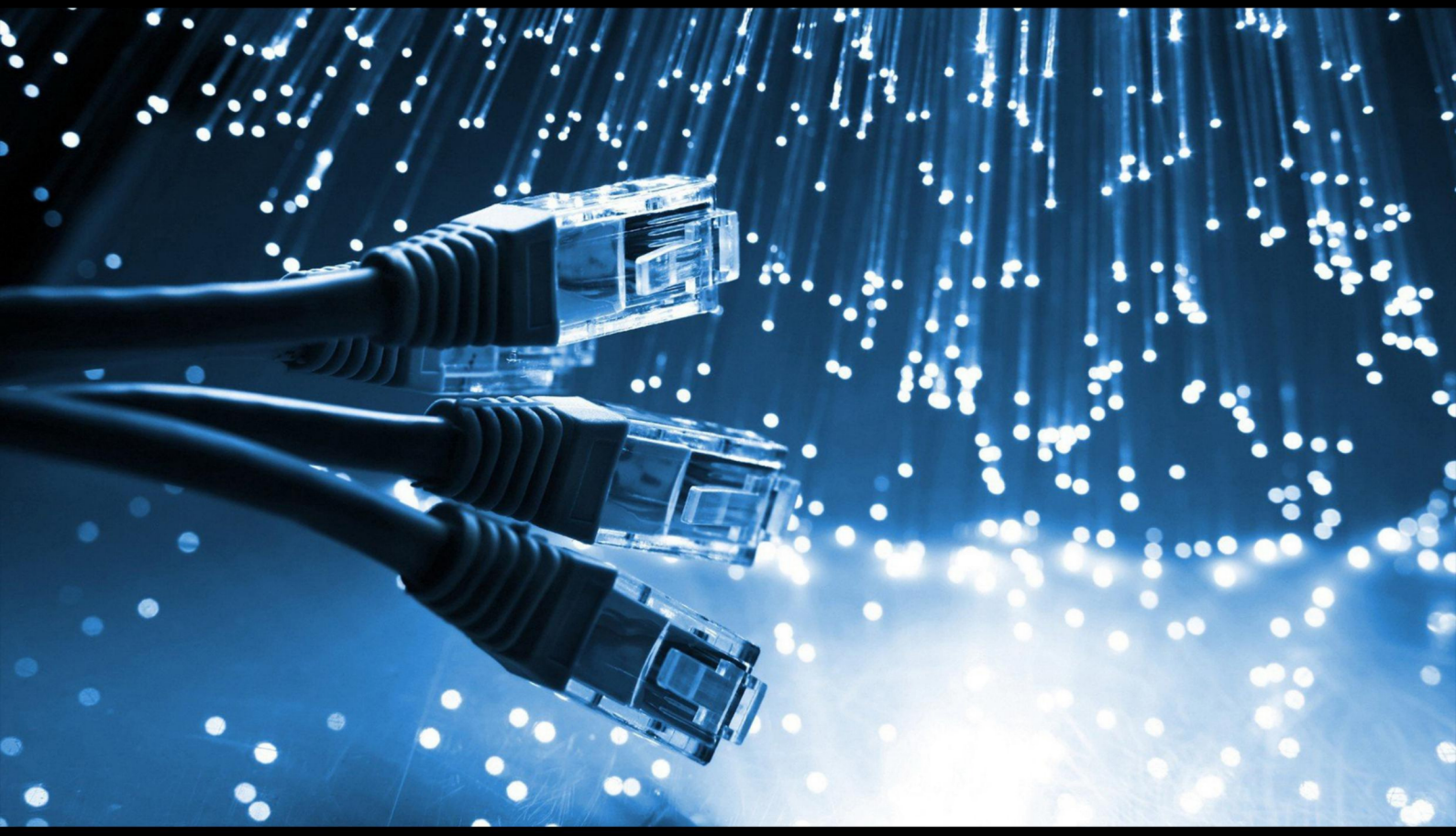
JCSI

Journal of Computer Sciences Institute

Volume 25/2022



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

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

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

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

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

Recenzenci numeru:

dr inż. Maria Skublewska-Paszkowska
dr inż. Małgorzata Plechawska-Wójcik
dr inż. Marek Miłosz, prof. uczelni
dr inż. Jakub Smołka
dr inż. Tomasz Nowicki
dr inż Sławomir Przyłucki
dr Mariusz Dzieńkowski
dr inż. Marcin Badurowicz
dr inż. Tomasz Szymczyk
dr inż. Krzysztof Dziedzic
dr inż. Jacek Kęsik
dr inż. Sylwester Korga
dr inż. Grzegorz Kozieł
dr Waldemar Suszyński
dr inż. Piotr Kopniak

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

Projekt okładki:
Marta Zbańska

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

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

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

Reviewers:

Maria Skublewska-Paszkowska
Małgorzata Plechawska-Wójcik
inż. Marek Miłosz, prof. uczelni
Jakub Smołka
Tomasz Nowicki
Sławomir Przyłucki
Mariusz Dzieńkowski
Marcin Badurowicz
Tomasz Szymczyk
Krzysztof Dziedzic
Jacek Kęsik
Sylwester Korga
Grzegorz Kozieł
Waldemar Suszyński
Piotr Kopniak

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

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 280–287
Received: 1 June 2022
Accepted: 17 October 2022

Comparison of hybrid and native iOS mobile application development technologies

Porównanie technologii hybrydowych i natywnych do wytwarzania aplikacji mobilnych iOS

Michał Kocki, Michał Urban*, Piotr Kopniak

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

Abstract
Mobile applications dedicated to iOS can be developed natively or hybrid. The subject of this paper is to compare these technologies on the example of a created mobile application. The performance of both technologies has been examined on the example of two test devices and smartphone operated by iOS. Considering the performance analysis, the compilation time of the application, reading and writing data from the cloud database, as well as the time of sorting the read data was examined. On the test devices, it was checked how intense is the use of the system. The obtained results confirm that producing a mobile application in native technology is more performance efficient.

Keywords: iOS; native technologies; hybrid technologies; performance

Streszczenie
Aplikacje mobilne dedykowane na system operacyjny iOS mogą być wytwarzane natywnie bądź hybrydowo. Tematyką artykułu jest porównanie tych technologii na przykładzie utworzonej aplikacji mobilnej. Zbadana została wydajność obu technologii na przykładzie dwóch urządzeń testowych i smartfona posiadającego system iOS. Biorąc pod uwagę analizę wydajności, został zbadany czas kompilacji aplikacji, odczyt i zapis danych z bazy danych w chmurze, a także czas sortowania odczytanych danych. Na urządzeniach testowych sprawdzono jak bardzo wytworzone aplikacje obciążają system. Otrzymane wyniki potwierdzają, że wytwarzanie aplikacji mobilnej w technologii natywnej jest wydajniejsze.

Słowa kluczowe: iOS; technologie natywne; technologie hybrydowe; wydajność

*Corresponding author
Email address: michal.urban@pollub.edu.pl (M. Urban)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

W ostatnich latach można zauważyć wzrost liczby wytwarzanych aplikacji mobilnych. Technologie elektroniki użytkowej, takie jak telefony, tablety, zegarki czy nawet inteligentni asystenci coraz częściej pojawiają się w domach. Aby nadążyć za rozwijającymi się urządzeniami, musiały pojawić się nowe technologie, które pozwoliłyby na umiejętne i szybkie wytwarzanie aplikacji. Dzięki temu postępowi zostały rozwinięte technologie hybrydowe, które w odróżnieniu od natywnych, pozwalają na obsługę wielu systemów operacyjnych jednocześnie.

Jedną z hybrydowych technologii wytwarzania aplikacji mobilnych jest Ionic – wydany w 2013 roku framework daje programistom ogromne możliwości: wytwarzanie aplikacji internetowych i mobilnych na systemy operacyjne iOS i Android pisząc jeden kod [1].

Technologie natywne mają znacznie mniejsze możliwości w kontekście wytwarzania aplikacji na wiele platform, lecz ich popularność jest wciąż porównywalna z technologiami hybrydowymi [2]. Jedną z nich jest język programowania Swift, który został opracowany przez Apple. Prosta struktura języka połączona z intuicyjnością środowiska programistycznego Xcode pozwala na wytworzenie natywnego rozwiązania w szybkim czasie. Ten język programowania jest kompatybilny z Objective-C – językiem programowania, który był rekomendowany przez Apple przed premierą języka Swift, służącym do wytwarzania aplikacji mobilnych [3].

Niniejszy artykuł ma na celu porównanie obu wyżej wymienionych technologii – Ionic i Swift – na podstawie aplikacji mobilnej.

W badaniu porównano czas kompilacji w obu technologiach gotowej aplikacji testowej. Następnie dokonano analizy parametrów zapisu i odczytu danych z bazy danych SQLite.

Znaleziono niewielką liczbę publikacji naukowych na temat porównania technologii hybrydowych i natywnych, o sprzecznych wnioskach. Jedną z nich jest publikacja D. Dobrzańskiego i W. Zabierowskiego, w której zostaje porównana technologia Xamarin i dwie technologie natywne na systemy operacyjne iOS i Android. Wyniki tej pracy pokazują, że wydajność aplikacji pisanych w obu technologiach jest porównywalna [4].

Kontrastująca do niej jest publikacja autorstwa P. Grzmila, M. Skublewskiej-Paszkowskiej, E. Łukasik i J. Smołki [5], w której autorzy porównują aplikacje napisane natywnie na system Android, z rozwiązaniem w technologii Xamarin. Ten artykuł naukowy wyraźnie

wskazuje, że technologie natywne mają lepszą wydajność względem hybrydowych.

Artykuł naukowy autorstwa T. Dorfer, L. Demetz, S. Huber [6] wskazuje na potencjalne wady technologii hybrydowych w kontekście wykorzystania zasobów urządzeń mobilnych. Wyniki tych badań pokazują, że aplikacje w technologiach natywnych mniej obciążają urządzenie pod kątem baterii, czy też procesora i pamięci RAM.

L. Corral, A. Janes, T. Remencius wydali artykuł naukowy „Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments" w 2012 roku. Praca pokazuje wady i zalety technologii hybrydowych w wielu aspektach – analizowane są potrzeby biznesowe, a także wyzwania i możliwości związane z wytwarzaniem aplikacji w technologiach multiplatformowych. Autorzy prowadzą dyskusję na ten temat, która pozwala wywnioskować, że wady i zalety technologii hybrydowych są definiowane i określane przez potrzeby aplikacji, która ma być wytwarzana [7].

Wśród publikacji dotyczących porównanie technologii wytwarzania aplikacji uniwersalnych i natywnych nie odnaleziono takich, które porównują technologię hybrydową (Ionic) i natywną (Swift) na platformę iOS.

2. Metodyka badań

W celu przeprowadzenia badań, wytworzone zostały dwie bliźniacze aplikacje służące do pomiaru liczby kroków użytkownika wykonanych w ciągu dnia. Pierwszy projekt został utworzony natywnie w języku Swift, drugi w hybrydowej technologii Ionic.

Do utworzenia aplikacji oraz do ich zbadania**,** wykorzystano następujące technologie oraz narzędzia:

- środowisko programistyczne Xcode,
- środowisko programistyczne Visual Studio Code,
- język programowania Swift,
- framework Ionic Capacitor,
- framework Angular,
- język TypeScript.

Pierwsze badanie zostało wykonane na dwóch urządzeniach testowych – laptopach z zainstalowanym systemem operacyjnm macOS Monterey 12.3.1. Pierwsze urządzenie posiada procesor Intel Core i5, 2 GHz; natomiast drugie urządzenie testowe posiada procesor Intel Core i5, 1.4 GHz. Oba urządzenia posiadają 16 GB pamięci RAM.

Badanie dotyczyło czasu kompilacji gotowej aplikacji testowej, która służy do pomiaru liczby kroków wykonanych w ciągu dnia. Celem tego badania było porównanie, która technologia posiada lepszą wydajność w kwestii kompilacji gotowego rozwiązania.

Aby otrzymać odpowiednią próbkę, która posłuży do interpretacji i obiektywnych wyników badania, wykonano kompilację gotowej aplikacji 25 razy.

Badanie zostało wykonane w środowisku Xcode, po uprzednim włączeniu funkcji odczytu szczegółowego czasu kompilacji aplikacji.

Kolejne badania dotyczyły zapisu i odczytu danych z lokalnej bazy danych w aplikacjach wytworzonych w obu technologiach.

W tym przypadku wykorzystana została lokalna baza danych SQLite, która została wybrana ze względu na swoją niezawodność i popularność w środowisku aplikacji mobilnych.

Pierwszy krok badania to uruchomienie aplikacji na urządzeniu testowym i zalogowanie się. Następnie wywołano funkcję zapisu danych z bazy przez odpowiedni przycisk. Każde wywołanie funkcji zapisu powoduje usunięcie wszystkich rekordów z tabeli i zapisanie w niej 100, 10 000 oraz 100 000 nowych rekordów z losowymi danymi. Przed zapisem danych uruchamiany jest z poziomu kodu stoper, który pozwala na zmierzenie czasu wykonywania tej operacji. Czas zapisu wyświetlany jest w konsoli debugującej w Xcode.

Kolejny krok to wywołanie funkcji odczytującej dane z bazy danych, która powoduje wysłanie zapytania w języku SQL i interpretację wyniku w odpowiedniej formie. Każde uruchomienie funkcjonalności pobiera i interpretuje w zależności od badania 100, 10 000 i 100 000 rekordów.

Dodatkowo po kliknięciu przycisku w aplikacji uruchamiany jest stoper wywoływany bezpośrednio z kodu aplikacji, który pozwala zmierzyć szczegółowy czas pobierania danych.

Dla uzyskania obiektywnych i wiarygodnych wyników, zapis, a także odczyt przykładowych danych powtórzono 25 razy.

3. Badania

Eksperymenty polegały na porównaniu wydajności aplikacji napisanych w technologiach Ionic Capacitor oraz iOS Swift. Porównywany był czas kompilacji ukończonej aplikacji testowej, a także wydajność w połączeniu z bazą danych SQLite. Każde badanie zostało wykonane w 25 próbach, aby zapewnić wiarygodność wyników.

3.1. Badanie czasu kompilacji

Tabela 1 przedstawia porównanie czasu kompilacji aplikacji pomiędzy dwiema technologiami – iOS i Ionic. Można zauważyć, że kompilacja w środowisku hybrydowym jest szybsza, niż natywnie.

Technologia natywna kompiluje gotową aplikację w czasie pomiędzy 18,099 s, a 21,967 s, co daje amplitudę różnicy czasu na poziomie 3,868 s. Technologia hybrydowa Ionic ma znacznie niższy wynik w tym przypadku – różnica najwyższego i najniższego czasu wynosi 1,331 s.

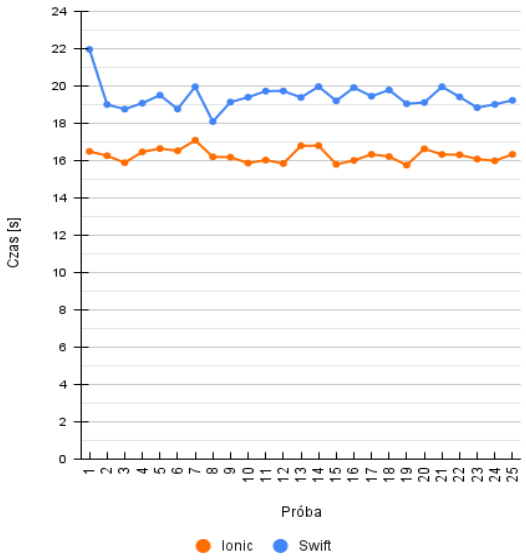
Na uwagę zasługuje również fakt, że technologia Ionic osadza swoją aplikację na widoku łączącym (Bridge Controller), który wyświetla natywnie aplikację w technologii webowej na urządzeniu mobilnym. Skutkiem jest to, że niezależnie od tego czy aplikacja jest pusta, czy ma w sobie funkcjonalności, to jej czas kompilacji jest podobny.

Technologia natywna kompiluje każdy element osobno i z każdym dodaniem nowego pakietu lub nowej funkcjonalności, czas kompilacji będzie coraz większy.

Na podstawie tych wyników możemy stwierdzić, że technologia Ionic szybciej kompiluje gotową aplikację niż Swift.

Tabela 1: Czas kompilacji w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	21,967	16,498
2	19,011	16,268
3	18,764	15,892
4	19,086	16,474
5	19,512	16,649
6	18,770	16,533
7	19,964	17,095
8	18,099	16,207
9	19,143	16,190
10	19,402	15,872
11	19,732	16,035
12	19,737	15,848
13	19,389	16,800
14	19,973	16,807
15	19,209	15,807
16	19,920	16,013
17	19,458	16,341
18	19,789	16,224
19	19,054	15,764
20	19,121	16,636
21	19,963	16,335
22	19,421	16,318
23	18,845	16,087
24	19,021	15,995
25	19,235	16,340



Rysunek 1: Porównanie czasu kompilacji gotowej aplikacji w obu technologiach.

3.2. Badanie czasu zapisu do bazy danych

Kolejnym etapem badań było porównanie czasów zapisu danych, który wyrażono w sekundach, które zostało wykonane na urządzeniu testowym – smartfonie iPhone 12 mini, posiadającym system iOS. Aplikacje podłączały się do lokalnej bazy danych SQL obsługiwanej przez system SQLite. Aplikacje wysyłały dane przy pomocy zapytań SQL.

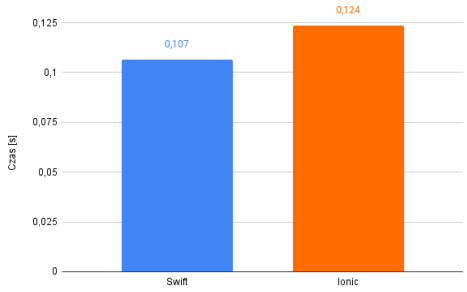
W tym przypadku udostępniane dane to liczba kroków użytkownika i identyfikator czasowy.

3.2.1. Zapis 100 rekordów do bazy

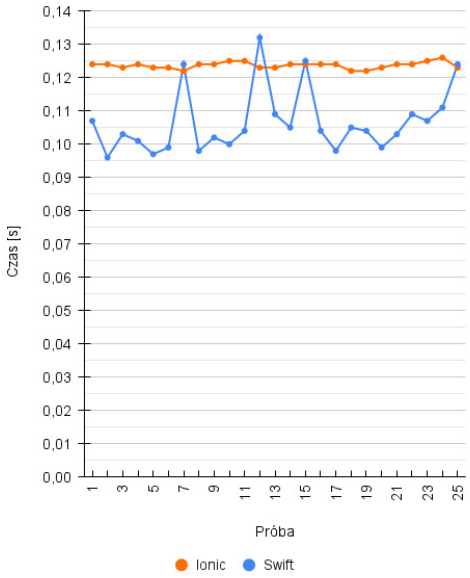
Na podstawie tabeli 2 można wywnioskować, że, czas zapisu 100 rekordów do bazy danych jest zauważalnie krótszy w przypadku języka programowania Swift. Potwierdza to rysunek 2, na którym porównano uśrednione czasy zapisu. Różnica pomiędzy badanymi technologiami jest na poziomie 14%.

Tabela 2: Czas zapisu 100 rekordów do bazy danych w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	0,107	0,124
2	0,096	0,124
3	0,103	0,123
4	0,101	0,124
5	0,097	0,123
6	0,099	0,123
7	0,124	0,122
8	0,098	0,124
9	0,102	0,124
10	0,100	0,125
11	0,104	0,125
12	0,132	0,123
13	0,109	0,123
14	0,105	0,124
15	0,125	0,124
16	0,104	0,124
17	0,098	0,124
18	0,105	0,122
19	0,104	0,122
20	0,099	0,123
21	0,103	0,124
22	0,109	0,124
23	0,107	0,125
24	0,111	0,126
25	0,124	0,123



Rysunek 2: Porównanie uśrednionego czasu zapisu 100 rekordów do bazy danych.

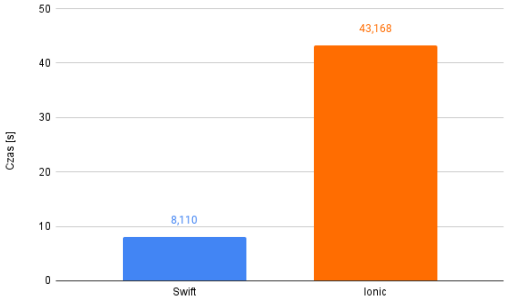


Rysunek 3: Porównanie czasów zapisu 100 rekordów do bazy danych.

3.2.2.Zapis 10 000 rekordów do bazy

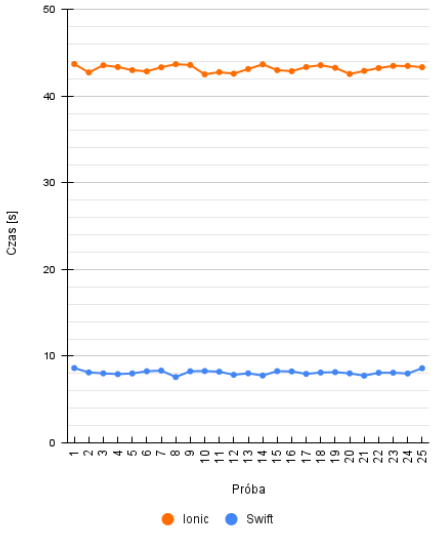
Tabela 3: Czas zapisu 10 000 rekordów do bazy danych w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	8,637	43,691
2	8,133	42,719
3	8,025	43,544
4	7,935	43,344
5	8,016	42,984
6	8,265	42,842
7	8,342	43,312
8	7,599	43,674
9	8,266	43,576
10	8,291	42,500
11	8,217	42,754
12	7,857	42,578
13	8,029	43,112
14	7,774	43,654
15	8,278	42,984
16	8,246	42,866
17	7,963	43,341
18	8,118	43,557
19	8,164	43,254
20	8,022	42,541
21	7,756	42,897
22	8,093	43,225
23	8,100	43,472
24	8,005	43,465
25	8,616	43,324



Rysunek 4: Porównanie uśrednionego czasu zapisu 10 000 rekordów do bazy danych w obu technologiach.

Tabela 3 prezentuje wyniki badania zapisu 10 000 rekordów do bazy danych. Można wywnioskować, że znaczną przewagę ma w tym przypadku język programowania Swift. Framework Ionic zapisuje taką liczbę rekordów średnio o 85% wolniej (rysunek 4).

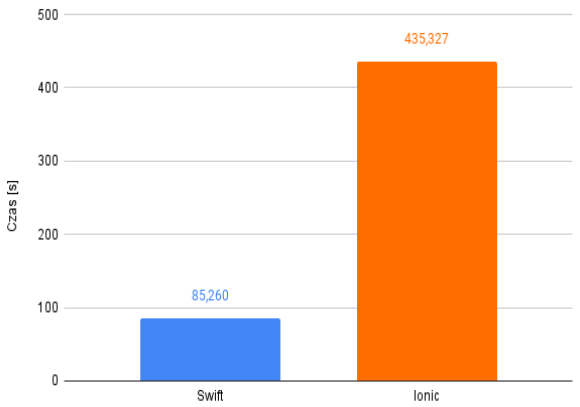


Rysunek 5: Porównanie czasów zapisu 10 000 rekordów do bazy danych.

3.2.3. Zapis 100 000 rekordów do bazy

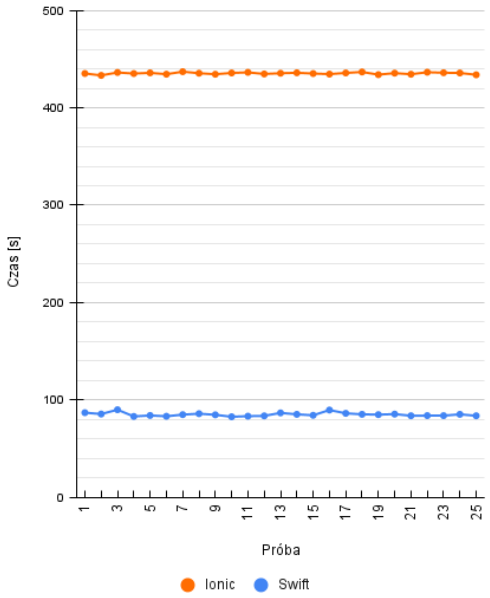
Tabela 4: Czas zapisu 100 000 rekordów do bazy danych w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	87,127	435,182
2	85,748	433,232
3	90,161	436,214
4	83,275	435,120
5	84,265	435,841
6	83,446	434,458
7	85,124	437,032
8	86,098	435,463
9	84,944	434,412
10	82,952	435,741
11	83,542	436,341
12	83,864	434,698
13	86,905	435,442
14	85,432	435,911
15	84,451	435,132
16	89,843	434,546
17	86,465	435,741
18	85,451	436,741
19	85,131	433,986
20	85,669	435,541
21	83,985	434,421
22	84,128	436,471
23	84,090	435,943
24	85,441	435,713
25	83,956	433,841



Rysunek 6: Porównanie uśrednionego czasu zapisu 100 000 rekordów do bazy danych w obu technologiach.

Tabela 4 prezentuje wyniki badania zapisu do lokalnej bazy danych w wymiarze 100 000 rekordów. Średnia dla języka programowania Swift to w tym przypadku 85,260 s, a dla frameworka Ionic – 435,327 s (rysunek 6). Wyniki różnią się średnio o 350 s na korzyść języka programowania Swift.



Rysunek 7: Porównanie czasów zapisu 100 000 rekordów do bazy danych.

Wyniki badania czasu zapisu w bazie danych prezentują znaczną przewagę technologii natywnej nad technologią hybrydową. Dla każdej liczby rekordów średni czas zapisu do bazy danych jest o dużo niższy w technologii Swift.

Dodatkowo, rysunek 5 pokazuje następujące amplitudy wahania czasu zapisu: 1,038 s dla aplikacji napisanej w technologii Swift; i 1,191 s dla aplikacji hybrydowej wykonanej w technologii Ionic. Wahania w obu przypadkach nie są znaczące (ok. 1 s), więc można stwierdzić, że oba rozwiązania mają podobną niezawodność oraz stabilność wykonywanych operacji.

Na podstawie powyższego badania można wywnioskować, że technologia natywna znacznie szybciej obsługuje lokalną bazę danych.

3.3. Badanie czasu odczytu danych z bazy danych

Następnie zbadano szybkość odczytu danych, która została wyrażona w sekundach. Podobnie jak dla zapisu danych, wykorzystana została lokalna baza danych oparta o SQLite. Odczytane dane zostały wypisane w konsoli debugującej aplikację. Tabele 5 oraz 6 przedstawią wyniki czasowe odczytu danych z bazy kolejno dla 10 000 i 100 000 rekordów w obu technologiach.

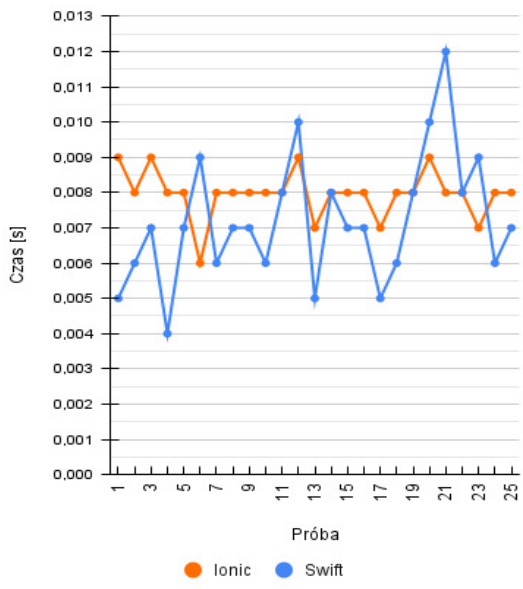
Odczyt 100 rekordów z bazy

Tabela 5: Czas odczytu 100 rekordów z bazy danych w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	0,005	0,009
2	0,006	0,008
3	0,007	0,009

4	0,004	0,008
5	0,007	0,008
6	0,009	0,006
7	0,006	0,008
8	0,007	0,008
9	0,007	0,008
10	0,006	0,008
11	0,008	0,008
12	0,010	0,009
13	0,005	0,007
14	0,008	0,008
15	0,007	0,008
16	0,007	0,008
17	0,005	0,007
18	0,006	0,008
19	0,008	0,008
20	0,010	0,009
21	0,012	0,008
22	0,008	0,008
23	0,009	0,007
24	0,006	0,008
25	0,007	0,008

Badanie odczytu 100 rekordów z bazy danych wskazuje nam, że obie porównywane technologie radzą sobie z tym procesem podobnie. Na podstawie rysunku 8 można wywnioskować, że czasy odczytu między technologiami mają marginalną różnicę.



Rysunek 8: Porównanie czasów odczytu 100 rekordów z bazy danych.

Na podstawie powyższych danych można obliczyć prędkość odczytu rekordów. Swift (prędkość odczytu na poziomie 14285 rekordów na sekundę) w tym przypadku również wygrywa z frameworkiem Ionic (prędkość odczytu na poziomie 12500 rekordów na sekundę), osiągając wynik o 25% wyższy.

3.3.1. Odczyt 10 000 rekordów z bazy

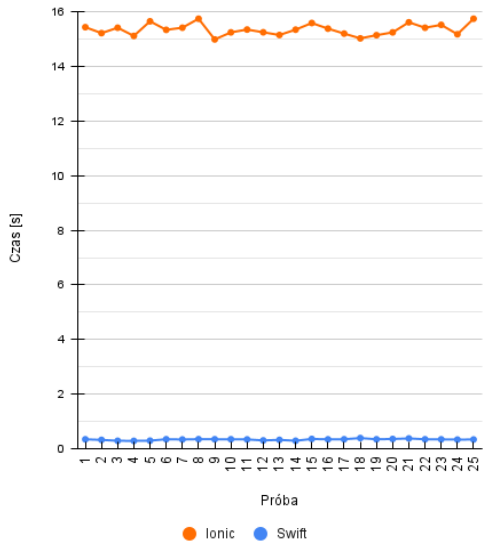
Tabela 6: Czas odczytu 10 000 rekordów z bazy danych w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	4,493	15,432
2	4,814	15,214
3	4,041	15,412
4	4,555	15,112
5	3,917	15,647
6	4,655	15,335
7	2,624	15,413
8	5,218	15,741
9	5,273	14,984
10	4,720	15,242
11	5,315	15,345
12	5,289	15,245
13	4,728	15,146
14	4,625	15,341
15	2,677	15,584
16	3,847	15,379
17	4,046	15,197
18	5,006	15,023
19	3,546	15,139
20	3,415	15,242
21	3,117	15,611
22	4,284	15,412
23	3,812	15,517
24	3,783	15,173
25	4,281	15,741

Czas odczytu danych różni się między sobą w poszczególnych próbach. Na podstawie rysunku 9 można wywnioskować, że aplikacja napisana w języku programowania Swift jest stabilniejsza w kwestii odczytu danych z bazy danych.

Na podstawie wyników (tabela 6) możemy również wyliczyć liczbę odczytanych rekordów na sekundę. Wynik dla języka programowania to dużo większa liczba odczytanych rekordów na poziomie 29 tysięcy rekordów na sekundę. Framework Ionic odczytuje w tym

samym czasie tylko 652 rekordy, co potwierdza przewagę natywnego rozwiązania.

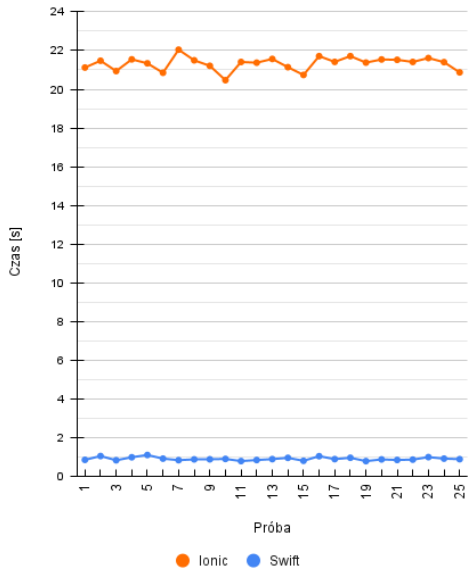


Rysunek 9: Porównanie czasów odczytu 10 000 rekordów z bazy danych.

3.3.2.Odczyt 100 000 rekordów z bazy

Tabela 7: Czas odczytu 100 000 rekordów z bazy danych w technologii Ionic i Swift

Lp.	Swift – czas [s]	Ionic – czas [s]
1	0,861	21,123
2	1,058	21,475
3	0,840	20,941
4	0,995	21,546
5	1,112	21,341
6	0,925	20,854
7	0,845	22,046
8	0,886	21,497
9	0,890	21,212
10	0,909	20,472
11	0,798	21,411
12	0,852	21,374
13	0,899	21,567
14	0,965	21,142
15	0,812	20,741
16	1,051	21,712
17	0,899	21,416
18	0,969	21,713
19	0,796	21,379
20	0,884	21,541
21	0,856	21,522
22	0,873	21,413
23	1,003	21,614
24	0,925	21,402
25	0,896	20,875



Rysunek 10: Porównanie czasów odczytu 100 000 rekordów z bazy danych.

Badanie odczytu z bazy danych wykonane na 100 000 rekordach wykazało, że język Swift podczas odczytu danych z bazy danych jest wydajniejszy od technologii hybrydowej. Framework Ionic zapisuje taką liczbę rekordów średnio o około 20 sekund wolniej od języka programowania Swift.

Na podstawie wykresu (rys. 10) możemy wywnioskować, że Swift stabilniej odczytuje dane z bazy danych.

W przypadku 100 000 rekordów, prędkości zapisu dla obu technologii mają duże różnice. Framework Ionic w przypadku odczytu takiej ilości danych jest wolniejszy i osiąga prędkość 4687 rekordów na sekundę, natomiast język programowania Swift odczytuje dane z prędkością ponad 111 tysięcy rekordów na sekundę.

Odczyt 100, 10 000, a także 100 000 rekordów z bazy danych, zbadany zarówno dla języka programowania Swift, jak i frameworku Ionic potwierdza, że język programowania Swift posiada szybszy odczyt z bazy danych niż framework Ionic.

4. Wnioski

W artykule została przedstawiona analiza porównawcza technologii natywnej z wykorzystaniem języka Swift oraz hybrydowej Ionic na przykładzie autorskiej aplikacji mobilnej z identycznymi funkcjonalnościami. Aplikacja została przetestowana pod względem czasu

kompilacji, wydajności w połączeniu z lokalną bazą danych SQLite, mechanizmów sortowania oraz zużycia zasobów urządzenia.

Na podstawie wyników, które udało się otrzymać podczas badania, można stwierdzić, że aplikacja wytworzona natywnie kompiluje się wolniej, niż aplikacja napisana w technologii hybrydowej. Natomiast biorąc pod uwagę połączenie z lokalną bazą danych, można zdecydowanie stwierdzić, że technologia natywna Swift jest szybsza zarówno podczas zapisu, jak i odczytu danych z SQLite.

Jeżeli chcemy zdecydować się na tworzenie aplikacji tylko na system iOS z pominięciem systemu Android, szybszym rozwiązaniem jest wybranie technologii natywnej. W przypadku, gdy nasza aplikacja nie polega na funkcjach obciążających zasoby jak np. streaming to lepszym wyborem jest aplikacja hybrydowa. Jeden deweloper jest w stanie stworzyć jeden kod na system iOS, Android, aplikacje internetową oraz desktopową. Wystarczy specjalista od technologii webowych co jest dużą oszczędnością budżetu porównując do zatrudnienia czterech deweloperów specjalizujących się w każdej z wymienionych technologii.

Literatura

[1] Introduction to Ionic, https://ionicframework.com/docs#license, [18.01.2022]

[2] T. Vilček, T. Jakopec, Comparative analysis of tools for development of native and hybrid mobile applications, 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (2017) 1516-1521.

[3] The Swift Programming Language Documentation, https://docs.swift.org/swift-book/index.html, [18.01.2022]

[4] D. Dobrzański, W. Zabierowski, The Comparison of Native Apps Performance on iOS (Swift) and Android with Cross-platform Application – Xamarin. IJMCS Journal 8(3) (2017) 122-126.

[5] P. Grzmil, M. Skublewska-Paszkowska, E. Łukasik, J. Smołka, Performance Analysis of Native and Cross-platform Mobile Applications. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska 7 (2017) 50-53.

[6] T. Dorfer, L. Demetz, S. Huber, Impact of mobile cross-platform development on CPU, memory and battery of mobile devices when using common mobile app features, Procedia Computer Science 175 (2020) 189-196.

[7] L. Corral, A. Janes, T. Remencius: Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments, Procedia Computer Science 10 (2012) 1202–1207.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 288–296
Received: 9 June 2022
Accepted: 12 September 2022

Comparative analysis of web application interfaces with buttons in graphical and text form for universal design

Analiza porównawcza interfejsów aplikacji internetowych z przyciskami w formie graficznej i tekstowej do celów projektowania uniwersalnego

Adriana Osmulska*, Jakub Kaliszuk*, Małgorzata Plechawska-Wójcik, Mariusz Dzieńkowski

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

Abstract
The purpose of this article is to compare application interfaces with buttons in graphical and text form. For the needs of the research, a web application was prepared with the functionality to change the form of buttons. The analysis of the availability of the application was performed using the WAVE tool. A research group of 10 students was prepared for the eyetracking experiment and form. The surveys conducted on the eyetracker allowed to assess which version of the interface enables faster and correct execution tasks. LUT's list was used to test the quality of the designed interface.

Keywords: accessibility; web application; interface

Streszczenie
Celem niniejszego artykułu jest porównanie interfejsów aplikacji z przyciskami w formie graficznej i tekstowej. Na potrzeby przeprowadzenia badań przygotowano aplikację internetową z funkcją umożliwiającą zmianę formy przycisków. Analiza dotycząca dostępności aplikacji została wykonana przy pomocy narzędzia WAVE. Do przeprowadzenia eksperymentu z wykorzystaniem okulografu i ankiety przygotowano grupę badawczą stanowiącą 10 studentów. Badania przeprowadzone za pomocą okulografu pozwoliły ocenić, która wersja interfejsu umożliwia szybsze i poprawne wykonanie zadań. Do zbadania jakości wykonanego interfejsu użyto listy kontrolnej LUT.

Słowa kluczowe: dostępność; aplikacja internetowa; interfejs

*Corresponding author
Email addresses: adriana.osmulska@pollub.edu.pl, jakub.kaliszuk@pollub.edu.pl, m.plechawska@pollub.pl, m.dzienkowski@pollub.pl

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Celem niniejszego artykułu jest zbadanie różnic i podobieństw w podejściu użytkownika do aplikacji udostępniającej te same możliwości jednak za pośrednictwem dwóch różnych interfejsów, gdzie zasadniczą różnicą jest rodzaj zastosowanych przycisków. W pierwszej aplikacji dostępne są przyciski w formie tekstowej, w drugiej natomiast zastosowano ikony. Badanie zostało przeprowadzone z użyciem technologii śledzenia ruchu oczu zwanej eyetrackingiem. Uzyskane wyniki pozwoliły na ocenę jak taka różnica w interfejsie wpłynie na szybkość wykonywania akcji podczas korzystania z aplikacji.

W odniesieniu do celu wyszczególniono kilka pytań badawczych:

- Która wersja interfejsu pozwoli użytkownikom na szybsze wykonywanie czynności w systemie?
- Która wersja interfejsu będzie bardziej dostępna dla użytkowników?
- Czy rodzaj przycisku ma wpływ na długość fiksacji (koncentracji) wzroku użytkownika?
- Jak subiektywna ocena interfejsu odnosi się do wyników przeprowadzonych badań?

Postawiona została teza, że użycie przycisków w formie graficznej pozwalają na szybsze wykonywanie akcji przez użytkownika.

2. Przegląd literatury

Tworząc aplikacje internetowe oraz innego rodzaju oprogramowanie należy zwrócić uwagę na docelową grupę użytkowników, w której mogą znaleźć się również osoby niepełnosprawne. W związku z tym przeprowadzono przegląd literatury pod kątem zagadnień, takich jak dostępność (ang. A*ccessibility*), okulografia (ang. *Eye-tracking*) oraz web design. Poświęcono też uwagę artykułom poruszającym tematy z dziedziny interfejsów i ich optymalizacji pod kątem osób niepełnosprawnych.

Informacje na temat dostępności serwisów, stron i aplikacji internetowych dla osób niepełnosprawnych można znaleźć w pracach [1-6]. Tematyka dostępności w odniesieniu do rozwiązań informatycznych jest zagadnieniem wciąż rozwijającym się, dlatego liczba badań w tym zakresie wciąż jest niewielka. Rzeczone prace traktują o przystosowaniu Internetu do potrzeb osób niepełnosprawnych. Celem tych prac było ustalenie, czy można zaprojektować interfejs, który w szczególności zaspokoi potrzeby użytkowników słabowidzących, poprawi ich wydajność i ogólne wrażenia z korzystania z aplikacji.

W artykule [6] dokonano przeglądu konkretnych barier w dostępie do komputera i sieci, z jakimi borykają się pacjenci z niepełnosprawnością wzroku. Opisano metody oceny klinicznej, podsumowano tradycyjne oraz

nowsze metody wspomagające technologie komputerowe dla zapewnienia powszechnej dostępności dla słabowidzących, omówiono także pojawiające się technologie i przyszłe kierunki w tym obszarze. W badaniu [7] Giorgio Brajnik proponuje metodę badania dostępności stron internetowych nazwaną "Barrier walkthrough". Polega ona na przygotowaniu bardzo szczegółowych scenariuszy uwzględniając kategorie użytkowników (np. użytkowników słabowidzących lub użytkowników z upośledzeniem ruchowym). W następnym kroku należy określić cele użytkownika, które najczęściej definiuje się jako konkretne przypadki użycia. Następnie należy określić bariery, jakie mogą wystąpić dla konkretnego użytkownika. Lista barier zdefiniowana jest w punktach kontrolnych dostępności WCAG 2.0 [8]. Następnie strona jest badana biorąc pod uwagę utworzone przypadki testowe. Każda bariera ma swoje konsekwencje, które zostały podzielone na cztery kategorie: skuteczność (możliwość osiągnięcia wyznaczonego celu, błędy i problemy podczas realizacji zadania), produktywność (ilość czasu, wysiłku, zasobów poświęconych na rozwiązanie konkretnego zdania, satysfakcja (zadowolone i frustracja podczas wykonywania zadania) bezpieczeństwo (określenie jak duże przełożenie mają występujące błędy na bezpieczeństwo użytkownika, utratę danych itp.).

W badaniu [4] wykorzystano m.in modyfikację metody „Barrier walkthrough" zaproponowaną przez Giorgio Brajnika [7] i WCAG 2.0 pod kątem treści internetowych. Ta metoda stosuje metodę ekspercką. Technika ta polega na ustalaniu priorytetów oddziaływań barier w zależności od kontekstu. Metoda umożliwia określenie nasilenia każdej z barier i identyfikację problemów z dostępnością, przy zastosowaniu skali opisowej.

W pracy badawczej [9] udowodniono, że zagadnienie dostępności jest ważne podczas tworzenia stron i aplikacji internetowych. Pokazuje, że zagadnienie to jest związane z cechą użyteczność (ang. *Usability*) i należy dążyć, żeby strona, poza tym, że jest użyteczna, była dostępna jednakowo dla osób w pełni sprawnych jak i tych niepełnosprawnych.

W badaniu [9] wymieniono też najpopularniejsze (stan na rok 2017) walidatory i inne przydatne narzędzia pomocne w badaniu różnych apeksów stron internetowych. Wśród nich zostały wymienione takie jak: HTML Validator [18] , Utilitia [19], Wave [20], Functional Accessibility Evaluator [21], aChecker [22], TAW Web Accessibility Test [23]. W pracy [12] zostały zbadane różnice w doświadczeniach użytkownika (ang, *User experience*) z pracą z przyciskami GUI utworzonymi w oparciu o Google Map i OpenStreetMap. Dla obu aplikacji zostały przygotowane zadania, na podstawie których porównywano obie aplikacje. Kluczową różnicą w obu przypadkach jest rozmieszczenie przycisków. W przypadku Google Maps przyciski znajdują się w trzech rogach ekranu, co powoduje fiksację oczu właśnie w tych miejscach. W przypadku OpenStreetMap przyciski znajdują się w dwóch rogach ekranu, co skutkuje zmniejszoną aktywnością oczu. Użytkownicy, w każdej aplikacji, mieli do zrobienia trzy zadania. Przy pomocy okulografu zbadano jak przebiegał proces ich rozwiązywania.

W pracy [13] poświęcono uwagę zagadnieniu interakcji użytkowników z wynikami wyszukiwarki WWW. Wyniki uzyskane przez badaczy pokazują w jaki sposób uczestnicy badania (tutaj główne studenci) selekcjonują wyniki wyszukiwarki dla różnego typu zapytań i jak długo zajmuję im wybranie odnośnika. Czasy te różnią się i wynoszą kolejno od 5 do 6 dla prostych i 11 sekund dla bardziej skomplikowanych zapytań. Ponadto wartym zauważenia jest fakt, że największą uwagę badani poświęcali pierwszemu i drugiemu wynikowi z 11 prezentowanych. Na wyniki prezentowane na pozycji trzeciej, czwartej i piątej poświęcali porównywalnie tyle samo czasu co dla pozycji pierwszej. Natomiast pozostałe pozycje skupiły uwagę badanych na ok. 0,2 sekundy. Praca pokazuje, że dla użytkowników ważne jest pozycjonowanie elementów na ekranie. Im wyżej coś się znajduje tym jest dla nich ważniejsze, a elementy znajdujące się niżej często mogą być po prostu pomijane.

Kolejna praca, badająca podejście użytkownika do zaprojektowanego interfejsu [14] pokazuje w jaki sposób ludzie postrzegają tą samą aplikację w różnych kontekstach. W badaniu przedstawiono dwie aplikacje stworzone w oparciu o popularne media społecznościowe (takie jak Twitter czy Facebook) z czego jedna jest aplikacją natywną stworzoną na system Android, natomiast druga jest aplikacją PWA (ang. *Progressive Web Application*). Funkcjonalnie aplikacje umożliwiają wykonanie tych samych czynności, jednak zaproponowana warstwa wizualna zawiera pewne różnice. Mimo zauważalnych różnic w interfejsach użytkownicy ocenili obie aplikacje jako spójne.

Badanie [15] przedstawia w jaki sposób zwyczajowo realizowane jest podejście do badania z użyciem okulografu. Badana tam aplikacja "FastCat" sprawdzana jest na konkretnych wcześniej przygotowanych przypadkach. Opracowanych zostało pięć scenariuszy a później sprawdzone były czasy ich wykonania, liczba zarejestrowanych fiksacji wzroku oraz liczba popełnionych przez użytkowników błędów. Dzięki temu można konkludować jakie błędy i kluczowe miejsca do poprawy ma strona.

Badanie w pracy [16] miało na celu sprawdzić sposób, w jaki ludzie szukają ikon na ekranie. Uczestnikami eksperymentu było 10 studentów studiów licencjackich Rice University. Badanie wykazało, że uczestnicy rzadko skupiali wzrok na ikonach, które znali wcześniej, oraz że uczestnicy stosowali skuteczną strategię wyszukiwania polegającą na badaniu ikon rozpraszających znajdujących się najbliżej ich aktualnego punktu skupienia.

Badania w artykule [17] polegały na stworzeniu inteligentnego algorytmu do poprawy jakości interfejsów w oparciu o UX badany przy pomocy okulografu. W badaniu użyto przykładowego interfejsu odtwarzacza multimedialnego. Badanie miało na celu śledzenie ruchów oczu w przypadku ośmiu użytkowników oraz ich interakcji ze stworzonym interfejsem. Uzyskane dane

były później punktem wyjścia dla utworzenia nowej generacji widoku odtwarzacza, który w założeniu miał być lepszy od poprzedniej iteracji. Był on poprawiany przez sztuczną inteligencję aż do 10. generacji w oparciu o użyty i opracowany w badaniu algorytm. Jest to nowatorskie badanie w tej dziedzinie i może być punktem wyjścia dla kolejnych badań tego typu i udoskonalenia algorytmów, gdzie to użytkownicy poprzez interakcje z oprogramowaniem będą mogli pomóc twórcom oprogramowania w designerom w tworzeniu coraz to lepszych warstw wizualnych aplikacji.

3. Metody badawcze w analizie interfejsów

W pracach polegających na badaniu interfejsów użytkownika stosuje się wiele metod badawczych. Jedną z nich jest metoda heurystyk [11]. Polega ona na eksperckiej ocenie w jakim stopniu badana strona internetowa spełnia przyjęte normy i heurystyki opracowane na przestrzeni lat przez różnych badaczy zajmujących się tematem np. heurystyki Nielsena [10]. Kolejna metodą jest metoda wędrówki poznawczej (ang. *Cognitive walkthrough*) [7]. Test polega na wykonywaniu przez eksperta wcześniej zdefiniowanych i przygotowanych zadań i szczegółową ocenę. Ta metoda często jest powiązana z metodą heurystyczną i pomaga w ocenie strony przez eksperta wykonującego badanie. Kolejna metoda to testy użyteczności [11]. Polega na wykonaniu zadań przez użytkownika, który może przekazać swoje uwagi i spostrzeżenia na temat badanych aplikacji. Często podczas badania wykorzystuje się analizę ruchu oczu. Pozwala dokładnie prześledzić jak użytkownik korzysta z przygotowanego interfejsu. Dzięki zastosowaniu tej metody można zobaczyć na co badany zwraca uwagę i jak dużo czasu poświęca na poszczególne elementy.

4. Badania

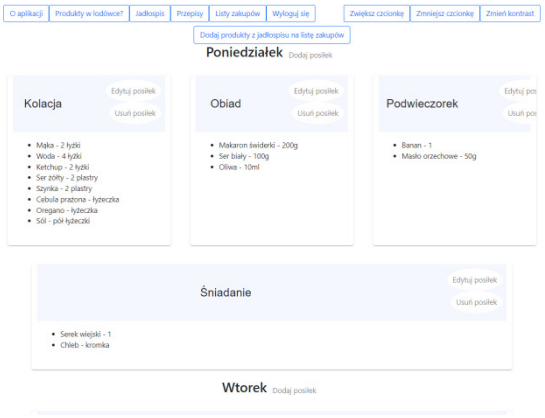
Metoda wykorzystywana w niniejszym badaniu to śledzenie ruchu oczu. Badania zastały przeprowadzone na przygotowanych zrzutach ekranów gotowej aplikacji. Na ich podstawie przygotowano zbiór zadań badawczych, które zostały wykonane przez osoby biorące udział w eksperymencie.

4.1. Aplikacja testowa

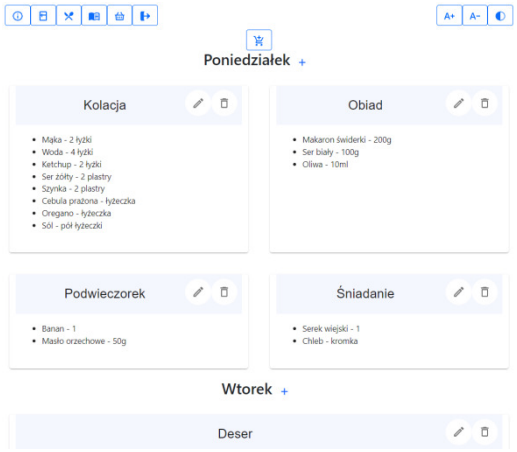
Na potrzeby eksperymentu została stworzona aplikacja służąca do kontrolowania zawartości lodówki, planowania posiłków i list zakupów napisana w technologii React JS, wykorzystująca bazę danych Firebase. Została uzupełniona przykładowymi danymi co pozwoliło na stworzenie scenariuszy, które później zostały użyte do przeprowadzenia badań. Aplikacja dodatkowo została rozbudowana o specjalnie zaprojektowaną funkcję zmieniającą przyciski z formy graficznej na tekstową i odwrotnie, aby ułatwić przeprowadzenie badań. Na Rysunkach 1-2 przedstawiono przykładową stronę interfejsu aplikacji w dwóch przygotowanych wersjach.

4.2. Plan badań

Badania zostały podzielone na trzy części: sprawdzenie interfejsów obu wersji aplikacji z użyciem narzędzia WAVE, przeprowadzenie eksperymentu przy użyciu okulografu i wykonanie ankiety przez grupę badawczą.



Rysunek 1: Przykład interfejsu z przyciskami w formie tekstowej.



Rysunek 2: Przykład interfejsu z przyciskami w formie graficznej.

WAVE jest internetowym narzędziem służącym do walidacji stron i aplikacji internetowych pod kątem ich dostępności. Przy jego pomocy można sprawdzić czy na stronie znajdują się błędy, które mogą stanowić przeszkodę w korzystaniu z niej przez osoby z niepełnosprawnościami.

W eksperymencie została przygotowana próba badawcza 10. uczestników podzielonych na dwie grupy. Grupa, która dostała do wykonania 10 zadań z wykorzystaniem widoków aplikacji z przyciskami w formie graficznej - dalej nazywana "Metoda A" oraz grupa, która dostała te same zadania, ale do wykonania z użyciem tekstowych odpowiedników przycisków - dalej nazywana "Metoda B". Próba badawcza została dobrana w taki sposób, żeby uczestnicy mieli doświadczenie w projektowaniu interfejsów. Wszyscy uczestnicy byli studentami Politechniki Lubelskiej kierunku Informatyka, w wieku 23-25 lat.

Ankieta oceniająca jakość interfejsu aplikacji jest zmodyfikowaną listą kontrolną LUT [24] składającą się z 18. pytań z odpowiedziami w skali Likerta. Ankieta

została wypełniona przez uczestników, których znajomość aplikacji ograniczała się tylko do poznania interfejsu podczas badania na okulografie. Z powodu przyjętej metodyki badań, uczestnicy nie byli wstanie odpowiedzieć na pytania ze wszystkich obszarów oryginalnej listy LUT. Podobszary wykorzystane w ankiecie to: łatwość nawigowania, hierarchia informacji, struktura informacji, elementy ekranu - z obszaru "Nawigacja i struktura", pytania z podobszarów: layout i dobór barw - z obszaru "Interfejs aplikacji" z pominięciem pytań o dostosowaniu layoutu do różnych rozdzielczości i urządzeń mobilnych, oraz podobszary: teksty, nazewnictwo i etykiety - z obszaru "Tekst podstron". Pytania z obszarów "Komunikaty, feedback, pomoc dla użytkownika" i "Wprowadzanie danych" zostały pominięte. Ankieta została udostępniona badanym na Google Forms bezpośrednio po wykonaniu eksperymentu na okulografie.

Dane zebrane podczas badań zostały poddane analizie ilościowej i jakościowej. Wyniki zostały przeanalizowane oddzielnie dla każdego z etapów badania.

4.3. Scenariusz przebiegu badań

Na potrzeby eksperymentu zdefiniowano następujące scenariusze badawcze:

- S1: znalezienie przycisku, który przeniesie użytkownika na zakładkę z przepisami;
- S2: znalezienie sposobu na zmianę kontrastu;
- S3: znalezieniu przycisku dodającego posiłek do jadłospisu;
- S4: znalezienie przycisku dodającego jadłospis na listę zakupów;
- S5: wysłanie listy zakupów na E-mail użytkownik;
- S6: edytowanie posiłku “Śniadanie” w dniu poniedziałek;
- S7: usunięcie listy zakupów o nazwie “Sobota”;
- S8: dodanie brakujących produktów z lodówki na listę zakupów;
- S9: znalezienie elementu nawigacyjnego służący do wylogowania;
- S10: dodanie produktu do swojej lodówki.

Po wykonaniu badania na okulografie uczestnicy dostali do wypełnienia ankietę oceniającą jakość aplikacji.

4.4. Zastosowane metryki

W badaniach okulograficznych wyróżnia się podstawowe metryki bazujące przede wszystkim na pojęciach fiksacji (koncentracji) wzroku definiowanego jako mierzalne momenty zatrzymania wzroku w konkretnym punkcie oraz sakady definiowanej jako skokowe punkty pomiędzy dwoma punktami [27], pozwalające na późniejszą interpretacje uzyskanych wyników:

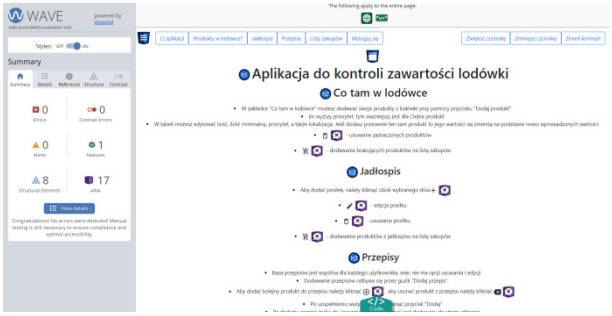
- Czas do pierwszej fiksacji wzroku [25] - czas mierzony od pojawienia się bodźca do wykrycia pierwszej fiksacji. Pozwala on sprawdzić jak dużo czasu zajmuje badanemu odnalezienie zadanego elementu.
- Czas trwania fiksacji wzroku [25] - czas skupienia wzroku na danym bodźcu. Jest to bardzo ogólna metryka, która służy sprawdzeniu przez jak długo badany skupia wzrok na poszczególnych bodźcach. Często może być trudna w interpretacji, ponieważ może być ciężko określić czy fiksacja wynika z zainteresowania danym obszarem czy użytkownik skupia na czymś wzrok przy podczas próby zrozumienia skomplikowanego zagadnienia pokazanego na ekranie.
- Liczba fiksacji wzroku [25] - liczba pojedynczych skupień wzroku. Im większa jest liczba fiksacji, tym więcej obszarów wzbudziło zainteresowanie użytkownika lub wręcz przeciwnie - ekran jest przeładowany ilością informacji i użytkownikowi ciężko znaleźć odpowiednią.
- Powtarzalność fiksacji wzroku [25] - liczba ponownych odwiedzin danego obszaru skupienia.
- Liczba sakad [25] - liczba sekwencji ruchu oka pomiędzy kolejnymi fiksacjami wzroku
- Długość ścieżki skanowania [25] - liczba następujących po sobie sakad i fiksacji.

W celu obliczenia metryki jakości interfejsu na podstawie listy kontrolnej LUT użyto punktów WUP (ang. *Web Usability Points*) [26].

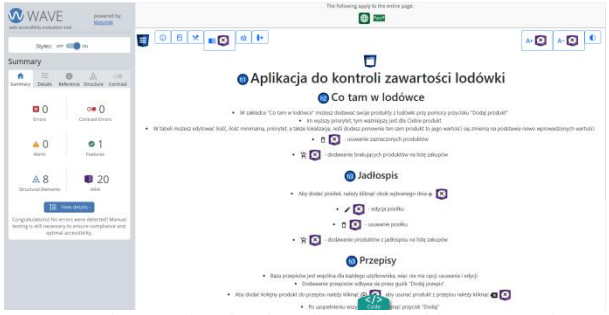
5. Wyniki badań

5.1. Wyniki badań narzędziem WAVE

Analiza narzędziem WAVE pokazuje, że obie wersje interfejsu zostały zaprojektowane poprawnie i przeszły test bez błędów krytycznych oraz ostrzeżeń. Pozytywny wynik testu klasyfikuje aplikację jako spełniająca wytyczne WCAG 2.0. Wyniki przedstawiono na Rysunkach 3 - 4.



Rysunek 3: Analiza tekstowej wersji interfejsu przy użyciu narzędzia WAVE.



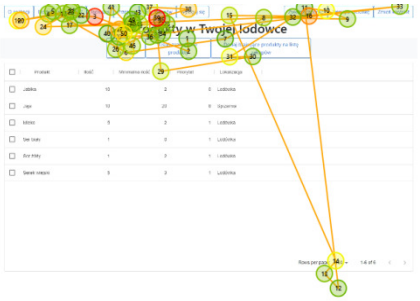
Rysunek 4: Analiza obrazkowej wersji interfejsu przy użyciu narzędzia WAVE.

5.2. Wyniki badań przy użyciu okulografu

5.2.1. Mapy cieplne i ścieżki patrzenia

Niniejszy rozdział przedstawia wyniki badań dwóch metod z podziałem na zadania. Zadania, dla których rozbieżności były znaczące, zostały uzupełnione o rysunki przedstawiające uzyskane wyniki.

Zadanie 1. Na wynikach widać, że badani mają pierwszy raz do czynienia z interfejsem aplikacji i dopiero ją poznają, dlatego ich ścieżki patrzenia są chaotyczne (przykładowa ścieżka skanowania na Rysunku 5). W tym zadaniu lepsze wyniki uzyskali badani przy pomocy metody B.

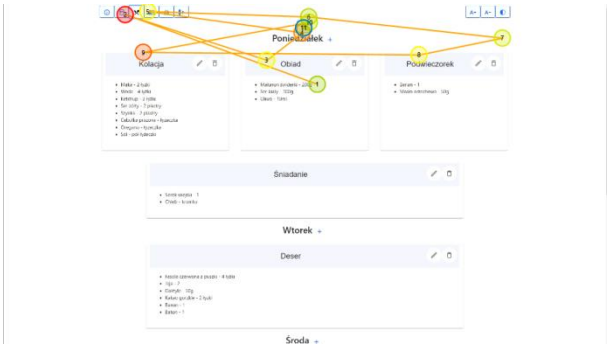


Rysunek 5: Przykładowa ścieżka skanowania dla zadania 1 dla metody B.

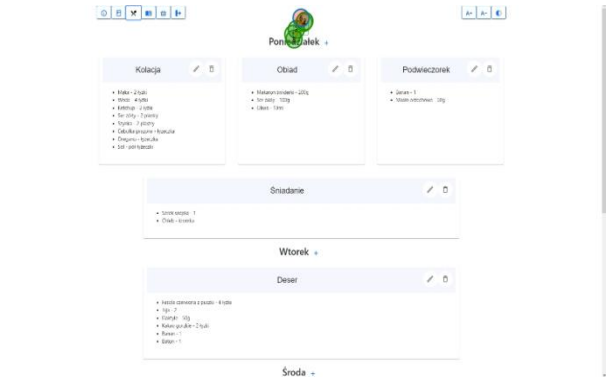
Zadanie 2. Wszyscy badani poprawnie wskazali element podany w zadaniu. Jednakże w przypadku metody B wyniki były lepsze, być może dlatego, że użyto bardzo popularnej ikonki używanej w aplikacjach właśnie do zmieniania kontrastu interfejsu. W przypadku metody A, badani poświęcali więcej czasu na czytanie etykiet innych przycisków, a także dwie osoby po znalezieniu poprawnego przycisku spojrzały na inne elementy co mogło świadczyć o tym, że nie są przekonane o swoim wyborze.

Zadanie 3. Ponownie było to pierwsze zetknięcie badanych z nową zakładką aplikacji, która, poza menu, znacznie różniła się od poprzedniego widoku, a także było w niej zawarte więcej elementów, więcej tekstu oraz były dwa sposoby poprawne sposoby na rozwiązanie tego zadania. To zadanie było dla badanych trudniejsze, zostało rozwiązane poprawnie tylko przez dwie osoby z metody A i trzy z metody B. Część badanych wskazywała przycisk służący do dodawania jadłospisu na listę zakupów jako rozwiązanie zadania (przykład na Rysunku 6).

Zadanie 4. Wszyscy badani metodą A odnaleźli przycisk, na co wskazywały fiksacje, jednakże po jego znalezieniu wracali do przeglądania strony, co wskazuje na to, że nie byli pewni jego poprawności. Tylko jedna osoba natychmiast odnalazła poprawny przycisk i po jego znalezieniu zakończyła zadanie (rozwiązanie na Rysunku 7). Natomiast w przypadku metody B dwie osoby prawdopodobnie źle zrozumiały polecenie, a jedna nie doczytała etykiety przycisku, bo po odnalezieniu go, kontynuowała poszukiwania (Rysunek 8). Ostatecznie tylko jedna osoba używająca metody A i dwie osoby korzystające z metody B poprawnie wskazały przycisk rozwiązujący zadanie.



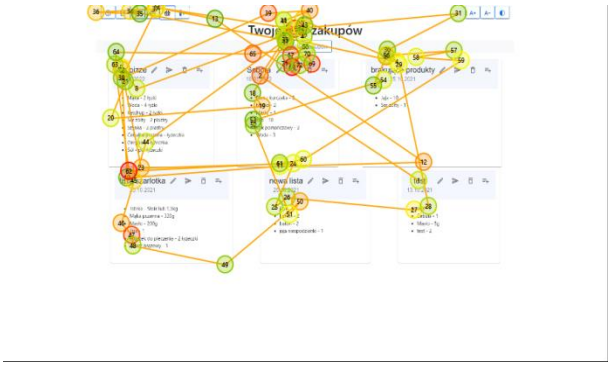
Rysunek 6: Niewłaściwie rozwiązane zadania 3, gdzie badany wskazał przycisk dodający jadłospis na listę zakupów.



Rysunek 7: Poprawnie rozwiązane zadanie 4.

Rysunek 8: Ścieżka skanowania, dla zadania 4, w której badany nie był pewien odnośnie znalezionego przycisku.

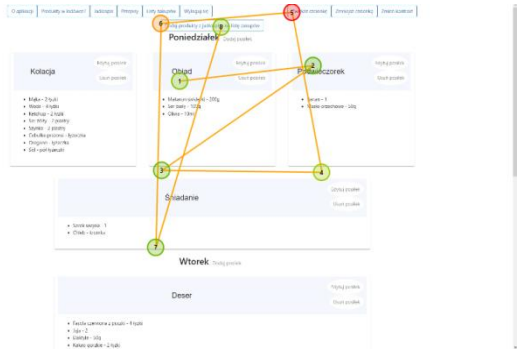
Zadanie 5. W tym zadaniu trudność mogło stanowić odnalezienie odpowiedniego elementu - w poleceniu sprecyzowano, aby użytkownik wykonał akcję na jednej spośród sześciu widocznych list. Zadanie w większości zostało wykonane poprawnie, tylko jedna osoba badana metodą A wskazał zły przycisk (Rysunek 9).



Rysunek 9: Nieprawidłowe wskazanie przycisku wysyłającego listę zakupów na adres email przez badanego metodą A.

Zadanie 6. Zadanie dotyczyło tego samego elementu co w poprzednim zadaniu, toteż badani nie mieli już problemu z odnalezieniem go. Jednak jedna osoba z używająca interfejsu metody A prawdopodobnie za szybko przeszła do kolejnego zadania, bo nie udało jej się wskazać poprawnego przycisku, natomiast w przypadku drugiej metody dwóch badanych wskazało przycisk do edycji listy, zamiast usunięcia - przyciski były bardzo blisko siebie i mogła to też być wina aparatury pomiarowej, która podczas badania się rozkalibrowała.

Zadanie 7. W obu grupach, trudność polegała na tym, że posiłki nie były ułożone w standardowej kolejności tylko alfabetycznej, więc badani najpierw patrzyli na pierwszy posiłek, gdzie śniadanie było akurat ostatnie. Jednej osobie badanej metodą A nie udało się odnaleźć odpowiedniego posiłku, natomiast przypadku metody B jedna osoba przeoczyła przycisk i wskazała na zawartość posiłku, a druga prawdopodobnie nie zrozumiała polecenia (Rysunek 10).



Rysunek 10: Ścieżka skanowania w której badany miał prawdopodobnie niewłaściwie zrozumiał treść zadania.

Zadanie 8. W tym zadaniu zdecydowanie lepiej poradziły sobie osoby badane metodą B, w której wszyscy poprawnie rozwiązali zadanie. W przypadku metody A trudność dla użytkowników stanowiło zrozumienie znaczenia ikony, co udało się tylko dwóm osobom. Pozostałe zwróciły uwagę na poprawny przycisk, ale nie były przekonane co do jego poprawności i ostatecznie wskazywały inne przyciski. Na zamieszczonych poniżej rysunkach widoczna jest rozbieżność w wyborze właściwego przycisku. Badana grupa w większość wskazywała ikonę po lewej, gdzie właściwą odpowiedzią byłą ta znajdująca się po prawej stronie ekranu. Mapy cieplne do tego zadania są przedstawione na Rysunkach 11 i 12.

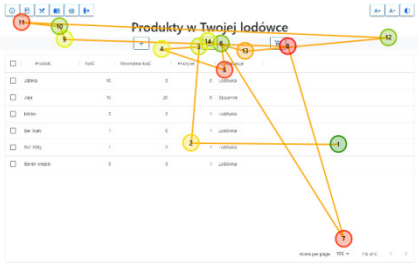
Rysunek 11: Mapa cieplna wykonania zadania 8 dla metody A.

Zadanie 9. W tym przypadku, użyto bardzo popularnej ikonki przycisku służącego do wylogowania użytkownika oraz umieszczono go w miejscu gdzie bardzo często znajduje się w aplikacjach internetowych (widoczne na mapie cieplnej na Rysunku 13). Poza osobą, u której wystąpił błąd w badaniu i nie ma żadnych informacji o jej wynikach, pozostałym badanym z obydwu grup udało się wykonać to zadanie poprawnie.

Rysunek 12: Mapa cieplna wykonania zadania 8 dla metody B.

Rysunek 13: Mapa cieplna pokazująca badanie metodą A w zadaniu 9.

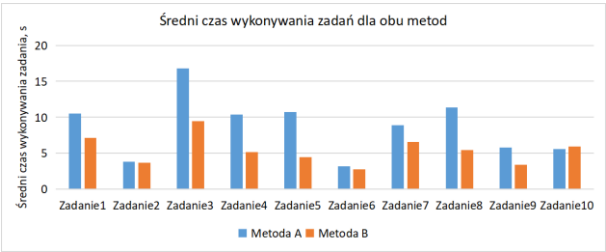
Zadanie 10. Podobnie jak w zadaniu 8. badani metodą A mieli problem ze zrozumieniem znaczenia ikony przycisku. Trzem osobom z tej grupy udało się wskazać poprawny przycisk, ale jedna z nich długo wahała się w wyborze. Jednej osobie nie udało się odnaleźć poprawnego przycisku (Rysunek 14.), a przy jednej ponownie wystąpił błąd i nie ma informacji o wynikach. W przypadku metody B dwóm osobom nie udało się odnaleźć przycisku.



Rysunek 14: Przykład błędnego rozwiązania zadania 10 metodą A.

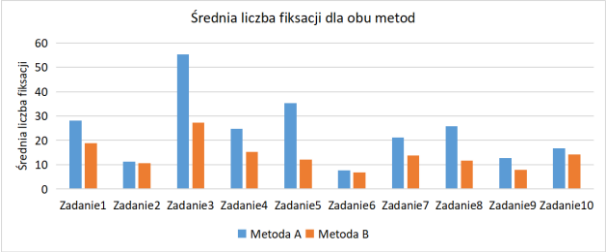
5.2.2.Wyniki pomiarów zarejestrowanych przez okulograf

Pierwszym czynnikiem uwzględnionym w analizie danych był czas wykonywania poszczególnych zadań przez osoby badane poszczególnymi metodami (przedstawiony na Rysunku 15). W przypadku metody A jest on zauważalnie dłuższy dla dziewięciu z dziesięciu zadań. W przypadku zadań numer 3, 4, 5 i 8 były one realizowane dwa razy dłużej. Jedyne zadanie które dla metody B zostało zrobione wolniej to zadanie nr. 10. Jednak w tym przypadku różnica jest niewielka i wynosi około 0.2 s.

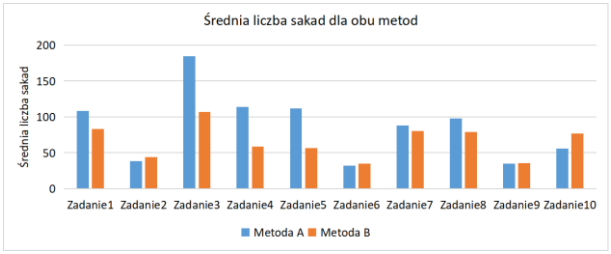


Rysunek 15: Średni czas wykonywania zadań dla obu metod.

Kolejnymi czynnikami, które zostały przeanalizowane były średnia liczba fiksacji (Rysunek 16) oraz średnia liczba sakad (Rysunek 17) dla zadania. W tym przypadku uzyskane wyniki pokrywają się z czasem wykonywania zadnia. Jednakże w przypadku liczby sakad występuje pewna rozbieżność. W zadaniach nr 2, 6 oraz 9 ich średnia liczba dla metody B jest większa lub taka sama jak dla metody A. Wynika to z charakterystyki ruchu oczu podczas czytania, gdzie często wykonywane są tzw. sakady regresywne (np. ruchy oczu w lewą stronę w tej samej linijce tekstu) [27]. Doprowadziło to do wykonania większej liczby ruchu oczu niż w przypadku interfejsu w wersji graficznej i pokazuje, że badani rzeczywiście skupiali się na czytaniu zawartości przycisków.

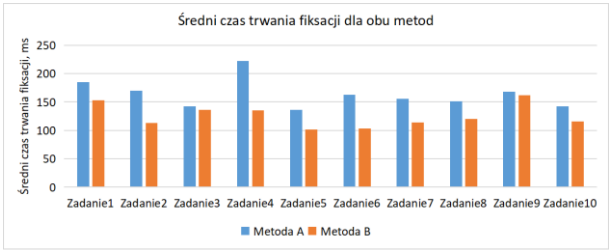


Rysunek 16: Średnia liczba fiksacji dla obu metod.

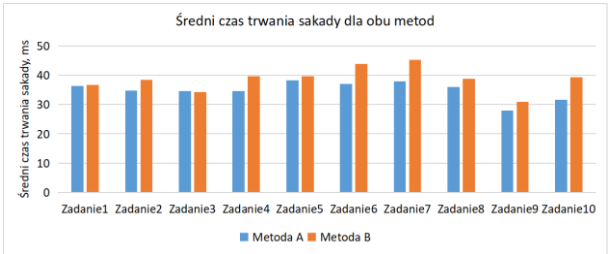


Rysunek 17: Średnia liczba sakad dla obu metod.

Kolejnymi czynnikami uwzględnionymi w badaniu były średni czas trwania fiksacji (Rysunek 18) oraz średni czas trwania sakady (Rysunek 19). W tym zestawieniu dokładnie widać, że dla wszystkich zadań dla metody A średni czas trwania fiksacji był większy. W przypadku metody B w każdym zadaniu przeważa czas trwania sakady. Wyniki te pokazują, że w przypadku grupy z interfejsami w wersji obrazkowej badani dłużej skupiali swój wzrok na konkretnych elementach a przesunięcia oczu były szybsze. W przypadku metody B badani musieli przeczytać tekst aby dowiedzieć się, co dany przycisk robi. W tej grupie średni czas trwania fiksacji jest niższy a czas trwania sakad jest większy. Różnice te związane są z procesem podążania wzroku za czytanym tekstem, Jest to szczególnie widoczne w przypadku, gdy tekst zawiera dłuższe słowa [27].



Rysunek 18: Średni czas trwania fiksacji dla obu metod.



Rysunek 19: Średni czas trwania sakady dla obu metod.

5.3. Ankieta LUT

Wyniki ankiety (Tabela 1) zawierają punktację dla trzech wyszczególnionych obszarów oraz współczynnik WUP. Wartość punktów WUP może się zmieniać od 1 do 5 – większa wartość oznacza lepiej zaprojektowany interfejs. W przypadku obszaru “Interfejs aplikacji” lepsze wyniki odnotowała wersja graficzna. Pytania z tego obszaru dotyczyły czytelności, spójności oraz doboru barw użytych w interfejsie. Większa różnicę, na korzyść wersji tekstowej, odnotowano w dwóch pozostałych obszarach. Wynika z tego, ze łatwiejszy w nawigacji oraz czytelniejszy był interfejs gdzie przyciski zawierały treści w formie tekstów. Można stwier-

dzić, że były one bardziej zrozumiałe dla badanych i zawierały więcej informacji.

Tabela 1: Wyniki ankiety LUT

Metoda	Średnie dla obszarów			WUP
	Nawigacja i struktura	Interfejs aplikacji	Treści podstron	
A	4,05	4,6	4,27	4,31
B	4,56	4,43	4,67	4,55

6. Wnioski

Według analizy przeprowadzonej przy pomocy narzędzia WAVE obydwa interfejsy spełniają wytyczne WCAG. W związku z tym można stwierdzić, że przygotowane interfejsy są dostępne.

Badanie przy pomocy okulografu pozwala stwierdzić, że metoda A osiągnęła około 60% poprawności rozwiązywanych zadań, natomiast metoda B - 74%. Można zauważyć, że w interfejsu w metodzie A dobór niektórych ikon dla badanych nie był oczywisty i stanowiło dla nich trudność zrozumienie ich przeznaczenia. Natomiast wyniki metody B wskazują na to, że przyciski w formie tekstowej zmuszają użytkownika do większego skupienia na elementach, przeczytaniu etykiet i dopiero podjęcia decyzji. Wpływa to na wolniejsze poruszanie się po stronie, jednak w badaniu zapewniło to większy odsetek poprawnie zrealizowanych zdań. W związku z powyższym nasuwa się wniosek, że niektóre ikony mogły być lepiej dobrane lub spersonalizowane, co wpłynęłoby pozytywnie na poprawność rozwiązywania zadań przez osoby badane metodą A.

Obszarem, nad którym mogą być prowadzone dalsze badania w tym temacie może być zapamiętywalność tego typu interfejsów. Czy znajomość układu strony i znaczenia poszczególnych przycisków będzie miała wpływ na szybkość wykonywania zadań.

Porównując wyniki listy kontrolnej LUT ponownie zauważalna jest przewaga interfejsu z przyciskami w formie tekstowej nad interfejsem z przyciskami w formie graficznej, w dwóch z trzech badanych obszarów otrzymał od ankietowanych więcej punktów. Interfejs z przyciskami w formie tekstowej uzyskał końcowy współczynnik WUP równy 4,55, co oznacza, że posiada drobne problem mogące obniżyć jakość pracy z aplikacją. Natomiast drugi interfejs otrzymał współczynnik WUP na poziomie 4,31, co stanowi, że w tym eksperymencie nie ma drastycznej przewagi jednej formy nad drugą.

Podsumowując i odpowiadając na postawione pytania badawcze, badania dowiodły, że obydwie wersje interfejsu są tak samo dostępne dla użytkowników. Interfejs z przyciskami w formie graficznej pozwolił badanym na szybsze wykonywanie zadań. Przyciski z etykietami w formie tekstowej, aby dobrze opisać funkcjonalność przycisku były dosyć długie przez co średni czas trwania fiksacji był nieznacznie dłuższy dla metody B. Subiektywna ocena interfejsu przez badanych wskazała interfejs z przyciskami w formie tekstowej jako lepszy, a na podstawie przeprowadzonych badań wiadomo, że osoby badane metodą B poprawniej wykonywała zadania, więc są to oceny współmierne.

Dodatkowe materiały

Ankieta zastosowania w badaniach jest dostępna pod linkiem: https://forms.gle/PNWPo3hSw1g19Ckc7

Literatura

[1] R. Antoszczak, Komunikatywność i użyteczność stron internetowych, Praca doktorska, Uniwersytet Warszawski, 2017.

[2] S. Harper, A. Q. Chen, Web accessibility guidelines: A lesson from the evolving Web, World Wide Web 15(1) (2012) 61-88.

[3] P. Brophy, J. Craven, Web Accessibility, Library Trends 55(4) (2007) 950-972.

[4] P. Acosta-Vargas, L. A. Salvador-Ullauri, S. Lulaj-Mora, A Heuristic Method to Evaluate Web Accessibility for Users With Low Vision, IEEE Access 7(1) (2019) 125634-125648.

[5] M. Bergel, A. Chadwick-Dias, L. LeDoux, T. Tullis, Web Accessibility for the Low Vision User, Usability Professionals Association (UPA), Montreal, 2005.

[6] M. F. Chiang, R. G. Cole, S. Gupta, G. E. Kaiser, J. B. Starren, Computer and World Wide Web accessibility by visually disabled patients: problems and solutions, Surv Ophthalmol 50(4) (2005) 394-405.

[7] G. Brajnik, Web Accessibility Testing: When the Method Is the Culprit, Proceedings of International Conference on Computers for Handicapped Persons 4061 (2006) 156–163, https://doi.org/10.1007/11788713_24.

[8] Wytyczne Web Content Accessibility Guidelines (WCAG 2.0), https://www.w3.org/TR/WCAG20/, [12.03.2022].

[9] H. Dudycz, Ł. Krawiec, Proporsal of the procedure of website usability evaluation, Informatyka Ekonomiczna, Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu 3(49) (2018) 65-77.

[10] R. Molich, J. Nielsen, Improving a human-computer dialogue, Com. of the ACM 33(3) (1990) 338-348.

[11] J. Allen, J. Chudley, Projektowanie witryn internetowych User eXperience, Smashing Magazine, 2013.

[12] P. Cybulski, T. Horbiński, User experience in using graphical user interfaces of web maps, ISPRS International Journal of Geo-Information, 9(7) (2020) 412.

[13] L. A. Granka, T. Joachims, G. Gay, Eye-tracking analysis of user behavior in WWW search, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (2004) 478-479.

[14] J. Sedkowska, How does the user experience of a progressive web application compare to native application? A case study on user's attitude in context of social media, Master thesis, Jönköping University, 2020.

[15] M. Jusiak, M. Miłosz, Analiza jakości interfejsu aplikacji internetowej z wykorzystaniem eye-trackingu–studium

przypadku. Journal of Computer Sciences Institute, 10 (2009) 62-66.

[16] M. D. Fleetwood, M. D. Byrne, Modeling the visual search of displays: a revised ACT-R model of icon search based on eye-tracking data, Human-Computer Interaction 21(2) (2006) 153-197.

[17] S. Cheng, A. K. Dey, I see, you design: user interface intelligent design system with eye tracking and interactive genetic algorithm, CCF Transactions on Pervasive Computing and Interaction 1(3) (2019) 224-236.

[18] Walidator W3, http://validator.w3.org, [12.03.2022].

[19] Projekt Utilitia, https://validator.utilitia.pl, [20.03.2022].

[20] Narzędzie Wave, http://wave.webaim.org, [20.03.2022].

[21] Narzędzie FAE, https://fae.disability.illinois.edu, [20.03.2022].

[22] Narzędzie achecker, http://achecker.ca, [22.03.2022]

[23] Narzędzie tawdis, http://www.tawdis.net [25.03.2022]

[24] M. Laskowski, Propozycje metodyk badania użyteczności interfejsów aplikacji, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, (4b) (2012) 21-24.

[25] R. J. Jacob, K. S. Karn, Eye tracking in human-computer interaction and usability research: Ready to deliver the promises, In The mind's eye, North-Holland, 2003.

[26] M. Miłosz, M. Borys, M. Laskowski, Memorability Experiment vs. Expert Method in Websites Usability Evaluation, In International Conference on Enterprise Information Systems, 2 (2013) 151-157.

[27] K. Hryniuk, Okulograficzne wsparcie badań nad procesem czytania, Lingwistyka Stosowana 4 (2011) 191-198.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 297–300
Received: 15 June 2022
Accepted: 4 July 2022

Comparison of EXT4 and NTFS filesystem performance

Porównanie wydajności systemu plików EXT4 i NTFS

Bartosz Piotr Sterniczuk

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

Abstract

The aim of this article is to compare the two most popular and competing file systems – EXT4 and NTFS in Ubuntu with the use of SSD disk. At the beginning of this article, a critical description of the literature was made, explaining the purposefulness of the research undertaken. Additionally, the basics of the operation of both discussed file systems are explained. The research consisted of copying files between two partitions and measuring the time of this operation using a specially developed system shell script in the bash language. The conducted research has shown that the EXT4 system is more effective than the NTFS system.

