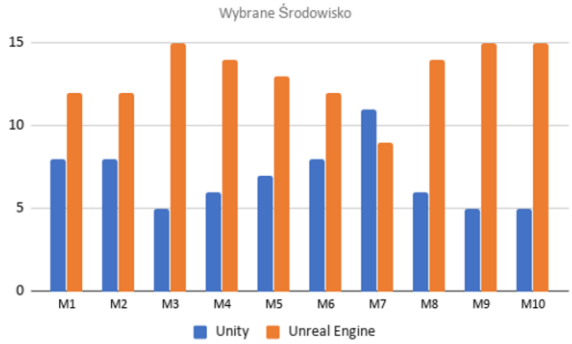
Pierwszy z przeprowadzonych testów, sprawdzający czas i obciążenie dał dość niespodziewane wyniki. Zaczynając analizę wyników od Unreal Engine można zauważyć, że podczas generowania poziomów szczegółowości poszczególnych modeli, dla obiektów, które miały więcej werteksów, narzędzie obecne domyślnie w tym silniku potrzebowało więcej czasu niż dla tych modeli, które miały mniej werteksów. Podobną obserwację można było zauważyć również na obciążeniu sprzętu, który był mocniej wykorzystywany podczas pracy nad bardziej skomplikowanymi obiektami.

Dokonując analizy tego samego testu w środowisku Unity Game Engine dostaje się niespodziewany rezultat. Otóż wykorzystując narzędzie AutoLOD do generowania poziomów szczegółowości modeli, dokonano obserwacji, że w momencie wybrania przycisku odpowiedzialnego za rozpoczęcie generacji nie pojawia się żadne okienko z paskiem pokazującym postęp generacji, jak to jest obecne w drugim środowisku, lecz tworzone są siatki z mniejszą szczegółowością od razu dostępne do użytku. ustawione W tym samym czasie do modelu zostaje dołączony odpowiedni komponent odpowiedzialny za zmianę poziomów szczegółowości gdy ten będzie się znajdował w odpowiedniej odległości od kamery. Wykonanie tego samego testu ponownie w celu sprawdzenia obciążenia sprzętu, na którym wykonywano badania, pokazało, że wykresy wyświetlające wydajność komputera nie zmieniają się.

Przeprowadzona ankieta została wypełniona przez dwadzieścia osób z czego osiemnaście z nich to byli mężczyźni, a reszta kobiety. Szesnaścioro z nich to były osoby pomiędzy 20 a 25 rokiem życia. Wyniki dla pierwszego pytania po zestawie zrzutów ekranów z wygenerowanymi modelami zostały przedstawione poniżej (Tabela 3).

Tabela 3: Wyniki ankiety sprawdzającej preferowane środowisko

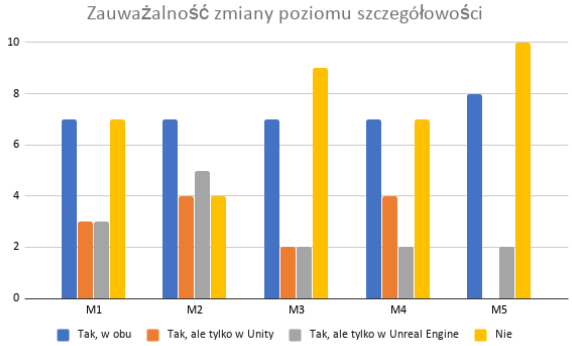
Model	Unity (osoby)	Unreal Engine (osoby)
M1	8	12
M2	8	12
M3	5	15
M4	6	14
M5	7	13
M6	8	12
M7	11	9
M8	6	14
M9	5	15
M10	5	15
Średnia	6,9	13,1



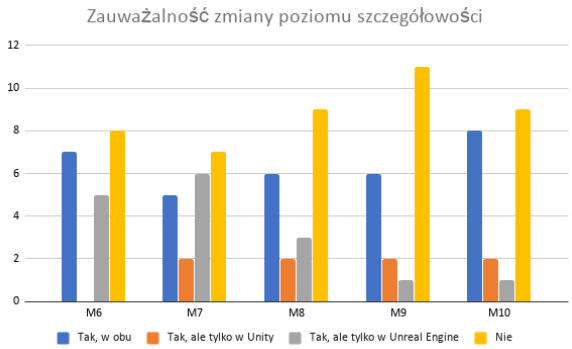
Rysunek 3: Wykres przedstawiający wybrane przez badanych środowisko

Jak to jest widoczne na powyższym rysunku (Rysunek 3) zdecydowanie większa część ludzi wybrała środowisko Unreal Engine. Ponadto tylko jeden model (M7) był uważany za lepszy w środowisku Unity Game Engine. W przedstawionej wcześniej tabelce (Tabela 1) poza dokładnymi wynikami ankiety widać również wyliczoną na ich podstawie średnią(ostatni rząd). Dzięki niej wiadomo, że średnio 65.5 % badanych wybierało Unreal Engine.

Drugim elementem badanym w przeprowadzonej ankiecie była widoczność zmian poszczególnych poziomów szczegółowości danych modeli na dołączonych po zestawie zrzutów ekranu gifach. Ankietowani wybierali czy widzieli/nie widzieli zmiany w obu środowiskach albo czy tylko w jednym z nich. Na zamieszczonych poniżej rysunkach (Rysunek 4 i 5) zostały przedstawione wyniki tych pytań.



Rysunek 4: Wykres przedstawiający odpowiedź ankietowanych na zauważalność zmian poszczególnych poziomów szczegółowości dla modeli 1-5



Rysunek 5: Wykres przedstawiający odpowiedź ankietowanych na zauważalność zmian poszczególnych poziomów szczegółowości dla modeli 6-10

6. Wnioski

W obecnym świecie gier komputerowych, który cały czas się rozwija nowe produkcje dużych firm zawierają coraz więcej modeli o coraz lepszej jakości. W celu utrzymania takiej samej płynności gry jak te poprzednie, studia gier muszą pracować nad optymalizacją swoich produkcji. Dzięki temu ich gra będzie dostępna dla większej liczby użytkowników. Jednym ze sposobów w jaki mogą tego dokonać jest zastosowanie teselacji. Dynamiczne podmienianie modeli na te o mniejszej szczegółowości znacząco poprawia płynność gry. Ręczne tworzenie takich modeli może zająć bardzo dużo czasu i zasobów, więc korzystanie z narzędzi, które automatycznie wygenerują te modele może być dobrym pomysłem. Niekomercyjne narzędzia, które zostały wcześniej przetestowane mogą być dobrym punktem startowym dla mniejszych programistów. Badania pokazały, że mimo szybszego oraz praktycznie nieodczuwalnego na sprzęcie generowania tych poziomów szczegółowości korzystając z narzędzia AutoLOD w środowisku Unity Game Engine, ankietowani woleli wygląd modeli, które zostały wykonane przez wbudowane w środowisko Unreal Engine narzędzie. Biorąc pod uwagę odpowiedzi ankietowanych na drugie pytania w przeprowadzonej ankiecie, większość ludzi nie zauważa różnic między mniej i bardziej szczegółowymi modelami. Dzięki tej informacji można przyjąć, że uwaga potencjalnego gracza nie będzie odciągana przez zmieniające się modele obecne w dalszej odległości od niego.