Keywords: EXT4; NTFS; performance file system; file system

Streszczenie

Celem artykułu jest porównanie dwóch najbardziej popularnych i konkurencyjnych systemów plików – EXT4 oraz NTFS w systemie Ubuntu przy wykorzystaniu dysku SSD. Na początku artykułu został wykonany krytyczny opis literatury wyjaśniający celowość podjętych badań. Dodatkowo wyjaśniono podstawy działania obu omawianych systemów plików. Badanie polegało na kopiowaniu plików pomiędzy dwoma partycjami i mierzeniem czasu wykonywania tej czynności przy użyciu specjalnie opracowanego skryptu powłoki systemu w języku bash. Przeprowadzone badania wykazały lepszą efektywność systemu EXT4 nad system NTFS.

Słowa kluczowe: EXT4; NTFS; porównanie system plików; system plików
Email address: **bartek114@autograf.pl**

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Systemy plików odgrywają znaczącą rolę w działaniu systemu operacyjnych, są one jednym z podstawowych składników mającym wpływ na jego działanie. Na przestrzeni lat powstawało ich bardzo dużo. Jednym z nich jest MINIX, na podstawie którego został zbudowany EXT (rozszerzony system plików) ewoluujący do wersji EXT4 (ang. Fourth Extended File System), który jest domyślnym systemem plików dla systemów operacyjnych typu Linux. Jego konkurencją jest NTFS (ang. New Technology File System), który został wyprodukowaną przez firmę Microsoft i jest on domyślnie używany w systemie z rodziny Windows.

Obecnie w literaturze możemy znaleźć wiele artykułów dotyczących badania różnic pomiędzy poszczególnymi systemami plików. Jednym z nich jest „Porównanie możliwości i cech współczesnych Linuxowych systemów plików: ext4, XFS, Btrfs" [1]. W przedstawionym artykule, autorzy porównują systemy plików należące do systemu z rodziny Linux. Wynikiem badań było znalezienie najbardziej wydajnego, którym według autorów został BTRFS. Należy zwrócić uwagę na datę powstania artykułu – 2017r. oraz na podzespoły których użyto do wykonania eksperymentów. W czasie wykonywania badania użyto dysku HHD, który był popularny w tamtym okresie. Z uwagi na obecny postęp technologii, zostały one wyparte przez dyski SSD (ang. Solid State Drive), które nie zawierają części ruchomych, dzięki czemu znacznie przyśpieszają proces zapisu i odczytu danych.

Autorzy artykułu „Comparing NTFS File System with ETX4 File System" [2] porównali dwa systemy plików NTFS i EXT4 również przy wykorzystaniu dysku HDD. Badania dotyczące szybkości oparli na testowaniu wydajności w odrębnych środowiskach operacyjnych. Podczas analizy uwzględnili szybkość zapisu i odczytu danych NTFS w systemie Windows 7, natomiast EXT4 w systemie Linux.

Celem niniejszej pracy jest porównanie dwóch systemów plików NTFS i EXT4 przy wykorzystaniu nowoczesnego dysku SSD w systemie operacyjnym Ubuntu. Postawiono hipotezę badawczą: ***System plików EXT4 jest bardziej wydajny niż system plików NTFS.***

2. System plików

System plików jest sposobem organizacji i przechowywania danych. Dzięki niemu system operacyjny jest w stanie szybko znaleźć położenie pliku, odczytać jego zawartość, a także łatwo określić ilość dostępnego miejsca na danej partycji/dysku. Dane zazwyczaj są przechowywane w postaci bloków, które po połączeniu tworzą klastry. Jako przykład można podać analogię do porządkowania zbiorów danych np. w bibliotekach. Książki posiadają swoje karty które wskazują gdzie faktycznie się one znajdują. Podobnie działają niektóre systemy plików wykorzystując część miejsca do opisu metadanych natomiast resztę przeznaczają na przechowywanie surowych danych. Należy również wspomnieć o jednej z ważniejszych cech, a mianowicie o przechowywaniu uprawnień do zasobów. Dzięki temu jesteśmy w stanie nakładać pewne ograniczenia na dane, co stanowi ich dodatkową ochronę. Do najważniejszych elementów składowych systemu plików należą:

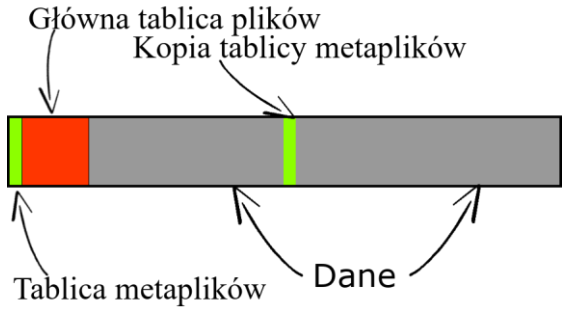
- bezpieczeństwo – mechanizm pozwalający na dostęp do danych wybranym użytkownikom,
- API – funkcje systemowe pozwalające na podstawowe operacje na plikach,
- przestrzeń nazw –sposób nazewnictwa oraz organizacja plików,
- implementacja – wiązanie modelu logicznego z modelem fizycznym [3].

3. System plików NTFS

NTFS (ang. New Technology File System) jest domyślnym system plików używanym w systemach z rodziny Windows. Został on zaprojektowany w celu zastąpienia systemu FAT, którego głównym ograniczeniem był maksymalny rozmiar plików – 4GB. NTFS nie ma odgórnego limitu pojedynczego rozmiaru pliku. Jedynym ograniczeniem jest całkowita pojemność wolumenu która wynosi: 2^{64}B, co w obecnych czasach jest ciężkie do osiągnięcia przez aktualnie produkowane dyski twarde. Jego implementacja składa się z czterech głównych elementów:

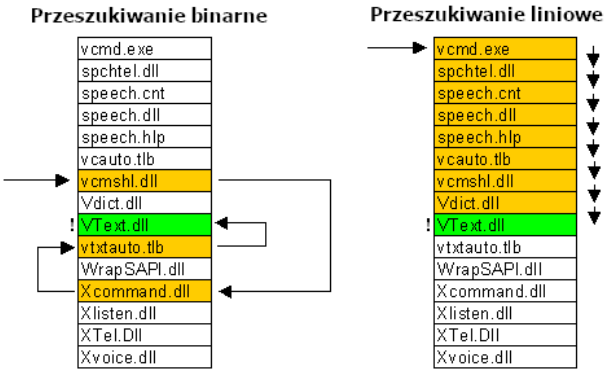
- sektor rozruchowy (ang. volume boot sector),
- blok metaplików (ang. metada files),
- główna tablica plików(ang. master file table),
- dane.

Sektor rozruchowy jest pierwszym blokiem występującym w utworzonej partycji. Składa się z dwóch elementów – BIOS Parameter Block oraz Volume Boot Code. Pierwszy z nich jest odpowiedzialny za przetrzymywanie informacji o partycji oraz jej rozmiarach, drugi natomiast przetrzymuje informacje potrzebne do załadowania systemu operacyjnego. W elemencie blok metaplików znajdują się metapliki. Są one oznaczone przy pomocy „$”, dostęp do nich jest utrudniony. Ten obszar zawiera dokładnie 16 elementów i jest odpowiedzialny za wykonywanie pewnych operacji dyskowych. W celu zapewnienia poprawności działania w razie uszkodzenia, jego kopia znajduję się zazwyczaj po środku dysku. Na Rysunku 1 zostały przedstawione omawiane elementy.



Rysunek 1: Podział partycji NTFS.

Główna tablica plików jest to plik podzielony na rekordy o stałym rozmiarze, zazwyczaj około 1kB. Jego głównym celem jest przechowywanie informacji o pliku, takich jak: nazwa, rozmiar, parametry dostępu a także wskaźnik do fizycznego położenia pliku. Należy podkreślić, iż katalogi są również przetrzymywane w omawianym elemencie. Są one podzielone na bloki, które zawierają referencję do pliku oraz jego podstawowe elementy. Ich implementacja jest wykonana przy wykorzystywaniu drzewa binarnego, dzięki czemu można zastosować metodę bisekcji do wyszukiwania plików znajdujących się w poszczególnych folderach. Odszukanie pliku zajmuje znaczniej mniej czasu niż w przypadku innych organizacji plików gdzie wyszukiwanie jest zaprogramowane liniowo. Omawiane przeszukiwanie zostało przedstawione na Rysunku 2. Należy dodać, iż system posiada dziennik zdarzeń, dzięki któremu jesteśmy w stanie kontrolować przerwane operacje. W przypadku wystąpienia awarii następuje przeglądanie dziennika, uszkodzona operacja jest odnajdywana, po czym następuje próba jej naprawienia. W przypadku braku zapisu logów, użytkownik jest obciążony długotrwałym procesem skanowania dysku [4].



Rysunek 2: Porównanie przeszukiwania binarnego z linowym [5].

4. System plików EXT4

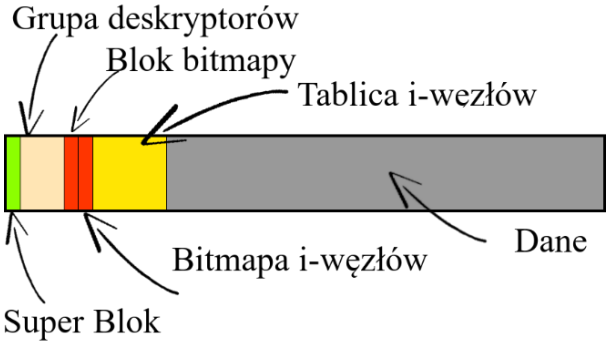
EXT4 (ang. Extended Filesystem) jest domyślnym systemem plików dla systemów z rodziny Linux. Stał się on następcą wcześniejszych wersji, które wywodzą się pierwotnie od systemu MINIX. Ponieważ bloki są numerowane przy pomocy 32 bitów, a każdy z nich jest 4 bitowy, to maksymalny rozmiar pliku może wynosić 16TB (2^{32}*4). Maksymalna wielkość partycji, może wynosić 1EB i jest ona ograniczona przez 48 bitowy znacznik bloku. Podobnie jak w przypadku NTFS, omawiany system składa się z następujących elementów:

- super blok,
- grupa deskryptorów,
- blok bitmapy,
- bitmapa i-węzłów,
- tablica i-węzłów,
- blok danych.

Na Rysunku 2 został przedstawiony schemat rozmieszczenia wymienionych elementów. Super blok jest elementem przechowującym informację o systemie plików, liczbie bloków i liczebności węzłów. Jego zdaniem jest połączenie całej struktury bloków w jedną całość. W blokach bitmapy i i-węzłów są przechowywane informacje na temat zajętości bloków, które są w użyciu.

Tablica i-węzłów, jest odpowiednikiem bloku MFT występującym w systemie NTFS. Jej głównym zadaniem jest przetrzymywaniem metadanych plików takich jak: typ pliku, identyfikator użytkownika, grupy oraz

czasów modyfikacji, utworzenia, itp. Najważniejszym elementem tablicy, jest przechowywanie wskaźników do numerów bloków, w których znajdują się dane. W przypadku dużego rozmiaru pliku, autorzy zaimplementowali możliwość używania wskaźników pojedynczych i potrójnych, które są nazwane wskaźnikami pośrednimi. Omawiany mechanizm polega na wskazaniu adresu następnego wskaźnika, który zawiera w sobie wskaźniki umożliwiające dostęp do kolejnych bloków danych. Dzięki takiemu mechanizmowi jesteśmy w stanie zaalokować większą ilość bloków, co związane jest z większym rozmiarem pliku [6].



Rysunek 2: Podział partycji Ext4.

5. Metodyka badawcza

W celu przeanalizowania wydajności obu omawianych systemów plików, wykonywano kopiowanie plików pomiędzy dwoma partycjami. Badania zostały wykonane na laptopie Lenovo IdeaPad 330-15ARR (81D200DFPB), który składa się z parametrów przedstawionych w Tabeli 1.

Tabela 1: Podzespoły zamontowane w urządzeniu, na którym zostało wykonane badanie

Podzespół	Opis
Procesor	AMD Ryzen 5 2500U 2,0GHz
Pamięć RAM	8GB, DDR4, 2133MHz
Dysk SSD	ATA Micron MTFD-DAK256TBN **Pojemność**: 250GB **Szybkość odczytu**: 530 MB/s, **Szybkość zapisu:** 500 MB/s
System operacyjny	Ubuntu 20.04 LTS

Każda z utworzonych partycji była sformatowana przy użyciu innego systemu plików – NTFS lub EXT4, dzięki czemu można było mierzyć czasy kopiowania. Do tego celu został napisany skrypt przy użyciu języka programowania powłoki bash, przedstawiony na Listingu 1. Po jego uruchomieniu, użytkownik otrzymuje dwa pytania, pierwsze z zapytaniem dotyczącym ilości kopiowanych plików, drugie dotyczące ich rozmiaru. W celu prawidłowego działania skryptu należy utworzyć dwa foldery: source oraz destination w katalogu domowym użytkownika, a później zamontować do nich wcześniej utworzone i odpowiedni sformatowane partycje. Skrypt automatycznie utworzy zadaną ilość plików, wykona kopiowania z pomiarem czasu, natomiast na samym końcu zajmie się usuwaniem wcześniej utworzonych plików. Po wykonaniu zadania zostaną wyświetlone wartości trzech czasów:

- real – rzeczywisty czas działania całego procesu,
- user – określa czas poświęcony przez procesor na wykonanie żądanego procesu,
- system – upływ czasu przeznaczony na komunikację z jądrem systemu operacyjnego.

Do celów badania pod uwagę wzięto czas user oraz sys. Jako wynik końcowy podana została suma wyżej wymienionych czasów.

Listing 1: Skrypt języka bash wykonujący kopiowanie plików, wraz z pomiarem czasu

```
#!/bin/bash
echo "How much file do you want to create?"
read amount
echo "How much file size do you want to create? (data in MB)"
read size
source=/home/ubuntu/source/directory
mkdir $source
destination=/home/ubuntu/destination/directory
for i in $(seq 1 $amount); do
 dd if=/dev/zero of=$source/file.data_$i bs=1M &>/dev/null count=$size
done
time cp -rf $source $destination
rm -rf $source
rm -rf $destination
```

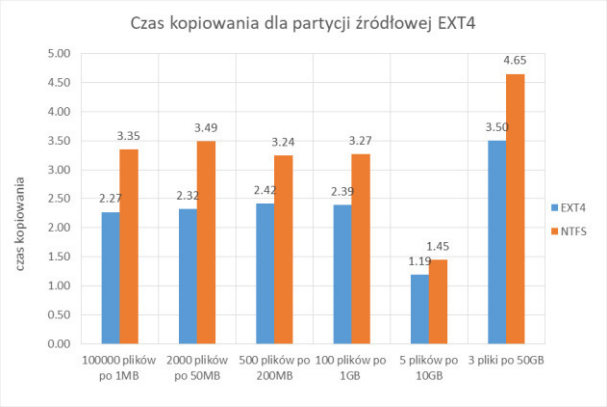
Badanie zostało wykonane kilkukrotnie dla różnych konfiguracji. Zmianie ulegał rozmiar i ilości kopiowanych plików. Dokładne dane poszczególnych przypadków badania były następujące:

1. 100000 plików po 1 MB każdy,
2. 2000 plików po 50MB każdy,
3. 500 plików po 200 MB każdy,
4. 100 plików po 1 GB każdy,
5. 5 plików po 10 GB każdy,
6. 3 pliki po 50 GB każdy.

Każdy typ badania został wykonany dwukrotnie. Jako wynik przyjęto średnią z tych dwóch prób. W celu wyeliminowania przekłamań związanych z kopiowaniem pomiędzy partycjami tego samego sytemu plików wykonano dwa warianty badania. W pierwszym wariancie partycją źródłową był NTFS, natomiast drugim EXT4. W celu umożliwienia wykonywania operacji na partycji w formacie NTFS w systemie Ubuntu użyto domyślnego sterownika ntfs-3g.

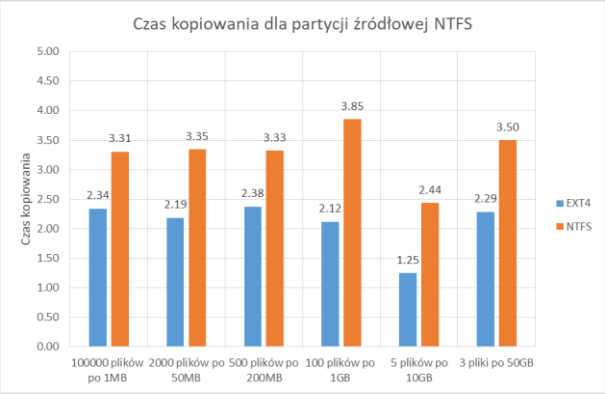
6. Wyniki badań

Na Rysunku 3 zostały przedstawione czasy kopiowania plików z partycji źródłowej EXT4 na oddzielne partycje – NTFS oraz EXT4. Dla każdej z sześciu prób system plików EXT4 okazał się lepszy, uzyskując szybszy czas kopiowania. Najmniejsza różnica czasu była zauważalna dla piątej próby, przy kopiowaniu 5 plików po 10GB.



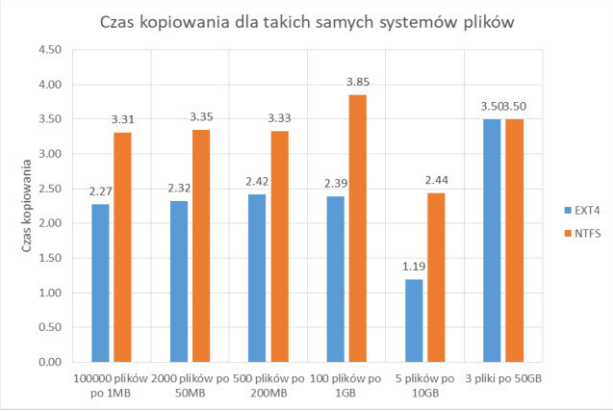
Rysunek 3: Czas kopiowania plików z partycji źródłowej EXT4.

Rysunek 4 przedstawia czasy kopiowania plików z partycji źródłowej NTFS na partycje z systemami plików: NTFS oraz EXT4. Tu również szybszy okazał się system EXT4. Największą różnicę udało się uzyskać podczas wykonywania czwartej próby, w przypadku 100 plików po 1GB każdy.



Rysunek 4: Czas kopiowania plików z partycji źródłowej NTFS.

W celu lepszego zobrazowania i uniknięcia przekłamań związanych z przenoszeniem danych pomiędzy partycjami o różnych systemach plików, na Rysunku 5 został pokazany wykres, w którym zestawiono przypadki kopiowania plików na takie same systemy plików. Na postawie przedstawionego wykresu można zobaczyć, iż również w tym przypadku zwycięzcą został system EXT4.



Rysunek 5: Zestawienie czasów kopiowania dla przenoszenia danych pomiędzy takimi samymi systemami plików.

Jedynym przypadkiem posiadającym taki sam czas kopiowania jest przypadek szósty, w którym wykonywano kopiowanie 3 plików po 50GB każdy. Pomimo tego wyjątku wszystkie pozostałe badania wykazują znaczącą przewagę dla EXT4. Największa przewaga została uzyskana podczas wykonywania badania kopiowania 100 plików po 1 GB każdy.

7. Podsumowanie

Na podstawie przeprowadzonych badań stwierdzamy, że system plików EXT4 wykazał się większą szybkością kopiowanych plików. Nawet pomimo kopiowania na inny system plików (Rysunek 4) osiągnął lepszy czas, niż konkurencyjny system firmy Microsoft. Przeprowadzone badania dowodzą postawionej we Wstępie tezy. Należy również podkreślić, iż celem badania było przeprowadzanie doświadczeń w jednym systemie operacyjnym - Ubuntu, zatem trzeba być świadomym, iż system NTFS mógłby uzyskać lepsze wyniki podczas wykonywania testów w systemie z rodziny Windows. Różny schemat uprawnień, dostępów, a także sposób przechowywania danych, utrudnia działanie w przypadku zmiany systemu operacyjnego. Zatem nie należy pochopnie zmieniać systemu plików, tylko ze względu na lepsze wyniki prędkości.

Porównując otrzymane rezultaty do badań przedstawionych we Wstępie [1,2], można stwierdzić, iż wykonanie ich na dysku SSD znacząco zmniejszyło czasy kopiowania. Dodatkowo należy zwrócić uwagę, iż system EXT4 jest szybszy zarówno podczas korzystania z dysku SSD jak i HDD. Dzięki takiemu porównaniu możemy powiedzieć, iż implementacja zwycięskiego systemu jest wykonana lepiej, gdyż osiąga lepsze wyniki niezależnie od platformy testowej. Bezpłatny dostęp do implementacji EXT4, a także możliwość nieograniczonej edycji przez społeczność użytkowników spowodowały, iż rozwiązanie w postaci darmowego systemu plików może konkurować z komercyjnymi rozwiązaniami.

Literatura

[1] B. Kossak, M. Pańczyk, Porównanie możliwości i cech współczesnych Linuxowych systemów plików: ext4, XFS, Btrfs, Journal of Computer Sciences Institute 4 (2017) 131-136.

[2] V. Dhjaku, N. Xoxa, A. Bame, I. Tafa, Comparing NTFS File System with ETX4 File System, In RTA-CSIT (2018) 176-180.

[3] E. Nemeth, G. Snyder, T. R. Hein, B. Whaley, D. Mackin, UNIX and Linux System Administration Handbook (5th Edition), Pearson Education, 2018.

[4] Dokumentacja NTFS napisana przez Richarda Russona i Yuvala Fledela, https://dubeyko.com/development/FileSystems/NTFS/ntfsdoc.pdf, [11.06.20222].

[5] Strona internetowa poświęcona wykładom z przedmiotu Systemy Operacyjne, https://students.mimuw.edu.pl/SO/Projekt02-03/temat2-g1/ntfs.htm, [11.06.2022].

[6] Strona internetowa poświęcona systemowi Ext4, https://ext4.wiki.kernel.org/index.php/Main_Page, [11.06.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 301–308
Received: 15 June 2022
Accepted: 4 July 2022

Usability analysis of the user interface of movie-related websites in terms of universal design

Analiza użyteczności interfejsów stron internetowych o tematyce filmowej pod kątem projektowania uniwersalnego

Jakub Sokół*, Karol Bielec, Maria Skublewska-Paszkowska

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

Abstract
The aim of the paper is the usability analysis of the user interface of movie-related websites in terms of universal design. The analysis was performed in order to verify the influence of implementing the universal design concepts on services usability. The studies were based on two web services: existing Filmweb and its auctorial equivalent. For the purpose of analysis the following research methods were used: questionnaires, experiment using eye-tracking technique and validator tests. Moreover, there was compliance with World Wide Web Consortium (W3C) and Web Content Accessibility Guidelines (WCAG) standards checked for both services. Ten participants aged 23-25 took part in the study. Results analysis have shown that deploying universal design conceptions significantly increases usability of the interface.

Keywords: usability; eye-tracking; user interface; universal design

Streszczenie
Celem artykułu jest analiza użyteczności interfejsów stron internetowych o tematyce filmowej pod kątem projektowania uniwersalnego. Analiza dotyczyła weryfikacji wpływu zastosowania założeń projektowania uniwersalnego na użyteczność serwisów. Badania wykonano bazując na dwóch serwisach internetowych: istniejącym Filmweb oraz jego autorskim odpowiedniku. Podczas analizy użyto następujących metod badawczych: ankiety, eksperymentu wykorzystującego technikę eyetrackingową oraz testów przeprowadzonych walidatorami. Dodatkowo, dla obu serwisów sprawdzono zgodność ze standardami World Wide Web Consortium (W3C) i Web Content Accessibility Guidelines (WCAG). W badaniach wzięło udział dziesięciu uczestników w wieku od 23 do 25 lat. Analiza wyników wskazała, iż wdrożenie koncepcji projektowania uniwersalnego znacząco poprawia użyteczność interfejsu.

Słowa kluczowe: użyteczność; okulografia; interfejs użytkownika; projektowanie uniwersalne

*Corresponding author
Email address: jakub.sokol@pollub.edu.pl (J. Sokół)

©Published under Creative Common License (CC BY-SA v4.0)

1. Introduction

With the development of the film industry, companies responsible for accessing content on such topics create IT solutions to open up to new recipients. As a result of these actions, modern implementations of functionalities focus on aesthetic aspects, while pushing usability into the background. A potential user decides to choose between websites with similar content or functionalities – as they concern the same topic – but with a distinctive graphic design. Nowadays, concepts such as availability are often ignored in interface design. In this paper, an attempt was made to emphasise the influence of complying with terms of universal design on interface intuitiveness. Those terms consist of e.g.: simple and intuitive use, perceptible information. Considering mentioned concepts, authors created service with consistent graphic design, assuring high content readability.

Usability is considered to be resultant of factors such as: learnability, efficiency, memorability, errors and satisfaction. It's a characteristic of software quality from perspective of user [1]. Therefore, usability focuses on user experience with emphasis on the indicators such as knowledge of using the Internet or sight defects. In order to conduct this kind of studies, there was a variety of researches performed, including: eyetracking experiment, questionnaires, and testing by the use of online validators.

2. Literature review

2.1. Objective analysis research

In the article "A measurable website usability model: Case Study University of Jordan" [2], the usability of websites was examined on the example of the website of the University of Jordan. The authors proposed a hierarchical model of website usability assessment consisting of 9 main categories and 24 measurable criteria. There was expected the most unified usability measuring model by integrating numerous methods tested by various media, such as a smartphone, a tablet or a desktop browser. The criteria proposed by the authors included: headings (check if they summarise the content of page), mobile (adaptation to devices with smaller screens), page names, internal links, compatibility with operating systems and browsers, universal language (a native version of the website), server response time, server behaviour (error handling) or code quality

(meeting the requirements of W3C). The criteria were repeated in 9 categories, therefore they have been given weights. The free tools: Nibbler, Gtmetrix, Checkmy-colours and PowerMapper were used for the study, and the usability of the website of the University of Jordan was verified. Each experiment was repeated 6 times, where 5 different subpages were checked with the use of the above-mentioned tools. For each criterion, points ranging from 0-10 were assigned. The obtained results were inserted into the equation with weights depending on the number of occurrences of the criteria in order to obtain the percentage response. The result of the research was, on the one hand, the indication of numerous errors in the website code, and on the other hand, positive results for the universality of the language, compatibility, refreshing, links to the home page, page names or versions for mobile devices.

The aim of the study in the article "Accessibility Testing of European Health-Related Websites" [3] was to research websites related to health care in 9 European countries to assess their accessibility with an emphasis on users with disabilities. The research was carried out thanks to the automation of software testing and statistical analysis of users' feelings. 48 pages from Eastern Europe were compared with 51 pages from Western and Northern Europe. The research was divided into 3 phases: testing with AChecker, then with Nibbler, and finally filling in questionnaire by users. The main assumption was to find aspects such as the ability to read CAPTCHA tests or keyboard navigation that negatively affected the accessibility of websites for users with disabilities. No correlation was found in the results between the economic situation of the countries where the pages came from and their sizes in kilobytes, the number of barriers or their accessibility results measured with the Nibbler tool. The different percentage of age of citizens also did not affect the differences in the approach to page design between the eastern, western and northern parts of Europe.

The aim of the research in the article "Enhanced Colour Scheme Assessment Tool (COSAT 2.0) for Improving Webpage Colour Selection" [4] was to verify the impact of colour selection on the website on its visual usability. The authors used the COSAT 2.0 tool that defined the principles of colour selection together with a guide in terms of increasing usability. The COSAT influenced aspects such as colour combinations, their contrast and brightness. The website was researched and it was improved on an ongoing basis along with the tips obtained from user testing and the COSAT, WCAG and W3C tools. The research results clearly indicated that the pages created by developers using this tool obtained high usability ratings, which proved the significant impact of COSAT rules on the feeling of using a web application.

The authors of the paper "User Experience Analysis of binus.ac.id Website with The Usability Testing Perspective (A Case Study Approach)" [5] researched the BINUS University website to see how transparent it was to potential users such as prospective students or their parents. Two most important aspects of the study were: quality assurance and page loading speed. The main goal was to find missing elements that had not been initiated since the implementation of the website. The university website was tested with the Web Accessibility Evaluation tool (WAVE) to list the problems affecting the website accessibility and with the Catchpoint software for connection analysis and page loading speed. The results were to serve as an indication of what aspects were omitted and which had a significant impact on the performance and availability of the website. These aspects were: page header and footer contrast, lack of labels, especially for the page logo.

The authors of the article "Accessibility of environmental data for sharing: The role of UX in large cyberinfrastructure projects" [6] examined which of the user experience (UX) tests were implemented by the DataONE research service over the decade. Between years 2009 and 2019, employees created tools for managing the company's data, with one of the groups working on the new solution for usability and diagnosis. They developed 44 tests to verify the usability of their products and websites at different stages of the development process. The research was divided into 4 stages: design, development, implementation and maintenance. The process involved from 5 up to 14 users depending on the test type. Such solutions were applied as polls, interviews, usability tests in the form of finding an element on the page without and with hints, discussion in a focus group, research in terms of fulfilling heuristics (mainly Nielsen) or testing with an eye-tracker. The authors emphasized the significant impact of testing cyberinfrastructure design. The solutions they implemented improved subsequent versions of their websites with a simultaneous increase in user satisfaction and the usability of large and complex projects such as websites.

2.2. Non-objective analysis research

In the article "The Effects of the Floating Action Button on Quality of Experience" [7], the impact of the Floating Action Button (FAB) on the quality of receiving the page was examined from the user's point of view. It was highlighted that several user interface (UI) and user experience design specialists were sceptical about the implementation of this solution in both web and mobile application interfaces. They argued that the FAB was a distraction, interfering with other functionalities, as well as unable to be used in applications designed for iOS. The purpose of this article was to verify the FAB for a quality user experience. Both static and animated buttons were tested and it compared to another toolbar. The research was conducted on advanced users with the use of Nielsen heuristics to assess the usefulness of the FAB in the application. Users had to perform specific tasks using the FAB, and then a toolbar, during which the execution time, the fact that the task was completed, the number of errors and the degree of difficulty of the task were counted. As a result of the research, it turned out that the use of FAB did not have a positive effect on

the usability of the application, but it improved the aesthetic aspect.

In the article "Web Navigation and Usability Analysis of Educational Websites in Pakistan" [8], the trunk test was used to verify the quality of navigation on the website and to test the comfort of use and usability with over 100 websites of universities in Pakistan. The trunk test was carried out on the basis of 6 open questions regarding the placement of elements on the page, such as: where to search for the content, what were the users options at this stage in the application or what subpage the user was on and what were the main sections of the page. The research was carried out by the authors of the articles and the results were also compared between their achievements. As a result of the study, a conclusion was drawn that it was necessary to introduce amendments to the navigation methods used in Pakistani websites. About 43% of Pakistani educational sites had problems including navigating and finding desired content due to the lack of indicators for the current path and insufficient search filters.

The aim of the research in the article "Redevelopment of the Predict: Breast Cancer website and recommendations for developing interfaces to support decision-making" [9] was to create an interface for a breast cancer prognosis tool. The website was designed to facilitate postoperative treatment as well as visualize the disease prognosis as much as possible for patients and their doctors. The research was conducted with the use of questionnaires with a five-point scale of assessments, exchange of opinions in an interview as well as usability tests based on interviews with patients and doctors. Problems were detected with the recognition or finding of a clickable button. The design was based on the prioritizing user needs. As a result, an interface more accessible to a wider audience was obtained thanks to new forms of visualization, continuous data updates, advanced content aimed at clarifying issues or implementing solutions that predict, for example, the course of the disease. The main conclusion was that the participation of a potential user in the creation process, as well as basing the design architecture with the end user in mind, gave the opportunity to benefit from the aspects of life intended for them and concerning them – with an emphasis on predicting the future.

The aim of the article "Usability evaluation of a library website with different end user groups" [10] was to verify how diverse the end users of the library website were. The research was carried out in terms of usability, the components of which were: effectiveness, efficiency and satisfaction. Usability was tested through the thought-to-voice protocol, journal analysis and questionnaires. Efficiency was measured as the time in seconds required to complete a set task. Efficiency was assessed on a five-point scale using a task-based methodology. User satisfaction was measured using a questionnaire using a five-point scale. As a result of the static analysis, the conclusion was obtained that different groups of end users achieved different degrees of effectiveness and efficiency, while maintaining no differences between the level of satisfaction in groups. Participants did not reach the “useful” threshold. In view of the detected shortcomings, the authors proposed solutions to improve the usability of the website.

With regard to literature review authors have formulated the following hypothesis: “deployment of universal design concepts significantly improves interface usability”.

3. Research methods

3.1. Research objects

The first research object was the online movie-related service Filmweb.pl. The website offers some of the functionalities for users who are not logged in and extended actions for those who has acquired an account. Service client can browse information about movies, movie crew, posters, trailers and rankings. In case when user is logged in, he or she can add a content in form of reviews and ratings or mark some of the content to get notifications. The service is used by a large community, thanks to comprehensive contents of database about cinema movies, television movies and series, as well as editors section with news from the world of motion pictures. The second service was created by the authors of this paper. Both websites were similar in case of functionalities but with different implementation of user interface. Authors’ interface was implemented with use of components assuring accessibility and readability of content. There was intuitive layout created in contrary to Filmwebs vertical elements arrangement to decrease time that user spends on subpage to seek for information.

In order to create the service similar to Filmweb, the authors used React – a JavaScript based library with MUI components pack [11]. React is based on components managing their own state that allows developers to compose complex user interfaces [12]. The project was developed with use of the Node Package Manager (NPM) [13]. It allowed to maintain versions of libraries that project used, as well as building and running it. For application development purposes authors chose Visual Studio Code IDE (Integrated Development Environment) that allowed to manage project files and integrate it with programming plugins and the Git version control system. The Git is a tool that ensures managing project as remote repository on GitHub platform [14]. For the duration of analysis, created website was hosted by Firebase hosting [15].

Authors focused on functionalities connected to displaying and adding content of singular movie. Moreover, emphasis was put on distinguishing page contents from advertisements. There were defined personas and interface mockups created during design process of the application. It helped to determine functional and non-functional requirements. Moreover, created mockups allowed to avoid initial mistakes with content placement within the interface before implementation. Figure 1 shows an example view of the user interface implemented for the purpose of the study.

Figure 1: Example view from authors' interface.

3.2. Research criteria

Questionnaires focused on the user satisfaction and mainly verified if examined website is readable and easy to navigate. There was considered criteria such as: layout, color theme, navigating ease, information structure. Before filling in the questionnaires, users cannot be able to see interface of newly developed service to avoid a situation when user knows the arrangement of the graphical user interface Tests carried out with eye-tracking technique as a practical approach were performed to assess average element searching time, number of fixations and time to first fixation on searched element. There were also heatmaps generated to visualise frequency of visiting specific interface areas (Figure 3). In addition, there was validators tests carried out to verify quality of source code and check compliance with WCAG and W3C guidelines. Those directives consider criteria such as: contrast errors and warning, code structure errors, warnings and Accessible Rich Internet Implementation (ARIA) attributes count.

3.3. Research procedure

3.3.1.Questionnaire

In order to carry out the research, both tools for obtaining the most objective result of the website usability assessment were used, as well as methods taking into account the subjective feelings of users. Ten people aged 23-25 with similar education, who regularly use internet services, were involved in the research. The surveys focused on the highest possible accessibility and intuitiveness of using a given website. To evaluate quality of the user interface the Lublin University of Technology (LUT) questionnaire [1] was used. The participants focused on LUT's areas such as "Navigation and structure", "Messages, feedback, user support", "Application interface", "Text of subpages", "Data entry", with each question using a scale from 1 to 5, where 1 meant worst score and 5 the best one. Working with both services, users were supposed to complete the same set of tasks. After realizing all tasks users were given questionnaires to rate their experience. Example scenario of testing the service before filling the questionnaire:

1. Finding where user is actually now.
2. Locating the searchbar.
3. Distinguish different sections after reaching each one.
4. Remembering the path to go back.

3.3.2.Eye-tracker

The eye-tracker tests were carried out in accordance with the research scenarios prepared after the preliminary analysis of interface not complying with universal design concepts as well as interface implemented by authors. Scenarios consisted of sequence of tasks, that user had to perform. They concerned issues affecting the usability such as: finding a given element on the page, testing the speed of finding an information with different contrasts, or the impact of advertisements display methods and their placement on distinguishing them from subpage content.

Table 1: Eye-tracker specification

Visual angle accuracy	0.5-1.0°
Sampling rate	60Hz or 150Hz
Calibration	5 or 9 point
Permissible head movement	35cm (vertically) x 22cm (horizontally)
Head movement depth range	±15cm
Physical parameters (dimensions)	235 x 45 x 47 mm, 125g

A single scenario considered actions like reading the task and finding the element on page. Example scenario of usability testing was placed in Table 2.

Table 2: Scenario of locating interface elements that implements concepts of universal design

Name: Speed analysis of locating interface elements that implements concepts of universal design.		
Research aim: Verify impact of elements arrangement in user interface on locating speed.		
Initial conditions: Next views are presented to user.		
Final conditions: Data gathered from eye-tracker has been saved after end of scenario.		
Participants in research: 10		
Scenario steps		
Number	**Description**	**Expected result**
1	Instruction is viewed to user that concerns locating of adding movie button.	User begins the research.
2	View of interface fulfilling the universal design concepts is presented to user.	User tries to locate the element.
3	User has visually located searched button.	User begins next scenario.
End of scenario		

The time that participants spent on completing the task of the scenario was measured. Additionally, the eye-tracker system recorded their eye movements, thanks to

which it could visualize the "route" on the website interface. The same scenarios were conducted using interfaces of both services. The specification of eye-tracker was presented as contents of Table 1.

3.3.3.Validators

Testing with the free web-based WAVE [16] tool was aimed at obtaining errors in the source code that have a real impact on the reception, and thus the usability of the page, such as the lack of alternative text in the case of an unloaded image or inadequate contrast. After conducting the tests, the researchers received the results in the form of a list of errors and warnings related to specific elements of the website.

In order to highlight the errors contained in the code, but which may not be directly reflected in the visualization of the page, but influencing its functioning, the W3C validator was used [17]. The validator listed errors and warnings in the source code and marked, for example, which attributes were missing in a given element.

AChecker validator [18] is an online tool that provides website accessibility review taking into consideration up to AAA level WCAG 2.0 guidelines. By testing with AChecker, authors received the list of contradictions with accessibility guidelines on appropriate level.

4. Results

4.1. Eye-tracker test results

Results concerning the average task realisation time per respondent gathered from eye-tracker test were presented in Figure 2. Analysis of eye-tracker test results showed if participants had problems finding the desirable element. Heatmaps gathered thanks to research scenarios were marking more issues in case of Filmweb interface versus interface proposed for the purpose of the study. The plot contained standard deviations for both tested interfaces as well.

In Figure 3 the most visited areas concerning eye movement were visualised. Heatmap is a resultant of series of completing research scenario by all respondents. The most visited areas, marked with red color, presented in Figure 3 are: poster, trailer miniature, title, information about movie crew and rating section. Considering the fact, that task was to find button allowing user to enter the trailers section, this view made numerous difficulties during the experiment. The correct solution was to click on poster, which was not intuitive. Participants had trouble recognising that, until he or she hovered over this area. There also wasn't appropriate information present to give user information about action performed after clicking that element. Users were misled by the play button, that indicated possibility of watching trailers of movie.

4.2. Questionnaire results

After conducting LUT questionnaire, the WUP (Web Usability Points) indicator was calculated [1]. Table 3 contains partial respondent-based and average WUP scores for both interfaces. WUP rate was counted using formula presented in equation 1 [1].

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

where: n_a is areas number, s_i is subareas number in area i, q_{ij} is questions number in area i and subarea j, p_{ijk} is grade for question number k in subarea j and area i.

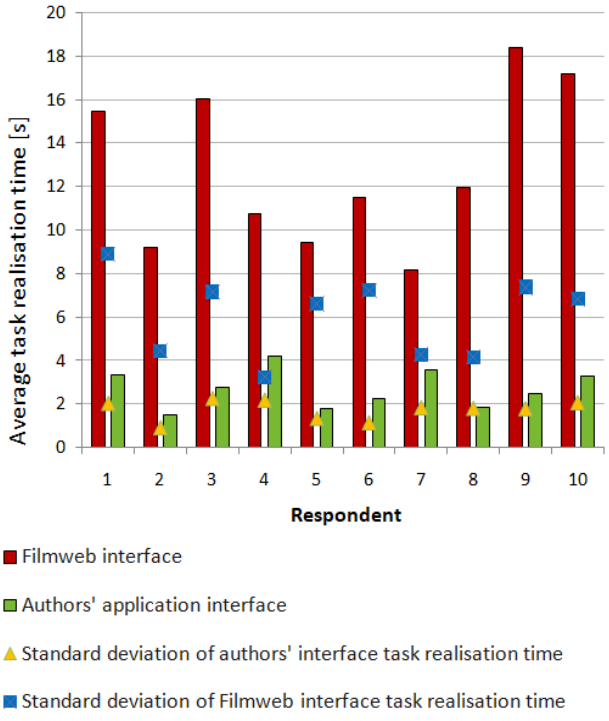


Figure 2: Average task realization time for both interfaces.

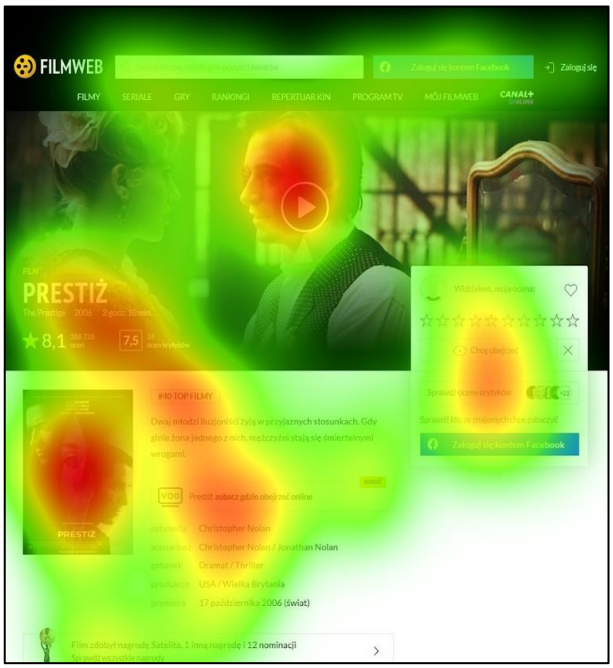


Figure 3: Heatmap for Filmweb interface examine scenario.

Table 3: Partial and average WUP scores for both interfaces

WUP scores		
Interviewee	Authors' interface	Filmwebs interface
1	4.76	3.25
2	4.41	3.16
3	4.44	3.56
4	4.70	3.21
5	4.42	3.34
6	4.30	3.08
7	4.24	3.14
8	4.46	3.22
9	4.35	3.05
10	4.35	3.16
Average WUP	**4.44**	**3.22**

4.3. Validator results

Obtained results from WAVE validator test were presented in Figure 4. Web Accessibility Evaluation allowed to outline problems contained in both services such as: errors, contrast errors, alerts, features, structural elements and ARIA.

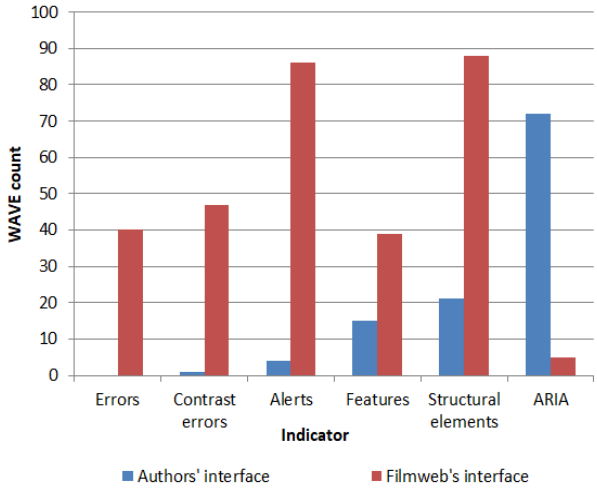


Figure 4: WAVE validator results visualization.

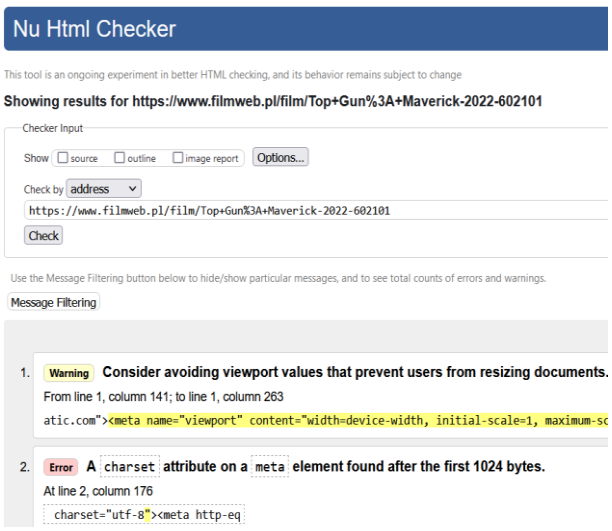


Figure 5: W3C validator results for Filmweb subpage interface.

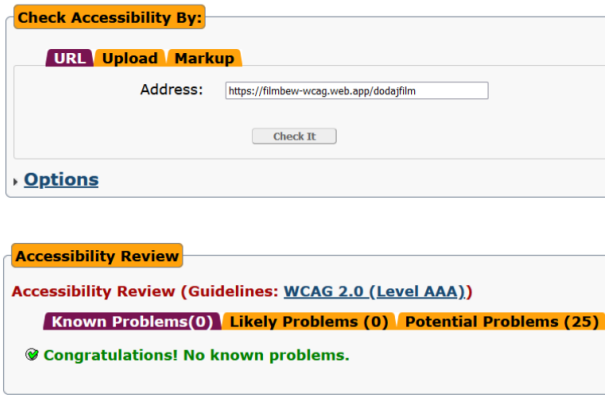


Figure 6: AChecker validation results for authors' interface.

The W3C validator helped to distinguish warnings and errors contained in the code in a list form. Tests were performed for chosen subpages of Filmweb and in contrast for authors' interface. Example results for analysed services were presented in Figure 4.

Compliance with WCAG 2.0 guidelines was checked and visualised also by the AChecker tool. Figure 5 contains screenshot of a test result for interface proposed by the authors.

5. Discussion

Taking into consideration objective and non-objective research methods, the authors were able to verify the hypothesis formulated at the beginning of the paper. Objective methods were eye-tracking and validator testing. On the other hand, there was one non-objective method, the questionnaire. Research group which includes ten people who was asked to rate both tested interfaces using LUT questionnaire and take part in eye-tracking study.

5.1. Objective methods

Eye-tracking and validators were two main objective usability analysis methods used in articles discussed in literature review.

Eye-tracking method proposed in article "Accessibility of environmental data for sharing: The role of UX in large cyberinfrastructure projects" [6] ensures objective assessment intuitiveness of interface taking into consideration such factors as examining time, time to first fixation and correctness of examination. Collected data were gathered in tables. In the study presented in this paper, the authors also interpreted heatmaps. Analysis of average scenario completing time shown that executing the task of finding element takes 474% more time in case of interface not fulfilling universal design concepts (Figure 2).

The authors have predicted that some of the Filmweb views can be misleading so they prepared some hints for the specific scenarios. It was important to distinguish the time that participant took to accomplish task from the time to the first fixation. As an example, it can be observed during scenarios for finding trailers sections in Filmweb. This problem has reflection in results and states interface designed for purpose of study as more intuitive (Table 3).

Another method for objective interface analysis was to use validators such as AChecker, W3C, and WAVE. Similar tools were also presented in articles, where European health-related websites were tested [3] and websites user experience was analysed [5]. Authors of this paper has rejected the Nibbler tool, hinted by mentioned articles from literature review [2, 3], because of its inability to point specific service subpage to test. WAVE results represented by alerts, errors and contrast errors were many times higher. On the other hand, such score in section like ARIA was positive indicator that reflected on containing attributes in code that have major effect on accessibility. According to the W3C validator results, Filmweb interface had numerous usability defects. AChecker tool allowed to verify compliance with WCAG 2.0 guidelines [4]. The interface created for the purpose of the study fulfills the universal design guidelines, what comes out in favor of validator examination. For example, analysis of movie-info subpage for Filmweb interface has count for known problems at rate of 88 in contrary to authors' interface with zero known problems (Figure 6).

5.2. Non-objective methods

The non-objective interface analysis was mainly realised by employment of questionnaire [9, 10]. Authors availed of LUT questionnaire to gather respondent's subjective evaluation after using both compared interfaces. Afterwards, the WUP indicator was calculated, which was in average 38% higher in case of interface fulfilling universal design concepts. Despite the contrast warning during validator analysis, authors have decided to visualise movie rating with the use of stars symbols instead of numeric grade. This decision was influenced by the interviews with users, who claimed that star rating has significantly higher aesthetic value [7]. According to problems raised by authors of article "Redevelopment of the Predict: Breast Cancer website and recommendations for developing interfaces to support decision-making" authors of this paper had created appbar with service name in it, in order to ensure simple navigation [8].

6. Conclusions

Comparative analysis of user interfaces of movie related services allowed to indicated factors having impact on usability. The obtained results have shown that modern design of interface used by movie services is more appealing but it often doesn't follow universal design guidelines. Thanks to eye-tracking technique there were distinguished elements and areas that drew attention of potential user. Variety of heatmaps brings up intuitiveness issue. Heatmaps created from interface fulfilling universal design concepts show that user was able to find element or information without analysing whole page view. The questionnaire result analysis and the indicators calculated with its use shown that usability of interface which sticks to WCAG and universal design guidelines was considerably higher. Validator testing pointed out that created interface was more accessible.

The obtained results allowed to verify the hypothesis "deploying universal design concepts significantly improves interface usability". Tests carried out on both interfaces had shown that universality factors had major impact on participants scores. Interface that implements terms of universal design turned out to be less problematic and got better feedback in contrary to one that did not. Therefore, the hypothesis is true.

References

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

[2] I. Abuqaddom, H. Alazzam, A. Hudaib, F. Al-Zaghoul, A measurable website usability model: Case Study University of Jordan,10th International Conference on Information and Communication Systems (2019) 83-87, https://doi.org/10.1109/IACS.2019.8809145.

[3] C. Sik-Lany, É. Orbán-Mihálykó, Accessibility Testing of European Health-Related Websites, Arabian Journal for Science and Engineering 44 (2019) 9171-9190, https://doi.org/10.1007/s13369-019-04017-z.

[4] S. V. Fernandez, M. A. Majid, N. A. Abu Bakar, M. Fakhreldin, Enhanced Colour Scheme Assessment Tool (COSAT 2.0) for Improving Webpage Colour Selection, Proceedings - 2021 International Conference on Software Engineering and Computer Systems and 4th International Conference on Computational Science and Information Management (2021) 459-464, https://doi.org/10.1109/ICSECS52883.2021.00090.

[5] Y. Kurniawan, G. H. Prasetya, F. Malvin, S. Dharmawan, N. Anwar, Johan, User Experience Analysis of binus.ac.id Website with The Usability Testing Perspective (A Case Study Approach), Proceedings of 2021 International Conference on Information Management and Technology (2021) 423-428, https://doi.org/10.1109/ICIMTech53080.2021.9535055.

[6] R. Volentin, A. Specht, S. Allard, M. Frame, R. Hu, L. Zolly, Accessibility of environmental data for sharing: The role of UX in large cyberinfrastructure projects, Ecological Informatics 63 (2021) 101317-8, https://doi.org/10.1016/j.ecoinf.2021.101317.

[7] J. Pibernik, J. Dolic, H. A. Milicevic, B. Kanizaj, The Effects of the Floating Action Button on Quality of Experience, Future Internet 11(7) (2019) 148-158, https://doi.org/10.3390/fi11070148.

[8] N. Nouman, A. Umer, Web Navigation and Usability Analysis of Educational Websites in Pakistan, 2019 Seventh International Conference on Digital Information Processing and Communications (2019) 57-62, https://doi.org/10.1109/ICDIPC.2019.8723704.

[9] G. D. Farmer, M. Pearson, W. J. Skylark, A. L. J. Freeman, D. J. Spiegelhalter, Redevelopment of the Predict: Breast Cancer website and recommendations for developing interfaces to support decision-making, Cancer Medicine 10 (2021) 5141-5153, https://doi.org/10.1002/cam4.4072.

[10] K. Kous, M. Pušnik, M. Heričko, G. Polančič, Usability evaluation of a library website with different end usergroups, Journal of Librarianship and Information Science 52 (2020) 75-90, https://doi.org/10.1177/0961000618773133.

[11] Material UI. React UI framework documentation, https://mui.com/material-ui/getting-started/overview/, [10.10.2021].

[12] React. A JavaScript library for building user interfaces, https://reactjs.org/docs/getting-started.html, [10.10.2021].

[13] Npm documentation, https://docs.npmjs.com/about-npm, [10.10.2021].

[14] Git. Version Control System documentation, https://git-scm.com/docs/git, [10.10.2021].

[15] Hosting Firebase documentation, https://firebase.google.com/docs/hosting, [01.03.2022].

[16] WAVE Web Accessibility Evaluation Tool, https://wave.webaim.org/, [01.03.2022].

[17] W3C Validator, https://validator.w3.org/, [01.03.2022].

[18] AChecker Web Accessibility Checker, https://achecker.achecks.ca/checker/index.php, [01.03.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 309–314
Received: 16 June 2022
Accepted: 2 July 2022

Evaluation of Flutter framework time efficiency in context of user interface tasks

Ocena wydajności czasowej frameworku Flutter w kontekście obsługi interfejsów użytkownika

Damian Białkowski*, Jakub Smołka

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

Abstract
The article describes a comparative analysis of the time performance between native Android applications (created with the Android SDK and Java language) and applications created with the Flutter multi-platform framework. The study consisted of creating three pairs of applications that are functionally identical with each other using both programing tools, and then examining the time taken to perform individual actions by both applications. The functionality of the test applications consisted mainly of tasks related to operating on the user interface. The study was carried out on a Huawei P30 phone using the Perfetto tool. Results confirm that native apps are more time efficient than Flutter apps.

Słowa kluczowe: Flutter; szkielet wieloplatformowy; aplikacja mobilna; Android

Streszczenie
Artykuł opisuje analizę porównawczą wydajności czasowej aplikacji natywnych systemu Android (stworzonych za pomocą Android SDK oraz języka Java) oraz aplikacji stworzonych za pomocą wieloplatformowego frameworku Flutter. Badanie polegało na stworzeniu trzech par identycznych ze sobą funkcjonalnie aplikacji za pomocą obu rozwiązań, a następnie zbadaniu czasu wykonania poszczególnych działań przez obie aplikacje. Funkcjonalność aplikacji testowych składała się głównie z zadań z zakresu operowania na interfejsie użytkownika. Badanie zostało przeprowadzone na smartfonie Huawei P30 za pomocą narzędzia Perfetto. Wyniki potwierdzają lepszą wydajność czasową aplikacji natywnych względem aplikacji Fluttera.

Keywords: Flutter; cross – platform framework; mobile app; Android

*Corresponding author
Email address: damian.bialkowski@pollub.edu.pl (D. Białkowski)

©Published under Creative Common License (CC BY-SA v4.0)

1. Introduction

Cross-platform technologies are one of the most popular development directions of IT industry. The essence of these solutions is the ability to create applications for many different environments using only one code base. This solution has two obvious advantages. Firstly, it significantly reduces the time needed to deliver the end product to all platforms, which translates into the amount of resources needed to complete the project, and thus the cost of product delivery. The second advantage is that a multi-platform software developer does not need to know all native technologies for each target platforms.

Flutter framework is being developed by Google corporation and was released in 2017. This tool allows for creating Web, mobile (Android and iOS) as well as desktop and embedded applications. It uses Dart as its programming language. Dart is a language optimized for user interfaces creation.

Cross-platform frameworks have significant advantages over native solutions of specific environments, however there are concerns regarding their quality and usability. The objections mainly concern the efficiency of multi-platform solutions, as well as the possibility of obtaining the same functionalities and appearance of the application as in the case of native applications.

In case of mobile applications performance testing, researchers usually inspect such actions as: sorting data, database writing and reading or downloading file from the Internet. This article, however, describes the less frequently studied aspect of mobile applications, which is the display and manipulation of user interfaces. In regard to the Flutter framework, it is even more important, because the manufacturer strongly emphasizes its usefulness as a tool for building user interfaces.

The purpose of the study is to determine whether there is a difference between the time efficiency of an application created with the Flutter framework and the time efficiency of a native Android application (created with the Android SDK and Java language). Tasks performed by applications consists of user interface operations.

2. Literature review

Despite the short presence of the Flutter framework on the market, first scientific publications about this tool are gradually appearing.

The Authors of the article [1] conducted a multi-criteria analysis of applications written in the Flutter framework. Execution times of the following operations were examined: writing and reading data from a file, sorting data table, writing and reading data from a local

database. Results were compared to native Android application. In most cases, the native application has proven to be more efficient or as efficient as the application developed with Flutter. Additionally, the size of the source code for both applications was examined. However, no tests have been undertaken on the display of the user interface.

In the article [2], the Authors examined the performance differences between mobile applications of Android and iOS systems created with the following tools: Flutter, React Native and native languages, respectively Java for Android and Swift for iOS. The execution time of the following tasks was measured: HTTP query with the retrieval of response in JSON format, displaying the first 5 elements of the list on the screen, writing 7 elements of the list to the local database and reading 5 elements from this database. On both platforms native applications performed list display and database read tasks faster. Native applications of iOS also performed data retrieval via HTTP protocol faster. In the remaining cases, there was no significant difference in execution time or multi-platform applications completed faster. The authors indicated that cross-platform frameworks often match the performance of native applications.

An example of using the Flutter framework application in a real IT system is presented in the article [3] in which the Authors created a system to support shipment tracking by using the GPS technology. The system included both viewing shipments by customers and handling the status of the package by the supplier. It consisted of a web application created with HTML, CSS and JavaScript tools, a mobile application created with the Flutter framework, an application server created in the Node.js technology, while the Firebase Realtime Database was used as a database. The results of the article show the usability of the Flutter tool in real-life projects. However, the efficiency of the system components has not been tested.

For a deeper analysis of the cross-platform tools efficiency problem, it is also worth taking into account the studies on frameworks other than Flutter, as usually those technologies are also compared to native applications.

The Authors of the article [4] have analyzed the impact of using various programming tools on the performance of a mobile application. The subject of the research were the following programming tools: Android SDK with Java, Android NDK, Xamarin and Apache Cordova. For each of the platforms, an application was created that tested the execution time of the following tasks: sorting an array of 100 000 items, saving a 10 MB file to the device's storage, and reading a 10 MB file from the device's storage. However, the authors did not specify unambiguously which tool is the most efficient, because in various test cases, applications in various technologies were the fastest. The differences in execution times were determined to be imperceptible to the user. It is also important that the authors found large differences in the results between the tests carried out on the physical device and on the emulator.

The Authors of the article [5] conducted a study comparing the performance of mobile applications created with the Xamarin tool (in two versions - Forms and Native) to applications created in native technologies of Android and iOS mobile systems. The execution time was tested of applications that performed the following test scenarios: computing the number of π to ten thousandths of a decimal place, writing and reading this number to a file, downloading a file of approximately 7 MB over the network, and reading location coordinates. In the vast majority of cases, native applications have proven to be more efficient, although these have not always been a big difference. The authors of the article concluded that the benefits of using a multi-platform tool justify its use, as long as the disadvantages of this solution are acceptable and do not have a critical impact on the project implementation.

In the article [6], the authors described the concept of creating cross-platform hybrid applications and the reason why this type of approach is present on the market. This approach was compared to two other solutions to the problem: creating native applications for each of the target mobile operating systems and creating web applications optimized for mobile devices. It has been shown that hybrid applications can combine the advantages of using one code base on many platforms, as is the case with web applications, as well as the ability to access native functions of mobile devices such as a camera or GPS by using capable API of the hardware platform.

The authors of the article [7] conducted an analysis of the possible impact of multi-platform applications on the mobile technology market. Possible advantages, disadvantages and the general impact of selecting one of the paradigms: single-platform or multi-platform on the application life cycle were shown. The analysis was carried out from the perspective of three groups related to the application: customers, programmers and suppliers of the platform on which the application runs. One of the anticipated drawbacks is the inability to fully utilize the hardware's capabilities. Flutter framework, however, differs from the solutions available at the time of writing the article, and the authors themselves promote it as a tool that allows one to achieve performance comparable to that for native applications. The authors also consider the possibility that cross-platform applications may not provide user experiences comparable to native applications.

3. Research method

In order to compare the time efficiency between native applications and applications developed in the Flutter framework, three pairs of Android mobile applications have been developed. Each pair of applications consists of one native application created with the Android SDK and Java language, while the other one is created with the Flutter framework. Both applications in a pair were created in such a way to be as similar as possible in terms of functionality, appearance and logic.

Each of the pairs of applications executes one of the test scenarios. The test scenarios consist of specific actions on the user interface. The work time of the threads responsible for displaying the user interface was measured, both for individual tasks and for the entire test.

The performance of the application is understood as the threads' work time, other thread states have not been taken into account. Measured time will therefore be the time of the threads' work during the execution of the task. It is worth noting that this is not the total time of the task execution, but the time during which the thread was up and running for a specific task. Such a measurement option was decided because it was the most reliable and the most platform-independent method of conducting the test.

In order to measure the work times of the threads, Android's built-in system tracing function was used. The Perfetto tool was used to trace the system. The reports generated by this tool were analyzed in an Excel spreadsheet.

Since it would be difficult to isolate the performance of individual tasks in the Perfetto report graph, 5-second breaks were introduced between individual tasks. This enables easy verification of subsequent tasks. It also has no effect on the results because the wait is not implemented in such a way that the thread is asleep, not blocked, and not doing any work. As the thread runtime is tested and not the total test execution time, the results are not negatively affected.

Research was carried out on Huawei P30 smartphone with EMUI operating system (version 12) based on Android API version 29.

3.1. Research scenarios

Each test scenario was carried out in following manner:

1. The researcher launches script initializing system tracing on smartphone via USB connection from computer command line.
2. The test application is launched on the mobile device.
3. The test application performs the tasks defined by the scenario. The course of the study is observed visually by the researcher.
4. After 60 seconds system tracing is stopped and Perffeto report is opened in Web browser. Report file is saved for further analysis.
5. After performing all the tests in the series, the researcher saves the relevant data from the result files to an Excel spreadsheet. The data in the sheet is analyzed and compared with the results for the second application in pair.

3.1.1. Research scenario no. 1

The first research scenario involves performing operations on images. The test consists of the following tasks:

1. Application displays an image of specific size.
2. Image is replaced with different image.
3. Image is moved down. The motion is animated. Animation duration time is 2 seconds.
4. Red color filter is applied to the image.
5. Image is scaled to specific size.

3.1.2. Research scenario no. 2

This scenario is based on performing actions with text characters. The test scenario includes the following tasks:

1. Text consisting of 2 000 random alphanumeric signs in a scrollable text field is displayed.
2. Font size is changed.
3. The displayed text is changed to a new random text with 10 000 random alphanumeric signs.
4. The font color is changed to green.
5. Underline is applied to the text.

3.1.3. Research scenario no. 3

The last scenario includes operations on various popular user interface elements. The scenario includes the following tasks:

1. Application window with Drawer (swiping navigation menu), Appbar (bar with application title displayed on top of application) and Floating Action Bar (button displayed over application main content) is displayed.
2. 3 radio buttons are displayed in the application window.
3. One of the radio buttons is selected.
4. Radio buttons are deleted and 6 input text fields are displayed instead in the application window.
5. the text „HelloWorld!" is set in the input fields.
6. Input fields are deleted and a scrollable list of 150 elements is displayed. Each list element is a text label with a index of the element in list.
7. Scrolling the list by several items.

4. Results

During the tests, it turned out that while the native application performed the operations responsible for the user interface in 2 threads, as expected, the application developed with Flutter had 4 threads responsible for the user interface. The Flutter application uses both the main application thread (in the Android documentation it is called the UI thread) and the Render thread, typical for Android applications, however, the system trace output file also shows a second thread named UI and a thread named Raster. These threads are used by the Flutter system, which is confirmed by the documentation of this framework [8].

In order to distinguish between the two UI threads, the main, typical Android thread, was named the UI application thread, and the second UI thread was named Flutter's UI thread.

Since there are only 2 threads in the native application, while in the Flutter application there are 4, in order to make a fair comparison for the Flutter application, the working times of both UI threads were added together and compared to the UI thread time of the native application.

The work times of the Render and Raster Flutter threads were added and compared with the work time of Render thread of the native application.

4.1. Results for research scenario no. 1

As a result of the research carried out for the tests of the first research scenario, the following results were obtained:

The results for UI threads' (Table 1) show a significant difference between the execution time of the first scenario in UI threads. The native application finished work much faster.

Table 1: UI threads' work time for first research scenario

	Flutter application	Native application
Task no.	Average execution time ± standard deviation (ms)	Average execution time ± standard deviation (ms)
1	365.66 ± 32.35	212.22 ± 14.31
2	64.23 ± 6.00	194.01 ± 10.41
3	278.84 ± 8.29	181.86 ± 3.42
4	4.72 ± 0.37	2.25 ± 0.11
5	5.07 ± 0.17	3.72 ± 0.65
Entire test	718.53 ± 38.82	594.07 ± 13.35

The difference is especially noticeable in the first task, in the UI threads of both applications. The native application completed the task more than 100 ms faster. This is especially important because this task includes the start of the application. The difference is so large that during the observation of the course of the study, the difference in the time of starting the application was visible to the naked eye.

The second, big difference was observed in the time it takes to change the image. This time, Flutter application did the task much faster. However, when observing the examination in the Flutter application, the image "blinked" - the first image disappeared for a moment, revealing the application background below the image, and only later was the second image displayed. In the case of a native application, the image was replaced immediately, without this "blink". Thus, in terms of aesthetics, the task was performed better in the native application.

The native application performed almost all tasks faster, except for task no. 2.

Table 2: Render threads' work times for first research scenario

	Flutter application	Native application
Task no.	Average execution time ± standard deviation (ms)	Average execution time ± standard deviation (ms)
1	32.10 ± 2.01	55.28 ± 4.69
2	28.17 ± 11.24	49.37 ± 2.53
3	399.20 ± 8.12	334.55 ± 7.30
4	12.55 ± 1.17	10.56 ± 13.16
5	6.59 ± 0.92	4.69 ± 0.85
Entire test	478.62 ± 14.40	454.45 ± 14.91

In the case of rendering threads (Table 2), the difference between the execution times of both applications is small and in practice it is not large enough to be significant in the context of indicating a faster application.

In this case, the Flutter application performed the first 2 tasks faster, while the native application was faster from third task to the end of test.

Analyzing the results for the first of the test scenarios, it can be concluded that the main performance difference lies in the work time of the UI threads, and in particular for the task involving launching the application.

To sum up, in the first test scenario the native application turned out to be more efficient than the application created with Flutter.

4.2. Results for research scenario no. 2

The research for the second test scenario led to the following results:

Table 3: UI threads' work time for second research scenario

	Flutter application	Native application
Task no.	Average execution time ± standard deviation (ms)	Average execution time ± standard deviation (ms)
1	316.75 ± 8.85	168.09 ± 12.02
2	24.87 ± 4.16	64.11 ± 4.30
3	106.35 ± 6.57	163.48 ± 7.67
4	36.47 ± 9.64	3.18 ± 0.34
5	59.74 ± 18.93	118.31 ± 3.83
Entire test	549.97 ± 16.90	517.16 ± 12.15

The results (Table 3) indicate that once again the native application was executed more efficiently (in the context of UI threads) than the Flutter application. In the case of the second test scenario, however, the differences in the total work time of the threads were much smaller than in the first test scenario.

Detailed analysis of the table shows that in the case of text operations, some tasks were completed faster in the native application, and some in the Flutter application. Invariably, a big difference appeared when the application was started. It is worth noting that the difference in the execution times of task no. 1 for UI threads is again the largest difference among the execution times of individual tasks.

Table 4: Render threads' work times for second research scenario

	Flutter application	Native application
Task no.	Average execution time ± standard deviation (ms)	Average execution time ± standard deviation (ms)
1	34.64 ± 2.64	63.37 ± 2.10
2	20.49 ± 3.17	12.96 ± 3.90

3	8.60 ± 6.64	6.44 ± 0.87
4	45.92 ± 4.85	14.98 ± 1.25
5	13.09 ± 3.00	8.04 ± 1.00
Entire test	122.74 ± 10.96	105.3.85

The work time differences for rendering threads are even lower than for UI threads (Table 4). Similar to the first test scenario, the native application turned out to be only slightly more efficient than the Flutter application.

The native application did all the tasks faster except for the first task.

The difference in the runtime of rendering threads can again be considered too small to indicate a significant performance difference to the detriment of the Flutter framework.

4.3. Results for research scenario no. 3

The results of the third test scenario are as follows (Table 5, Table 6):

Table 5: UI threads' work times for third research scenario

	Flutter application	Native application
Task no.	Average execution time ± standard deviation (ms)	Average execution time ± standard deviation (ms)
1	296.11 ± 9.73	106.98 ± 21.53
2	13.39 ± 2.42	10.74 ±0.71
3	31.00 ± 2.03	82.27 ± 5.68
4	53.02 ± 6.04	15.97 ± 0.86
5	68.92 ± 8.12	13.63 ± 1.19
6	50.46 ± 5.22	86.85 ± 6.17
7	20.73 ± 3.42	16.26 ± 1.05
Entire test	533.63 ± 18.19	332.83 ± 27.81

As in the 2 previous test scenarios, the native application again performed tasks faster for the UI thread (Table 5).

The native application performed 5 out of 7 tasks faster. The differences in the execution times of the entire test, similar to the first test scenario, are large, as the difference is over 200 ms, which, taking into account the test duration, is a very significant difference in favor of the native application.

Table 6: Render threads' work times for third research scenario

	Flutter application	Native application
Task no.	Average execution time ± standard deviation (ms)	Average execution time ± standard deviation (ms)
1	29.58 ± 2.58	26.09 ± 1.39
2	6.30 ± 0.80	7.36 ± 0.28
3	59.08 ± 4.59	118.85 ± 5.46
4	21.41 ± 2.78	8.03 ± 1.01
5	50.61 ± 2.36	8.42 ± 0.71
6	14.07 ± 1.83	45.30 ± 2.90
7	11.52 ± 1.23	35.66 ± 1.10
Entire test	192.59 ± 5.72	249.70 ± 7.64

Interestingly, rendering threads ran faster in the Flutter application (Table 6). This is the only case in the entire study where the average overall test execution times are lower for the Flutter application.

Moreover, it should be noted that the difference in the duration of the entire test is quite large. When it comes to specific operations, the native application performed 3 tasks faster, while the Flutter application - 4.

5. Conclusions

The study was successfully completed. All tests were performed and a complete set of data was obtained for analysis.

In the first test scenario, the native application definitely outperformed the Flutter application. Total thread working times were lower, and most individual tasks were completed faster by the native application. The result of the first test scenario show difference in performance of both applications. It should be emphasized that this performance difference is mainly caused by the difference in the operating times of the UI threads. Among the specific tasks, the biggest difference in execution time can be seen for task no. 1.

Much lower differences in execution times in the second research scenario. The test execution times for both the UI threads and rendering threads have little variation. The test confirms the performance differences, but in this case they were not large.

The tests of the last research scenario, like the tests of the previous scenarios, indicate a performance difference in terms of UI threads, however, for the third test scenario, the Flutter application performed faster for the rendering threads. This is the only such case in the entire study.

Summarizing the results of all tests, it can be confirmed that the native application are much more efficient in the UI thread than the Flutter application. A particularly big difference is visible when the application is started. This indicates that Flutter has to perform heavy tasks at the initiation of the application, which leads to a significant increase in the startup time of the application.

The slight performance difference was noticeable for rendering threads. Of course, in 2 out of 3 tests the native application was more efficient, but the differences were insignificant each time. Therefore, it can be concluded that Flutter's applications are satisfactorily efficient in this respect.

The research confirmed the differences in time efficiency between applications developed in the Flutter framework and mobile applications.

Android native applications are more efficient, so creating a native application will be better choice if top performance is critical aspect of application.

Considering the current state of knowledge in the area of the study, it has been confirmed again that native applications are more efficient. The study confirmed this statement for UI-based tests, which hasn't been researched before this study.

Study presents usefulness of UI-based tests in context of mobile frameworks quality evaluation.

Literature

[1] D. Gałan, K. Fisz, P. Kopniak, A multi-criteria comparison of mobile applications built with the use of Android and Flutter Software Development Kits, Journal of Computer Sciences Institute, 19 (2021) 107-113, https://doi.org/10.35784/jcsi.2614.

[2] L. P. Barros, F. Medeiros, E. Moraes, A. F. Júnior, Analyzing the Performance of Apps Developed by using Cross-Platform and Native Technologies, International Conference on Software Engineering and Knowledge Engineering (SEKE 2020).

[3] A. M. Qadir, P. Cooper, GPS-based Mobile Cross-platform Cargo Tracking System with Web-based Application, 2020 8th International Symposium on Digital Forensics and Security (ISDFS) 2020 1-7, https://doi.org/10.1109/ISDFS49300.2020.9116336.

[4] P. Kotarski, K. Śledź, J. Smołka, Analysis of the impact of development tools used on the performance of the mobile application, Journal of Computer Sciences Institute 6 (2018) 68-72, https://doi.org/10.35784/jcsi.642.

[5] P. Grzmil, M. Skublewska-Paszkowska, E. Łukasik, J. Smołka, Performance Analysis of Native and Cross-Platform Mobile Applications, Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska 7(2) (2017) 50-53, https://doi.org/10.5604/01.3001.0010.4838.

[6] C. M. Pinto, C. Coutinho, From Native to Cross-platform Hybrid Development, 2018 International Conference on Intelligent Systems (IS) (2018) 669-676, https://doi.org/10.1109/IS.2018.8710545.

[7] L. Corral, A. Janes, T. Remencius, Potential Advantages and Disadvantages of Multiplatform Development Frameworks – A Vision on Mobile Environments, Procedia Computer Science 10 (2012) 1202-1207, https://doi.org/10.1016/j.procs.2012.06.173.

[8] Flutter framework documentation – performance monitoring, https://docs.flutter.dev/perf/ui-performance, [14.06.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 315–322
Received: 17 June 2022
Accepted: 4 July 2022

Analysis of the graphical user interface of the online store, taking into account the methods of universal design
Analiza graficznego interfejsu użytkownika sklepu internetowego z uwzględnieniem metod projektowania uniwersalnego

Cezary Altmajer*, Piotr Błażewicz*, Maria Skublewska-Paszkowska

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

Abstract
Universal design is a philosophy of creating various products and the environment to adapt to the most comprehensive possible group of recipients. This article aims to compare interfaces with the principles of universal design and application interfaces that ignore such principles. After the literature review, the following research hypotheses were posed: "Higher interface contrast affects the visibility and speed of searching for individual elements of user interface" and "The arrangement of interface elements has a significant impact on navigating the website". The research was conducted on two websites. The Empik storefront was a webservice that did not comply with universal design principles. The application that follow the regulations of universal design was created for the purpose of research. Three methods of measuring the quality of interfaces were used in the study: WAVE tool, eye-tracking tests, and subjective assessment using the LUT questionnaire (Lublin University of Technology). Eye tracking study showed that participants needed an average of 2 times less time to locate elements on a high-contrast interface and 4 times less time to locate all components placed compliant with generally accepted design standards.

Keywords: universal design; contrast; arrangement of graphic elements; interface

Streszczenie
Projektowanie uniwersalne jest filozofią tworzenia różnych produktów oraz otoczenia tak, aby było ono dostosowane do jak najszerszego grona odbiorców. Celem niniejszego artykułu jest porównanie interfejsów z uwzględnieniem zasad projektowania uniwersalnego oraz interfejsów aplikacji, które pomijają takie zasady. Po zapoznaniu się z przeglądem literatury postawione zostały następujące hipotezy badawcze: „Większy kontrast interfejsu ma wpływ na widoczność oraz szybkość wyszukiwania poszczególnych elementów interfejsu graficznego" oraz „Rozmieszczenie elementów interfejsu ma zasadniczy wpływ na efektywność poruszania się po serwisie". Badania przeprowadzone zostały na dwóch serwisach internetowych. Serwisem niespełniającym zasad projektowania uniwersalnego była witryna sklepu Empik. Zaś aplikacja spełniająca zasady projektowania uniwersalnego została stworzona na potrzeby badań. W pracy zastosowano trzy metody pomiaru jakości interfejsów: narzędzie WAVE, badania okulograficzne oraz badania subiektywnej oceny za pomocą ankiety LUT. Badania okulograficzne wykazały, że uczestnicy badania potrzebowali średnio 2 razy mniej czasu na zlokalizowanie elementów na stronie o wysokim kontraście oraz 4 razy mniej czasu na znalezienie wszystkich komponentów umieszczonych w miejscach zgodnych z ogólnie przyjętymi normami projektowania.

Słowa kluczowe: projektowanie uniwersalne; kontrast; rozmieszczenie elementów graficznych; interfejs

*Corresponding author
Email address: cezary.altmajer@pollub.edu.pl (C. Altmajer), piotr.blazewicz@pollub.edu.pl (P. Błażewicz)

©Published under Creative Common License (CC BY-SA v4.0)

1. Introduction

The universal design is a philosophy of creating various products and the environment to adapt to the most comprehensive possible group of recipients [11]. It cannot be expected that everyone's requirements and needs will be fulfilled. You can only try to extend the group of users through various facilities. The most common problems of the interfaces of websites are the size, color and arrangement of elements, which makes it difficult for people with visual impairments to locate the things they need.

Hundreds of thousands customers are visiting the online stores every day. It is therefore important that the websites provide the most important principles of universal design, such as: equitable use, flexibility in use, simple and intuitive, perceptible information, tolerance for error, low physical effort, size and space for approach and use. Most people entering the website expect a search engine and a login form in the upper part, so they direct their eyes there first. An important element is also the ability to change the contrast and the ability to enlarge the elements of the graphical user interface (GUI) and the font. It is much easier to find an object of interest on a high contrast page.

The purpose of this article is to analyze the comparison of interfaces, taking into account the principles of universal design and application interfaces that ignore such principles. The analysis refers to three methods of study: subjective assessment carried out using the LUT survey, assessment of the Web Content Accessibility Guidelines (WCAG) 2.0 rules using the WAVE tool and an eye-tracking test, during which the time to locate individual elements of the graphical interface was

measured. The following study hypotheses were defined: “Higher interface contrast affects the visibility and speed of searching for individual elements of user interface” and “The arrangement of interface elements has a significant impact on navigating the website”.

2. Literature review

The article [1] presents study assessing the impact of the choice of development technology on the quality of website accessibility. The authors focused in particular on programming languages, Web and JavaScript frameworks and content management systems. Their availability was automatically assessed using QualWeb and Wapplayzer. Comparing Internet frameworks, the study indicated lower availability of websites using Microsoft ASP.NET technology com-pared to Ruby on Rails and Twitter Bootstrap.

The purpose of the study presented in publication [2] was to examine the Web Accessibility Initiative (WAI) guidelines on web accessibility in order to integrate it into the IT systems teaching program. The authors used the WebXact Accessibility Assessment Tool to test the best website pages from 23 California State University campuses to determine the level of compliance with federal standards. The results of the study showed that most of the pages tested did not fulfill the WAI guidelines.

The study presented in article [3] consisted in examining the availability of websites. Received the answer for the questions such as: why the topic of accessibility is so rare and what can be done to solve this problem from the perspective of students and lecturers were obtained. For this purpose, a survey among students was conducted. The study results showed that most of the participants were unfamiliar with any WAI guidelines. Conversely, with the knowledge of the WCAG, the majority were familiar with these guidelines.

The purpose of the study presented in publication [4] was to verify pages for compliance with availability. The authors of the article used the home pages of Kentucky academic libraries for this purpose which were tested using WebXACT and Semantic Data Extractor. The results of the study showed that only one institution created an extensive outline based on the use of the headline. The home pages of only two libraries had at least one access key on their home page.

The authors of the article [5] described the guidelines for accessibility such as proper screen size or font sizes together with the most effective practices for creating websites. The purpose of the work was to develop a tool and logic to verify the accessibility of websites, and then to examine professionally designed interfaces. Then, the developed tool was used to test 62 professional websites. During the study, 64 significant availability errors were found.

The purpose of the article [6] was to analyze software for automatic website validation according to WCAG 2.0 guidelines. The Moodle platform was used for this purpose during the evaluation. In the article, the authors assessed the selected tool and compared the testable WAVE errors with the WCAG 2.0 guidelines. It was found that WAVE performed well despite the lack of several WCAG 2.0 guidelines and that the tool is more suitable for developers than for general users.

The authors [7] review in their article the techniques for using the Web to read the screen. The review covered techniques for removing excessive content, handling online transactions, interacting with speech, and automating assistants. All of these techniques used non-semantic knowledge of web content to improve web usability. This review showed that understanding web content semantics is the overarching topic that drives web usability techniques.

During the study [8], the websites of the best universities were tested for accessibility using WCAG 2.0. The individual pages were assessed using HERA, Test de accessibilidad Web (TAW) and the Firefox Accessibility Evaluation Toolbar. Result of study showed that most educational sites follow less than 50% of the guidelines.

According to the knowledge of the authors of the analyzed publications, the study on the search time for elements on given pages with low and high contrast and elements located in places compliant and inconsistent with the generally accepted principles of universal design has not been performed.

3. Study methods

Three methods of measuring the quality of GUI were used in this study: WAVE tool, eye tracking tests and LUT questionnaire.

Individual stages of the study were as follow:

1. Defining the study issues, aim and study hypotheses.
2. Selection of websites under study.
3. Creating an online store application to carry out the study.
4. Development of study scenarios.
5. Design of experiments.
6. Conduct and registration of study.
7. Processing of collected data and analyzing them.
8. Evaluation of the results and conclusions.

3.1. Study object

The study was conducted on two websites. The site that did not fulfill the principles of universal design was Empik [9]. The page [10], which complies with the principles of universal design, was created for the purpose of this study. It is an online store that allowed you to register a user, log in (Figure 1) and purchase products. You can also change the contrast, resize text and interface elements and underline hyperlinks.

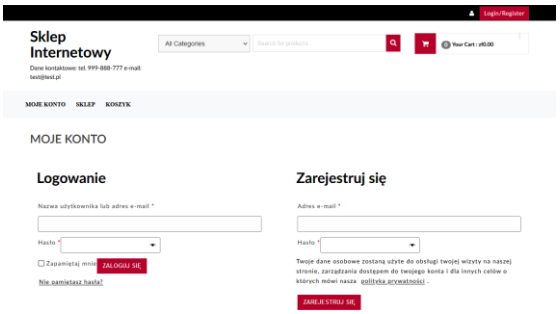


Figure 1: The "my account" subpage that allows you to log in and register.

3.2. Study group

Twenty IT students of the Lublin University of Technology with extensive experience in navigating websites participated in the study. Each of these people participated in the eye tracking study and half of them completed the LUT questionnaire.

3.3. WAVE Tool

In the first part of the experiment, the WAVE (Web Accessibility Evaluation Tool) validator was used [12]. This method enables quick and automatic examination of the website in terms of the availability of the WCAG 2.0 standard. This tool identifies site errors in 3 categories: contrast errors, page element structure and ARIA.

3.4. Eye tracking study

Then, an eye tracker study was conducted on Gazepoint GP3 HD [13] using 4 scenarios consisting of two different sets of commands.

The first set was conducted for searching elements on a low and high contrast page. The set contained the following commands:

1. Please locate the product search engine.
2. Please locate the product named "Billy Summers".
3. Please locate your shopping cart.
4. Please locate the "go to payment" button.
5. Please locate the field for entering discount coupons.
6. Please locate the "login" button.
7. Please locate "forgot password".
8. Please locate the total order value.
9. Please locate the "privacy policy".
10. Please locate the button "buy and pay".

The second set's purpose was to examine the search time for page elements located in places compatible and incompatible with generally accepted universal design principles for this type of pages and it contained the following commands:

1. Please locate the Product Finder.
2. Please locate the button that will take you to the login page.
3. Please locate your shopping cart.
4. Please locate social media.
5. Please locate the category change list.
6. Please locate the contact details.
7. Please locate the button leading to the top of the page.
8. Please locate the button that opens the accessibility menu.
9. Please locate the "send" button.
10. Please locate the "register" button.

When analyzing the data collected after the study, particular attention was paid to the time needed to complete a given task and the number of incorrectly completed tasks.

3.5. LUT Survey

The final stage of the experiment was to conduct a subjective assessment by means of the LUT questionnaire [14] among the participants of the study.

The survey was divided into 5 areas and 13 subareas. The questions can be rated on a scale from 1 to 5, where 1 was the worst and 5 was the best. Each participant was to complete the questionnaire after checking out the site, based on the following scenario:

1. The user enters the home page.
2. The user selects the element that allows to go to the "my account" subpage.
3. The user registers on the website.
4. The user logs in to his account.
5. The user goes to the store subpage.
6. The user changes the contrast.
7. The user increases the text size.
8. The user selects one of the products by going to its description.
9. The user adds 2 items of the product to the cart.
10. The user goes to the shopping cart.

The final rating of the examined website is the WUP (Web Usability Points) [14] indicator, the value of which ranges from 1 to 5, where 1 was the worst and 5 was the best.

4. Results

The static analysis was performed using the car library, minqa and carData packages and a tool created in the R language. In order to verify whether the distribution of the samples was similar to the normal distribution, the Shapiro-Wilk test was performed. Additionally, Levene's test and Student's t-test were performed to verify if the data were significantly different.

4.1. Eye tracking study – Time To First Fixation (TTFF)

Table 1 shows the times to the first fixation collected during the study on the impact of the interface layout on the search speed. Given data (table 1) relates to elements compatible (UD-enabled) and incompatible (No-UD) with generally accepted universal design principles.

Table 1: TTFF to locate an element relative to its arrangement

	UD-enabled website Time (ms)	No-UD website Time (ms)
Screen 1	815.6	2535.6
Screen 2	793.8	2330.9
Screen 3	272.1	1545.2
Screen 4	612.5	2295.7

Screen 5	832.1	2123.1
Screen 6	487.3	2249.4
Screen 7	753.2	3164.8
Screen 8	1299.1	4,622.7
Screen 9	617.7	3662.1
Screen 10	459.0	3595.8
Mean	694.24	2812.53
Variance	77986.85	856712.89
Std. Dev.	279.26	925.58
Conf. Int.	199.77	662.12
Shapiro-Wilk	0.93	0.93
Test Levene	5.01	
T-test	$1.78 * 10^{-6}$	

Table 2 shows the times to the first fixation of locating the element, collected during the tests, in which the influence of GUI contrast was taken into account. The speed of searching for elements in terms of low and high contrast was tested.

Table 2: TTFF to locate the element in terms of contrast

	UD-enabled website Time (ms)	No-UD website Time (ms)
Screen 1	683.4	2796.5
Screen 2	384.3	1444.9
Screen 3	356.5	2926.8
Screen 4	565.3	2132.7
Screen 5	924.2	1320
Screen 6	785.9	1523.4
Screen 7	437.7	1623.1
Screen 8	520.4	3632.2
Screen 9	594.3	3749.4
Screen 10	317.3	2707
Mean	38635.76	820907.72
Variance	556.93	2385.60
Std. Dev.	196.56	906.04
Conf. Int.	140.61	648.14
Shapiro-Wilk	0.95	0.91
Test Levene	23.62	
T-test	$6.96 * 10^{-6}$	

4.2. Eye tracking study – fixation number

Tables 3 – 5 present the study results of the fixations number for the page elements searching, which are placed in accordance with the generally accepted principles of universal design and for elements that do not fulfill these rules. The greater the number of fixations mean, the longer the user analyzed the page. This element determines whether the page is properly designed.

Table 3: The number of fixations for the study of the search time of page elements located in places compliant with the generally accepted principles of universal design

Scr.	UD-enabled website				
	Mean	Std. Dev.	Variance	Conf. Int.	Shapiro-Wilk
1	6.25	7.07	50.09	3.31	0.58
2	7.5	7.89	62.26	3.69	0.66
3	5.25	3.12	9.77	1.46	0.91
4	6.85	4.39	19.29	2.05	0.82
5	7.75	5.41	29.35	2.53	0.85
6	5.6	3.47	12.04	1.62	0.77
7	5.4	4.51	20.35	2.11	0.74
8	7	5.16	26.63	2.41	0.79
9	4.75	2.57	6.61	1.21	0.83
10	4.6	2.81	7.93	1.31	0.91

Table 4: The number of fixations for the study of the search time of page elements located in places inconsistent with the generally accepted principles of universal design

Scr.	No-UD website				
	Mean	Std. Dev.	Variance	Conf. Int.	Shapiro-Wilk
1	15.6	9.08	82.56	4.25	0.79
2	38.5	23.86	569.53	11.16	0.91
3	16.2	10.37	107.64	4.85	0.91
4	18.3	10,41	108.32	4.87	0.91
5	32.15	20.36	414.66	9.53	0.86
6	17.45	9.65	93.21	4.51	0.94
7	15.6	9.85	97,20	4.61	0.91
8	15	10.67	113.89	4.99	0.91
9	14.1	8.43	71,14	3.94	0.91
10	15.35	9.92	98,45	4.64	0.81

Table 5: Levene's test and t-test for pages with different arrangement of elements

	Test Levene's	T-test
Screen 1	0.76	$8.31 * 10^{-4}$
Screen 2	0.36	$2.63 * 10^{-6}$
Screen 3	0.18	$5.89 * 10^{-5}$
Screen 4	0.51	$5.65 * 10^{-5}$
Screen 5	0.56	$7.59 * 10^{-6}$
Screen 6	0.01	$7.91 * 10^{-6}$
Screen 7	0.14	$1.52 * 10^{-4}$
Screen 8	0.27	$4.52 * 10^{-3}$
Screen 9	0.02	$2.97 * 10^{-5}$
Screen 10	0.56	$3.81 * 10^{-5}$

Tables 6 - 8 contain the fixation numbers for high and low contrast pages.

Table 6: Number of fixations for a high contrast page

Scr.	UD-enabled website				
	Mean	Std. Dev.	Variance	Conf. Int.	Shapiro-Wilk
1	5.4	5.44	29.60	3.89	0.80
2	10.9	10.64	113.21	7.61	0.84
3	4.5	1.64	2.72	1.18	0.91
4	5.8	2.25	5.06	1.61	0.89
5	8.2	6.62	43.95	4.74	0.79
6	5.6	2.87	8.26	2.05	0.95
7	6.5	2.46	6.05	1.76	0.92
8	9.8	4.70	22.17	3.36	0.76
9	11	5.92	35.11	4.23	0.92
10	5.7	1.88	3.56	1.35	0.96

Table 7: Number of fixations for the low contrast page

Scr.	No-UD website				
	Mean	Std. Dev.	Variance	Conf. Int.	Shapiro-Wilk
1	15.9	16.86	284.54	12.06	0.78
2	17.9	11.15	124.32	7.97	0.89
3	15	11.24	126.44	8.04	0.85
4	14.5	8.94	80.05	6.40	0.85
5	9	4.13	17.11	2.95	0.94
6	11.9	5.51	30.32	3.93	0.98
7	11.5	5.64	31.83	4.03	0.92
8	18.1	12.62	159.43	9.03	0.71
9	23.2	13.91	193.51	9.95	0.91
10	17.6	10.96	120.26	7.84	0.88

Table 8: Levene's test and t-test for pages with different contrasts

	Test Levene's	T-test
Screen 1	2.37	$7.73 * 10^{-2}$
Screen 2	0.08	$1.68 * 10^{-1}$
Screen 3	7.95	$9.11 * 10^{-3}$
Screen 4	3.11	$7.99 * 10^{-3}$
Screen 5	0.42	$7.50 * 10^{-1}$
Screen 6	2.07	$4.89 * 10^{-3}$
Screen 7	4.61	$1.93 * 10^{-2}$
Screen 8	1.83	$6.72 * 10^{-2}$
Screen 9	4.57	$2.00 * 10^{-2}$
Screen 10	4.64	$3.32 * 10^{-2}$

4.3. Eye tracking study – fixation duration

Tables 9 – 11 contain the fixation times for searching page elements placed in accordance with the generally accepted principles of universal design and elements that do not meet these rules.

Table 9: Mean, standard deviation, variance and confidence interval for the study of the search time for page elements located in places compliant with the generally accepted principles of universal design

Scr.	UD-enabled website Time (ms)			
	Mean	Std. Dev.	Variance	Conf. Int.
1	1678.42	1564.61	2447995.29	732.25
2	1907.94	1812.01	3883383.94	848.04
3	1300.58	701.95	492736.05	328.52
4	1684.47	819.26	671198.46	383.42
5	2196.91	1284.58	1650162.87	601.21
6	1562.29	819.28	671221.19	383.43
7	1817.72	1419.83	2015914.12	664.50
8	2177.73	1526.89	2331414.39	714.61
9	1389.60	629.13	395806.92	294.44
10	1419.14	534.54	285736.24	250.17

Table 10: Mean, standard deviation, variance and confidence interval for the study of the search time for page elements located in places inconsistent with the generally accepted principles of universal design

Scr.	No-UD website Time (ms)			
	Mean	Std. Dev.	Variance	Conf. Int.
1	4831.41	3013.48	9081101.16	1410.35
2	13035.15	7692.57	59175683.37	3600.23
3	5547.35	3793.79	14392883.21	1775.55
4	6297.19	3552.58	12620859.60	1662.66
5	10219.49	5801.92	33662385.35	2715.38
6	5941.09	3477.95	12096150.57	1627.73
7	5273.11	3574.93	12780130.81	1673.11
8	5194.85	3470.65	12045432.56	1624.31
9	4633.13	3335.15	11123246.81	1560.89
10	5078.13	3158.88	9978538.93	1478.40

Table 11: Levene's test and t-test for pages with different arrangement of elements

	Test Levene's	T-test
Screen 1	4.51	$1.79 * 10^{-4}$
Screen 2	1.68	$2.23 * 10^{-7}$
Screen 3	0.31	$1.69 * 10^{-5}$
Screen 4	5.14	$1.68 * 10^{-6}$
Screen 5	0.31	$5.06 * 10^{-7}$
Screen 6	1.99	$2.94 * 10^{-6}$
Screen 7	1.26	$2.68 * 10^{-4}$
Screen 8	0.41	$1.02 * 10^{-3}$
Screen 9	0.83	$1.24 * 10^{-4}$
Screen 10	1.14	$9.48 * 10^{-6}$

Tables 12-14 show the results of the fixation duration studies for high and low contrast pages.

Table 12: Mean, standard deviation, variance, and Confidence Interval for high contrast page element search time study

Scr.	UD-enabled website Time (ms)			
	Mean	Std. Dev.	Variance	Conf. Int.
1	1804.18	1542.37	2378906.54	1103.34
2	3417.68	2637.59	6956881.90	1886.81
3	1310.69	377.96	142855.73	270.37
4	1454.97	408.92	167220.23	292.52
5	2162.55	1931.44	3730480.22	1381.67
6	1622.28	734.66	539732.52	525.54
7	1694.12	731.79	535527.33	523.49
8	2541.17	884.40	782171.50	632.66
9	3044.46	1824.92	3330349.51	1305.47
10	1431.41	512.44	262599.42	366.58

Table 13: Mean, standard deviation, variance, and Confidence Interval for low contrast page element search time study

Scr.	No-UD website Time (ms)			
	Mean	Std. Dev.	Variance	Conf. Int.
1	4976.25	4029.96	16240641.52	2882.86
2	5892.29	2638.92	6963903.92	1887.77
3	4827.24	3922.59	15386790.30	2806.05
4	4748.96	2811.99	7907340.59	2011.58
5	3866.21	2481.01	6155390.26	1774.80
6	4374.55	1979.56	3918657.82	1416.09

7	3829.01	1933.06	3736746.42	1382.83
8	6842.46	4329.81	18747217.79	3097.35
9	7629.29	4827.16	23301490.09	3453.14
10	6038.32	4671.37	21821701.92	3341.69

Table 14: Levene's test and t-test for pages with different contrasts

	Test Levene's	T-test
Screen 1	2.16	$3.20 * 10^{-2}$
Screen 2	0.01	$5.03 * 10^{-2}$
Screen 3	7.88	$1.13 * 10^{-2}$
Screen 4	10.55	$1.77 * 10^{-3}$
Screen 5	0.55	$1.04 * 10^{-1}$
Screen 6	8.01	$6.40 * 10^{-4}$
Screen 7	6.59	$4.29 * 10^{-3}$
Screen 8	12.82	$6.49 * 10^{-3}$
Screen 9	4.71	$1.16 * 10^{-2}$
Screen 10	5.18	$6.18 * 10^{-3}$

4.4. LUT survey

The WUP factor obtained using the LUT questionnaire were subjected to statistical analysis, i.e. calculation of the mean, standard deviation, variance and Levene's test. The results are presented in Table 15.

Table 15: WUP coefficients for LUT questionnaires

Participant	UD-enabled service WUP score	Non-UD service WUP score
1	4.88	3.13
2	4.89	3.46
3	4.69	2.22
4	4.80	3.19
5	4.73	2.49
6	4.98	3.56
7	4.78	2.39
8	4.67	3.43
9	4.69	3.25
10	4.83	3.42
Mean	4.79	3.05
Std. Dev.	0.01	0.24
Variance	0.11	0.49
Test Levene	6.39	

4.5. WAVE Tool

The results of the analysis conducted by the WAVE tool are presented in Table 16. The results were obtained with the use of a web browser plug-in and placed in the table for comparison.

Table 16: WAVE analysis results

Wave category	Sub-category	No. of errors site without UD	No. of errors site with UD
Errors	Missing alternative text	80	0
	Linked image missing alternative text	10	0
	Missing form label	2	5
	Empty button	22	1
	Empty link	2	3
	TOTAL	116	9
Contrast Errors	Very low contrast	335	3
	TOTAL	335	3
Alerts	Select missing label	0	1
	Redundant alternative text	50	2
	A nearby image has the same alternative text	5	0
	Skipped heading level	3	0
	Possible heading	3	0
	Redundant link	129	0
	Noscript element	2	1
	Very small text	38	1
	Redundant title text	6	0
	Accesskey	0	1
	Tabindex	0	1
	TOTAL	236	7
Features	Alternative text	8	0
	Null or empty alternative text	36	10
	Linked image with alternative text	110	2
	Form label	2	0
	Language	1	1
	Skip link	0	2
	Skip link target	0	1
	TOTAL	157	16
Structural Elements	Heading 1	1	2
	Heading 2	13	13
	Heading 3	17	0
	Heading 5	1	0
	Heading 6	4	0
	Unordered list	585	5
	Inline frame	4	0
	Header	1	1
	Navigation	8	1
	Main content	1	1
	Footer	1	1
	TOTAL	636	24
ARIA	ARIA	4	10

	ARIA label	4	18
	ARIA tabindex	6	9
	ARIA alert or live region	1	0
	ARIA hidden	31	6
	TOTAL	46	43

5. Discussion and conclusions

The purpose of this article was to perform a comparative analysis of graphical interfaces of two websites. Literature review and a comparative analysis on the basis of the obtained research results allowed for an unambiguous conclusion of the hypotheses: “Higher interface contrast affects the visibility and speed of searching for individual elements of user interface” and “The arrangement of interface elements has a significant impact on navigating the website”.

Using the WAVE Evaluation Tool, errors of both services were determined in three categories: contrast errors, page element structure and ARIA. Compared to the implemented application meeting the principles of universal design, the Empik website had many more errors from the above-mentioned categories what can conduct to lower comfort of using the website.

Based on the conducted eye tracking study, both hypotheses "Higher interface contrast affects the visibility and speed of searching for individual elements" and "The arrangement of interface elements has a significant impact on the effectiveness of navigating the website" have been confirmed. The purpose of the first set of scenarios was to test the time needed to locate certain elements on the interfaces of low and high contrast pages. By analyzing the results from tables 12-14, it can be concluded that the study participants needed an average of 2 times less time to find all the components on the high contrast page, while maintaining the average number of fixations twice as low. The search times for both the low and high contrast website were similar, but this was due to the fact that the elements were located in places compliant with the generally accepted principles of web application design, so that the user, after reading the command, intuitively directed his eyes, regardless of the contrast, in a given area of the interface. The second set of scenarios was responsible for determining the time needed to find elements placed in places compatible and inconsistent with the generally accepted principles of universal design for this type of pages. Based on the results from tables 9-11, it can be seen that the study participants took an average of almost 4 times more time to locate all components located in places that do not comply with generally accepted design standards. In this case, the average number of fixations was three times higher. Similarly, to the first set of scenarios, some users searched for both sets of items at a similar time, which could be caused by their proficiency in navigating pages or by accidentally finding the item they are looking for.

Based on the analysis of the number of fixations for the study of the search time of page elements located in places compliant and incompliant with the generally accepted principles of universal design from tables 3 and 4, it can be clearly stated that the samples have a normal distribution. Moreover, the data in table 5 indicate that the samples have a homogeneous variance and are statistically significantly different. Similar conclusions were drawn when analyzing the number of fixations for pages with high and low contrast (table 6 – 7). The tested samples have a normal distribution, homogeneous variance and in 9 out of 10 cases they are statistically significantly different.

When analyzing the fixation duration data for different arrangement of elements (table 11), it can be stated that all samples have homogeneous variance and are statistically significantly different. Similar conclusions were drawn when analyzing the fixation duration for pages with different contrast (table 14). The tested samples in 9 out of 10 cases have a homogeneous variance and are statistically significantly different.

The study conducted by filling in a subjective questionnaire confirmed that the website designed in accordance with the principles of universal design is more user-friendly. The study participants rated the page that fulfilled the principles of universal design 57% higher than the page that did not meet the principles of universal design (table 15). The biggest difference in the survey results was observed for the area related to the choice of colors and the structure of the website. The questions in this section were rated higher by 1.875 on average. The smallest differences were in the area of text, nomenclature and labels and amounted to 1.64 on average.

When analyzing the available literature, it was noticed that many professionally designed websites have significant accessibility errors. They do not have the ability to change the contrast, change the size of the elements and do not introduce new technologies. In addition, WAVE has been found to be the best tool for developers to examine web pages for WCAG 2.0 availability. Other tools do not have the ability to accurately diagnose page problems in all three main categories: contrast errors, page element structure, and ARIA.

The conducted analysis confirmed that taking into account the principles of universal design when creating websites improves their usability, transparency, intuitiveness and accessibility. The application becomes available to a wider audience.

Literature

[1] C. Duarte, I. Matos, J. Vicente, A. Salvado, C. M. Duarte, L. Carriço, Development Technologies Impact in Web Accessibility, the 13th Web for All Conference (2016) 1–4, https://doi.org/10.1145/2899475.2899498.

[2] M. Eyadat, D. Fisher, Web accessibility in information systems, International Journal of Web Information Systems 3(4) (2007) 363–377, https://doi.org/10.1108/17440080710848134.

[3] M. Ferati, B. Vogel, Accessibility in Web Development Courses: A Case Study, Informatics 7(1) (2020) 8, https://doi.org/10.3390/informatics7010008.

[4] M. Providenti, R. Zai, Web accessibility at Kentucky's academic libraries, Library Hi Tech 25(4) (2007) 494–508, https://doi.org/10.1108/07378830710840455.

[5] P. Panchekha, A. T. Geller, M. D. Ernst, T. Zachary, K. Shoaib, Verifying That Web Pages Have Accessible Layout, 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (2018) 1–14, https://doi.org/10.1145/3192366.3192407.

[6] E. M. Pivetta, C. Flor, D. S. Saito, V. R. Ulbricht, Analysis of an Automatic Accessibility Evaluator to Validate a Virtual and Authenticated Environment, International Journal of Advanced Computer Science and Applications 4(4) (2013) 15–22, https://doi.org/10.14569/IJACSA.2013.040403.

[7] I. V. Ramakrishnan, V. Ashok, S. M. Billah, Non-visual Web Browsing: Beyond Web Accessibility, Universal Access in Human–Computer Interaction 10278 (2017) 322–334, https://doi.org/10.1007/978-3-319-58703-5_24.

[8] N. Kesswani, S. Kumar, Accessibility analysis of websites of educational institutions, Perspectives in Science 8 (2016) 210–212, https://doi.org/10.1016/j.pisc.2016.04.031.

[9] Serwis internetowy sklepu empik, https://www.empik.com/, [01.02.2022].

[10] Sklep internetowy spełniający zasady projektowania uniwersalnego, http://magister.cba.pl/, [01.02.2022].

[11] Kompendium wiedzy o testach oraz źródło nowin ze świata testowania, https://testerzy.pl/baza-wiedzy/artykuly/projektowanie-uniwersalne, [26.01.2022].

[12] Walidator WAVE, https://wave.webaim.org/, 1.02.2022].

[13] Gazepoint GP3 HD Eye Tracker, https://www.gazept.com/product/gp3hd/, [01.06.2022].

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

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 323–329
Received: 29 June 2022
Accepted: 29 August 2022

Perfomance analysis of libraries for testing web applications on the ASP.NET Core platform

Analiza wydajności bibliotek do testowania aplikacji internetowych na platformie ASP.NET Core

Karol Niedziela*, Jakub Nieradko

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

Abstract
This paper conducts a performance analysis of three libraries: XUnit, NUnit, MSTest, aiming to compare the time performance. The performance was checked using load test, synchronous and asynchronous tests. The synchronous and asynchronous tests were divided into groups of 10, 25, 50 and 100 test cases. The tests were carried out using an in-house project written on the ASP.NET Core platform.

Keywords: software engineering; unit tests; performance; ASP.NET Core; C#

Streszczenie
W artykule została przeprowadzona analiza wydajności trzech bibliotek: XUnit, NUnit, MSTest, mająca na celu porównanie wydajności czasowej. Wydajność została sprawdzona przy wykorzystaniu testu obciążeniowego, testów synchronicznych oraz asynchronicznych. Testy synchroniczne oraz asynchroniczne zostały podzielone na grupy po 10, 25, 50 oraz 100 przypadków testowych. Dla każdej grupy zostało wykonane po trzydzieści pomiarów czasowych. Badania zostały wykonane przy pomocy autorskiego projektu napisanego na platformie ASP.NET Core.

Słowa kluczowe: inżynieria oprogramowania; testy jednostkowe; wydajność; ASP.NET Core; C#

*Corresponding author
Email address: **karol.niedziela@company.com** (K. Niedziela)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Na przestrzeni lat oprogramowanie komputerowe ewoluowało i zmieniało się wraz z rozwojem technologii. Obecny rynek IT oferuje coraz bardziej rozbudowane systemy informatyczne, których rozbudowa i utrzymanie wymaga odpowiednich narzędzi umożliwiających unikania błędów, pojawiających się w trakcie użytkowania aplikacji przez klientów. Wraz z rosnącym zapotrzebowaniem na skalowalne oprogramowania, powstało wiele technik, które miały zapewnić kod o wysokiej jakości i małej liczbie błędów. W tym celu zostały opracowane narzędzia umożliwiające sprawdzanie aplikacji bez konieczności manualnego testowania działania logiki kodu, które dotychczas było pierwszym etapem weryfikacji działania systemu. Jedną z tych technik jest pisanie testów jednostkowych [1].

Testowanie jednostkowe jest procesem rozwoju oprogramowania, w którym najmniejsze testowalne części aplikacji, zwane jednostkami, są indywidualnie i niezależnie sprawdzane pod kątem poprawności działania. Ta metoda testowania jest wykorzystywana podczas procesu tworzenia oprogramowania przez programistów. Test jednostkowy składa się zazwyczaj z trzech etapów, zdefiniowanych przez zasadę AAA (ang. Arrange – Act – Assert). W pierwszym etapie (ang. Arrange) przygotowywane są wszystkie dane wejściowe oraz warunki wstępne. W następnym etapie (ang. Act) zostaje wywołana metoda, która aktualnie jest testowana. W ostatnim etapie (ang. Assert) zostaje sprawdzona zgodność pomiędzy zwróconymi, a oczekiwanymi wartościami.

Głównym celem przeprowadzania testów jest sprawdzenie poprawności działania określonych funkcjonalności systemu. Aby testy były skuteczne, powinny być przeprowadzane głównie w miejscach występowania potencjalnych błędów oraz w miejscach, gdzie logika biznesowa jest złożona. Testowanie jednostkowe jest ważnym krokiem w procesie rozwoju oprogramowania, ponieważ jeśli jest wykonane poprawnie, pozwala wykryć wczesne błędy w kodzie, które mogą być trudniejsze do znalezienia w późniejszych etapach testowania [2]. Tak wykonane testy pozwalają uniknąć niezadowolenia klientów oraz użytkowników. Poza tym, są źródłem informacji o działaniu aplikacji dla osób pracujących aktualnie nad systemem oraz dla przyszłych programistów. Poprzez użycie narzędzi przeznaczonych do tworzenia testów, czas poświęcony na ich pisanie zostaje znacznie skrócony i ułatwia poprawne identyfikowanie przypadków testowych.

Celem pracy jest analiza wydajności bibliotek do testowania aplikacji internetowych na platformie ASP.NET Core. Celem analizy jest wskazanie czy istnieje biblioteka osiągająca znacznie lepsze rezultaty czasowe w różnych typach badań. Analizę przeprowadzono z wykorzystaniem aplikacji internetowej napisanej przy użyciu platformy .NET w wersji 6.0 oraz języka C# w wersji 10. W tym celu zbadano trzy najpopularniejsze biblioteki: XUnit, NUnit i MSTest. Zostały one porównane na podstawie szybkości czasu wykona-

nia testu obciążeniowego, testów synchronicznych oraz asynchronicznych. Przeprowadzone badania pozwolą na poznanie różnic pomiędzy wymienionymi bibliotekami. Przeprowadzona analiza zapoznaje z metodyką pisania testów oraz umożliwi wybór odpowiedniego narzędzia do ich implementacji. Otrzymane wyniki pozwolą wskazać najszybszą bibliotekę w badanych obszarach.

2. Przegląd literatury

Została wykonana analiza dostępnych materiałów badawczych przed przeprowadzeniem badań, które są tematem artykułu. Dotychczas opublikowano niewiele prac poświęconych analizie porównawczej bibliotek. Jednym z artykułów poruszających tematykę jest praca z Journal of Computer Science Institute pod tytułem "Porównanie wybranych narzędzi do przeprowadzenia testów jednostkowych" [3]. W artykule analizie zostały poddane trzy biblioteki: MSTest, NUnit i XUnit. Celem analizy było porównanie szybkości wykonywania testów w sposób szeregowy oraz równoległy. Testy zostały zaimplementowane dla aplikacji imitującej działanie internetowego konta bankowego. W pracy zostały krótko opisane wykorzystane biblioteki oraz pojęcia ściśle związane z testowaniem oprogramowania. Aby porównać szybkość działania każdej biblioteki została przetestowana funkcjonalność odpowiadająca za utworzenie stu tysięcy kont bankowych, a następnie dodanie ich do kolekcji. Najlepszy średni czas uzyskała biblioteka MSTest. Na tle przeprowadzonych badań najlepszą biblioteką okazała się biblioteka MSTest, następnie NUnit, natomiast najgorsze wyniki uzyskała biblioteka XUnit.

Poza wymienionym artykułem, większość prac skupia się na procesie wytwarzania oprogramowania, gdzie jednym z filarów są testy jednostkowe oraz wpływ testów jednostkowych na jakość kodu. Na Uniwersytecie Technologicznym Chalmers zostało przeprowadzone badanie [4], które miało odpowiedzieć na dwa pytania badawcze. Pierwsze pytanie brzmiało "W jakim stopniu pokrycie testami jednostkowymi jest skorelowane z liczbą błędów?". Drugie pytanie było związane z korelacją między pokryciem kodu testami jednostkowymi, a postrzeganą jakością kodu. Badanie składało się z dwóch niezależnych części. Pierwszą częścią była ankieta przeprowadzona na dwóch firmach w Szwecji oraz trzech przedsiębiorstwach i uniwersytecie w Brazylii. Drugą częścią było studium przypadku. Ankieta została przeprowadzona na 241 członkach zespołów programistów, gdzie nie było jasno zdefiniowanej procentowej wartości, jaką powinien być pokryty kod. W związku z tym, testy jednostkowe były pisane w różnych ilościach dla różnych części systemu. Ankietowani otrzymali do wypełnienia trzyczęściową ankietę. Pierwsza i druga część ankiety obejmowała sposób implementacji przypadku testowego względem głównej testowanej funkcjonalności. Trzecia część dotyczyła uruchamiania testów po zakończeniu zadania oraz przed integracją danej części systemu. Odpowiedzi udzieliło 201 osób. Studium przypadku obejmowało sprawdzenie pokrycia kodu w plikach przy użyciu narzędzia do pomiaru procentowego pokrycia kodu. Dodatkowo sprawdzono liczbę błędów kodu z małym procentowym pokryciem, wykorzystując zgłoszenia o błędach w systemie kontroli wersji. Na podstawie uzyskanych wyników w odniesieniu do pierwszego pytania badawczego nie znaleziono korelacji. Natomiast dla drugiego pytania badawczego korelacja była niska. Pozwala to stwierdzić, że jakość systemu nie musi być niska, jeśli pokrycie kodu testami jednostkowymi nie jest wysokie. Kolejna pozycja porusza tematykę mocnych i słabych stron stosowania testów jednostkowych oraz przyczynę ich pisania [5]. Badanie zostało przeprowadzono z wykorzystaniem ankiety przygotowanej dla kilku firm. Głównym powodem pisania testów jednostkowych były wymagania zewnętrzne. Testy miały służyć jako dokumentacja techniczna systemu. Innym częstym powodem było stosowanie przez organizację metodyki zwinnej (ang. Agile), która zakłada pisanie testów jednostkowych. Jako wady wskazano brak miar pozwalających na walidację użyteczności testów jednostkowych oraz trudność w wybraniu obszarów systemu, które powinny być testowane. Kolejną pozycją jest pytanie zadane na najpopularniejszym forum programistycznym StackOverflow. Pytanie brzmiało „Jakie są dobre sposoby, aby przekonać sceptycznych programistów w zespole o wartości testów jednostkowych?" [6]. Najczęściej pojawiającymi się odpowiedziami były:

- możliwość szybkiego sprawdzenia dużych zmian w kodzie i zapewnienie, że zmieniony lub dopisany kod działa prawidłowo,
- uzyskanie kodu o lepszej jakości,
- uzyskanie wizualnej informacji zwrotnej o poprawności działania (testy, które uzyskały zakładany rezultat są oznaczone zielonym kolorem),
- uzyskanie dokumentacji oraz przykładów, za co dany kod jest odpowiedzialny,
- zmniejszenie liczby błędów.

Następne badanie pokazuje, że przeprowadzenie testów zapewnia skalowalne i niepodatne na błędy oprogramowanie [7]. Porusza tematykę metodyki TDD (ang. Test Driven Development), powstałej wraz ze wzrostem popularności implementacji warstwy testowej i rozwojem procesu wytwarzania oprogramowania. Metodyka ta odwraca klasyczny model, gdzie najpierw zostaje napisany kod, a następnie zostaje on przetestowany. Dzięki zastosowaniu TDD, kod jest pokryty w 100% przez testy jednostkowe, co pozwala zapewnić wysoką jakość i minimalną liczbę błędów w samej logice kodu.

Dodatkowo istnieje kilka pozycji literaturowych opisujących sposób implementacji testów jednostkowych. Według książki Vladimira Khorikova [8], test jednostkowy jest zautomatyzowanym testem, które ma za zadanie weryfikację małej ilości kodu w szybkim czasie. Kolejną pozycją jest książka zawierająca część teoretyczną oraz część praktyczną [9]. W części teoretycznej wytłumaczono, czym jest testowanie jednostkowe, jakie plusy płyną z pisania testów jednostkowych i jaki wpływ mają one na system, ale także i programistów. W części praktycznej omówiono podstawy różnych bibliotek. Książka oferuje także instrukcje krok po

kroku w zakresie podstawowego tworzenia testów jednostkowych. Zawiera przykłady kodu z języków programistycznych takich jak Java, C++ oraz C#.

3. Badane biblioteki

W poniższym rozdziale zostały opisane analizowane biblioteki.

3.1. NUnit

NUnit jest open-source'ową biblioteką przeniesioną z JUnit [10]. Najnowsza wersja Nunit to NUnit3, która została przepisana z wieloma nowymi funkcjami i posiada wsparcie dla wszystkich platform .NET. Biblioteka NUnit jest licencjonowana na zasadach licencji MIT. Wcześniejsze wersje wydania korzystały z licencji NUnit. Obie licencje pozwalają na używanie tej biblioteki w komercyjnych jak i darmowych aplikacjach. W momencie pisania pracy, biblioteka NUnit została pobrana ponad 174 milionów razy. W momencie pisania pracy, na StackOverflow było około 26000 pytań oznaczonych jako NUnit.

3.2. XUnit

XUnit jest darmową, open-source'ową biblioteką do przeprowadzania testów jednostkowych [11]. Biblioteka jest tworzona przez społeczność. Została napisana przez twórcę jednej z wersji biblioteki NUnit. Spośród trzech analizowanych bibliotek jest najnowsza i najbardziej łatwa do rozbudowy. Biblioteka jest licencjonowana na zasadach licencji Apache 2. W momencie pisania pracy, biblioteka XUnit została pobrana ponad 200 milionów razy. Do tej pory na StackOverflow znajduje się około 1000 pytań oznaczonych jako XUnit.

3.3. MSTest

MSTest jest domyślną biblioteką do pisania testów, która jest dostarczana wraz z Visual Studio. Biblioteka jest wieloplatformowa, co umożliwia wykorzystywanie jej dla wszystkich platform .NET [12]. Pierwsza wersja MSTest nie była typu open-source. Jednakże wraz z wyjściem wersji drugiej, biblioteka jest typu open-source. Kod jest dostępny na platformie GitHub i jest licencjonowana na zasadach licencji MIT. W momencie pisania pracy, biblioteka MSTest została pobrana ponad 130 milionów razy. W momencie pisania pracy, na StackOverflow jest około 11000 pytań oznaczonych jako MSTest.

4. Przebieg badań

W poniższym rozdziale zostało opisane środowisko testowe wraz z użytymi aplikacjami oraz narzędziami. W celu porównania wydajności zostały zmierzone czasy wykonania testu wydajnościowego, testów synchronicznych oraz asynchronicznych.

4.1. Środowisko testowe

Przeprowadzone badania zostały wykonane przy pomocy urządzeń, których parametry zostały przedstawione w Tabeli 1.

Tabela 1: Komponenty środowiska testowego

Komponent	Komputer
System operacyjny	Windows 10, 64 bit
Procesor	Intel Core i5-10400
Dysk SSD	ADATA 512GB M.2 PCIe
Pamięć RAM	16 GB

4.2. Aplikacja testowa

Do przeprowadzenia badań została przygotowana aplikacja internetowa, dla której zostały wykonane testy jednostkowe. Główną funkcjonalnością aplikacji jest wsparcie zarządzania codziennymi potrzebami domowników. Aplikacja umożliwia współdzielenie informacji o listach zakupów, terminach rachunków do zapłacenia. Projekt aplikacji został podzielony na cztery warstwy, zgodnie z założeniami dotyczące czystej architektury. Poszczególne warstwy to:
• domenowa,
• aplikacji,
• infrastruktury,
• API (Application Programming Interface – interfejs programistyczny aplikacji).
Aplikacja została napisana w języku C# w wersji 10 oraz bazy danych MySQL. W Tabeli 2 zostały przedstawione użyte narzędzia do przeprowadzenia badań.

Tabela 2: Narzędzia wykorzystane na środowisku testowym

Narzędzie	Wersja
Język	10.0
.NET 6.0	6.0
Baza danych	MySQL 8.0.23
Visual studio	2022 Enterprise
XUnit	2.4.1
NUnit	3.13.2
MSTest	16.11.0
Shoudly	4.0.3

4.3. Test obciążeniowy

W celu porównania wydajności, pierwszym badaniem było przygotowanie testu, wymagającego dłuższego czasu wykonywania. W każdej z badanych bibliotek został przygotowany test jednostkowy dla metody, która odpowiada za dodanie do listy produktów określonej liczby elementów, w tym przypadku 5000. Dla każdej biblioteki test został uruchomiony trzydziestokrotnie, pomijając pierwsze uruchomienie. Dla każdej biblioteki został obliczony średni czas wykonania testów oraz zostało obliczone odchylenie standardowe.

4.4. Testy synchroniczne

Badanie miało na celu zmierzenie czasów wykonywania testów synchronicznych. Synchronicznie wykonywany kod oznacza, że linie kodu są wykonywane po kolei i dopiero po zakończeniu jednej linii następuje odblokowanie następnej. Innymi słowy, aby przejść do na-

stępnej linii, trzeba poczekać na zakończenie poprzedniej. Badanie zostało podzielone na grupy 10, 25, 50 i 100 testów dla każdej z trzech badanych bibliotek. Dla każdej grupy przypadków testowych zostało przeprowadzonych po 30 pomiarów czasowych.

4.5. Testy asynchroniczne

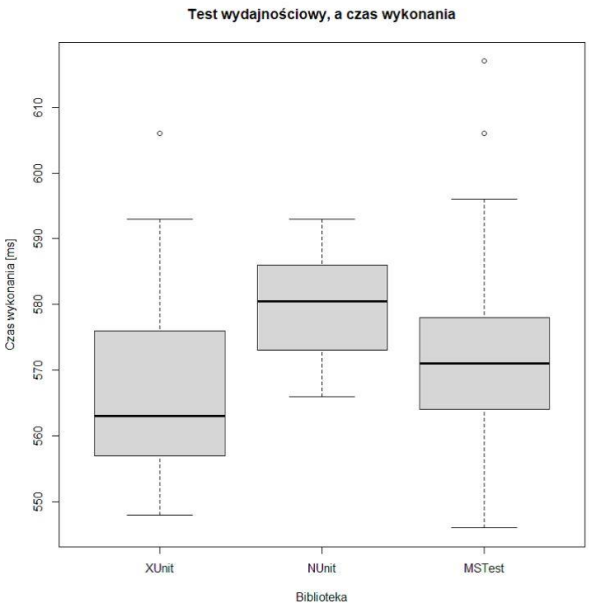
Następnym przeprowadzonym badaniem było wykonanie testów asynchronicznych i zmierzenie czasów ich wykonywania. Programowanie asynchroniczne to model programowania, w którym operacje wykonywane są w sposób niesekwencyjny. Model ten jest przeciwieństwem programowanie synchronicznego. W takim modelu operacje wymagającego czasu nie blokują programu, a kontynuują jego wykonywanie. Po zakończeniu operacji sukcesem lub porażką, wywoływane jest zdarzenie, które sygnalizuje, że wynik jest gotowy do wykonania na nim kolejnych kroków. Bez programowania asynchronicznego użytkownik miałby zablokowany graficzny interfejs. Dzięki programowaniu asynchronicznemu użytkownik może wykonywać swoje zadania w aplikacji, podczas gdy procesy działają w tle, co zwiększa komfort użytkowania. Przykładem problemu może być pobieranie danych z bazy danych zawierającej miliony rekordów. Aby zbadać jak radzą sobie badane biblioteki z kodem wykonywanym asynchronicznie, zostały przygotowane testy w grupach po 10, 25, 50 oraz 100 testów dla każdej z trzech badanych bibliotek. Dla każdej grupy przypadków testowych zostało przeprowadzonych po trzydzieści pomiarów czasowych.

5. Wyniki i analiza badań

W poniższym rozdziale zostały przedstawione wyniki przeprowadzonych badań opisanych w rozdziale 4.

5.1. Test obciążeniowy

Otrzymane pomiary zostały przedstawione na Rysunku 1, wykorzystując wykresy pudełkowe.



Rysunek 1: Wykresy pudełkowe dla testu obciążeniowego.

Ponadto została obliczona średnia czasu wykonywania testu oraz odchylenie standardowe. W Tabeli 3 zostały przedstawione wyniki.

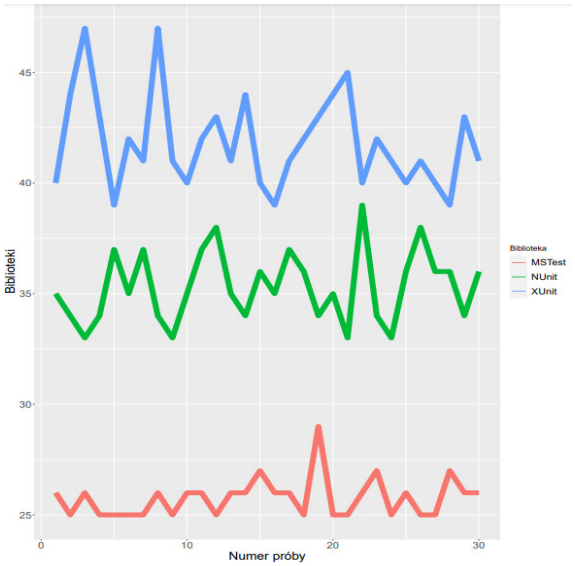
Tabela 3: Średnia oraz odchylenie standardowe testu

Biblioteka	Średni czas [ms]	Odchylenie standar-
XUnit	568,17	14,77
NUnit	579,77	7,85
MSTest	573,43	14.58

Na podstawie otrzymanych rezultatów można zauważyć, że trzy badane biblioteki wykonały test w podobnym czasie. Jednak najszybszy średni czas uzyskała biblioteka XUnit, natomiast najgorszy średni czas biblioteka NUnit. W przypadku odchylenia standardowego najmniejsze odchylenia uzyskała biblioteka NUnit, a najwyższe odchylenie biblioteka XUnit. Pomimo wyników różniących się maksymalnie o kilkanaście milisekund oraz najlepszego średniego czasu wykonania biblioteki XUnit. Biblioteka NUnit ma najmniejsze odchylenie, co przy każdorazowym uruchamianiu testów na przykład podczas wdrażania zmian na konkretnym środowisku może przemawiać na korzyść tejże biblioteki.

5.2. Testy synchroniczne

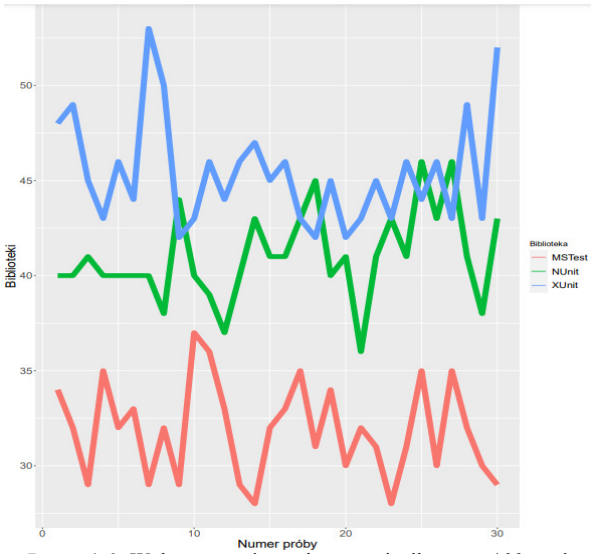
Na Rysunku 2 zostały przedstawione wykresy z wynikami pomiaru czasów dla grupy pięćdziesięciu testów synchronicznych, dla każdej z trzech badanych bibliotek w formie wykresu liniowego. Wykres koloru czerwonego odpowiada bibliotece MSTest, wykres koloru zielonego bibliotece NUnit, a wykres koloru niebieskiego bibliotece XUnit.



Rysunek 2: Wykresy czasów wykonywania dla grupy 50 testów synchronicznych.

Rysunek 3 przedstawia wykresy z wynikami pomiaru czasów dla grupy stu testów synchronicznych, dla każdej z trzech badanych bibliotek. Wykres koloru czerwonego odpowiada bibliotece MSTest, wykres koloru

zielonego bibliotece NUnit, a wykres koloru niebieskiego bibliotece XUnit.



Rysunek 3: Wykresy czasów wykonywania dla grupy 100 testów synchronicznych.

W Tabeli 4 został przedstawiony średni czas i odchylenie standardowe dla każdych grup testów oraz trzech badanych bibliotek.

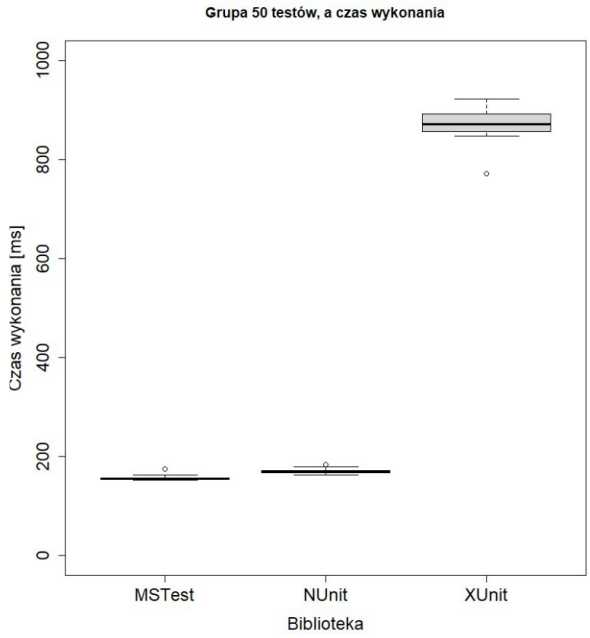
Tabela 4: Średnia oraz odchylenie standardowe dla różnych

Biblioteka	Grupa	Średni czas	Odchylenie
MSTest	10	13,17	0,46
	25	19,63	0,71
	50	25,77	0,90
	100	31,88	2,49
NUnit	10	21,07	0,25
	25	27,9	1,21
	50	35,3	1,62
	100	41,03	2,40
XUnit	10	21,07	0,52
	25	27,47	1,28
	50	41,83	2,13
	100	45,43	2,88

Na podstawie otrzymanych wyników dla poszczególnych bibliotek można wywnioskować, że jeśli testowany jest czysty kod języka wyniki są bardzo zbliżone. Najgorsze wyniki uzyskała biblioteka XUnit, gdzie jest dwukrotnie wolniejsza niż biblioteki NUnit oraz MSTest. Jedynie przy bardzo małej liczbie testów biblioteka XUnit uzyskuje podobne czasy do biblioteki NUnit. Dla każdej badanej grupy przypadków testowych najlepsze średnie rezultaty czasowe uzyskała biblioteka MSTest.

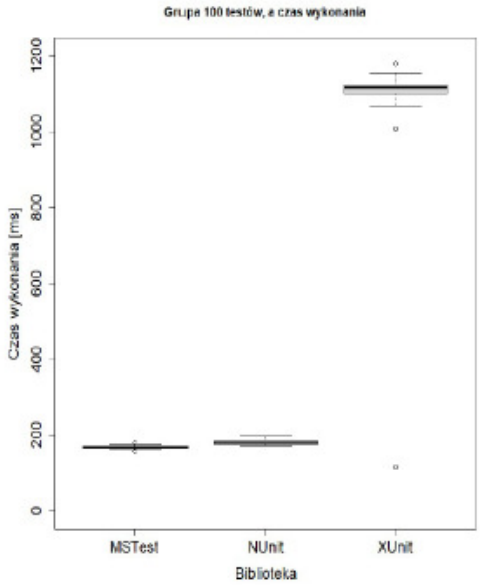
5.3. Testy asynchroniczne

Na Rysunku 4 został przedstawiony wykres z wynikami pomiaru czasów dla grupy pięćdziesięciu testów asynchronicznych.



Rysunek 4: Wykres pudełkowy dla grupy 100 testów asynchronicznych.

Na Rysunku 5 został przedstawiony wykres z wynikami pomiaru czasów dla grupy stu testów synchronicznych dla każdej z trzech badanych bibliotek.



Rysunek 5: Wykres pudełkowy dla grupy 100 testów asynchronicznych.

Dla każdej badanej grupy przypadków testowych zostały policzone średnia czasów wykonywania oraz odchylenie standardowe. Wyniki zostały przedstawione w Tabeli 5.

Tabela 5: Średnia oraz odchylenie standardowe dla różnych grup testów asynchronicznych

Biblioteka	Grupa	Średni czas	Odchylenie
MSTest	10	130,73	3,01
	25	138,57	3,15
	50	155,9	4,23
	100	167,03	4,59
NUnit	10	139,67	3,87
	25	150,27	4,38
	50	170,27	4,74
	100	180,43	6,00
XUnit	10	707,53	26,82
	25	767,83	26,47
	50	873,33	28,66
	100	1077,84	31,23

W porównaniu do wcześniej wykonanych testów synchronicznych widać dużą różnicę pomiędzy biblioteką XUnit, a biblioteką NUnit i MSTest. Dla każdej z badanych grup przypadków testowych biblioteka XUnit osiąga najwyższe średnie czasy oraz ma duże wahania pomiędzy wykonywaniem poszczególnych prób. Duża dysproporcja może być spowodowana samą budową biblioteki XUnit i jej różnicami względem dwóch pozostałych. Biblioteka NUnit została stworzona, wzorując się na bibliotece MSTest. Natomiast biblioteka XUnit stosuje inne podejście. Przykładowo biblioteka NUnit uruchamia wszystkie testy używając instancji tej samej klasy, natomiast biblioteka XUnit tworzy osobną instancję klasy dla każdego pojedynczego testu. Ponadto biblioteki NUnit i MSTest używają specjalnych znaczników, określających metodę służącą do zwalniania zasobów. W przypadku XUnit wykorzystywany jest interfejs wbudowany w język C#, który może wykonywać dodatkowe operacje, wpływające na czas wykonania testu. Pomimo dużej przyjemności z pisania testów w bibliotece XUnit, otrzymane rezultaty wskazują na dużą dysproporcję. W przeprowadzonym badaniu największą badaną grupą było 100 przypadków testowych. Dla porównania dla aplikacji używanych przez klientów, często liczba testów sięga kilku, a nawet kilkunastu tysięcy przypadków testowych. W związku z tym, używanie biblioteki XUnit może spowodować kilkusekundowe opóźnienie w czasie wykonywania testów z porównaniem do biblioteki NUnit i MSTest. Ma to duże znaczenie, ponieważ testy są najczęściej uruchamianie przy każdorazowym dodawaniu zmian w kodzie źródłowym.

5.4. Analiza wariancji

Kolejnym etapem było przeprowadzenie analizy wariancji. Analizie wariancji poddano test obciążeniowy, grupy testów synchronicznych i asynchronicznych. Z powodu otrzymania wartości p value znacząco odbiegającej od poziomu istotności przyjęto zapis < 0.05 dla wartości mniejszej od 0.05 i zapis > 0.05 dla wartości większej niż 0.05. Pierwszym etapem było wykonanie analizy wariancji dla testu obciążeniowego.

Tabela 6: Wyniki post-hoc dla grupy 50 testów asynchronicznych

Biblioteka	Biblioteka	Wartość p value
NUnit	XUnit	< 0.05
MSTest	XUnit	> 0.05
MSTest	NUnit	> 0.05

Na podstawie otrzymanych wyników z analizy wariancji można stwierdzić, że biblioteki MSTest i XUnit oraz MSTest i NUnit nie różnią się istotnie statystycznie. Kolejno została przeprowadzona analiza wariancji dla każdej grupy testów synchronicznych. W Tabeli 7 zostały przedstawione rezultaty dla grupy 50 testów synchronicznych.

Tabela 7: Wyniki post-hoc dla grupy 50 testów synchronicznych

Biblioteka	Biblioteka	Wartość p value
NUnit	XUnit	< 0.05
MSTest	XUnit	< 0.05
MSTest	NUnit	< 0.05

W Tabeli 8 zostały przedstawione rezultaty dla grupy 100 testów synchronicznych.

Tabela 8: Wyniki post-hoc dla grupy 100 testów synchronicznych

Biblioteka	Biblioteka	Wartość p value
NUnit	XUnit	< 0.05
MSTest	XUnit	< 0.05
MSTest	NUnit	< 0.05

Na podstawie otrzymanych wyników z analizy wariancji można stwierdzić, że tylko dla bibliotek NUnit i XUnit dla grupy 10 i 25 testów synchronicznych, biblioteki różnią się statystycznie istotnie. Dla pozostałych przypadków biblioteki nie różnią się istotnie statystycznie. Następnie została przeprowadzona analiza wariancji dla testów asynchronicznych. W Tabeli 9 zostały przedstawione rezultaty dla grupy 50 testów asynchronicznych.

Tabela 9: Wyniki post-hoc dla grupy 50 testów asynchronicznych

Biblioteka	Biblioteka	Wartość p value
NUnit	XUnit	< 0.05
MSTest	XUnit	< 0.05
MSTest	NUnit	< 0.05

W Tabeli 10 zostały przedstawione rezultaty dla grupy 100 testów asynchronicznych

Tabela 10: Wyniki post-hoc dla grupy 100 testów asynchronicznych

Biblioteka	Biblioteka	Wartość p value
NUnit	XUnit	< 0.05
MSTest	XUnit	< 0.05
MSTest	NUnit	< 0.05

Na podstawie otrzymanych wyników z analizy wariancji, we wszystkich przypadkach poza jednym, biblioteki różnią się istotnie statystycznie. Wyjątkiem jest grupa 10 testów asynchronicznych, gdzie biblioteki MSTest oraz NUnit nie różnią się istotnie statystycznie.

6. Podsumowanie

W celu porównania trzech najpopularniejszych bibliotek zostały wykonane pomiary czasów dla testu obciążeniowego, testów synchronicznych oraz asynchronicznych. Dla testu obciążeniowego zostało wykonanych po 30 pomiarów dla każdej z badanych bibliotek. Testy synchroniczne i asynchroniczne zostały podzielone na grupy po 10, 25, 50 oraz 100 testów. Dla każdej grupy zostało przeprowadzonych po 30 pomiarów. Dla przygotowanego testu obciążeniowego wszystkie badane biblioteki uzyskały przybliżone wyniki. Najlepszy średni czas uzyskała biblioteka XUnit.

W kolejnym etapie badań porównano czasy wykonania w grupach 10, 25, 50 i 100 testów synchronicznych. Wszystkie badane biblioteki w każdej grupie uzyskały niskie rezultaty. Jednak najlepszy średni czas w każdej grupie uzyskała biblioteka MSTest. Następnie zostały porównane czasy wykonywania testów asynchronicznych. Na podstawie przeprowadzonych badań można stwierdzić, że biblioteka XUnit osiąga znacznie gorsze wyniki czasowe w porównaniu do biblioteki NUnit oraz MSTest. Biblioteki NUnit i MSTest uzyskały bardziej zbliżone wyniki względem siebie. Jednakże najszybsza w badaniu przeprowadzonym dla testów asynchronicznych była biblioteka MSTest. Tak duża dysproporcja może wynikać z różnic w budowie biblioteki. Tworzenie nowej instancji dla każdego testu jednostkowego w przypadku biblioteki XUnit może mieć przełożenie w czasie wykonywania testów.

Czas wykonywania określonej grupy testów jest aspektem, który znacząco może wpłynąć na wydajność pracy programistów. W przeprowadzonych badaniach największą uruchamianą grupą za jednym razem była grupa 100 testów. Już dla tej grupy występowały kilkunasto lub kilkudziesięcio milisekundowe różnice. W aplikacjach komercyjnych liczba testów jest znacznie większa. Na podstawie przeprowadzonych badań najszybsza z badanych bibliotek jest biblioteka MSTest. Uzyskała najlepsze średnie czasy dla każdej grupy z przeprowadzonych testów synchronicznych i asynchronicznych.

Literatura

[1] J. Albahari, C# 9.0 in a nutshell: The definitive reference, O'Reilly Media, Sebastopol, 2021.

[2] R. Osherove, The Art of Unit Testing, Second Edition with examples in C#, Manning Publications, Shelter Island, 2013.

[3] P. Strzelecki, M. Skublewska-Paszkowska, Pórownanie wybrancych narzędzi do przeprowadzania testów jednostkowych, Journal of Computer Sciences Institute 9 (2018) 334-339.

[4] L. Gren, A. Vard, On the relation between unit testing and code quality, Proceedings of the 43rd Euro Micro Conference on Software Engineering and Advanced Applications (SEAA), IEEE Xplore (2017) 52-56.

[5] P. Runeson, A survey of unit testing practices IEEE software 23, 4 (2006) 22–29.

[6] Czy testowanie jednostkowe przynosi efekty?, https://stackoverflow.com/questions/67299/is-unit-testing-worth-the-effort, [21.05.2022].

[7] G. Sochacki, B. Pańczyk, Test-Driven Development jako narzędzie optymalizacji procesu wytwarzania oprogramowania na platformie JEE, Journal of Computer Sciences Institute, 4 (2017) 112-116.

[8] V. Khorikov, Unit testing principles, practices and pattterns, Simon and Schuster, New York, 2020.

[9] P. Hamill, Unit Test Frameworks: Tools for High-Quality Software Development, O'Reilly Media, Sebastopol, 2004.

[10] Dokumentacja XUnit, https://xunit.net/#documentation, [21.05.2022].

[11] Dokumentacja NUnit, https://docs.nunit.org/, [21.05.2022].

[12] Dokumentacja MSTest, https://docs.microsoft.com/en-us/dotnet/api/microsoft.visualstudio.testtools.unittesting?view=visualstudiosdk-2022, [21.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 330–336
Received: 20 June 2022
Accepted: 20 September 2022

Analysis of the ergonomics of e-commerce websites

Analiza ergonomii stron internetowych z branży e-commerce

Jarosław Chmal*, Monika Ptasińska, Maria Skublewska-Paszkowska

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

Abstract

The following paper includes research about ergonomics of e-commerce web applications. The main purpose of the experiment was to compare existing application of Morele.net shop and developed prototype of application using eyetracking examination and survey. The study carried out on a group of 40 students provided heat maps, scan paths, number of fixations and saccades, times to the first fixation in area of interest, task completion times, assessments of both applications in the form of WUP indicators. Based on the qualitative and quantitative analysis, conclusions were drawn confirming the hypothesis put forward in the work that there is an impact of ergonomic placement of navigation elements on the accessibility and usability of the application, as well as the time of performing tasks in it.

Keywords: ergonomics; usability; accessibility; eyetracking; universal design

Streszczenie

Niniejszy artykuł dotyczy badań związanych z ergonomią aplikacji internetowych z branży e-commerce. Głównym celem eksperymentu było porównanie istniejącej aplikacji sklepu Morele.net oraz opracowanego prototypu aplikacji z wykorzystaniem badania okulograficznego i ankiety. Z badań przeprowadzonych na grupie 40 studentów otrzymano mapy cieplne, ścieżki skanowania, liczby fiksacji, liczby sakkad, czasy do pierwszej fiksacji w obszarze zainteresowania, czasy wykonania zadań oraz oceny obu aplikacji w postaci wskaźnika WUP. Na podstawie analizy jakościowej i ilościowej wyciągnięto wnioski potwierdzające postawioną w pracy hipotezę o wpływie ergonomicznego rozmieszczenia elementów nawigacyjnych na dostępność i użyteczność aplikacji oraz czas wykonywania w niej zadań.

Słowa kluczowe: ergonomia; użyteczność; dostępność; okulografia; projektowanie uniwersalne

*Corresponding author

Email address: jaroslaw.chmal@pollub.edu.pl (J. Chmal)

©Published under Creative Common License (CC BY-SA v4.0)

1. Introduction

Ergonomics is a scientific discipline that deals with the optimization of the workplace and equipment in order to reduce the biological cost of work and increase its efficiency [1]. In the context of web applications, ergonomics refers to interfaces that are structured according to the user's characteristics. They should be intuitive, i.e. built in a way that does not require any prior preparation or training from a human. The ergonomics of the website interface is realized through features such as usability and accessibility. An important aspect is also the concept of Universal Design [2], which was born in 1983 from the previously known concepts of accessibility and barrier-free design. The originator was the architect Ron Mace, who argued that a design adapted for people with disabilities would also benefit the general public. Together with a group of architects, designers, engineers and environmental design researchers, Mace developed Seven Principles of Universal Design (UD).

The idea behind this approach is to design various types of products using innovative solutions that meet the requirements of various user groups and ensure equal and fair access. Over the years, universal design solutions have been used in the context of various areas of life, including web design.

The above-mentioned issues have been used in websites and web applications, including the design of online stores. The almost unlimited choice of products, bargain prices and convenience have made online shopping an indispensable part of many people's lives.

Due to the COVID-19 virus spread around the world, many people have decided to shop online for the first time in their lives. It was possible to conduct research on the behaviour of inexperienced users on the Internet [3]. Comparing the experiences of savvy and inexperienced consumers when shopping online can help retailers build satisfying e-commerce platforms with newbie digital consumers in mind. Inexperienced consumers have higher expectations of websites. If the website does not meet their requirements, they often give up the purchase.

In the sales industry, especially in online stores, visual information has a direct impact on consumers' purchasing decisions [4].

Thanks to technology such as eyetracking, and often also other auxiliary technologies and devices, web designers are able to check how the reception of a specific image affects not only the perception of a given product, but also what it evokes emotions in the user [5] [6]. This allows for the selection of graphics in such a way as to evoke specific feelings in the consumer, which translates into encouraging the user to buy the goods. It can therefore be concluded that cognitive aspects have a significant impact on the use of e-commerce websites by consumers [7].

Designing an ergonomic interface for an online store requires recognizing the elements that affect the user experience and the availability of the application. For this purpose, research methodology should be developed and conclusions on interface design should be gathered. The aim of the article is to verify the impact of ergonomic placement of navigation elements on the accessibility and usability of the application, as well as on the time of performing tasks in it. The e-commerce portal Morele.net will be compared with the prototype of the online store website, specially prepared, in accordance to universal design and features of usability and accessibility. An eye tracker and a LUT (Lublin University of Technology) survey were used for the comparison. The collected results were subjected to qualitative and quantitative analysis, as described in the discussion. The article has been summarized in the last chapter.

2. Literature review

Collecting scientific articles made it possible to deepen the knowledge of the factors that have the greatest impact on the usability and accessibility of websites.

In the article [4] research was conducted on a group of twenty students, ten women and ten men. The experiment consisted in recording participants' eye movement data after viewing a photo of a store shelf with hair shampoos, broken down by area of interest. The relationship between final consumer choice and time to first fixation in Area of Interest (AOI), duration of first AOI fixation, total duration of AOI visits, and number of AOI visits was analysed. Based on the data obtained, it was noted that the total duration of AOI visits and the number of AOI visits play a significant role in the final choice of consumers.

The conclusion of the research is that in the process of interaction with a product, customers are influenced by its location and category, and saccadic behaviour occurs frequently. Looking at a product for a long time or repeatedly correlates with the fact that customers are more likely to buy a given product, which is why it is so important to present the product in the most visually advantageous way.

The work [8] presents the operation of the algorithm for collecting and analysing data from the eye tracker in order to develop typical scanning paths (sequences of eye movements) of visual elements on websites. Forty users were asked to fill out short questionnaires to collect demographic data and then go through the selected websites twice.

First, the research was to find the elements specified in the experiment (in no more than 120 seconds), and then search for any product of their interest. The gender similarity of the results suggests that gender may cause some differences in shared scanning paths. However, the lengths of the common male and female group scan paths on the same pages for specific search tasks are mostly the same. Common scanning paths can be used to design or redesign websites to improve user experience. The authors-made algorithm can be beneficial for web designers to understand how users interact with web pages.

By using eye tracking technology, data can be obtained on how people feel and how participants use them in the process of interacting with the site. The collection of such data through a specially designed e-commerce website that uses popular templates is presented in the article [9]. The research group was selected on the basis of multilingualism, age, education and occupation, therefore twenty Northwest University students were selected equally from each gender. The research was based on eye tracking, collecting and analysing the duration of fixations and the fixation sequence. The results of the experiment show that during the template browsing process, eyesight was guided from top to bottom and from left to right. Due to the tendency of users to look from left to right, important attributes should be placed on the left side. It is also common to focus your eyesight on the upper, middle part of the page [10]. Therefore, when designing a website, this style and user's habits should be respected. Filling every centimeter of space should be avoided, because users perceive simple projects much better, and in addition, too much elements may overload the visual memory.

An equally important aspect in designing websites is their functionality and usability. Online stores are designed to make it easier for people to access an unlimited number of products from around the world in a quick and easy way. Consumers should have access to the product description and specifications before purchasing. It is very important to enable users to return to previously viewed products, as they rarely make a purchase during the first visit to the store [11].

Considering the increase in the volume of online sales and the growing proportion of elderly people in the population, efforts should be made to reduce the digital exclusion that they often experience. Reduced technical proficiency and impaired eyesight are the main factors contributing to problems with operating websites, including online shopping. Although older people are usually wary of online shopping, research [12] shows that each age group feels a certain risk. Its reduction is influenced by, among other things, buying products from larger, well-known brands and not storing your data on websites. Younger age groups have a much better experience of displaying proposed offers on websites, because they often shop online and browse products in search of new products, which is why it is interesting for them.

In order to understand the needs and expectations of users, and to optimize the design of the website, it is necessary to analyse the relationship between the website usability and user satisfaction [5]. In terms of usability, special attention should be paid to website layout issues, such as the navigation bar design and page layout, as well as to website performance indicators, including browser and error management issues.

User satisfaction is influenced by web design attributes, not the personal characteristics of users, such as gender, age, education, or Internet experience. When designing the layout of an e-commerce website, special attention should be paid to the placement and presentation of photos, as they will largely determine whether the consumer will make a purchase [2].

3. Work scope

Based on the literature research, the following thesis was formulated: "There is an impact of ergonomic placement of navigation elements on the accessibility and usability of the application, as well as the time of performing tasks in it".

3.1. Eye tracker

The eyetracking technology was used for the research. 40 students (4 females, 36 males) of the Lublin University of Technology in the field of Computer Science participated in the research. The subjects were aged 23 – 25, most of them wore glasses or contact lenses. The participants were divided into two independent groups. Each group was to perform the same tasks, but on two different e-commerce applications, respectively: Morele.net and on a prototype of a web application made in accordance with the principles of universal design. The participants' task was to follow the screen and perform specific activities (according to planned scenarios), including searching for a specific product on the page (exactly the same product was to be searched on each page).

The eyetracking metrics considered in the study are the number of fixations [4], answering the question of whether the tested interface element focuses the user's attention, as well as saccades [4] – rapid eye movements between consecutive fixations. The data obtained in the form of heat maps were also analysed. On their basis, it can be concluded which elements of the website interface are the most popular among users. It made it possible to find out where the participants direct their eyes by default, having the task of finding a specific navigation element on the page, and thus whether it is located where users expect to find it. Moreover, based on the defined areas of interest, the times to the first AOI fixation were obtained.

3.2. LUT survey

After the completion of the eye tracker study, participants were presented with a LUT survey [1] to be filled in, in which they had to answer 32 questions about the impressions related to the use of the application. Each question was answered on a five-point Likert scale to determine whether there were problems with the functionality of the application or not. On the basis of the answers, the WUP (Web Usability Points) indicators were calculated, which allowed to determine the subjective quality of the interface [13].

4. Experiment

The experiment was performed with the Gazepoint GP3 HD eye tracker [14] connected to an Acer Nitro 5 AN517-41-R48Y laptop with the following parameters: processor: AMD Ryzen 7 5800H (8 x 3.2GHz), 32GB DDR3 operating memory, Nvidia Geforce RTX 3060 6GB graphics card, 512GB SSD disk, 17.3" screen with 1920x1080 resolution, Windows 10 x64 Education.

The Gazepoint Control software was installed on the laptop, which was needed to carry out the eye tracker research. The test stand was located in the laboratory of the Department of Computer Science of the Lublin University of Technology. Before starting the study, the eye tracker had to be calibrated for each of the 40 participants. After successful calibration, participants proceeded to perform tasks designed to test the availability of content, taking into account the quality of the interface.

Figure 1: Test stand.

The research scenarios developed for the purpose of the research included the following tasks for the participants:

1. Locating the item that allows you to search for products in the application.
2. Locating the element that allows you to register / log in to the application.
3. Locating the category of smartphones and smartwatches.
4. Locating the item named smartphones / all smartphones.
5. Locating the element indicating the user's current location on the website.
6. Locating the element that allows you to change the way of displaying products on the list.
7. Locating the item that allows you to add the first smartphone on the list to the cart.
8. Locating the item that allows you to go to the cart.
9. Locating the item that allows you to remove the smartphone from the cart.
10. Locating all items that allow you to go to the previous subcategory.

The tasks were carried out using screenshots of individual, analogous subpages of both online stores. The user did not interact with the application, but his eyesight was monitored while displaying subsequent screenshots.

5. Results

The collected results from the eye tracker and the survey made it possible to compare the ergonomics of the two web applications.

The results of the tests of statistical significance (means/medians) showed that the differences between the two samples are statistically significant (Tables 2,3,4). Levene's test was used to determine the homogeneity of variance. In the case when the distribution of both samples was normal, the results of the Student's T-test were taken into account, otherwise the Mann-Whitney's test was used.

5.1. Eye tracker

Based on the research results, a quantitative analysis of selected eye tracking measures was conducted – time to first fixation in AOI, number of fixations, number of saccades and the time taken to complete each task.

The comparison of the times to the first fixation of the analyzed element for ten tasks for the application prototype and the Morele.net store (Table 1) showed that the average time for the prototype was more than two times smaller and averaged 1.12s ± 0.17s – for Morele.net this average was 2.77s ± 0.86s.

Table 1: Times to first fixation

No.	Application prototype time [*s*]	Morele.net store time [*s*]
1	1.30	2.92
2	1.07	2.76
3	1.35	2.61
4	1.12	1.74
5	0.89	2.53
6	1.16	5.00
7	1.27	2.54
8	0.99	2.75
9	0.87	2.65
10	1.19	2.17
Mean	1.12	2.77
St. Dev.	0.17	0.86
Var.	0.028	0.734
Conf. int.	0.012	0.322
Shapiro-Wilk Test	0.695	0.002
Mann-Whitney Test	1.083×10^{-5}	

Figure 2 shows the average fixation number, which is more than twice as high for the Morele.net application.

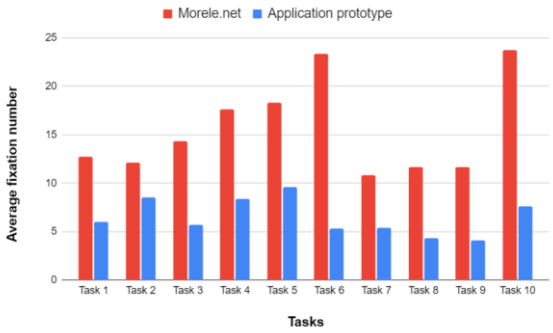


Figure 2: Average fixation number for tasks.

Table 2 presents the results of Levene, T-Student and Mann-Whitney tests calculated on the basis of the fixation numbers.

Table 2: Results of the Levene, T-student and Mann-Whitney tests for fixations number

No.	Levene Test	T-student Test	Mann-Whitney Test
1	0.0464	2.503×10^{-3}	9.234×10^{-4}
2	0.7124	9.507×10^{-2}	8.035×10^{-2}
3	0.0213	5.466×10^{-5}	5.102×10^{-5}
4	0.2958	4.539×10^{-3}	1.141×10^{-3}
5	0.6197	1.068×10^{-2}	4.720×10^{-4}
6	0.0145	4.369×10^{-8}	1.090×10^{-7}
7	0.0028	3.001×10^{-3}	6.364×10^{-3}
8	0.0071	3.336×10^{-4}	7.697×10^{-5}
9	0.0300	7.312×10^{-6}	2.435×10^{-5}
10	0.0072	3.684×10^{-5}	4.988×10^{-5}

Figure 3 presents average saccades number for each task in both applications. Based on the results, the Levene, T-Student and Mann-Whitney tests were performed (Table 3).

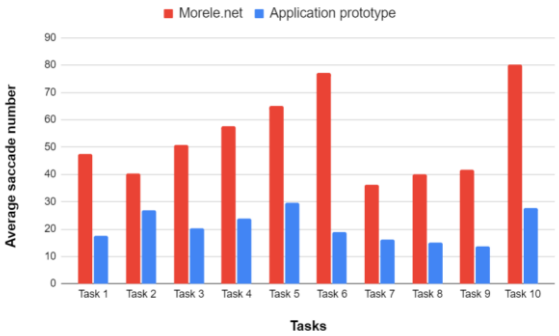


Figure 3: Average saccades number for tasks.

Table 3: Results of the Levene, T-student and Mann-Whitney tests for saccades number

No.	Levene Test	T-student Test	Mann-Whitney Test
1	0.0021	2.503×10^{-3}	3.051×10^{-5}
2	0.1428	8.327×10^{-2}	1.162×10^{-1}

3	0.0027	2.144×10^{-4}	3.061×10^{-5}
4	0.2798	7.039×10^{-3}	4.365×10^{-5}
5	0.0350	1.631×10^{-3}	3.928×10^{-4}
6	0.0001	3.228×10^{-7}	1.136×10^{-7}
7	0.0009	2.648×10^{-3}	4.668×10^{-3}
8	0.0078	1.050×10^{-3}	2.312×10^{-4}
9	0.0034	4.444×10^{-5}	3.212×10^{-6}
10	0.0001	1.268×10^{-5}	2.026×10^{-5}

A summary of the average time of completing tasks for both applications is presented in Figure 4. The prototype achieved significantly lower results. Table 4 shows the significance test results for the average task completion times.

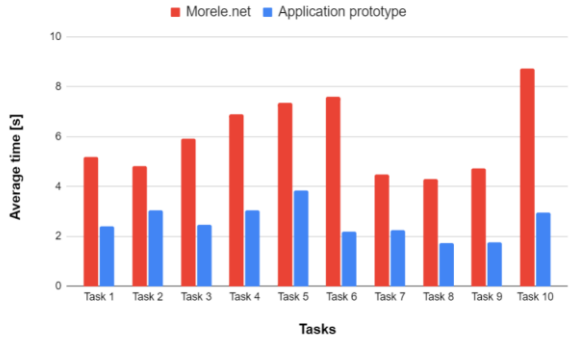


Figure 4: Average time of completing tasks.

Table 4: Results of the Levene, T-student and Mann-Whitney tests for time of completing tasks

No.	Levene Test	T-student Test	Mann-Whitney Test
1	0.0272	2.435×10^{-4}	4.909×10^{-6}
2	0.2906	1.865×10^{-2}	1.954×10^{-2}
3	0.0001	1.819×10^{-5}	2.884×10^{-6}
4	0.2653	2.721×10^{-4}	1.831×10^{-5}
5	0.5789	1.602×10^{-3}	9.105×10^{-5}
6	0.0385	5.157×10^{-8}	1.451×10^{-1}
7	0.0027	3.147×10^{-4}	3.722×10^{-4}
8	0.0333	3.804×10^{-5}	4.909×10^{-6}
9	0.0093	9.472×10^{-7}	2.317×10^{-8}
10	0.0052	8.119×10^{-7}	7.571×10^{-7}

Figures 5 and 6 present heat maps for one task performed on both applications interfaces. The colours orange, yellow and green meant less and less concentration of the respondents.

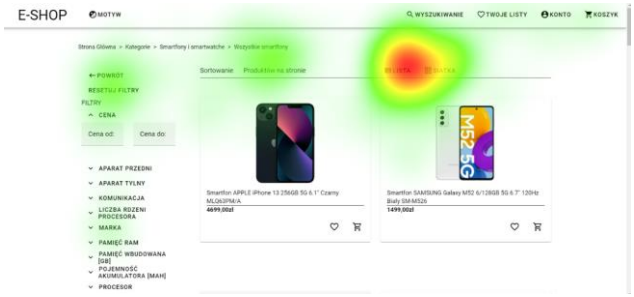


Figure 5: Heat map for application prototype for task 6.

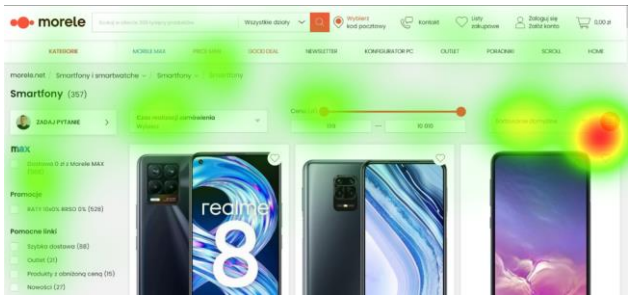


Figure 6: Heat map for application Morele.net for task 6.

Figures 7 and 8 show the scanning paths for task 8.

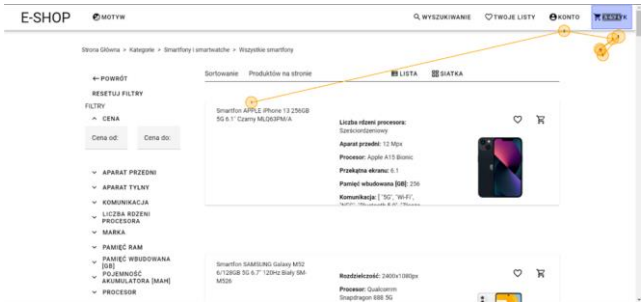


Figure 7: Scan path for application prototype for task 8.

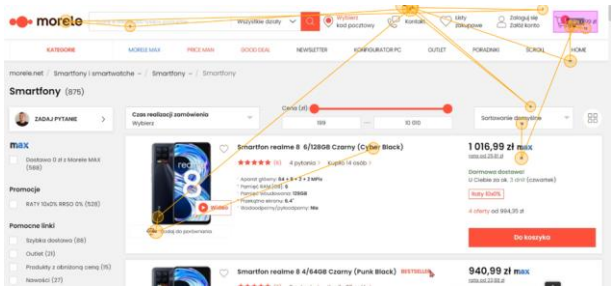


Figure 8: Scan path for application Morele.net for task 8.

The generated scan paths and heat maps allowed for a qualitative analysis.

5.2. LUT

Table 5 contains WUP indicators, which were calculated with the use of a dedicated formula [1], using the points obtained from the LUT questionnaire. Twenty participants in one group of the study evaluated the prototype of an application designed in accordance with the principles of universal design, and the other group

evaluated the application of the Morele.net store. The mean of the WUP for the prototype was 4.78, which means there were no usability issues or issues affecting the user experience. For Morele.net WUP score was 3.57, which may suggest the existence of single minor usability problems that could reduce the quality of working with the application (Figure 9).

Table 5: WUP indicators for LUT survey

Respondent	WUP – aplication prototype	WUP – Morele.net application
1	4.86	3.19
2	4.86	3.46
3	4.68	2.52
4	4.78	3.22
5	4.69	2.73
6	4.83	4.21
7	4.76	3.64
8	5.00	3.53
9	4.71	3.61
10	4.78	4.18
11	4.85	4.05
12	4.89	4.14
13	4.70	4.20
14	4.78	3.19
15	4.69	2.51
16	4.90	3.65
17	4.72	3.65
18	4.67	3.97
19	4.69	3.83
20	4.81	3.94

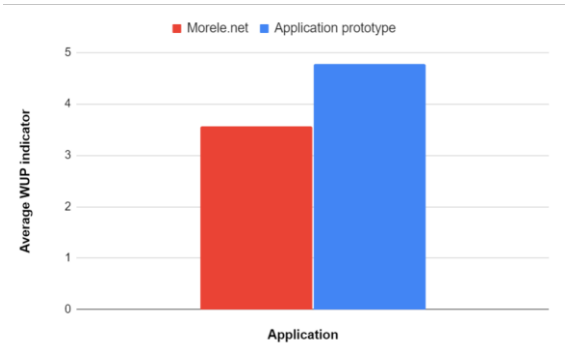


Figure 9: Average WUP indicator for both applications.

6. Discussion

The analysis of the results obtained with the eye tracker clearly showed that it was easier for users to find the elements they were looking for in the application supporting universal design. The application prototype, made in accordance with the principles of universal design, had lower number of fixations. The average number of fixations for the application prototype was lower by 58.4% than the average number of fixations for the Morele.net store (Figure 2). These results show that the application designed in a minimalistic, transparent manner and in accordance with the principles of universal design increases the usability. Users' attention was less distracted due to fewer items. The short time of searching for elements (time to first fixation) means that they are placed on the website in places that are intuitive for the user.

Similarly, in the article [6], based on the research on the number of fixations, it was found that a website rich in graphic and colour elements has a positive effect on the visual perception by users, but this translates into difficulties in searching for given elements.

In order to verify the search times, the times to the first fixation as well as the times of performing individual tasks were analysed. The comparison of the times to the first fixation of the analysed element (Table 1) for ten tasks for the application prototype and the Morele.net store showed that the average time to the first fixation for the prototype was 59.6% lower. On the other hand, in the case of task completion time (Figure 4), the time for the application supporting universal design was reduced by 57%.

When performing tasks for an application compliant with the principles of universal design, the average number of saccades was lower by 60.9% in relation to the average number of saccades on the Morele.net store website (Figure 3) [4]. It follows that users made fewer involuntary eye movements to find what they were looking for.

During the qualitative analysis, it was found that locating the item is successful for both applications. However, with Morele.net, the scanning path is longer – the user had to search longer to find the target (Figure 7, 8).

Certain patterns of behaviour are visible when searching for items. Similar conclusions were drawn in the article [9]. After performing the eye tracker tests, it was found that the first eye movement was focused on the upper left corner of the page. The typical behaviour of the study participants was to browse the page in order: top to bottom and left to right. The conclusion is that these areas are key when designing a website, because they can provide visitors with the most important content. Therefore, it is important for web designers to use this layout. This allows to work more efficiently with website or application.

The heat maps generated for individual tasks were also analysed. The red colour marks the area in which users focused their eyes most – there was the area of interest [4]. As can be seen in Figures 5 and 6, a much greater distraction of users occurs when performing a task on the Morele.net website.

The calculation of the WUP indicator (Table 5) on the basis of the data collected from the LUT survey and its analysis made it possible to conclude that universal design increases the usability of web applications. For the Morele.net application, the WUP amounted to an average of 3.57 and for the developed prototype: 4.78 (Figure 9), with the maximum possible result equal to 5.

7. Conclusion

Based on the analysis of data from the conducted research, it was confirmed that there is an impact of the ergonomic location of navigation elements on the availability and usability of the application, as well as the time of performing tasks in it. Website operations are performed much faster and without losing the user's attention. Better structure of the elements made it is also more accessible to electronic readers. Intuitive navigation and efficiency in using the website have a big impact on its usability.

From an ergonomic point of view, the web application should not contain unnecessary and distracting graphic elements, because although it positively affects the visual perception of users, they make navigation on the site difficult. The positive impact of universal design on the ergonomics of websites and internet applications is noticeable. When designing an e-commerce store, the functionality of the website is important, while maintaining its usability.

The research was conducted on a group of IT students. Due to the field of study and age, they were experienced users in using e-commerce websites. As reported in [12] and [15], there are significant differences in the way of using websites between experienced and inexperienced users, as well as those of different age groups.