Literatura

[1] P. Glancey, The Complete History of Computer and Video Games. EMAP IMAGES 1996.

[2] A. Okita, Learning C# Programming with Unity 3D, Wyd. CRC Press Taylor & Francis Group, Nowy Jork 2015.

[3] Wszystkie potrzebne informacje o blueprintach dostępnych w Unreal Engine, https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html [06.09.2020].

[4] Krótkie wyjaśnienie Level of Details oraz spsosób ich tworzenia w Unreal Engine, https://docs.unrealengine.com/en-US/Engine/Content/Types/StaticMeshes/HowTo/LODs/index.html[06.09.2020].

[5] T. Norton, Learning C# by Developing Games with Unity 3D Beginner's Guide, Wyd. Packt Publishing, Birmingham 2013.

[6] W. Goldstone, Unity Game Development Essentials, Wyd. Packt Publishing, Birmingham 2009.

[7] A. Cookson, R. DowlingSoka, C. Crumpler, Unreal Engine w 24 godziny. Nauka tworzenia gier. Helion 2016.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 417–420
Received: 29 November 2020
Accepted: 7 December 2020

Analysis of the use of the UTAUT model for modeling the information technology acceptance process

Analiza wykorzystania modelu UTAUT do modelowania procesu akceptacji technologii informacyjnych

Magdalena Czerwinska*

Department of Economics and Economic Management, Lublin University of Technology, Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

The article is devoted to the issues of UTAUT (the Unified Theory of Acceptance and Use of Technology) models, which are currently the most commonly used tools of IT acceptance assessment. The aim of the article was to characterize the structure and stages of evolution of the above-mentioned models, to analyze the practical use of these models and to perform a bibliometric analysis of publications on UTAUT. The method of literature analysis and the SciVal Scopus and Google Trends tools were used to analyze UTAUT content search trend statistics. The described models are useful for testing technology acceptance by users with different characteristics in different organizations. The flexibility of the models in terms of extending and modifying them for the needs of various areas of IT technology implementation was demonstrated.

Keywords: information technologies acceptance models; UTAUT

Streszczenie

Artykuł jest poświęcony problematyce modeli UTAUT (Uogólnionej Teorii Akceptacji i Użytkowania Technologii) które są obecnie najczęściej wykorzystywanymi modelami akceptacji technologii informacyjnych. Celem artykułu było scharakteryzowanie struktury oraz etapów ewolucji wyżej wymienionych modeli, przeprowadzenie analizy praktycznego wykorzystania tych modeli oraz dokonanie analizy bibliometrycznej publikacji dotyczących UTAUT. Zastosowano metodę analizy literatury oraz narzędzia SciVal Scopus i Google Trends do analizy statystyki trendów wyszukiwania treści dotyczących UTAUT. Udowodniono, że opisane modele są przydatne do badania akceptacji technologii przez użytkowników o różnych cechach w różnych organizacjach. Wykazano elastyczność modeli w zakresie ich rozszerzania i modyfikowania na potrzeby różnych obszarów wdrażania technologii IT.

Słowa kluczowe: modele akceptacji technologii informacyjnych; UTAUT

*Corresponding author

Email address: m.czerwinska@pollub.pl (M. Czerwinska)

1. Introduction

Research on the various aspects of IT use is necessary as the IT technologies are spreading worldwide. The widely understood determinants of processes related to the adoption, implementation, management and effectiveness of technologies are the subject of scientific research. Whether a specific technology will be adopted in practice depends on many factors of various nature - technical and technological solutions are important here, but also infrastructural, economic and legal conditions. It turns out that the process of technology acceptance by potential users is equally important, as it has a direct impact on the actual use of the implemented solutions.

One of the research trends is the analysis of the relationship between the adoption of technology and the variables that influence it. Technology acceptance models UTAUT are an example of this.

Information technology acceptance models are used to explain and forecast the behaviour of Internet users regarding various manifestations of their online activity - searching for information, using websites, online shopping, activity in social media and using health information available on the Internet. They help to understand the factors influencing the acceptance of information technologies.

The UTAUT model [1] is not the first information technology acceptance model. It appeared in 2003 as a uniform version of the repeatedly modified TAM models (Technology Acceptance Model - TAM [2], Extension of the Technology Acceptance Models - TAM2 [3] and TAM3 [4]). It was another step in the improvement of technology acceptance research tools. It was developed to provide a more complete picture of the acceptance process. It has integrated key elements from eight models (Theory of Reasoned Action - TRA [5], Theory of Planned Behaviour - TPB [6], Technology Acceptance Model - TAM [7], Model of PC Utilization - MPCU [8], Motivational Model - MM [9], Social Cognitive Theory - SCT [10], Extension of the Technology Acceptance Model - TAM2 [11] and Diffusion of Innovation Model - DOI [12]). It assumes that the direct determinants of the intention and behaviour of use are four key determinants (expected performance, expected effort, social impact and facilitating conditions).

The effect of these four constructs depends on age, gender, experience, and voluntary use [13] as shown in Figure 1.

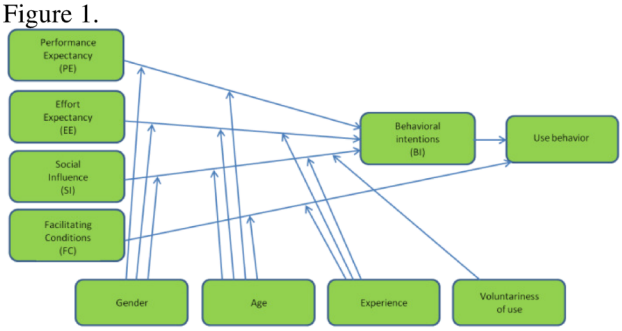


Figure 1: UTAUT model

Its authors quickly saw the need for further modifications, as a result of which the UTAUT2 model was created in 2012 [14]. It added three new determinants of technology acceptance (hedonic motivation, price value and habit) and redefined the previous constructs [15], which is presented in Figure 2.

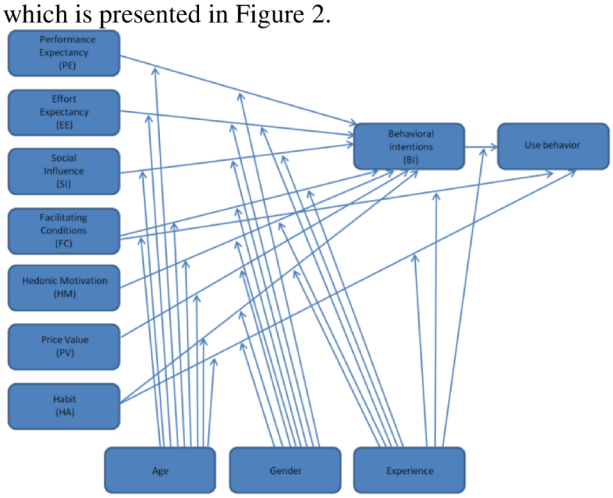


Figure 2: UTAUT2 model

Work on UTAUT modifications is still ongoing. In 2015, V. Venkatesh, J.Y.L. Thong and X. Xu [16] published an article in which they reviewed and synthesized the literature on UTAUT from September 2003 to December 2014. They conducted a theoretical analysis of UTAUT and its extensions and developed a plan for further research.