Literature

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

[2] M. Hartono, A. H. Kusumo, D. A. Asikin, Affective – Cognitive – Usability (ACU) Model Incorporating Eye Tracking Analysis for Redesigning the e-Commerce Website, 21st Congress of the International Ergonomics Association (IEA 2021), IEA 2021, Lecture Notes in Networks and Systems 223 (2021), https://doi.org/10.1007/978-3-030-74614-8_5.

[3] S. Hamid, N. Z. Bawany, K. Zahoor, Assessing Ecommerce Websites: Usability and Accessibility Study, International Conference on Advanced Computer Science and Information Systems, ICACSIS (2020) 199-204, https://doi.org/10.1109/ICACSIS51025.2020.9263162.

[4] M. Gao, F. Meng, Y. Meng, Analysis of Consumer Supermarket Shopping Behaviors Based on Eye Movement Information, 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (2020) 18-22, https://doi.org/10.1109/CISP-BMEI51763.2020.9263613.

[5] I. Dianat, P. Adeli, M. Asgari Jafarabadi, M. A. Karimi, User-centred web design, usability and user satisfaction: The case of online banking websites in Iran, Applied Ergonomics 81 (2019) 102892-102900, https://doi.org/10.1016/j.apergo.2019.102892.

[6] C. Desrochers, P.-M. Léger, M. Fredette, S. Mirhoseini, S. Sénécal, The arithmetic complexity of online grocery shopping: the moderating role of product pictures, Industrial Management and Data Systems 119 (2019) 1206-1222, https://doi.org/10.1108/IMDS-04-2018-0151

[7] H. Fu, G. Manogaran, K. Wu, M. Cao, S. Jiang, A. Yang, Intelligent decision-making of online shopping behavior based on internet of things, International Journal of Information Management 50 (2020) 515-525 https://doi.org/10.1016/j.ijinfomgt.2019.03.010.

[8] S. Eraslan, Y. Yesilada, Patterns in eyetracking scanpaths and the affecting factors, Journal of Web Engineering 14 (2015) 363-385.

[9] S. Zhu, X. He, H. Yu, Research on interactive design of multilingual e-commerce platform based on eye tracker, Journal of Physics: Conference Series 1486 042008 (2020), http://doi.org/10.1088/1742-6596/1486/4/042008.

[10] I. Schröter, N. R. Grillo, M. K. Limpak, M. K., B. Mestiri, B. Osthold, F. Sebti, M. Mergenthaler, Webcam eye tracking for monitoring visual attention in hypothetical online shopping tasks, Applied Sciences 11(19) (2021), 9281 https://doi.org/10.3390/app11199281.

[11] P. Sulikowski, T. Zdziebko, D. Turzyński, Modeling online user product interest for recommender systems and ergonomics studies, Concurrency and Computation: Practice and Experience 31(22) (2017), 1-9 https://doi.org/10.1002/cpe.4301.

[12] J. McIntosh, X. Du, Z. Wu, G. Truong, Q. Ly, R. How, S. Viswanathan, T. Kanij, Evaluating Age Bias in E-commerce, 13th International Workshop on Cooperative and Human Aspects of Software Engineering (2021) 31-40, http://doi.org/10.1109/CHASE52884.2021.00012.

[13] M. Miłosz, M. Borys, M. Laskowski, Memorability Experiment vs. Expert Method in Websites Usability Evaluation. ICEIS 2013 – Proceedings of the 15th International Conference on Enterprise Information Systems, Angers, France 3 (2013) 176–182, http://doi.org/10.5220/0004453801510157.

[14] Gazepoint – Product specification, https://www.gazept.com/product/gp3hd/, [19.06.2022].

[15] Z. Tupikovskaja-Omovie, D. J. Tyler, Experienced versus inexperienced mobile users: eye tracking fashion consumers' shopping behaviour on smartphones, International Journal of Fashion Design, Technology and Education (2021) 178-186 https://doi.org/10.1080/17543266.2021.1980614.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 337–344
Received: 24 June 2022
Accepted: 3 July 2022

Improving the interface of an e-commerce website by applying universal design principles

Usprawnienie interfejsu serwisu e-commerce dzięki zastosowaniu zasad projektowania uniwersalnego

Mateusz Krzysztof Polewski*, Albert Rachwał*, Mariusz Dzieńkowski, Małgorzata Plechawska-Wójcik

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

Abstract
This paper presents an analysis of two websites in terms of accessibility and usability. An authorial e-commerce website with improvements for people with disabilities was implemented. The website was compared with a popular commercial service - Amazon. The study was conducted on a group of ten students and used the eye tracking method, a questionnaire developed for the purpose of the study and the LUT checklist. Additionally, an accessibility study was performed using the WAVE web accessibility evaluation tool. In the eye tracking study five measures were selected to evaluate the websites: the task completion time, the mean fixation time, the mean number of fixations, the mean saccade duration, and the mean number of saccades. On the basis of the obtained results and after their initial processing basic statistics and box plots were created to facilitate their interpretation. The results obtained show that the authorial e-commerce website achieved faster task completion times, comparable levels of eye tracking measures, higher scores on the LUT checklist and fewer errors that were diagnosed by the WAVE validator than the Amazon website.

Keywords: universal design; eye tracking; usability; accessibility; e-commerce websites

Streszczenie
Praca przedstawia analizę dwóch serwisów pod kątem dostępności i użyteczności. Zaimplementowany został autorski serwis e-commerce posiadający usprawnienia dla osób niepełnosprawnych. Serwis ten został porównany z popularną witryną działającą komercyjnie - serwisem Amazon. W badaniach zrealizowanych na grupie dziesięciu studentów wykorzystano technikę eyetrackingową, opracowaną do tego celu ankietę oraz listę kontrolną LUT. Przeprowadzono również badanie dostępności internetowym narzędziem WAVE. Do oceny serwisów w badaniu okulograficznym wybrano pięć miar: czas wykonania zadania, średni czas fiksacji, średnia liczba fiksacji, średni czas trwania sakady oraz średnia liczba sakad. Na podstawie uzyskanych wyników i po ich wstępnej obróbce sporządzono podstawowe statystyki oraz wykresy pudełkowe ułatwiające ich interpretację. Otrzymane wyniki pokazują, że strona autorska uzyskała krótsze czasy wykonania zadań, porównywalne poziomy miar eyetrackingowych, wyższe oceny na podstawie listy kontrolnej LUT oraz mniejszą liczbę błędów, które zostały zdiagnozowane walidatorem WAVE niż strona Amazon.

Słowa kluczowe: projektowanie uniwersalne; eyetracking; użyteczność; dostępność; strony e-commerce

*Corresponding author
Email address: albert.rachwal@pollub.edu.pl (A. Rachwal), mateusz.polewski@pollub.edu.pl (M. K. Polewski)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Niniejsza praca ma na celu usprawnienie interfejsu serwisu e-commerce poprzez zastosowanie zasad projektowania uniwersalnego. W związku z tym, analizie poddano trzy obecnie najbardziej popularne serwisy e-commerce, a następnie określono potrzeby osób z niepełnosprawnościami i zaprojektowano widoki uwzględniające nowe funkcjonalności, takie jak duże i czytelne przyciski z tekstem opisującym działanie danego przycisku oraz korespondującą kolorystyką. Ułatwiono funkcję wyszukiwania, dzięki zastosowaniu większych zdjęć produktów oraz redukcji nadmiarowej treści, aby podkreślić najważniejsze informacje. Nowe funkcjonalności miały ułatwić postrzeganie serwisu i nawigowanie po nim. Na podstawie zaprojektowanych widoków, została stworzona witryna internetowa spełniająca założone wymagania. W celu weryfikacji możliwości usprawnień interfejsu, utworzona autorska strona, została porównana z istniejącą, popularną witryną o podobnej tematyce. Porównania dokonano za pomocą badania okulograficznego, ankiety, listy kontrolnej LUT oraz automatycznego narzędzia WAVE [1]. Działania te pozwoliły ocenić, czy dzięki zastosowaniu projektowania uniwersalnego istnieje możliwość poprawy działania interfejsu popularnej marki e-commerce.

Celem badań było sprawdzenie, czy dodanie do stron internetowych usprawnień dla osób niepełnosprawnych, spowoduje że osoby bez niepełnosprawności będą także mogły wydajnie z nich korzystać zachowując jednocześnie dobre doświadczenia użytkownika.

Na potrzeby badania zdefiniowano następujące pytania badawcze:

- Jaka forma przedstawiania treści zostanie uznana przez użytkowników za najbardziej przyjazną?
- Czy poprawa jakości formularzy wpłynie pozytywnie na odbiór strony?

- Czy dodanie usprawnień dla osób niepełnosprawnych nie utrudni pracy osobom bez niepełnosprawności?
- Czy zastosowanie zasad projektowania uniwersalnego ma pozytywny wpływ na dobre doświadczenia użytkownika?

2. Pojęcia teoretyczne dotyczące projektowania uniwersalnego

Zasady projektowania uniwersalnego mają na celu ułatwić osobom niepełnosprawnym korzystanie z nowoczesnych technologii w sposób nie odbiegający od zamierzeń twórców, tak aby były one użyteczne dla wszystkich w jak największym stopniu. Dzieje się to poprzez implementację dodatkowych funkcjonalności, które będą łatwe w użyciu mimo barier poznawczych. Powinno to zostać wykonane w taki sposób, aby żadna grupa nie została dotknięta „stygmatyzacją", a wszyscy użytkownicy mogli korzystać z jednego, tego samego rozwiązania.

2.1. Badanie dostępności

Dostępność [2] strony jest atrybutem szczególnie ważnym podczas tworzenia strony działającej równie dobrze dla osób zdrowych, jak również posiadających ograniczenia percepcji. Dostępność określa się jako możliwość swobodnego poruszania po stronie internetowej i korzystania z jej funkcjonalności, co często wymaga zaimplementowania dodatkowych mechanizmów, które będę rekompensować użytkownikom braki w postrzeganiu.

2.2. Metryki WCAG

Skrótem WCAG, składającym się z pierwszych liter Web Content Accessibility Guidelines [3], oznacza się wytyczne dotyczące dostępności treści internetowych. Jest to międzynarodowy standard określający w jaki sposób informacje umieszczane na stronach internetowych są dostępne dla osób niepełnosprawnych. Obecnie funkcjonujący standard WCAG 2.1 zawiera bardzo wiele kwestii dotyczących dostępu do stron internetowych. Wśród najważniejszych należy wymienić [4]:

- *Postrzegalność* - odnoszącą się do odbioru treści na stronie za pomocą różnych zmysłów. Z tego punktu widzenia istotne jest zapewnienie alternatywnych źródeł postrzegania informacji, takich jak napisy pod filmami czy możliwość odsłuchania tekstu. Obejmuje ona również problemy związane z kontrastem i wielkością czcionki.
- *Operatywność* - określającą sposoby korzystania ze strony przez osoby z różnymi dysfunkcjami, tak aby można było mieć dostęp do wszystkich jej elementów wyłącznie za pomocą klawiatury lub dostępnych na rynku alternatyw dla myszy komputerowej.
- *Zrozumiałość* - mówiącą o tym, że każdy powinien być w stanie zrozumieć treści prezentowane na stronach www. W tym celu nie powinno się tam używać skomplikowanego języka, ani terminologii technicznej. Odbiór strony powinien być łatwy, poprzez proste nazewnictwo, spójną treść i zastosowanie przewidywalnych wzorców prezentacji treści.
- *Solidność* - czyli używanie języka znaczników HTML i reguł CSS zgodnie z praktykami czystego kodu, tak aby strony internetowe mogły być kompatybilne z urządzeniami i narzędziami, które mają ułatwiać odbiór osobom niepełnosprawnym.

2.3. Badanie użyteczności

Ważną cechą dobrego serwisu internetowego jest zapewnienie wysokiego poziomu użyteczności [5] poprzez zaprojektowanie widocznego i ułatwiającego nawigowanie układu elementów strony. Zgodnie z normą ISO 9241 [6] użyteczność to miara wydajności, efektywności i satysfakcji z jaką dany produkt może być używany przez określonych użytkowników dla osiągnięcia określonych celów w określonym kontekście użycia. Do badania użyteczności można wykorzystać różne metody: eyetracking, ocenę ekspercką uzyskaną na podstawie heurystyk [7] oraz list kontrolnych, takich jak lista LUT [8].

3. Przegląd literatury

Przed przystąpieniem do prac nad usprawnieniem interfejsu serwisu internetowego wykonano przegląd literatury z tej tematyki. Miał on na celu zidentyfikowanie czynników, które uważane są przez naukowców i specjalistów z tej dziedziny jako mające istotny wpływ na odbiór witryn internetowych przez użytkowników. W artykułach naukowych korespondujących z tematem pracy poszukiwano zarówno przydatnych wskazówek dotyczących projektowania interfejsów, jak i procedur służących do ich oceny.

O tym dlaczego warto używać okulografu i korzyściach jakie płyną z jego zastosowania można się dowiedzieć z artykułu *Eye Tracking and Web Experience* [9]. Opisano w nim szczegółowo działanie tego urządzenia i proces wykorzystania go podczas prowadzania badań. W artykule zwrócono uwagę na to, że projektowanie stron internetowych zapewniających pozytywne doświadczenia użytkownika nie jest już luksusem, ale koniecznością, aby mieć przewagę na konkurencyjnym rynku. Eyetracking może odgrywać ważną rolę w procesie skutecznego projektowania stron www i jako coraz tańsze narzędzie ma obecnie potencjał, by stać się standardem stosowanym w procesie projektowania witryn internetowych.

W artykule *A Comparative Eye Tracking Study of Usability - Towards Sustainable Web Design* [10] badacze usiłowali zidentyfikować elementy zwiększające użyteczność na stronie www. Dane zebrano za pomocą okulografu oraz z kwestionariuszy po przeprowadzeniu badań na grupie użytkowników. Okazało się z nich, że najwięcej uwagi przyciągają górna i prawa część strony internetowej, a najmniej część dolna. Respondenci osiągali najlepsze zrozumienie danych liczbowych dzięki przedstawieniu ich w formie tabel lub grafik. Kolejnym wnioskiem jest to, że nie powinno się używać więcej niż dwóch rodzajów czcionek. Ponadto intensywne kolory

można umieszczać jedynie w miejscach, na które chcę się zwrócić szczególną uwagę. Na zachowania użytkowników na stronie duży wpływ ma jej struktura. Powinna ona być prosta, podzielona na sekcje oraz zawierać niedużą ilość tekstu.

Dla lepszego zrozumienia potrzeb branży e-commerce i sposobów poprawy użyteczności tego typu serwisów sięgnięto do pracy *UX method development from Usability testing with Eye tracking for E-commerce* [11]. Artykuł ten przedstawia proces projektowania serwisu i testowania go pod kątem użyteczności. W celu zebrania danych badani testowali serwis w miejscu przypominającym ich rzeczywiste warunki pracy. Podczas pracy rejestrowano ruchu gałek ocznych oraz nagrywano zachowania uczestników, aby uchwycić ich mowę ciała, a po teście użytkownicy dodatkowo wypełnili formularze. Badanie obejmowało zrealizowanie trzech scenariuszy, w których znajdowało się 30 zadań. W pierwszym testowano funkcjonalności jednej strony, w drugim porównywano podobne elementy występujące na różnych witrynach. W trzecim scenariuszu, który nie doszedł do skutku, planowano zbadać zachowanie użytkowników, podczas zakupów internetowych realizowanych w ich własnych domach. Po zrealizowaniu badań opracowano metodę, którą nazwano metodą dwóch konkurentów. Za jej pomocą firma zajmująca się handlem elektronicznym może porównać własną witrynę z dwiema witrynami konkurencji i dzięki temu może dowiedzieć gdzie i dlaczego konsument zdecydował się zrobić zakup swoich produktów.

W artykule *Research on interactive design of multilingual e-commerce platform based on eye tracker* [12] analizowano wykorzystanie wielojęzycznej platformy handlu elektronicznego. Dane zebrano poprzez pomiar czasu dotarcia do poszczególnych elementów strony oraz rejestrację sekwencji obserwacji poszczególnych obiektów. Na podstawie uzyskanych wyników wysunięto wniosek, że prosty projekt zrobił na użytkownikach najlepsze wrażenie. W związku z tym powinno się stawiać na prostotę przy projektowaniu stron www, unikać dążenia do zapełniania każdego centymetra powierzchni strony, bo powoduje to nadmierne obciążenie pamięci wzrokowej. Podczas eksperymentu zauważono, że pierwsze spojrzenia były kierowane na lewy górny róg strony, a elementy strony były odwiedzane w kolejności od góry do dołu i od lewej do prawej strony.

Reklamy są nieodłącznym elementem stron internetowych. Dlatego też dwie następne prace, dotyczą możliwości udoskonalenia witryn e-commerce pod kątem prezentowania reklam. W artykule *Research on the degree of attraction to users of ads at different positions during targeted operations* [13], opublikowano wyniki badań, w których brało udział 30 użytkowników, wśród których były osoby z wadami wzroku, wykonujące określone zadania na różnych stronach internetowych. Eksperyment, podczas którego rejestrowano ruch oczu zaprojektowano tak, aby zapewnić użytkownikom skupienie się na osiągnięciu celu, a jednocześnie sprawdzano, na ile reklamy odciągają ich uwagę. Okazało się, że miejsce umieszczenia reklam jest istotne, a największą szansę na ich zauważenie jest koniec toru ruchu punktu spojrzenia. Celem kolejnego artykułu pt. *The effects of attention inertia on advertisements on the WWW* [14] było zaobserwowanie zmian w rozkładzie uwagi użytkowników na reklamy banerowe, podczas przeglądania kolejnych podstron serwisu internetowego. Proces przeglądania serwisu od strony domowej do strony z konkretną informacją został nazwany ścieżką. Na podstawie wyników stwierdzono, że reklama internetowa umieszczona w początkowych i końcowych fazach ścieżki powinna być wyceniana wyżej, niż reklama znajdująca się w fazie środkowej. Oznacza to, że odbiorcy byli bardziej wrażliwi na reklamę peryferyjną.

W pracy *Designing Usable Web Forms – Empirical Evaluation of Web Form Improvement* [15] zebrano 20 wytycznych dotyczących projektowania funkcjonalnych formularzy internetowych i udowodniono ich skuteczność. Jak pokazały wyniki badań okulograficznych i wyniki zebrane za pomocą kwestionariuszy, użytkownicy, pracując z usprawnionymi formularzami, wypełniali je szybciej, przy średnio mniejszej liczbie prób oraz z wykonując mniejszą liczbę ruchu gałek ocznych. Dodatkową formą weryfikacji doświadczeń użytkowników było przeprowadzenie wywiadu, z którego wynika, że badani mieli większą ogólną satysfakcję podczas korzystania z ulepszonych formularzy.

W artykule *Markov chain to analyze web usability of a university website using eye tracking data* [16] analizowano dane ruchu gałek ocznych w dwóch grupach: 56 osobowej grupie uczniów szkoły średniej oraz 66 osobowej grupie studentów, którzy wykonywali 10 zadań na stronie uniwersytetu Cagliari. Zebrane dane poddano analizie łańcuchem Markova oraz przedstawiono szczegółową procedurę przeprowadzenia takiego badania. Najważniejszym wnioskiem płynącym z tych badań jest to, że typowy użytkownik nie przewija strony, a jedynie przegląda widoczne informacje, dlatego też najważniejsze sekcje strony powinny być umieszczone na górze, a ważne informacje podkreślone poprzez kontrastowy kolor.

4. Plan badań

W celu weryfikacji możliwości usprawnienia interfejsu serwisu e-commerce pod kątem dostępności i użyteczności, zostały przeprowadzone badania porównawcze dla dwóch systemów. Jednym z nich była popularna na świecie marka z branży e-commerce – Amazon, a drugim – prototypowy serwis zaimplementowany na potrzeby tego badania. W niniejszej pracy skupiono się na zbadaniu pewnych szczególnych elementów tych serwisów, takich jak lista produktów, strona prezentacji produktu, zawartość koszyka oraz formularz. Elementy te mogą w znaczący sposób wpływać na odczucia użytkownika.

4.1. Metody badawcze

Oba serwisy zostały poddane badaniom okulograficznym z udziałem grupy badawczej mającej wykonywać na nich serię zadań. Po ich zrealizowaniu, uczestnicy wypełnili również ankiety dotyczące przeglądanych

stron. Do celów porównawczych zostały wybrane następujące miary okulograficzne:

- *Średni czas wykonania zadania* - jest to czas odmierzany od momentu, kiedy użytkownikowi zostanie wyświetlona strona, na której ma wykonać zadanie, do chwili, kiedy potwierdzi wykonanie zadania przejściem do kolejnego zadania lub gdy upłynie 15 sekund - ustalony maksymalny czas wyświetlenia strony. Na podstawie czasów uzyskanych przez każdego z badanych dla poszczególnych zadań zostanie obliczona średnia.
- *Średnia liczba fiksacji* - jest to liczba fiksacji dla badanej grupy podczas oglądania danej strony. Mniejsza liczba fiksacji może wskazywać na to, że poszukiwane informacje były łatwiejsze do znalezienia.
- *Średni czas fiksacji* - świadczy o tym, ile czasu użytkownicy utrzymywali wzrok nieruchomo na danym obszarze. Większa długość fiksacji może oznaczać, że osobie podoba się oglądana treść lub uważa ona, że w danym obszarze może pozyskać potrzebną informację. Krótkie czasy fiksacji mogą być interpretowane jako oznaka niepewności badanego i potrzeby dalszego wyszukiwania informacji.
- *Średnia liczba sakad* - to miara określająca liczbę ruchów gałek ocznych podczas skanowania wzrokowego, dopóki badany nie odnajdzie niezbędnych informacji. Mniejsza liczba sakad może być oznaką, że informacja jest łatwiejsza do znalezienia lub lepiej uporządkowana.
- *Średni czas sakady* - jest wyznacznikiem ile czasu upływa pomiędzy kolejnymi punktami fiksacji. Krótszy czas sakady może oznaczać większe uporządkowanie treści lub przewidywalność rozmieszczenia elementów na stronie, natomiast długie czasy sakad mogą świadczyć o trudności w odnalezieniu informacji.

Do ułatwienia interpretacji wyników zostały użyte wykresy ramka-wąsy, nazywane również wykresami pudełkowymi (ang. *box-plot*). Dolna podstawa pudełka oznacza pierwszy kwartyl, a górna podstawa trzeci kwartyl. Środkowa kreska znajduje się w miejscu mediany, która jest drugim kwartylem, czyli wartością środkową. Dolny wąs określa 25% obserwacji o wartościach niższych od pierwszego kwartyla, natomiast górny wąs - 25% obserwacji o wartościach wyższych od trzeciego kwartyla. Krańce wąsów oznaczają wartość maksymalną i minimalną obserwacji.

4.2. Stanowisko badawcze

Przeprowadzone badania miały miejsce w laboratorium należącym do Katedry Informatyki Politechniki Lubelskiej. Zostały zapewnione w nim jednakowe, sztuczne warunki oświetlenia podczas całego badania. Przy stanowisku znajdował się fotel z oparciem na ramiona, zapewniający wygodną pozycję w trakcie badania. Poza tym miał on możliwość regulacji wysokości siedzenia, co było niezbędne ze względu na różnice we wzroście uczestników badań. Urządzeniem użytym w badaniach był eyetracker Gazepoint GP3 HD [17], współpracujący z komputerem Acer Nitro 5. Poza badanym w pomieszczeniu, gdzie były realizowane badania, przebywał również moderator kontrolujący poprawny przebieg eksperymentu. Do obsługi eyetrackera wykorzystano oprogramowanie GP Controll oraz iMotions 9.0 [18].

4.3. Grupa badawcza

Osoby biorące udział w eksperymencie to studenci Politechniki Lubelskiej, stanowiący grupę 10 osób, siedmiu mężczyzn i trzech kobiet. Były to osoby w wieku 22-25 lat, sprawnie posługujące się komputerem. Miały one już wcześniej styczność z serwisami e-commerce oraz posiadały podstawową wiedzę dotyczącą gospodarki elektronicznej. Sześcioro wśród badanych nosiło okulary lub soczewki kontaktowe.

4.4. Scenariusz badań

Każdy uczestnik badań swój udział w eksperymencie rozpoczynał od kalibracji okulografu, po czym wykonywał zestaw 21 zadań dla obu serwisów, polegających na wyszukiwaniu określonych elementów w ich interfejsach. Każde zadanie polegało na wyszukaniu wzrokowym elementu pełniącego jakąś konkretną funkcję w prezentowanym widoku strony internetowej.

W Tabeli 1 zostały wypisane kolejno przeprowadzone zadania. Dziesięć zadań jest analogicznych dla obydwu badanych serwisów e-commerce, mających na celu wykrycie różnic między nimi i sprawdzeniu, które rozwiązanie cechuje się lepszymi doświadczeniami użytkowników oraz krótszym czasem wykonania. Zadanie jedenaste miało sprawdzić, czy łatwe jest znalezienie elementu odpowiedzialnego za zmianę wielkości czcionki i kontrastu. Zadanie to było wykonywane tylko na jednej ze stron, ponieważ tylko strona autorska posiadała konieczne ku temu funkcjonalności.

Tabela 1. Treść zadań

Lp.	Treść zadania
1	Znajdź przycisk, który otwiera menu strony
2	Znajdź przycisk, za pomocą którego przejdziesz do konta użytkownika
3	Znajdź przycisk, który zmienia liczbę produktów w koszyku
4	Znajdź przycisk, który usuwa elementy z koszyka
5	Znajdź miejsce, gdzie możesz ustalić przedział cenowy
6	Znajdź produkt o najlepszej opinii
7	Znajdź przewidywany termin dostawy
8	Znajdź markę produktu w specyfikacji
9	Znajdź pole, w którym wpiszesz kod pocztowy
10	Znajdź pole, w którym wybierzesz kraj
11	Znajdź przyciski do zmiany kontrastu oraz wielkości czcionki

4.5. Ankieta

Po przeprowadzeniu eksperymentu eyetrackingowego badani odpowiadali na 6 pytań zawartych w ankiecie, które dotyczyły oglądanych przez nich stron. Pytania użyte w ankiecie miały na celu zebranie informacji na

temat tego jak badani oceniają zastosowane rozwiązania na prezentowanej stronie www. Oto treść tych pytań:

1. Czy podoba Ci się koncepcja listy kategorii w postaci kafelek ze zdjęciami?
2. Czy uważasz, że element którejś z oglądanych przez Ciebie stron uniemożliwiał Ci pełne postrzeganie treści?
3. Czy wolisz oglądać zbiór wyszukiwanych produktów w postaci kafelek czy listy?
4. Który rodzaj zmiany ilości produktów w koszyku preferujesz?
5. Który rodzaj formularza był dla Ciebie bardziej czytelny?
6. Jakie zmiany na oglądanych stronach ułatwiłyby Ci łatwiejsze z nich korzystanie?

Oprócz powyższych pytań każdy z ankietowanych wypełnił również dwie listy kontrolne LUT: pierwsza dla serwisu e-commerce oraz druga dla serwisu prototypowego.

5. Wyniki

Po przeprowadzeniu badań i zebraniu ankiet podjęto się analizy wyników, mającej na celu zidentyfikowanie elementów, które udało się skutecznie usprawnić w interfejsie e-commerce.

5.1. Wyniki ankiet

W ankiecie wszyscy badani stwierdzili, że żaden z elementów, z których zbudowane były strony, nie utrudniał im ich odbioru. Układ formularza został oceniony jako bardziej czytelny, kiedy poszczególne pola były umieszczone jedno pod drugim, unikając sytuacji, gdy pola znajdowały się równolegle. Na pytanie czy listy produktów w formie kafelek prezentują się lepiej niż zwykłe listy, odpowiedzi rozłożyły się po połowie. Pytanie otwarte dotyczyło tego, jakie zmiany w prezentowanej stronie chcieliby wprowadzić badani. Wśród zaproponowanych usprawnień pojawiły się następujące sugestie:

- Zmniejszenie liczby elementów wyświetlanych jednocześnie na stronie.
- Zwiększenie wielkości przycisków.
- Wyróżnienie przycisków odpowiadających za otwarcie menu.
- Zwiększenie odstępów pomiędzy poszczególnymi elementami, które były szczególnie małe w serwisie Amazon, gdzie przyciski i informacje są ułożone zbyt blisko siebie.
- Ograniczenie liczby kolorów.

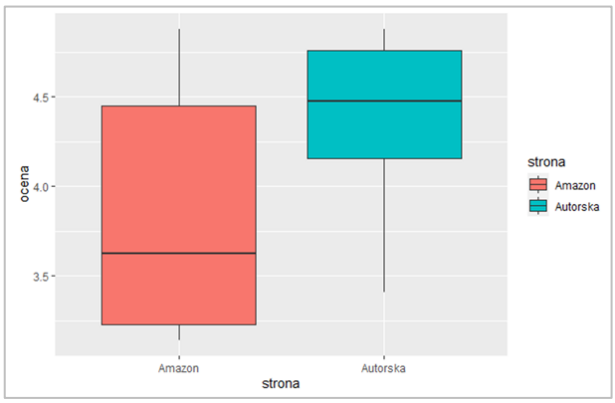
5.2. Wyniki list LUT

Wypełnienie przez badanych list kontrolnych LUT pozwoliło na obliczenie współczynnika WUP (ang. Web Usability Points) [19] dla każdego serwisu. Wartość tego współczynnika zawiera się w przedziale od 1 do 5. W Tabeli 2 przedstawiono wyniki WUP, jakie otrzymano na podstawie ocen wystawionych przez poszczególnych użytkowników.

Tabela 2: Wartość współczynnika WUP

Osoba	Serwis Amazon	Autorski serwis
1	4,01	4,81
2	4,60	4,37
3	3,81	4,78
4	4,88	4,70
5	3,44	3,41
6	4,86	4,88
7	3,14	4,58
8	3,18	4,18
9	3,37	4,15
10	3,18	3,44

Na Rysunku 1 znajduje się wykres pudełkowy sporządzony na podstawie danych z Tabeli 2, z którego widać, że około 75% ocen na stronie Amazon cechuje się niższą wartością, niż mediana na stronie autorskiej. W przybliżeniu 50% osób oceniło stronę autorską co najmniej na 4,5, zaś dla strony Amazon było to jedynie 25% głosów.



Rysunek 1: Wykres pudełkowy przedstawiający współczynnik WUP.

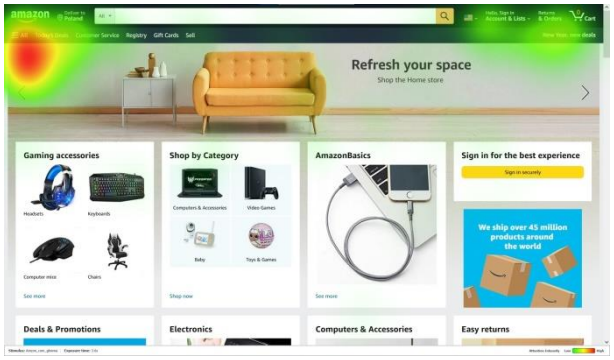
5.3. Analiza eyetrackingowa

Dzięki przeprowadzeniu badań okulograficznych można było bardzo łatwo uzyskać ścieżki skanowania dla wszystkich zadań, wygenerowane oddzielnie dla każdego z badanych. Dało to łącznie 210 obrazów zawierających oglądaną stronę www z nałożoną ścieżką, czyli trasą jaką podążał wzrok badanego.

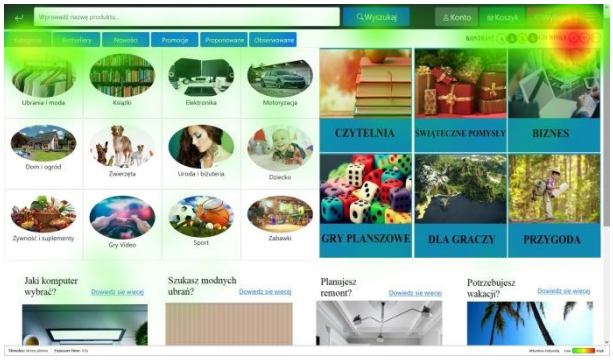
Poza ścieżkami skanowania dla każdego zadania otrzymano zagregowaną mapę cieplną. Obszary oznaczone kolorem oznaczają miejsca, w których było najwięcej fiksacji lub ich czas trwania był długi. Kolor zielony oznacza niewielkie zainteresowanie danym obszarem, a czerwony duże zainteresowanie. Jak widać z map cieplnych przedstawionych na Rysunkach 2 i 3, najwięcej uwagi poświęcano obszarom, których przeskanowanie było niezbędne do wykonania zadań. Kolor zielony to miejsca eksplorowane przez część badanych zanim ich uwaga dotarła do właściwych informacji.

Rysunki 2 i 3 pokazują mapy cieplne po wykonaniu zadania 1, polegającego na znalezieniu przycisku, który otwiera menu strony. Intensywny kolor czerwony pokazuje tendencję użytkowników do poszukiwania przyci-

sków nawigacyjnych w lewym oraz prawym górnym rogu strony.



Rysunek 2: Mapa cieplna strony Amazon dla zadania 1.



Rysunek 3: Mapa cieplna strony autorskiej dla zadania 1.

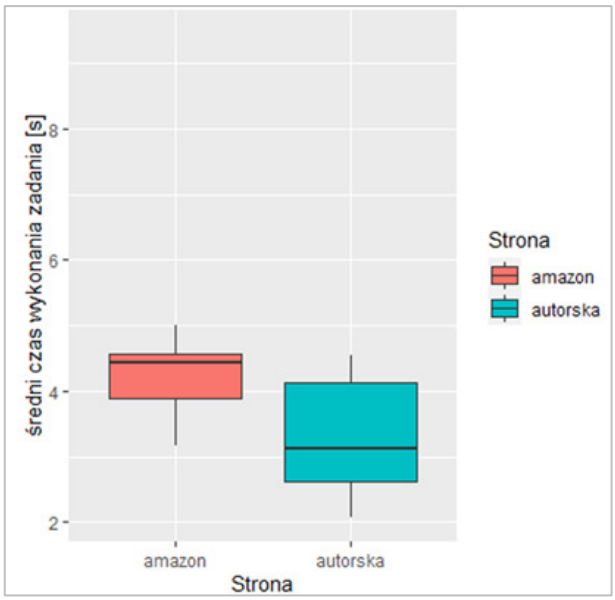
5.4. Czasy realizacji zadań

W Tabeli 3 znajduje się zestawienie czasów wykonania zadań wraz z odchyleniami standardowymi dla strony Amazon oraz dla strony autorskiej.

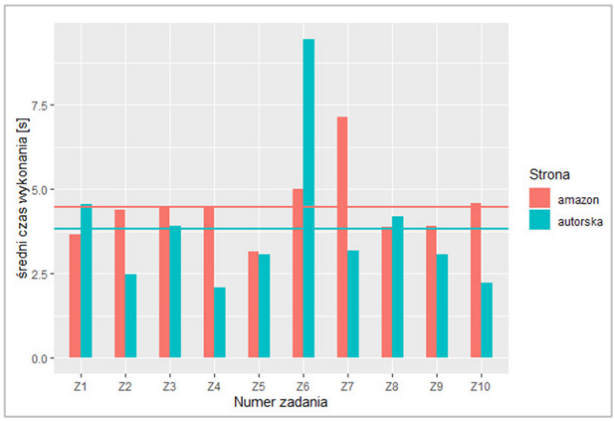
Tabela 3: Czasy wykonania zadań dla dwóch badanych stron

Numer zadania	Strona autorska		Serwis Amazon	
	Średni czas [s]	Odchylenie std [s]	Średni czas [s]	Odchylenie std [s]
Zad. 1	4.54	3,25	3,64	1,59
Zad. 2	2.47	1,24	4,39	1,75
Zad. 3	3.90	2,20	4,49	1,66
Zad. 4	2,07	0,77	4,46	2,10
Zad. 5	3,07	1,16	3,16	0,74
Zad. 6	9,45	3,11	5,00	2,15
Zad. 7	3,18	1,74	7,14	3,90
Zad. 8	4,19	3,25	3,88	2,42
Zad. 9	3,06	1,12	3,90	1,28
Zad. 10	2,22	0,81	4,59	2,37

Na Rysunku 4 znajduje się wykres pudełkowy utworzony na podstawie Tabeli 3. Widać na nim, że realizacja zadań na stronie autorskiej trwała krócej. Natomiast Rysunek 5 przedstawia średni czas wykonania poszczególnych zadań na obu stronach. Poziomymi liniami zostały zaznaczone wartości średnie.



Rysunek 4: Wykres pudełkowy przedstawiający średnie czasy wykonania zadań dla dwóch badanych stron.

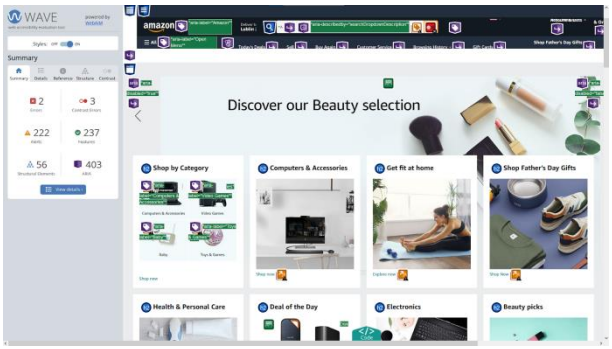


Rysunek 5: Wykres przedstawiający średnie czasy wykonania poszczególnych zadań.

5.5. Analiza dostępności narzędziem WAVE

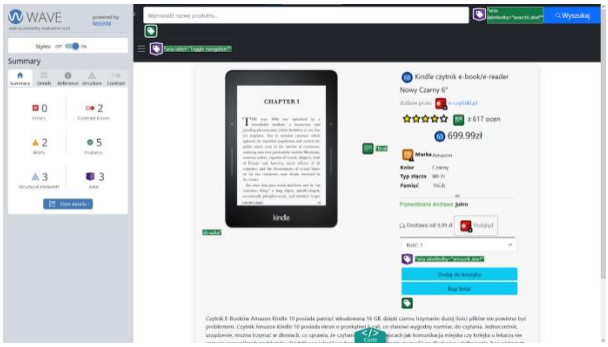
WAVE jest narzędziem służącym do badania i oceny dostępności danego serwisu, za pomocą takich metryk jak kontrast czy widoczność elementów. Za pomocą wtyczki internetowej, dedykowanej dla przeglądarki Firefox zostały zbadane oba analizowane serwisy: Amazon i strona autorska. Do analiz zostały wybrane po 3 podstrony: strona główna, strona do wyszukiwania produktu oraz strona produktu.

Badanie serwisu Amazon, wykazało wiele błędów i ostrzeżeń związanych m.in. z elementami niskokontrastującymi, pustymi znacznikami „label", małymi tekstami, czy brakiem, lub źle dostosowanymi tekstami alternatywnymi. Przykładowo, raport z analizy strony głównej pokazanej na Rysunku 6, informuje o dużej liczbie zbyt długich tekstów alternatywnych przy obrazkach produktów, co może utrudniać korzystanie ze strony za pośrednictwem np. czytników ekranu.



Rysunek 6: Strona główna serwisu Amazon - ocena narzędziem WAVE.

Badanie strony głównej i strony do wyszukiwania produktu w serwisie autorskim, nie wykazało błędów, jednakże pojawiło się ostrzeżenie związane z brakiem nagłówka „h1". W przypadku strony produktu na Rysunku 7, zostały rozpoznane dwa ostrzeżenia i dwa błędy kontrastu, które można wyeliminować uruchamiając w serwisie tryb wysokiego kontrastu.



Rysunek 7: Strona produktu serwisu autorskiego - ocena narzędziem WAVE.

6. Wnioski

Niniejsza praca miała na celu usprawnienie interfejsu serwisu e-commerce poprzez zastosowanie zasad projektowania uniwersalnego. Cel zrealizowano tworząc prototypową stronę e-commerce oferującą zbliżone funkcjonalności do istniejących obecnie na rynku, jednak podjęto się dokonania pewnych modyfikacji w celu zbadania, czy zostaną one odebrane przez użytkowników jako pozytywne i czy zostaną otrzymane istotne różnice w efektywności korzystania z usprawnionego serwisu. Nowo zaprojektowana strona autorska cechowała się udogodnieniami dla osób niepełnosprawnych, które nie zostały zaimplementowane we wstępnie przeanalizowanych serwisach komercyjnych. Jednocześnie funkcjonalności te miały zostać dodane w taki sposób, aby nie wpłynąć na jakość odbioru strony przez osoby bez niepełnosprawności. Weryfikacji usprawnień dokonano za pomocą badania okulograficznego przeprowadzonego na grupie dziesięciu studentów. Badana grupa wypełniła również skonstruowaną na potrzeby eksperymentu krótką ankietę połączoną z listą kontrolną LUT.

W odpowiedzi na pytanie jaką formę przedstawienia treści preferują badani, uzyskano informacje, że podczas korzystania ze stron oczekują oni mniejszej liczby elementów oraz mniejszej liczby kolorów jednocześnie wyświetlanych na stronie, jak również zwiększenia odstępów pomiędzy elementami, które bardzo często w serwisach e-commerce są położone blisko siebie, po to aby zmieścić jak najwięcej treści. Dzięki ankiecie otrzymano informację, że 60% badanych preferuje zmianę liczby produktów w koszyku za pomocą przycisków, 30% w formie listy rozwijanej, a tylko jedna osoba jako pole do uzupełnienia. W kwestii przedstawienia produktów jako zwykła lista lub lista z kafelkami, badani podzielili swoje głosy po 50%. Natomiast 90% badanych uznała, że przedstawienie kategorii produktów w formie kafelek dostępnych na stronie jest korzystnym rozwiązaniem.

Na pytanie, czy poprawa jakości formularza wpłynie pozytywnie na odbiór strony, wszyscy badani odpowiedzieli, że preferują sytuację kiedy poszczególne pola formularza znajdują się bezpośrednio pod sobą, czyli tak jak to zostało zaprojektowane na stronie autorskiej. Pozytywny wpływ takiego układu formularza widać również po czasach realizacji zadań. Na podstawie tabeli 3 widać, że na stronie autorskiej szybciej o ponad 20% przebiegło wykonanie zadania nr 9 oraz o 50% zadania nr 10. W celu łatwiejszego odbioru strony przez osoby niepełnosprawne strona autorska posiada możliwości zmiany kontrastu i wielkości czcionki.

Do badania okulograficznego użyto pięciu miar. Średni czas wykonania zadań prawie we wszystkich przypadkach okazał się krótszy na stronie autorskiej, podobnie liczba i długość fiksacji mogą wskazywać na pewne usprawnienia w odbiorze strony. Zarówno czasy sakad, jak i ich liczba osiągały nieznacznie niższe wartości dla strony Amazon, co może oznaczać na szybsze osiąganie kolejnych punktów fiksacji w trakcie skanowania wzrokowego, może to również świadczyć o wyższym uporządkowaniu informacji w tym serwisie.

Ze względu na fakt, że większość wyników uzyskanych dla wybranych miar eyetrackingowych różni się nieznaczne, za wyjątkiem czasu wykonania zadań, który był krótszy na stronie autorskiej, można odpowiedzieć twierdząco na główne pytanie badawcze zawarte w celu, że dodanie usprawnień dla osób niepełnosprawnych nie wpływa negatywnie na odbiór stron przez osoby bez niepełnosprawności.

Do przeanalizowania błędów występujących na stronach użyto internetowego narzędzia WAVE. Wykazało ono pewną liczbę błędów w serwisie Amazon, jednak była to niewielka liczba usterek na tle innych serwisów, które zostały przebadane przez autorów w fazie przygotowania do badań. Na stronie autorskiej liczba błędów była znacznie mniejsza, a na większości podstron błędy nie występowały. W odpowiedzi na czwarte pytanie badawcze, brak błędów na stronie zinterpretować w ten sposób, że zastosowanie zasad projektowania uniwersalnego ma pozytywny wpływ na doświadczenia użytkownika.

Na podstawie postawionych w pracy pytań oraz przeprowadzonych badań można stwierdzić, że współcześnie działające serwisy e-commerce potrzebują dalszej pracy nad dostosowaniem ich treści oraz przebu-

dowy struktury do potrzeb osób dotkniętych ograniczeniami percepcji. Narzędzia takie jak eyetracker, lista kontrolna LUT oraz walidator WAVE już teraz mają duże znaczenie w procesie oceny jakości interfejsów, a w przyszłości ich rola będzie jeszcze większa.

Literatura

[1] WAVE Web Accessibility Evaluation Tool, https://wave.webaim.org/, [31.05.2022].

[2] Introduction to Web Accessibility, https://www.w3.org/WAI/fundamentals/accessibility-intro/, [25.06.2022].

[3] Web Content Accesibillity Guidelines (WCAG) 2.1, https://www.w3.org/TR/WCAG21/, [01.06.2022].

[4] MDN PLUS, Understanding the Web Content Accessibility Guidelines, https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/, [25.06.2022].

[5] Wikipedia, Web usability, https://en.wikipedia.org/wiki/Web_usability, [16.06.2022].

[6] Wikipedia, ISO 9241, https://en.wikipedia.org/wiki/ISO_9241, [16.06.2022].

[7] W. Heret, Heurystyki - Czyli skuteczna analiza użyteczności twojej strony, GrupaTense (2022), https://www.grupa-tense.pl/blog/heurystyki-czyli-skuteczna-analiza-uzytecznosci-twojej-strony/, [16.06.2022].

[8] M. Miłosz, M. Plechawska-Wójcik, M. Borys, M. Laskowski, Quality improvement of ERP system GUI using expert metod: A case study, In 2013 6th International Conference on Human System Interactions, (2013) 145-152.

[9] S. Djamasbi, Eye Tracking and Web Experience, THCI, 2014.

[10] M. Țichindelean, M. T. Țichindelean, I. Cetină, G. Orzan, A comparative eye tracking study of usability - towards sustainable web design, Sustainability (Switzerland), 13(18) (2021), https://aisel.aisnet.org/thci/vol6/iss2/2, [16.06.2022].

[11] M. Nilsson, UX method development from Usability testing with Eye tracking for E-commerce, Malmö University, Faculty of Culture and Society, 2018.

[12] S. Zhu, X. He, H. Yu, Research on interactive design of multilingual e-commerce platform based on eye tracker, Paper presented at the Journal of Physics: Conference Series, 1486(4) (2020), http://dx.doi.org/10.1088/1742-6596/1486/4/042008.

[13] L. Ran, X. Zhang, H. Luo, H. Hu, Z. Wang, Research on the degree of attraction to users of ads at different positions during targeted operation, (2018) 364-370, https://www.doi.org/10.1007/978-3-319-60492-3_35.

[14] J. Wang, R. Day, The effects of attention inertia on advertisements on the WWW. Computers in Human Behavior, *23*(3), (2007) 1390-1407, https://www.doi.org/10.1016/j.chb.2004.12.014.

[15] M. Seckler, S. Heinz, J.A. Bargas-Avila, K. Opwis, A.N. Tuch, Designing usable web forms- empirical evaluation of web form improvement guidelines, Paper presented at the Conference on Human Factors in Computing Systems - Proceedings, (2014) 1275-1284, https://www.doi.org/10.1145/2556288.2557265.

[16] G. Zammarchi, L. Frigau, F. Mola, Markov chain to analyze web usability of a university website using eye tracking data, Statistical Analysis and Data Mining, 14(4), (2021) 331-341, https://www.doi.org/10.1002/sam.11512.

[17] Gazepoint GP3 HD Eye Tracker, https://www.gazept.com/product/gp3hd/, [31.05.2022].

[18] Imotions Eye Tracking, https://imotions.com/eye-tracking/, [31.05.2022].

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

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 345–349
Received: 28 June 2022
Accepted: 12 September 2022

Comparative of React and Svelte programming frameworks for creating SPA web applications

Porównanie szkieletów programistycznych React i Svelte do tworzenia aplikacji internetowych typu SPA

Sebastian Dubaj*, Beata Pańczyk

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

Abstract
The aim of the article is to perform a comparative analysis of two programming frameworks on the example of one of the most popular currently React v17 and Svelte, which is gaining considerable popularity. Two applications with the same functionalities in both analyzed frameworks were implemented to conduct the research. The analysis mainly concerns the rendering times of views, but the application structures and their size are also compared. As a result of the research, it was found that the Svelte application is much more efficient compared to the React application.

Keywords: React; Svelte; performance; comparative analysis

Streszczenie
Celem artykułu jest przeprowadzenie analizy porównawczej dwóch szkieletów programistycznych na przykładzie jednego z najpopularniejszych obecnie React v17 oraz Svelte, który zdobywa znaczną popularność. Do przeprowadzenia badań zaimplementowano dwie autorskie aplikacje o takich samych funkcjonalnościach w obu analizowanych szkieletach programistycznych. Analiza dotyczy przede wszystkim czasów renderowania widoków, ale porównywane są także struktury aplikacji oraz ich rozmiar. W wyniku badań stwierdzono, że aplikacja Svelte jest znacznie bardziej wydajna w porównaniu do aplikacji React.

Słowa kluczowe: React; Svelte; wydajność; analiza porównawcza

*Corresponding author
Email address: **sebastian.dubaj@pollub.edu.com** (S. Dubaj)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Zapotrzebowanie na strony internetowe ciągle rośnie, wiąże się z tym ciągłe poszukiwanie i tworzenie nowych narzędzi do ich tworzenia. Początkowo portale internetowe przedstawiały jedynie treści, bez możliwości interakcji z użytkownikiem. Były tworzone jedynie aplikacje statyczne za pomocą plików HTML, kaskadowych arkuszy styli oraz języka programowania JavaScript. Wraz ze wzrostem popularności Internetu i coraz większych wymagań użytkowników, powstają nowe rozwiązania zwiększające efektywną interakcję z użytkownikiem.

W przeszłości jedynym rozwiązaniem na tworzenie serwisów internetowych były aplikacje typu MPA (ang. Multi Page Application), za pomocą którego można tworzyć aplikacje wielostronicowe. Tego typu aplikacje działały w trybie żądanie-odpowiedź przesyłane w trybie synchronicznym, co wiąże się z przeładowywaniem od początku każdej strony. To z kolei powoduje wolniejsze ładowanie stron, pobieranie danych oraz wyświetlaniem treści, co wpływa na długi czas oczekiwania i gorsze doświadczenie użytkownika przy korzystaniu z aplikacji MPA [1].

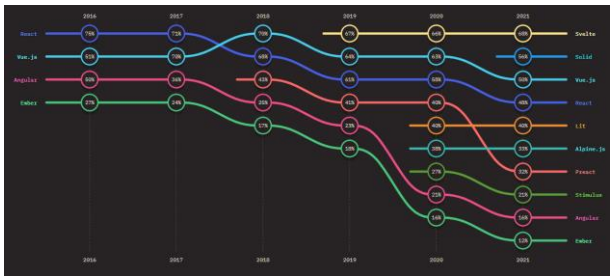
Konkurencyjność stron oraz dbanie o to, aby użytkownik pozostał na stronie jak najdłużej, chętnie na nią wracał oraz miał przyjazne doświadczenia, zmusiła do stworzeniu nowego mechanizmu aplikacji – aplikacji typu SPA [2] (ang. Single Page Application). Obecnie jest to najpopularniejszy sposób na tworzenie aplikacji internetowych. Pozwala na przyjemniejsze korzystanie ze stron, głównie przez to, że strona nie musi się przeładowywać kilka razy, a w związku z tym jej działanie jest znacznie szybsze, ponieważ treść oraz dane są pobierane w tle (asynchronicznie).

Tworzenie coraz bardziej skomplikowanych aplikacji powodowało przedłużenie procesu ich powstawania, dlatego pojawiły się szkice programistyczne (ang. frameworks). Za pomocą gotowych szkiców, aplikacje można tworzyć znacznie wydajniej co wpływa na szybkość prac nad projektem. Z biegiem lat powstawały nowe szkielety programistyczne (np. Angular, React, Vue.js, Preact, Svelte) [3], które różnią się strukturą i zasadą działania, ale wszystkie bazują na języku JavaScript i żądaniach asynchronicznych. Każdy szkic programistyczny ułatwia tworzenie aplikacji, lecz ich liczba utrudnia podjęcie decyzji, który należy wybrać. Szkice różnią się między innymi szybkością renderowania widoków, wydajnością, strukturą kodu i poziomem trudności samej implementacji.

2. Cel i hipotezy badawcze

Celem badań jest porównanie dwóch szkieletów programistycznych do tworzenia aplikacji internetowych typu SPA. Analiza dotyczy React v17, który obecnie jest najpopularniejszym frameworkiem oraz Svelte, który jest nowoczesnym rozwiązaniem i w ostatnim

czasie zaczął zdobywać popularność i według statystyk znajduje się na 4 miejscu po względem używalności (Rys. 1). Podstawowymi kryteriami porównania narzędzi są: wydajność, metryki kodu, struktura aplikacji, rozmiar aplikacji, łatwość nauki.



Rysunek 1: Popularność frameworków Javascript w roku 2021 [3].

Postawiona została następująca hipoteza badawcza: *„Aplikacja Svelte 3.0 szybciej wykonuje operacje na komponentach graficznego interfejsu użytkownika w porównaniu do analogicznej aplikacji React 17.0”.*

3. Przegląd literatury

Istnieje wiele artykułów porównujących biblioteki JavaScript. Dotyczą one głównie najpopularniejszych rozwiązań takich jak React, Vue czy Angular [4-6]. Natomiast artykułów dotyczących porównania Svelte oraz React jest zaledwie kilka. Jednym z nich jest artykuł [7], w którym autor porównuje narzędzie Angular oraz Svelte. Przeprowadzone badania dotyczą wydajności dwóch takich samych aplikacji napisanych z wykorzystaniem wskazanych narzędzi. Na podstawie wyników autor wywnioskował, że aplikacja Svelte była znacznie wydajniejsza, a w jednym ze scenariuszy Svelte był aż 4-krotnie szybszy w renderowaniu komponetów.

Artykuł [8] dotyczy bibliotek React, Vue, Angular oraz Svelte. Autor powierzchownie opisuje każdy z frameworków, a do badań dotyczących metryki kodu, wydajności, rozmiaru aplikacji, wykorzystuje cztery takie same aplikacje napisane z wykorzystaniem każdego narzędzia. Zgodnie z wnioskami autora tej pracy, Svelte jest najlepszy pod względem liczby linii kodu, natomiast w przypadku wydajności Svelte jest na drugim miejscu, a React na ostatnim.

4. React

React [9] jest najpopularniejszą i często wykorzystywaną biblioteką języka programowania JavaScript, przeznaczoną do tworzenia interfejsów użytkownika. Została stworzona w 2013 roku przez programistów platformy społecznościowej Facebook, w celu usprawnienia budowy serwisów internetowych.

Biblioteka słynie z wielu koncepcji i rozwiązań, między innymi jest to **wirtualne drzewo DOM** (ang. Virtual Document Object Model). Podejście to pozwala na wyszukiwanie różnic zachodzących w aplikacji, porównując zmiany między wirtualnym i prawdziwym DOM, następnie aktualizuje zmiany w komponentach, które tego wymagają, zamiast przeładowywać całą stronę.

Kolejną ze szczególnych cech React jest język skryptowy JSX (JavaScript XML), który jest rozszerzeniem języka JavaScript o możliwość wstawiania znaczników HTML do kodu JavaScript. JSX znacznie poprawia czytelność kodu do poziomu kodu napisanego w HTML.

5. Svelte

Svelte [10-13] jest biblioteką języka JavaScript, która dopiero zdobywa popularność. Została stworzona w 2016 roku przez Richa Harrisa. Svelte w dosłownym tłumaczeniu oznacza elegancki, szczupły, co idealnie opisuje narzędzie, ponieważ aplikacja Svelte zajmuje mało pamięci, a kod źródłowy zawiera stosunkowo mało linii kodu w przeciwieństwie do jego konkurentów.

Główną cechą Sveltepowrót do korzystania z rzeczywistego DOM (ang. Real DOM), który w przeciwieństwie do React nie tworzy wirtualnego DOM. Real DOM jest tworzony podczas kompilacji aplikacji do kodu JavaScript i służy do aktualizowania danego komponentu, gdy został on zmieniony.

Dużą zaletą Svelte jest fakt, że posiada niski próg wejścia, to znaczy, że początkujący programiści będą potrafili szybko nauczyć się jego działania.

6. Metoda badań

Do przeprowadzenia badań stworzono dwie proste aplikacje testowe w React oraz Svelte z identycznymi komponentami interfejsu użytkownika, o takiej samej funkcjonalności. Każda aplikacja służy do tworzenia losowych użytkowników, którzy posiadają zdjęcie, imię, nazwisko, wiek, numer telefonu. Wszystkie dane zostały zapisane w pliku JSON, z którego aplikacje pobierają losowe dane. Dane zostały pobrane z ogólnodostępnego i darmowego API [14], a przykładowy obiekt z danymi użytkownika przedstawiono na listingu 1.

Listing 1: Struktura danych pobranych z randomAPI

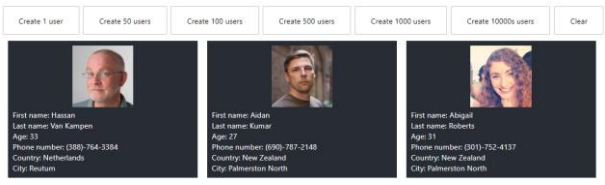
```
{
 "results": [
  {
   "gender": "male",
   "name": {
    "title": "mr",
    "first": "brad",
    "last": "gibson"
   },
   "location": {
    "street": "9278 new road",
    "city": "kilcoole",
    "state": "waterford",
    "postcode": "93027",
    "coordinates": {
     "latitude": "20.9267",
     "longitude": "-7.9310"
    },
    "timezone": {
     "offset": "-3:30",
     "description": "Newfoundland"
    }
```

```
    },
    "email": "brad.gibson@example.com",
    "login": {
      "uuid": "155e77ee-ba6d-486f-95ce-0e0c0fb4b919",
      "username": "silverswan131",
      "password": "firewall",
      "salt": "TQA1Gz7x",
      "md5": "dc523cb313b63dfe5be2140b0c05b3bc",
      "sha1": "7a4aa07d1bedcc6bcf4b7f8856643492c191540d",
      "sha256":
"74364e96174afa7d17ee52dd2c9c7a4651fe1254f471a78bda0190135dcd3480"
    },
    "dob": {
      "date": "1993-07-20T09:44:18.674Z",
      "age": 26
    },
    "registered": {
      "date": "2002-05-21T10:59:49.966Z",
      "age": 17
    },
    "phone": "011-962-7516",
    "cell": "081-454-0666",
    "id": {
      "name": "PPS",
      "value": "0390511T"
    },
    "picture": {
      "large": "https://randomuser.me/api/portraits/men/75.jpg",
      "medium":
"https://randomuser.me/api/portraits/med/men/75.jpg",
      "thumbnail":
"https://randomuser.me/api/portraits/thumb/men/75.jpg"
    },
    "nat": "IE"
  }
 ],
 "info": {
  "seed": "fea8be3e64777240",
  "results": 1,
  "page": 1,
  "version": "1.3"
 }
}
```


Obie aplikacje na początkowej stronie zawierają elementy:

- przycisk do tworzenia 1 użytkownika,
- przycisk do tworzenia 50 użytkowników,
- przycisk do tworzenia 100 użytkowników,
- przycisk do tworzenia 500 użytkowników,
- przycisk do tworzenia 1000 użytkowników,
- przycisk do tworzenia 10000 użytkowników.

Widok aplikacji po wygenerowaniu losowych użytkowników przedstawiono na Rysunku 3.



Rysunek 2: Widok aplikacji z wygenerowanymi użytkownikami.

Aplikacje uruchomiono na przeglądarce Google Chrome w wersji: 97.0.4692.71 (64-bitowa).

Badanie wydajności polegało na sprawdzeniu czasów renderowania żądanej liczby komponentów. Jeden jeden komponent odpowiada jednemu wygenerowanemu użytkownikowi.

Do badań wykorzystano stanowisko badawcze o następującej specyfikacji:

- procesor: Intel Core i5-7200U 2.5Ghz,
- pamięć: 8GB RAM,
- dysk: 240GB SSD,
- karta graficzna: GeForce 940MX 2GB,
- system operacyjny: Windows 10 Home,
- model: Acer Aspire F5-573G.

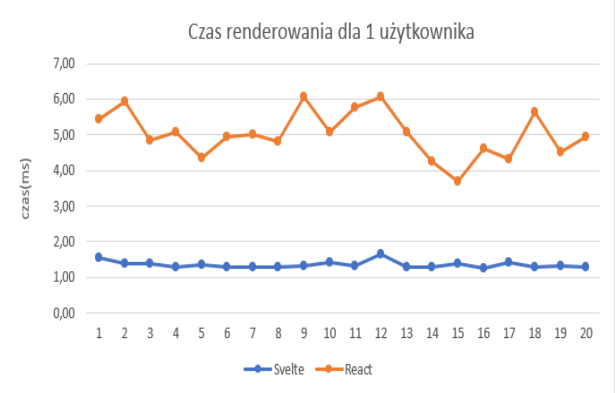
Testy wykonano według scenariuszy:

- Scenariusz 1 – pomiar czasu renderowania komponentów dla 1, 50, 100, 500, 1000 oraz 10000 użytkowników z wykorzystaniem metod console.time oraz console.timeEnd.
- Scenariusz 2 – czas wyszukiwania jednego użytkownika spośród 50, 100, 500, 1000 oraz 5000.

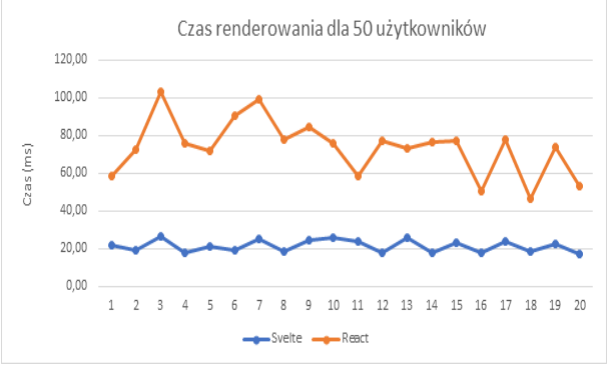
Ponadto porównano liczbę linii kodu, liczby folderów i plików oraz rozmiar gotowych aplikacjach.

7. Wyniki badań

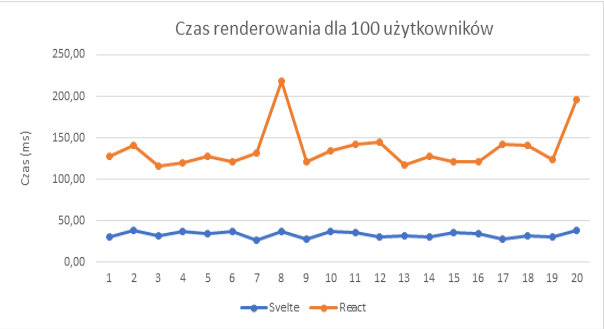
Wyniki dla scenariusza 1 przedstawione zostały na Rysunkach 3-8.



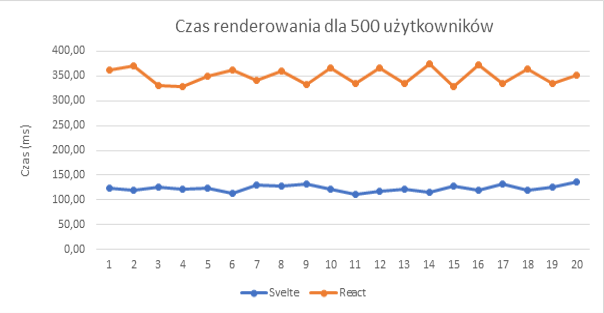
Rysunek 3: Wykres z czasami renderowania dla 1 użytkownika.



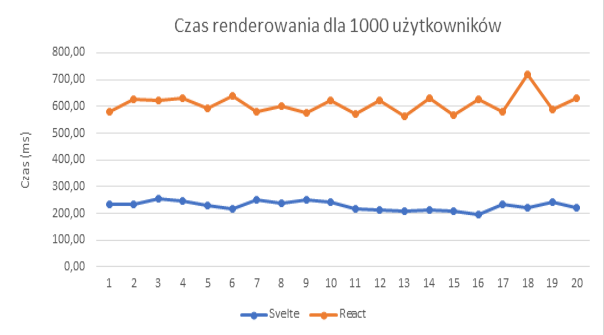
Rysunek 4: Wykres z czasami renderowania dla 50 użytkowników.



Rysunek 5: Wykres z czasami renderowania dla 100 użytkowników.



Rysunek 6: Wykres z czasami renderowania dla 500 użytkowników.

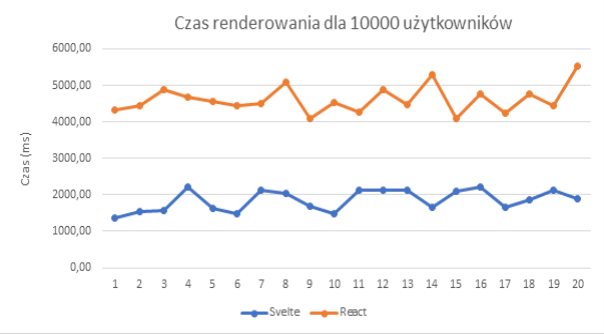


Rysunek 7: Wykres z czasami renderowania dla 1000 użytkowników.

Średnie wyniki przedstawia Tabela 1.

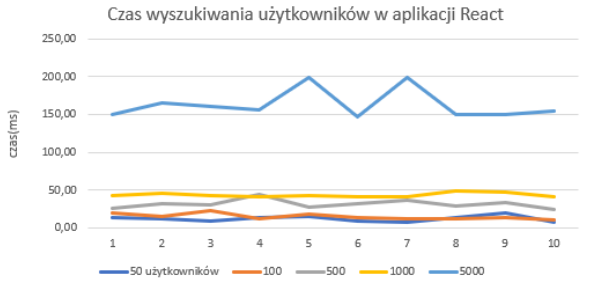
Tabela 1: Średnie czasy renderowania

Średnie czasy renderowania [ms]						
Liczba użytkowników	1	50	100	500	1000	10000
Svelte	1.36	21	33	123	228	1851
React	5	73	136	350	608	4617

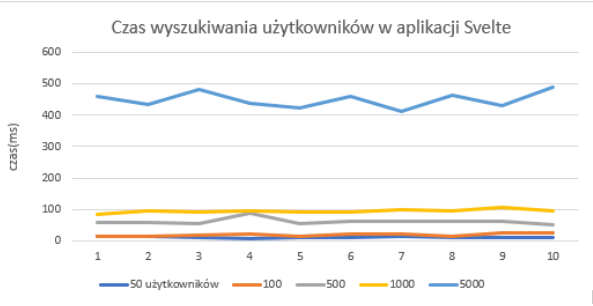


Rysunek 8: Wykres z czasami renderowania dla 10000 użytkowników.

Wyniki pomiarów dla scenariusza 2 przedstawiono na Rysunkach 9-10.



Rysunek 9: Czasy wyszukiwania użytkowników w aplikacji React.



Rysunek 10: Czasy wyszukiwania użytkowników w aplikacji Svelte.

Średnie czasy wyszukiwania użytkowników w obu aplikacjach przedstawiono w Tabeli 2.

Tabela 2: Średnie czasy wyszukiwania

Średnie czasy wyszukiwania [ms]					
Liczba użytkowników	50	100	500	1000	5000
Svelte	12	15	32	44	163
React	12	20	62	95	448

Metryki kodu i rozmiar obu aplikacji przedstawiono w Tabeli 3.

Tabela 3: Metryka kodu dla aplikacji React oraz Svelte

	React	Svelte
Liczba linii kodu w folderze src	381	317
Liczba linii kodu we wszystkich plikach	39652	7708
Liczba plików JavaScript	9	4
Liczba plików z rozszerzeniem .svelte	0	5
Liczba plików we wszystkich folderach	18	16
Rozmiar aplikacji	616 kB	352 kB

8. Analiza wyników

Analizując wyniki przedstawione na Rysunkach 3-8 można stwierdzić, że wydajność w generowaniu losowych użytkowników jest znacznie lepsza w przypadku aplikacji Svelte. Dla każdego przypadku ze scenariusza wykonano po 20 pomiarów i w żadnym z nich React nie był lepszy. Na wykresach można zauważyć, że w przypadku React pomiary nie były jednakowe, pojawiały się znaczne odchylenia pomiędzy testami, co oznacza, ze framework nie działa zbyt stabilnie. W przypadku Svelte nie było tak wielkich odchyleń, jedynie przy generowaniu 10000 użytkowników, pojawiały się różnice

odchyleń o maksymalnie 1 sekundę, natomiast odchylenia dla React wynosiły ponad 1,5 sekundy.

W przypadku scenariusza 2 badano czas wywoływania funkcji wyszukiwania użytkowników z wybranym imieniem. Wyniki były nieco inne niż w przypadku scenariusza 1, ponieważ to aplikacja React była wydajniejsza. W pomiarach wyszukiwania wśród 50, 100, 500 lub 1000 użytkowników były niewielkie, natomiast w przypadku wyszukiwania dla 5000 użytkowników, różnica czasów wynosiła średnio 284 ms, co daje wynik 3 razy lepszy dla aplikacji React.

Z Tabeli 3 wynika, że lepsze wyniki uzyskał Svelte, ponieważ w każdym przypadku zawiera mniej linii kodu, nie licząc przypadków z liczbą plików z rozszerzeniem .svelte. Aplikacja React korzysta tylko z plików JavaScript. Liczba linii kodu miała wpływ też na rozmiar aplikacji Svelte rozmiar aplikacji. Aplikacja React jest prawie 2 razy większa niż Svelte.

9. Wnioski

Na podstawie uzyskanych wyników z wykorzystaniem prostych aplikacji testowych, można stwierdzić, iż Svelte jest znacznie wydajniejszy od React dla przypadku renderowania komponentów. Dla liczby komponentów (100), aplikacja Svelte była nawet o 400% wydajniejsza niż aplikacja React. Przy większej liczbie komponentów (10000), aplikacja Svelte była szybsza o ponad 2 sekundy, czyli aż o 155%. Porównując odchylenia standardowe wyników można również stwierdzić lepszą stabilność aplikacji Svelte, ponieważ w wynikach dla aplikacji React wartości były bardziej rozproszone.

Aplikacja React była wydajniejsza dla przypadku wyszukiwania a największa różnica czasowa wynosiła średnio 300ms.

Liczba linii kodu oraz rozmiar aplikacji były znacznie mniejsze dla Svelte.

W artykule [4] autor analizował wydajność bibliotek Angular oraz Svelte porównując między innymi czasy renderowania komponentów w obu aplikacjach. We wnioskach wskazano, że aplikacja Svelte była 4-krotnie szybsza od aplikacji Angular. W niniejszym artykule porównywano React i Svelte a wnioski także wskazują na lepszą wydajność renderowania komponentów w przypadku Svelte.

React oraz Svelte są wydajnymi narzędziami, jednak w kwestii renderowania komponentów, Svelte jest wyraźnie wydajniejsze, co potwierdza założoną hipotezę.

Literatura

[1] Single Page Application (SPA) vs Multi Page Application (MPA): Pros and Cons, http://merehead.com/blog/single-page-application-vs-multi-page-application/, [23.05.2022].

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

[3] Popularność frameworków JavaScript w 2021 roku, https://2021.stateofjs.com/en-US/libraries/front-end-frameworks, [07.05.2022].

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

[5] J. Wróbel, Porównanie narzędzi do tworzenia aplikacji typu SPA na przykładzie Ember i React, Journal of Computer Sciences Institute 11 (2019) 145-148.

[6] S. Aggarwal, Modern Web-Development Using ReactJS, International Journal of Recent Research Aspects, 5(1) (2018) 133-137.

[7] G. Białecki, B. Pańczyk, Performance analysis of Svelte and Angular applications, Journal of Computer Sciences Institute 19 (2021) 139-143.

[8] Xu Wenquing, Benchmark Comparison of JavaScript Frameworks, M.Sc. Computer Science Interactive Digital Media, 2021.

[9] Dokumentacja React, https://reactjs.org/, [05.11.2021].

[10] Dokumentacja Svelte, https://svelte.dev/docs, [05.11.2021].

[11] O. Therox, Svelte i TypeScript, https://svelte.dev/blog/svelte-and-typescript, [05.11.2021].

[12] S. Kołodziejczak, Svelte – wszystko co powinieneś wiedzieć o nowej wersji tego narzędzia, https://geek.justjoin.it/svelte-frontend, [05.11.2021].

[13] T. Tolliday, Getting Acquainted With Svelte, the New Framework on the Block, https://css-tricks.com/getting-acquainted-with-svelte-the-new-framework-on-the-block/, [05.11.2021].

[14] Generator losowych użytkowników, https://randomuser.me/, [18.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 350–357
Received: 1 July 2022
Accepted: 31 August 2022

Application of universal design principles in the creation of websites oriented toward visually impaired persons

Zastosowanie zasad projektowania uniwersalnego w tworzeniu stron internetowych ukierunkowanych na osoby z niepełnosprawnością wzrokową

Paweł Sławomir Galiński*, Mateusz Klimkowicz*, Mariusz Dzieńkowski

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

Abstract
The subject of this study is to show the impact of universal design principles on everyday work with web content for people with visual impairments. The first part of the study is an experiment using an eye tracker, comparing the effects of using high contrast or changing the font size on the subjects' efficiency in performing simple tasks. The second part of the study is a survey comparing the feelings about using a website strictly compliant with universal design principles and its counterpart containing noticeable errors. The eye-tracking results were analyzed to assess time to first fixation, average fixation time and time spent on a designated area of interest. The results indicate an increase in efficiency and comfort with digital content designed according to universal design principles.

Keywords: universal design; user experience; eye tracking

Streszczenie
Celem pracy jest przedstawienie wpływu, jaki ma zastosowanie zasad projektowania uniwersalnego na codzienną pracę z treściami internetowymi dla osób z wadą wzroku. Pierwszą częścią badania jest eksperyment z wykorzystaniem okulografu, porównujący wpływ zastosowania wysokiego kontrastu lub zmiany rozmiaru czcionki na efektywność wykonywania przez osoby badane prostych zadań. Druga część badania to przeprowadzenie ankiety porównującej odczucia z korzystania ze strony internetowej ściśle spełniającej zasady projektowania uniwersalnego oraz jej odpowiednika zawierającego wyraźne błędy. Rezultaty badań okulograficznych zostały poddane analizie pod kątem oceny czasu do pierwszej fiksacji, średniego czasu fiksacji oraz czasu spędzonego na wyznaczonym obszarze zainteresowania. Wyniki badań wskazują na wzrost efektywności i komfortu z użytkowania treści cyfrowych zaprojektowanych zgodnie z zasadami projektowania uniwersalnego.

Słowa kluczowe: projektowanie uniwersalne; wrażenia użytkownika; okulografia

*Corresponding author
Email address: pawel.galinski@pollub.edu.pl (P. S. Galiński), mateusz.klimkowicz@pollub.edu.pl (M. Klimkowicz)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Współcześnie, społeczeństwo w coraz większym stopniu zwraca uwagę na problemy osób niepełnosprawnych w codziennym życiu. Bariery, które przed nimi stoją, dotyczą nie tylko sfery finansowej czy architektonicznej, ale również w dużej mierze mają związek ze sferą informatyczną. Przeglądanie internetu stało się codziennością aktualnej cywilizacji, co sprawia, że dostępność stron internetowych jest zagadnieniem niezbędnym dla osób słabo widzących. Twórcy treści muszą być świadomi potrzeb możliwie najszerszego grona odbiorców, tak by nie wykluczać osób ze szczególnymi potrzebami od dostępu do informacji i nowoczesnych usług. Celem projektowania uniwersalnego jest wyrównanie szans i uniknięcie dyskryminacji. Z tego powodu niezbędne są ciągłe badania nad usprawnieniem istniejących procesów i koncepcji, ponieważ ciągle wiele wiodących rozwiązań informatycznych nie uwzględnia wyzwań stojących przed osobami niepełnosprawnymi.

Pozytywne działania na rzecz osób niepełnosprawnych postępują wraz z rosnącą świadomością społeczeństwa popartą o zdobycze wiedzy. Celem tych poczynań jest dążenie do tego, by dostęp do informacji i możliwość uczestnictwa w cyfrowym życiu stały się normą i podstawowym prawem dla każdego człowieka. Wytyczne dotyczące dostępności treści cyfrowych WCAG (ang. Web Content Accessibility Guidelines) to zestaw dyrektyw stanowiących o treściach internetowych wysokiej jakości, tj. będących dostępnych dla jak największej grupy odbiorców ze szczególnym naciskiem na osoby niepełnosprawne. Ich zasady dostępne na trzech poziomach zgodności [1] stają się normami prawnymi na poziomie międzynarodowym [2, 3]. Zgodnie z ogólnie przyjętymi zasadami, treści cyfrowe powinny być więc użyteczne, elastyczne i intuicyjne.

W niniejszej pracy główny nacisk postawiony jest na dolegliwościach osób niedowidzących, które zmuszone są do korzystania z komputera przy pomocy okularów. Analizie zostały poddane istniejące rozwiązania, które w różny sposób zwiększają użyteczność interfejsów stron internetowych. Były to dwie funkcjonalności: zmiany rozmiaru wyświetlanej na ekranie czcionki oraz możliwość zmiany kontrastu. Na podstawie przeprowadzonych badań zestawiających wykorzystanie tych rozwiązań z treściami nieprzestrzegającymi wytycznych WCAG, skupiono się na określeniu, w jakim stopniu

zmienia się efektywność pracy w internecie osoby z niepełnosprawnością wzrokową w zależności od warunków ich styczności z przedstawianymi treściami. Celem przedstawionych badań jest odpowiedź na pytanie badawcze: „Czy zastosowanie zasad projektowania uniwersalnego przy tworzeniu stron internetowych przyspiesza korzystanie z niej osobom z wadą wzroku?". W tym celu postawiono trzy hipotezy weryfikowane poprzez zrealizowane badania:

H1. Użytkownik z wadą wzroku będzie wykonywał zadane czynności szybciej na stronie z kontrastem spełniającym wymogi standardu WCAG AAA niż na stronie niespełniającej wymogów WCAG AA.
H2. Strony opracowane zgodnie z zasadami projektowania uniwersalnego są lepiej oceniane przez użytkowników i w związku z tym osiągają lepsze wskaźniki współczynnika WUP (ang. Web Usability Points).
H3. Zwiększenie rozmiaru czcionki wpływa w wyższym stopniu na szybkość realizacji zadań na stronie internetowej przez osoby z wadą wzroku niż zwiększenie na niej kontrastu.

Na potrzeby artykułu dokonano przeglądu literatury oraz sformułowano cel badań. Opracowana strona internetowa służąca jako obiekt badań, jest witryną symulującą działanie rzeczywistej strony informacyjnej prowadzonej przez firmę taksówkarską w jednym z polskich miast. Na jej podstawie przeprowadzono badania z wykorzystaniem okulograf na osobach mających krótkowzroczność. Dodatkowo wykorzystano listę kontrolną LUT (ang. Lublin University of Technology) [4, 5], umożliwiającą ocenienie przez użytkowników jakości interfejsu strony internetowej. Pozwoliło to dokładniej poznać odczucia odbiorców treści internetowych, dzięki otrzymanej informacji zwrotnej pogrupowanej według konkretnych obszarów analizowanych witryn.

2. Przegląd literatury

Na potrzeby realizacji badania, należy zapoznać się dokładniej z definicją projektowania uniwersalnego, jakie kryteria świadczą o uniwersalności oraz trzeba zrozumieć codzienne problemy i realia życia osób z niepełnosprawnością wzrokową. W artykule [6] autorzy określili proces projektowania uniwersalnego jako tworzenie produktów pozostających funkcjonalnymi dla możliwie najszerszej grupy użytkowników. Dostępność na ogół jest definiowana jako projektowanie pod kątem osób z niepełnosprawnościami. W artykule podkreślono ważność tego, aby dostępność nieustannie koncentrowała się na osobach niepełnosprawnych, jednocześnie integrując się z projektowaniem uniwersalnym stron internetowych dla wszystkich użytkowników. Artykuł zawiera opis działań World Web Consortium (W3C) i Web Accessibility Initiative (WAI) nad potrzebami użytkowników we wszystkich technologiach sieciowych. Autorzy we wnioskach stwierdzają, że skupienie się na osobach niepełnosprawnych zapobiega pominięciu ich szczególnych potrzeb pośród pozostałych problemów. Zwrócono również uwagę, że zintegrowane badania i rozwój zagadnień dotyczących rozwoju dostępności, wspomaga również ogół społeczeństwa. Poza kwestią dostępności treści internetowych za pośrednictwem komputerów bądź laptopów, w dzisiejszych czasach równie ważne jest wzięcie pod uwagę również rynku mobilnego. Artykuł [7] opisuje, w jaki sposób postępujący rozwój usług sieciowych oddziałuje na wytyczne definiujące kryteria dostępności w podstawowych technologiach internetowych takich jak HTML (ang. HyperText Markup Language) i CSS (ang. Cascading Style Sheets). Zdefiniowano podstawowe komponenty określające dostępność w sieci takie jak: treści, technologie sieciowe, narzędzia wytwórcze i oszacowujące dostępność, technologie wspomagające oraz użytkowników. Stwierdzono ogólny wzrost świadomości i zrozumienia potrzeby projektowania uniwersalnego jednak rosnące znaczenie rynku mobilnego stawia nowe wyzwania w tej kwestii. W tym celu została opracowywana specyfikacja Independent User Interface (IndieUI), ułatwiająca aplikacjom internetowym pracę z różnego rodzaju kontekstów. Zdarzenia zostają obsługiwane bez konieczności rozpoznania jak je wywołano, co pozytywnie wpływa na dostępność. Autorzy stwierdzają, że istnieje potrzeba dogłębnych badan i analiz dotyczących wsparcia dla dostępności każdego współzależnego komponentu stron internetowych, szczególnie w kontekście mobilnym.

Do ważnej obserwacji doszli autorzy artykułu [8], w którym stwierdzają, że obecnie w podejściu do projektowania uniwersalnego, główną uwagę zajmuje niepełnosprawność, a nie zdolność. Z tego powodu proponują modyfikację tej postawy w podejście nazwane „ability-based design", polegające na skupieniu się w procesie projektowania na zdolnościach użytkownika w przeciwieństwie do jego mankamentów. Zdefiniowane zasady takiego projektowania to kolejno: zdolność, poczytalność, adaptacja, przejrzystość, wydajność, detekcja kontekstu i łatwość dostępu do sprzętu. Usprawnienie procesu projektowania o te reguły pozwoliło badaczom zaobserwować poprawę satysfakcji użytkowników z powodu korzystania z typowych urządzeń, jednak zaktualizowanych o oprogramowanie dostosowane do ich możliwości.

W kolejnym artykule [9] zwrócono uwagę na fakt, że koncepcja dostępności cyfrowej różni się w zależności od kultury czy też docelowej grupy odbiorczej. Pod określeniem „projektu dla wszystkich", „uniwersalnego dostępu" czy „projektowania globalnego", kryje się jedna filozofia tworzenia treści dla możliwie jak największej liczby zastosowań danej usługi. Autorzy, starając się odpowiedzieć czy te określenia różnią się od siebie, zwracają uwagę, że spośród 400 przebadanych publikacji, w większości z nich brakuje jednoznacznej definicji dostępności, czy chociaż odnośnika do precyzyjnego sformułowania tego terminu. Wnioskiem badania jest brak spójności omawianych definicji pomiędzy regionami całego świata. Z tego powodu autorzy opowiadają się, za dokładniejszym zdefiniowaniem pojęcia projektowania uniwersalnego, co może mieć pozytywny wpływ na zgodność z normami całej sieci i przyszły rozwój treści.

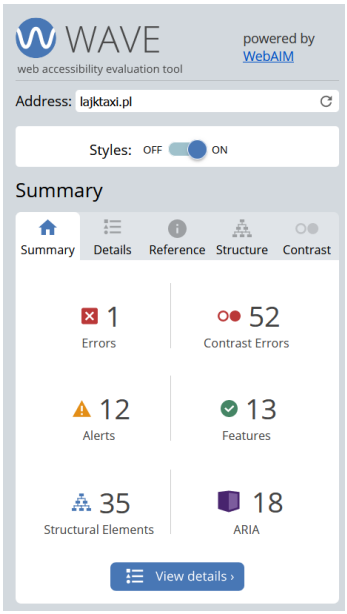
Koncentrując się nad aplikacjami internetowymi projektowanymi uniwersalnie, autorzy artykułu [10] przedstawiają proces refaktoryzacji jako technikę poprawy użyteczności interfejsów stron internetowych dla użytkowników z niepełnosprawnościami. Autorzy prezentują sposoby radzenia sobie dużych korporacji z problemem dostępności ich usług. Zrobili to na przykładzie analizy dwóch aplikacji poczty elektronicznej. Skupiono się na refaktoringu interfejsu, który ma poprawić użyteczność strony dla osób korzystających z czytników ekranowych, bez uszczerbku na funkcjonalności dla pozostałych odbiorców. Proces określono jako sposób na zachęcenie programistów do ponownego odkrywania sposobów ulepszania interfejsów, biorąc pod uwagę nie tylko biegłych użytkowników, ale również osoby niedoświadczone bądź niepełnosprawne. Przeanalizowane rozwiązania pozwoliły wywnioskować, że refaktoryzacja strony do stanu przystępnego dla każdej osoby, przy użyciu możliwie jak najmniejszej liczby zasobów powinna motywować menedżerów do inwestycji w tej kwestii oraz do zmiany polityki dostępności usług. Z tego powodu naukowcy mają za zadanie opracowywanie coraz skuteczniejszych technik pomagającym programistom w osiągnieciu tych celów. Tę kwestię podjęto również w artykule [11]. Autorzy podają cztery sposoby na poprawę dostępności witryn opartych na systemach zarządzania treścią typu Joomla! oraz WordPress. Pierwszy stosuje wtyczkę, która wyszukuje problemy ze stroną oraz pozwala użytkownikowi w wygodny sposób na wprowadzenie brakujących atrybutów na stronie. Według autorów ta metoda jest odpowiednia dla użytkowników, którzy nie mają doświadczenia z informatyką. Dwie kolejne metody opierają się na przeciążeniu kodu HTML lub CSS/SCSS (ang. Sassy CSS) z wykorzystaniem wtyczki. Ostatni sposób polega na znalezieniu wtyczki odpowiedzialnej za niepoprawne działanie strony oraz ułatwia jej poprawienie. Według autorów ta ostatnia metoda jest zalecana tylko dla użytkowników z doświadczeniem w tworzeniu aplikacji internetowych.

Prowadząc badanie nad zastosowaniem zasad projektowania uniwersalnego, warto zapoznać się z codziennością osób z ograniczeniami. Opierając się na wynikach badań, w których wykazano, że problemy ze wzrokiem źle wpływają na psychikę młodzieży, opracowano artykuł [12], którego celem jest badanie stopnia lęku i depresji wśród grupy osób z wrodzoną ślepotą. Czterdzieści osób niewidomych wypełniło ankietę z wykorzystaniem alfabetu Braille'a, ich odpowiedzi zestawiono z wynikami otrzymanymi od grupy o tej samej wielkości zdrowych osób. Wykazano, że poziom depresji nie zależał od sprawności osoby badanej, zależność ta występowała jednak wyraźnie w kwestii lęków. Autorzy sugerują, że strach w dużym stopniu jest powodowany problemami z orientacją w terenie, co wiąże się z obawą przed wypadkiem. Można stąd wywnioskować, że redukcja barier przed osobami z niepełnosprawnością, jest moralnym obowiązkiem społeczeństwa, we wszelkich dziedzinach życia.

3. Metoda badawcza

3.1. Opis przedmiotu badań

Badania przeprowadzono z wykorzystaniem dwóch stron internetowych: LajkTaxi.pl [13] oraz TaxiMaxi [14]. Pierwsza ze stron została wybrana na podstawie analizy dostępności 20 losowo wyszukanych stron internetowych oferujących usługi taksówkarskie. Ocena ich dostępności została zrealizowana za pomocą zdalnego narzędzia WAVE [15, 16], które służy twórcom treści internetowych do weryfikacji stanu dostępności projektowanych przez nich serwisów.



Rysunek 1: Podsumowanie analizy strony internetowej za pomocą narzędzia WAVE.

Na potrzeby badania, ze zbioru przeanalizowanych stron wybrano tę, w której wykryto największą liczbę nieprawidłowości, które mogą mieć wpływ na komfort użytkowania z wadą wzroku (Tabela 1). Strona, która została z nią porównana to TaxiMaxi. Została ona przygotowana od podstaw z wykorzystaniem systemu typu CMS (ang. Content Management System) o nazwie WordPress [17]. System ten wyróżnia się dużą łatwością wdrażania nowych serwisów. Z wykorzystaniem szeroko rozbudowanej biblioteki wtyczek [18], można efektywnie od podstaw zaprojektować oraz wdrożyć witrynę internetową o wybranej tematyce, którą następnie można w szybki sposób usprawniać oraz rozwijać o dodatkowe możliwości. Dzięki tym zaletom treść oraz wygląd strony można w łatwy sposób modyfikować, aby dostosować ją pod kątem różnych aspektów dostępności. Umożliwiło to przygotowanie strony spełniającej wymagania WCAG 2.1 oraz dynamiczne modyfikowanie jej kompozycji na potrzeby kolejnych prób ukierunkowanych na weryfikację hipotez H1 i H3.

Obie omawiane strony internetowe zostały wykorzystane w celu weryfikacji hipotezy H2. Na jej potrzeby poddano ankietyzacji 20 osób przy użyciu listy kontrolnej LUT [4, 5] będącej listą opracowaną na podstawie heurystyk Nielsena-Molicha [5]. Autorska strona inter-

netowa - TaxiMaxi, została użyta do przebadania 23 osób przy użyciu okulografu.

Tabela 1: Rezultat analizy narzędziem WAVE porównywanych stron

Podstrona	Rodzaj	Liczba wystąpień na stronie LajkTaxi.pl	Liczba wystąpień na stronie TaxiMaxi
Strona główna	Błąd	1	3
	Błąd kontrastu	46	0
	Alert	12	8
	Właściwości	13	7
	Aria	10	24
	Elementy strukturalne	35	27
Usługi	Błąd	6	3
	Błąd kontrastu	23	0
	Alert	14	6
	Właściwości	12	11
	Aria	11	22
	Elementy strukturalne	25	21
Cennik	Błąd	1	4
	Błąd kontrastu	81	0
	Alert	75	7
	Właściwości	9	3
	Aria	11	21
	Elementy strukturalne	18	15
Galeria	Błąd	1	3
	Błąd kontrastu	10	0
	Alert	10	26
	Właściwości	9	23
	Aria	11	23
	Elementy strukturalne	17	21
Kontakt	Błąd	4	3
	Błąd kontrastu	25	0
	Alert	10	10
	Właściwości	11	10
	Aria	9	24
	Elementy strukturalne	24	23

3.2. Ankietowanie za pomocą listy kontrolnej LUT

Na potrzeby badania wybrano 20 osób. Ankietowani byli osobami doświadczonymi w korzystaniu z serwisów internetowych. Grupa składała się z osób różnej płci w przedziale wiekowym od 21 do 50 lat. Grupa badawcza składająca się z 20 osób pozwala wykryć niemal wszystkie problemy. Nawet 5 osób, iteracyjnie przeprowadzając badanie, jest w stanie z 80% skutecznością wykryć trapiące stronę błędy [5].

Ankieta zawierająca pytania z listy kontrolnej LUT została wykonana przy pomocy Google Forms [19]. Każdy z ankietowanych otrzymał 2 formularze, każdy z nich dotyczył osobnej, porównywanej witryny internetowej. Struktura ankiety składa się z 32 pytań pogrupowanych według 5 obszarów:

- nawigacja i struktura,
- komunikaty, feedback, pomoc dla użytkownika,
- interfejs aplikacji,
- treść podstron,
- wprowadzanie danych.

Inspekcja wykonana za pomocą listy kontrolnej przez anonimowe osoby pozwala, na poznanie odczuć rzeczywistych użytkowników strony. Po otwarciu i skorzystaniu z podanych witryn oceniający wyrażają opinię odnośnie kolejnych zagadnień, w skali od 1 (najniższa ocena) do 5 (ocena najwyższa). Wykorzystanie listy kontrolnej LUT pozwala na wyznaczenie współczynnika WUP [4], dzięki któremu istnieje możliwość porównania między sobą obu badanych stron internetowych. Metryka jest wyznaczana za pomocą wzoru 1:

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

gdzie n_a to liczba obszarów, s_i to liczba podobszarów w obszarze i, q_{ij} to liczba pytań w obszarze i oraz podobszarze j, z kolei p_{ijk} to ocena pytania o numerze k w podobszarze j obszaru i [5]. Wysoka wartość współczynnika WUP oznacza dobrze zaprojektowany interfejs, im ta wartość jest niższa, tym gorzej przebiega współpraca z badaną stroną.

3.3. Badanie z wykorzystaniem okulografu

Do analizy hipotez H1 i H3 wykorzystano okulograf. Grupą badawczą były 24 osoby ze stwierdzoną krótkowzrocznością, na co dzień pracujące w okularach korekcyjnych [20]. Byli to studenci w wieku od 20 do 26 lat. Każda z osób miała duże doświadczenie w użytkowaniu internetu. W badanej grupie 19 osób miało wadę wzroku w zakresie od -1 do -3 dioptrii, 5 osób charakteryzowało się wyższą wadą w zakresie od -3 do -6 dioptrii. Na potrzeby doświadczenia, eksperyment z użyciem okulografu, realizowany był przez osoby bez założonych okularów korekcyjnych. Miało to na celu weryfikację, w jakim stopniu narzędzia wspomagające osoby z wadą wzroku, rzeczywiście usprawniają korzystanie z internetu. Każdy z badanych przeszedł wstępne szkolenie, przedstawiające sposób realizacji eksperymentu oraz zaznajamiające go z oprzyrządowaniem.

Badanie wykorzystywało wyłącznie autorską stronę internetową TaxiMaxi [14], która podległa modyfikacjom.

Rysunek 2: Przykładowy fragment strony z kontrastem spełniającym wymogi WCAG AAA.

Rysunek 3: Fragment strony z kontrastem niespełniającym wymogów WCAG AA.

Na potrzeby weryfikacji hipotezy H1, elementy zaprojektowane zgodnie z zasadami projektowania uniwersalnego (Rysunek 2), przekształcono w taki sposób, aby nie spełniały wymogów WCAG na poziomie AA m.in. poprzez zmniejszenie rozmiaru czcionki bądź obniżenie kontrastu (Rysunek 3). W drugim przypadku, w celu weryfikacji hipotezy H3, stronę zmodyfikowano tak, aby prezentowana treść miała podwyższony kontrast bądź czcionka była wyświetlana w większym rozmiarze.

Badanie przebiegało w postaci pokazu slajdów, w którym najpierw osobie badanej przedstawiane było polecenie do zrealizowania. Po zapoznaniu się z treścią zadania, badany przechodził do następnego slajdu zawierającym zrzut ekranowy fragmentu strony internetowej. Gdy badany wykonał dane polecenie, przechodził do kolejnego. W sumie każda osoba badana otrzymała zestaw 15 zadań. W każdym z nich prezentowane fragmenty stron były wyświetlone w innej formie po to, aby zminimalizować zapamiętywanie przez badanych ich treści i struktury. Dodatkowo, w celu wyeliminowania efektu uczenia, eksperyment zaprojektowano w formie testu porównawczego A/B [21]. Dzięki temu można było zbadać czas realizacji zadań na tych samych fragmentach stron internetowych, ale prezentowanych różnych jej formach. Grupa badawcza została podzielona na dwie równe podgrupy po 12 osób. Grupa A otrzymała zestaw pytań, na które udzielała odpowiedzi po obejrzeniu fragmentu strony internetowej zaprojektowanej zgodnie z zasadami projektowania uniwersalnego (H1) lub z podwyższonym kontrastem (H3). Grupa B otrzymała taki sam zestaw zadań, ale przedstawione fragmenty strony internetowej były niezgodne z zasadami projektowania uniwersalnego (H1) lub miały powiększoną czcionkę (H3).

Eksperyment zrealizowano w Katedrze Informatyki Politechniki Lubelskiej, zapewniając badanym komfortowe warunki pracy oraz odpowiednie warunki oświetlenia. Osoba kontrolująca przebieg doświadczenia, zaznajamiała badanego ze scenariuszem badania, a następnie zostawiała osobę badaną samodzielnie na stanowisku, monitorując przebieg procesu z pewnej odległości. Dzięki temu starano się wyeliminować wpływ stresu wynikającego z udziału moderatora, a jednocześnie zachować kontrolę nad prawidłowością wykonywania czynności.

Sala, w której prowadzono badania, została wyposażona w laptop z ekranem 17″ w rozdzielczości 1920x1080 wyświetlający obraz z wysoką częstotliwością 144 Hz. Okulograf zdalny (Rysunek 4), będący na wyposażeniu stanowiska laboratoryjnego to Gazepoint GP3 HD [22], który dzięki swojej konstrukcji jest lekki, przenośny i łatwy w konfiguracji [23]. Na stanowisku badawczym okulograf stał pomiędzy ekranem laptopa, a twarzą osoby badanej. Przed rozpoczęciem eksperymentu każda osoba musiała zostać skalibrowana, aby zapewnić jak najdokładniejszy pomiar, eliminując nieprawidłowości mogące wynikać np. z różnic budowy oka badanego. Doświadczenie rozpoczynano, gdy kalibracja została wykonana na poziomie „Excellent”. Za pomocą oprogramowania iMotions [24] podłączonego do okulografu, dokonano analizy zgromadzonych danych. Z poziomu programu dostępne są nagrania każdej osoby uczestniczącej w eksperymencie i realizującej scenariusz badawczy. Tuż po zakończeniu badania istnieje możliwość wygenerowania ścieżek skanowania, map cieplnych oraz analizy z wykorzystaniem tzw. obszarów zainteresowania (AOI, ang. Area of Interest).

Rysunek 4: Okulograf Gazepoint GP3 HD [22].

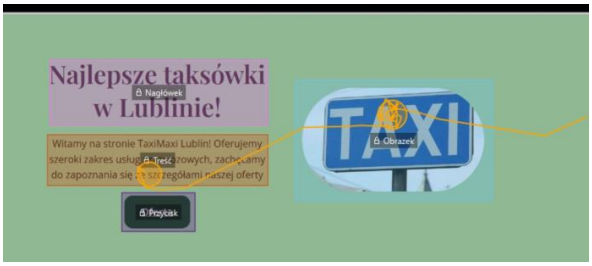
Podczas analizy danych, wzięto pod uwagę następujące miary [25]:

- czas do pierwszej fiksacji (TTFF - ang. Time to First Fixation) - ilość czasu, jaki zajęło osobie badanej spojrzenie na wyznaczony obszar zainteresowania,
- czas przebywania (ang. dwell time) - ilość czasu jaką respondent poświęcił na obserwowanie zadanego obszaru zainteresowania,
- czas trwania pierwszej fiksacji (ang. first fixation duration) – dostarcza informacji, jak długo trwała pierwsza fiksacja.

4. Wyniki badań

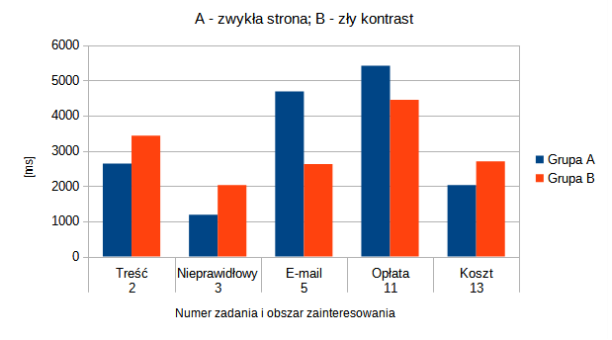
4.1. Weryfikacja hipotezy H1

Przedstawione w tym rozdziale wykresy dotyczą weryfikacji hipotezy H1. Rysunek 6 dotyczy zadań, w których należało znaleźć elementy takie jak treść, przyciski lub pole do wprowadzenia tekstu na zaprezentowanych kolejno stronach. Na rysunku 5 zobrazowano sposób, w jaki wyznaczono poszczególne obszary zainteresowania w programie iMotions, które następnie zostały poddane analizie.



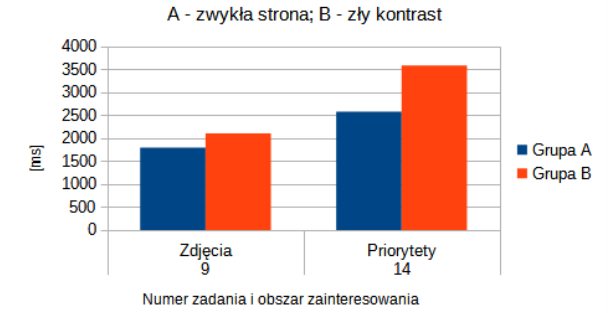
Rysunek 5: Obszary zainteresowania wyznaczone dla jednego z pytań z zestawu.

Z wykresu (Rysunek 6) można wywnioskować, że w przypadku trzech zadań (2, 3 i 13), na stronach zaprojektowanych zgodnie z zasadami projektowania uniwersalnego, użytkownicy byli w stanie szybciej zrealizować polecenia. Jednak, w przypadku dwóch zadań oznaczonych numerem 5 i 11, wyniki okazały się inne od spodziewanych.



Rysunek 6: Średnia arytmetyczna czasów do pierwszej fiksacji dla dwóch grup badawczych dot. wybranych zadań.

W przypadku zadań nr 9 i 14, w których badani byli proszeni o policzenie elementów prezentowanych na stronie, czasy przebywania w określonych obszarach były krótsze w grupie A – uczestników, którym wyświetlano strony dostępne niż w grupie B (Rysunek 7). Między zwykłą stroną, a stroną z małym kontrastem wystąpiła szczególnie duża różnica podczas realizacji zadania nr 14.



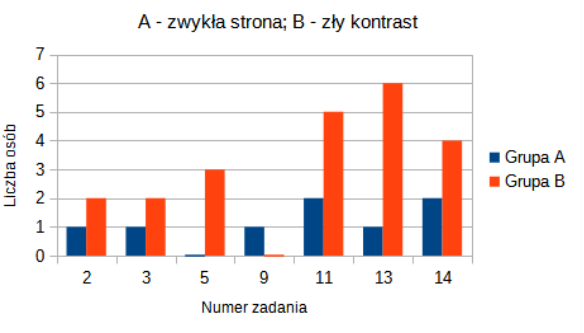
Rysunek 7: Średnia arytmetyczna czasów przebywania w poszczególnych obszarach zainteresowania dla dwóch grup badanych dla poleceń nr 9 i 14.

Także analiza ścieżek ruchu oczu wykazała, że wzrok ankietowanych był zdecydowanie bardziej chaotyczny przy wykonywaniu poleceń na stronach z nieprawidłowym kontrastem. Natomiast porównanie obrazów map cieplnych, pokazane na Rysunkach 8 i 9, pozwala wysnuć wniosek, że badani musieli ze znacznie większą intensywnością skupiać wzrok na niewyraźnym tekście.

Rysunek 8: Mapa cieplna dla zadania nr 14 na stronie z prawidłowym kontrastem.

Kolejnym ważnym spostrzeżeniem podczas realizacji zadań było to, że respondenci nie byli w stanie zrealizować 12,2% poleceń w przypadku stron ze złym kontrastem (Rysunek 10). Na stronie spełniającej wymagania WCAG współczynnik niewykonanych zadań wyniósł 4,4%.

Rysunek 9: Mapa cieplna dla zadania nr 14 na stronie z nieprawidłowym kontrastem.



Rysunek 10: Liczba osób, które nie były w stanie zrealizować poszczególnych zadań.

Zestawienie znajdujące się w Tabeli 2 pokazuje, że brak wystarczającego kontrastu może powodować, że użytkownicy internetu mogą nie być w stanie korzystać ze strony internetowej do realizacji swoich celów.

Tabela 2: Liczba osób w poszczególnych grupach, które nie były w stanie zrealizować zadań

	Grupa A	Grupa B
Liczba niewykonanych zadań	8	22

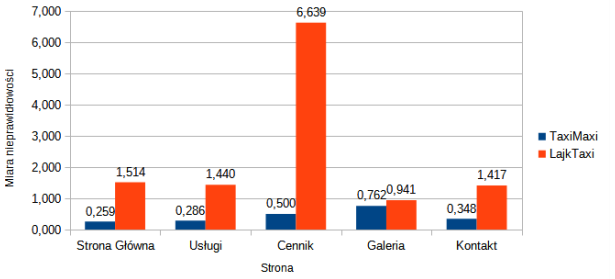
4.2. Weryfikacja hipotezy H2

4.2.1. Miara nieprawidłowości

Biorąc pod uwagę wyniki z analizy stron LajkTaxi.pl oraz autorskiej strony TaxiMaxi narzędziem WAVE (Tabela 1), wywnioskowano, że rezultaty badania mogą znacząco się różnić w zależności od wielkości i złożoności strony. Z tego powodu zaproponowano wzór wskazujący na to, jaki jest stopień niepoprawności wykonania strony względem jej zawartości:

$$miara\ nieprawidłowości = \frac{e + c + 0.5 * a}{s} \quad (2)$$

gdzie: e oznacza liczbę błędów na stronie, c liczbę błędów kontrastu, a to liczba alertów, z kolei s to zsumowana liczba elementów strukturalnych strony. Im wyższy jest otrzymany współczynnik, tym więcej zdiagnozowano nieprawidłowości na stronie. Na podstawie wzoru (2) policzono miarę nieprawidłowości dla porównywanych stron.



Rysunek 11: Miara nieprawidłowości badanych stron.

Przedstawione na Rysunku 11 wyniki, w których wzięto pod uwagę ilość treści znajdującej się na każdej z podstron, strona TaxiMaxi zaprojektowana zgodnie z zasadami projektowania uniwersalnego, w każdym przypadku otrzymała niższą wartość użytego współczynnika od podstron strony LajkTaxi.pl.

4.2.2. Współczynnik WUP

Z kolei współczynnik WUP dla strony TaxiMaxi jest znacznie wyższy od współczynnika WUP strony LajkTaxi (Tabela 3). Na podstawie wyników z przeprowadzonej ankiety oraz analizy z wykorzystaniem narzędzia internetowego WAVE, można stwierdzić, że hipoteza H2 jest prawdziwa.

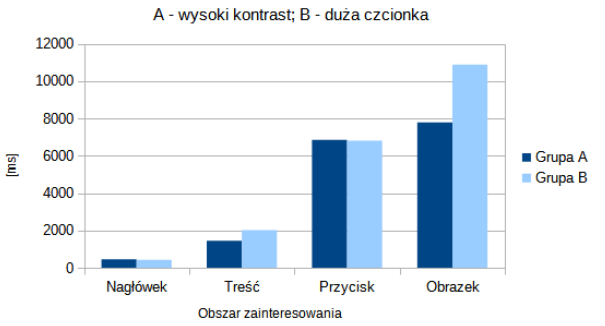
Tabela 3: Współczynnik WUP

Strona internetowa	LajkTaxi.pl	TaxiMaxi
Współczynnik WUP	3,71±0,24	4,45±0,13

Współczynnik obliczony na podstawie wzoru (1) wynosi 4,45±0,13 dla strony zgodnej z paradygmatem projektowania uniwersalnego oraz 3,71±0,24 dla strony porównywanej. Płynie z tego taki wniosek, że wyższą jakość pod względem dostępności ma strona prototypowa utworzona na potrzeby badań. Użytkownicy odbierali ją jako bardziej dopracowaną i odczuwali mniejszy dyskomfort w trakcie korzystania. Najwyższe różnice w ocenach wystąpiły w kwestiach związanych z graficznym interfejsem aplikacji oraz z komunikatami i informacjami zwrotnymi przekazywanymi dla użytkownika.

4.3. Weryfikacja hipotezy H3

Na Rysunku 12 przedstawiono wyniki eksperymentu zrealizowanego za pomocą eyetrackera, w których porównywano czasy trwania pierwszej fiksacji na elementach strony takich jak: nagłówek, treść, przycisk i obrazek. Grupa badawcza A oglądała strony w wysokim kontraście (Rysunek 13), natomiast grupa B patrzyła na strony z powiększoną czcionką (Rysunek 14). W przypadku nagłówka i przycisku wyniki dla obu grup były bardzo podobne. Dla obszaru zawierającego kontent strony oraz obrazek wyniki dla obu grup wyraźnie były lepsze dla stron z dużym kontrastem niż z powiększoną czcionką.



Rysunek 12: Średnia arytmetyczna czasów pierwszej fiksacji na wybranych elementach stron z dużym kontrastem (Grupa A) oraz powiększoną czcionką (Grupa B).

Zrzuty ekranowe z Rysunków 13 i 14 przedstawiają strony główne badanych serwisów wyświetlane odpowiednio dla grupy A i grupy B. Analiza ścieżek skanowania wykazała, że badani skanowali strony w kolejności: nagłówek, treść, przycisk oraz obrazek.

Rysunek 13: Strona z podwyższonym kontrastem.

Rysunek 14: Strona z powiększoną czcionką.

5. Wnioski

Celem artykułu była analiza wpływu korzystania ze stron utworzonych zgodnie z paradygmatem projektowania uniwersalnego. W badaniach zastosowano ankietę LUT oraz okulograf, które zrealizowano na dwóch grupach osób z wadą wzroku. Na podstawie uzyskanych wyników udowodniono, że zastosowanie zasad projektowania uniwersalnego ułatwia użytkownikom korzystanie ze stron internetowych. Poza tym eksperymentalnie potwierdzono, że ze strony z wysokim kontrastem korzysta się szybciej niż ze strony z dużą czcionką. Analiza wykazała także, że pomimo koncentracji deweloperów stron na ich zgodności z zasadami projektowania uniwersalnego, pewnych niedoskonałości nie sposób było uniknąć. Oznacza to, że dobrą praktyką jest już na etapie tworzenia strony dodatkowe jej sprawdzenie za pomocą automatycznych walidatorów (np. WAVE).

Literatura

[1] A. Kirkpatrick, J. O Connor, A. Campbell, M. Cooper, Web Content Accessibility Guidelines (WCAG) 2.1, 2018.

[2] Ustawa o dostępności cyfrowej stron internetowych, https://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20190000848/T/D20190848L.pdf, [12.05.2022].

[3] Dyrektywa Parlamentu Europejskiego i Rady (UE) 2016/2102 z dnia 26 października 2016 r. w sprawie dostępności stron internetowych i mobilnych aplikacji organów sektora publicznego, https://eur-lex.europa.eu/legal-content/PL/TXT/HTML/?uri=CELEX:32016L2102&from=PL, [12.05.2022].

[4] M. Milosz, M. Borys, M. Laskowski, Memorability Experiment Vs. Expert Method in Websites Usability Evaluation, ICEIS, 2013.

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

[6] S. L. Henry, S. Abou-Zahra, J. Brewer, The role of accessibility in a universal web, W4A 2014 - 11th Web for All Conference (2014), https://doi.org/10.1145/2596695.2596719.

[7] S. Abou-Zahra, J. Brewer, S. L. Henry, Essential components of mobile web accessibility, W4A 2013 - International Cross-Disciplinary Conference on Web Accessibility (2013) 1-4, https://doi.org/10.1145/2461121.2461138.

[8] J. O. Wobbrock, S. K. Kane, K. Z. Gajos, S. Harada, J. Froehlich, Ability-based design: Concept, principles and examples, ACM Transactions on Accessible Computing 3(3) (2011) 1-27, https://doi.org/10.1145/1952383.1952384.

[9] H. Persson, H. Ahman, A. A. Yngling, Universal design, inclusive design, accessible design, design for all: different concepts-one goal? On the concept of accessibility-historical, methodological and philosophical aspects, Universal Access in the Information Society 14 (2015) 505-526, https://doi.org/10.1007/s10209-014-0358-z.

[10] A. Garrido, G. Rossi, N. M. Medina, Improving accessibility of Web interfaces: refactoring to the rescue, Universal Access in the Information Society, 13(4) (2014) 387-399, https://doi.org/10.1007/s10209-013-0323-2.

[11] B. Csontos, I. Heckl, Improving accessibility of CMS-based websites using automated methods, Univ Access Inf Soc 21 (2022) 491–505, https://doi.org/10.1007/s10209-020-00784-x.

[12] N. Bolat, B. Dogangun, M. Yavuz, T. Demir, L. Kayaalp, Depression and anxiety levels and self-conceptcharacteristics of adolescents with congenital complete visual impairment, Turk Psikiyatri Derg Summer 22(2) (2011) 77-82.

[13] Strona internetowa LajkTaxi, https://lajktaxi.pl/, [14.01.2022].

[14] Autorska strona tworzona zgodnie z zasadami projektowania uniwersalnego, https://dev-pawelgal.pantheonsite.io/, [14.01.2022].

[15] WAVE Web Accessibility Evaluation Tool, https://wave.webaim.org/ , [18.11.2021].

[16] A. Sharma, R. K. Choudhary, Web Accessibility of Indian University Library Website: An Evaluation with WAVE Website Evaluation Tool, Library Philosophy and Practice, 2021.

[17] Zagadnienie dostępności w Wordpress, https://make.wordpress.org/accessibility/handbook/, [22.09.2021].

[18] Wtyczka Wordpress: Wp Accessibility, https://wordpress.org/plugins/wp-accessibility/, [22.09.2021].

[19] Formularze Google, https://docs.google.com/forms/, [14.01.2022].

[20] Pozwolenie na przeprowadzenie eksperymentu z udziałem ludzi: "Pozytywna opinia Komisji ds. Etyki Badań Naukowych nr 9/2019 z dnia 27.05.2019 r. dotycząca badań użyteczności oraz ergonomii interfejsów.

[21] R. Kohavi, R. Longbotham, D. Sommerfield, R. M. Henne, Controlled experiments on the web: survey and practical guide, Data Mining & Knowledge Discovery 18(1) (2009) 140-181, https://doi.org/10.1007/s10618-008-0114-1.

[22] Okulograf Gazepoint, https://www.gazept.com/product/gp3hd/, [14.06.2022].

[23] Prawidłowe ustawienie okulografu przy laptopie, https://www.gazept.com/dl/gazepoint_laptop_mount.pdf, [14.06.2022].

[24] Platforma iMotions, https://imotions.com/, [18.01.2022].

[25] Opis metryk okulograficznych dostępnych z oprogramowania iMotions, https://imotions.com/blog/10-terms-metrics-eye-tracking/, [14.06.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 358–361
Received: 9 July 2022
Accepted: 25 September 2022

Comparative analysis of the performance of the Flax engine and Unity

Analiza porównawcza wydajności silników Flax engine i Unity

Wojciech Szelug*

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

Abstract
This article compares the two game engines Unity and Flax engine. The research consisted in comparing the efficiency of individual engines using two similar applications. The first one generates from several hundred to several thousand physical objects and examines the efficiency of engines in terms of physics simulations. The second application generates a forest area filled with the same tree model. The application aims to study the behavior of the engine for various graphics settings. In both applications, parameters such as the number of frames per second, consumption of the processor, graphics card and RAM were tested. Research shows that Unity is more efficient than Flax Engine.

Keywords: Unity; Flax Engine; game engine

Streszczenie
W niniejszym artykule porównane zostały dwa silniki gier Unity i Flax engine. Badania polegały na porównaniu wydajności poszczególnych silników za pomocą podobnych do siebie dwóch aplikacjach. Pierwsza z nich generuje od kilkuset do kilku tysięcy modeli 3D kul i bada wydajność silników pod względem symulacji fizyki. Druga aplikacja generuje teren lasu wypełniony takim samym modelem drzewa. Aplikacja ma na celu zbadanie zachowania silnika dla różnych ustawień graficznych. W obu aplikacjach badane zostały takie parametry jak: liczba klatek na sekundę, zużycie procesora, karty graficznej i RAM. Z badań wynika, że Unity jest wydajniejszy od Flax Engine.

Słowa kluczowe: Unity; Flax Engine; silniki gier

*Corresponding author
Email address: **Wojciech.Szelug@gmail.com** (W. Szelug)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Od 2017r. na platformie Steam publikowanych jest ponad pięćset gier miesięcznie w tym głównie gry niezależne (ang. indie) [1] a najczęściej używanymi silnikami są Unity, Unreal i GameMaker (Tabela 1). Silnik Flax Engine nie został wymieniony na tej stronie, jednak trzeba brać pod uwagę to, iż widnieje tam komunikat o tym, że te dane mogą być niedokładne

Tabela 1: Popularność silników gier na Steam dla 42163 gier

Silnik	Liczba użyć	Procent użyć
Unity	27233	65%
Unreal	6895	16%
GameMaker	2822	7%
Pozostałe	5 213	12%

Flax Engine został wydany w pełnej wersji w roku 2020 przez Wojciecha Figata [2]. Celem Flax jest stworzenie silnika do gier podobnego do Unity, Unreal engine i CryEngine oraz do silników open-source takich jak Godot, Banshee, Lumix. Obecnie najnowsza wersja silnika to 1.3, ale ma być on dalej rozwijany.

Unity jest jednym z najpopularniejszych silników gier na rynku [3] i posiada ogromną społeczność, która nieustannie tworzy poradniki i udziela się na forach i serwerach Discorda. Mimo że silniki Flax Engine i Unity są darmowe, to tylko do pewnego progu zarobku. W wersji osobistej można korzystać z narzędzia za darmo do momentu osiągnięcia przychodów w wysokości stu tysięcy dolarów rocznie dla Unity [4] i dwudziestu pięciu tysięcy dolarów przychodów na kwartał dla Flax [5].

W tym artykule porównane zostały wyżej wymienione silniki gier na dwa sposoby. Pierwszy polegał na stworzeniu podobnych aplikacji, w której wygenerowano określoną liczbę obiektów w kształcie kuli i porównano liczba klatek na sekundę, średnie zużycie wątku procesora, średnie zużycie karty graficznej i średnie zużycie pamięci RAM w MB. Drugi sposób polegał na stworzeniu ogromnego terenu lasu z różnym dystansem rysowania cieni, dla każdego silnika i sprawdzeniu tych samych parametrów co wcześniej. Oba rodzaje aplikacji zostały stworzone z tych samych obiektów 3D i za pomocą podobnych skryptów napisanych w języku C#.

Artykuł został napisany by sprawdzić, jak radzi sobie Flax Engine w porównaniu z Unity. Flax jest nowym silnikiem i nie został jeszcze przetestowany w ten sposób.

2. Przegląd literatury

Każdy silnik został zaprojektowany osobno a to znaczy, że istnieją różnice między nimi. Najważniejsze różnice

to wydajność poszczególnych silników. Można znaleźć kilka artykułów, w których porównywane są parametry wydajnościowe silników gier. Najczęściej porównywany jest Unity z innym również popularnym silnikiem. W artykule Huberta Żukowskiego [6] Unity jest porównywany z CryEngine. Autor stworzył podobne do siebie aplikacje, w których wygenerował kilka tysięcy kul. Z badań wynika, że Unity jest bardziej wydajny z mniejszą liczbą obiektów. Gdy obiektów jest dziesięć tysięcy to CryEngine radzi sobie lepiej, ale Unity nadal zużywa mniej zasobów komputera.

W artykule pod tytułem "Porównanie wydajności silników gier na wybranych platformach" [7], autor porównuje Unity oraz Unreal Engine przy pomocy stworzonych przez siebie podobnych gier wirtualnej rzeczywistości na urządzenia mobilne. Z jego badań wynika, że Unity jest nieznacznie wydajniejsze od Unreal Engine.

Andrzej Barczak i Hubert Woźniak [8] porównują Unity, CryEngine i Unreal Engine, lecz w ich badaniach wykorzystali aplikacje w których stworzyli teren lasu a następnie porównywali czas tworzenia się pojedynczej klatki dla poszczególnych silników. Ich wyniki wskazują, że Unity robi to najszybciej w prawie każdym przypadku.

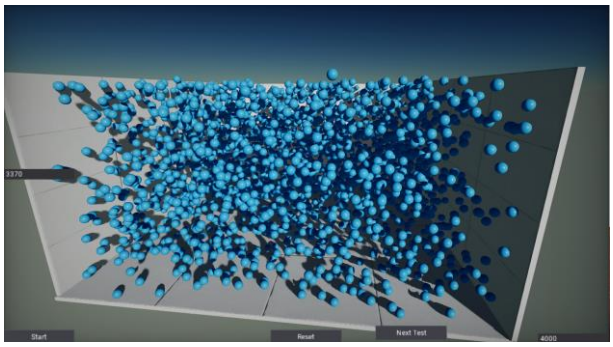
W artykule autorów Eleftheria Christopoulou, Stelios Xinogalos [9] stworzone zostały gry mobilne przy pomocy silników Unity i Unreal Engine 4. Ten artykuł miał na celu sprawdzić, który silnik nadaje się lepiej do tego typu zadań. Tutaj znowu wykazano, że Unity jest lepszy, ale ze względu na większą liczbę poradników i większą społeczność. Sprawdzono też wielkość aplikacji przed i po zbudowaniu z czego wynikło, że Unity zmniejszyło rozmiar aplikacji a Unreal zwiększyło.

W ostatnim artykule [10] porównano Unity i Unreal Engine w aspekcie tworzenia wirtualnych pokazów modeli pochodzących ze skanowania 3D. Ponownie stwierdzono, że to Unity jest wydajniejsze pod każdym względem.

Patrząc na wyżej wymienione badania można postawić tezę, że silnik Unity jest bardziej wydajny niż Flax Engine

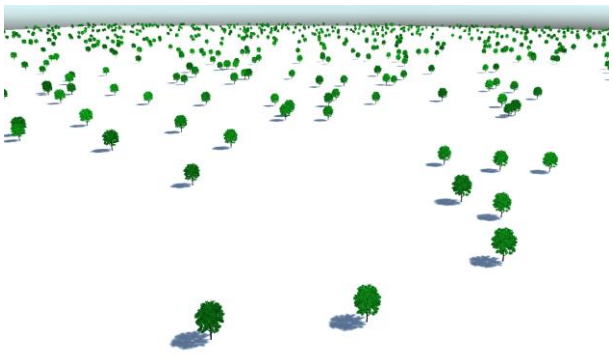
3. Metoda badań

Do badań stworzono podobne do siebie aplikacje wykorzystujące te same modele kuli i podobne skrypty. Aplikacje, które generują modele 3D kuli zaprogramowano tak by liczbę kul można było wpisać w polu tekstowym. W trakcie badań wygenerowano 1000, 5000, 10000 i 15000 kul na które działa grawitacja. Kule podczas testów poruszają się i odbijają się od siebie. Obiekty umieszczane są w zamkniętej platformie tak by miały one ze sobą jak najczęstszy kontakt. Dla każdego obiektu zachowane są zasady dynamiki i istnieje minimalna siła tarcia. Taka aplikacja pozwala obciążyć komputer na tyle by sprawdzić jak poszczególne silniki radzą sobie z tym zadaniem (Rysunek 1).



Rysunek 1: Aplikacja generująca modele 3D kul na silniku Flax Engine.

Drugi rodzaj aplikacji wykorzystuje model drzewa. Model ten składa się z 44848 wielokątów [11]. W aplikacji stworzono płaski teren, na którym umieszczono w losowych miejscach wspomniany model drzewa w liczbie 250, 500 i 1000. Odległość rysowania cieni ustawiono na zero (brak cienia), połowę długości terenu (odległość średnia) i maksymalną (cały cień). Przykładowa aplikacja została pokazana na Rysunku 2.



Rysunek 2: Aplikacja lasu ze średnim dystansem rysowania cieni i z 1000 drzew na silniku Unity.

Oba badania sprawdzały takie parametry jak: liczba klatek na sekundę, średnie zużycie wątku procesora w procentach, zużycie karty graficznej w procentach i średnie zużycie RAM w MB. By zarejestrować te parametry użyto autorskiego programu napisanego w języku Python. W programie tym użyto biblioteki GPUtil [12] do odczytania wykorzystanie karty graficznej oraz biblioteki psutil [13] do odczytania zużycia wątku procesora i zużycia RAM. Do wyznaczenia liczby klatek na sekundę użyto skryptu napisanego w testowanych aplikacjach. Wyniki synchronizowano czasem uniksowym [14].

Badania zostały przeprowadzone na komputerze o parametrach przedstawionych w Tabeli 2.

Tabela 2: Parametry komputera

Element	Wersja
System	Windows 10
Procesor	AMD Ryzen 5 5600X
Pamięć RAM	32GB DDR3
GPU	GeForce RTX 3070
Rozdzielczość	1920x1080

4. Wyniki

Pierwsze testy wykonano na aplikacji generującej modele 3D kul. Wyniki przedstawiono w tabelach dla różnych obciążeń

Tabela 3: Wyniki pomiarów dla obciążenia 1000 kul

	FPS	CPU	RAM [MB]	GPU
Flax	144,11	3,75%	165,94	43,85%
Unity	143,99	6,61%	129,14	37,45%

Pierwszy test (Tabela 3) okazał się zbyt małym obciążeniem by pokazać znaczącą różnicę w wynikach dla FPS. Pozostałe wyniki pokazują, że Unity zużywa więcej mocy obliczeniowej procesora, ale za to wykorzystuje mniej pamięci RAM i karty graficzne

Na podstawie testu drugiego (Tabela 4), podobnie jak na podstawie testu pierwszego, nie można określić znaczącej różnicy w liczbie FPS dla poszczególnych silników, co nadal uniemożliwia wybranie lepszego silnika pod względem wydajności. Pod względem zużycia podzespołów wynik zostaje taki sam jak w teście pierwszym, czyli silnik Unity zużywa więcej procesora natomiast Flax więcej RAM i karty graficznej.

Tabela 4: Wyniki pomiarów dla obciążenia 5000 kul

	FPS	CPU	RAM [MB]	GPU
Flax	145,60	13,08%	215,34	75,56%
Unity	144,00	24,68%	170,67	37,65%

Tabela 5: Wyniki pomiarów dla obciążenia 10000 kul

	FPS	CPU	RAM [MB]	GPU
Flax	58,96	16,08%	278,76	55%
Unity	107,42	37,44%	229,56	39,43%

Tabela 6: Wyniki pomiarów dla obciążenia 15000 kul

	FPS	CPU	RAM [MB]	GPU
Flax	30,94	17,08%	365,43	40,29%
Unity	55,46	42,93%	294,02	35,72%

Wyniki testu trzeciego i czwartego (Tabela 5 i Tabela 6) ukazują, który silnik radzi sobie lepiej z symulacją fizyki z dużą liczbą obiektów 3D. Unity osiągnął większą liczbą klatek na sekundę w obu testach. Unity zużywa więcej procesora, ale za to mniej pamięci RAM i karty graficznej.

Tabela 7: Brak cienia

250 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	144,50	3,53%	163,35	66,30%
Unity	143,89	2,35%	134,78	35,79%
500 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	144,19	4,55%	165,93	88,86%
Unity	143,83	4,08%	139,02	51,33%
1000 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	111,02	1,69%	165,32	97,35%
Unity	132,77	6,11%	144,68	91,40%

Tabela 8: Średni cień

250 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	136,23	2,55%	180,44	95,02%
Unity	143,91	3,77%	142,37	66,07%
500 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	90,34	1,27%	182,30	98,00%
Unity	121,16	6,18%	151,23	97,47%
1000 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	54,71	0,87%	177,71	98,76%
Unity	62,70	5,58%	174,27	98,25%

Wyniki testów na drugiej aplikacji pokazano w Tabelach 7, 8, 9. Unity i w tych przypadkach osiągnęło większą liczbę klatek na sekundę za wyjątkiem aplikacji, w której drzewa nie rzucały cienia przy liczbie drzew 250 i 500 (Tabela 7). Tam liczba klatek na sekundę osiągnęła w obu silnikach maksymalną wartość. Wyniki zużycia podzespołów pokazują, że Unity zużywa więcej procesora, gdy drzewa rzucają cień, ale zużywa mniej RAM i karty graficznej w pozostałych przypadkach. Silnik Flax zużywa stałą ilość RAM w zależnie od dystansu

cieniowania natomiast Unity ma tendencję do zwiększania zużycia RAM o niewielką ilość, gdy zwiększa się liczba drzew.

Tabela 9: Maksymalny cień

250 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	119,49	1,88%	163,72	97,61%
Unity	139,75	5,76%	163,63	94,56%
500 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	70,66	1,04%	166,22	98,89%
Unity	87,48	5,57%	169,26	97,17%
1000 drzew				
	FPS	CPU	RAM [MB]	GPU
Flax	39,45	0,73%	162,28	99,00%
Unity	42,27	5,26%	178,67	97,45%

5. Wnioski

Testy na aplikacjach stworzonych na silnikach Unity i Flax pozwalają na sformułowanie następujących wniosków:

- Unity osiąga lepsze wyniki w liczbie wyświetlanych klatek na sekundę dla obu rodzajów testów,
- Flax zużywa więcej pamięci RAM i procentowo karty graficznej. Sama pamięć jest wykorzystywana w stałej ilości,
- Unity wykorzystuje więcej procesora a pamięć RAM wykorzystuje zależnie od potrzeb.

Wyniki potwierdzają tezę, którą postawiono w rozdziale drugim. Silnik Unity jest wydajniejszy od silnika Flax we wszystkich przeprowadzonych testach. Silnik ten dobrze radzi sobie z obliczaniem fizyki jak i rysowaniem cieni dla skomplikowanych obiektów jak drzewo wykorzystane w aplikacji.

Literatura

[1] Dane o liczbie wydawanych gier w poszczególnych latach, https://steamdb.info/stats/releases/?tagid=0, [14.01.2022].

[2] Informacje o Flax Engine, https://flaxengine.com/blog/category/releases/, [12.06.2022].

[3] Dane dotyczące popularności silników gier na Stream, https://steamdb.info/tech/, [14.01.2022].

[4] Licencja personalna dla Unity, https://unity3d.com/unity/activation/personal, [12.06.2022].

[5] Licencja dla Flax, https://flaxengine.com/licensing/, [12.06.2022].

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

[7] P. Skop, Porównanie wydajności silników gier na wybranych platformach, Journal Computer Sciences Institute 7 (2017) 116-119.

[8] A. Barczak, H. Woźniak, Comparative Study on Game Engines, STUDIA INFORMATICA Systems and information technology 1-2 (2019) 5-24, https://doi.org/10.34739/si.2019.23.01.

[9] E. Christopoulou, S. Xinogalos, Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices, International Journal of Serious Games 4 (2017) 21-36, https://doi.org/10.17083/ijsg.v4i4.194.

[10] A. M. Ciekanowska, A. K. Kiszczak – Gliński, K. Dziedzic. Analiza porównawcza wydajności silników Unity i Unreal Engine w aspekcie tworzenia wirtualnych pokazów modeli pochodzących ze skanowania 3D, Journal Computer Sciences Institute 20 (2021) 247-253, https://doi.org/10.35784/jcsi.2698.

[11] Model drzewa, https://www.turbosquid.com/pl/3d-models/trees-realistic-3d-1218366, [12.06.2022].

[12] Kod źródłowy biblioteki GPUtil, https://github.com/anderskm/gputil, [12.06.2022].

[13] Kod źródłowy biblioteki psutil, https://github.com/giampaolo/psutil, [12.06.2022].

[14] Opis czasu uniksowego, https://en.wikipedia.org/wiki/Unix_time, [12.06.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 362–365
Received: 1 September 2022
Accepted: 29 September 2022

Comparison of the cross-platform mobile applications performance in 2D graphics processing

Porównanie wydajności wieloplatformowych aplikacji mobilnych w przetwarzaniu grafiki 2D

Adam Drzewiecki*

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

Abstract
The article presents a comparison of cross-platform frameworks in 2D graphics processing. The tests consisted in applying filters to 2D images, and then the image processing time and standard deviation were measured. The test will be conducted on the Huawei P8 and Samsung A50 phones. The comparison concerns two very popular cross-platform programming frameworks: Flutter and Xamarin.

Keywords: Flutter; Xamarin; cross-platform framework performance

Streszczenie
Artukuł przedstawia porównanie frameworków wieloplatformowych w przetwarzaniu grafiki 2D. Badania polegają na nakładaniu wybranych filtrów na obrazy 2D, a następnie jest mierzony czas nakładania dla danego filtra oraz obliczane odchylenie standardowe. Badanie zostanie przeprowadzone na telefonie Huawei P8 i Samsung A50. Porównanie dotyczy dwóch bardzo popularnych wieloplatformowych szkieletów programistycznych (framework), czyli odpowiednio Flutter oraz Xamarin.

Słowa kluczowe: Flutter; Xamarin; wydajność frameworków wieloplatformowych

*Corresponding author
Email address: adam.drzewiecki@pollub.edu.pl (A. Drzewiecki)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

W dzisiejszych czasach frameworki wieloplatformowe stają się coraz bardziej popularne. Telefony komórkowe są znacznie szybsze niż kilka lat temu, dzięki czemu wydajność aplikacji napisanych w takich frameworkach jest porównywalna z wydajnością aplikacji natywnych. Wieloplatformowość pozwala na napisanie aplikacji tylko raz i taka aplikacja będzie mogła działać na wszystkich systemach mobilnych. Dzięki frameworkom wieloplatformowym można zaoszczędzić czas oraz pieniądze [1], ponieważ aplikacja może zostać napisana tylko raz i będzie działała na wszystkich systemach mobilnych. W artykule przedstawiono analizę wydajności frameworków wieloplatformowych z punktu widzenia przetwarzania grafiki 2D. Porównane zostaną frameworki Flutter oraz Xamarin. Do przeprowadzenia badań zostaną utworzone 2 takie same aplikacje w obu frameworkach. Badanie będzie polegało na nakładaniu filtrów na obrazy 2D, a następnie zostanie zmierzony czas przetwarzania grafiki oraz zostanie obliczone odchylenie standardowe.

2. Przegląd literatury

W artykule [1] przebadano wydajność frameworka Xamarin oraz aplikacji natywnej m.in. dla szybkości generowania interfejsu użytkownika. Stworzono 2 aplikacje, jedna z nich była aplikacją natywną napisaną w języku Java, a druga we frameworku Xamarin. Aplikacje zostały przetestowane na tym samym urządzeniu oraz zmierzono czas wykonywania poszczególnych operacji. Framework Xamarin okazał się gorszy z punktu widzenia wydajności niż aplikacja natywna.

W artykule [2] obiektem badań były narzędzia programistyczne służące do tworzenia aplikacji na systemie Android, zbadano Android SDK i język Java, Android NDK, Xamarin oraz Apache Cordova. Zbadano wydajność poszczególnych frameworków dla odczytu/zapisu dużych plików oraz dla sortowania tablicy. Wszystkie testy zostały przeprowadzone na urządzeniu fizycznym oraz emulatorze android. Na podstawie przeprowadzonych testów nie można jednoznacznie stwierdzić, które narzędzie pozwala tworzyć najwydajniejsze aplikacje.

W artykule [3] celem badań było porównanie wydajności frameworka Xamarin oraz języka Java na systemach Android oraz Windows 10 Mobile. Przetestowano szybkość uruchamiania aplikacji oraz szybkość odczytu i zapisu plików. Testy zostały przeprowadzone na emulatorze z systemem Android 7 oraz Windows 10 Mobile. Zebranie wyników badań na platformie Android wskazuje na wysokie spowolnienie działania spowodowane wykorzystaniem wieloplatformowej technologi, natomiast na systemie Windows 10 Mobile różnica wydajności pomiędzy aplikacją wieloplatformową oraz natywną była dużo mniejsza.

W artykule [4] obiektem badań było zbadanie wydajności przechwytywania obrazów oraz filtrów graficznych dla frameworka Xamarin. W celu przeprowadzenia badań, w środowisku Xamarin stworzona została aplikacja na systemy Android i Windows 10 Mobile (UWP). Aplikacje zostały zainstalowane na dwóch

urządzeniach fizycznych i zostały zmierzona wydajność aplikacji. Okazało się, że na czas wczytywania obrazu wpływ ma rozmiar pliku oraz zastosowanie filtru.

W artykule [5] porównano tworzenie aplikacji wykorzystując technologie natywne dla Androida oraz we frameworku Flutter. Stworzono 2 aplikacje, jedna z nich była aplikacją natywną a druga we frameworku Flutter. Do zbadania obciążenia procesora podczas operacji sortowania danych wykorzystano moduł Android Profiler oraz narzędzie DevTools. Aplikacja natywna okazała się bardziej wydajna niż aplikacja napisana we frameworku Flutter, technologia natywna miała również większe wsparcie społeczności oraz więcej dostępnych bibliotek.

3. Metoda badawcza

Badanie będzie polegało na zmierzeniu wydajności aplikacji mobilnej napisanej w różnych technologiach wieloplatformowych. Zbadany zostanie czas przetwarzania grafiki 2D oraz zmierzone odchylenie standardowe z pomiarów czasu.

Do przeprowadzenia badania zostaną utworzone dwie takie same aplikacje we frameworkach Flutter oraz Xamarin. Na obrazy będą nakładane różne filtry (np. Filtr Gaussa, Progowanie), a następnie będzie mierzony czas przetworzenia obrazu oraz odchylenie standardowe. Każdy pomiar będzie dodawany do listy, a następnie po przeprowadzeniu 100 pomiarów będzie można odczytać średni wynik dla danego filtra. Badanie zostanie przeprowadzone na 2 urządzeniach. Do przeprowadzenia badań został wybrany telefon ze starszą wersją androida o gorszych parametrach technicznych oraz telefon z najnowszą wersją Androida o lepszych parametrach. Taki zestaw urządzeń pokaże, jak aplikacje wieloplatformowe zachowują się na różnych urządzeniach oraz czy wybór technologii wieloplatformowej jest przyszłościowy.

3.1. Urządzenia badawcze

Pierwszym urządzeniem będzie telefon fizyczny Huawei P8 z systemem Android 6.0. Premiera tego telefonu miała miejsce w 2015 roku, jego parametry odbiegają od dzisiejszych standardów. Badanie pokaże, czy aplikacje wieloplatformowe są wydajne na starszych urządzeniach.

Drugim urządzeniem będzie telefon fizyczny Samsung A50 z systemem Android 11. To urządzenie jest najlepszym telefonem pod względem wydajności spośród wszystkich urządzeń badawczych. Premiera tego telefonu odbyła się w 2019 roku, w porównaniu do Huawei P8 jego parametry techniczne są dużo lepsze oraz ten telefon powinien przetwarzać grafikę wydajniej.

3.2. Scenariusze badawcze

Aby przeprowadzić badania opracowano 10 scenariuszy badawczych, które będą polegały na pomiarze czasu przetwarzania obrazu dla filtrów:

1. Skala szarości
2. Rozmycie Gaussa
3. Sepia
4. Wyostrzanie
5. Modulacja
6. Luminacja
7. Saturacja
8. Efekt Hue
9. Rozjaśnianie
10. Przyciemnianie

4. Aplikacje opracowane na potrzeby badań

Aby przeprowadzić badania, zostały utworzone 2 takie same aplikacje we frameworkach Flutter oraz Xamarin. Każda z aplikacji zawierała 2 ekrany. Na ekranie głównym możemy wybrać filtr oraz rozdzielczość obrazu, a na drugim ekranie nałożyć filtr na obraz oraz zmierzyć czas przetwarzania wybranego obrazu. Aplikacja we frameworku Xamarin została napisana w języku C# w środowisku Visual Studio, a aplikacja we frameworku Flutter w języku Dart w środowisku Visual Studio Code.

Dla każdego filtra badanie zostało przeprowadzone 30 razy, a następnie z pomiarów czasu została obliczona średnia oraz odchylenie standardowe. Aby obliczyć średnią oraz odchylenie standardowe, po przeprowadzeniu pomiaru wynik był dodawany do listy.

Rysunek 1: Aplikacja mobilna do przeprowadzenia badań.

5. Wyniki

Wyniki badań zostały umieszczone w poniższych tabelach. W każdej tabeli znajduje się średni czas przetwa-

rzania obrazu dla danego filtra oraz odchylenie standardowe ze zmierzonych wartości. Wyniki zostały podzielone na tabele dla każdego urządzenia badawczego.

5.1. Analiza wyników frameworka Xamarin

Tabela 1 oraz tabela 2 przedstawiają wyniki badań dla frameworka Xamarin dla wszystkich urządzeń badawczych. W pierwszej tabeli zostały umieszczone wyniki dla urządzenia Huawei P8.

Tabela 1: Średnie czasy nakładania filtrów dla frameworka Xamarin - Huawei P8 Lite

Nazwa filtra	Czas (ms)	Odchylenie (ms)
Skala szarości	28,2	6,9
Rozmycie Gaussa	33,7	7,2
Sepia	29,7	5,2
Wyostrzanie	30,1	6,2
Modulacja	28,8	7,4
Luminacja	26,6	5,0
Saturacja	27,4	6,4
Efekt hue	28,0	5,6
Rozjaśnianie	28,4	6,5
Przyciemnianie	28,2	5,9

Druga tabela przedstawia wyniki badań dla urządzenia Samsung A50. Możemy zobaczyć znaczną różnicę w czasie przetwarzania obrazu, telefon Samsung A50 przetwarzał grafikę prawie 10 razy szybciej niż telefon Huawei. To samo można stwierdzić o odchyleniu standardowym, wartości odchylenia standardowego są nawet 15-20 razy mniejsze w przypadku niektórych filtrów w porównaniu z urządzeniem Huawei.

Tabela 2: Średnie czasy nakładania filtrów dla frameworka Xamarin - Samsung A50

Nazwa filtra	Czas (ms)	Odchylenie (ms)
Skala szarości	2,1	0,4
Rozmycie Gaussa	3,2	0,3
Sepia	1,7	0,1
Wyostrzanie	1,8	0,3
Modulacja	1,2	0,2
Luminacja	1,4	0,3
Saturacja	2,1	0,1
Efekt hue	2,2	0,4
Rozjaśnianie	2,4	0,4
Przyciemnianie	1,7	0,2

5.2. Analiza wyników frameworka Flutter

W tabeli 3 oraz tabeli 4 zostały umieszczone wyniki dla frameworka Flutter. W porównaniu do frameworka Xamarin, framework Flutter przetwarzał grafikę 3-4 razy szybciej. Wartości odchylenia standardowego są podobne dla obu frameworków i wynoszą około 30% wartości czasu. Tabele zawierają wyniki dla obu urządzeń badawczych.

Tabela 3: Średnie czasy nakładania filtrów dla frameworka Xamarin - Huawei P8 Lite

Nazwa filtra	Czas (ms)	Odchylenie (ms)
Skala szarości	6,3	1,5
Rozmycie Gaussa	6,1	2,3
Sepia	4,7	1,2
Wyostrzanie	5,5	1,3
Modulacja	7,2	1,5
Luminacja	6,1	2,0
Saturacja	7,1	2,1
Efekt hue	4,9	1,7
Rozjaśnianie	5,4	1,5
Przyciemnianie	5,6	1,4

Druga tabela przedstawia wyniki badań dla urządzenia Samsung A50. Można zauważyć znaczną różnicę w czasie przetwarzania obrazu na korzyść urządzenia Samsung A50, różnice wartości odchylenia standardowego są także zauważalne.

Tabela 4: Średnie czasy nakładania filtrów dla frameworka Xamarin - - Samsung A50

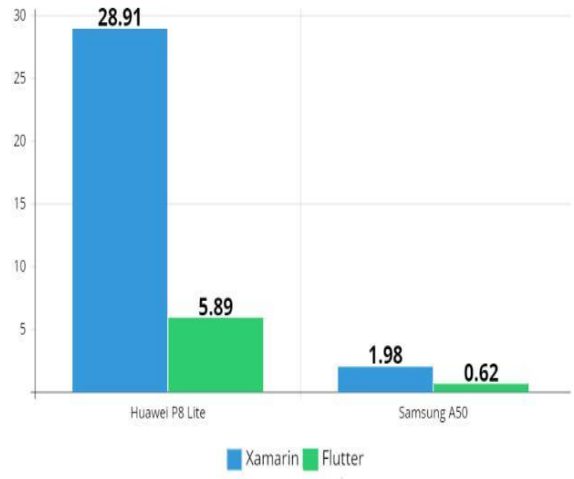
Nazwa filtra	Czas (ms)	Odchylenie (ms)
Skala szarości	0,4	0,1
Rozmycie Gaussa	1,1	0,1
Sepia	0,7	0,1
Wyostrzanie	0,3	0,2
Modulacja	0,5	0,1
Luminacja	0,8	0,2
Saturacja	0,6	0,2
Efekt hue	0,3	0,1
Rozjaśnianie	0,8	0,2
Przyciemnianie	0,7	0,2

6. Dyskusja

6.1. Interpretacja wyników

Z danych w tabelach z wynikami można wyciągnąć kilka wniosków:

- Wszystkie filtry nakładają się w podobnym czasie dla tego samego obrazu oraz urządzenia. Można stwierdzić, że samo przetwarzanie pikseli obrazu zajmuje niewiele czasu, natomiast najwięcej czasu zajmuje wyświetlenie obrazu ,
- Można zauważyć różnicę pomiędzy urządzeniami badawczymi. Na nowszym urządzeniu badawczym (Samsung A50) filtry nakładały się szybciej niż na starszym telefonie,
- Framework Flutter okazał się bardziej wydajny niż Xamarin. We Flutterze wszystkie filtry nakładały się 3-4 razy szybciej niż w Xamarinie,
- Odchylenie standardowe dla niektórych filtrów wyniosło nawet 50% wartości czasu przetwarzania dla danego filtru, więc wartości czasów cechował silny rozrzut od średniej.



Rysunek 2: Porównanie średnich ze wszystkich filtrów dla danego frameworka i urządzenia badawczego.

Powyższy wykres przedstawia średnią ze wszystkich filtrów dla danego frameworka i dla każdego urządzenia badawczego. Widać wyraźną różnicę pomiędzy frameworkiem Flutter oraz Xamarin. Największa różnica jest widoczna na starszym urządzeniu Huawei P8, gdzie framework Flutter przetwarzał grafikę 5 razy szybciej. Mniejsza różnica była na urządzeniu Samsung A50, gdzie Flutter przetwarzał grafikę 3 razy szybciej. Należy również zwrócić uwagę na różnicę wartości tych pomiarów. Na urządzeniu Samsung ta różnica wynosi tylko 1,36ms, a na Huawei P8 aż 23.02ms. Z punktu widzenia użytkownika korzystającego z aplikacji, 1,36ms może być różnicą praktycznie nie zauważalną, natomiast różnicę 23,02ms była odczuwalna w trakcie wykonywania badań, co oznacza, że framework Flutter będzie dużo lepszym wyborem jeśli aplikacja będzie wspierała stare urządzenia.

6.2. Weryfikacja hipotez

W artykule zostały postawione 2 hipotezy:

- Framework Flutter cechuje się wyższą wydajnością w przetwarzaniu grafiki 2D wśród obu frameworków wieloplatformowych,
- Urządzenia z nowszą wersją systemu Android przetwarzały grafikę wydajniej niż wersje ze starszą wersją system.

Po przeanalizowaniu wyników badań można stwierdzić, że obie hipotezy zostały spełnione. Framework Flutter przetwarzał grafikę wydajniej niż framework Xamarin. Widać również różnicę pomiędzy urządzeniami badawczymi, urządzenie z nowszą wersją Androida przetwarzało grafikę wydajniej niż urządzenie ze starszą wersją systemu. Firma Google co roku wprowadza szereg usprawnień w nowszych wersjach Androida, co ma duży wpływ na wydajność aplikacji. Należy pamiętać, że nowsze wersje systemu Android są zainstalowane na nowych urządzeniach, które bardzo często mają lepsze parametry techniczne niż urządzenia ze starszymi wersjami Androida.

6.3. Porównanie z przeglądem

Wyniki badań są bardzo podobne do tych, które można znaleźć w artykułach z przeglądu literatury.

W artykule [1] porównano wydajność frameworka Xamarin oraz aplikację natywną dla systemu Android. Po przeprowadzeniu badań okazało się, że framework Xamarin okazał się gorszy z punktu widzenia wydajności niż rozwiązanie natywne.

W artykule [5] porównano wydajność frameworka Flutter z aplikacją natywną dla systemu Android. Wyniki badań pokazały, że aplikacja napisana we Flutterze oraz aplikacja natywna cechują się podobną wydajnością. Na podstawie 2 powyższych artykułów można stwierdzić, że framework Flutter cechuje się wyższą wydajnością niż framework Xamarin, ponieważ Flutter wyróżnił się wyższą wydajnością niż Xamarin w porównaniu do aplikacji natywnej.

7. Wnioski

Na podstawie przeprowadzonych badań można wysnuć następujące wnioski:

- Framework Flutter cechuje się większą wydajnością w przetwarzaniu grafiki 2D niż framework Xamarin, wynika to najprawdopodobniej z architektury tego frameworka. Flutter do renderowania interfejsu używa silnika Skia, natomiast Xamarin korzysta ze środowiska Mono, które konwertuje elementy layoutu do kontrolek natywnych, co zajmuje więcej czasu niż renderowanie interfejsu we Flutterze,
- Urządzenia z nowszą wersją systemu Android przetwarzały grafikę wydajniej niż wersje ze starszą wersją systemu.

Literatura

[1] I. Bodia, M. Plechawska-Wójcik, Performance comparison of the Xamarin platform and native applications for Android operating system, Journal of Computer Sciences Institute, 5 (2017) 208-212, https://doi.org/10.35784/jcsi.624

[2] P. Kotarski, K. Śledź, J. Smołka, Analysis of the impact of development tools used on the performance of the mobile application, Journal of Computer Sciences Institute 6 (2018) 68-72, https://doi.org/10.35784/jcsi.642

[3] D. Wieczorek, J. Smołka, Comparison of performance multi-platform application core on Android and Windows 10 Mobile, Journal of Computer Sciences Institute 6 (2018) 87- 91. https://doi.org/10.35784/jcsi.647

[4] M. Dras, G. Fila, M. Plechawska-Wójcik, Analysis of Xamarin capabilities for building mobile multi-platform applications, Journal of Computer Sciences Institute 7 (2018) 183-190, https://doi.org/10.35784/jcsi.675

[5] D. Gałan, K. Fisz, P. Kopniak, A multi-criteria comparison of mobile applications built with the use of Android and Flutter Software Development Kits, Journal of Computer Sciences Institute 19 (2021) 107-113, https://doi.org/10.35784/jcsi.2614

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 366–370
Received: 13 July 2022
Accepted: 20 September 2022

A comparative analysis of performance of Flutter and Xamarin development frameworks

Analiza porównawcza wydajności szkieletów programistycznych Flutter oraz Xamarin

Mateusz Uciński*, Mariusz Dzieńkowski

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

Abstract
This article presents a comparative performance analysis of two cross-platform development frameworks Flutter and Xamarin. Using these technologies identical test applications running on Windows and Android were created. Each of these applications included functionalities to run test scenarios. They concerned calculating the thirtieth word of the Fibonacci sequence, sorting with the MergeSort algorithm lists consisting of five thousand and ten thousand elements, performing basic database operations on the database, such as record, reading, searching, modifying and deleting data. The scenarios were repeated ten thousand times, and the average execution times of the operations were analyzed. The results did not conclusively show which framework is more efficient. However, in general, it can be concluded that for applications running on Android and Windows that perform a lot of calculations or save large amounts of data or search and at the same time modify data, the Flutter framework will be a better solution.

Keywords: Flutter; Xamarin; cross-platform development frameworks

Streszczenie
Artykuł ten przedstawia wydajnościową analizę porównawczą dwóch wieloplatformowych szkieletów programistycznych Flutter oraz Xamarin. Przy pomocy tych technologii utworzono identyczne aplikacje testowe działające pod kontrolą systemu Windows oraz systemu Android. Każda z tych aplikacji zawierała funkcjonalności umożliwiające przeprowadzenie scenariuszy testowych. Dotyczyły one obliczenia trzydziestego wyrazu ciągu Fibonacciego, posortowania algorytmem przez scalanie list składających się z pięciu tysięcy oraz dziesięciu tysięcy elementów, wykonania na bazie danych podstawowych operacji takich jak: zapis, odczyt, wyszukanie, modyfikacja i usunięcie danych. Scenariusze zostały powtórzone dziesięć tysięcy razy, a analizie zostały poddane średnie czasy wykonania danych operacji. Wyniki nie wykazały jednoznacznie, który szkielet jest wydajniejszy. Jednak generalnie można stwierdzić, że dla aplikacji pracujących na systemach Android i Windows, które wykonują dużo obliczeń lub zapisują duże ilości danych czy wyszukują i jednocześnie modyfikują dane, lepszym rozwiązaniem będzie szkielet programistyczny Flutter.

Słowa kluczowe: Flutter; Xamarin; wieloplatformowe szkielety programistyczne

*Corresponding author
Email address: s97270@pollub.edu.pl (M. Uciński)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Popularność urządzeń mobilnych takich jak telefony, smartfony oraz tablety w ostatnich latach szybko rośnie. W związku z tą tendencją ich znaczenie, a także liczba aplikacji dostępnych na takie urządzenia również wzrasta. Pod koniec roku 2021 w sklepie Google Play Store [1] można było znaleźć ponad cztery miliony aplikacji mobilnych, natomiast w sklepie Apple App Store liczba aplikacji wyniosła ponad dwa miliony [2]. Wśród użytkowników komputerów stacjonarnych dominuje system Windows. Jest to ogromna grupa potencjalnych odbiorców oprogramowania. Jednak stosowanie natywnych rozwiązań do tworzenia aplikacji dla każdej platformy jest drogie w utworzeniu i utrzymaniu. W związku z zapotrzebowaniem na rozwiązanie pozwalające na tworzenie aplikacji wieloplatformowych powstało wiele szkieletów programistycznych, które pozwalają rozwiązać ten problem. Szkielety programistyczne są rodzajem oprogramowania pomocniczego, które definiują strukturę aplikacji, mechanizmy działania oraz dostarczają zestaw komponentów i bibliotek, które ułatwiają i przyspieszają tworzenie aplikacji. Niemal każdy język programowania posiada swoje szkielety programistyczne. Każdy z nich oferuje inne funkcje, rozwiązania i narzędzia. Dlatego kluczową kwestią, przed którą stoi deweloper oprogramowania jest wybór odpowiedniej technologii, która sprosta przedstawionym problemom.

Szkielety programistyczne, pozwalające na tworzenie aplikacji, które są budowane i uruchamiane na wielu platformach sprzętowych oraz wielu systemach operacyjnych, są obecnie powszechnie stosowane. Dotyczy to zwłaszcza tworzenia aplikacji na systemy Android oraz iOS, gdyż systemy te skierowane są głównie na urządzenia mobilne. Jednak wieloplatformowość nie dotyczy jedynie tych dwóch systemów. Szkielety programistyczne obsługują również środowiska komputerów stacjonarnych na systemy Windows oraz Linux, jak również można dzięki nim utworzyć aplikacje przeglądarkowe.

W tej pracy skupiono się na porównaniu dwóch wieloplatformowych szkieletów programistycznych Flutter [3] oraz Xamarin [4]. Ich analiza została przeprowadzona na najważniejszych platformach urządzeń mobilnych i stacjonarnych, czyli Android oraz Windows. Flutter jest szkieletem programistycznym z otwartym kodem źródłowym, którego rozwiązania są dostępne dla Androida, iOSa, Linuxa, MacOSa, wiodących przeglądarek internetowych oraz od wersji 2.10 stabilnie wspiera rozwiązania dla systemu Windows. Jest on niemal sześć lat młodszy od drugiego narzędzia, z którym będzie porównany. Xamarin to także szkielet z otwartym kodem źródłowym, umożliwiającym tworzenie aplikacji dla systemów iOS, Android i Windows za pomocą platformy .NET w języku programowania C#. Rozwiązanie to zostało stworzone w 2011 roku przez firmę o tej samej nazwie. W 2016 roku nabył ją Microsoft [5], który cały czas rozwija tę technologię.

Celem niniejszej pracy jest analiza porównawcza dwóch szkieletów programistycznych Flutter i Xamarin działających na wielu platformach. W zrealizowanych badaniach skupiono się na sprawdzeniu efektywności wykorzystania zasobów sprzętowych przez aplikacje utworzone na bazie porównywanych szkieletów na dwóch platformach testowych: mobilnej oraz komputerze osobistym. W ramach pracy została sformułowana następująca hipoteza badawcza: *Aplikacje oparte na szkielecie programistycznym Flutter realizujące zadania obliczeniowe lub zadania na bazie danych takie jak zapis, modyfikacja, wyszukiwanie danych, są wydajniejsze od aplikacji zbudowanych na bazie platformy Xamarin, zarówno na systemach Android jak i Windows*.

2. Przegląd literatury

Rosnąca popularność szkieletów programistycznych, które umożliwiają tworzenie aplikacji wieloplatformowych, spowodowała wzrost liczby dywagacji na temat wyboru najbardziej odpowiedniego rozwiązania. Powstało zatem wiele artykułów oraz publikacji naukowych zawierających porównania wieloplatformowych szkieletów programistycznych z rozwiązaniami natywnymi oraz porównujących różne szkielety programistyczne ze sobą.

W pracy S.Hedlund'a i Y.R. Wright'a [6] autorzy przeprowadzili analizę narzędzi Xamarin i Flutter. Twórcy w swojej pracy postawili trzy pytania badawcze. Pierwsze z nich dotyczyło jakości dokumentacji, w jakie zaopatrzone są szkielety, ponieważ jest to główne źródło wiedzy o technologii dla programistów. Drugie pytanie dotyczyło złożoności oraz rozmiaru kodu źródłowego tworzonych za pomocą tych narzędzi aplikacji. Pytanie to wynikało z istniejącego związku między dużą ilością i złożonością kodu, a czasem tworzenia oraz łatwością utrzymania aplikacji. Kolejne pytanie dotyczyło wydajności obu szkieletów programistycznych pod względem wykorzystania pamięci RAM, obciążenia procesora oraz czasu uruchomienia aplikacji. W celu znalezienia odpowiedzi na ostatnie pytanie autorzy pracy utworzyli dwie aplikacje, które w ramach testu wydajnościowego wykorzystania procesora wykonywały algorytm wyszukiwania liczb pierwszych dla czterech różnych wartości. Drugim testem było sprawdzenie zarządzania pamięcią aplikacji tworzących nieskończenie długie listy obiektów. Po wykonaniu badań autorzy stwierdzili, że Xamarin posiada obszerniejszą i dokładniejszą dokumentację. Ze względu na to, że oba szkielety programistyczne bazują na składni C złożoność była podobna, dlatego decydującym aspektem był rozmiar gotowej aplikacji utworzonej na system Android. Wyniki porównania złożoności kodu i rozmiaru aplikacji okazały się lepsze dla Fluttera. Także w odpowiedzi na trzecie pytanie badawcze postawione w pracy, Flutter uzyskał znaczną przewagę, ponieważ do wykonania obliczeń potrzebował trzykrotnie mniej czasu niż aplikacja napisana w Xamarin.

W opublikowanej pracy pod tytułem „Comparison and evaluation of crossplatform framework and development of a digital health platform using selected framework" [7] autor - Md Shoaibe Anwar porównuje cztery wieloplatformowe szkielety programistyczne: React-native, Flutter, Ionic oraz Xamarin. Analiza ta została przeprowadzana w sześciu aspektach: wspierane platformy i systemy operacyjne, architektura platformy, dostępność gotowych interfejsów programistycznych, język programowania tworzenia aplikacji, publikowanie aplikacji (czas, rozmiar aplikacji), wsparcie języka (dokumentacja, społeczność skupiona wokół niego, popularność), warunki tworzenia aplikacji (czas tworzenia projektu, struktura kodu oraz tworzenie testów). Po wykonaniu zestawienia uwzględniającego wyżej wymienione aspekty autor ogłosił szkielet Flutter, jako najlepszą technologię do utworzenia aplikacji i następnie ją przebadał urządzeniem śledzącym wzrok.

W roku 2021 czterech autorów opublikowało pracę pod tytułem „A Comparison of Native and Cross-Platform Frameworks for Mobile Applications" [8]. W pracy tej przeprowadzona została analiza trzech wieloplatformowych szkieletów programistycznych oraz rozwiązań natywnych dla urządzeń mobilnych z systemem operacyjnym Android oraz iOS, dla dwóch aplikacji testowych. Pierwsza, bardzo prosta aplikacja, wyświetlała tylko napis „Hello World" na środku ekranu. W tym przypadku zbadano jej rozmiar, czas uruchamiania oraz zapotrzebowanie na pamięć RAM. Druga aplikacja była bardziej złożona i służyła do zbadania czasu podczas graficznego przedstawienia treści. Składała się ona z jednego głównego widoku, przy pomocy którego można przejść do pozostałych stron testowych. Jedna strona nie realizowała żadnego zadania i działała w tle. Trzy pozostałe umieszczone w osobnym wątku, wykonywały zadania obciążające procesor tj. czytanie i zapisywanie pliku lub wysyłanie zapytań HTTP do silnika wyszukiwarki Google. Testy rozmiarów aplikacji wykazały, że Flutter oraz Xamarin pracujące na systemie iOS znacznie zwiększają swój rozmiar w porównaniu do systemu Android. Podobną zależność zauważono w czasach reakcji uruchomienia aplikacji. Wyniki zarządzania pamięcią pokazały, że Flutter jest szkieletem wymagającym największej ilości pamięci RAM. Natomiast wyniki wykorzystania procesora były

zaskakujące, ponieważ okazało się, że Flutter wykonywał szybciej zadania niż aplikacje napisane w natywnych rozwiązaniach. React Native oraz Xamarin nie kończyły zleconych im zadań w ustalonym czasie dwudziestu sekund. W teście odczytu i zapisu Xamarin był szkieletem, który potrzebował najwięcej zasobów procesora. Testy obsługi żądań HTTP wykazały, że dla systemu Android Flutter oraz rozwiązania natywne najmniej obciążały procesor. Natomiast pracując w systemie iOS Flutter jako jedyny nie ukończył zadania. Autorzy artykułu podsumowali zrealizowane testy oraz swoje doświadczenia związane z procesem tworzenia aplikacji i wskazali na szkielet Flutter jako rozwiązanie najlepsze spośród pozostałych analizowanych szkieletów.

3. Metoda badawcza

Przeprowadzone w ramach pracy badania miały na celu porównanie dwóch technologii, szkieletów Xamarin i Fultter, pod względem ich wydajności. Wydajność rozumiana tu była jako czas wykonania poszczególnych operacji. W tym celu dokonano pomiarów czasów obliczenia trzydziestego wyrazu ciągu Fibonacciego, posortowania algorytmem przez scalanie list składających się z pięciu tysięcy oraz dziesięciu tysięcy elementów, wykonania na bazie danych operacji zapisu, odczytu, wyszukiwania, modyfikacji i usuwania danych. Pomiary dla każdej operacji były powtarzane dziesięć tysięcy razy, a ich wyniki poddano uśrednianiu. Badania przeprowadzono na platformie mobilnej Android oraz na komputerze osobistym.

3.1. Aplikacje testowe

W celu przeprowadzenia badań, a następnie analizy zebranych wyników, utworzono dwie aplikacje testowe. Pierwsza została zbudowana za pomocą szkieletu programistycznego Flutter 2.10.5 wykorzystując do tego język Dart w wersji 2.17.1. Druga z aplikacji została utworzona w szkielecie programistycznym Xamarin 17.1.0.329 w języku C# w wersji 7.3. W obu aplikacjach zaimplementowano funkcjonalności wymagane do przeprowadzenia scenariuszy badawczych. Aplikacje testowe zostały uruchomione w trybie „release", ponieważ szczególnie w przypadku języka Dart i frameworka Flutter ma to znaczenie, gdyż stosowane są różne sposoby kompilacji projektu (JIT lub AOT) [9].

3.2. Środowiska testowe

Testy aplikacji na platformie mobilnej zostały przeprowadzone na tym samym środowisku emulowanym, aby wyniki były jak najmniej zakłócone przez inne procesy działające na urządzeniu. Konfiguracja tego środowiska znajduje się w Tabeli 1.

Tabela 1: Konfiguracja mobilnego środowiska testowego

Procesor	Google APIs Intel Atom (x86_64)
System operacyjny	API level 30
Pamięć RAM	1536 MB
Ilość rdzeni procesora	4

Druga część badań była przeprowadzona na komputerze osobistym w identycznych warunkach, czyli w trybie wysokiej wydajności ze stałym dostępem do prądu. Na komputerze tym, którego parametry przedstawione są w Tabeli 2, został uruchomiony emulator.

Tabela 2: Konfiguracja testowego komputera osobistego

Procesor	Intel Core i7-11370H 12MB
Pamięć RAM	32GB DDR4 3200MHz
Dysk SSD	Micron MTFDHBA512QFD
System operacyjny	Windows 10 Pro 64-bit
Karta graficzna	NVIDIA GeForce RTX 3050

Do mierzenia czasów realizacji testów zostały użyte różne biblioteki dla Fluttera i Xamarina, posiadające tą samą nazwę Stopwatch [10, 11].

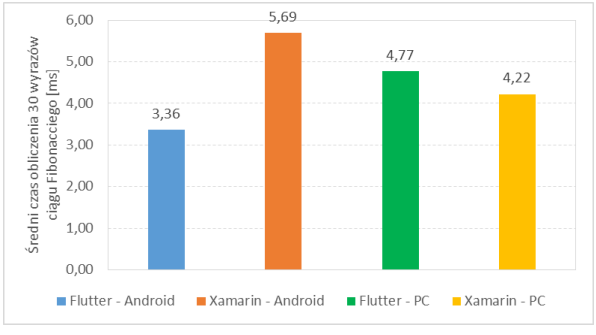
3.3. Kryteria porównawcze i scenariusze testowe

Przed wykonaniem badań na dwóch szkieletach programistycznych dobrano kryteria, względem których będą one porównywane oraz opracowano scenariusze badawcze, za pomocą których aplikacje będą testowane. Ustalono, że głównym kryterium porównawczym będzie czas wykonania zadania. Scenariusze dotyczyły realizacji prostych zadań, podczas których rejestrowany był czas. Badania były przeprowadzone według następujących scenariuszy testowych:

- obliczenie ciągu Fibonacciego do 30-tego wyrazu,
- sortowanie algorytmem przez scalanie losowo wygenerowanych list o wielkości 5000 i 10 000,
- zapis 10 000 rekordów do lokalnej bazy danych,
- odczyt pierwszych 10 000 rekordów,
- modyfikacja 5000 losowo wybranych rekordów,
- odczyt 5000 losowo wybranych rekordów,
- usunięcie 10 000 rekordów.

4. Wyniki badań

Każdy ze scenariuszy został powtórzony 10 000 razy dla każdej aplikacji testowej, na tych samych zestawach danych. Rysunki 1-3 dotyczą pierwszych dwóch scenariuszy testowych, natomiast pozostałe to scenariusze związane z operacjami na bazie danych.