2. Research methodology

As part of the own research, an analysis of the practical use of these models and a bibliometric analysis of publications on UTAUT were carried out. The literature analysis method and the SciVal Scopus and Google Trends tools were used to analyze UTAUT content search trends statistics.

3. Results of research

The UTAUT model, due to its simplicity and versatility, is used to test the acceptance of various technologies related to information and communication systems, general-purpose systems, office systems and specialized business systems. The material scope covered by UTAUT's applications is very wide and varied. Examples include the application of the UTAUT model to research: factors influencing IT services in healthcare [17], acceptance of Healthcare wearable devices (HWDs) [18], determinants of using mobile learning (m-learning) [19], determinants of high school students' educational use of YouTube [20], online airline ticket purchase [21], the use of interactive whiteboards in education [22], social robotics [23], rehabilitation technologies [24], the use of electronic medical records [25], understanding the patients and clinicians behavioural intentions to use telemedicine equipment [26], understanding and learning about the factors that influence the intention of the end-user to use a new technology in the form of cloud-based mHealth services [27], researching e-marketing services in developing countries [28], mobile banking systems [29], acceptance and use of big data techniques in services companies [30].

Data for the analysis the research of topic UTAUT was collected from Google Trends (https://trends.google.com/trends) and was standardized. The greatest interest in the searched keywords is expressed by 100, while the lack of interest or insufficient data is expressed by 0. Google Trends presents data from different geographic locations in the selected time range settings. Queries are collected from five specialized search engines: Internet, Graphics, News, Google Shopping, and YouTube Search. As the first articles on UTAUT appeared in 2003 (in September), the data for the study was collected from the beginning of Google Trends, i.e. from 2004.

The first approach was to collect data on the search for the following terms: TAM, Technology Acceptance Model, UTAUT and UTAUT2. Figure 3 shows the worldwide interest (in search results) from January 1, 2004 to November 2020. Data from Google Trends was downloaded on November 20, 2020 (November data is incomplete). Interest in the UTAUT term began in November 2004.

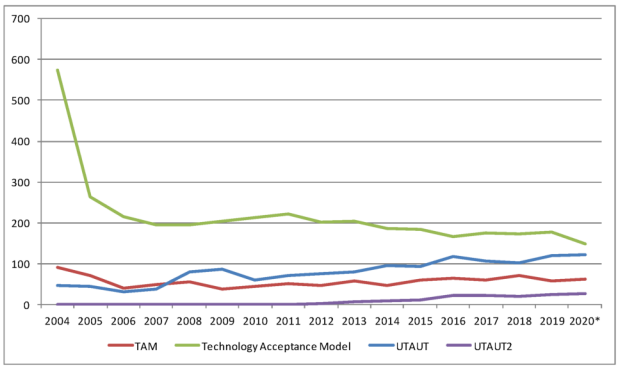


Figure 3: Worldwide interest over time for TAM, Technology Acceptance Model, UTAUT i UTAUT2; source: Google Trends

While Figure 3 shows changes in interest over time, Figure 4 illustrates where the term UTAUT was most popular during the period considered. The saturation of yellow in the drawing indicates the level of interest, with the more intense the colour the more interest. The greatest interest in UTAUT was expressed by Internet

users in Malaysia, Ghana, Indonesia, South Africa and the Netherlands. This is reflected in scientific articles on UTAUT. Table 1 shows the most prolific countries and regions publishing research in the research area.

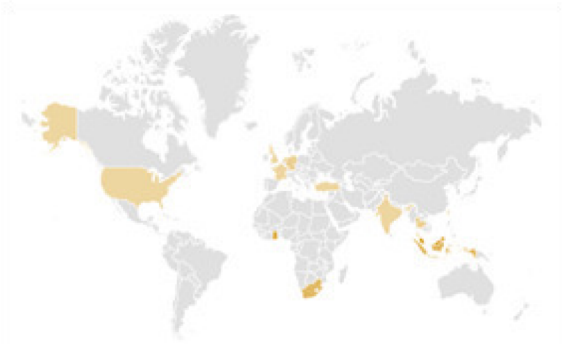


Figure 4: Compared breakdown by countries in period from 1 January, 2004 to 20 November, 2020 for the search term: UTAUT; source: Google Trends (most active countries mark with different levels of yellow colour)

Table 1: The most prolific countries and regions publishing research in the research area – UTAUT

Top countries/regions
Entity: UTAUT - keywords · Year range: 2010 to 2019 · Data source: Scopus, up to 18 Nov 2020 ·

Countries & territories	Scholarly Output	Views Count
Malaysia	187	10,638
United States	151	12,161
Indonesia	95	3,455
China	93	5,022
United Kingdom	92	6,663
Taiwan	74	4,771
India	58	2,599
Germany	53	1,958
Australia	48	2,591
South Korea	47	2,583

Over the last decade, the number of scientific publications in the UTAUT area indexed in the Scopus database has also increased (52 publications were indexed in 2010, 320 in 2019) (Figure 5).

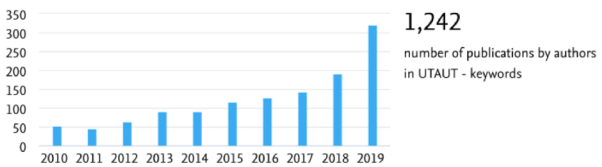


Figure 5: Number of UTAUT publications indexed in Scopus in period from 2010 to 2019; source: SciVal

The Keyphrases SciVal tool was also used in the work. It allows a more detailed analysis of the 50 most important key phrases related to a topic or research area.

The bibliometric analysis using Scopus, the largest database of peer-reviewed literature, and SciVal shows that the scientific output related to UTAUT is strong and growing in recent years. The key topics in this field are presented in Figure 6, where united theory, acceptance, technology acceptance and UTAUT are the fastest growing in recent years.