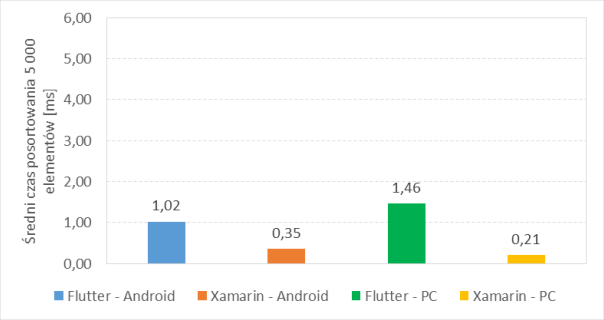


Rysunek 1: Średni czas obliczenia trzydziestego wyrazu ciągu Fibonacciego.

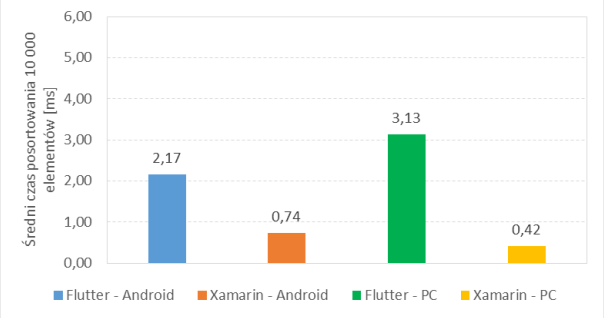
Scenariusz, w którym wyliczano trzydziesty wyraz ciągu miał za zadanie sprawdzić, jak aplikacje bazujące na danym szkielecie programistycznym radzą sobie z zadaniami liniowymi. Na Rysunku 1 widoczna jest duża różnica między wynikami dla aplikacji działają-

cych na systemie Android. Flutter osiągnął znacznie krótszy czas realizacji zadania niż Xamarin. Flutter był o więcej niż 2 ms szybszy. Natomiast na systemie Windows wyniki dla obu aplikacji przy realizacji tego samego scenariusza były zbliżone, choć nieznaczną przewagę uzyskało rozwiązanie opartego na języku C#.

W kolejnym scenariuszu badano, jak aplikacje oparte na szkieletach Flutter i Xamarin radzą sobie z sortowaniem listy algorytmem MergeSort. Z rysunków 2 i 3 widać, że z tym zadaniem znacznie lepiej poradził sobie Xamarin na obu platformach. W teście polegającym na posortowaniu 5 000 elementów i 10 000 elementów, różnice są proporcjonalne do liczby elementów do posortowania.

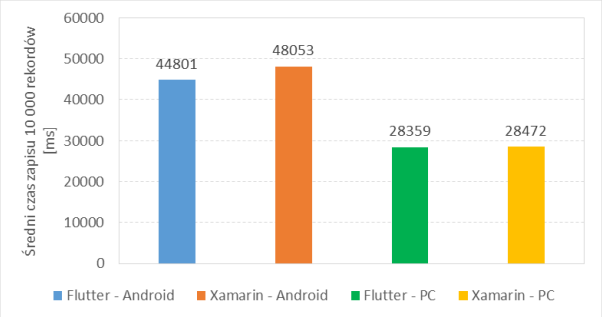


Rysunek 2: Średnie czasy sortowania listy 5 000 elementów algorytmem MergeSort.



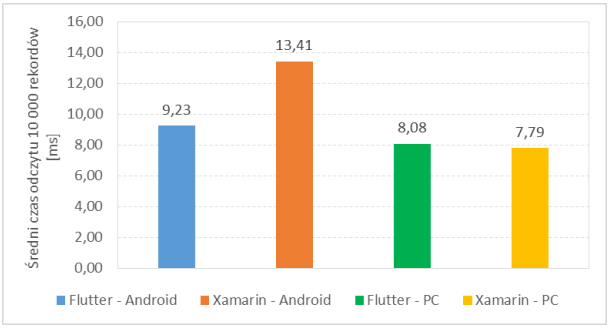
Rysunek 3: Średnie czasy sortowania listy 10 000 elementów algorytmem MergeSort.

Następnie scenariusze testowe dotyczyły operacji na bazie danych. W tym teście Flutter osiągnął znaczną przewagę nad Xamarinem w środowisku mobilnym. Flutter był szybszy o ponad 3 sekundy podczas zapisu rekordów do bazy (Rysunek 4). Na komputerze stacjonarnym przewaga Fluttera była bardzo mała i wyniosła niecałe 100 milisekund.



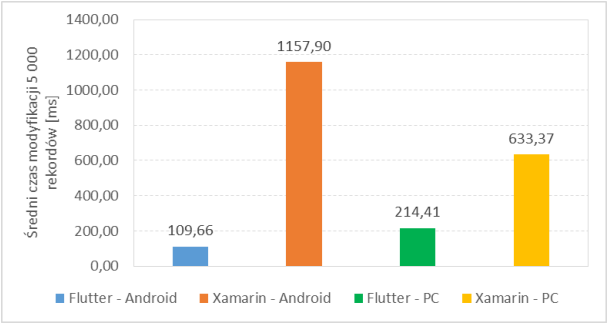
Rysunek 4: Średnie czasy wykonania 10 000 operacji zapisu rekordów do bazy danych.

W scenariuszu testowym polegającym na odczycie 10 000 rekordów, na platformie mobilnej Flutter ponownie był szybszy niż Xamarin (Rysunek 5). Jego średni czas odczytu był o 4,18 ms krótszy. Na platformie Windows różnice były niewielkie. W tym przypadku Xamarin był o 0,29 ms szybszy.



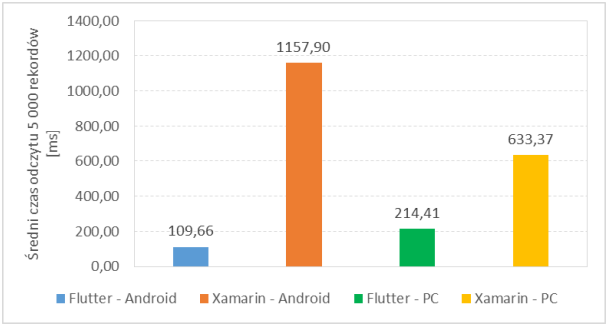
Rysunek 5: Średnie czasy odczytu pierwszych 10 000 rekordów z bazy danych.

Scenariusz dotyczący modyfikacji losowo wybranych rekordów wskazał, na znaczną przewagę Fluttera, w obu środowiskach testowych (Rysunek 6). W środowisku mobilnym Xamarin był ponad 8 razy wolniejszy od szkieletu Google'a. W drugim środowisku testowym przewaga ta zmalała. W tym przypadku Flutter był prawie 3-krotnie szybszy od Xamarina.



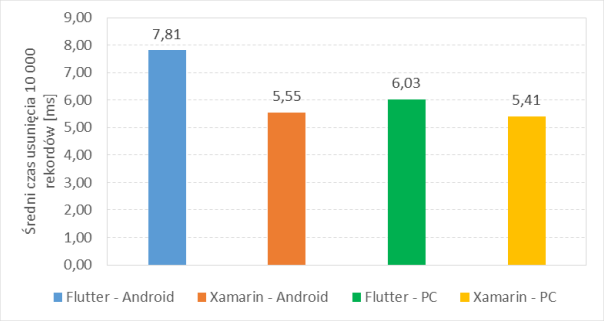
Rysunek 6: Średnie czasy modyfikacji 5 000 losowo wybranych rekordów z bazy danych.

Kolejny scenariusz polegał na odczycie 5 000 losowo wybranych rekordów z bazy danych. Podobnie jak w poprzednim scenariuszu Flutter był szybszy na obu platformach testowych (Rysunek 7). Na platformie mobilnej Xamarin był wolniejszy o ponad sekundę, natomiast na komputerze osobistym był wolniejszy o 418,96 ms.



Rysunek 7: Średnie czasy odczytu 5000 losowo wybranych rekordów z bazy danych.

Również w scenariuszu usunięcia 10 000 rekordów z bazy danych Xamarin okazał się szybszy na obu platformach testowych (Rysunek 8). Na platformie z zainstalowanym systemem Android miał on czas krótszy od Fluttera o 2,26 ms. Natomiast w środowisku z systemem Windows różnica ta była znacznie mniejsza i wyniosła jedynie 0,6 ms.



Rysunek 8: Średnie czasy usunięcia 10 000 rekordów z bazy danych.

5. Podsumowanie

Celem badań było porównanie testowych aplikacji mobilnych na system Android oraz testowych aplikacji na system Windows zbudowanych przy pomocy szkieletów programistycznych Flutter oraz Xamarin. Na potrzeby eksperymentu opracowane zostały dwie aplikacje z identycznymi funkcjonalnościami. Następnie przeprowadzono pomiary czasu realizacji każdego scenariusza testowego, a wyniki uśredniono. Uzyskane wartości czasu stanowiły podstawę do porównania obu szkieletów programistycznych.

Pierwsze trzy scenariusze, związane z wydajnością obliczeniową szkieletów programistycznych nie pokazały jednoznacznych wyników. Wyliczenie n-tego wyrazu ciągu Fibonnaciego na platformie mobilnej wskazało na wyższość Fluttera, natomiast testy na komputerze osobistym wskazywały na Xamarina. W kolejnym scenariuszu, w którym sortowano wartości z listy składającej się z 5 000 elementów i z listy składającej się z 10 000 elementów, wykorzystano algorytm sortowania przez scalanie. W obu przypadkach i na obu platformach testowych szkielet programistyczny Microsoftu uzyskał znaczną przewagę.

Druga cześć eksperymentu dotyczyła realizacji operacji na bazie danych. W tych badaniach Flutter okazał się szybszy w trzech na pięć przypadków testowych. Te trzy przypadki to operacje zapisu danych, wyszukania rekordu w bazie oraz wyszukania i modyfikacji rekordu w bazie. Natomiast Xamarin miał przewagę w operacjach odczytu wszystkich rekordów oraz usunięciu wszystkich rekordów.

Po analizie otrzymanych wyników, trudno jednoznacznie określić, który z dwóch porównywanych szkieletów programistycznych jest wydajniejszy. Jednak można potwierdzić hipotezę, że dla aplikacji na system Android i Windows, które mocno bazują na obliczeniach lub są to aplikacje, które będą wymagały sporej ilości zapisywania danych i ich wyszukiwania wraz z modyfikowaniem, lepszym rozwiązaniem będzie szkielet programistyczny Flutter.

Należy jednak zaznaczyć, że badania te były ograniczone jedynie do dwóch platform testowych i nie badały innych możliwych scenariuszy wykorzystania aplikacji. Przy wyborze wieloplatformowego szkieletu programistycznego należy także wziąć pod uwagę system operacyjny, na którym głównie będzie pracowała aplikacja.

Literatura

[1] Liczba dostępnych aplikacji w Google Play Store, https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/, [07.06.2022].

[2] Liczba dostępnych aplikacji w Apple App Store, https://www.statista.com/statistics/779768/number-of-available-apps-in-the-apple-app-store-quarter/, [07.06.2022].

[3] Flutter, https://flutter.dev/, [07.06.2022].

[4] Xamarin, https://docs.microsoft.com/en-us/xamarin/, [07.06.2022].

[5] G. Versluis, A Brief History of Xamarin. In Xamarin Forms Essentials, Apress, Berkeley, CA, (2017) 3-18.

[6] Y. Rasmusson Wright, S. Hedlund, Cross-platform Frameworks Comparison: Android Applications in a Cross-platform Environment, Xamarin Vs Flutter. (2021) 1-45.

[7] M. Anwar, Comparison and evaluation of cross-platform framework and development of a digital health platform using selected framework (2021) 1-30.

[8] P. Nawrocki, K. Wrona, M. Marczak, B. Śnieżyński, A comparison of native and cross-platform frameworks for mobile applications, Computer, 54(3) (2021) 18-27.

[9] Dart overview, https://dart.dev/overview, [12.09.2022].

[10] Stopwatch class, https://docs.microsoft.com/pl-pl/dotnet/api/system.diagnostics.stopwatch?view=net-6.0, [12.09.2022].

[11] Stopwatch class, https://api.flutter.dev/flutter/dart-core/Stopwatch-class.html, [12.09.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 371–378
Received: 1 August 2022
Accepted: 20 September 2022

Interface usability analysis of selected streaming services in Poland

Analiza użyteczności interfejsów wybranych serwisów streamingowych w Polsce

Paweł Nankiewicz*, Mateusz Niemczuk*, Małgorzata Plechawska-Wójcik

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

Abstract
The aim of this paper is to analyse the interface quality of selected services providing video-on-demand on the Polish internet market. Interfaces of three services were evaluated (Netflix, Amazon Prime Video, and Player). The evaluation was conducted using an expert review, an eye-tracking study, and an accessibility assessment using the WAVE web tool. For the study, a research thesis was established – Netflix has the most usable interface among the three streaming portal interfaces studied. As a result of the evaluation of three research methods, the thesis has been confirmed.

Keywords: web usability; web accessibility; eye-tracking; heuristic evaluation; VoD platforms

Streszczenie
Celem niniejszej pracy jest dokonanie analizy jakości interfejsu wybranych serwisów udostępniających usługę wideo na życzenie na polskim rynku internetowym. Ocenie poddano interfejsy trzech serwisów: Netflix, Amazon Prime Video oraz Player. Ewaluację przeprowadzono, wykorzystując analizę ekspercką, badanie okulograficzne oraz ocenę dostępności internetowym narzędziem WAVE. Na potrzeby badania postawiono tezę badawczą – wśród trzech badanych interfejsów portali streamingowych, serwis Netflix posiada najbardziej użyteczny interfejs. W wyniku przeprowadzonej ewaluacji trzema metodami badawczymi została ona potwierdzona.

Słowa kluczowe: użyteczność WWW; dostępność WWW; okulografia; ocena heurystyczna; platformy VoD

*Corresponding author
Email address: pawel.nankiewicz@pollub.edu.pl (P. Nankiewicz), mateusz.niemczuk@pollub.edu.pl (M. Niemczuk)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Ze względu na stale rosnącą popularność serwisów streamingowych, zdecydowano się dokonać ewaluacji użyteczności graficznych interfejsów (GUI, ang. Graphical User Interface) wybranych platform udostępniających usługę wideo na życzenie (ang. Video on Demand) w Polsce.

Istnieje wiele metod badania użyteczności interfejsów [1], które można klasyfikować według różnych aspektów. Pierwsza klasyfikacja umożliwia podział metod na automatyczne, gdzie procedura oceny jest realizowana w znacznym stopniu przy użyciu komputera oraz manualne, mogące być tylko w niewielkim stopniu wspomagane narzędziami komputerowymi, a główna waga oceny jest realizowana z udziałem człowieka. Wyróżnia się także klasyfikację dokonującą podziału ze względu na sposób realizacji. Wśród nich wymienia się testowanie, inspekcje, wywiad, modelowanie analityczne czy symulacje. Metody badania użyteczności interfejsu można podzielić także na wymagające udziału użytkowników lub oparte na badaniach ekspertów. Wśród pierwszej grupy najczęściej stosuje się test Kruga [2], badanie okulograficzne (ang. Eyetracking) [3] czy zbieranie opinii użytkowników oprogramowania (ang. User Feedback). Metody eksperckie to natomiast głównie analiza ekspercka (ang. Expert Review) [4], uproszczona wędrówka poznawcza (ang. Cognitive Walkthrough) [5], rozwinięta wędrówka poznawcza (ang. Pluralistic Walkthrough) czy ocena heurystyczna (ang. Heuristic Evaluation) [6].

Analiza ekspercka jako metoda analizy jakości interfejsu jest jedną z częściej stosowanych podczas badań. Wielu badaczy wymienia jej zalety, takie jak łatwość zastosowania przy minimalnym nakładzie czasu, kosztów i zasobów [7]. W porównaniu do metod z udziałem użytkowników końcowych metoda ta nie wymaga analizowania wielu aspektów takich jak odpowiedni wybór grupy badawczej, czy właściwych metod oceny. Umożliwia jednak wykrycie nawet 85% błędów systemu już przy wykorzystaniu pracy od trzech do pięciu ekspertów. Podczas badań [8], których celem było przeanalizowanie graficznego interfejsu użytkownika systemu ERP, z zamiarem jego poprawy, stwierdzono, że metoda ekspercka była właściwym wyborem, a dobrze zaprojektowana lista kontrolna (określona przez autorów mianem „listy LUT") może być zastosowana do oceny GUI różnego rodzaju serwisów i aplikacji internetowych. Lista te nie tylko pozwoliła ocenić interfejs aplikacji, ale wskazała także jego problemy i słabe strony. Umożliwiło to na szybkie skorygowanie błędów oraz wdrożenie poprawek do GUI systemu w krótkim czasie. Inne badanie [6] skonstruowano tak, aby odpowiedzieć na pytanie, czy połączenie oceny heurystycznej i testów oprogramowania z użytkownikami końcowymi dostarcza dokładniejszych rezultatów niż zastosowanie tylko jednej metody. Badanie rozpoczęto od przeprowadzenia analizy eksperckiej serwisu internetowego z udziałem pięciu specjalistów, którzy wykryli 59 problemów

z użytecznością. Następnie ten sam interfejs został zbadany przez sześciu użytkowników końcowych. Znaczna część problemów z użytecznością (90%), które wskazali użytkownicy, została już wcześniej wykryta w pierwszej fazie badań. Konkluzją tych badań jest stwierdzenie, że ocena użytkowników końcowych może być metodą uzupełniającą do ewaluacji metodami eksperckimi. Metod heurystycznych używa się w szerokim zakresie do badania interfejsów serwisów stworzonych do różnych potrzeb. Przeprowadzane były badania [9] dotyczące zarówno stron instytucji rządowych, jak i systemów opieki medycznej. Przykładem jest analiza użyteczności interfejsu Narodowego Systemu Zdrowia w Iranie, w jej wyniku przedstawiono szereg problemów oraz krytycznych błędów. Eksperci przedstawili praktyczne rekomendacje, które powinny być wdrożone, aby zwiększyć użyteczność systemu.

Okulografia, jako kolejna metoda badań, stosowana jest do analizy jakości interfejsu. Wyniki są przedstawiane w różnych formach, ułatwiających analizę rezultatów m.in. mapy fiksacji, mapy cieplne czy filmy z punktem skupienia wzroku [1]. Naukowcy z Uniwersytetu na Florydzie przeprowadzili badanie, którego celem była ocena użyteczności i wymagań dotyczących zadań poznawczych w internetowym środowisku nauczania Algebra Nation [10]. W jego czasie wykonano analizę zależności pomiędzy danymi eyetrackingowmi a standardowymi danymi z testów użyteczności, które skoncentrowane były na skuteczności i wydajności studentów podczas wykonywania zadań. We wnioskach przedstawiono stwierdzenie, że analiza ruchu gałek ocznych dostarcza więcej informacji na temat tego, w jaki sposób użytkownicy reagowali na wizualne elementy interfejsu aplikacji. W konsekwencji metoda okulograficzna uzupełniła tradycyjne metody testowania jakości interfejsu aplikacji webowych, dostarczając jednocześnie ważnych danych, których analiza w jeszcze większym stopniu może przyczynić się do poprawy warstwy graficznej systemu. Badania eye-trackingowe stosuje się także w obszarach związanych z marketingiem sieciowym [11], w celu zwiększenia atrakcyjności oraz poprawy intuicyjności interfejsu. Kwestie ergonomii oraz intuicyjności interfejsu stały się także przedmiotem badań naukowców [12], którzy przeanalizowali problematykę spadku zdolności poznawczych u starszych ludzi. Zrozumienie różnic wynikających z wieku użytkowników w poruszaniu się po serwisach internetowych może stać się ważnym aspektem w tworzeniu interfejsów aplikacji internetowych w epoce starzejących się społeczeństw oraz cyfryzacji coraz większej liczby usług. Obszar nawigacji w interfejsie graficznym oraz zdolność do efektywnego odnajdowania jego właściwych elementów były przedmiotem badań [13, 14]. Metody okulograficzne zostały wykorzystane jako narzędzie pomocnicze w ocenie właściwego rozmieszczenia elementów w kabinie statku pasażerskiego czy też zachowania użytkownika w pracy z interfejsem nawigacyjnym map internetowych. W obu badaniach zaprezentowano użyteczność tej metody w tworzeniu interfejsów przyjaznych użytkownikom końcowym. Użycie wielu technik badawczych podczas przeprowadzania analiz umożliwia szersze spojrzenie na badane zagadnienie. Naukowcy z Politechniki Lubelskiej [15] w swojej pracy wykorzystali scenariusze oraz ankiety użytkownika, nagrania video, protokół współbieżnego myślenia na głos oraz indywidualny wywiad pogłębiony. Wszystkie te metody, wraz z zastosowaniem okulografii, pozwoliły na opracowanie kompleksowej oceny, a w następstwie rekomendacji mających na celu zwiększenie użyteczności usług internetowej platformy do fakturowania.

Ważnym aspektem w tworzeniu oprogramowania jest dostosowanie jego warstwy graficznej do potrzeb wszystkich użytkowników, także tych z niepełnosprawnościami lub zaburzeniami poznawczymi. Metody automatyczne umożliwiają analizę użyteczności interfejsu zgodnie z obowiązującymi standardami dostępności WCAG (ang. Web Content Accessibility Guidelines). Naukowcy z Uniwersytetu Technicznego w Minna (Nigeria) przeprowadzili badania użyteczności interfejsu stron internetowych [16], należących do 36 uniwersytetów federalnych w kraju, kładąc szczególny nacisk na obszar dostępności dla użytkowników. Po przebadaniu stron narzędziami automatycznymi (WAVE, HERA, Achecker) stwierdzono, że wszystkie serwisy posiadają wiele błędów w badanym obszarze, zatem nie są w pełni zgodne z WCAG. Ewaluacja pozwoliła autorom na wypracowanie rekomendacji, dotyczących elementów wymagających poprawy i przystosowania serwisów do standardu dostępności. Podobne badania [17] przeprowadzili naukowcy z Australii, którzy skupili się na analizie 30 stron internetowych B2C (ang. business-to-consumer). Celem analizy było wskazanie błędów obniżających poziom satysfakcji konsumentów w różnym wieku i z różnym stopniem niepełnosprawności. Zaproponowano zalecenia, których wdrożenie poprawi dostępność stron internetowych dla osób z niepełnosprawnością sensoryczną, motoryczną i poznawczą w witrynach e-commerce B2C.

2. Cel i zakers

Celem pracy jest przeprowadzenie analizy jakości interfejsu wybranych serwisów udostępniających usługę wideo na życzenie (ang. Video on Demand) na polskim rynku internetowym. Na podstawie ewaluacji wyłoniono aplikację internetową posiadającą najlepszy interfejs pod względem użyteczności.

Zakres pracy obejmuje dokonanie przeglądu literatury związanej z tematyką badań, dobór metod analizy użyteczności interfejsu, wyselekcjonowanie obiektów badań, zaplanowanie eksperymentów badawczych, przeprowadzenie badań metodą ekspercką, okulograficzną oraz badań dostępności, analizę wyników i sformułowanie wniosków z przeprowadzonej ewaluacji.

Na potrzeby badania postawiono następującą tezę badawczą – wśród trzech badanych interfejsów portali streamingowych, serwis Netflix posiada najbardziej użyteczny interfejs. Sformułowano także szczegółowe pytania badawcze:

1. Która aplikacja posiada najbardziej użyteczny interfejs na podstawie oceny heurystycznej?

2. Który serwis posiada lepiej rozmieszczoną wyszukiwarkę?
3. Która aplikacja posiada najbardziej intuicyjny interfejs odtwarzacza treści wideo?
4. Który portal w największym stopniu spełnia wytyczne WCAG 2.1?

3. Obiekty badań

Badania skupione były na ocenie użyteczności interfejsu wybranych serwisów streamingowych. Podczas selekcji obiektów do badań, zostały przeanalizowane interfejsy wielu platform oraz rankingi popularności serwisów świadczących usługę wideo na życzenie. Ważnym kryterium doboru obiektów była ocena interfejsów pod kątem posiadania tych samych funkcjonalności. Ostatecznie do analizy użyteczności interfejsów wybrano trzy portale streamingowe – Netflix, Amazon Prime Video oraz Player.

4. Plan badań

Podczas badań jakości interfejsu zaplanowano użycie trzech metod badawczych. Ewaluację zrealizowano z wykorzystaniem analizy eksperckiej, badania okulograficznego oraz oceny dostępności metodami automatycznymi. Plan badań określono w następujących etapach:

1. Sformułowanie tezy oraz pytań badawczych.
2. Wybór obiektów badawczych (platform VoD).
3. Przeprowadzenie oceny jakości interfejsu metodą ekspercką.
4. Przeprowadzenie badania eyetrackingowego.
5. Przeprowadzenie ewaluacji dotyczącej dostępności.

4.1. Analiza ekspercka

W części badawczej metodą ekspercką z wykorzystaniem listy LUT (ang. Lublin University of Technology) [4], dobrana grupa ekspertów miała za zadanie dokonać oceny serwisów będących obiektem badań pod względem użyteczności w obszarach m.in. nawigacji i struktury, komunikatów i pomocy dla użytkownika, interfejsu aplikacji czy treści podstron.

Przebadane zostały podstrony i funkcjonalności dostępne w analizowanych serwisach VoD m.in. strony pomocy, strony z biblioteką multimediów, odtwarzacz wideo, ustawienia, a także funkcjonalność logowania, wyszukiwania, filtrowania i dodawania treści do list ulubionych.

Wybrano grupę czterech osób mających wiedzę dotyczącą obszarów związanych z informatyką oraz pracą z interfejsami współczesnych aplikacji internetowych. Byli to studenci studiów magisterskich drugiego roku na kierunku Informatyka z Politechniki Lubelskiej. Grupa ta nie brała udziału w pozostałych eksperymentach tj. badaniu eyetrackingowym oraz analizie dostępności.

W eksperymencie użyto listę LUT, którą dostosowano pod obiekty badań. Oryginalna lista była wykorzystana wcześniej w innych badaniach na Politechnice Lubelskiej [4]. Usunięto z niej obszar związany z prowadzaniem danych, ponieważ w ser-wisach streamingowych funkcjonalność ta nie jest istotna. Wprowadzanie danych jest ograniczone do uwierzytelnienia użytkownika oraz wyszukiwania treści. Zrezygnowano także z pytań dotyczących poprawnego zachowania się serwisu na urządzeniach o różnych rozdzielczościach. Spowodowane było to faktem, że każdy z badanych serwisów posiada aplikację mobilną, w konsekwencji czego użytkownicy nie używają wersji przeglądowej na urządzeniach przenośnych. Oceny poszczególnych obszarów w zastosowanej liście wyrażone są w pięciostopniowej skali Likerta. Opis ocen, ich interpretację i znaczenie zostały opisane w publikacji „Memorability Experiment Vs. Expert Method in Websites Usability Evaluation" [4].

Wyniki przyjętej metody oceny wykorzystano do obliczenia metryki WUP (ang. Web Usability Points) [4] – złożonego współczynnika użyteczności stron internetowych. Metryka WUP wykorzystuje oceny przyznawane przez ekspertów dla każdego pytania z listy LUT. Wartość WUP waha się od 1 do 5. Im wyż-sza jej wartość tym użyteczność interfejsu jest większa.

Badanie metodą ekspercką rozpoczęto od przedstawienia uczestnikom celu badania oraz przebiegu procesu ewaluacji. Następnie eksperci przystąpili do analizy aplikacji będących obiektem badań. W tej części badania nie zastosowano żadnych ram czasowych dla oceny. Następnie przedstawiono im przygotowaną listę LUT. Eksperci, przeprowadzając ewaluację, odpowiadali na pytania, oceniając obszary zawarte w opracowanej liście. Podczas oceny mieli dostęp do badanych obiektów. Po zakończeniu ewaluacji zebrano listy LUT i rozpoczęto proces analizy otrzymanych wyników. Obliczono współczynniki WUP oraz dokonano dyskusji rezultatów.

4.2. Badanie okulograficzne

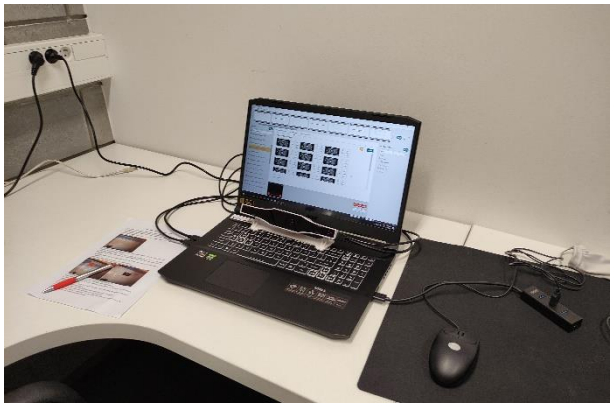
W części badawczej z wykorzystaniem eyetrackera zaplanowano eksperyment, polegający na statycznej analizie wybranych elementów interfejsu badanych serwisów.

Podczas ewaluacji przyjęto strategię, która zakładała wykonanie badań na poszczególnych, wybranych podstronach. W konsekwencji wyselekcjonowano interfejsy realizujące funkcjonalność odtwarzania wideo, a także strony głównej z elementami biblioteki treści multimedialnych.

W skład grupy badawczej weszło 9 osób (dwie kobiety i siedmiu mężczyzn) w wieku 23-25 lat. Wszyscy badani to studenci kierunków technicznych na Wydziale Elektrotechniki i Informatyki na Politechnice Lubelskiej. W grupie badawczej znajdowały się osoby z krótkowzrocznością. Podczas badania miały one założone okulary lub soczewki kontaktowe.

Eksperyment badawczy został przeprowadzony w Laboratorium Akwizycji Ruchu i Ergonomii Interfejsów (LARiEI) Katedry Informatyki Politechniki Lubelskiej. Badania zostały przeprowadzone z udziałem osób pełniących funkcję moderatora. W pomieszczeniu laboratorium zapewniono optymalne warunki dla uczestników badania – stały (niezmienny) poziom oświetlenia, a także ograniczono wszelkie źródła dźwięku z ze-

wnątrz. Na wyposażenie stanowiska badawczego (Rysunek 1) złożyło się: laptop z oprogramowaniem iMotions 9.0, eyetracker Gazepoint GP3 HD [18], myszka, klawiatura oraz ergonomiczne krzesło komputerowe zapewniające prawidłową pozycję podczas badania.



Rysunek 1: Stanowisko badawcze.

Badanie każdego uczestnika przebiegało w taki sam sposób. Na wstępie osoba badana została zapoznana z zasadami oraz przebiegiem eksperymentu. Każdy z uczestników odpowiedział na pytanie, czy miał wcześniej styczność z którymkolwiek z serwisów będących obiektem badań (wszystkie odpowiedzi były akceptowane). Następnie, gdy uczestnik zajął miejsce na stanowisku badawczym, przeprowadzono kalibrację sprzętu. Uczestnicy podczas badania nie korzystali z myszki, używali jedynie klawisza spacji powodującego przejście do kolejnego zadania szybciej niż domyślny czas trwania zadania, czyli 10 sekund. Podczas badania uczestnikom przedstawiono do wykonania zadania, które polegały na znalezieniu na badanej stronie określonych elementów interfejsu:

1. Znajdź element interfejsu, którego użyjesz by włączyć napisy podczas odtwarzania.
2. Znajdź element interfejsu odpowiadający za wyszukiwanie treści.
3. Znajdź element interfejsu, którego użyjesz by uruchomić odtwarzacz w pełnym oknie.
4. Znajdź element interfejsu, którego użyjesz by zmienić poziom dźwięku.

Kolejność wyświetlania zadań, ich treść oraz kolejność badanych witryn została dobrana tak, by badany nie mógł przyzwyczaić się do interfejsu jednej z aplikacji, a także by wzrok osoby badanej przed wyświetlaniem każdego interfejsu znajdował się w centrum ekranu.

Po przeprowadzonym eksperymencie, uczestnikom zadano pytanie – która strona posiada najbardziej użyteczny interfejs, a która najmniej. Przyjmowaną odpowiedzią było uszeregowanie nazw badanych serwisów VoD w kolejności od tego, który według uczestników badania ma najbardziej użyteczny interfejs, do tego, który ma najmniej użyteczny. Dodatkowo uczestnicy badania mogli wyrazić swoją opinię na temat interfejsów serwisów będących przedmiotem ewaluacji oraz podzielić się uwagami z przebiegu samego eksperymentu.

Dane otrzymane w wyniku badań metodą okulograficzną przeanalizowano zarówno pod kątem jakościowym, jak i ilościowym. Ewaluacji jakościowej dokonano poprzez analizę map cieplnych oraz ścieżek skanowania. Na potrzeby przeprowadzenia analizy ilościowej wybrano metryki, a na ich podstawie dokonano analizy porównawczej. Skupiono się głównie na badaniu fiksacji i sakad. Sakady to szybkie ruchy oczu, fiksacje natomiast to okresy koncentracji wzroku podczas, których informacje trafiają do mózgu, gdzie są dalej przetwarzane [19]. Wyselekcjonowano następujące metryki ilościowe: czas wykonania zadania, liczba fiksacji, średni czas trwania fiksacji, średni czas trwania sakady oraz średnia amplituda sakady.

Dla czytelnej prezentacji metryk ilościowych zdecydowano się na użycie wykresów pudełkowych (ang. box-plot). Za ich pomocą można odczytać takie dane takie jak: dolny kwartyl (dolna krawędź pudełka), górny kwartyl (górna krawędź pudełka), czarna kropka w pudełku na wykresie oznacza medianę. Położenie wąsów oznacza wartość minimalną (dolny wąs) oraz wartość maksymalna (górny wąs).

4.3. Badanie dostępności

W części badawczej dotyczącej analizy dostępności wykorzystano narzędzie automatyczne, które pozwoliło dokonać sprawdzania, czy badane platformy udostępniające usługę wideo na życzenie spełniają wytyczne WCAG 2.1.

Podczas przygotowania badań dostępności wyselekcjonowano interfejsy serwisów streamingowych, które realizują tę samą funkcjonalność, a częstość ich używania jest największa ze względu na specyfikę badanego obszaru. Ewaluacji poddano interfejsy realizujące funkcjonalność logowania, odtwarzania treści wideo oraz strony głównej z elementami biblioteki treści multimedialnych.

Badanie dostępności rozpoczęto od przygotowania narzędzia WAVE [20]. Zainstalowano wtyczkę do przeglądarki internetowej Microsoft Edge, następnie uruchamiano badane strony i włączano wtyczkę. Po zakończonym badaniu pobierano wyniki oraz widok z narzędzia. Proces ten powtórzono dla każdego badanego interfejsu, we wszystkich trzech porównywanych serwisach VoD.

5. Wyniki

Po przeprowadzeniu eksperymentów badawczych dokonano wnikliwej analizy wyników. Z powodu szerokiego charakteru badań, przedstawione zostały tylko wybrane rezultaty.

5.1. Wyniki analizy eksperckiej

W Tabeli 1 przedstawiono wartości metryki WUP (ang. Web Usability Points), wyliczone dla każdego obiektu badań, na podstawie analizy przeprowadzonej przez każdego eksperta. Metryka została wyznaczona w oparciu o oceny, które eksperci przyznali odpowiadając na pytania zawarte w liście LUT.

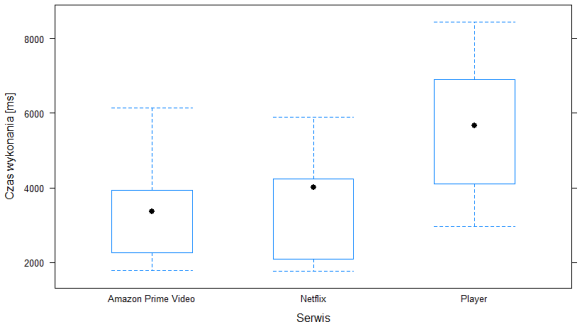
Tabela 1: Wartości metryki WUP (E - oznacza eksperta)

Serwis	E 1	E 2	E 3	E 4	M ± SD
Netflix	4,35	4,48	4,60	4,67	4,52 ± 0,14
Amazon Prime Video	4,00	4,32	4,10	3,58	4,01 ± 0,31
Player	4,08	4,22	2,90	4,30	3,86 ± 0,68

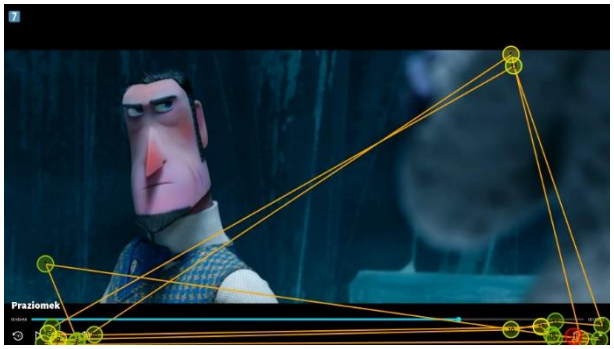
Na podstawie ocen ekspertów można stwierdzić, że interfejs o najwyższej użyteczności posiada serwis Netflix, gdzie wartość średnia wyniosła 4,52, natomiast interfejs o najniższej użyteczności strona Player (M = 3,86). Największe różnice między ocenami ekspertów odnotowano dla aplikacji Player.

5.2. Wyniki badania okulograficznego

Porównanie odmiennego umiejscowienia (w trzech badanych serwisach) elementu interfejsu odpowiedzialnego za włączenie napisów podczas odtwarzania pozwoliło sformułować następujące wnioski. Ukrycie tej funkcji pod ikoną ustawień (zębatki) w serwisie Player jest rozwiązaniem nieintuicyjnym i może powodować zagubienie użytkownika w interfejsie. Świadczy o tym najdłuższy czas wykonania zadania 1 (Rysunek 2) oraz ścieżka skanowania wzroku (Rysunek 3), na której widać, że uczestnik miał problemy z odnalezieniem poszukiwanego elementu interfejsu.



Rysunek 2: Czasy wykonania zadania znalezienia elementu interfejsu, który umożliwia włączenie napisów podczas odtwarzania.

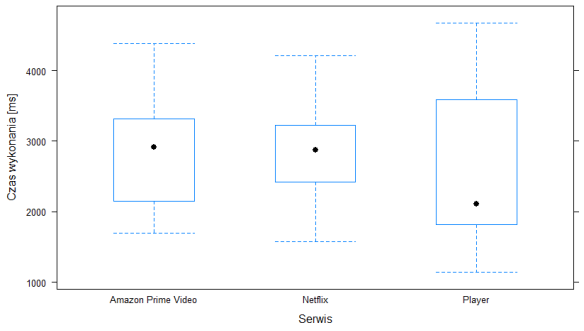


Rysunek 3: Wynik realizacji zadania 1 w postaci ścieżki skanowania wzroku w serwisie Player.

Na Rysunku 4 przedstawiono mapę ciepną realizacji zadania 1, na której można zauważyć, że większość uczestników badania miała problem z odnalezieniem celu. Czerwony kolor oznacza dużą intensywność uwagi badanych w zarówno lewym jak i prawym dolnym rogu ekranu.

Rysunek 4: Wynik realizacji zadania 1 w postaci mapy cieplnej w serwisie Player.

Rozmieszczenie wyszukiwarki treści w różnych miejscach górnego paska nawigacyjnego nie ma dużego wpływu na proces odnalezienia tego elementu przez użytkownika. Ukrycie wyszukiwarki pod ikoną lupy jest rozwiązaniem intuicyjnym i nie wprowadza użytkownika w błąd. Na Rysunku 5 przedstawiono czasy realizacji zadania 2. Można zauważyć, że uczestnicy odnajdowali wyszukiwarkę w podobnym czasie na każdej stronie, przy czym największe zróżnicowanie w czasie wykonania odnotowano dla aplikacji Player.



Rysunek 5: Czasy wykonania zadania znalezienia elementu interfejsu, który umożliwia wyszukiwanie treści.

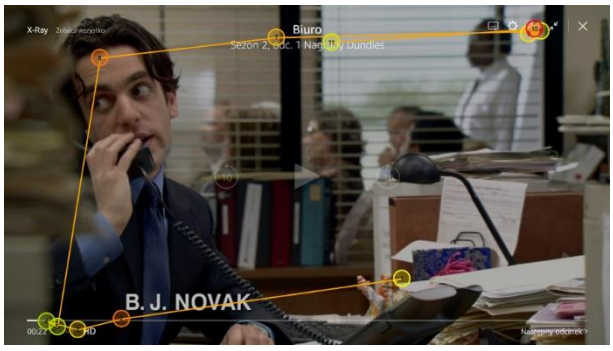
Umieszczenie elementu interfejsu, który umożliwia uruchomienie odtwarzacza w pełnym oknie (zadanie 3), w prawym dolnym rogu (serwis Netflix oraz Player) jest rozwiązaniem bardziej intuicyjnym, niż zlokalizowanie go w prawym górnym rogu ekranu (serwis Amazon Prime Video). Na Rysunkach 6 i 7 przedstawiono ścieżki skanowania wzroku wybranych uczestników badania w aplikacjach Netflix oraz Amazon Prime Video. Widoczne jest, że badany podczas poszukiwania tego elementu interfejsu kierował wzrok w prawy dolny róg ekranu. Gdy nie odnalazł go w tym miejscu w serwisie Amazon Prime Video, skierował wzrok w prawy górny róg, wykonując zadanie poprawnie. Przedstawiona na Rysunku 8 mapa cieplna pokazuje, że inni uczestnicy eksperymentu także spoglądali intuicyjnie w prawy dolny róg w poszukiwaniu celu.

Rysunek 6: Wynik realizacji zadania 3 w postaci ścieżki skanowania wzroku w serwisie Netflix.

Rysunek 7: Wynik realizacji zadania 3 w postaci ścieżki skanowania wzroku w serwisie Amazon Prime Video.

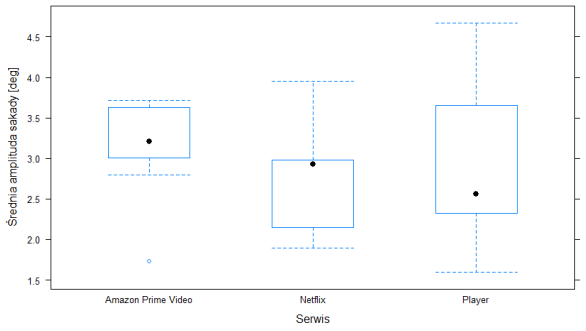
Rysunek 8: Wynik realizacji zadania 3 w postaci mapy cieplnej w serwisie Amazon Prime Video.

Ostatnim zadaniem (4), które zostało zrealizowane przez uczestników badania, było odnalezienie elementu interfejsu odpowiedzialnego za zmianę poziomu głośności podczas odtwarzania. Z uwagi, że prezentacja graficzna poszukiwanego elementu w trzech serwisach jest prawie identyczna (ikona głośnika), analizę oparto głównie o położenie tego elementu na ekranie. Stwierdzono, że poprawną i bardziej intuicyjną lokalizacją tego elementu jest lewy dolny róg. Badani poszukiwali ikony zmieniającej poziom dźwięku właśnie w tym miejscu, co obrazują ścieżki skanowania wzroku (Rysunek 9 i 10). Umieszczenie tego elementu w prawym górnym rogu w serwisie Amazon Prime Video skutkowało najdłuższym czasem realizacji zadania 4, największą liczbą fiksacji oraz największą średnią amplitudą sakady (Rysunek 11).



Rysunek 9: Wynik realizacji zadania 4 w postaci ścieżki skanowania wzroku w serwisie Amazon Prime Video.

Rysunek 10: Wynik realizacji zadania 4 w postaci ścieżki skanowania wzroku w serwisie Player.



Rysunek 11: Średnia amplituda sakady dla zadania znalezienia elementu interfejsu, który umożliwia zmianę poziomu dźwięku.

5.3. Wyniki badania dostępności

W ewaluacji badanych serwisów świadczących usługę wideo na życzenie pod względem cyfrowej dostępności wykorzystano metodologię zawartą w dokumencie WCAG-EM [21]. Spośród badanych aplikacji internetowych wyselekcjonowano podstrony, które realizują tę samą funkcjonalność. Następnie wykorzystano narzędzie automatyczne WAVE do dokonania oceny zgodności badanych obiektów z wymogami WCAG 2.1.

W pierwszej kolejności przebadano trzy interfejsy odpowiadające za funkcjonalność logowania do badanych platform VoD. Na podstawie danych zawartych w Tabeli 2 wskazano, że każda z badanych aplikacji nie jest pozbawiona problemów. Dwie z nich: Netflix oraz Player posiadają problemy z kontrastem, który w sposób znaczący może wpływać na jakość korzystania z aplikacji przez osoby z problemami z narządem wzroku. W obu aplikacjach problemy te występują w obszarze

etykiet i przycisków w panelu logowania, a więc kluczowym z punktu użyteczności miejscu serwisu, liczba wykrytych błędów dla każdej z badanych stron jest podobna. Najmniejsza wykryta liczba alertów została wskazana po analizie strony Player, ale warto też zauważyć, że na tej platformie występuje najwięcej błędów kontrastu. Biorąc pod uwagę zalecenia WCAG należy wskazać najbardziej dostępną stronę realizującą funkcjonalność logowania jako tą będącą częścią platformy Amazon Prime Video.

Tabela 2: Liczba wykrytych problemów na stronach logowania

Serwis	L. błędów	L. błędów kontrastu	L. alertów
Netflix	4	1	7
Amazon Prime Video	4	0	10
Player	3	5	3

W kolejnym etapie poddano ewaluacji interfejsy realizujące funkcjonalność odtwarzacza multimediów. Wyniki zebrane podczas badań przedstawiono w Tabeli 3. W badanym obszarze narzędzie nie wykryło błędów kontrastu na żadnym z analizowanych serwisów. Najmniejszą liczbą problemów w kategorii błędów charakteryzuje się odtwarzacz platformy Netflix – wykryto tylko jeden problem dotyczący zbyt małego rozmiaru czcionki. Wykryty błąd może stanowić poważne utrudnienie dla użytkowników z wadami narządu wzroku. Pozostałe platformy, Player z wykrytą liczbą 11 błędów oraz Amazon Prime Video z wykrytą liczbą 66 błędów, w tym obszarze wypadły gorzej od platformy Netflix. Dominowały głównie błędy związane z brakiem tekstów alternatywnych i występowaniem redundantnych linków. Liczba problemów w kategorii alertów jest najmniejsza dla platformy Netflix. Podsumowując wyniki w części ewaluacji cyfrowej dostępności odtwarzaczy wideo, można stwierdzić, że platforma Netflix udostępnia najbardziej zgodny z wytycznymi WCAG odtwarzacz treści multimedialnych.

Tabela 3: Liczba wykrytych problemów na stronach odtwarzaczy wideo

Serwis	L. błędów	L. błędów kontrastu	L. alertów
Netflix	0	0	12
Amazon Prime Video	36	0	28
Player	2	0	25

Ostatnim etapem badań dostępności była ewaluacja interfejsów realizujących funkcjonalność stron głównych. Zbiorcze wyniki dotyczącego tego obszaru zebrano w Tabeli 4. Na żadnej ze stron nie wykryto błędów kontrastu. Na stronie głównej platformy Netflix nie wykryto żadnych problemów w kategorii błąd, natomiast na stronach platform Player oraz Amazon Prime Video wykryto odpowiednio 1 i 30 błędów. Problemy te związane są najczęściej z brakującymi etykietami. Problemy w kategorii alerty w największej liczbie zostały wykryte na platformie Netflix, ale większość z nich jest związana z nie użyciem przez twórców strony hierarchii nagłówków oraz faktu, że aplikacja ta prezentuje bardzo dużo treści na swojej stronie głównej. Najmniej problemów w tej kategorii znaleziono na serwisie Amazon Prime Video. Podsumowując analizę stron głównych badanych platform streamingowych można stwierdzić, że najlepsze oceny podczas ewaluacji zdobyła platforma Netflix, pomimo dużej liczby wykrytych alertów.

Tabela 4: Liczba wykrytych problemów na stronach głównych

Serwis	L. błędów	L. błędów kontrastu	L. alertów
Netflix	0	0	194
Amazon Prime Video	30	0	9
Player	1	0	36

Wszystkie z przeanalizowanych interfejsów zawierają elementy, które narzędzie WAVE zakwalifikowało do kategorii błędów lub ostrzeżeń. Na podstawie przeprowadzonych ewaluacji stron logowania, odtwarzacza multimediów i stron głównych, wyników zawartych w Tabelach 2-4, a także oceny osób prowadzących badania stwierdzono, że to portal Netflix posiada najlepszy interfejs pod względem cyfrowej dostępności.

6. Wnioski

Celem pracy było przeprowadzenie analizy jakości interfejsu wybranych serwisów VoD na polskim rynku internetowym. Wyniki przeprowadzonej ewaluacji potwierdzają postawioną na początku tezę badawczą – wśród trzech badanych interfejsów portali streamingowych, serwis Netflix posiada najbardziej użyteczny interfejs. Zrealizowanie poszczególnych eksperymentów pozwoliło odpowiedzieć na sformułowane wcześniej pytania badawcze:

1. Która aplikacja posiada najbardziej użyteczny interfejs na podstawie oceny heurystycznej? – Podczas realizacji badań metodą ekspercką, badani dokonali oceny interfejsów serwisów VoD, korzystając z opracowanej ankiety LUT. Na podstawie ocen zostały obliczone wartości metryki WUP (ang. Web Usability Points). Najwyżej oceniona została aplikacja Netflix, zatem według ekspertów posiada ona interfejs najbardziej użyteczny.
2. Który serwis posiada lepiej rozmieszczoną wyszukiwarkę? – Analiza wyników badania okulograficznego pozwoliła stwierdzić, że rozmieszczenie wyszukiwarki w różnych miejscach górnego paska nawigacyjnego nie ma dużego wpływu na proces odnalezienia tego elementu przez użytkownika. W konsekwencji tego nie została wskazana jednoznacznie jedna aplikacja, która posiada najlepiej rozmieszczony element interfejsu służący do wyszukiwania treści.
3. Która aplikacja posiada najbardziej intuicyjny interfejs odtwarzacza treści wideo? – Analiza wyników

z przeprowadzonego eksperymentu okulograficznego pozwoliła stwierdzić, że standardowe umiejscowienie elementów sterujących odtwarzaniem treści jest najbardziej intuicyjnym rozwiązaniem. W serwisie Amazon Prime Video odmienne umiejscowienie tych elementów może wpłynąć na dłuższy czas ich wyszukiwania. Wyniki badań wskazały, że w tym obszarze najbardziej intuicyjny interfejs odtwarzacza treści posiada serwis Netflix.

4. Który portal w największym stopniu spełnia wytyczne WCAG 2.1? – Na podstawie rezultatów badań otrzymanych przy użyciu narzędzia WAVE dostrzeżono, że wszystkie z analizowanych interfejsów aplikacji streamingowych zawierają błędy, które mogą powodować problemy z dostępnością. Analiza wyników pozwoliła stwierdzić, że serwis Netflix posiada najlepszy interfejs pod względem dostępności cyfrowej – zawiera najmniejszą liczbę błędów, w konsekwencji spełnia wytyczne WCAG w największym stopniu.

Po przeprowadzeniu ewaluacji i dokonaniu analizy wyników, autorzy niniejszej pracy dostrzegają obszary do rozszerzenia badań. W celu uzyskania bardziej szczegółowych wyników warte rozważenia byłoby przeprowadzenie eksperymentu eyetrackingowego z większą liczbą zadań, na liczniejszej grupie badawczej. Cenne byłoby również przeprowadzenie ewaluacji użyteczności interfejsów aplikacji mobilnych przebadanych serwisów ze względu na rosnącą popularność urządzeń przenośnych. Zauważono także możliwość zwiększenia liczby badanych serwisów świadczących usługę wideo na życzenie, tak by ująć lepiej różnorodność rynku usług VoD w Polsce.

Literatura

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

[2] S. Krug, Don't make me think!, revisited: A common sense approach to web usability, 3rd Edition, New Riders, 2014.

[3] A. Majooni, A. Akhavan, D. Offenhuber, An eye-tracking study on usability and efficiency of blackboard platform, In 2018 9th International Conference on Applied Human Factors and Ergonomics (2018) 281-289.

[4] M. Miłosz, M. Borys, M. Laskowski, Memorability experiment vs. expert method in websites usability evaluation, Proceedings of the 15th International Conference on Enterprise Information Systems (2013) 176-182.

[5] M. Plechawska-Wojcik, S. Lujan-Mora, Ł. Wojcik, Assessment of user experience with responsive web applications using expert method and cognitive walkthrough: A case study, Proceedings of the 15th International Conference on Enterprise Information Systems (2013) 60-67.

[6] F. Paz, F. A. Paz, D. Villanueva, J. A. Pow-Sang, Heuristic evaluation as a complement to usability testing: A case study in web domain, In 2015 12th International Conference on Information Technology – New Generations (2015) 546-551.

[7] R. Khajouei, A. A. Azizi, A. R. Atashi, Usability evaluation of an emergency information system: A heuristic evaluation, Journal of Health Administration 16 (52) (2013) 61-72.

[8] M. Miłosz, M. Plechawska-Wojcik, M. Borys, M. Laskowski, Quality improvement of ERP system GUI using expert method: A case study, In 2013 6th International Conference on Human System Interactions (2013) 145-152.

[9] F. R. Jeddi, E. Nabovati, R. Bigham, R. Farrahi, Usability evaluation of a comprehensive national health information system: A heuristic evaluation, Informatics in Medicine Unlocked 19 (2020) 100332.

[10] J. Wang, P. Antonenko, M. Celepkolu, Y. Jimenez, E. Fieldman, A. Fieldman, Exploring relationships between eye tracking and traditional usability testing data, International Journal of Human-Computer Interaction 35 (6) (2019) 483-494.

[11] X. Yuan, M. Guo, F. Ren, F. Peng, Usability analysis of online bank login interface based on eye tracking experiment, Sensors & Transducers 165 (2) (2014) 203-212.

[12] J. C. Romano Bergstrom, E. L. Olmsted-Hawala, M. E. Jans, Age-related differences in eye tracking and usability performance: Website usability for older adults, International Journal of Human-Computer Interaction 29 (8) (2013) 541-548.

[13] Y. Liu, T. Xi, Study on the interface usability of light control panel in large cruise ship cabin based on eye movement experiment, In 2019 6th International Conference on Information Science and Control Engineering (2019) 913-922.

[14] S. M. Manson, L. Kne, K. R. Dyke, J. Shannon , S. Eria, Using eye-tracking and mouse metrics to test usability of web mapping navigation, Cartography and Geographic Information Science 39 (1) (2012) 48-60.

[15] M. Miłosz, M. Chmielewska, Usability testing of e-government online services using different methods – A case study, In 2020 13th International Conference on Human System Interaction (2020) 142-146.

[16] S. A. Adepoju, I. S. Shehu, Usability evaluation of academic websites using automated tools, In 2014 3rd International Conference on User Science and Engineering (2014) 186-191.

[17] O. Sohaib, K. Kang, E-commerce web accessibility for people with disabilities, Complexity in Information Systems Development, Springer, 2017.

[18] Parametry techniczne eyetrackera Gazepoint GP3 HD, https://www.gazept.com/product/gp3hd, [06.05.2022].

[19] A. Bojko, Eye tracking the user experience: A practical guide to research, Rosenfeld Media, 2013.

[20] WAVE Web Accessability Evaluation Tool, https://wave.webaim.org, [19.05.2022].

[21] WCAG 2 Overview, https://www.w3.org/WAI/standards-guidelines/wcag, [12.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 379–383
Received: 16 August 2022
Accepted: 25 September 2022

Comparative analysis of Web application development on Java and PHP

Analiza porównawcza wytwarzania aplikacji internetowych na przykładzie Javy oraz PHP

Kacper Truszkowski*, Maciej Pańczyk

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

Abstract

This article presents a comparative analysis of two programming languages used to develop web applications. Two most popular programming languages Java and PHP were compared. The evaluated criteria were: implementation and performance, the time of performing specific operations on 100, 1000 and 10000 records was checked. A code analysis was carried out to determine in which language it is easier to implement an application with identical functionalities. Before the implementation of the application, articles, books and websites on a similar topic were reviewed. In order to test the efficiency, the Postman developer tool was used. The comparison shows that PHP is a more efficient language for developing web applications than Java.
Keywords: Java; PHP; web application

Streszczenie

Niniejszy artykuł przedstawia analizę porównawczą dwóch języków programowania używanych do wytwarzania aplikacji internetowych. Porównano dwa najbardziej popularne języki programowania Java oraz PHP. Oceniane kryteria to: implementacja oraz wydajność. Sprawdzono czas wykonywania konkretnych operacji na 100, 1000 oraz 10000 rekordach. Została przeprowadzona analiza kodu w celu ustalenia w którym języku łatwiej jest zaimplementować aplikację posiadającą identyczne funkcjonalności. Przed implementacją aplikacji, dokonano przeglądu artykułów, książek oraz stron internetowych poświęconych podobnemu zagadnieniu. W celu zbadania wydajności wykorzystano narzędzie developerskie Postman. Z porównania wynika, że PHP jest wydajniejszym językiem wytwarzania aplikacji internetowych niż Java.
Słowa kluczowe: Java; PHP; aplikacje internetowe

*Corresponding author

Email address: **kacper.truszkowski@pollub.edu.pl** (K. Truszkowski)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

W ciągu ostatnich lat wzrosło zapotrzebowanie na usługi informatyczne, rozwinęło się wiele nowych języków programowania, a wiele z nich zostało unowocześnionych. Wytworzone narzędzia mają na celu zwiększenie bezpieczeństwa oraz wydajności tworzonych aplikacji, jednocześnie ułatwiając ich implementację. Obecnie tworzone aplikacje są wdrażane w oparciu o wiele technologii, które się wzajemnie uzupełniają. Programiści maja do dyspozycji coraz nowsze i bardziej zaawansowane szkielety programistyczne. Wspomniane szkielety odpowiadają za czynności, które kiedyś były zadaniem programisty. Aplikacje internetowe powoli wypierają z naszego życia aplikacje desktopowe. Bardzo dynamicznie rozwijająca się technologia pozwala na wykorzystanie dużej elastyczności narzędzi programistycznych. Brak konieczności instalacji specjalnego oprogramowania na komputerze użytkownika czy też niezależność systemu operacyjnego składają się na dużą popularność aplikacji internetowych. W niniejszej pracy zestawiono analizę wydajnościową najpopularniejszych języków do wytwarzania aplikacji internetowych Java oraz PHP [1]. Dzięki analizie porównawczej możliwe było wyciągniecie wniosków o tym, jaka technologia jest wydajniejsza, w jakim scenariuszu.

2. Cel i zakres badań

Celem poniżej przedstawionego artykułu jest przeprowadzenie analizy porównawczej dwóch języków programowania stosowanych do wytwarzania aplikacji internetowych Java oraz PHP. Badaniom poddano wydajność oraz przeprowadzono analizę kodu, dzięki której można było określić w którym języku łatwiej jest zaimplementować małą aplikację internetową. Postawiono następujące tezy badawcze:

T1: Dla mniejszej aplikacji internetowej bardziej wydajnym językiem programowania będzie PHP.

T2: Dla aplikacji napisanej w języku Java aby otrzymać podobne wyniki czasowe co w przypadku języka PHP są wymagane większe zasoby sprzętowe.

T3: Implementacja aplikacji internetowej w języku PHP jest prostsza i mniej rozbudowana.

3. Przegląd literatury

Analizując obecną literaturę niełatwo znaleźć zestawienie porównujące języki Java oraz PHP. Porównanie tych języków stanowi pewne wyzwanie z powodu różnic jakie wynikają podczas implementacji aplikacji. Artykułem, który pod pewnymi kryteriami jest zbliżony do niniejszej pracy, jest artykuł pt. „Tworzenie aplikacji internetowych na platformie JEE I PHP – analiza porównawcza”[2]. Autor zestawia wyżej wymienione technologie, jednak podczas tworzenia aplikacji PHP użył szkieletu programistycznego Laravel. Oznacza to że osiągnięte wyniki będą się różnić od aplikacji napisanej w czystym PHP. Autor porusza w swojej pracy między innymi takie zależności jak:

- wydajność podczas operowania na danych z bazą danych,
- bezpieczeństwo,
- analiza kodu podczas implementacji.

Powyższe kryteria poza kryterium dotyczącym bezpieczeństwa są równoważne z zakresem badawczym niniejszego artykułu i pozwalają na wyznaczenie technologii, która będzie lepsza w analizowanych scenariuszach

4. Analizowane technologie

Badania zostały przeprowadzone na dwóch aplikacjach napisanych w różnych językach programowania Java oraz PHP. Języki te bardzo różnią się od siebie. Pierwszy z nich jest językiem obiektowym, który wymaga na urządzeniu posiadania maszyny wirtualnej (ang. Java Virtual Machine, JVM). PHP w przeciwieństwie do Javy nie wymaga maszyny wirtualnej. Jest to jednak język strukturalny, który zawiera elementy języka obiektowego [3].

4.1. Java

Java jest obiektowym językiem programowania ogólnego zastosowania. Podstawowa część składni została zaczerpnięta z języka C++. Programy wykonywane w tym języku są kompilowane do postaci binarnej, którą wykonuje maszyna wirtualna [4]. Głównymi zaletami języka są:

- obiektowość,
- niezawodność oraz bezpieczeństwo.

4.2. PHP

PHP to skryptowy język, który służy do implementowania aplikacji webowych w czasie rzeczywistym. Większość powszechnie używanych aplikacji internetowych, np. Facebook czy Wikipedia, jest napisana w tym języku. Język PHP najczęściej wykorzystuje się do obsługi formularzy na stronach internetowych [5].

5. Metoda badań

W celu przeprowadzenia badań zostały zaimplementowane dwie aplikacje napisane w języku Java oraz PHP. Zawierają one identyczne funkcjonalności. Aplikacje nie mają zaimplementowanej części wizualnej aby wydajność była badana w miarodajny sposób. Pomiary zostały wykonane przy użyciu narzędzia Postman [6]. Dane w obu przypadkach były generowane poprzez bibliotekę Faker [7]. Biblioteka Faker umożliwiła uzupełnienie bazy losowymi danymi na których były wykonywane operację zapisywania, wyświetlania, edycji oraz usuwania danych. Poniżej zostały przedstawione fragmenty kody odpowiadające za generowanie danych w obu aplikacjach (Listing 1, Listing 2)

Listing 1: Fragment kodu odpowiadający za generowanie danych w aplikacji webowej napisanej w języku PHP

```
<?php
require_once "vendor/autoload.php";
$faker = Faker\Factory::create();
$db = new PDO(
   'mysql:host=127.0.0.1;dbname=
    crudoperation',
   'root',
   '',

array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES
utf8")
);
$inserts = array();
$queryFormat = '("%s", "%d")';
for ($i=1; $i<=1; $i++)
{
$name = $faker->firstName;
$rating = $faker->numberBetween($min = 1, $max = 10);
$inserts[] = sprintf( $queryFormat, $name, $rating );
}
$query = implode( ",", $inserts );
$db->query( 'INSERT INTO crud (name, rating) VALUES '.$query );
?>
```

Listing 2: Fragment kodu odpowiadający za generowanie danych w aplikacji webowej napisanej w języku Java

```
packagecom.testApplication.movielibrary;
import com.github.javafaker.Faker;

public class FakerDemo {
   public static void main(String[] args) {
Faker faker = new Faker();
for(int i=0; i < 10; i++){
String firstName = faker.name().firstName();
int numberBetween = faker.number().numberBetween(0,10);
System.out.println(firstName + " " + numberBetween);
   }}
}
```

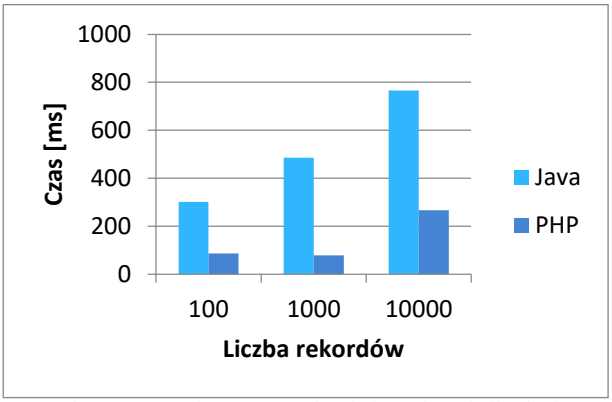
Na pojedynczych rekordach zostały wykonane operacje edycji oraz usuwania danych. W przypadku zapisywania oraz wyświetlania danych operacje były wykonywane na 1, 100, 1000 i 10000 rekordach. Wszystkie operacje zostały wykonane 5 razy w celu uzyskania miarodajnego wyniku. Podczas operacji mierzony był czas realizacji zapytania do bazy.

6. Platforma testowa i wyniki badań

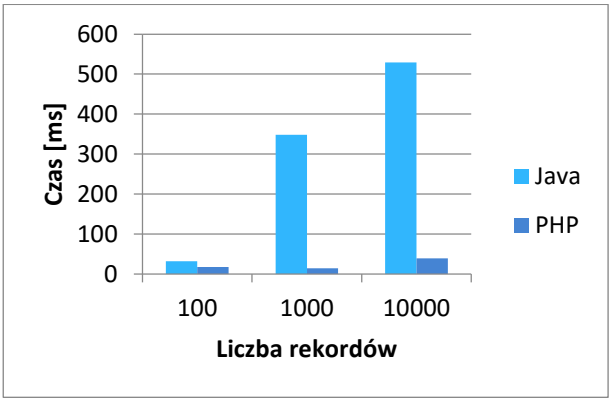
Badania zostały przeprowadzone na tym samym stanowisku przy wyłączonych wszelkich aplikacjach, które mogłyby mieć wpływ na wyniki. Dostęp do aplikacji w obu przypadkach odbywał się za pomocą usługi localhost na serwerze Apache. Do porównywania wydajności została użyta baza danych MySQL. Wykorzystywana była taka sama baza danych zatem zapytania SQL były w obu przypadkach identyczne. Ta Komputer, który został użyty do testów posiadał następujące parametry:

- system operacyjny: Windows 10 Home, 64 bitowy,
- procesor: AMD Ryzen 5 1600 3,2 GHz six core,
- pamięć Ram 16 GB,
- dysk HDD – 500 GB.
- karta graficzna: NVIDIA GeForce GTX 1050 Ti

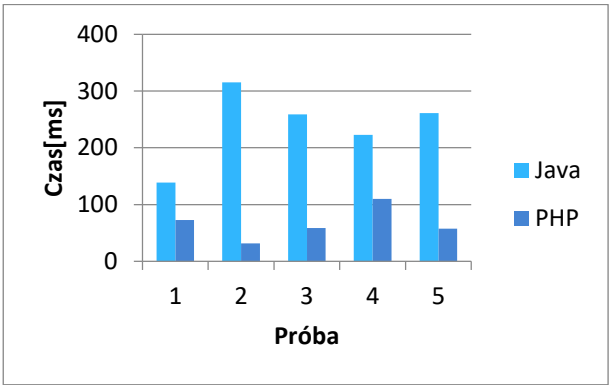
Rysunki 1- 4 prezentują średnie wyniki jakie otrzymały obie aplikacje w przypadku wykonywania poszczególnych operacji. Na wykresach nie został przedstawiony czas wykonywania zapisywania oraz wyświetlania na pojedynczym rekordzie ponieważ na tle innej ilości rekordów byłby on niewidoczny.



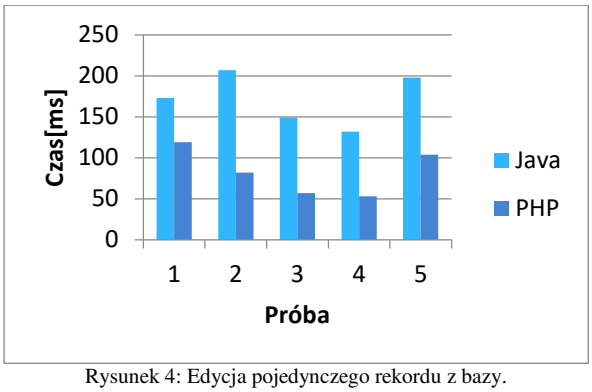
Rysunek 1: Porównanie czasów zapisu do bazy danych dla obu języków dla rożnej liczby zapisywanych rekordów.



Rysunek 2: Porównanie czasów wyświetlania danych z bazy dla obu języków i różnej liczby zapisywanych rekordów.



Rysunek 3: Usuwanie pojedynczego rekordu z bazy.



Rysunek 4: Edycja pojedynczego rekordu z bazy.

7. Analiza kodu

Analiza zaimplementowanych aplikacji testowych pozwoliła na ocenę, w której technologii, Java czy PHP, początkujący programista będzie mógł łatwiej napisać aplikację internetową o identycznych funkcjach. Podczas tworzenia aplikacji w języku Java stworzenie konkretnej funkcjonalności odbywało się w trzech klasach. Najpierw została stworzona klasa ze zmiennymi, odpowiadająca poszczególnym polom w bazie (Movie). Następnie została stworzona klasa, która odpowiadała ze komunikację z bazą danych (MovieRepository), a na końcu klasa, gdzie zaimplementowana metoda została wywoływana (MovieController) [8]. Podczas implementacji aplikacji w języku Java została wykorzystana biblioteka Lombok [9], która posłużyła do wygenerowania konstruktorów oraz ułatwiła implementację funkcji występujących w aplikacji. Zależności obiektów zostały wstrzyknięte przy pomocy biblioteki Spring. Adnotacje znajdujące się nad klasami (@Data, @NoArgsConstructor, @Autowired) pozwalają nam na korzystanie z nich bez dodatkowych instalacji. Dodatkowo została wykorzystana funkcja „query", która wspomaga wymianę danych z bazą danych. W przypadku aplikacji w języku PHP wszystko odbywało się w jednym miejscu i było bardziej przejrzyste, co sprawia że początkujący programista łatwiej może się odnaleźć w kodzie. Podczas implementacji została wykorzystana funkcja odpowiadająca za połączenie z bazą danych (mysqli) oraz funkcje odpowiadające za kontakt aplikacji z bazą danych (mysqli_query, mysqli_fetch_assoc). Dla obu aplikacji do wygenerowania danych na których były testowane funkcje posłużyła biblioteka Faker. Jeśli chodzi o ilość dostępnych materiałów i dokumentację języka, Java ma pod tym względem bardzo dużą przewagę nad językiem PHP. Pamięć RAM, jakiej wymagała poszczególna aplikacja także została zmierzona narzędziem Postman.

8. Analiza wyników

Na podstawie przedstawionych wyników można zauważyć przewagę aplikacji napisanej w języku PHP. Podczas wykonywania zapisu rekordów do bazy Java wykonuje się prawie czterokrotnie wolniej niż w przypadku zapisywania danych w aplikacji napisanej w języku PHP. W aplikacji napisanej w technologii PHP różnice czasowe pomiędzy wyświetlaniem ilości

rekordów są praktycznie niewidoczne. Ilość wyświetlanych danych nie wpływała na szybkość wykonywania zapytania. Inaczej jednak było w przypadku aplikacji napisanej w języku Java. Większa ilość rekordów sprawiała, że czas wykonywania zapytania do bazy znacząco się wydłużał. W przypadku 10000 rekordów było to średnio prawie 529,2ms, natomiast dla aplikacji napisanej w języku PHP wynosiło średnio 39ms. Wykonywane operacje na pojedynczych rekordach takie jak usuwanie czy edytowanie danych również wskazują na wyższą wydajność języka PHP. Różnice czasowe nie były jednak na tyle duże aby użytkownik podczas korzystania z obu aplikacji potrafił wskazać, która aplikacja działa szybciej. Duży wpływ na otrzymane wyniki mogło mieć środowisko na którym zostały przeprowadzone badania z powodu użycia dysku HDD, który jest wolniejszy od dysku SDD. Język PHP jest językiem skryptowym, nie wymaga kompilacji. Inaczej było w przypadku aplikacji napisanej w języku Java gdzie kompilacja jest wymagana. Czas kompilacji nie został uwzględniony podczas wykonywania pomiarów, różnice pomiędzy technologiami byłyby wtedy jeszcze większe. Poniżej znajduje się Tabela 1 przedstawiająca statyski dotyczące implementacji poszczególnych aplikacji.

Tabela 1:Statystyki aplikacji

Statystyki	Aplikacja w języku PHP	Aplikacja w języku Java
Liczba klas użytych do implementacji	6	5
Liczba bibliotek	1	4
Liczba dedykowanych funkcji	5	7
Liczba linii kodu	101	109
Zużycie pamięci RAM podczas wyświetlania pojedynczego rekordu	110 KB	204 KB
Zużycie pamięci RAM podczas dodawania pojedynczego rekordu	165 KB	270 KB
Rozmiar aplikacji	52,61 KB	110 KB

9. Wnioski

Wyniki jednoznacznie wskazują, która technologia jest wydajniejsza w przypadku aplikacji posiadającej takie same funkcjonalności. Trzeba jednak zwrócić uwagę, że język PHP służy wyłącznie do implementacji aplikacji internetowych a Java jest językiem obiektowym ogólnego zastosowania. Zatem pomimo tego iż język Java wymaga większych zasobów sprzętowych aby uzyskać podobne czasy co aplikacji napisana w języku PHP to ma ogólniejsze zastosowanie. Napisane aplikacje posiadają podstawowe funkcjonalności aplikacji webowej, nie zawierają warstwy odpowiadającej za widok. Największe dysproporcje czasowe pojawiają się podczas wyświetlania 10000 rekordów z bazy gdzie średnio język PHP osiąga dwunastokrotnie większą prędkość podczas wykonywania zapytania. Wyniki definitywnie potwierdzają tezy T1 i T2. Po przeanalizowaniu kodu pod względem implementacji, wysunięto wniosek iż implementacja w języku PHP jest prostsza dla początkującego programisty, głównie dlatego iż wszystko znajduje się w jednym miejscu i używane metody nie wymagają dodatkowych adnotacji nad klasami jak w przypadku aplikacji Java. Podczas implementacji aplikacji w języku Java trzeba było wykorzystać więcej bibliotek oraz funkcji, niż w przypadku aplikacji napisanej w języku PHP. Pomimo że dodawanie biblioteki odbywało się w bardzo prosty sposób, to jednak początkujący programista może o tym nie wiedzieć i wtedy implementacja staje się trudniejsza. Została zatem potwierdzona teza T3. Ilość linii kodu pomiędzy aplikacjami jest bardzo podobna. Jednakże większa ilość funkcji oraz bibliotek, które zostały wykorzystane w aplikacji Java w dużym stopniu zmniejszają objętość kodu. Jeśli chodzi o zużycie pamięci RAM podczas wykonywania operacji dodawania oraz wyświetlania pojedynczego rekordu więcej pamięci wymaga aplikacja napisana w języku Java.

Literatura

[1] K. Arnold, J. Gosling, D. Holmes, The Java Programming Language, Fourth Edition, Addison Wesley Professional 2005.

[2] S. Jędruszak, B Jędruszak B. Pańczyk, Tworzenie aplikacji internetowych na platformie JEE I PHP – analiza porównawcza, Journal of Computer Sciences Institute 11 (2019) 86-90 https://doi.org/10.35784/jcsi.145.

[3] A. Kołtun, B. Pańczyk, Analiza porównawcza narzędzi do badania wydajności aplikacji internetowych, Journal of Computer Sciences Institute 17 (2020) 351-357 https://doi.org/10.35784/jcsi.2209.

[4] J. Farrell, Java Programming Cengage Learning, 2011.

[5] Dokumentacja Php, https://www.php.net/manual/en, [21.04.2022].

[6] HTTP request methods, https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, [22.04.2022].

[7] Dokumentacja biblioteki Faker, https://javastart.pl/baza-wiedzy/frameworki/javafaker, [22.04.2022].

[8] Dokumentacja Java api: https://docs.oracle.com/javase/7/docs/api, [21.04.2022].

[9] Dokumentacja biblioteki Lombok, https://javastart.pl/baza-wiedzy/frameworki/project-lombok, [27.07.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 384–392
Received: 31 August 2022
Accepted: 20 September 2022

User experience analysis while visiting selected virtual museums

Analiza doświadczenia użytkownika podczas zwiedzania wybranych wirtualnych muzeów

Iwona Poleszak*, Mariusz Dzieńkowski

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

Abstract
The paper concerns the study of User Experience by focusing on the usability and user satisfaction aspects. The purpose of this paper was to evaluate the usability of two selected virtual museums conducted with 3 research methods: using an eye tracker, a System Usability Scale (SUS) survey and the Nielsen's heuristics. The examination was conducted on the following museums that offer virtual tours: the Museum of Musical Instruments in Poznan and the Zamosc Museum in Zamość. The participants of the eye tracking experiment and the SUS survey were 22 students of Computer Science at the Lublin University of Technology, while the Nielsen's heuristics analysis was performed by three graduate students with relevant qualifications. The obtained eye tracking data, the results of the SUS questionnaires and the evaluation of the Nielsen heuristics were analyzed quantitatively. A qualitative analysis of eye tracking results in the form of heat maps and scanpaths was also performed. As a result, it was revealed that in the eye tracking study, the analyzed websites obtained comparable results. However, in the test performed with the SUS usability survey the Museum of Musical Instruments achieved a better result than the Zamosc Museum. The expert team using the Nielsen's heuristics for evaluation also ranked the museum higher.

Keywords: user experience; usability; eye tracking; virtual museum

Streszczenie
Artykuł dotyczy badania doświadczenia użytkownika poprzez skupienie się na aspekcie użyteczności oraz satysfakcji odbiorcy. Celem pracy była ocena użyteczności dwóch wybranych muzeów wirtualnych dokonana trzema narzędziami - za pomocą okulografu, ankiety użyteczności SUS (ang. System Usability Scale) oraz heurystyk Nielsena. Materiałem badawczym były dwie witryny internetowe umożliwiające wirtualne zwiedzanie: Muzeum Instrumentów Muzycznych w Poznaniu oraz Muzeum Zamojskie w Zamościu. Uczestnikami badania okulograficznego oraz ankiety użyteczności SUS było 22 studentów kierunku Informatyka na Politechnice Lubelskiej, z kolei analizę z wykorzystaniem heurystyk Nielsena przeprowadziły 3 osoby, mające stosowne do tego celu kwalifikacje. Dane okulograficzne, wyniki ankiet SUS oraz oceny poziomów realizacji heurystyk Nielsena zostały poddane analizie ilościowej. Przeprowadzono również analizę jakościową na wynikach badań eyetrackingowych w postaci map cieplnych i ścieżek skanowania. W efekcie przeprowadzonych badań okazało się, że w badaniu eyetrackingowym analizowane witryny uzyskiwały porównywalne wyniki. Natomiast w teście użyteczności wykonanym za pomocą ankiety SUS Muzeum Instrumentów Muzycznych osiągnęło lepszy wynik niż Muzeum Zamojskie. Zespół ekspercki wykorzystujący heurystyki Nielsena również wyżej ocenił to muzeum.

Słowa kluczowe: doświadczenie użytkownika; użyteczność; okulografia; wirtualne muzeum

*Corresponding author

Email address: iwona.poleszak@pollub.edu.pl (I. Poleszak)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Z każdym kolejnym rokiem coraz więcej muzeów decyduje się na cyfryzację zgromadzonych zbiorów i udostępnienie ich w Internecie. Jest to nie tylko odpowiedź na ludzką potrzebę obcowania z dziedzictwem kulturowym, lecz także jego nowoczesna forma promocji. Oglądanie zbiorów muzealnych z całego świata przez Internet staje się coraz popularniejszą alternatywą dla podróży. Cyfrowa transformacja pozwala na udostępnienie zbiorów szerszemu gronu osób, które mogą bez presji czasu kontemplować sztukę w domowym zaciszu. Tego rodzaju wizyty pozwalają na zaznajomienie się ze zbiorem, a to z kolei może zachęcić do osobistej wizyty.

Na początku pandemii COVID-19, gdy dostęp do dzieł zgromadzonych w muzeach był niemożliwy, wirtualna rzeczywistość stała się niezwykle istotna. Mimo że, internetowe wersje muzeów nie są nowością, to jednak podczas pandemii przeżywały swój renesans. Dzięki wirtualnej rzeczywistości nie trzeba jeździć do Londynu, aby zobaczyć imponujące dzieła zgromadzone w Natural History Museum, czy do Amsterdamu, aby kontemplować dzieła Van Gogha zgromadzone w Rijksmuseum. Muzea wirtualne dzięki swojej interaktywności przez dłuższy czas mogą skupić na sobie uwagę zainteresowanych sztuką i historią ludzi.

Jednak popularność danego muzeum nie jest tylko wynikiem posiadania i prezentacji atrakcyjnych zbiorów. Użytkownicy zwracają również uwagę na inne aspekty takie jak oferowane funkcjonalności witryny

zawierającej interaktywną wystawę, łatwość jej użytkowania i szeroką dostępność. Odbiorcy treści powinni być w stanie intuicyjnie nawigować po stronie muzeum, a także płynnie, bez większych opóźnień, przemierzać kolejne jego miejsca. Niefunkcjonalny, trudny w obsłudze, nieprzemyślany interfejs wirtualnego muzeum może zniechęcić do jego odwiedzania.

W związku z tym ważne jest testowanie użyteczności i dostępności systemów komputerowych, żeby sprawdzić, czy są one przyjazne dla użytkownika, łatwe w obsłudze i dostępne dla osób niepełnosprawnych. Istnieje wiele metod przeprowadzenia takiej analizy, jednak najczęściej używane są badania eksperckie oraz badania z udziałem użytkowników. Obie metody zostały wykorzystane w niniejszej pracy, aby dokładnie i wieloaspektowo przetestować wybrane wirtualne muzea.

Analiza satysfakcji za pomocą ankiety SUS jest szybką, prostą, tanią i uniwersalną metodą badania doświadczenia użytkownika wchodzącego w interakcję z danym systemem komputerowym [1]. Osobie badanej zostaje przedstawionych 10 aspektów, za pomocą których formułuje ona oceny, wykorzystując do tego pięciostopniową skalę opartą na skali Likerta. Wynik testu przedstawia się punktowo w zakresie 0 - 100. Interfejsy, które uzyskały co najmniej 68 punktów, określa się jako użyteczne.

Jednym z najczęściej stosowanych sposobów testowania użyteczności są heurystyki opracowane przez Jakoba Nielsena [2]. Są one zbiorem zadań weryfikujących różne aspekty interfejsów w zakresie użyteczności i optymalizacji, zgodnych z praktykami UX, które należy wykonać w ramach testowania systemu. Oprócz samych testów, istotny jest fakt, iż heurystyki Nielsena naprowadzają na istniejące problemy z dostępnością strony internetowej.

Innym sposobem badania użyteczności może być użycie eyetrackera – urządzenia do śledzenia stanów oczu, który daje zobiektywizowane wyniki w postaci liczbowej [3]. Jest to badanie, które pozwala nie tylko przeanalizować bieżące rozmieszczenie elementów strony internetowej, ale także zrozumieć, gdzie intuicyjnie powinna być umieszczona testowana funkcjonalność. Badanie dostarcza informacji, czy użytkownik zauważa elementy istotne z punktu widzenia celów działania strony.

W ramach tej pracy zostały przebadane interfejsy dwóch wirtualnych muzeów za pomocą metod przedstawionych powyżej. Celem tych badań było sprawdzenie oraz porównanie użyteczności ich interfejsów, a w szczególności narzędzi umożliwiających ich obsługę i nawigowanie, satysfakcji użytkowników oraz aspektów związanych z dostępnością.