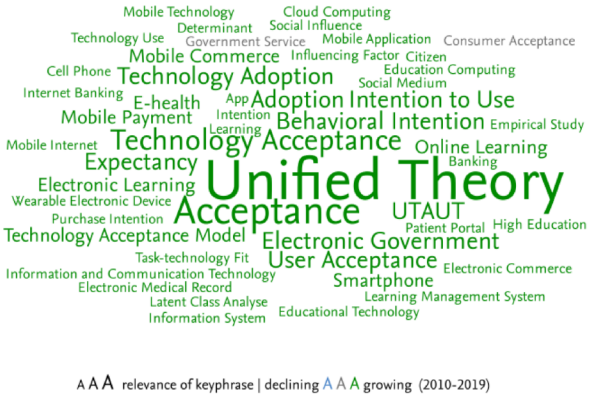


Figure 6: UTAUT key topics; source: SciVal

4. Conclusions

An analysis of the literature on information technology acceptance models, especially the UTAUT model, provided evidence about the usefulness of these models for testing technology acceptance by users with different characteristics in different organizations. These models are widely used. Their flexibility in terms of extending and modifying them for the needs of various areas of IT technology implementation has been demonstrated. Many of the analyzed articles contained extensions to the original UTAUT version, which indicates that no single optimal version of the model has been identified so far. Diversification of the research area forces modifications of the model to a specific research situation. There are still open possibilities to expand and improve the model in order to make it more useful.

Data from Google Trend shows that the interest in the UTAUT model still has strong regional differentiation.

Similarly, the analysis carried out with the use of Scopus and SciVal leads to the conclusion that the scientific achievements in this field also come mainly from selected locations.

The development of the Internet and the emergence of new types of online services give rise to the need to explain ever new areas of human activity in this area. The use of ICT concerns various human behaviours, the Internet is not a homogeneous environment, providing only information, the spectrum of its applications is becoming wider. All this implies the necessity of creating more and more advanced theoretical models to explain network behaviour. The more so as it should be remembered that online activity and the use of ICT take place in a diverse environment and it is very important to take into account the aspects of the environment in which the entity using ICT is located. The virtual and real world interpenetrate and condition each other.

It seems, therefore, that work on information technologies acceptance models should be directed towards creating their holistic versions, covering the broadest possible set of behaviour determinants of entities using ICT.

References

[1] V. Venkatesh, M.G. Morris, G.B. Davis, F.D. Davis, User acceptance of information technology: Toward a unified view, MIS quarterly (2003) 425-478.

[2] F.D. Davis, A technology acceptance model for empirically testing new end-user information systems: Theory and results, PhD Thesis, Massachusetts Institute of Technology, 1985.

[3] V. Venkatesh, F.D. Davis, A theoretical extension of the technology acceptance model: Four longitudinal field studies, Management science 46(2) (2000) 186-204.

[4] V. Venkatesh, H. Bala, Technology acceptance model 3 and a research agenda on interventions, Decision sciences 39(2) (2008) 273-315.

[5] M. Fishbein, I. Ajzen, Understanding attitudes and predicting social behaviour, 1980.

[6] I. Ajzen, From intentions to actions: A theory of planned behavior. In: Action control, Springer, Berlin, Heidelberg 1985, p. 11-39.

[7] F.D. Davis, A technology acceptance model for empirically testing new end-user information systems: Theory and results, PhD Thesis, Massachusetts Institute of Technology, 1985.

[8] R.L. Thompson, C.A. Higgins, J.M. Howell, Personal computing: toward a conceptual model of utilization, MIS quarterly (1991) 125-143.

[9] F.D. Davis, R.P. Bagozzi, P.R. Warshaw, Extrinsic and intrinsic motivation to use computers in the workplace, Journal of applied social psychology 22(14) (1992) 1111-1132.

[10] A. Bandura, Social cognitive theory of personality, Handbook of personality 2 (1999) 154-196.

[11] V. Venkatesh, F.D. Davis, A theoretical extension of the technology acceptance model: Four longitudinal field studies, Management science 46(2) (2000) 186-204.

[12] E.M. Rogers, Diffusion of innovations, Free Press, New York 2003, p. 551.

[13] M. Alshehri, S. Drew, T. Alhussain, R. Alghamdi, op. cit.

[14] V. Venkatesh, J.Y.L. Thong, X. Xu, Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology, MIS quarterly 36(1) (2012) 157-178.

[15] T. Escobar-Rodríguez, E. Carvajal-Trujillo, Online purchasing tickets for low cost carriers: An application of the unified theory of acceptance and use of technology (UTAUT) model, Tourism Management 43 (2014) 70-88.

[16] V. Venkatesh, J.Y.L. Thong, X. Xu, op.cit.

[17] N. Phichitchaisopa, T. Naenna, Factors affecting the adoption of healthcare information technology. EXCLI journal 12 (2013) 413.

[18] H. Wang, D. Tao, N. Yu, X. Qu, Understanding consumer acceptance of healthcare wearable devices: An integrated model of UTAUT and TTF. International Journal of Medical Informatics (2020) 104156.

[19] C. M. Chao, Factors determining the behavioral intention to use mobile learning: An application and extension of the UTAUT model, Frontiers in psychology 10 (2019) 1652.

[20] S. Bardakcı, Exploring High School Students' Educational Use of YouTube. International Review of Research in Open and Distributed Learning 20(2) (2019).

[21] T. Escobar-Rodríguez, E. Carvajal-Trujillo, op.cit.

[22] K.T. Wong, T. Teo, S. Russo, Interactive whiteboard acceptance: Applicability of the UTAUT model to student teachers, The Asia-Pacific Education Researcher 22(1) (2013) 1-10.

[23] G. Wolbring, L. Diep, S. Yumakulov, N. Ball, D. Yergens, Social robots, brain machine interfaces and neuro/cognitive enhancers: Three emerging science and technology products through the lens of technology acceptance theories, models and frameworks, Technologies 1(1) (2013) 3-25.

[24] L. Liu, A. Miguel Cruz, A. Rios Rincon, V. Buttar, Q. Ranson, D. Goertzen, What factors determine therapists' acceptance of new technologies for rehabilitation–a study using the Unified Theory of Acceptance and Use of Technology (UTAUT), Disability and rehabilitation 37(5) (2015) 447-455.

[25] J. Tavares, T. Oliveira, Electronic health record patient portal adoption by health care consumers: an acceptance model and survey, Journal of medical Internet research 18(3) (2016) e49.

[26] A. Kohnke, M.L. Cole, R. Bush, Incorporating UTAUT predictors for understanding home care patients' and clinician's acceptance of healthcare telemedicine equipment, Journal of technology management & innovation 9(2) (2014) 29-41.

[27] F. Khatun, M.J.U. Palas, P.K. Ray, Using the Unified Theory of Acceptance and Use of Technology model to analyze cloud-based mHealth service for primary care, Digital Medicine 3(2) (2017) 69.

[28] P.A. Nuq, B. Aubert, Towards a better understanding of the intention to use eHealth services by medical professionals: the case of developing countries, International Journal of Healthcare Management 6(4) (2013) 217-236.

[29] E. Purwanto, J. Loisa, The intention and use behaviour of the mobile banking system in Indonesia: UTAUT Model. Technology Reports of Kansai University 62(06) (2020) 2757-2767.

[30] J.P. Cabrera-Sánchez, Á.F. Villarejo-Ramos, Acceptance and use of big data techniques in services companies. Journal of Retailing and Consumer Services 52 (2020) 101888.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 421–427
Received: 29 November 2020
Accepted: 23 December 2020

Transport preferences of the students and employers in Lublin University of Technology

Ocena preferencji transportowych pracowników i studentów Politechniki Lubelskiej

Jakub Bis[a,*], Magda Kojro[b]

[a]*Department of Economics and Economic Management, Lublin University of Technology, Nadbystrzycka 38B, 20-618 Lublin, Poland*
[b]*Zakład Zagospodarowania Odpadów, Wólka Rokicka 100, 21-100 Lubartów, Poland*

Abstract
The problem of movement is very important in the context of logistic management, which has been gaining importance in recent years with the increasing phenomenon of people moving to large urban agglomerations. The article presents the transport preferences of employees and students the Lublin University of Technology. IT tools were used for the analysis, such as the Analysis ToolPak add-on and PowerMap in Microsoft Excel.

Keywords: transport preferences; logistic management; PowerMap

Streszczenie
Problem przemieszczania się jest bardzo istotny w kontekście zarządzania logistycznego, które zyskuje na znaczeniu w ostatnich latach wraz z nasilającym się zjawiskiem przenoszenia się ludności do dużych aglomeracji miejskich. W artykule przedstawiono skąd dojeżdżają na Politechnikę Lubelską pracownicy i studenci oraz ich preferencje transportowe. Do analizy wykorzystano narzędzia informatyczne takie jak dodatek Analysis ToolPak oraz PowerMap w programie Microsoft Excel.

Słowa kluczowe: preferencje transportowe; zarządzanie logistyczne; PowerMap

*Corresponding author
Email address: j.bis@pollub.pl (J. Bis)

1. Introduction

Management is a discipline of science, the task of which is to take actions and decisions regarding human, material, capital and information resources in such a way that their functioning is as effective as possible and leads to the achievement of the assumed goals. On the other hand, the dynamic development of cities and the mobility of society strengthens the importance of logistics, which is responsible, inter alia, for the effectiveness of the movement of people. The use of both of these sciences is necessary from the point of view of the topic discussed in this article, which is the transport preferences of students and employees of the Lublin University of Technology. The aim of this study is to identify the preferences related to travel to the university of students and employees of the Lublin University of Technology, and then, based on data analysis using IT tools, to develop proposals for changes in the area of logistics management at the university under study. The first part of the article reviews the literature research on student behavior and transport preferences. The second part discusses the results of own research conducted among students and employees of the Lublin University of Technology. The third part of the work consists of recommendations and suggestions from the conducted research.

2. Transport preferences and behavior in the light of research

Logistics management is a process of planning, implementing and controlling efficient, cost-effective flow and storage of raw materials, production inventories, final goods and the corresponding information, from the point of obtaining raw materials to the point of consumption, in order to best adapt to the requirements customers. Such management is an activity that creates the entire concept of logistics undertakings, taking into account their course both in the company and at partners, and coordinating the implementation of this concept by appropriate organizational units using appropriate management and control instruments [1].

One of the dominant functions of the city is to enable its inhabitants to move regardless of the premises and methods of traveling. In response to the constantly emerging transport needs or the emergence of other negative effects of the movement of people in cities, the concept of urban logistics was created, which is to be a solution to the emerging development and functional problems of cities (and other administrative units that are not formally cities). The concept of urban logistics has been given many definitions, among which it is worth quoting one, most relevant to the topic of the work: "urban logistics is all the processes of managing the flows of people, cargo and information within the

city's logistics system, in accordance with the needs and development goals of the city, respecting the protection of the natural environment, taking into account that the city is a social organization whose ultimate goal is to meet the needs of its users" [2].

As the research conducted by the Public Transport Authority in Gdynia has shown, the share of the use of passenger cars in passenger transport has been steadily increasing over the last 5 years, to the detriment of public transport. In 2008, 51.9% of respondents used public transport (bus, trolleybus and rail), and seven years later this number dropped to 39.8%. At the same time, the share of road transport increased by 10.8 percentage points. It is worth noting, however, that the inhabitants of Gdynia have become more willing to use the means of transport, which is a bicycle (in the last seven years, the share of bicycle trips increased by 1.4 percentage points) [3].

The results of the research led by Sokołowicz in [4] show that the most frequently chosen means of transport was public transport (over 750 indications), the less frequently chosen passenger car (over 600 indications), while the least willingly chosen means of transport was public transport (approx. 300 indications). It is worth noting that people traveling to the university at least 3 times a week are more likely to travel by public transport (53%), while the less often students come to the university, the more often they choose to travel by foot, car, bike or use buses/trains.

Interesting results were brought by a study focusing on the traffic situation during rush hours in the most crowded Polish cities. For this purpose, travel time of the busiest streets in three seasons: off-peak (23 May), 8 August during the holiday peak and off-peak on 23 and 24 May. The results of the experiment showed that the 5-kilometer section in the city of Lublin was completed in 10 minutes at the time of the lowest traffic congestion, i.e. driving at a speed of 30 km / h, while during the greatest transport congestion the travel time was 40 minutes, i.e. only 7.5 km / h, which is slightly faster than human fast walking (it is assumed that this speed is approx. 6 km / h). In addition, the article contains information that the average speed of driving during the street rush in the city of Lublin was only about 8.1 km / h, only one city - Rzeszów - had a worse result [5].

More information on the transport preferences of commuters and residents is provided by a survey of the transport market in Lublin carried out in 2016, which covered 100 randomly selected people, and the tool that accompanied it was a questionnaire containing mainly closed questions. The study checked the frequency of using a given means of transport. The most popular turned out to be own vehicle, where almost half of the respondents indicated that it is used every day and which received the fewest answers "I do not use at all". About 20% of people used public and private transport (buses, buses) almost equally every day. The least frequently chosen means of transport were taxi rides, rail transport and private transport [6].

The research conducted in Gdańsk, which included 764 students from 12 schools in the 2013/2014 school year, allowed for the following conclusions:

1. A passenger car was owned by 83% of lower secondary school students' households.
2. According to middle school students, the most important features of public transport are: directness, frequency and speed. Secondary school students considered other important transport suggestions in the following order: information, rhythm and comfort.
3. During city trips, lower secondary school students mostly used public transport, which they made 79% of trips during the research.
4. The most popular means of public transport in the travel structure of middle school students was the bus, while travel by individual transport was led by a passenger car [7].

A review of international research on transport preferences shows that students at universities around the world prefer to travel by car [8]. Many university campuses are suffering from serious mobility problems resulting from excessive use of the private car by students, teachers and administrative staff [9]. The interpretation of data exposed a significant statistical hypothesis that the students would use public transport more often if made available in certain hours [10]. The survey findings support to the notion that bicyclists everywhere have similar attitudes about what the types of improvements required for increasing bicycling and enhancing their experiences. In addition, local conditions and practices have an impact on the relevance of specific issues [11]. Very important are the topographical factors. In Kobe University in Japan the authors determined if walking was included in the choice set. One striking reason for occasional walking rather than taking a bus was to meet friends who walked to campuses [12]. Research in Australia has found demographic and travel habit differences between native-born Australians and immigrants but the reasons for these differences are not clear [13]. Neighbourhood type of residence was an important indicator of a student's transportation life-style. Strong associations between travel attitudes, residential location preferences and a student's transportation life-style was also observed. Post-secondary students are at an important stage in their life-course where they begin to form habitual travel behaviour as young adults [14].