2. Przegląd literatury

Steve Krug w swojej książce [4] skupił się na zrozumieniu doświadczenia użytkownika poprzez analizę funkcjonalności witryn internetowych. Podczas pisania książki kierował się ideą: „Nie każ mi myśleć". Przedstawione przemyślenia autora dotyczą rozwiązań, które wymagają jak najmniejszego zaangażowania użytkownika w przetworzenie treści wyświetlanych na stronie. Książka pt. *„Nie każ mi myśleć! O życiowym podejściu do funkcjonalności stron internetowych"* skupia się na ocenie interfejsu jak najbardziej przyjaznego dla użytkownika końcowego. Pozycja ta dotyczy planowania stron, tak aby użytkownicy nie byli sfrustrowani zbyt skomplikowaną nawigacją po serwisie. Autor skupia się również na testowaniu stron jako kolejnym koniecznym elemencie w celu osiągnięcia satysfakcji użytkownika.

W książce pt. *„Badania jako podstawa projektowania User Experience"* [5] zostały przedstawione metody badania doświadczenia użytkownika, ze wskazaniem ich pozytywnych stron oraz ograniczeń. Jej autorzy przekazali dobre praktyki, które warto wykorzystać podczas oceny doświadczenia użytkownika. Generalnie metody badania doświadczenia użytkownika można podzielić na trzy grupy: zadaniowe testy użyteczności, eyetracking oraz zdalne badania ewaluacyjne. Do pierwszej grupy można zaliczyć: testy z pomiarem wykonania i testy porównawcze, wspólne odkrywanie, metoda zaznajamiania, testy mobilne i w terenie oraz testy na makietach i na prototypie, które można realizować w trzech wariantach: w postaci testu 5 sekund, czarnoksiężnika z krainy Oz, RITE (szybkiego iteracyjnego testu i ewaluacji). Podczas stosowania eyetrackingu można zastosować protokół głośnego myślenia, patrzenie swobodne lub zadania. Natomiast zdalne badania ewaluacyjne mogą być moderowane lub niemoderowane.

W artykule *„Characterising online museum users: a study of the National Museums Liverpool museum website"* [6] został poruszony temat skutecznego przyciągania uwagi użytkowników. W ramach tej pracy zrealizowano badania, które dowiodły, że użytkownicy w krótkim czasie decydują o swoim pozostaniu na danej witrynie lub jej opuszczeniu. Artykuł skupia się na istotnych elementach doświadczenia użytkownika, które stanowią kluczowe elementy pomagające w podjęciu tej decyzji. Autorzy artykułu na podstawie strony internetowej National Museums Liverpool zbadali szeroki zakres charakterystyk użytkowników podczas ich wizyt na badanej stronie.

W książce *„Measuring the user experience: collecting, analyzing, and presenting usability metrics"* [7] autor także koncentruje się na metodach badawczych doświadczenia użytkownika. Prezentowane są metryki wykorzystywane do oceny User Experience oraz możliwości ich analizy. Dostarczone są szczegółowe wskazówki, w jaki sposób można mierzyć użyteczność dowolnego typu produktu za pomocą określonej metody.

W kolejnej pracy [8] autorzy proponują wykorzystanie nowoczesnych technologii komputerowych np. Big Data w przestrzeni muzealnej, dzięki której możliwe jest uchwycenie wszystkich różnic w zachowaniach odwiedzających. W artykule autorzy rozwinęli ideę wykorzystania techniki eyetrackingowej, aby lepiej zrozumieć, w jaki sposób kontemplowana jest sztuka i zaproponować nowe wrażenia odwiedzającym muzea. Jedną z propozycji jest stworzenie spersonalizowanych

doświadczeń dla turystów odwiedzających muzea. Użycie eyetrackingu w przestrzeni muzeum ma na celu zrozumienie w jaki sposób odwiedzający odczytują dzieła sztuki oraz wykorzystanie tej wiedzy do zmiany sposobu, w jaki te dzieła są wyjaśniane. Jak dotąd w kwestii spersonalizowania wrażeń nie podjęto szczególnych kroków. Wszystkie przewodniki zarówno w formie pisanej, jak i audio oraz aplikacje komputerowe są takie same, choć oczekiwania różnych grup odwiedzających znacznie się od siebie różnią.

W kolejnej pozycji pt. *„Eye tracking in tourism”* [9], autorzy prezentują przykłady proaktywnych działań, które można podjąć, bazując na wynikach przeprowadzonych badań. Należy wykorzystać fakt, że turyści znacznie bardziej koncentrują się na dziele, gdy jest przy nim opis w postaci tylko jednej linii. Istotną kwestią jest także odbiór treści muzeum przez różne grupy wiekowe. Nastolatkowie, czy też młodzi dorośli, o wiele szybciej przyswoją działanie wirtualnej rzeczywistości.

3. Cel i zakres pracy

Celem pracy jest ocena doświadczenia użytkownika, na którą składają się takie elementy jak użyteczność, dostępność i satysfakcja, dwóch wybranych muzeów wirtualnych. Badania zostały zrealizowane za pomocą eyetrackera stacjonarnego w laboratorium, wypełnienia ankiety użyteczności w skali SUS oraz przeprowadzenia audytu według 10 heurystyk Nielsena.

Zakres prac obejmował przegląd literatury, selekcję obiektów do badań – dwóch muzeów wirtualnych, dobór metod badawczych, opracowanie zadań dla użytkowników do badań eyetrackingowych, zaprojektowanie oraz przeprowadzenie eksperymentu, a także analizę wyników i sformułowanie wniosków.

4. Metoda badawcza

W zaprojektowanym eksperymencie wykorzystano trzy metody oceny użyteczności oraz jakości interfejsu. Podzielono go na 3 etapy:

- eyetracking - pierwsza część eksperymentu składająca się z ośmiu zadań wykonanych przez użytkowników,
- ankieta SUS - wypełniana bezpośrednio po zakończonym eksperymencie eyetrackingowym,
- ankieta według heurystyk Nielsena - oceniana przez zespół ekspercki.

Do porównania między sobą wirtualnych muzeów, zostały wykorzystane następujące miary:

- czas do pierwszej fiksacji w obszarze zainteresowania,
- liczba poprawnie ukończonych zadań,
- czas przebywania w obszarze zainteresowania,
- wskaźnik SUS,
- ocena heurystyk Nielsena.

4.1. Obiekty badań

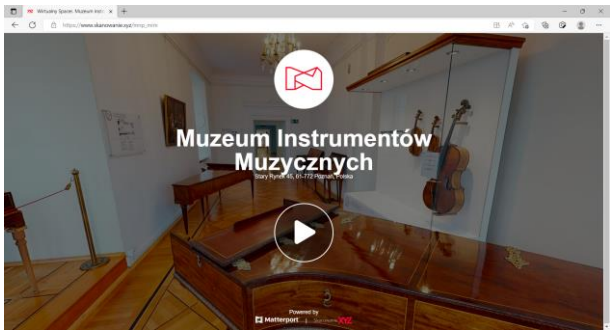
Obiekty, które zostały wybrane do badań, to polskie, dostępne online i atrakcyjne wirtualne muzea różniące się tematyką prezentowanych kolekcji (tabela 1). Pierwsze z nich to Muzeum Instrumentów Muzycznych (MIM), które jest częścią Muzeum Narodowego w Poznaniu (rysunek 1). Posiada ono szczególnie bogatą kolekcję instrumentów lutniczych, a eksponaty pochodzą z wielu kontynentów oraz różnych okresów. Zbiory zawierają ponad dwa tysiące obiektów i są przedmiotem badań dla naukowców nie tylko związanych z muzyką.

Drugie muzeum to Muzeum Zamojskie (MZ) zawierające głównie eksponaty etnograficzne oraz archeologiczne (rysunek 2). Znajduje się w nim bogaty dział numizmatyczny, a także gromadzone są księgozbiory.

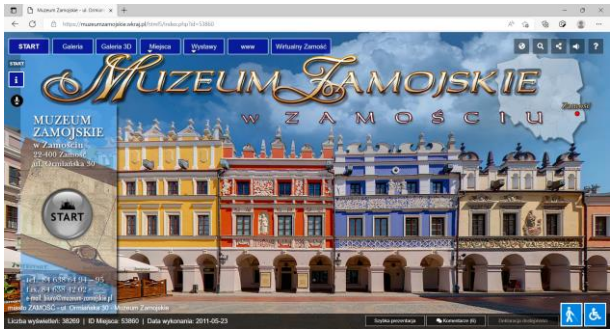
Tabela 1: Badane obiekty - wybrane wirtualne muzea

Lp.	Nazwa muzeum	Adres strony
1	Muzeum Instrumentów Muzycznych w Poznaniu	https://mnp.art.pl/oddzialy/muzeum-instrumentow-muzycznych/
2	Muzeum Zamojskie w Zamościu	https://muzeumzamojskie.wkraj.pl

Do badań wykorzystano przedstawione wcześniej muzea, ze względu na ich podobieństwo w zakresie sposobu prezentacji zbiorów, wielkość oferty, zapewnienie zbliżonego zestawu funkcji do ich obsługi i nawigacji. W badaniach zrealizowanych w ramach tej pracy, skoncentrowano się właśnie na ocenie użyteczności i dostępności witryn wyposażonych w narzędzia do obsługi i nawigowania po wirtualnym muzeum.



Rysunek 1: Witryna Muzeum Instrumentów Muzycznych w Poznaniu.



Rysunek 2: Serwis Muzeum Zamojskiego w Zamościu.

4.2. Grupa badawcza

W badaniach eyetrackingowych wzięły udział 22 osoby. Ci sami uczestnicy wypełnili ankietę SUS po zakończonych badaniach. W grupie badawczej ~72% stanowili mężczyźni - studenci ostatniego roku kierunku Informa-

tyka drugiego stopnia studiów na Politechnice Lubelskiej. W eksperymencie wzięły również udział trzy osoby z tytułem mgr inż. będące absolwentami kierunku informatyka. Żaden z uczestników eksperymentu nie miał do czynienia z badanymi witrynami muzeów, ale aż połowa osób brała wcześniej udział w podobnym badaniu eyetrackingowym.

4.3. Stanowisko badawcze

Badania zostały przeprowadzone przez moderatora w pokoju zlokalizowanym w laboratorium Katedry Informatyki Politechniki Lubelskiej (KI). W pomieszczeniu tym zapewnione są odpowiednie warunki oświetleniowe, a także wygodne stanowisko badawcze (rysunek 3).

Eksperyment został przygotowany i przeprowadzony na komputerze przenośnym ASUS G750JX-T4191H wyposażonym w procesor Intel Core i7-4700HQ i pamięć RAM 16GB. Laptop działa pod kontrolą systemu operacyjnego Windows 10.

Do badań został użyty eyetracker stacjonarny Tobii TX300 [10], którego główne parametry to częstotliwość próbkowania wynosząca 300Hz przy śledzeniu obuocznym oraz dokładność równa 0,4° dla obu oczu mierzona w idealnych warunkach. Śledzenie ruchu gałek oparte jest na technice ciemnej źrenicy. Eyetracker umożliwia detekcję stanów oka takich jak fiksacje, sakady, zmiany średnicy źrenicy oraz mrugnięcia. Zakres poruszania głową podczas badań to 35,6 cm w poziomie oraz 17,8 cm w pionie. Oczy uczestnika badań powinny być oddalone od eyetrackera na odległość 50-80 cm. Bodźce wizualne, czyli zrzuty ekranowe z wirtualnych muzeów, były wyświetlane na ekranie TFT (przekątna 23”, rozdzielczość 1920x1080) zintegrowanym z eyetrackerem z wbudowaną kamerą rejestrującą twarz badanej osoby.

Rysunek 3: Stanowisko badawcze w KI.

Oprogramowaniem, w którym zaprojektowano i przeprowadzono eksperyment oraz dokonano rejestracji nagrań było Tobii Studio 3.4.5 (rysunek 4). Po wykonaniu badań, w Tobii Studio można wygenerować mapy termiczne, ścieżki skanowania, a także wyznaczyć obszary zainteresowania. Program ten umożliwia także pobieranie danych w formie liczbowej w postaci pliku csv.

4.4. Eksperyment

Przeprowadzenie badania okulograficznego oraz ankiety w skali SUS odbyło się w laboratorium, natomiast analiza heurystyczna Nielsena została wykonana zdalnie.
Oto etapy eksperymentu:

1. Zdefiniowanie problemu badawczego i określenie celu badań.
2. Dobór metod badawczych.
3. Wyselekcjonowanie obiektów do badań - dwóch wirtualnych muzeów.
4. Opracowanie poleceń dla użytkowników i przygotowanie wizualnych bodźców.
5. Wykonanie projektu eksperymentu w Tobii Studio.
6. Zorganizowanie grupy badawczej.
7. Zapoznanie uczestników z celem badań oraz przebiegiem eksperymentu.
8. Przeprowadzenie badania okulograficznego.
9. Wypełnienie ankiety SUS.
10. Analizy heurystyczna wykonana przez ekspertów.
11. Obróbka danych oraz ich analiza.
12. Przedstawienie wniosków z badań – ocena wirtualnych muzeów.

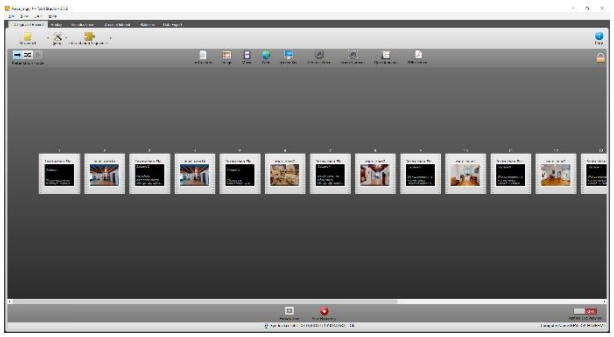
Tabela 2: Zadania dla uczestników badania eyetrackingowego

Kod zadania	Treść zadania
MZ1	W pomieszczeniu, w którym obecnie jesteś, znajduje się eksponat, który został opatrzony opisem. Znajdź element, na którym należy kliknąć, aby pokazał się ten opis.
MZ2	Chcesz się zorientować, w którym miejscu muzeum obecnie się znajdujesz. Musisz do tego celu wykorzystać mapę. Jakiego narzędzia w tym celu użyjesz?
MZ3	Wskaż ikonę, na którą należy kliknąć, aby uruchomić wyszukiwarkę.
MIM1	Wskaż element, na który należy umieścić kursor, aby ujrzeć opis instrumentu stojącego przed Tobą.
MIM2	Wskaż element, na który należy kliknąć, aby zmienić piętro zwiedzania.
MIM3	Wskaż element, na który należy kliknąć, aby wyświetlić pomoc.
MIM4	W którym miejscu należy kliknąć, aby przybliżyć się do kontrabasu?
MIM5	Gdzie należy kliknąć, aby dostać się do menu?

W eksperymencie eyetrackingowym wykorzystano osiem bodźców wizualnych - statycznych obrazów z dwóch wirtualnych muzeów:

- Muzeum Zamojskie – 3 zrzuty ekranu
- Muzeum Instrumentów Muzycznych – 5 zrzutów ekranu

Każdy bodziec był poprzedzony planszą zawierającą tekst polecenia do wykonania. Treść wszystkich zadań znajduje się w tabeli 2.



Rysunek 4: Zrzut ekranowy okna programu Tobii Studio przedstawiający fragment projektu eksperymentu.

Ankieta SUS (tabela 3) zawiera standardowe stwierdzenia, które zostały nieznacznie zmodyfikowane na potrzeby eksperymentu. Uczestnicy oceniali wirtualne muzea posługując się skalą od „Zdecydowanie się nie zgadzam" do „Zdecydowanie się zgadzam", która w zależności od aspektu daje inną liczbę punktów. Wynikiem jest suma punktów pomnożona przez 2,5, która daje wynik w przedziale od 0 do 100.

Tabela 3: Aspekty ankiety SUS

Lp.	Aspekty
1	Gdybym miał korzystać z wirtualnego zwiedzania, to korzystałbym z tak przygotowanego muzeum.
2	Obsługa muzeum jest niepotrzebnie skomplikowana.
3	Wirtualne muzeum jest łatwe w użyciu.
4	Będę potrzebował(a) wsparcia technicznego, aby korzystać z wirtualnego muzeum.
5	Różne funkcje wirtualnego muzeum są łatwo dostępne.
6	W wirtualnym muzeum jest zbyt wiele niespójności.
7	Większość osób będzie w stanie opanować obsługę wirtualnego muzeum bardzo szybko.
8	Wirtualne muzeum jest kłopotliwe w użyciu.
9	Czuję się bardzo pewnie korzystając z wirtualnego zwiedzania.
10	Musiałem(am) opanować wiele umiejętności przed rozpoczęciem pracy z wirtualnym muzeum.

Ostatnim etapem eksperymentu była analiza heurystyczna, w której eksperci - absolwenci kierunku Informatyka, mieli za zadanie przeanalizować interfejsy obu muzeów pod kątem 10 heurystyk Nielsena wymienionych w tabeli 4. Przy udzielaniu odpowiedzi posługiwali się skalą, zgodnie z którą 1 - oznacza „całkowicie się nie zgadzam", 5 - oznacza „całkowicie się zgadzam", a 3 to odpowiedź „nie wiem".

Tabela 4: Treść dziesięciu heurystyk Jakoba Nielsena

Lp.	Treść heurystyk
1	Pokazuj status systemu.
2	Zachowaj zgodność pomiędzy systemem a rzeczywistością.
3	Daj użytkownikowi pełną kontrolę.
4	Trzymaj się standardów i zachowaj spójność.
5	Zapobiegaj błędom.
6	Pozwalaj wybierać, zamiast zmuszać do pamiętania.
7	Zapewnij elastyczność i efektywność.
8	Dbaj o estetykę i umiar.
9	Zapewnij skuteczną obsługę błędów.
10	Zadbaj o pomoc i dokumentację.

5. Wyniki badań

5.1. Analiza jakościowa danych eyetrackingowych na podstawie map cieplnych

Mapy cieplne prezentują obszary witryny internetowej, na których użytkownicy skupiali swój wzrok podczas badania. Jest to sposób graficznej prezentacji danych dotyczących zachowania użytkowników na wyświetlanym wizualnym bodźcu. Rysunki 5 - 8 przedstawiają wybrane mapy cieplne dla bodźców wyświetlanych uczestnikom badań.

Rysunek 5 przedstawia mapę cieplną, na której gorący obszar wskazuje przezroczysty element z rysunkiem oka, który został uznany przez respondentów jako ten, który po wskazaniu kursorem myszy spowoduje wyświetlenie opisu eksponatu. Innym gorącym miejscem jest obszar, zawierający element graficzny ze strzałką, znajdujący się w centralnym miejscu wyświetlanego obrazu. Przez wielu uczestników badań był on interpretowany jako ten, który umożliwi dostęp do informacji o obiekcie muzealnym.

Rysunek 5: Mapa cieplna obrazująca wirtualną ekspozycję MZ - sala z prezentacją ilustrującą życie codzienne mieszczaństwa od XVI do XVIII w.

Rysunek 6: Mapa cieplna obrazująca wirtualną ekspozycję MZ - kuchnia mieszczańska z XVIII w.

Na mapie cieplnej na rysunku 6 zaznaczono miejsce w prawym górnym rogu, w którym znajduje się ikona z rysunkiem globu. Oprócz tego miejsca, widoczne są jeszcze trzy inne gorące obszary: przycisk „Miejsca",

ikona z rysunkiem oka umożliwiająca wyświetlenie opisu eksponatu oraz ikona do włączenia trybu pełnoekranowego (usytuowana przy prawej krawędzi okna). Obszarowo równie dużą gorącą powierzchnię ma element do wyświetlenia opisu obiektu.

Rysunek 7: Mapa cieplna obrazująca wirtualną ekspozycję MIM.

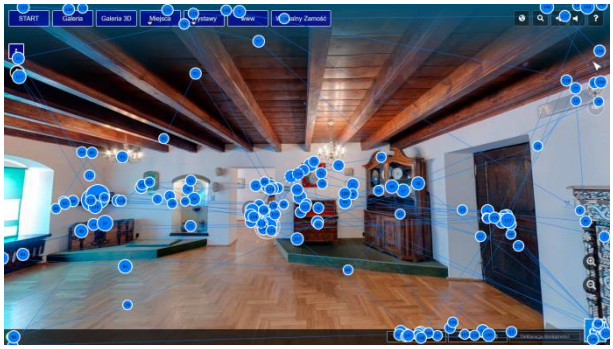
Rysunek 7 przedstawia mapę, na podstawie której można stwierdzić, że uczestnicy badań prawidłowo znajdowali narzędzie zmiany piętra wśród kilku ikon znajdujących się w lewym dolnym rogu wyświetlanego bodźca. Jednak bliskie położenie czerwonych obszarów na poszczególnych ikonach wśród wielu znajdujących się tam narzędzi może świadczyć o problemach interpretacyjnych użytkowników. Uczestnicy mieli problem z przypisaniem konkretnej ikony do pełnionej przez nią funkcji. Ostatni gorący obszar, patrząc od lewej strony, wskazuje właściwe narzędzie - właściwą ikonę.

Rysunek 8: Mapa cieplna obrazująca wirtualną ekspozycję MIM.

Z mapy cieplnej na rysunku 8 wynika, że z zadaniem tym uczestnicy mieli duży problem. Dwa obszary gorące położone w innym miejscu niż to właściwe, dowodzą, że badani uważali, że do przybliżenia się do gabloty, w której znajduje się kontrabas, służą inne narzędzia. Docelowe miejsce, oznaczone kołem i znajdujące się na podłodze przed gablotą jest pokryte tylko obszarem ciepłym (żółtym i zielonym), a nie gorącym.

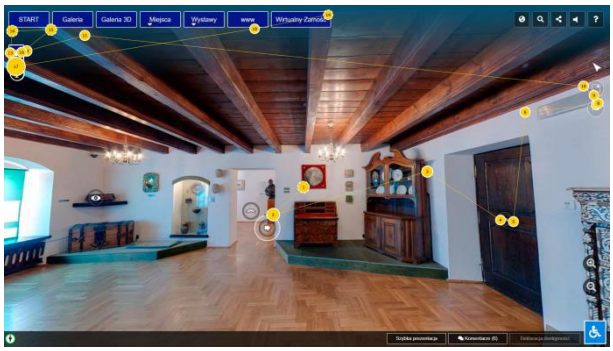
5.2. Analiza jakościowa danych eyetrackingowych na podstawie ścieżek skanowania

Ścieżki skanowania zawierają wizualny bodziec (w tym przypadku wybraną ekspozycję z danego muzeum) z nałożonymi na niego kołami połączonymi liniami, interpretowanymi odpowiednio jako fiksacje i sakady. Trasy jakie przemierza wzrok znacznie się różnią między uczestnikami badań. Rysunki 9 - 11 przedstawiają wybrane ścieżki skanowania, pokazujące zarówno pomyślne, jak i błędne wykonanie zadań.



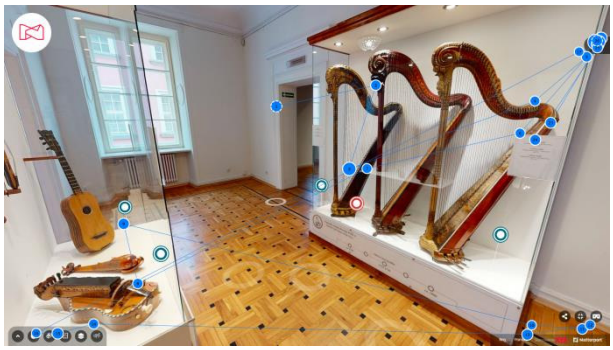
Rysunek 9: Ścieżka skanowania na ekspozycji MZ uczestnika o identyfikatorze P06, który poprawnie wykonał zadanie.

Przebieg ścieżki skanowania na rysunku 9 pokazuje szczegółowe przeglądanie centralnej części bodźca przedstawiającego pomieszczenie muzeum oraz narzędzia rozlokowane wzdłuż górnej, prawej i dolnej krawędzi okna. O zakończeniu zadania sukcesem, tzn. o odnalezieniu właściwego narzędzia do wyświetlenia informacji o eksponacie, świadczy duże nagromadzenie fiksacji w tym miejscu oraz duży rozmiar koła jednej z dłuższych fiksacji oznaczonej numerem 68.



Rysunek 10: Ścieżka skanowania na ekspozycji MZ uczestnika o identyfikatorze P25, który nie wykonał zadania poprawnie.

Przykład z rysunku 10 ukazuje ścieżkę skanowania osoby, która wskazała ikonę przedstawiającą literę ‘i’, zamiast ikony z rysunkiem oka umieszczonej na eksponacie. Prawdopodobnie jest to wynik niezrozumienia polecenia przez uczestnika badania.



Rysunek 11: Ścieżka skanowania na ekspozycji MIM uczestnika o identyfikatorze P20, który poprawnie wykonał zadanie.

Na kolejnym przykładzie na rysunku 11, osoba badana poprawnie zidentyfikowała ikonę, służącą do rozwinięcia opcji w menu. Śledząc przemieszczanie się punktu patrzenia, można zaobserwować, że respondent przyglądał się eksponatom w muzeum, a także skanował inne narzędzia powiązane z funkcjonalnościami wirtualnego muzeum znajdujące się na dolnej krawędzi okna.

Rysunek 12: Ścieżka skanowania na ekspozycji MIM uczestnika o identyfikatorze P30, który nie wykonał zadania.

W ostatnim przykładzie (rysunek 12) uczestnik badań w ikonie zawierającej rysunek strzałki skierowanej w górę, widział narzędzie, za pomocą którego można dotrzeć do opcji menu. Ikona ta znajdowała się w lewym dolnym rogu okna przeglądarki i nie została zauważona.

5.3. Analiza ilościowa danych eyetrackingowych na podstawie obszarów zainteresowania

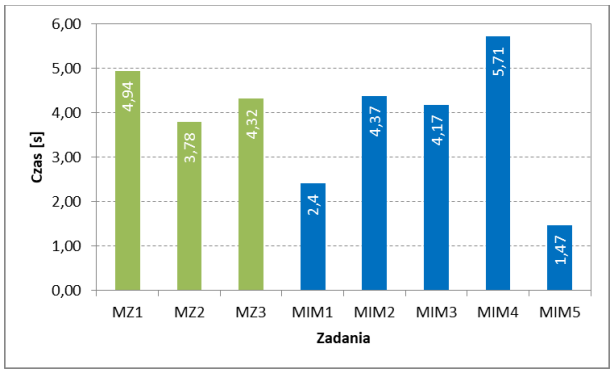
Analiza ilościowa została wykonana na podstawie obszarów zainteresowania nałożonych na szukane obiekty (rysunek 13). Obszary te miały zwykle większy rozmiar niż otaczane nimi, szukane elementy, ze względu na niedokładności występujące między rzeczywistym, a rejestrowanym punktem patrzenia.

Rysunek 13: Zrzut ekranowy fragmentu okna programu Tobii Studio przedstawiający bodziec z nałożonym obszarem zainteresowania.

Na rysunku 13 szukany przez respondentów element służący do wyświetlenia informacji o danym obiekcie znajduje się w obszarze zainteresowania przedstawionego w postaci fioletowego koła z umieszczoną wewnątrz etykietą „AOI_8".

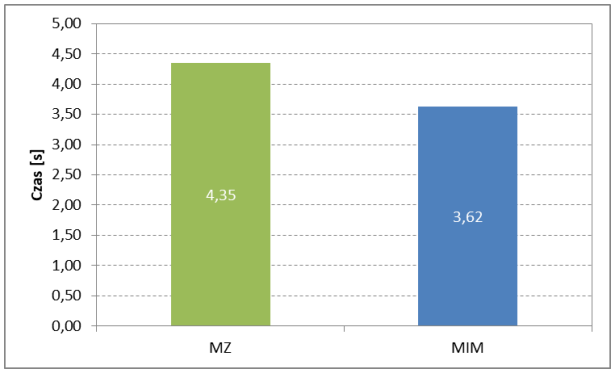
Po wyznaczeniu obszarów zainteresowania wybrano metryki, na bazie których przeprowadzono analizę statystyczną i utworzono wykresy. Do analizy ilościowej wykorzystano następujące miary eyetrackingowe:

- czas do pierwszej fiksacji w obszarze zainteresowania (TTFF - Time To First Fixation),
- liczba respondentów, których uwaga znajdowała się w obszarze zainteresowania (AOI - Area of Interest), tzn. którzy poprawnie wykonali zadanie,
- czas przebywania w obszarze zainteresowania.



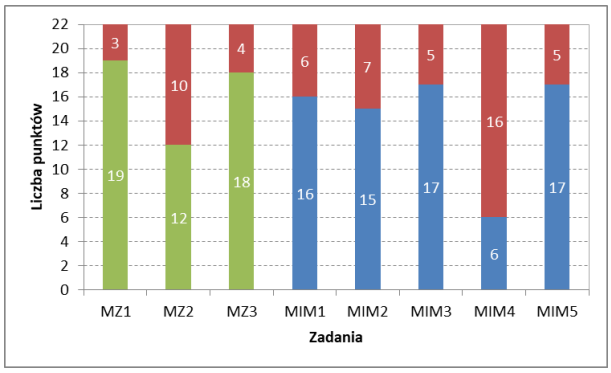
Rysunek 14: Średni czas do pierwszej fiksacji na szukanym elemencie (kolor zielony - Muzeum Zamojskie, kolor niebieski - Muzeum Instrumentów Muzycznych).

Z rysunku 14 wynika, że trzy zadania dla MIM (MIM2-MIM4) miały porównywalne średnie czasy do pierwszej fiksacji tak jak zadania dotyczące MZ. Dwa pozostałe polecenia MIM1 i MIM5, ze względu na swoją prostotę miały znacznie krótsze czasy realizacji. W związku z tym średni czas realizacji zadań dla MZ był o 0,73 sekund dłuższy niż dla MIM (rysunek 15).

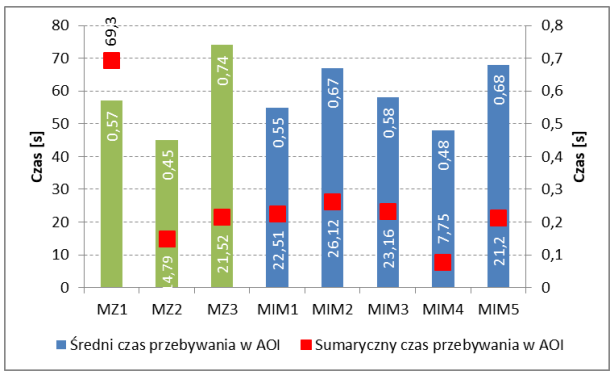


Rysunek 15: Uśredniony czas do pierwszej fiksacji dla wszystkich zadań dot. Muzeum Zamojskiego (kolor zielony) oraz wszystkich zadań dot. Muzeum Instrumentów Muzycznych (kolor niebieski).

Analizując liczbę poprawnie wykonanych zadań (rysunek 16), do których zakwalifikowane są te, w przypadku których wzrok znalazł się w obszarze otaczającym szukany element, lepsze średnie wyniki użyteczności osiągnęło Muzeum Zamojskie (16,33 pkt.) niż Muzeum Instrumentów Muzycznych (14,20 pkt.). Jednak należy zwrócić uwagę na jedno zadanie (MIM4), z którym użytkownicy mieli szczególnie dużo problemów, ponieważ tylko 6-ściu z nich poprawnie je wykonało.



Rysunek 16: Wyniki poprawności wykonania zadań przez użytkowników (kolor czerwony – liczba respondentów, którzy nie wykonali prawidłowo zadania, kolor zielony i niebieski – liczba respondentów, którzy prawidłowo wykonali zadanie odpowiednio dla MZ i MIM).

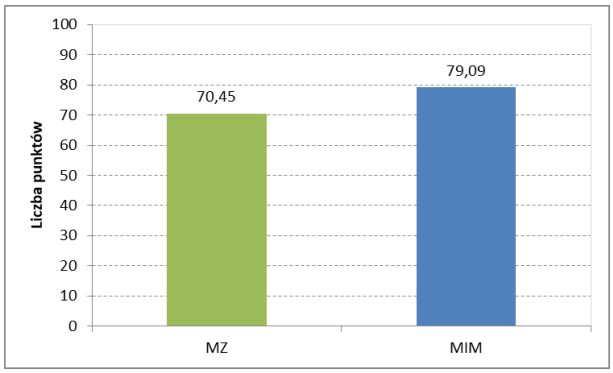


Rysunek 17: Średni i sumaryczny czas przebywania w danym obszarze zainteresowania.

Ostatnią miarą eyetrackingową podlegającą analizie był czas przebywania w obszarze zainteresowania. Wykres z rysunku 17 pokazuje średnie i sumaryczne czasy przebywania w AOI. Długi czas przebywania w przypadku zadania MZ3 (lokalizacji wyszukiwarki) wynikał z tego, że uczestnicy zastanawiali się czy ikona przedstawiona w postaci lupy da dostęp do wyszukiwarki, ponieważ w witrynie znajdowały się inne podobne narzędzia. W przypadku MIM długie czasy przebywania wystąpiły podczas szukania symbolu, za pomocą którego można wyświetlić opis instrumentu oraz rozwinąć menu. Krótkie czasy przebywania dla MZ2 oraz MIM4 były związane ze znalezieniem elementu, dzięki któremu wyświetlona zostanie mapa oraz elementu, za pomocą którego będzie można przybliżyć się do gabloty z instrumentem. Te krótkie czasy przebywania w przypadku tych zadań wynikają stąd, że mało osób poprawnie zidentyfikowało szukane elementy.

5.4. Wyniki ankiety użyteczności SUS

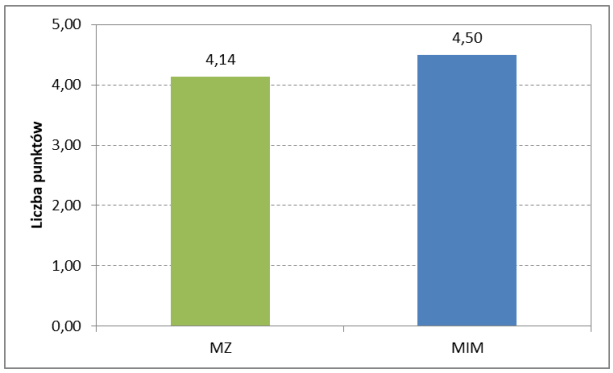
Na rysunku 18 przedstawiono poziom użyteczności analizowanych witryn za pomocą wskaźnika SUS. Oba wirtualne muzea osiągnęły wynik powyżej 68 punktów, co jest interpretowane jako wynik dobry. Muzeum Instrumentów Muzycznych zostało ocenione wyżej (79,09 pkt.) niż Muzeum Zamojskie (70,45 pkt.).



Rysunek 18: Poziom użyteczności wirtualnych muzeów.

5.5. Wyniki badań heurystyk Nielsena – analiza ilościowa

Analiza heurystyczna Nielsena została wykonana przez trzech absolwentów Informatyki, a następnie ich oceny uśredniono.



Rysunek 19: Średnia ocena interfejsów na podstawie heurystyk Nielsena.

Wyniki wykazały, iż lepiej ocenione zostało Muzeum Instrumentów Muzycznych (4,50 pkt.) niż Muzeum Zamojskie (4,14 pkt.), co przedstawia rysunek 19. Także poszczególne średnie wyniki każdej heurystyki dla MIM były wyższe lub równe niż wyniki MZ (tabela 5).

Tabela 5: Średnia punktacja dla każdej heurystyki

Lp.	Muzeum Instrumentów Muzycznych	Muzeum Zamojskie
1	4,67	4,33
2	5,00	4,67
3	3,67	3,67
4	4,33	3,67
5	5,00	4,67
6	5,00	5,00
7	4,67	4,00
8	4,67	3,67
9	3,00	3,00
10	5,00	4,67

6. Podsumowanie

Pandemia COVID-19 spowodowała, że instytucje kultury zostały pozamykane lub ograniczyły swą dostępność dla zwiedzających. Z dnia na dzień ludzie zostali pozbawieni możliwości bezpośredniego tradycyjnego obcowania z kulturą. W związku z tym dobrym pomy-

słem było przeniesienie przez różne instytucje części swej działalności do wirtualnego świata. Takie działania podjęły między innymi muzea, które przez tworzenie lub unowocześnianie istniejących witryn internetowych otworzyły się dla szerszego niż wcześniej kręgu odbiorców kultury. W celu spełniania swojej roli, a więc otwartości i przyciąganiu zwiedzających, strony internetowe muszą być łatwe w użyciu, intuicyjne i atrakcyjne. W ramach tej pracy zostały wybrane dwa muzea, które przetestowano pod kątem doświadczenia użytkownika za pomocą trzech metod.

W efekcie przeprowadzonych badań okazało się, że w badaniu eyetrackingowym porównywane witryny uzyskiwały porównywalne wyniki. Średni czas do pierwszej fiksacji był krótszy w przypadku Muzeum Instrumentów Muzycznych, natomiast liczba poprawnych realizacji zadań była wyższa dla Muzeum Zamojskiego. Można więc wyciągnąć wniosek, że Muzeum Instrumentów Muzycznych było przeszukiwane szybciej, lecz nie zawsze wirtualne zwiedzanie i przeszukiwanie kończyło się sukcesem. Z kolei w przypadku Muzeum Zamojskiego częściej zostawał znaleziony szukany element, ale zajmowało to więcej czasu.

W teście użyteczności wykonanym za pomocą ankiety SUS Muzeum Instrumentów Muzycznych osiągnęło lepszy wynik. Również wykorzystując heurystyki Nielsena witryna ta była oceniana wyżej przez panel ekspercki. Podsumowując, oba muzea osiągnęły wysokie wyniki, jednak w ocenach użytkowników i ekspertów nieco lepiej wypadło Muzeum Instrumentów Muzycznych.

Literatura

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

[2] J. Nielsen, 10 Usability Heuristics for User Interface Design, https://www.nngroup.com/articles/ten-usability-heuristics/, [13.09.2022].

[3] K. Kałan, D. Karpiuk, M. Dzieńkowski, Usability analysis taking into consideration the aspects of accessibility of selected university websites, Journal of Computer Sciences Institute, 21 (2021) 295-302, https://doi.org/10.35784/jcsi.2725, [01.09.2022].

[4] S Krug, Nie każ mi myśleć! O życiowym podejściu do funkcjonalności stron internetowych. Wydanie III, Helion, Gliwice, 2014.

[5] I. Mościkowska, B. Rogoś-Turek, Badania jako podstawa projektowania user experience, PWN, Warszawa, 2020.

[6] D. Walsh, M. M. Hall, P. Clough, J. Foster, Characterising online museum users: a study of the National Museums Liverpool Museum website, International Journal on Digital Libraries, 21(2) (2020) 75-87, https://doi.org/10.1007/s00799-018-0248-8, [15.08.2022].

[7] T. Tullis, Measuring the user experience: collecting, analyzing, and presenting usability metrics, Morgan Kaufmann Publishers, USA, 2008.

[8] L. Fauvelle, Reinventing the museum experience with eye-tracking, https://www.intotheminds.com/blog/en/reinventing-museum-experience-eye-tracking/ ,[01.08.2022].

[9] M. Rainoldi, M. Jooss, Eye tracking in tourism, Springer, 2020.

[10] Tobii Pro TX300, https://www.tobiipro.com/product-listing/tobii-pro-tx300/, [15.08.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 393–400
Received: 5 September 2022
Accepted: 29 September 2022

Analysis of selected features of application based on monolithic and microservice architecture

Analiza wybranych cech aplikacji opartych na architekturze monolitycznej i mikrousługowej

Kamil Jaskot*, Sławomir Przyłucki

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

Abstract
The article describes the performance of applications built in monolithic and microservice architectures. The base of research includes application supporting prescription management developed with the use of Spring Framework technology and implemented in the Docker Swarm test environment. The tested applications were subjected to various loads in the form of sending HTTP requests that simulated user behaviour. The research has proven that an application created based on microservices architecture offers better traffic handling in case of high load. Scaling a microservice application allows for greater gains in performance measured as quantity served client requests per unit of time than scaling a monolithic application under the same conditions scaling.

Keywords: microservices; monolith-software architecture; scaling services; spring framework

Streszczenie
Artykuł przedstawia porównanie wydajności aplikacji utworzonych w architekturze monolitycznej i mikrousługowej. Zakres badań obejmuje aplikacje wspomagające zarządzanie receptami, utworzone przy wykorzystaniu technologii Spring Framework i wdrożone w środowisku testowym Docker Swarm. Aplikacje poddano różnym obciążeniom w postaci wysyłania zapytań HTTP, które symulowały zachowanie użytkowników. Przeprowadzone badania dowiodły, że aplikacja utworzona w oparciu o architekturę mikrousług lepiej radzi sobie z obsługą ruchu w przypadku dużego obciążenia. Skalowanie aplikacji mikrousługowej pozwala na uzyskanie większego przyrostu wydajności mierzonej jako liczba obsłużonych żądań klientów w jednostce czasu niż skalowanie aplikacji monolitycznej przy tych samych warunkach skalowania.

Słowa kluczowe: mikrousługi; architektura monolityczna oprogramowania; skalowanie usług; spring framework

*Corresponding author
Email address: kamil.jaskot@pollub.edu.pl (K. Jaskot)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Wiele istniejących już aplikacji często jest zaimplementowanych przy wykorzystaniu architektury monolitycznej. Stale rosnący kod aplikacji wprowadza większą złożoność i skomplikowanie co ostatecznie doprowadza do ujawnienia technicznych i organizacyjnych ograniczeń aplikacji. Często takie aplikacje stają się ciężkie w utrzymaniu i za duże aby mógł je w pełni zrozumieć jakikolwiek programista. W rezultacie naprawienie błędów i wdrażanie nowych funkcjonalności jest coraz bardziej trudne i czasochłonne. Rozwiązaniem pozwalającym rozwiązać te ograniczenia jest przebudowanie aplikacji do architektury wykorzystującej mikrousługi. W ogólnym ujęciu, każda aplikacja posiada dwie kategorie wymagań. Pierwsza obejmuje wymagania funkcjonalne określające co aplikacja ma robić i zazwyczaj architektura ma niewiele z nią wspólnego. Można zaimplementować zestaw przypadków w niemal dowolnej architekturze. Architektura ma znaczenie ponieważ umożliwia aplikacji spełnienie drugiej kategorii wymagań dotyczących jakości usług [1].

Przetwarzaniem w chmurze nazywamy model, który umożliwia dostęp na żądanie do zasobów obliczeniowych takich jak m.in. przestrzeń dyskowa, serwery, aplikacje czy usługi [2]. Coraz więcej firm decyduje się na wykorzystanie technologii chmurowych ze względu na ich skalowalność i elastyczność kosztów widząc w nich strategię biznesową, która pozwala na konkurencję z innymi firmami i realizację celów biznesowych [3, 4]. Wykorzystanie zalet oferowanych przez rozwiązania chmurowe i DevOps (ang. Development And Operations) wymaga aby podczas budowania aplikacja spełniała ich założenia. Niezależne wdrażanie wymaga posiadania systemu składającego się z modułów, które można niezależnie od siebie zaprojektować, zaimplementować, przetestować i wdrożyć. Dlatego w ostatnich latach architektura mikrousług stała się niezbędnym elementem rozwoju aplikacji wdrażanych w środowiskach chmurowych [5, 6].

Nie ma złotego środka dlatego bardzo ważnym jest aby odpowiednio dobrać architekturę do konkretnej aplikacji biorąc pod uwagę to co chcemy osiągnąć. Niekiedy dekompozycja aplikacji monolitycznej w mikrousługową może przynieść odwrotne skutki do zamierzonych. W tym artykule zostanie przedstawiona analiza porównawcza cech funkcjonalnych architektury monolitycznej i mikrousługowej na przykładzie aplikacji wykorzystującej technologie Spring Framework.

1.1. Przegląd literatury

W pracy [7] autorzy przeprowadzili badania zastosowania architektury monolitycznej i mikrousług do analizy systemu do zarządzania dystrybucją DBM (ang. Distribution Management System). System DBM optymalizuje przepływ mocy i zapobiega przeciążeniom. Przedstawiona w artykule aplikacja służy do analizy rozproszonej topologii modelu sieci elektroenergetycznej. Testy zostały przeprowadzone dla różnych przypadków testowych, a otrzymane wyniki zostały porównane pomiędzy aplikacją monolityczną i aplikacją mikrousług. Według otrzymanych wyników podział usługi na wiele mikrousług przyczynia się do przyspieszenia obliczeń i uzyskiwania wyników takich jak m.in. faza, moc czy hierarchia obwodów.

Aby dokonać oceny procesu migracji aplikacji o architekturze monolitycznej do mikroserwisowej oraz sprawdzić czy aplikacja wciąż spełnia określone wymagania należy wykonać testy wydajności. Z badań przeprowadzonych w pracy [8] wynika, że zaproponowany system wykorzystujący architekturę mikroserwisową zapewnia niemal zbliżoną wydajność do monolitu. Jednak może być skalowany nawet dziesiątki razy, a dzięki modułowości, możliwe jest zwiększenie liczby instancji każdego komponentu w celu uzyskania lepszych czasów odpowiedzi.

Przepustowość aplikacji mierzona jako liczba żądań na sekundę jest podstawowym atrybutem przy obliczaniu wydajności. W ramach badań [9] autorzy przeprowadzili testy obciążenia aplikacji w celu uzyskania danych związanych z wydajnością i dostępnością dla poszczególnych implementacji metod komunikacji pomiędzy usługami (gRPC, RabbitMQ, REST API). W celu pomiaru wydajności zrealizowano trzy przypadki testowe przy wykorzystaniu narzędzia JMeter [10]. Każdy z przypadków miał na celu zbadanie przepustowości każdej z metod IPC (ang. Interprocess Communication). Czas trwania, każdego testu wynosił 180 sekund i polegał na ciągłym wysyłaniu wcześniej ustalonych żądań do systemu i czekaniu na odpowiedź przez wirtualnych użytkowników.

Analiza oprogramowania stworzonego za pomocą architektury monolitycznej i mikrousługowej może zostać przeprowadzona z wykorzystaniem testów wydajności. Porównując ten sam system w dwóch rożnych architekturach uwagę należy skupić na liczbie wysyłanych zapytań, czasie obsłużenia zapytania, ilości obsłużonych zapytań w zależności od liczby wysyłanych zapytań na jedną usługę lub ilości wątków [11].

W artykule [12] zostały scharakteryzowane różne wzorce projektowe odnoszące się do architektury mikrousług oraz omówione zostały zasady, które pozwalają na ich poprawną klasyfikację. Przeprowadzono systematyczne badanie, aby zidentyfikować zgłoszone użycie mikrousług i na podstawie przypadków użycia wyodrębnić wspólne wzorce i zasady. W efekcie wyróżniono trzy wzorce orkiestracji i przechowywania danych, które wydają się być powszechnie stosowane podczas wdrażania systemów opartych na mikrousługach. Niektóre wzorce takie jak np. wzorzec hybrydowy (ang. Hybrid Pattern) lub wzorzec rejestracji i udostępniania usług (ang. Service Registry Pattern) były używane głównie w celu migracji istniejących aplikacji monolitycznych i SOA (ang. Service-Oriented Architecture).

Z kolei badania przeprowadzone przez autorów artykułu [13] polegały na systematycznym przeglądzie literatury, z którego wybrano 8 istotnych artykułów. W przypadku wzorców projektowych przegląd wyłonił 44 wzorce dla mikroserwisów, co pozwoliło zaproponować opartą na nich taksonomię (wzorce: Front-End, Back-End, DevOps, IOT, migracji i orkiestracji).

Na podstawie porównań narzędzi Locust, Gatling i JMeter można stwierdzić, że każde z nich wykonuje swoją pracę doskonale. Locust [14] to oparte na języku Python narzędzie do testowania obciążenia aplikacji w postaci wysyłanych żądań, uważane za proste w użyciu i używane do testowania rozproszonego. Zyskuje przewagę, jeżeli chodzi o wydajność. Gatling [15] to platforma o otwartym kodzie źródłowym oparta na Netty. Bazuje na języku Scala oraz narzędziach Akka. Zapewnia doskonałą obsługę protokołu HTTP (ang. Hypertext Transfer Protocol) oraz umożliwia obsługę innych protokołów. Jego przewaga nad pozostałymi narzędziami to dostarczane wizualizacje wyników. Apache JMeter to kolejne narzędzie dostępne jako aplikacja komputerowa z przyjaznym dla użytkownika graficznym interfejsem użytkownika. Wspiera wykorzystanie systemu dedykowanych wtyczek, a jego popularność opiera się m.in. na różnorodności obsługiwanych protokołów i łatwości użytkowania [16].

Docker wykorzystując Docker Compose, tworzy odpowiedni ekosystem dla aplikacji opartych na architekturze mikroserwisów. Docker Compose tworzy uporządkowane grupy kontenerów na podstawie stworzonego pliku konfiguracyjnego YAML (ang. YAML Ain't Markup Language). Wirtualizacja oparta na kontenerach oferuje doskonałe narzędzia do obniżenia kosztów wdrażania aplikacji i zapewnia pełny wgląd w różne problemy związane z mikrousługami, które mogą się pojawić na etapie wdrażania jak i działania [17].

1.2. Charakterystyka badanych architektur

Architektura monolityczna jest tradycyjnym podejściem do projektowania systemów, w którym wszystkie moduły lub usługi są zawarte w jednej aplikacji, a wszystkie funkcje systemu muszą być wdrożone razem. Najpopularniejszym i najczęściej spotykanym przykładem jest monolit jednoprocesowy. Charakteryzuje się tym, że cały kod systemu wdrażany jest jako pojedynczy proces. Tego typu systemy mogą być prostymi systemami rozproszonymi, a ich funkcjonowanie sprowadza się najczęściej do operacja odczytu i zapisu danych z bazy danych i prezentacji ich w aplikacjach webowych. Wraz ze wzrostem systemu rozrasta się również monolit co może doprowadzić do powstania monolitu modułowego. Monolit modułowy jest rozszerzeniem monolitu jednoprocesowego, w którym pojedynczy proces składa się z wielu modułów. Umożliwia to niezależną i równoległą pracę nad każdym z modułów jednak podczas

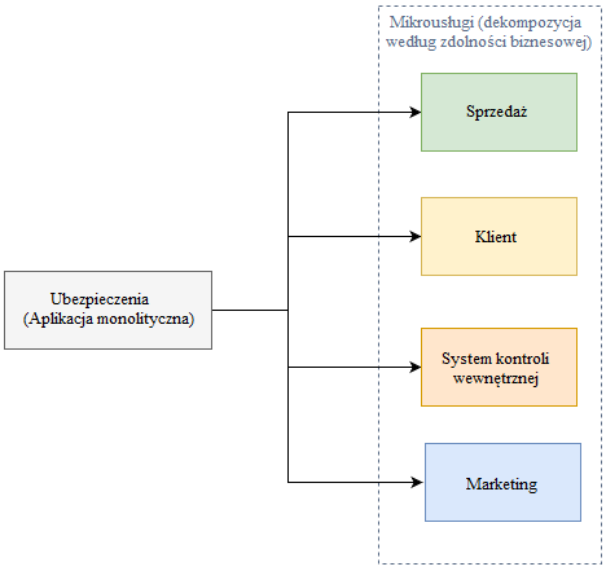
procesu wdrażania muszą one być ze sobą połączone. Jednym z największych problemów jakie wynikają z monolitu modułowego jest dekompozycja bazy danych dlatego czasami podejmowane są próby rozdzielenia bazy danych według poszczególnych modułów. Monolit rozproszony podobnie jak monolit modułowy jest podzielony, ale już nie na moduły lecz na usługi. Swoją architekturą bardzo często przypomina SOA, a nawet mikrousługi jednak w rzeczywistości usługi są silnie ze sobą powiązane przez co cały system musi być wdrożony razem.

Architektura mikrousługowa charakteryzuje się podziałem systemu na mniejsze, niezależne i luźno powiązane ze sobą usługi, które mogą być wdrażane niezależnie od siebie. Taka modułowość jest niezbędna przede wszystkim podczas tworzenia dużych i złożonych aplikacji. Każda usługa implementuje własny zestaw funkcji biznesowych i udostępnia na zewnątrz tylko niezbędne informacje wymagane przez inne usługi w postaci API (ang. Application Programming Interface). Pozostałe informacje takie jak m.in. sposób implementacji funkcjonalności, użyte technologie czy modele danych są ukrywane co pozwala na swobodne i niezależne zarządzanie usługą, wprowadzanie zmian czy dalszy rozwój. Pozwala to na wyróżnienie i oddzielenie fragmentów, które mogą być dowolnie zmieniane od tych, których zmiana jest trudniejsza. Zmiany wprowadzane wewnątrz jednej mikrousługi nie powinny wpływać na inne usługi, które się z nią komunikują [18].

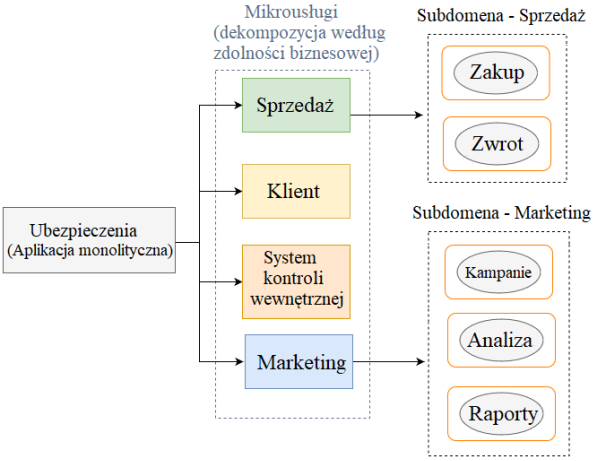
2. Metody dekompozycji aplikacji monolitycznych

W przypadku podjęcia decyzji o dekompozycji aplikacji monolitycznej do postaci struktury rozproszonej opartej o mikrousługi pojawiają się problemy związane z odpowiednim doborem wzorca dekompozycji i technik przejścia. Obecnie nie istnieje jedna metoda przebudowy struktury, która byłaby uniwersalna dla każdej aplikacji i każdego przypadku dokonując odpowiedniego podziału. Temat wyboru odpowiedniej metody według posiadanego modelu i celu jaki ma być osiągnięty jest przedmiotem wielu badań i propozycji.

Jednymi z popularnych zaproponowanych rozwiązań są dekompozycje według zdolności biznesowej i subdomeny. Dekompozycja według zdolności biznesowej przedstawiona na rysunku 1 polega na wyodrębnieniu usług odzwierciedlających jej działanie czyli to czym firma się zajmuje, aby pozyskiwać wartości np. sprzedaż, oferowanie usług czy kampania reklamowa. Natomiast dekompozycja według subdomeny, której przykład przedstawia rysunek 2 skupia się na budowaniu aplikacji wokół modelu domeny zorientowanego na obiekty. Rozwiązanie to wywodzi się z podejścia DDD (ang. Domain-Driven Design). Głównym założeniem jest utworzenie i orkiestracja usług wokół osobnych modeli domeny określających przestrzeń problemu aplikacji. Przykładowymi modelami domeny mogą być np. klient, zamówienie i sprzedaż [19]. Opóźnienia sieciowe są nieuniknioną częścią aplikacji utworzonej w architekturze mikrousługowej. Utworzone usługi mogą wymieniać między sobą bardzo dużą liczbę informacji. Aby temu zapobiec można pogrupować te usługi na podstawie transakcji. Pozwala to na uzyskanie szybszych czasów odpowiedzi i nie trzeba martwić się o spójność danych. Takie podejście nazywane jest dekompozycją według transakcji [20].



Rysunek 1: Dekompozycja aplikacji monolitycznej według zdolności biznesowej.



Rysunek 2: Dekompozycja aplikacji monolitycznej według subdomeny.

Głównym powodem podziału aplikacji monolitycznej do postaci mikrousług jest skalowanie. Możliwości skalowania aplikacji najczęściej definiuje się w postaci modelu skalowalności nazywanego sześcianem skalowania. Definiuje on trzy drogi skalowania według swoich osi: X, Y i Z. Skalowanie w osi Y polega na wykorzystaniu dekompozycji funkcjonalnej czyli podzieleniu monolitycznej aplikacja na zestaw usług według pewnych cech takich jak funkcjonalność. Otrzymane w ten sposób usługi mogą być skalowane niezależnie od siebie według osi X. Takie skalowanie polega na horyzontalnym duplikowaniu czyli powielaniu instancji. Skalowanie w osi Z czyli partycjonowanie danych polega na skalowaniu przez podział podobnych cech takich jak np.

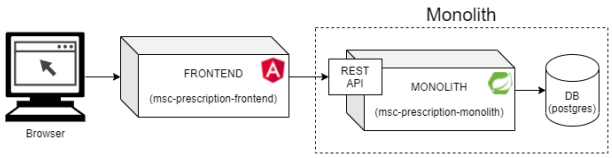
identyfikator zamówienia lub nazwisko klienta. W przeciwieństwie do skalowania w osi X, każda instancja usługi jest odpowiedzialna tylko za konkretny podzbiór cech np. obsługa klientów według przedziałów ustalonych na podstawie pierwszych liter nazwiska. Aby zdecydować do jakiej instancji trafi dane żądanie według przyjętego podziału wykorzystywany jest router pełniący rolę filtrów przychodzących żądań.

Często z dekompozycją aplikacji monolitycznej wiąże się konieczność podziału bazy danych. Idealnym rozwiązaniem jest utworzenie dla każdej z usług osobnej bazy danych o ile jest to wymagane i do której dostęp ma tylko i wyłącznie jedna usługa. Jednak istnieje jeszcze kilka sposobów zapewniających ukrywanie informacji o modelu danych i dostępu do nich. Można wykorzystać jedną bazę danych posiadającą zestaw tabel, do których dostęp mają wyłącznie konkretne usługi (do każdej z tabeli może odwoływać się tylko jedna usługa). Drugim podobnym rozwiązaniem jest utworzenie prywatnych schematów w bazie danych według usług. W pojedynczej bazie danych spójność danych zapewniały transakcje bazodanowe. Natomiast zachowanie spójności danych pomiędzy usługami obecnie stanowi spore wyzwanie i wymaga wykorzystania dodatkowych mechanizmów takich jak np. wzorzec saga [21].

3. Metody badawcze i plan eksperymentu

Do przeprowadzenia badań został wykorzystany zaprojektowany w tym celu system wspomagający zarządzanie receptami. Za stan początkowy przyjęto zapisane w systemie recepty wystawione przez lekarza. Głównym użytkownikiem systemu jest pacjent, który może zrealizować swoje recepty. W tym celu w pierwszej kolejności wybiera jedną z przypisanych do niego recept, a następnie wybiera leki razem z ich ilością do realizacji zamówienia. System zwraca listę aptek posortowaną malejąco według ilości posiadanych leków wskazanych przez pacjenta. W celu złożenia zamówienia użytkownik wybiera jedną z dostępnych aptek, a następnie uzupełnia dane niezbędne do jego realizacji np. adres zamówienia i po zatwierdzeniu danych w podsumowaniu zamówienie zostaje utworzone. Część dotyczącą płatności stanowi API dla zewnętrznych systemów i odpowiada za zmianę statusu zamówienia pozwalając na jego dalszą realizację. Kolejne funkcjonalności dotyczą pracownika apteki. Ma on wgląd we wszystkie zamówienia i możliwość ich realizacji, która polega na przygotowaniu wybranych przez pacjenta leków zmieniając ich status w systemie na gotowy. Następnie zamówienie jest przekazywane do wysyłki otrzymując status „gotowe do wysyłki” i kod dostawy. Proces związany z dostawą podobnie jak część związana z płatnością stanowi API dla zewnętrznych systemów dlatego obecnie skupia się jedynie na zmianach statusu za pomocą protokołu HTTP. W celu zachowania spójności systemu podczas implementowania go w dwóch różnych architekturach została utworzona aplikacja frontend w postaci strony internetowej wykorzystując takie technologie jak Angular i język TypeScript. Pozwoliło to na zachowanie spójności modelu pomiędzy aplikacjami.

W pierwszej kolejności system został zaimplementowany wykorzystując architekturę monolityczną według schematu przedstawionego na rysunku 3. Jest to monolit jednoprocesowy ponieważ cały kod systemu jest ze sobą ściśle powiązany i wymaga wdrożenia jako jeden proces. Cały system zawiera się w jednym serwisie posiadającym jedną bazę danych do którego odwołuje się aplikacja frontend. Do implementacji wykorzystano obiektowy język programowania Java w wersji 11 oraz Spring Framework, pozwalający na szeroki wybór możliwości podejścia do implementacji wszystkich zdefiniowanych funkcjonalności. Jako narzędzie wspomagające budowę i zarządzanie projektem wykorzystano Maven.

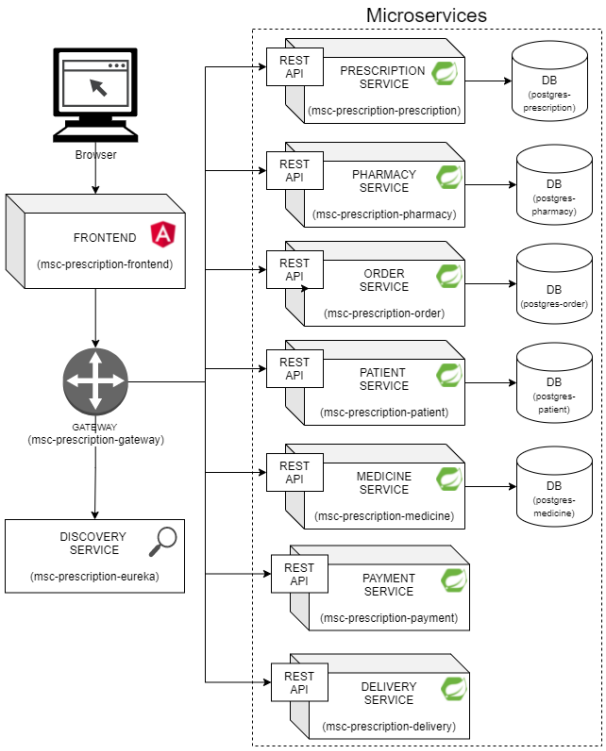


Rysunek 3: Architektura monolityczna zaimplementowanego systemu.

Do przebudowy aplikacji ze struktury monolitycznej do struktury opartej o mikrousługi została wybrana metoda podziału według funkcjonalności. Jest to idealne rozwiązanie pozwalające na skalowanie wyodrębnionych funkcjonalności np. ustawiając odpowiednią liczbę replikacji, a tym samym umożliwia łatwy dalszy rozwój systemu. Główną korzyści płynącą z podziału według funkcjonalności jest możliwość zmiany rozmiaru podstawowej infrastruktury, według wymagań niezbędnych do uruchomienia dla różnych obciążeń co zwiększa elastyczność w zakresie optymalizacji rozłożenia przychodzącego ruchu. Aby uzyskać pełne korzyści wybranej metody dekompozycji należy również dokonać podziału bazy danych tak aby każda usługa stanowiąca odrębną funkcjonalność ukrywała własny model danych i dostęp do niego jednocześnie udostępniając jedynie niezbędne operacje w postaci interfejsów.

Aplikacja mikrousługowa została utworzona przy wykorzystaniu takich samych technologii jak aplikacja monolityczna korzystając z narzędzi udostępnionych w ramach Spring Cloud będącego lekkim frameworkiem pozwalającym na implementację architektury mikroserwisowej. Oferuje on m.in. takie rozwiązania jak routing, rejestracje i odkrywanie usług, połączenia pomiędzy usługami, równoważenie ruchu, bezpiecznik czy rozproszone wiadomości [22]. Przyjęto podział według następujących funkcjonalności: prescription, pharmacy, order, patient, medicine, payment i delivery według założenia, że każda z usług powinna być wyodrębniona według zasady pojedynczej odpowiedzialności SRP (ang. Single Responsibility Principle) i zbioru realizowanych funkcjonalności skupiając się wokół obiektu domenowego (Domain-Driven Design). Podział bazy danych został wykonany według wzoru DPS (ang. Database Per Service) zachowując spójność modelu danych [21]. W rezultacie otrzymano pięć osobnych baz danych: postgres-prescription, postgres-pharmacy, post-

gres-order, postgres-patient, postgres-medicine. Architekturę zaimplementowanego systemu przedstawia rysunek 4.



Rysunek 4: Architektura mikrousługowa zaimplementowanego systemu.

Środowisko badawcze stanowią aplikacje utworzone w architekturze monolitycznej i mikrousługowej opisane powyżej. Zostały one wdrożone przy wykorzystaniu narzędzia wirtualizacji Docker Swarm na laptopie:

- procesor: Intel core i7-8750H,
- pamięć RAM: 16GB DDR4,
- system operacyjny: Windows 10,
- Docker Engine: v20.10.7,

W badaniach wykorzystywane są dwie konfiguracje platformy sprzętowej w formie maszyny wirtualnej zdefiniowane jako:

- VMB (Virtual Machine – Base) – wirtualna maszyna podstawowa o ograniczonych zasobach (vCPU = 4, RAM = 6 GB),
- VME (Virtual Machine – Extended) – wirtualna maszyna rozszerzona o nieograniczonych zasobach tj. wykorzystująca dostępne zasoby laptopa (vCPU=12, RAM=16GB),

W celu wdrażania aplikacji na środowisku testowym Docker Swarm dla każdej z aplikacji utworzono plik Dockerfile na podstawie którego zostały utworzone obrazy, które następnie umieszczono w repozytorium Docker Hub. Proces budowy i umieszczania obrazów w repozytorium odbywał się przy wykorzystaniu narzędzia CI/CD GitHub Actions. Wykorzystano obrazy apache maven w wersji 3.8.4-jdk-11-slim oraz środowiska JRE (ang. Java Runtime Environment) w wersji openjdk11:jre-11.0.6_10-alpine.

Dane testowe zostały wygenerowane za pomocą kodu napisanego w języku Java w aplikacji testowej wykorzystując Framework Hibernate i zapisane do bazy danych PostgreSQL. Na ich podstawie utworzono skrypty SQL tworzące testowe bazy danych dla aplikacji monolitycznej i mikroserwisowej oraz wypełniające je zawsze tymi samymi rekordami:

- Liczba lekarstwa: 10 000,
- Liczba aptek: 1 000,
- Liczba sztuk leku w aptece: 500,
- Liczba pacjentów: 2 000,
- Każdy pacjent posiada 10 recept i do każdej z nich przypisanych jest 6 leków po 5 sztuk do wybrania,

Do przeprowadzenia testów aplikacji wykorzystano narzędzie Gatling w wersji 3.7, które jest darmowym narzędziem pozwalającym na przeprowadzanie testów wydajnościowych. Wybrano je ze względu na możliwość definiowania scenariuszy testowych opartych o komunikację za pomocą protokołu HTTP. Oferuje funkcjonalności umożliwiające szczegółowe zdefiniowanie scenariuszy jak np. budowanie i przetwarzanie wysyłanych żądań i odpowiedzi w formacie JSON (ang. JavaScript Object Notation) odpowiadającym strukturze obiektów w aplikacjach testowych, sprawdzanie statusów odpowiedzi, wykorzystywanie instrukcji warunkowych oraz pętli, tworzenie zmiennych zapisywanych w sesji użytkownika oraz wykorzystanie funkcjonalności języka Java [15]. Aplikację do testowania stworzono wykorzystując narzędzie Maven, konwersje pomiędzy obiektami Java oraz ich reprezentacją w formacie JSON wykonano przy użyciu biblioteki Gson w wersji 2.9.0. Symulacja składa się z dwóch scenariuszy odpowiadającym korzystaniu z aplikacji przez użytkowników. W celu jak najdokładniejszego odzwierciedlenia tych procesów została utworzona aplikacja webowa stanowiąca część frontend dla testowanych aplikacji pozwalająca na realizację zamówienia od strony użytkowników systemu. Pozwoliło to na zdefiniowane niezbędnych żądań HTTP, sposobu przetwarzania danych po stronie użytkownika oraz zasymulowanie korzystania ze strony internetowej.

3.1. Pierwszy scenariusz badawczy

Pierwszy scenariusz symulacji stanowi odwzorowanie przejścia użytkownika przez aplikację od momentu wczytania danych poprzez złożenie zamówienia, aż po moment jego dostarczenia. Składa się z kroków, których implementację przedstawiono na listingu 1:

1. Pobranie wszystkich zamówień pacjenta.
2. Pobranie wszystkich recept pacjenta i wykonanie dla każdej z nich poniższych czynności.
3. Pobranie recepty.
4. Wyszukanie i pobranie listy aptek według dostępnych leków z recepty.
5. Utworzenie zamówienia wybierając pierwszą aptekę.
6. Pacjent płaci za zamówienie.
7. Pobranie zamówień dla apteki w której zostało złożone powyższe zamówienie.
8. Pobranie zamówienia.

9. Przygotowanie leku przez pracownika apteki dla każdej pozycji z zamówienia.
10. Pracownik apteki inicjalizuje wysyłkę. Firma kurierska jest powiadamiana o przesyłce do doręczenia i generowany jest kod wysyłki.
11. Wysyłka zamówienia.
12. Dostarczenie przesyłki.

Listing 1: Implementacja pierwszego scenariusza symulacji w narzędziu Gatling

```
ScenarioBuilder patientOrderScenario =
scenario("patientOrderScenario")
  .feed(feeder).tryMax(2)
  .on(exec(getOrdersByPatientId)) .tryMax(2)
  .on(exec(getPrescriptionsByPatientId))
  .exitHereIfFailed().foreach("#{prescriptionIds}",
  "prescriptionId" ).on(getPrescriptionById
  .exitHereIfFailed()
  .exec(getPharmaciesWithAvailableMedicines)
  .exitHereIfFailed()
  .exec(createOrder).exitHereIfFailed()
  .exec(payOrderById).exitHereIfFailed()
  .exec(getOrdersByPharmacyId).exitHereIfFailed()
  .exec(getOrderById).exitHereIfFailed()
  .foreach("#{orderItemIds}","orderItemId")
  .on(preparedOrderItem)
  .exec(readyToSend).exitHereIfFailed()
  .exec(deliveryOrderSent).exitHereIfFailed()
  .exec(deliveryOrderDelivered)
  .exitHereIfFailed());
```

3.2. Drugi scenariusz badawczy

Drugi scenariusz symulacji został utworzony analogicznie do pierwszego tak aby odzwierciedlał zachowanie użytkowników, którzy przeglądają swoje recepty. Umożliwia zasymulowanie procesów w systemie, które są uruchamiane bez związku z zadaniem generacji zamówienia, dlatego został nazwany jako „obciążenie tła". Na listingu 2 przedstawiono jego implementację:

1. Pobranie wszystkich recept pacjenta.
2. Pobranie wszystkich zamówień pacjenta.
3. Każda z recept pacjenta zostaje pobrana po kolei w celu wyświetlenia szczegółowych informacji.
4. Pobranie wszystkich recept i zamówień pacjenta, powrót do okna wyboru kolejnej recepty.

Listing 2: Implementacja drugiego scenariusza symulacji w narzędziu Gatling

```
ScenarioBuilder backgroundUser =
  scenario("backgroundUserScenario")
  .feed(feederBackgroundUser)
  .exec(getPrescriptionsByPatientId)
  .exec(getOrdersByPatientId).exitHereIfFailed()
  .foreach("#{prescriptionIds}", "prescriptionId")
  .on(getPrescriptionById
  .exec(getPrescriptionsByPatientIdWithoutSaveIds)
  .exec(getOrdersByPatientId));
```

3.3. Przebieg eksperymentu

Proces testowania polegał na wdrożeniu kolejno aplikacji monolitycznej, mikroserwisowej bez replikacji i mikroserwisowej z replikacjami na konkretnym środowisku testowym, a następnie dla każdej z nich przeprowadzenie symulacji według przyjętych scenariuszy. Kroki przeprowadzenia symulacji dla każdego scenariusza wyglądały następująco:

1. Wdrożenie aplikacji na środowisko testowe (Docker Swarm) za pomocą pliku docker-stack.
2. Przeprowadzenie symulacji ustawiając liczbę użytkowników oraz obciążenie tła według obecnie realizowanego scenariusza.
3. Zapis wyników.
4. Restart środowiska testowego.

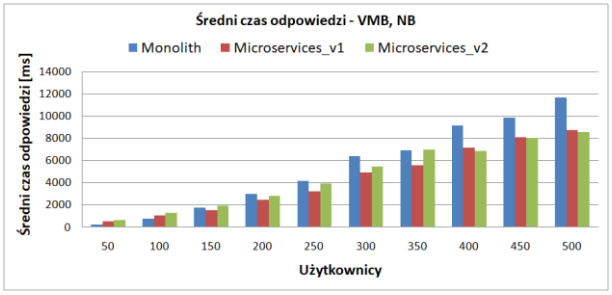
Po wykonaniu wszystkich pomiarów dane eksperymentalne w postaci czasów odpowiedzi serwera oraz parametry związane z obciążeniem aplikacji na wysłane żądania zostały zebrane i pogrupowane według scenriuszy. Testowane aplikacje oznaczono w następujący sposób:

- Monolith – aplikacja monolityczna,
- Microservices_v1 – aplikacja mikroserwisowa bez replikacji (wszystkie komponenty z ustawioną ilością replik na 1),
- Microservices_v2 – aplikacja mikroserwisowa z replikacjami dla VMB,
- Microservices_v3 – aplikacja mikroserwisowa z replikacjami dla VME,

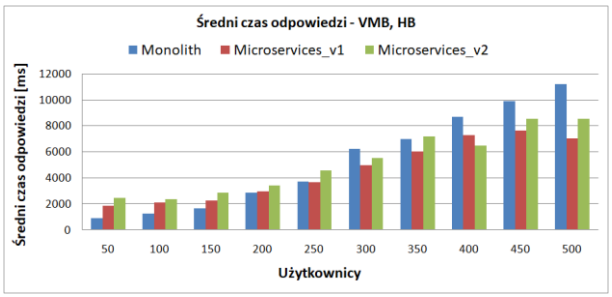
4. Wyniki badań

Wyniki każdej symulacji zostały zebrane, przetworzone i prezentowane przez narzędzie Gatling w formie strony internetowej, która oferuje wiele danych zebranych w postaci interaktywnych wykresów i tabel z których można odczytać szczegółowe dane dotyczące symulacji np. jak zmieniała się liczba aktywnych uczestników w jednostce czasu, liczbę żądań i odwiedzi na sekundę, dystrybuantę czasu odpowiedzi. Uwagę skupiono głównie na zakładce „GLOBAL", w której znajdują się najważniejsze dane z wyników symulacji zebrane w postaci wykresu „Indicators", który przedstawia rozkład czasów odpowiedzi w standardowych zakresach, oraz tabeli „STATISTICS" ze standardowymi danymi statystycznymi takimi jak m.in. minimalny, maksymalny i średni czas odpowiedzi, odchylenie standardowe, percentyle i liczba żądań na sekundę. Stanowi to przejrzystą formę wizualizacji danych ułatwiając ich odczyt i interpretację. W celu porównanie wydajności aplikacji według przyjętych kryteriów wyniki otrzymane z narzędzia Gatling zostały pogrupowane i zebrane w formie tabel na podstawie, których zostały wygenerowane wykresy.

Aplikacja monolityczna charakteryzuje się krótszym czasem odpowiedzi dla scenariuszy z niskim obciążeniem w postaci liczby użytkowników jednak wraz ze wzrostem obciążenia, a tym samym liczby żądań różnica w czasach odpowiedzi zwiększa się na korzyść aplikacji mikroserwisowej i wynosi coraz więcej. Dla podstawowej maszyny wirtualnej zmiana ta następuje dla ok. 200 – 300 użytkowników w zależności od obciążenia dodatkowego i replikacji aplikacji mikrousługowej co przedstawiają wykresy na rysunkach 5 i 6.

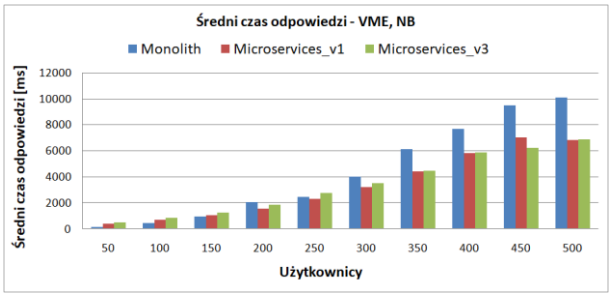


Rysunek 5: Średni czas odpowiedzi, brak dodatkowego obciążenia, podstawowa maszyna wirtualna.

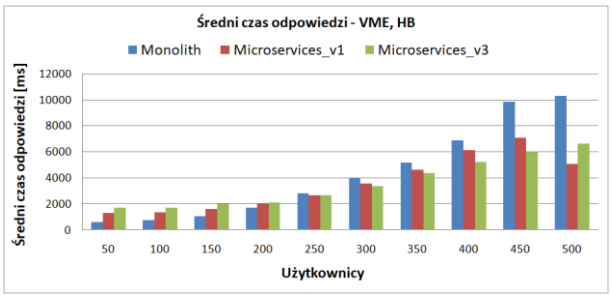


Rysunek 6: Średni czas odpowiedzi, silne obciążenie dodatkowe, podstawowa maszyna wirtualna.

Zmiana na korzyść aplikacji mikrousługowej dla rozszerzonej maszyny wirtualnej zachodzi podobnie w granicach scenariuszy dla 200 – 300 użytkowników i jest widoczna na wykresach przedstawionych na rysunkach 7 i 8.

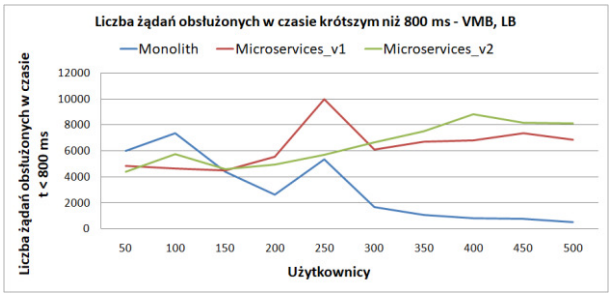


Rysunek 7: Średni czas odpowiedzi, brak dodatkowego obciążenia, rozszerzona maszyna wirtualna.

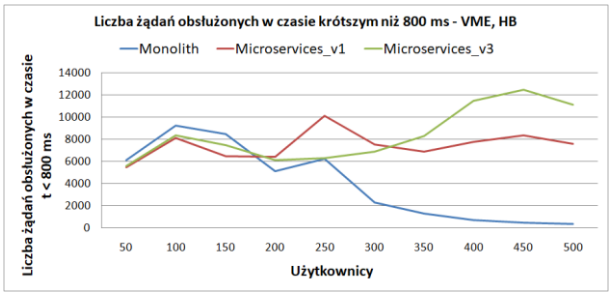


Rysunek 8: Średni czas odpowiedzi, silne obciążenie dodatkowe, rozszerzona maszyna wirtualna.

Rysunki 9 i 10 przedstawiają wykresy liczby żądań obsłużonych w czasie krótszym niż 800 ms. Jest to domyślna wartość ustawiona w wykorzystywanym narzędziu Gatling i określa górną granicę czasu odpowiedzi, do której odpowiedź na żądanie uznawana jest za zrealizowaną w krótkim czasie. Do ok. 150 użytkowników dla podstawowej maszyny wirtualnej i ok. 200 dla rozszerzonej maszyny wirtualnej lepiej radzi sobie z obsługą żądań aplikacja monolityczna, ale później wraz ze wzrostem obciążenia różnica diametralnie się zmienia na korzyść aplikacji mikroserwisowej. Wraz ze wzrostem obciążenia liczba żądań obsłużonych w czasie $t < 800$ ms dla aplikacji monolitycznej maleje podczas gdy dla aplikacji mikroserwisowej rośnie w zależności od scenariusza do 400 lub 450 użytkowników. Na wykresach widoczne jest, że dla aplikacji monolitycznej i mikroserwisowej bez replikacji następuje gwałtowny wzrost wartości dla obciążenia w postaci od 200 do 250 użytkowników oraz znaczny spadek wartości dla scenariuszy od 250 do 300 użytkowników.



Rysunek 9: Liczba żądań obsłużonych w czasie krótszym niż 800 ms, słabe obciążenie dodatkowe, podstawowa maszyna wirtualna.



Rysunek 10: Liczba żądań obsłużonych w czasie krótszym niż 800 ms, silne obciążenie dodatkowe, rozszerzona maszyna wirtualna.

5. Podsumowanie

Tabela 1 przedstawia zestawienie ze sobą badanych aplikacji pod względem otrzymanych wartości parametrów rejestrowanych w poszczególnych scenariuszach testowych. Wyróżniono dwie kategorie obciążenia aplikacji, małe i duże na podstawie liczby wysyłanych żądań związanych z liczbą wirtualnych użytkowników. Dodatkowo w tabeli uwzględniono podział według konfiguracji platformy sprzętowej na VMB i VME. Kolorami odpowiadającymi poszczególnym rodzajom aplikacji zaznaczono, które z nich prezentują najlepsze wyniki dla danego scenariusza i kryterium.
Oznaczenia kolorów:

- niebieski – aplikacja oparta o architekturę monolityczną,
- pomarańczowy – aplikacja wykorzystująca mikrousługi,
- zielony – aplikacja wykorzystująca mikrousługi ze skalowaniem,
- szary – wyniki porównywalne dla wszystkich aplikacji,

Tabela 1: Zestawienie badanych aplikacji pod względem otrzymanych wartości parametrów rejestrowanych w poszczególnych scenariuszach testowych

Badane kryterium	Małe obciążenie		Duże obciążenie	
	VMB	VME	VMB	VME
Liczba poprawnie obsłużonych żądań				
Liczba przetworzonych żądań na sekundę				
Różnica liczby obsłużonych żądań w jednostce czasu pomiędzy aplikacją mikrousługową i monolityczną				
Średni czas odpowiedzi				
Odchylenie standardowe czasu odpowiedzi				
Liczba żądań obsłużonych w czasie krótszym niż 800 ms				
Procentowy udział żądań z czasem odpowiedzi krótszym niż 800 ms spośród poprawnie obsłużonych żądań				

Z tabeli wynika, że dla małego obciążenia zarówno dla środowiska VMB jak i VME niemal dla każdego badanego kryterium lepsza jest aplikacja oparta o architekturę monolityczną. Jedynie w przypadku liczby poprawnie obsłużonych żądań wyniki są porównywalne dla wszystkich aplikacji, a podczas dużego obciążenia dla podstawowej maszyny wirtualnej widać przewagę aplikacji monolitycznej. Natomiast dla VME najlepsze wyniki osiąga aplikacja mikrousługowa ze skalowaniem. Aplikacja wykorzystująca mikrousługi bez skalowania najlepiej sprawdza się w przypadku dużego obciążenia dla podstawowej maszyny wirtualnej. Osiąga ona najlepsze rezultaty dla kryteriów związanych z liczbą przetworzonych żądań na sekundę, średnim czasem odpowiedzi oraz odchyleniem standardowym, które wskazuje na jej stabilność podczas zmiany obciążenia. Podczas dużego obciążenia dla rozszerzonej maszyny wirtualnej najlepsze wyniki dla wszystkich kryteriów osiąga aplikacja wykorzystująca mikrousługi ze skalowaniem. Wynika to z większych zasobów sprzętowych wpływających na wydajność uruchamianych replik. Skalowanie aplikacji mikrousługowej umożliwia przede wszystkim poprawę wydajności związanej z szybszą obsługą większej liczby żądań na co wskazują kryteria związane z obsługą żądań z czasem odpowiedzi krótszym niż 800 ms.

Literatura

[1] C. Richardson, Mikroserwisy: Wzorce z przykładami w języku Java, PWN, 2020.

[2] P. Mell, T. Grance, et al. The NIST definition of cloud computing. National Institute of Standards and Technology Special Publication 800-145, Gaithersburg (2011) 1-7.

[3] V. Andrikopoulos, T. Binz, F. Leymann, S. Strauch, How to adapt applications for the cloud environment. Challenges and solutions in migrating applications to the cloud, Computing 95(6) (2013) 493–535.