3. Research Methodology

In order to find out what are the transport preferences of people associated with the university, the authors conducted a study of full-time and part-time students as well as employees of the Lublin University of Technology. In data processing, an add-in to Microsoft Excel called PowerMap was used, when it was possible to visualize the spatial distribution of the studied group. PowerMap lets authors choose an entity in CRM, choose a view, and plots all of the individual records from that view on one map. Scientist using PowerMap

can set more than one entity, and more than one view at a time. They can also save their PowerMap preferences, including map configurations, zoom levels and views. The best features of this add-in are: support for any entity with an address field, click on a pin on a map, and a summary of the record will display, open a record directly from the map, by clicking on the name of the record in the summary view, choose a separate pin for each selected entity or view or put PowerMap directly on a CRM form. The Analysis ToolPak extension [15-17] allowed authors to calculate the median, and this tool uses appropriate statistical or engineering macro functions to calculate the results and display them in the result table. The survey questionnaire was the tool used during the own research using the Google Docs form in the second decade of 2019. The questions asked to the respondents were aimed at determining the popularity of various types of transport used by students and employees to reach the university, determining the reasons influencing the choice of the means of transport as well as factors encouraging and discouraging the choice of a given means of transport by the respondents. The aim was also to identify the problems that may be encountered during the trip to the university and get acquainted with ideas for improving commuting.

The 357 people participated in the survey, including 142 university employees (39.8%), 201 full-time students (56.3%; including 143 undergraduate students and 58 second degree students) and 14 part-time students (3, 9%). The gender structure of the respondents is almost proportionally distributed. Men constituted 51.3% of the respondents and women 48.7%. Women employed at the Lublin University of Technology constituted 45.8% of all surveyed employees, and female students made up a group of 50.7% of the surveyed students.

The questionnaire was completed by people, 73.1% of whom visit the university at least 4 times a week, 23.8% of respondents visit it 2 to 3 times a week, and the rest of the group less frequently.

4. Analysis and Result

People who took part in the survey were asked about where they come from. One full-time student and two extramural students declared that they traveled to the university from the Mazowieckie Voivodship. One part-time student comes from the Podkarpackie Voivodeship and three people, including one employee and two full-time students, admitted that they come from the Świętokrzyskie Voivodeship. The vast majority replied that they reached the Lublin University of Technology from the Lubelskie Voivodeship (98%). The vast majority of respondents live in the city of Lublin (62.6%) and its neighbouring areas (18.9%). A significant number of people come from the county of Świdnica (6.3%) and Lubartów (4.0%). For the remaining areas, the ratios were equal to or less than 1.4%.

The structure of people commuting to the university from the Lubelskie Voivodeship shows that the employees of the Lublin University of Technology are people living in the city of Lublin and its surroundings, in contrast to students who come to the university from closer and further distant communes of the Lublin Voivodeship.

Map 1: Number of university students and employees who travel to the university from the Lublin voivodeship
Source: Own study using PowerMap in Microsoft Excel based on own research.

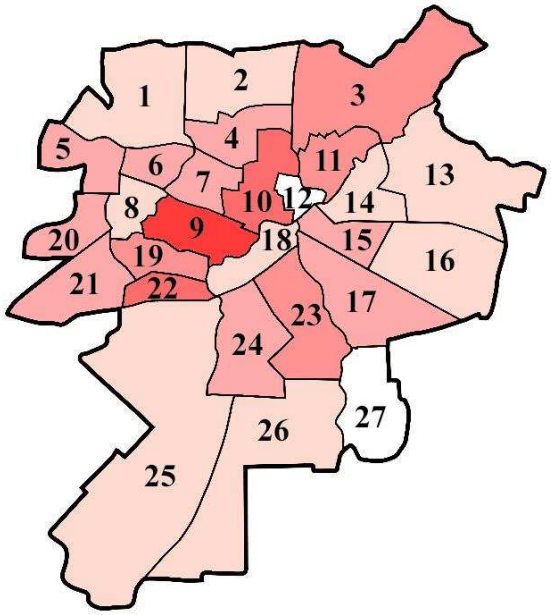
The respondents are mostly people coming to the university from the Rury district (19.6% of respondents living in the city of Lublin), which also includes the Lublin University of Technology. A large percentage of the respondents live in the neighboring districts, incl. Czuby Południowe (9.1%), Śródmieście (7.3%), Czuby Północne (6.8%), Dziesiąta (5.5%) and distant districts such as Ponikwoda (5.9%) and Kalinowszczyzna. The smallest number of respondents came from the districts furthest from universities, located near the city limits, and from the Za Cukrownia district, which is located directly behind the university. This may be due to the fact that the area is rather industrial in nature. None of the surveyed people do not live in the Old Town and Głusk districts. In the city plan presented below, the districts are marked with colors, taking into account the number of respondents living in the area (Map 2).

The questionnaire was started with the question about the distance that students and university staff have to cover in order to reach it. The smallest indicated value for the entire research sample was 100 meters, and the highest was 120 kilometers. Both numbers were indicated by full-time students. For university employees, these values were respectively 200 meters and 86 kilometers. 25% of the total value for the studied population is or is lower than 3.4 km, and the median is 6.5 km.

This means that half of the respondents reach the university from places located 6.5 km or less away from it, for the remaining people the distance is greater. In the case of university employees, half of the respondents live in areas 6 km or less from the Lublin University of Technology, and half of all surveyed students reach the university from places located up to 8 km away. Further data analysis shows a discrepancy between the presented results. It turns out that 75% of the surveyed students have to travel 25 km or less to get to the university, while the same number of employees come from areas 10.8 km or closer.

Table 1: Number of respondents in the city of Lublin.
Source: Own study based on the results of the conducted research.

Number on the map	City district	Number of respondents living in a district
1.	Sławin	5
2.	Czechów Północny	5
3.	Ponikwoda	13
4.	Czechów Południowy	6
5.	Szerokie	6
6.	Sławinek	8
7.	Wieniawa	6
8.	Konstantynów	4
9.	Rury	43
10.	Śródmieście	16
11.	Kalinowszczyzna	12
12.	Stare Miasto	0
13.	Hajdów-Zadębie	1
14.	Tatary	3
15.	Bronowice	8
16.	Felin	3
17.	Kośminek	6
18.	Za Cukrownią	1
19.	Czuby Północne	15
20.	Węglin Północny	6
21.	Węglin Południowy	7
22.	Czuby Południowe	20
23.	Dziesiąta	12
24.	Wrotków	8
25.	Zemborzyce	3
26.	Abramowice	2
27.	Głusk	0



Map 2: Distribution of the surveyed people in the city of Lublin.
Source: Own study based on the results of the conducted research.

The maximum distance of residence for learners is 120 km, and for university employees it is almost twice as small, at 86 km. In addition, part-time students have to cover a longer distance to reach the university campus than full-time students. The above information coincides with the respondents' declarations related to the place of residence, which proved that the employees of the education centre mainly live in the city of Lublin and its neighbouring areas, and that students often come from other municipalities of the Lublin voivodship, remote from Lublin. The average commuting distance of learners is 17.5 km and is 3 percentage points higher than the average for the studied sample, which is 14.5 km. The average travel distance for university employees is 9.9 km.

The next question in the questionnaire concerned the travel time from the place of residence to the Lublin University of Technology. The respondents were to enter the number of minutes corresponding to the average one-way travel times, including the shortest and the longest, e.g. during rush hours. The smallest indicated value for the shortest journey time was equal to 1 minute, and the highest - 90 minutes. For the longest time of reaching the university, the indications were equal to 3 minutes and 180 minutes, respectively. Correlation coefficients were calculated for the data, which show that in the case of the shortest route times, there is a very large relationship between the distance to the place of residence and the travel time, where the correlation coefficient is 0.8. The slightly lower correlation coefficient between the route length and the longest travel time, amounting to 0.68, proves that there is also a fairly high relationship between them. 75% of the respondents believe that the shortest travel time takes them equal or less than 30 minutes, the remaining part of people reach the university in more than half an hour but less than 90 minutes. One-fourth of the respondents travel between their place of residence and the university in an average of 10 minutes or less under the most favourable conditions. Under unfavourable conditions, 25% of respondents declare that the travel time from their place of residence to the Lublin University of Technology is a maximum of 23 minutes or less. 75% of the respondents believe that their longest journey to the university is on average an hour or less, and the rest that this time is longer than an hour but less than 3 hours. The minimum average travel time to the university for all respondents is 22.2 minutes, while the maximum travel time in the street rush is 42.8 minutes. Generally speaking, it can be said that commuting to the Lublin University of Technology takes more time for people studying there than for the employed. The reason for such a situation may again be that the places of residence are far from the place of study of students. Detailed data on travel times to the university by individual groups of respondents are presented in the table 2.

Table 2: Summary of the respondents' travel times to the university. Source: Own study based on the results of the conducted research.

		Total	Employees	Students
Average fastest one-way travel time to the university (in minutes)	Minimum value	1.0	2.0	1.0
	The first quartile	10.0	10.0	10.0
	Median	20.0	15.0	20.0
	Third quartile	30.0	22.3	30.0
	Maximum value	90.0	90.0	90.0
	Average	22.2	18.3	24.8
Average longest one-way travel time to university (in minutes)	Minimum value	3.0	3.0	3.0
	The first quartile	23.0	20.0	25.0
	Median	40.0	30.0	40.0
	Third quartile	60.0	45.0	60.0
	Maximum value	180.0	180.0	120.0
	Average	42.8	36.8	46.7

The frequency of choice of means of transport by the surveyed clearly indicates the two most popular ways of traveling, which are passenger car and public transport. According to research, 70% and 50.4% of respondents use these vehicles, respectively, and in both cases, for over 32% of respondents, they constitute a permanent means of movement. Other ways of traveling are not chosen so many times. Very often, only 9.0% of the respondents travel to the university on foot, 26.6% do it less frequently, and 64.4% of the respondents never do it. Intercity rides by public transport are chosen by slightly over 21% of respondents, and 13.2% use the services of railway operators. 18.8% of respondents prefer commuting by bicycle, and only 1.4% declare that they are used very often. One person believes that from time to time they reach the university using the services provided by taxi drivers and 7 people declare that they use a motorcycle to get to the university.

For students and employees, the most important transport preferences are related to the travel time, no need to change trains and the frequency and punctuality of commuting. So the conclusion is that the trip to the university should be quick and direct to the destination. Although for 28.9% of respondents the number of parking spaces does not matter or is not to a large extent important, over 70% see the need for availability of parking spaces for comfortable travel. For 79.8% of students and employees of the Lublin University of Technology, the motivator when choosing the means of transport is the possibility of choosing the route independently and the independence of traveling, which is also reflected in the large number of votes for the importance of having your own means of transport (73.7% of respondents believe is a significant or very important factor). Respondents believe that safe travel is an important factor that they pay attention to. Only for a group of 3.4% of people it is an irrelevant postulate at all, and 3.1% have no opinion or the issue does not concern them. Less important for travellers are issues related to the comfort of driving or other types of movement, convenience, the ability to easily get to the means of transport and transport luggage with it. Costs incurred in connection with travel to the university are of significant importance for 73.9% of the respondents, but 18.8% is of secondary importance and for the rest of the group they are irrelevant. The factors assessed by the surveyed group as negligible are undoubtedly related to physical health and environmental protection. For 43.4% of students and employees participating in the survey, the type of travel should have a positive effect on health, for 34.7% it is of little importance and the group of 14.6% of respondents does not follow health postulates when choosing the method of travel to the university. When it comes to environmental performance, only 34.7% of respondents take into account the dangers of using less environmentally friendly means of transport, while 20.7% are indifferent to it, and 9% believe that this problem does not concern them or refrain from voice. The information received shows that for 10.9% of the respondents it is important how and what they travel and how others perceive it. The most numerous group of respondents, 53.2%, answered that the prestige and respect of the environment related to the way they move around did not matter to them.

5. Recommendations and suggestions

A significant proportion of respondents noticed that the problem is the poor condition of infrastructure, including no bicycle paths (eg. Tomasza Zana street, Aleje Kraśnickie, Czechów district have been mentioned) or no collision-free intersections for cyclists. The problem of obstructed bicycle traffic along Bystrzyca was also taken into account, which could be solved by creating two lanes for people traveling in opposite directions in this place. Bad infrastructure conditions for cyclists force cyclists to cycle on the pavement or road, which is not only uncomfortable and dangerous for themselves, but also poses a threat to pedestrians and makes it difficult for car drivers to ride. Employees and students also perceive the shortages or poor quality and bad placement of bicycle stands. This is related to the unwillingness to leave bikes in places where they are at risk of being stolen or damaged. The problem could be solved by new, better guarded parking spaces for bicycles, e.g. located near cameras and properly protected against

weather conditions. Another factor that discourages cycling is the lack of space for students and employees to refresh and change clothes. In the case of university employees, such a solution would allow them to freely change their clothes from sports clothes to more formal ones, because cycling in the clothes they wear to work is not only uncomfortable but also not very hygienic.

The most noticeable problem is the permanent traffic jams at Nadbystrzycka Street and partly at the exit from Wapienna Street, which leads to the university car park behind the Faculty of Management. This is due not only to the reprehensible condition of the roads, but also the lack of an additional driving lane for cars turning both from Nadbystrzycka Str. towards Wapienna Str. and from Wapienna Str. in Nadbystrzycka Str. Along with the planned renovation of this section of the road in the future, the creation of additional lanes should be considered, which would contribute to smoothing the traffic at the intersection.

According to the respondents, commuting by car is an alternative to commuting by public transport, because it is inefficient and often faces problems related to e.g. with long waiting times for the bus, delays and, above all, the lack of convenient connections. First of all, the respondents are of the opinion that commuting by bus often costs them as much as the cost of refuelling a car, therefore they choose to travel by car. Ticket prices should be competitively low to increase interest in collective travel. For this to happen, it would be possible to create special discounts on journeys for people working and studying at the Lublin University of Technology to make it more profitable for them to use the services offered by public transport in Lublin. Another solution would be the introduction by the university authorities of special financial allowances for employees using such journeys and the creation of an additional form of scholarship for students. Payments would be made upon presentation of relevant documents entitling to receive a given benefit, e.g. a monthly ticket for a given person.

Among the respondents studying at the Lublin University of Technology, the small number of parking spaces turned out to be the biggest problem. First of all, they notice that at least half of the people commuting to the university come from Lublin or its vicinity and choose the most convenient solution instead of using other means of transport. This, in turn, leads to a congestion in the parking lot behind the Faculty of Management, which is also wanted by people commuting from places far away from Lublin or those that have no other access than by car. In addition, parking spaces should be arranged in a more optimal way and then carefully marked so as to solve the problem of parking on several parking spaces or in a way blocking the exit of other drivers. Better organization and supervision of correct parking would certainly contribute to increasing the parking capacity and increasing the satisfaction of drivers. On the other hand, they would encourage people to travel to the university by car, which is not entirely consistent with the concept of sustainable development. The availability of a car park at the university is undoubtedly an advantage that encourages young people to choose one of the Faculties of the Lublin University of Technology. Therefore, it is not worth limiting students from using the car park, but it is worth promoting behaviours aimed at implementing the principles of caring for the environment.

One of the solutions in this case may be the activities carried out by the university authorities to increase self-awareness among students and employees of the Lublin University of Technology in the field of eco logic and principles of sustainable development, familiarizing them with the consequences of using various types of transport and promoting alternative solutions. As part of the program, an application could be created for drivers and other people commuting to the university, which, based on the entered data, would create groups of people arriving at the university at similar hours living in a given area. Such a program would operate on the basis of a sharing strategy (CarSharing) and its aim would be to enable shared journeys by students and university staff. This solution makes more people would reach the university, but with fewer cars. People who so far had to use uncomfortable public transport travel would travel in a much faster and more comfortable way, and since the cost of the trip would be spread among all passengers, it would be a win-win for them and for the driver.

6. Conclusions

Research on the transport preferences of students and employees of the Lublin University of Technology has drawn many conclusions regarding the perception of commuting to the university and the introduction of necessary changes to improve travel. Free access to the university, which is a place of work or study for the respondents, is certainly a factor that the respondents pay attention to and which encourages them to return to this place. Providing proper travel conditions is, on the one hand, a challenge for university and city authorities, and on the other hand, an obligation and an expression of caring for the needs of its clients. Students at Lublin University of Technology as at universities around the world prefer to travel by car [3] The conclusions from the study refer not only to transport preferences, but also realize that there is a need to conduct research on the opinions of people associated with the university, because it is they who often notice where the problems are and propose rational ways to solve them, due to the fact that they experience them themselves. In turn, the detection of existing problems and their repair contributes to the improvement of the functioning of the Lublin University of Technology, and this has a positive effect on its image and functioning.

Bibliography

[1] S. Krawczyk, Zarządzanie procesami logistycznymi, PWE, Warszawa 2001.

[2] J. Szołtysek, Logistyczne aspekty zarządzania przepływami osób i ładunków w miastach, Akademia Ekonomiczna w Katowicach, Katowice 2005.

[3] B. Orzechowski, O. Wyszomirski, Preferencje i zachowania komunikacyjne mieszkańców Gdyni. Raport z badań marketingowych ZKM w Gdyni, Zarząd Komunikacji Miejskiej w Gdyni, 2015.

[4] M. Sokołowicz, Preferencje transportowe studentów łódzkich uczelni, Katedra Gospodarki Regionalnej Środowiska Uniwersytetu Łódzkiego, Łódź 2011.

[5] K. Hebel, Zachowania transportowe mieszkańców w kształtowaniu transportu miejskiego. Fundacja Rozwoju Uniwersytetu Gdańskiego, Gdańsk 2013.

[6] A. Nieoczym., R. Longwic, W. Lotko, Badania rynku transportu zbiorowego w Lublinie, Autobusy, nr 6, 2016.

[7] W. Bartkowiak, Cała Polska stoi w korkach, Gazeta Wyborcza nr 208, wydanie z dn. 05.09.2008.

[8] B. Hasan-Basri, F.M. Berawi, N. Bakar, W. N. W. Mohamad, Logistics modelling for the university transport service using choice experiment, International Journal of Supply Chain Management Volume 9, Issue 3, 2020, Pages 105-114.

[9] S. Nash, R. Mitra, University students' transportation patterns, and the role of neighbourhood types and attitudes, Journal of Transport Geography Volume 76, April 2019, Pages 200-211.

[10] A. Bakdur, F. Masui, M. Ptaszynski, Predicting University Students' Public Transport Preferences for Sustainability Improvement, Advances in Intelligent Systems and Computing Volume 1251 AISC, 2021, Pages 362-376.

[11] N. Stamatiadis, A. Nikiforiadis, S. Basbas, P. Kopelias, E. Karantagli, A. Sitra, N. Mantas, Attitudes and Preferences of University Student Bicyclists: The Tale of Two Greek Cities, Advances in Intelligent Systems and Computing Volume 1278, 2021, Pages 945-953.

[12] N. Sanko, Activity-end access/egress modal choices between stations and campuses located on a hillside, Research in Transportation Economics Volume 83, November 2020, Article number 100931.

[13] R. Shafi, A. Delbosc, G. Rose, Understanding residential and travel preferences of South Asian international students, Australasian Transport Research Forum, ATRF 2019.

[14] L. dell'Olio, R. Cordera, A. Ibeas, R. Barreda, B. Alonso, J. L. Moura, A methodology based on parking policy to promote sustainable mobility in college campuses, Transport Policy Volume 80, August 2019, Pages 148-156.

[15] S. R. Janković, S.A. Mladenović, D. M. Mladenović, S. S. Zdravković, A. R. Uzelac, Big Data technology in traffic: A case study of automatic counters. Tehnika, 71(2), 2016, p. 281-288.

[16] J. Jankowski-Guzy, P. Cyplik, M. Adamczak, D. Głowacka-Fertsch, M. Chromińska, Modern forms of supporting business decisions in logistics. Business Logistics in Modern Management, 2018.

[17] M. Kajáti, E., Miškuf, P. Papcun, Advanced analysis of manufacturing data in Excel and its Add-ins. In 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI) (pp. 000491-000496) IEEE, 2017.