[4] P. Jamshidi, A. Ahmad, C. Pahl, Cloud migration research: A systematic review, IEEE Transactions on Cloud Computing 1(2) (2013) 142–157.

[5] A. Balalaie, A. Heydarnoori, P. Jamshidi, Migrating to Cloud-Native Architectures Using Microservices. An Experience Report, European Conference on Service-Oriented and Cloud Computing (2015) 201-215.

[6] L. Bass, I. Weber, L. Zhu, DevOps: A Software Architect's Perspective, O'Reilly, 2019.

[7] S. Stoja, S. Vukmirovic, N. Dalcekovic, D. Capko, Accelerating Performance in Critical Topology Analysis of Distribution Management System Process by Switching from Monolithic to Microservices, Revue Roumaine des Sciences Techniques Serie Electrotechnique et Energetique 63 (2018) 338-343.

[8] K. Cebeci, Ö. Korçak, Design of an Enterprise Level Architecture Based on Microservice, Bilişim Teknolojileri Dergisi 13 (2020) 357-371.

[9] B. Shafabakhsh, R. Lagerström, S. Hacks, Evaluating the Impact of Inter Process Communication in Microservice Architectures, International Workshop on Quantitative Approaches to Software Quality 2767 (2020) 55–63.

[10] Strona główna Apache JMeter, https://jmeter.apache.org/, [27.05.2022].

[11] V. Adamescu, Analysing monolithic and microservices software architecture for SME web services/applications, (2020) https://www.researchgate.net/publication/341353952_Analysing_monolithic_and_microservices_software_architecture_for_SME_web_servicesapplications

[12] D. Taibi, V. Lenarduzzi, P, Claus, Architectural Patterns for Microservices: A Systematic Mapping Study, Closer (2018) https://hdl.handle.net/10863/5599

[13] F. Vera-Rivera, H. Astudillo, M. Gaona, Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación, Revista Iberica de Sistemas e Tecnologias de Informacao E23 (2019) 107 - 120.

[14] Strona główna narzędzia testowego Locust, https://locust.io/, [27.05.2022].

[15] Dokumentacja narzędzia Gatling, https://gatling.io/docs/gatling/reference/3.7/, [27.05.2022].

[16] S. Shrivastava, S. B. Prapulla, Comprehensive Review of Load Testing Tools, IRJET (2020) 3392-3395.

[17] A. Raj, K. Jasmine, Building Microservices with Docker Compose, The International journal of analytical and experimental modal analysis XIII (2021) 1215- 1219.

[18] S. Newman, Budowanie mikrousług. Projektowanie drobnoziarnistych systemów, Helion, 2022.

[19] S. Newman, Monolith to Microservices. Evolutionary Patterns to Transform Your Monolith, O'Reilly, 2019.

[20] Decompose by transactions, https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-decomposing-monoliths/decompose-transactions.html, [27.05.2022].

[21] Wzorce projektowe architektury mikrousługowej, https://microservices.io/patterns/index.html, [27.05.2022].

[22] Dokumentacja Spring Cloud, https://spring.io/projects/spring-cloud, [27.05.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 401–408
Received: 3 September 2022
Accepted: 20 September 2022

Applying universal design principles to improve the websites of a selected university

Zastosowanie zasad projektowania uniwersalnego do ulepszenia stron internetowych wybranej uczelni wyższej

Tomasz Kamiński*, Paweł Kapica*, Mariusz Dzieńkowski

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

Abstract
The purpose of the study was to analyze a website of a selected university in terms of usability and accessibility of user interfaces with particular attention to the guidelines of the Web Content Accessibility Guidelines (WCAG) standard. After taking into consideration all the guidelines, an improved prototype version of the site was created that was free of the diagnosed errors and incompatibilities. Both sites were assessed using three methods: the questionnaire method with the LUT checklist, using the eye tracking technique and by means of an automated tool - the WAVE plugin attached to a web browser. Twenty participants took part in the questionnaire and the eye tracking study. The data obtained from the research experiment in which three methods were used were analyzed quantitatively. However, the results obtained after assessing with the eye tracking technique were additionally subjected to a qualitative analysis (heat maps, scan-paths). The results of the performed analyses show unequivocally that the prototype website prepared by the authors in accordance with universal design principles is clearly better in terms of usability and accessibility than the website of the selected university.

Keywords: universal design; usability; accessibility; eye tracking; WCAG; WAVE; LUT

Streszczenie
Celem pracy była analiza strony internetowej wybranej uczelni pod względem użyteczności i dostępności interfejsów użytkownika ze szczególnym uwzględnieniem wytycznych zawartych w standardzie WCAG (ang. Web Content Accessibility Guidelines). Po uwzględnieniu wszystkich wytycznych powstała ulepszona prototypowa wersja witryny, która została pozbawiona zdiagnozowanych błędów i niezgodności. Obie witryny zostały przebadane trzema metodami: metodą kwestionariuszową za pomocą listy kontrolnej LUT, z wykorzystaniem techniki okulograficznej oraz automatycznego narzędzia - wtyczki WAVE dołączonej do przeglądarki internetowej. W badaniach ankietowych i okulograficznych wzięło udział 20 uczestników. Dane pozyskane z badań wykonanych trzema metodami zostały poddane analizie ilościowej. Natomiast wyniki badań eyetrackingowych zostały dodatkowo poddane analizie jakościowej (mapy cieplne, ścieżki skanowania). Wyniki przeprowadzonych analiz jednoznacznie pokazują, że prototypowa witryna przygotowana przez autorów pracy zgodnie z zasadami projektowania uniwersalnego wyraźnie lepiej wypada pod względem użyteczności i dostępności niż witryna wybranej uczelni.

Słowa kluczowe: projektowanie uniwersalne; użyteczność; dostępność; eyetracking; WCAG; WAVE; LUT

*Corresponding author

Email address: **tomasz.kaminski1@pollub.edu.pl** (T. Kamiński), **pawel.kapica@pollub.edu.pl** (P. Kapica)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Strona internetowa uczelni wyższej odgrywa bardzo ważną rolę dla studentów, kandydatów na studia, pracowników oraz otoczenia biznesowego. Dlatego powinna ona mieć czytelny i komfortowy w obsłudze interfejs graficzny. Strona www jest także wizytówką uczelni, ponieważ zawiera o niej najważniejsze informacje oraz zapewnia różne formy kontaktu. Studentami uczelni są także osoby z rożnymi niepełnosprawnościami, dlatego tak ważne jest, aby każda instytucja tego typu zapewniła bezproblemowy dostęp do informacji zawartych także dla tej grupy osób. Kwestia ta jest w Polsce uwarunkowana prawnie przy pomocy ustawy [1] o dostępności cyfrowych stron internetowych i aplikacji mobilnych podmiotów publicznych z dnia 4 kwietnia 2019 r. Informacje zawarte w ustawie określają jasno jakie wytyczne powinny zostać zastosowane dla aplikacji mobilnych i witryn internetowych w zakresie dostępności. Wymagania te pokrywają się z punktami 9, 10 i 11 normy EN 301 549 V2.1.2 [2].

Projektowanie uniwersalne sugeruje jak projektować i wytwarzać produkty, które będą dostosowane do jak największej grupy odbiorców, bez potrzebny adaptacji lub specjalistycznego przygotowania, czy szkolenia [3, 4]. Zastosowanie zasad projektowania uniwersalnego do stron internetowych i systemów informatycznych umożliwia dostęp do informacji i usług szerokiej grupie odbiorców. Dzięki temu osoby z rożnymi dysfunkcjami (np. osoby z wadami wzorku, czy słuchu) będą w stanie obsługiwać takie systemy. Zasady projektowania uniwersalnego obejmują siedem aspektów [5]:

1. Równe użytkowanie dla osób o różnych zdolnościach.

2. Elastyczność użytkowania uwzględniająca szeroki zakres indywidualnych preferencji i możliwości.
3. Proste i intuicyjne użytkowanie, niezależne od doświadczenia, wiedzy, umiejętności językowych czy aktualnego poziomu koncentracji użytkownika.
4. Łatwa zauważalność niezbędnych dla użytkownika informacji.
5. Tolerancja na błędy – minimalizacja zagrożeń i negatywnych konsekwencji przypadkowych lub niezamierzonych działań.
6. Niski poziom wysiłku fizycznego.
7. Wymiary i przestrzeń umożliwiające podejście, sięgnięcie, manipulację i użycie, niezależnie od wielkości ciała, postawy i mobilności użytkownika.

Zwiększenie dostępności informacji zawartych na stronach internetowych zapewniają wytyczne WCAG. Wykorzystanie tych wskazówek sprawi, że informacje będą przystosowane także dla osób z niepełnosprawnością ruchową, nadwrażliwością na światło, z zaburzeniami mowy, z niepełnosprawnościami złożonymi, a także dla niektórych osób mających trudności w uczeniu się i z ograniczeniami poznawczymi. Główną zaletą WCAG jest niepowiązanie z konkretną technologią, dlatego umożliwia to implementację swoich kryteriów w dowolnym języku programowania. Struktura WCAG 2.1 opiera się na czterech głównych zasadach: postrzegalność, funkcjonalność, zrozumiałość, solidność (w polskim i unijnym prawie określana jako kompatybilność) [6].

Do najpopularniejszych badań użyteczności stron internetowych zalicza się: moderowane i niemoderowane testy z użytkownikami, analizę okulografem, badania ankietowe, sortowanie kart oraz prototypowanie na papierze. Moderowane testy z użytkownikami polegają na testowaniu użyteczności strony przy obecności opiekuna badania, podczas którego użytkownik wykonuje zadania. Obserwator zwraca uwagę na trudności napotykane przez osoby badane. Testy niemoderowane z użytkownikami przeprowadzane są bez obserwatora, dzięki czemu uczestnicy zachowują się w sposób naturalny, nie odczuwając żadnej presji. Za pomocą programów do nagrywania pulpitu organizatorzy badań monitorują działania użytkownika. Analiza okulografem polega na śledzeniu punktów skupienia (zatrzymań) oraz ścieżek podążania wzroku osoby badanej na badanym materiale wizualnym (bodźcu). W efekcie tego rodzaju badań otrzymuje się mapy ciepła, które pokazują elementy, jakie wzbudziły największe zainteresowanie na stronie www wśród uczestników eksperymentu. Za pomocą badań ankietowych specjaliści od UX (ang. User Experience) zapoznają się z opiniami respondentów na temat intuicyjności witryny, szybkości dotarcia do informacji i łatwości nawigowania. Kolejna metoda - sortowanie kart - pozwala na sprawdzenie, jak użytkownicy postrzegają hierarchię elementów na stronie internetowej. Inną metodą jest prototypowanie na papierze polegające na tworzeniu papierowych makiet z interfejsem stron internetowych i ich omawianiu z użytkownikami.

Niniejsza praca skupia się na użyteczności i dostępności interfejsu stron internetowych przeznaczonych dla wszystkich użytkowników, nie tylko osób pełnosprawnych. W ramach pracy wybrano przykładową stronę uczelni w Polsce, zaimplementowano alternatywną wersję tej strony, uwzględniającą zasady projektowania uniwersalnego oraz przeprowadzono badania tych witryn pod względem ergonomii, rozmieszczenia elementów, dostępności oraz łatwości nawigacji po witrynie.

2. Przegląd literatury

Ocenę serwisów internetowych uczelni wyższych w Polsce opisuje artykuł [7]. Badania zostały przeprowadzone za pomocą dwóch metod: eyetrackingowej oraz kwestionariuszowej. Dane eyetrackingowe zostały przenalizowane jakościowo i ilościowo, a dane z ankiety ilościowo. Do prezentacji wyników zastosowano między innymi mapy cieple czy też ścieżki skanowania.

Głównym zadaniem uniwersytetów jest kształcenie studentów z różnych środowisk, także i niepełnosprawnych. Aby sprostać różnorodnym potrzebom i możliwościom uczenia się w salach, nauczyciele muszą znaleźć innowacyjne metody dotarcia nie tylko do osób pełnosprawnych, ale także niepełnosprawnych. Jednym z potencjalnych rozwiązań jest UDL (ang. Universal Design for Learning) [8].

O problematyce dostępności publicznych serwisów internetowych dla osób z niepełnosprawnością traktuje artykuł [9]. Autorzy ukazują potrzebę dostosowania przekazywania informacji drogą internetową dla każdego użytkownika, podkreślają istotę projektowania uniwersalnego. Dodatkowo ukazują zasady i wytyczne, którymi należy się kierować podczas projektowania stron. Autorzy wskazują na znaczny w ostatnim czasie wzrost dostępności stron internetowych wśród instytucji w Polsce.

W artykule [10] zaprezentowano model ULD (ang. Universal Learning Design) - projektowania uniwersalnego w edukacji, jego zasad oraz stosowanych metod i form nauczania i uczenia się, szczególnie w odniesieniu do uczniów słabosłyszących i niesłyszących. Między innymi dzięki modelowi ULD funkcjonującego w ramach edukacji inkluzyjnej, zgodnego z bio-psychospołecznym modelem podejścia do niepełnosprawności. Autor wskazuje, że organizacja nauczania powinna zakładać wykorzystanie zróżnicowanych środków nauczania oraz materiałów, a także różne ośrodki pedagogiczne i technologiczne wspierające motywacje i zaangażowanie uczniów.

Bariery, które napotykają osoby z niepełnosprawnością wzroku w dostępie do stron internetowych ukazuje artykuł [11]. Badaniom zostały poddane strony głównych bibliotek polskich uniwersytetów. Do badania wykorzystano metodę jakościowo-heurystyczną i automatycznych procedur. Badania wykazały, że strony internetowe bibliotek uniwersyteckich w Polsce odbiegają od standardów stosowanych na świecie w zakresie dostępności dla osób niewidzących i słabowidzących.

O problemie nieprzystosowania stron internetowych dla seniorów i osób niepełnosprawnych opowiada arty-

kuł [12], w którym zostały wyspecyfikowane zasady projektowania i budowania stron internetowych dla tej grupy. Celem artykułu było zwrócenie uwagi na ten wciąż nierozwiązany problem i w związku z tym zostały zaprezentowane różnorodne propozycje udogodnień dla seniorów i osób niepełnosprawnych.

O podstawowych zasadach dostępności serwisów internetowych możemy dowiedzieć się z artykułu [13], dzięki czemu są one użyteczne dla wszystkich użytkowników, także tych niepełnosprawnych. Przedstawiono także korzyści z budowania stron internetowych zgodnych ze standardami dostępności W3C (ang. Word Wide Web Consortium).

W pracy [14] zostały przedstawione wyniki testów pięciu wybranych internetowych platform edukacyjnych pod względem ich zgodności z zasadami projektowania uniwersalnego oraz dostępności dla osób z różnymi rodzajami niepełnosprawności na podstawie realizacji wytycznych zawartych w rekomendacji ATAG 2.0 (ang. Authoring Tool Accessibility Guidelines). Badania pokazały, że główne funkcjonalności analizowanych platform są dostępne również dla osób niepełnosprawnych.

3. Cel i zakres pracy

Celem pracy jest analiza istniejącej strony uczelni wyższej w kontekście zgodności z zasadami projektowania uniwersalnego, ze szczególnym uwzględnieniem standardu WCAG 2.1, stworzenie nowej wersji strony pozbawionej wykrytych błędów oraz przeprowadzenie badań pozwalających na porównanie interfejsów obu stron za pomocą eyetrackera, ankiety i narzędzia WAVE. Zakres przeprowadzonych w pracy działań obejmował:

- wybór obiektu badawczego – witryny internetowej uczelni wyższej,
- opracowanie strony prototypowej – będącej odpowiednikiem analizowanej strony, ale uwzględniającej zasady projektowania uniwersalnego,
- dobór odpowiednich metod badawczych,
- zorganizowanie grupy badawczej,
- opracowanie i przeprowadzenie eksperymentu badawczego,
- analiza otrzymanych wyników i sformułowanie wniosków.

4. Metody badawcze

W pracy zostały zastosowane trzy metody oceny jakości interfejsów: eyetrackingowa, kwestionariuszowa oraz wykorzystująca narzędzie WAVE do analizy statycznej witryn internetowych.

4.1. Obiekty badań

Przy wyborze obiektu badań, autorom pracy zależało na znalezieniu takiej strony internetowej, która zawierałaby pewną liczbę błędów oraz elementów, które można byłoby poprawić. Ponadto ustalono, że będzie to strona uczelni wyższej. Ważne było również to, aby uczestnicy badania nie mieli wcześniejszej styczności z badaną stroną, co zostało przez nich potwierdzone przed rozpoczęciem badań. Przed realizacją eksperymentu przeprowadzona została analiza wielu stron internetowych uczelni wyższych, z których wybrano stronę Wyższej Szkoły Informatyki i Zarządzania w Bielsku-Białej (WSIIZ). Drugim obiektem badań była prototypowa strona internetowa utworzona na podstawie witryny WSIIZ, która nie zawierała problemów istniejącej strony nawiązujących do kwestii dostępności cyfrowej, takich jak postrzegalność czy funkcjonalność.

4.2. Grupa badawcza

W badaniach wzięło udział 20 osób. Wszystkie badane osoby były studentami Wydziału Elektrotechniki i Informatyki Politechniki Lubelskiej z kierunku Informatyka. Wszyscy uczestnicy badań nie mieli wcześniej do czynienia z analizowanymi w pracy stronami internetowymi.

4.3. Stanowisko badawcze

Badania przeprowadzone za pomocą stacjonarnego eyetrackera zostały wykonane w laboratorium Katedry Informatyki Politechniki Lubelskiej, w którym zapewnione zostały optymalne warunki oświetleniowe. Każda badana osoba znajdowała się w odpowiedniej odległości od eyetrackera, a krzesło pozwalało na regulację wysokości siedzenia, zapewniając w ten sposób swobodę ruchów.

Użyty do celów badań eyetracker zdalny Gazepoint GP3 HD [15] miał następujące parametry techniczne:

- częstotliwość próbkowania 150Hz przy śledzeniu obuocznym,
- dokładność (obuoczna) mierzona w idealnych warunkach: 0,5-1°,
- kalibracja 9-punktowa,
- technika śledzenia: jasna źrenica,
- zakres swobody poruszania głową: 35 cm w poziomie i 22 cm w pionie,
- odległość oczu badanego od eyetrackera: około 65cm,
- detekcja stanów oka: fiksacji, sakad, zmian średnicy źrenicy, mrugnięć.

Eyetracker był podłączony do komputera przenośnego Acer Nitro 5 wyposażonego w:

- procesor AMD Ryzen 7 5800H 3.20Gz,
- kartę graficzną NIVIDA Geforce RTX 3060 6GB,
- pamięć RAM 32GB,
- ekran Full HD o częstotliwości 144 Hz, rozdzielczości 1920 x 1080 oraz o przekątnej ekranu 17,3",
- system operacyjny Windows 10 x64.

Na komputerze było zainstalowane oprogramowanie iMotions 9.1, które umożliwia [16]:

- zaprojektowanie eksperymentu,
- przeprowadzenie kalibracji,
- nagrywanie kolejnych sesji uczestników badań,
- odtwarzanie i edycję nagrań,
- wizualizację wyników w postaci map cieplnych, map uwagowych i ścieżek skanowania, roi pszczół,

- wyznaczanie obszarów zainteresowania i robienie dla nich statystyk,
- eksport surowych danych i ich dalszą zaawansowaną analizę.

4.4. Eksperyment

Przed przeprowadzeniem eksperymentu właściwego konieczne było przygotowanie odpowiednich zadań dla uczestników, pozwalających na porównanie obu interfejsów. Bezpośrednio przed przystąpieniem do badania, każdy z uczestników został poinstruowany o dokładnym jego celu i przebiegu. Podczas badania obecna była osoba je nadzorująca. Osobie biorącej udział w eksperymencie było wyświetlane zadanie, a następnie po zapoznaniu się z nim, mogła ona przejść do kolejnej planszy, na której widoczny był zrzut ekranu witryny internetowej, której dotyczyło polecenie. Każdy z badanych miał do wykonania 10 takich zadań. Uczestnicy zostali podzieleni na dwie grupy: jedna wykonywała polecenia na istniejącej stronie uczelni, zaś druga na stronie prototypowej. Treść zadań, które wykonywali respondenci zebrano w Tabeli 1.

Tabela 1. Treść zadań

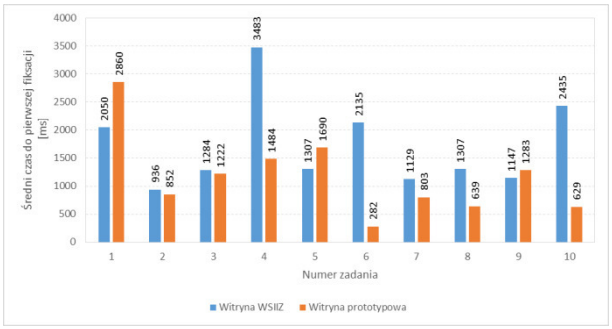
Lp.	Treść zadania
1	Na jakiej ulicy znajduje się uczelnia?
2	Które pole w formularzu jest nieobowiązkowe?
3	Na jakiej stronie się znajdujesz?
4	Kto jest kwestorem uczelni?
5	Znajdź przycisk do zmiany kontrastu.
6	Kiedy rozpoczyna się nabór kandydatów na studia?
7	Znajdź wyszukiwarkę (pole do wprowadzania szuka-nych słów i zwrotów).
8	Jakie są rodzaje przyznawanych świadczeń dla studentów?
9	Jaka jest cena wpisowa za czesne?
10	Znajdź przycisk uruchomienia Facebooka.

5. Wyniki badań

5.1. Wyniki badań eyetrackingowych – analiza ilościowa

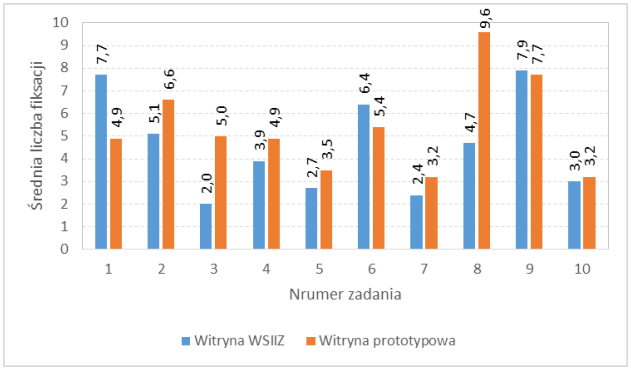
Przeprowadzenie analizy znacząco ułatwiły możliwości programu iMotions. Jedną z miar eyetrackingowych wykorzystanych do analiz był czas do pierwszej fiksacji, określający czas jaki upłynął od wyświetlenia bodźca do momentu kiedy wzrok osoby biorącej udział w eksperymencie znalazł się w określonym dla każdego zadania obszarze AOI (ang. Area of Interest). Uzyskane wyniki zostały uśrednione w obrębie badanej grupy oraz przedstawione na Rysunku 1.

TTFF (ang. Time to First Fixation) jest czasem, który upływa kiedy osoba badana skieruje swój wzrok na określony przez autorów obszar zainteresowania. Dla każdego zadania został ręcznie wyznaczony odpowiedni obszar. Czas do pierwszej fiksacji pozwala ocenić w jakim stopniu zbudowany interfejs jest intuicyjny i przyjazny dla użytkownika. Od tych aspektów zależy sprawność działania na stronie. W przypadku gdy użytkownik ma do czynienia z wysoko nieintuicyjnym interfejsem - czas ten będzie długi. Natomiast działając na przemyślanych i przejrzystych stronach, użytkownicy będą w stanie w krótkim czasie odnaleźć interesujące ich informacje i elementy.



Rysunek 1: Średni czas do pierwszej fiksacji na szukanym elemencie dla poszczególnych zadań wykonywanych na badanych stronach.

Na Rysunku 1 widoczne są czasy wykonywania poszczególnych zadań na stronie istniejącej oraz prototypowej - opracowanej przez autorów pracy. W przypadku siedmiu zadań czasy do pierwszej fiksacji były dłuższy dla strony WSIIZ, natomiast dla trzech zadań czas ten był dłuższy dla witryny prototypowej. Szczególnie duże różnice na korzyść strony autorskiej wystąpiły w czasach realizacji zadań nr 4, 6, 8 i 10. Jedynie przy wykonywaniu zadania 1 znacznie krótszy TTFF był dla strony WSIIZ.

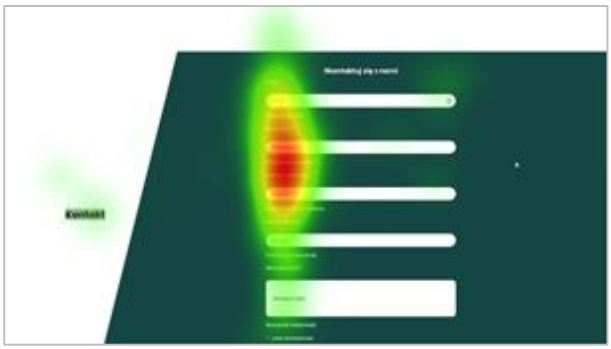


Rysunek 2: Średnia liczba fiksacji w obszarze zainteresowania dla poszczególnych zadań.

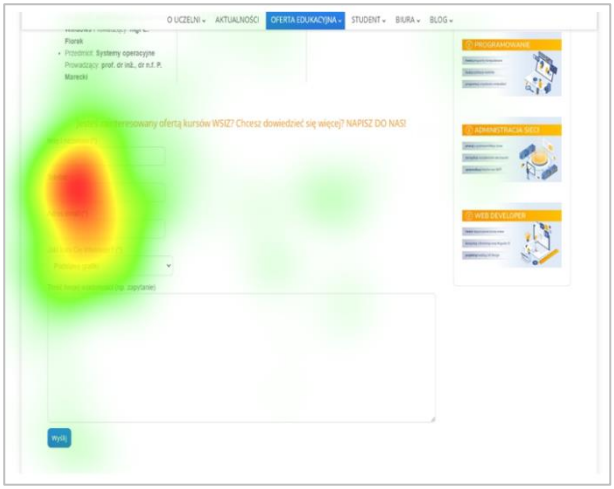
W drugiej analizie eyetrackingowej wykorzystano miarę, będącą sumaryczną liczbą wystąpień fiksacji w danym obszarze zainteresowania, w którym znajdował się szukany element lub szukana informacja. Metryka ta oznacza, jak często badana osoba skupiała swój wzrok na wyznaczonym obszarze. Na Rysunku 2 można zaobserwować średnie liczby fiksacji dla poszczególnych zadań, oddzielnie dla strony autorskiej oraz istniejącej - WSIIZ. W każdym z analizowanych przypadków liczba ta jest większa na stronie przygotowanej przez autorów. Z jednej strony może wynikać to z tego, że więcej zadań na stronie prototypowej zostało wykonanych prawidłowo oraz także może wskazywać na to że, wyznaczony obszar był bardziej zauważalny, bardziej przyciągający uwagę użytkowników.

5.2. Wyniki badań eyetrackingowych – analiza jakościowa

Analiza jakościowa została oparta na mapach cieplnych i ścieżkach skanowania. Mapy cieplne zostały wygenerowane dla grup osób dla poszczególnych zadań. Natomiast ścieżki skanowania przedstawiają trasy śledzenia pojedynczych osób na konkretnym bodźcu wizualnym powiązanym z danym zadaniem. Mapy cieplne to zrzuty ekranowe stron www z nałożonym transparentnie kolorem czerwonym, żółtym i zielonym - przedstawiającym intensywność patrzenia badanych osób. Kolor czerwony określa obszar, w którym długo lub często znajdowała się uwaga użytkowników. W zaprezentowanych na Rysunkach 3, 4, 5, 6 mapach cieplnych gorące obszary, pokrywają się z miejscem lokalizacji szukanych elementów. Z tego wynika, że uczestnicy badań prawidłowo namierzali szukane elementy. Jednak analizując dokładnie te mapy można stwierdzić, iż uczestnicy nie znajdowały szukanego elementu czy informacji od razu, co ukazują inne liczne zielone i żółte obszary. Rysunki 3 i 4 prezentują fragmenty map cieplnych dla zadania 2, zaś 5 i 6 dla zadania numer 7.

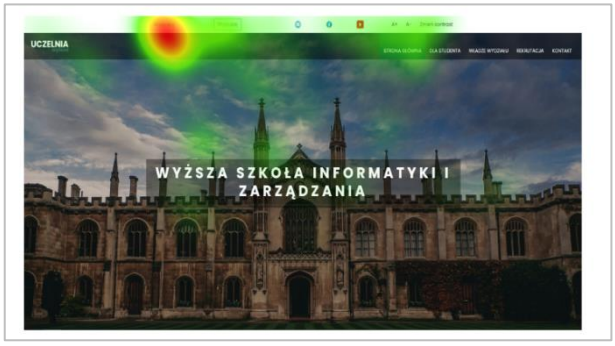


Rysunek 3: Wynik realizacji zadania 2 w postaci mapy ciepła dla witryny prototypowej.



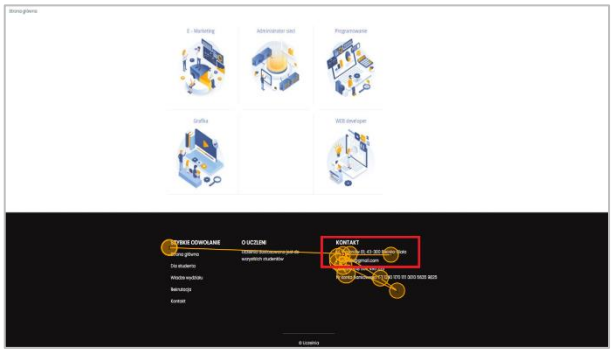
Rysunek 4: Wynik realizacji zadania 2 w postaci mapy cieplnej dla witryny WSIIZ.

Rysunek 5: Wynik realizacji zadania 7 w postaci mapy cieplnej dla strony WSIIZ.



Rysunek 6: Wynik realizacji zadania 7 w postaci mapy cieplnej dla strony autorskiej.

Na kolejnych Rysunkach (7 – 10) zaprezentowano przykładowe ścieżki skanowania. Czerwonym prostokątem oznaczono docelowe elementy, szukane przez uczestników badań.

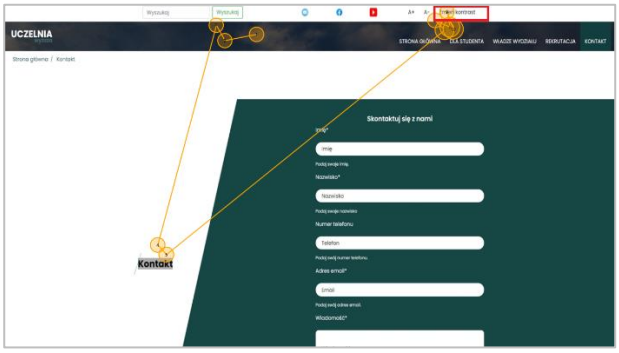


Rysunek 7: Wynik poprawnej realizacji zadania 1 w postaci ścieżki skanowania dla strony WSIIZ.

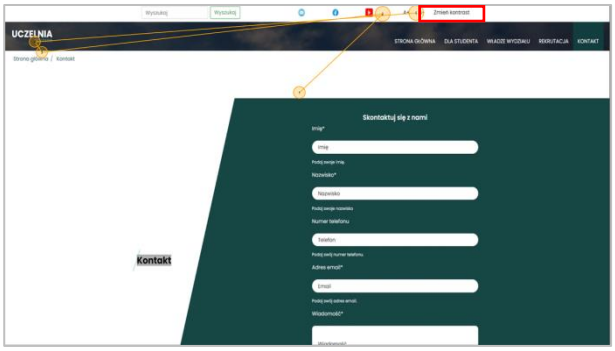
Rysunki 7 i 8 przedstawiają widok fragmentu strony WSIIZ z nałożonymi ścieżkami skanowania, podczas realizacji zadania 1 – szukania ulicy, przy której znajduje się uczelnia. Natomiast Rysunki 9 i 10 dotyczą strony prototypowej z formularzem, na której użytkownicy mieli odnaleźć narzędzie do zmiany kontrastu.

Rysunek 7: Wynik niepoprawnej realizacji zadania 1 w postaci ścieżki skanowania dla strony WSIIZ.

Rysunki 7 i 9 ukazują prawidłowe wykonanie zadania, polegające na odnalezieniu właściwego elementu. Natomiast na Rysunkach 8 i 10 można zauważyć, że osoby badane miały trudności z ze znalezieniem szukanych informacji tekstowych.



Rysunek 8: Wynik poprawnej realizacji zadania 5 w postaci ścieżki skanowania dla strony prototypowej.



Rysunek 9: Wynik nieprawidłowej realizacji zadania 5 w postaci ścieżki skanowania dla strony prototypowej.

5.3. Wyniki badań kwestionariuszowych

Celem tej części badań była ocena interfejsów obu witryn internetowych przez użytkowników metodą kwestionariuszową za pomocą listy kontrolnej LUT (skrót LUT pochodzi od Lublin University of Technology) [17]. Ankieta składa się z pięciu głównych obszarów, które są podzielone na podobszary i pytania w obrębie każdego z podobszarów. Na każde pytanie badana osoba odpowiadała oceną w zakresie od 1 do 5. Wartość 1 oznaczała bardzo słaby wynik, natomiast 5 - bardzo dobry. Ocenę jakości aplikacji przeprowadzono z wykorzystaniem metryki WUP (ang. Web Usability Points). Liczba punktów WUP może przyjmować wartości od 1 do 5, przy czym większa wartość oznacza lepiej zaprojektowany interfejs.

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

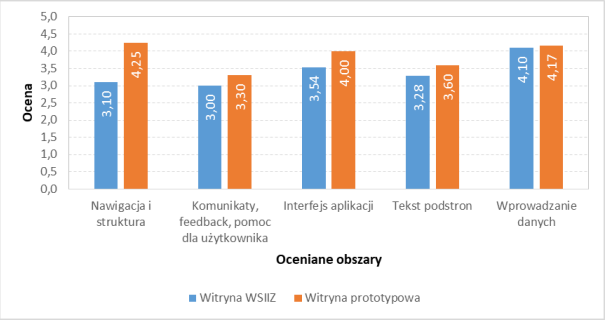
gdzie n_a jest liczbą obszarów, s_i liczbą podobszarów w obszarze j, q_{ij} liczbą pytań w obszarze i podobszarze j, a p_{ijk} jest oceną pytania o numerze k w obszarze i oraz podobszarze j.

Po przeanalizowaniu wyników ankiet i skorzystaniu z wzoru 1 otrzymano wyniki uwidocznione w Tabeli 2.

Tabela 2. Ocena jakości interfejsów witryn za pomocą metryki WUP

Witryna	Wskaźnik WUP	Odchylenie standardowe
WSIIZ	3,16	0,57
Prototypowa	4,05	0,55

Na Rysunku 11 przedstawiono średnie oceny dla poszczególnych obszarów ocenianych witryn internetowych.



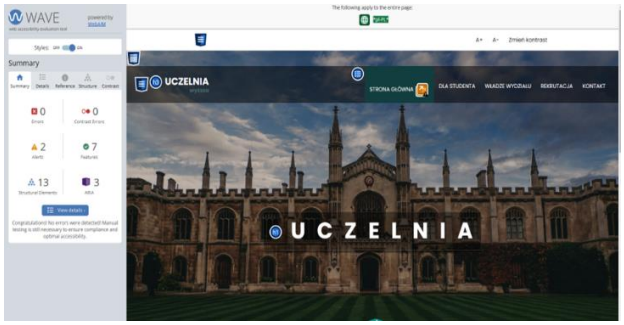
Rysunek 10: Wykres średnich ocen dla poszczególnych obszarów.

Z Rysunku 11 wynika, że użytkownicy we wszystkich obszarach wyżej ocenili stronę prototypową niż stronę WSIIZ. Stało się to dzięki wdrożeniu zmian wynikających z zasad projektowania uniwersalnego. Największa poprawa widoczna jest w obszarach nawigacji i struktury stron oraz kompozycji interfejsu serwisu. Wpływ na te aspekty miało wyeliminowanie błędów kontrastowych oraz uczynienie nawigacji bardziej uporządkowanej i przejrzystej. Na stronie prototypowej najniżej został oceniony obszar związany z tekstem podstron, choć mimo to ocena ta i tak była wyższa niż ocena strony WSIIZ.

5.4. Wyniki badań narzędziem WAVE

Następnym etapem eksperymentu było przeprowadzenie badania witryn za pomocą automatycznego narzędzia WAVE, za pomocą którego można ocenić różne aspekty witryn, a przede wszystkim poziom dostępności. Celem tego badania była analiza porównywanych interfejsów, polegająca na sprawdzeniu ich zgodności względem wytycznych WCAG, czyli zbioru zaleceń dotyczących tworzenia dostępnych serwisów internetowych. Wtyczka ta wykrywa i wskazuje niezgodności z wytycznymi w określonych kategoriach. Zdiagnozowane problemy

zgodności z WCAG mogą stanowić potencjale bariery w dostępności.



Rysunek 12: Okno prezentujące wyniki badania za pomocą narzędzia WAVE.

Na Rysunku 12 zaprezentowano widok otrzymany po przeanalizowaniu pięciu podstron danej witryny za pomocą automatycznego narzędzia. Pasek boczny widoczny po lewej stronie zawiera raport, z którego można odczytać informacje takie jak: liczba wykrytych błędów, liczba błędów związanych z występowaniem nieprawidłowego kontrastu, liczba ostrzeżeń – elementów, na które należy zwrócić uwagę, liczba poprawnie opisanych własności (atrybutów, które pozwalają wygenerować stronę zgodnie z dobrymi praktykami takimi jak np. określenie języka strony czy opisywanie obrazów tekstem alternatywnym) oraz liczba wykorzystanych elementów struktury języka HTML5 oraz atrybutów ARIA. W przypadku stron internetowych tworzonych ze szczególnym uwzględniam dostępności jedną z najważniejszych ról odgrywają atrybuty ARIA. Są one zbiorem specjalnych właściwości, które sprawiają iż strona jest łatwiejsza w korzystaniu dla osób z niepełnosprawnościami. Przykładowo do obsługi strony, która jest poprawnie opisana za pomocą tych własności powinien wystarczyć czytnik ekranowy, dzięki czemu osoby mające problem ze wzrokiem zapoznałyby się z jej zawartością i bez kłopotów mogłyby z niej korzystać.

Tabela 3. Wykryte nieprawidłowości z WCAG

Witryna	Średnia liczba wykrytych nieprawidłowości
WSIIZ	74,8
Prototypowa	3,4

W Tabeli 3 przedstawiono wyniki w postaci średniej liczby wykrytych niezgodności z zaleceniami występującymi na danej witrynie. Na liczbę nieprawidłowości składają się wszystkie błędy, błędy kontrastu oraz alerty. W przypadku badanych serwisów widać bardzo dużą różnicę między analizowanymi witrynami. Zdecydowanie mniej błędów wykryto w witrynie prototypowej. W witrynie przygotowanej do celów eksperymentu średnia liczba wszystkich nieprawidłowości w zakresie badanych podstron została zredukowana do poziomu średniego 3,4 błędów, co w porównaniu do liczby 74,8 wystąpień na stronach WSIIZ jest znaczącą poprawą.

6. Wnioski

Zastosowanie zasad projektowania uniwersalnego ma wpływ na jakość strony, a co się z tym wiąże na jej użyteczność i dostępność. W ramach pracy została stworzono alternatywna wersja serwisu internetowego Wyższej Szkoły Informatyki i Zarządzania. Następnie przeprowadzono eksperyment, który składał się z trzech etapów podczas których wykorzystano eyetracking, narzędzie WAVE do automatycznej analizy oraz gotowy kwestionariusz LUT. Badania skupiały się na użyteczności i dostępności, czyli przystosowaniu graficznego interfejsu do potrzeb osób niepełnosprawnych. Grupa 20 osób została podzielona na dwie części, które wykonywały ten sam zestaw poleceń, ale na innej witrynie: istniejącej lub prototypowej. Po badaniu z wykorzystaniem eyetrackera, uczestnicy wypełniali ankiety, w których oceniali daną stronę. W ostatnim etapie eksperymentu badacze posłużyli się narzędziem WAVE.

Uzyskane wyniki pokazują, że strona utworzona przez autorów została w większym stopniu przystosowana do standardu WCAG 2.1. W zakresie analizy jakościowej uzyskano duży materiał badawczy w postaci 20 map cieplnych dla każdego scenariusza i dwóch wersji witryn oraz 200 ścieżek skanowania. Na podstawie tych wyników można zauważyć, że badane osoby głównie koncentrowały się na prawidłowym wykonaniu polecenia. W badaniach ankietowych witryna autorska wypadła wyraźnie lepiej niż witryna istniejącej uczelni. Głównym mankamentem strony WSIIZ okazała się użyteczność interfejsu strony. Natomiast na witrynie prototypowej najgorzej wypadł element prezentujący treść, której zdaniem respondentów było za dużo. W badaniach za pomocą narzędzia WAVE, serwis WSIIZ również wypadł niekorzystnie. Wykazano, że posiada on bardzo dużą liczbę błędów i alertów. Pod tym względem witryna prototypowa wypadła zdecydowanie lepiej, gdyż podczas jej budowy zastosowano zasady projektowania uniwersalnego. Podsumowując, zastosowanie zasad projektowania uniwersalnego podczas tworzenia witryn www zwiększa ich poziom użyteczności i dostępności.

Literatura

[1] Ustawa z dnia 4 kwietnia 2019 r. o dostępności cyfrowej stron internetowych i aplikacji mobilnych podmiotów publicznych, http://orka.sejm.gov.pl/proc8.nsf/ustawy/3119_u.htm, [22.08.2022].

[2] Norma EN 301 549 V2.1.2, https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf, [01.09.2022].

[3] Dz.U. 2012 poz. 1169, Konwencji o prawach osób niepełnosprawnych, sporządzona w Nowym Jorku dnia 13 grudnia 2006 r., https://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=wdu20120001169, [01.09.2022].

[4] M. Skublewska-Paszkowska, M. Dzieńkowski, Przewodnik metodyczny „Projektowanie uniwersalne w informatyce" dla kierunku Informatyka, Lublin, 2021.

[5] The principles of universal design, NC State University, The Center for Universal Design, https://projects.ncsu.edu/ncsu/design/cud/about_ud/udprinciplestext.htm, [22.08.2022].

[6] Web Content Accessibility Guidelines (WCAG) 2.1, https://www.w3.org/Translations/WCAG21-pl/, [01.09.2022].

[7] K. Kałan, D. Karpiuk, M. Dzieńkowski, Usability analysis taking into consideration the aspects of accessibility of selected university websites. Journal of Computer Sciences Institute, 21 (2021) 295-302, https://doi.org/10.35784/jcsi.2725, [01.09.2022].

[8] K.A. Boothe, M. J. Lohmann, K. A. Donnell, D.D. Hall, Applying the principles of universal design for learning (UDL) in the college classroom, Journal of Special Education Apprenticeship 7/3 (2018), https://eric.ed.gov/?id=EJ1201588, [01.09.2022].

[9] A. Buczek, J. Sikorski, Dostępność stron internetowych wybranych instytucji użyteczności publicznej dla osób z niepełnosprawnością, Niepełnosprawność, 30 (2018) 281-294, https://doi.org/10.4467/25439561.NP.18.031.9869, [01.09.2022].

[10] E. Domagała-Zyśk, Projektowanie uniwersalne w edukacji osób z wadą słuchu, [w] Z problematyki teatrologii i pedagogiki, Wydawnictwo KUL, (2015) 553-568, http://hdl.handle.net/20.500.12153/3081, [01.09.2022].

[11] M. Fedorowicz-Kruszewska, M. Jarocki, Dostępność stron WWW polskich bibliotek uniwersyteckich dla osób z niepełnosprawnością wzroku - wyniki badań, Przegląd Biblioteczny, 78/4 (2010) 447-459, https://doi.org/10.36702/pb.416, [01.09.2022].

[12] A. Szewczyk, Seniorzy i osoby niepełnosprawne w społeczeństwie informacyjnym, Dostępność stron internetowych, Studia Informatica Pomerania 2/44 (2017) 43-61, http://dx.doi.org/10.18276/si.2017.44-05, [01.09.2022].

[13] D. Bednarczyk, Podstawowe reguły dostępności serwisów internetowych dla niepełnosprawnych użytkowników, Biuletyn EBIB 5/132 (2012) 1-9.

[14] A. Królak, Ocena wybranych platform do e-Learningu pod względem dostępności dla osób z niepełnosprawnościami i zgodności z zasadami projektowania uniwersalnego, Ekonomia – Wroclaw Economic Review, 23/1 (2017) 45-58, https://doi.org/10.19195/2084-4093.23.1.5, [01.09.2022].

[15] Gazepoint, https://www.gazept.com/product/gp3hd/, [01.09.2022].

[16] iMotions, https://imotions.com/, [25.08.2022].

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

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 25 (2022) 409–416
Received: 13 September 2022
Accepted: 29 September 2022

Swift performance in statistical aplications

Wydajność języka Swift w zastosowaniach statystycznych

Sylwester Tylec*, Karol Woś

Abstract
This paper aims to test Swift's performance against C++ in performing statistical calculations. The analyzed issues, apart from performance, are code transparency and syntax, libraries available for these languages and the use of device hardware resources during testing. For this purpose, a comparative analysis of the two above-mentioned languages was carried out, based on the results obtained from a series of experiments carried out with the use of specially developed test applications. The tests consisted in calculating the standard deviation, median and arithmetic, harmonic and geometric mean, and during the tests, execution times, operating memory usage and CPU load were recorded. Based on the results of the research, it was found that the Swift language is not optimized for statistical calculations.

Keywords: Swift; statistics; performance; analysis

Streszczenie
Niniejsza praca ma na celu przeprowadzenie testów wydajności języka Swift w porównaniu z językiem C++ przy wykonywaniu obliczeń statystycznych. Analizowanymi zagadnieniami poza wydajnością są przejrzystość i składnia kodu, biblioteki dostępne dla tych języków oraz wykorzystanie zasobów sprzętowych urządzenia podczas przeprowadzania testów. W tym celu przeprowadzono analizę porównawczą dwóch wyżej wymienionych języków, opierającą się na wynikach uzyskanych z serii eksperymentów przeprowadzonych przy użyciu specjalnie utworzonych aplikacji testowych. Testy polegały na liczeniu odchylenia standardowego, mediany i średnich arytmetycznej, harmonicznej i geometrycznej a w trakcie testów rejestrowano czasy wykonania, użycie pamięci operacyjnej i obciążenie procesora. Na podstawie wyników badań ustalono, że język Swift nie jest zoptymalizowany pod kątem obliczeń statystycznych.

Słowa kluczowe: Swift; statystyka; wydajność; analiza

*Corresponding author
Email address: **sylwester.tylec@pollub.edu.pl** (S. Tylec)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Programy komputerowe pomagają użytkownikom wykonywanie wielu czynności, takich jak obliczenia, symulacje, zarządzanie danymi, obróbka grafiki i wiele innych. Jednak, aby program jak najwydajniej pełnił swoją rolę, musi zostać napisany przy pomocy odpowiedniego i najbardziej adekwatnego do powierzonego zadania języka z wykorzystaniem jego bibliotek i funkcji. Mnogość języków wraz z dostępne dla nich narzędzia mogą przysparzać trudności podczas ich wyboru. Pod uwagę należy wziąć zadania, jakie zostaną podstawione przed programem.

Język Swift świetnie nadaje się do przeprowadzenia analizy ze względu na obiektowość, która jest obecnie najpopularniejszym podejściem podczas tworzenia aplikacji. Dodatkowym atutem przy uwzględnieniu tego języka jest fakt, że jest on stosowany zarówno na urządzeniach mobilnych, jak i desktopowych firmy Apple. Język C++ jest językiem znacznie starszym, gdyż powstał w roku 1985. Niemniej jednak w dalszym ciągu cieszy się dużą popularnością ze względu na dużą szybkość i wydajność, a wiele frameworków i narzędzi tworzonych oraz używanych w dzisiejszych czasach bazuje na tymże języku.

Znaczna część użytkowników sprzętu komputerowego jest w posiadaniu co najmniej dwóch urządzeń mogących służyć wykonywaniu obliczeń. Rosnąca popularność rozwiązań mobilnych sprawia, że coraz większą liczbę działań można przeprowadzić na sprzęcie przenośnym, który zazwyczaj nie dysponuje tak dużą mocą obliczeniową, jak urządzenia stacjonarne. Warto jest więc zastanowić się, jaki sprzęt i jakie narzędzia należy wybrać, aby jak najlepiej spełniały oczekiwania względem wydajności. Przeprowadzenie porównania i analizy wydajnościowej wymaga doboru języka, który z założenia może zostać wykorzystywany na takich samych płaszczyznach.

Niniejsza praca została poświęcona analizie porównawczej języków C++ i Swift. Porównanie będzie obejmować równoległe wykonywanie obliczeń statystycznych na tych samych danych źródłowych, co pozwoli na zbiorcze zestawienie wyników przeprowadzonych badań. Wśród nich zawierać się będą czasy przeprowadzenia operacji oraz obciążenie systemu podczas działania programu.

2. Cel badań

Niniejszy artykuł ma na celu przeprowadzenie analizy wydajności języka Swift pod kątem obliczeń statystycznych. W tym celu dokonano porównania analogicznych programów stworzonych w języku Swift i C++. Głównymi parametrami, które były brane pod uwagę przy analizie porównawczej były obciążenie procesora, zużycie pamięci RAM i czas wykonania programów. Pozostałymi badanymi aspektami podczas porównywania były budowa i składnia języka, dostępność bibliotek oraz popularność języków wśród społeczności.

3. Metoda badawcza

W celu przeprowadzenia analizy wydajności autorzy niniejszej pracy badawczej utworzyli trzy aplikacje różniące się stopniem złożoności. Zostały one zaimplementowane dla obu badanych języków. Aplikacje wykorzystują te same algorytmy i zostały napisane w możliwie identyczny sposób. Pierwsza z nich ma za zadanie obliczyć odchylenie standardowe, druga natomiast sortuje tablicę z danymi w celu znalezienia mediany, trzecia oblicza średnią arytmetyczną, geometryczną i harmoniczną. Podczas wykonywania badań zostały użyte te same zestawy danych, które były wygenerowane losowo w liczbie odpowiednio 100, 1000 i 10000. Łącznie przeprowadzono 82 pomiary, przy czym dla każdego zestawu danych w przypadku aplikacji na oba języki przeprowadzono po 7 pomiarów.

Badania odbywały się na urządzeniu testowym, którego specyfikacja przedstawiona jest w Tabeli 1. Oba programy zostały napisane i uruchomione w środowisku Xcode w wersji 13.2.1.

Tabela 1: Specyfikacja urządzenia testowego

Model urządzenia	MacBook Pro MYD92ZE/A/R1
Procesor	Apple M1
Pamięć RAM	16 GB
System operacyjny	macOS Monterey
Dysk	512 GB SSD PCIe
Rok premiery	2020

Na urządzeniu zostały wyłączone wszelkie niepotrzebne procesy oraz było ono podłączone do źródła zasilania, aby na czas przeprowadzenia pomiarów zasoby sprzętowe nie były niczym ograniczone, co zapewniło powtarzalne warunki testowe.

Czas wykonania programów mierzony był przy użyciu dostępnych w standardowych bibliotekach funkcji, które umożliwiają precyzyjne odmierzanie czasu. Dla języka Swift zastosowano funkcję „CFAbsoluteTimeCurrent()", natomiast w języku C++ była to funkcja „chrono::steady_clock::now()". Pozostałe parametry, czyli obciążenie wszystkich rdzeni procesora oraz zużycie pamięci operacyjnej mierzone były za pomocą wbudowanego w system operacyjny monitora aktywności. Podczas badania wydajności pod uwagę były brane najwyższe wartości dotyczące wykorzystania zasobów sprzętowych dla procesu odpowiadającego za dany program, z których następnie wyciągnięto średnią z pomiarów.

Strukturę kodu i składnię języka porównano posiłkując się oficjalną dokumentacją dostępną dla obu języków. Według autorów odpowiednim kryterium do zbadania popularności wśród społeczności była liczba wyników zapytań odnoszących się do danego języka w serwisach GitHub oraz Stack Overflow.

Algorytmy użyte w programach wykorzystanych w badaniach były napisane w najprostszy możliwy sposób w celu zmniejszenia liczby operacji. Fragmenty aplikacji zostały przedstawione na Listingach 1, 2, 3, 4, 5 oraz 6.

Listing 1: Fragment aplikacji liczącej średnie w języku C++

```cpp
double MeanA(double data[]) {
    double sum = 0.0, meanA;
    int i;
    for(i = 0; i < 10; ++i) {
    sum += data[i];
    }
    meanA = sum / 10;
    return meanA;
}
double MeanG(double data[]){
    double iloczyn = 1.0, meanG=0;
    int i;
    for(i = 0; i < 10; ++i) {
    iloczyn *= data[i];
    }
    meanG=pow(iloczyn, 1.0/10);
    return meanG;
}
double MeanH(double data[]) {
    double sum = 0.0, meanH;
    int i;
    for(i = 0; i < 10; ++i) {
    sum += 1/data[i];
    }
    meanH = 10/sum;
    return meanH;
}
```

Listing 2: Fragment aplikacji liczącej średnie w języku Swift

```swift
func meanA(arr: [Double])->Double{
    var suma, mean: Double
    mean=0
    suma=0
    for i in 0...n{
        suma+=arr[i]
    }
    mean=suma/g
    return mean
}
func meanG(arr: [Double])->Double{
    var iloczyn, meanG: Double
    meanG=0
    iloczyn=1
    for i in 0...n{
        iloczyn*=arr[i]
    }
    meanG=pow(iloczyn, (1/g))
    return meanG
}
func meanH(arr: [Double])->Double{
    var suma, meanH: Double
    meanH=0
    suma=0
    for i in 0...n{
        suma+=1/arr[i]
    }
    meanH=g/suma
    return meanH
}
```

Listing 3: Fragment aplikacji liczącej odchylenie standardowe w języku Swift

```
23      var suma, mean, standardDeviation: Double
24      standardDeviation=0
25      mean=0
26      suma=0
27      for i in 0...n{
28          suma+=arr[i]
29      }
30      mean=suma/g
31      for i in 0...n{
32          standardDeviation+=pow(arr[i]-mean, 2)
33      }
```

Listing 4: Fragment aplikacji liczącej odchylenie standardowe w języku C++

```
24  double calculateSD(double data[]) {
25    double sum = 0.0, mean, standardDeviation = 0.0
26    int i;
27
28    for(i = 0; i < 10; ++i) {
29      sum += data[i];
30    }
31
32    mean = sum / 10;
33
34    for(i = 0; i < 10; ++i) {
35      standardDeviation += pow(data[i] - mean, 2);
36    }
```

Listing 5: Fragment aplikacji liczącej medianę w języku Swift

```
13  func calculateMedian(arr: [Double]) -> Double {
14  let sorted = arr.sorted()
15  if sorted.count % 2 == 0 {
16  return Double((sorted[(sorted.count / 2)]
17      + sorted[(sorted.count / 2) - 1])) / 2
18  } else {
19  return Double(sorted[(sorted.count - 1) / 2])
20  }
21  }
```

Listing 6: Fragment aplikacji liczącej medianę w języku Swift

```
26  double mediana (double data[])
27  {
28      double mediana;
29      int n=10;
30      if (n%2 == 0)
31      {
32          mediana = data[(n-1)/2]+data[n/2];
33          mediana = mediana/2;
34      }
35      else
36      {
37          mediana = data [n/2];
38      }
39      return mediana;
40  }
```

4. Przegląd literatury

W celu dokonania analizy wydajności języka Swift przeprowadzono przegląd literatury dotyczący powiązanych zagadnień. Zważywszy na to, iż istnieje niewiele prac, w których autorzy porównują bezpośrednio język Swift z C++, użyta do napisania niniejszego artykułu literatura dotyczy zagadnień związanych z porównywaniem dwóch języków, dzięki czemu można było dobrać odpowiednie metody badawcze.

Istotnym źródłem wiedzy użytecznej podczas pisania tej pracy były oficjalne dokumentacje techniczne używanych języków [1, 2]. Znajdują się w nich wszelkie niezbędne informacje opisujące budowę języka, podstawowe biblioteki i instrukcje dla programistów, na podstawie których można zestawić ze sobą oba języki pod kątem składni, stopnia skomplikowania, występowania typów zmiennych i intuicyjności pisania kodu.

Posługując się wiedzą zawartą w książkach do nauki poszczególnych języków w naszym przypadku są to: „Swift. 4 Koduj jak mistrz" autorstwa J. Hoffmana [3] i „Język C++. Szkoła programowania" autorstwa S. Prata [4]. Mając na uwadze zróżnicowane poziomy trudności w nauce tych dwóch języków należy zastanowić się, który będzie lepszym wyborem w przypadku ukierunkowania na obliczenia statystyczne. Książki te przydatne były do analizowania składni kodu i praktyk stosowanych podczas tworzenia programów w tych językach.

Autorom niniejszej pracy nie udało się znaleźć publikacji poruszającej temat wykorzystania języka Swift w statystyce. Prace, które przykuły uwagę zajmowały się analizą i porównaniem innych języków, a ich konkluzją było przedstawienie ich zalet i wad. Pracami, w których autorzy porównywali ze sobą dwa zagadnienia istotnym kryterium był czas wykonywania i analiza kodu. „Analiza porównawcza wydajności frameworków Angular oraz Vue.js" [5], której autorami są R. Baida, M. Andriienko i M. Plechawska-Wójcik w dużym stopniu skupia się na pomiarze czasu działania dwóch aplikacji internetowych. Autorzy badali czas wysyłania żądań do serwera i czas renderowania stron www, a w wynikach przedstawili oni silne i słabe strony porównywanych rozwiązań.

Pracami o podobnej tematyce są „A Comparative Study: Java vs Kotlin Programming in Android Application Development" [6] oraz „Comparative Analysis of Python and Java for Beginners" [7]. Jedynym artykułem, który w jakiś sposób omawiał język Swift jest praca „Speed Performance Between Swift and Objective-C" autorstwa H. Singha [8]. Autor porównywał język Swift z jego prekursorem, czyli Objecive-C. Autor na podstawie porównania do innych języków wskazał zalety języka Swift i zaprezentował go jako nowy, intuicyjny i przyjazny programiście język, który ma z powodzeniem zastąpić objective-C.

Praca „Analiza porównawcza języków Kotlin i Java używanych do tworzenia aplikacji na system Android" autorstwa D. Sulowskiego i G. Kozieła opublikowana w grudniu 2019 traktuje o podobnym zagadnieniu [9]. Autorzy porównywali dwa języki programowania pod kątem ich wydajności i użyteczności. Pod uwagę brane były takie kryteria, jak obciążanie procesora, użycie pamięci RAM oraz czasy wykonania programów.

Są to typowe prace, w których został poruszony temat porównania dwóch bezpośrednio konkurujących ze sobą aspektów w dziedzinie informatyki. Autorzy w swoich pracach badali wykorzystanie zasobów przez

badane technologie, mierzyli czas potrzebny na wykonanie czynności będących celem badań oraz oceniali praktyczność i możliwości zastosowań tychże technologii. Zatem mogą stanowić ona dobry przykład na to, jak należy przeprowadzić podobne porównanie i w jaki sposób zinterpretować wyniki otrzymane podczas przeprowadzania badań oraz postawić odpowiednią hipotezę badawczą.

5. Porównanie cech badanych języków

5.1. Różnice w budowie języków

Pierwszą wyraźną różnicą jest inna składania języków. Jak każdy język programowania Swift i C++ są w jakimś stopniu do siebie podobne, jednak należy zauważyć różnice w sposobie deklaracji zmiennych, budowania prototypów funkcji, wywoływania funkcji w programie, operacjach na tablicach. W badanych programach różnice w budowie kodu są niewielkie i zaawansowane opanowanie tworzenia programów w języku C++ pozwala w miarę łatwo i intuicyjnie posługiwać się językiem Swift.

Najważniejsze różnice w składni porównywanych języków:

- język Swift charakteryzuje się bezwzględną przestrzenią nazw automatycznie obejmującą wszystkie typy zmiennych,
- W języku C++ trzeba zawsze zadeklarować przestrzeń nazw, z której pochodzi dana metoda. Uproszczona wersja została zaprezentowana na Listingu 7.

Listing 7: Deklaracja przestrzeni nazw w języku C++

```
5  using namespace std;
```

- W języku Swift trzeba zadeklarować, czy dana będzie stała, czy zmienna. Stałą wartość oznacza się deklaracją „let" a zmienną „var". Inicjalizowanie stałej w języku Swift należy przeprowadzić od razu, w przeciwnym razie kompilator będzie proponował zmianę użytkownikowi. Dodatkowo po znaku dwukropka należy przypisać typ danej. Przykład został pokazany na Listingu 8,
- W języku C++ domyślnie wszystkie dane są traktowane jako zmienne. Stałe dane trzeba wyraźnie zadeklarować. Na Listingu 9 pokazano deklarację typów danych w języku C++.

Listing 8: Deklaracja typów danych w języku Swift

```
8  var a: Int
9  let b: Double = 1
```

Listing 9: Deklaracja typów danych w języku C++

```
8  int a;
9  const int b=9;
```

- Deklaracja tablic nie różni się od siebie w znaczący sposób. W przypadku obu języków przy deklaracji typu danych należy umieścić znak kwadratowych nawiasów. Na Listingu 10 zaprezentowano przykład deklaracji tablicy w języku Swift,
- W przypadku języka Swift zawartość tablicy umieszcza się w kwadratowych nawiasach, natomiast w języku C++ w nawiasach klamrowych. Na Listingu 11 zaprezentowano przykład deklaracji tablicy w języku C++.

Listing 10: Deklaracja tablicy w języku Swift

```
12  let tablica: [Double]
13  tablica=[1,2,3,4,5]
```

Listing 11: Deklaracja tablicy w języku C++

```
17  double data[]={1,2,3,4,5};
```

- Badane języki używają bardzo zbliżonych deklaracji pętli For. W przypadku języka Swift operacji na iteratorze nie umieszcza się w nawiasach, operacje inkrementacji i dekrementacji wykonują się automatycznie w ustalonym przedziale. Deklaracja pętli For w języku Swift przedstawiona jest na Listingu 12,
- Zmienna iterowana w pętli for w języku C++ może być zadeklarowana wewnątrz pętli jak i przed nią, natomiast w języku Swift nie można umieścić deklaracji wewnątrz pętli, trzeba używać już istniejącej zmiennej. Deklaracja pętli For w języku C++ przedstawiona jest na Listingu 13.

Listing 12: Deklaracja pętli For w języku Swift

```
17      for i in 0...n{
18          suma+=arr[i]
19      }
```

Listing 13: Deklaracja pętli For w języku C++

```
27    for(i = 0; i < 10; ++i) {
28      sum += data[i];
29    }
```

- Deklarację funkcji w języku Swift należy zacząć od słowa „func". Tak jak w większości innych języków argumenty funkcji znajdują się po jej nazwie w nawiasach okrągłych. Typ zmiennej zwracanej przez funkcję umieszcza się po strzałce. Przykład deklaracji funkcji w języku Swift przedstawiony został na Listingu 14,
- Deklaracja funkcji w języku C++ rozpoczyna się od przypisaniu typu danej zwracanej przez tą funkcję, po której znajduje się nazwa funkcji. W okrągłych nawiasach podobnie jak w przypadku języka Swift umieszcza się argumenty funkcji. Przykład deklaracji funkcji w języku Swift przedstawiony został na Listingu 15.

Listing 14: Deklaracja funkcji w języku Swift

```
13  func calculateSD(arr: [Double])->Double{
```

Listing 15: Deklaracja funkcji w języku C++

```
22  double calculateSD(double data[]) {
```

- Deklaracje klas i struktur w obu porównywanych językach nie różnią się znacząco od siebie. W obu przypadkach deklarację poprzedza słowo kluczowe „struct” lub „class”. Zawartość stanowić mogą zmienne różnego typu z odpowiednimi specyfikatorami dostępu. W swifcie domyślnym specyfikatorem, czyli takim, który ustawia się automatycznie w przypadku braku przyznania wartości przez użytkownika jest Internal access. Zezwala on na dostęp do danego pola w obrębie całego projektu, czyli z poziomu pozostałych klas i funkcji. Domyślnym specyfikatorem w C++ jest private, co sprawia, że do tak oznaczonego pola nie ma dostępu spoza klasy. Definicje metod i konstruktorów nie różnią się od siebie, a do poszczególnych pól klas i struktur można odwołać się przez znak kropki następujący po utworzonym obiekcie.

5.2. Popularność wśród społeczności

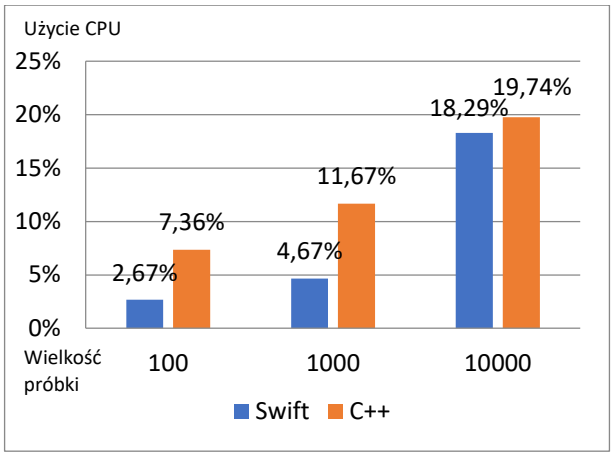
Na podstawie zapytań o język Swift na popularnych platformach dostępnych w Internecie w czerwcu 2022 roku uzyskano około 32 miliony wyników w wyszukiwarce Google, ponad 314 tysięcy pytań o język Swift na stronie Stack Overflow oraz 241 tysięcy wyszukań repozytoriów na portalu GitHub. Z kolei dla języka C++ jest to kolejno około 105 milionów, 768 tysięcy i 911 tysięcy. Świadczy to o dużo większym zainteresowaniu i wsparciu społeczności dla języka C++. Powodem może być wiek i ugruntowana pozycja na rynku. C++ to język o wielu zastosowaniach na różnych platformach, w odróżnieniu od języka Swift, będącego językiem stosunkowo młodym i używanym w znakomitej większości na urządzeniach firmy Apple.

6. Wyniki

Na podstawie wyników przeprowadzonych badań obliczono średnie wartości dla pomiarów poszczególnych parametrów. Odpowiednio dla każdego zestawu danych liczących 100, 1000 i 10000 wartości przedstawione zostały uśrednione wyniki z przeprowadzonych pomiarów. Jeden z programów miał za zadanie obliczyć odchylenie standardowe, natomiast drugi wyznaczał medianę danego zbioru liczb.

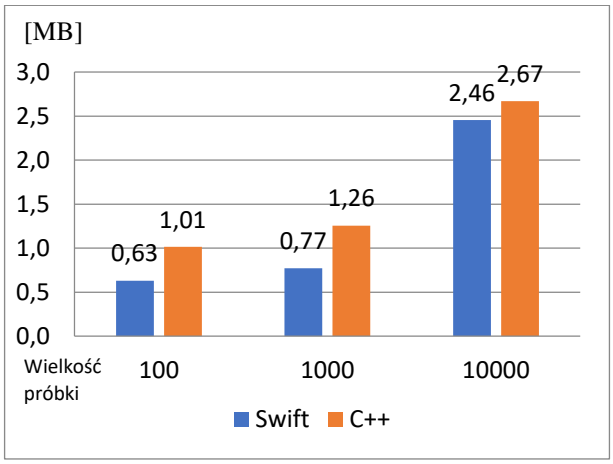
6.1. Seria pomiarowa 1

Pierwszy z omawianych programów miał za zadanie wyliczyć odchylenie standardowe dla każdego zestawu danych wprowadzonych statycznie do tablicy. Na Rysunku 1 przedstawione zostało średnie obciążenie procesora. Wyraźnie widać różnicę w wykorzystanej mocy obliczeniowej dla dwóch mniejszych zestawów danych, natomiast przy największym zestawie danych ta różnica jest znacząco mniejsza. Należy więc zauważyć, że wraz ze wzrostem ilości danych użycie procesora w przypadku obu badanych programów przedstawia się na podobnym poziomie, jednak w każdym z pomiarów język Swift wykazuje się mniejszym zapotrzebowaniem na moc obliczeniową procesora.



Rysunek 1: Obciążenie procesora wyrażone w %.

Na Rysunku 2 przedstawione zostało średnie zużycie pamięci RAM wyrażone w megabajtach. Swift również pod tym względem cechuje się mniejszym zapotrzebowaniem na zasoby sprzętowe. Różnica w wykorzystaniu pamięci między zastosowanymi językami jednak nie jest tak znacząca jak w przypadku zużycia procesora, a dla największego zestawu danych jest prawie niezauważalna. Wraz ze wzrostem ilości danych wpływ zastosowanego języka pod względem zużycia zasobów pamięci operacyjnej maleje.



Rysunek 2: Wykorzystanie pamięci RAM wyrażone w MB.

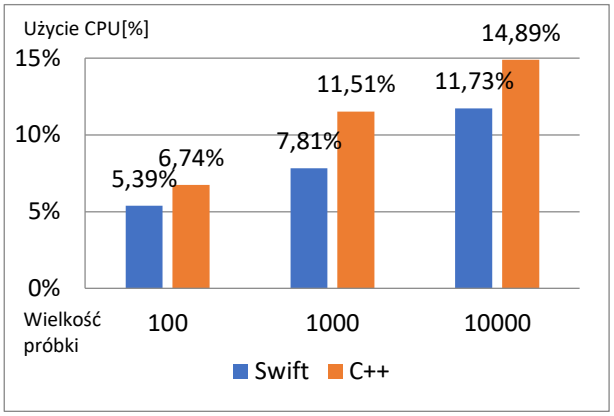
W Tabeli 2 przedstawione zostały czasy wykonania programu wyrażone w mikrosekundach. W przypadku czasów wykonania język Swift potrzebował znacząco więcej czasu w porównaniu do języka C++. Na podstawie tabeli można zauważyć, że wraz ze wzrostem liczby próbek różnica w czasie pomiędzy badanymi językami zmniejsza się.

Tabela 2: Czas wykonania programu wyrażony w mikrosekundach

Liczba próbek	Swift[µs]	C++[µs]
100	251,43	2,59
1000	672,86	22,74
10000	4885,71	234,12

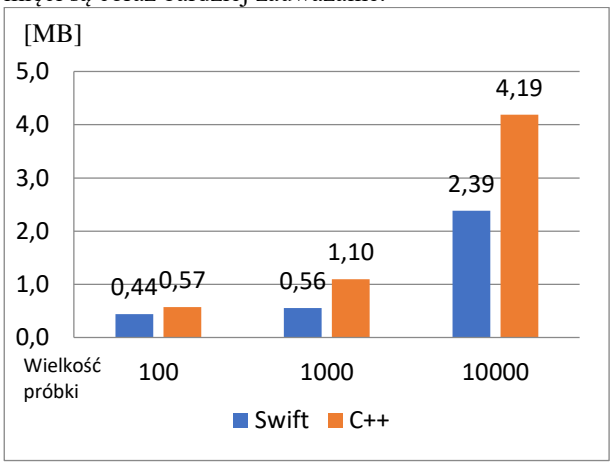
6.2. Seria pomiarowa 2

Drugi z badanych programów służył do obliczenia mediany i wykorzystywał zestawy danych, które użyte były przy pierwszym programie. Użycie mocy obliczeniowej procesora podczas działania drugiego programu przedstawione zostało na Rysunku 3. Różnice w wykorzystaniu zasobów procesora dla tej serii pomiarowej dla każdego z przypadków są mniejsze niż dla serii poprzednich pomiarów. Dla pierwszego zestawu danych średnia różnica w wykorzystaniu procesora nie przekracza jednego procenta, natomiast dla dwóch pozostałych badanych zestawów różnica ta jest nieznacznie większa, jednak nie przekracza ona czterech procent. Należy zauważyć, że w przypadku drugiej serii pomiarowej użycie procesora dla obu badanych programów przedstawia się na podobnym poziomie, jednak język Swift potrzebuje mniej zasobów sprzętowych.



Rysunek 3: Obciążenie procesora wyrażone w %.

Na Rysunku 4 przedstawione zostało średnie zużycie pamięci RAM wyrażone w megabajtach. Dla najmniejszego zestawu danych różnica w wykorzystaniu pamięci jest wręcz pomijalna, jednak dla pozostałych przypadków jest niemal dwukrotnie większa na korzyść języka Swift. Zauważyć należy tendencję odwrotną niż w przypadku pierwszej serii pomiarowej, gdyż wraz ze wzrostem ilości danych, różnice w wykorzystaniu pamięci są coraz bardziej zauważalne.



Rysunek 4: Wykorzystanie pamięci RAM wyrażone w MB.

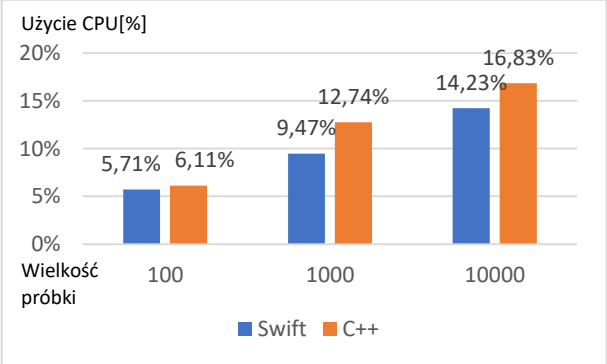
W Tabeli 3 przedstawione zostały czasy wykonania programu wyrażone w mikrosekundach. W przypadku języka Swift czas wykonania programu rośnie liniowo, natomiast dla języka C++ różnica w czasie dla zestawu 100 i 1000 wartości wynosi około dwa i pół, a dla próbek 1000 i 10000 wynosi ponad sześć.

Tabela 3: Czas wykonania programu w mikrosekundach

Liczba próbek	Swift[µs]	C++[µs]
100	592,86	20,25
1000	2872,86	48,98
10000	18300	303,46

6.3. Seria pomiarowa 3

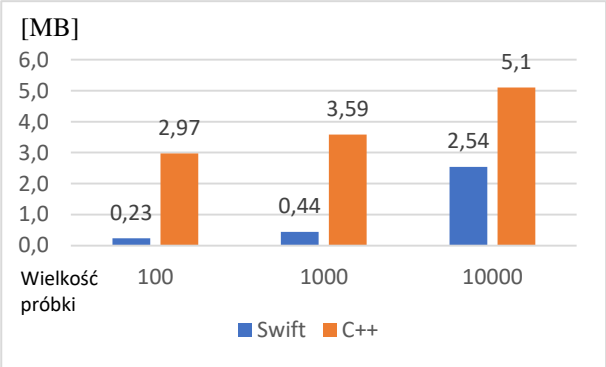
Program liczący średnie dla najmniejszej próbki liczącej 100 danych losowych potrzebuje prawie takiej samej mocy obliczeniowej procesora dla programów zarówno napisanych w języku Swift jak i C++, różnica wynosi zaledwie 0,4%. Dla dwóch większych zestawów danych różnice te wynoszą 3,27% oraz 2,6% odpowiednio dla próbek liczących 1000 i 10000 danych. Również w tym programie język Swift potrzebował mniejszej mocy obliczeniowej jednak różnica ta była o wiele mniejsza niż w przypadku dwóch poprzednich programów, mogła na to wpłynąć liczba obliczeń jakie były do wykonania, Program ten miał w sobie trzy funkcje, po jednej dla każdej z liczonych średnich i każda z nich zawierała pętlę, która musiała wykonać pewne działania arytmetyczne na wszystkich elementach tablicy. Na Rysunku 5 przedstawiono procentowe użycie procesora.



Rysunek 5: Obciążenie procesora wyrażone w %.

Podczas wykonywania programu liczącego średnie język C++ potrzebował znacznie większej ilości pamięci operacyjnej. Dal najmniejszej próbki liczącej zaledwie 100 danych język Swift potrzebował 0,23MB pamięci natomiast język C++ aż 2,97MB, różnica ta jest prawie trzynastokrotna. Druga próbka licząca 1000 obciążyła program na tyle, że potrzebował on średnio 3,59MB pamięci RAM w przypadku języka C++ 0,44MB w przypadku języka Swift, różnica nie jest aż tak duża jak dla pierwszej próbki, jednakże wciąż jest to ponad ośmiokrotność zapotrzebowania przez język Swift. Dla największej próbki zapotrzebowanie na pamięć operacyjną wynosiło 2,54MB w przypadku języka Swift oraz 5,1MB w przypadku języka C++, co oznacza, że C++ potrzebuje ponad dwa razy więcej pamięci do wykonania obliczeń. Tak rozbieżne wyniki dal obu języków mogą być spowodowane ilością obliczeń jakie

wykonuje program do policzenia trzech średnich. Na Rysunku 6 przedstawione zostały średnie użycie pamięci operacyjnej.



Rysunek 6: Wykorzystanie pamięci RAM wyrażone w MB.

W Tabeli 4 przedstawiono czasy wykonania programu w mikrosekundach. W przypadku pierwszej próbki liczącej zaledwie 100 danych różnica jest ponad stuosiemdziesięciokrotna. Program napisany w języku Swift potrzebuje średnio 281,19µs na wykonanie, natomiast program napisany w języku C++ potrzebuje zaledwie 1,51µs. Dla drugiej próbki liczącej 1000 danych różnica ta zmniejsza się i wynosi niewiele ponad 80 razy więcej, co i tak jest bardzo dużą różnicą, czasy jakich potrzebują programy do wykonania się przy tej próbce to 900,80µs w przypadku języka Swift oraz 11,18µs w przypadku języka C++. Największa próbka licząca 10000 danych obciążyła program w dużym stopniu, język Swift potrzebuje średnio 7368,57µs na wykonanie programu, a języka C++ 115,96µs co stanowi wartość ponad 60 razy mniejszą.

Tabela 4: Czas wykonania programu w mikrosekundach

Liczba próbek	Swift[µs]	C++[µs]
100	281,19	1,51
1000	900,80	11,18
10000	7368,57	115,96

7. Wnioski

W artykule dokonano analizy wydajności języka Swift w porównaniu do języka C++ podczas wykonywania obliczeń statystycznych. Na podstawie przeprowadzonych badań i zestawieniu wyników można zauważyć, iż programy napisane w języku Swift charakteryzują się niższym zapotrzebowaniem na moc obliczeniową, gdyż w przypadku obu badanych programów zużycie procesora i wykorzystanie pamięci operacyjnej było niższe.

Przy pierwszej serii pomiarowej badany program liczył odchylenie standardowe. Dla mniejszych zestawów danych jest różnica na korzyść języka Swift, a dla większych wykorzystanie procesora w przypadku dwóch języków było prawie takie samo. Wykorzystanie pamięci dla obu języków przedstawia się na zbliżonym poziomie, a dla największego z zestawów danych różnica w zapotrzebowaniu jest niemal pomijalna. Aplikacja napisana w języku Swift potrzebuje znacznie więcej czasu na wykonanie obliczeń, a zwiększenie rozmiaru danych, na których wykonywane są obliczenia zmniejsza różnicę w czasie wykonania.

Dla drugiej serii pomiarowej program obliczał medianę zestawu danych. Jest to zadanie bardziej skomplikowane niż w przypadku pierwszej serii, gdyż wszystkie liczby muszą zostać posortowane, co wprowadza konieczność wykonania większej liczby obliczeń. Wybór języka nie wpływa w znacznym stopniu na obciążenie procesora, jednak dla każdego zbioru liczb zauważyć należy przewagę języka Swift. Różnica w zużyciu pamięci operacyjnej nie wydaje się zależeć od rozmiaru zestawu danych, natomiast ponownie język Swift charakteryzuje się mniejszym wykorzystaniem zasobów urządzenia.

Trzecia seria pomiarowa przeprowadzona została dla programu liczącego zestaw trzech średnich. Jest to program o największej złożoności, składający się z trzech, niezależnie działających funkcji i zwracający wyniki trzech obliczeń. Procesor bez względu na zastosowany język był obciążony w bardzo podobnym stopniu, przy czym program napisany w języku Swift charakteryzował się nieznacznie mniejszym zapotrzebowaniem. Dużo większe różnice zaobserwować można pod kątem wykorzystanej pamięci operacyjnej. Dla dwóch mniejszych zestawów użycie pamięci było około 10 razy niższe, a dla największego z zestawów program napisany w języku Swift potrzebował około dwukrotnie mniej pamięci. Tak jak w przypadku dwóch poprzednich programów czas wykonania dla języka Swift był znacząco większy, jednak różnica w ilości danych wejściowych nie jest wprost proporcjonalna do czasu wykonania programu.

Programy napisane w języku Swift charakteryzują się mniejszymi wymaganiami sprzętowymi ze względu na ukierunkowanie na działanie pod kontrolą jednego systemu operacyjnego, który oficjalnie wspierany jest w znakomitej większości na urządzeniach firmy Apple. Jednak wraz ze wzrostem poziomu skomplikowania projektu przewaga polegającą na mniejszym zużyciu zasobów sprzętowych języka Swift zaczyna się zacierać, ponieważ wykorzystana moc obliczeniowa procesora, bądź zużycie pamięci zaczyna wzrastać do poziomów zbliżonych podczas działania programów napisanych w języku C++. Czas wykonania aplikacji jednoznacznie pokazuje, że język C++ jest wielokrotnie szybszy, a autorom nie udało się wskazać przewagi języka Swift na tym polu.

Ze względu na fakt, iż Swift jest dość młodym językiem istnieje możliwość, iż będzie on stale udoskonalony w kolejnych wersjach. C++ jest łatwiejszy, jeśli chodzi o poziom trudności podczas nauki oraz bardziej dostępny ze względu na to, że jest szeroko stosowany na najpopularniejszych systemach operacyjnych obsługiwanych przez urządzenia desktopowe. Języka Swift również można używać na większości urządzeń natomiast nie wszystkie są tak dobrze przystosowane do tego jak urządzenia firmy Apple. Ponadto duża dostępność bibliotek, gotowych programów, wpisów na forach sprawia, że jest on dobrym wyborem dla początkujących programistów. Jednak niewykluczone, że

w przyszłości rozwój języka Swift i zapotrzebowanie na aplikacje dedykowane konkretnemu środowisku będą na tyle duże, że popularność tego języka wzrośnie. Podczas wyboru języka do obliczeń statystycznych należy wziąć pod uwagę dostępny sprzęt oraz potrzeby użytkownika, dla słabszych urządzeń lepszym wyborem okazuje się język Swift, natomiast dysponując bardziej wydajną konfiguracją sprzętową wybór ten może być zależny wyłącznie od osobistych preferencji programisty.

Literatura

[1] Oficjalna dokumentacja języka Swift, https://www.swift.org/documentation, [15.06.2022].

[2] Oficjalna dokumentacja języka C++, https://docs.microsoft.com/pl-pl/cpp/cpp/?view=msvc-170, [15.06.2022].

[3] J. Hoffman, Swift 4. Koduj jak mistrz, Helion, Gliwice, 2018.

[4] S. Prata, Język programowania C++. Szkoła programowania, Helion, Gliwice, 2012.

[5] R. Baida, M. Andriienko, M. Plechawska-Wójcik, Analiza porównawcza wydajności frameworków Angular oraz Vue.js, Journal of Computer Sciences Institute 14 (2020) 59-64, https://doi.org/10.35784/jcsi.1577.

[6] S. Bose, A Comparative Study: Java vs Kotlin Programming in Android Application Development, International Journal of Advanced Research in Computer Science 9(3) (2018) 41-45.

[7] S. Khoirom, S. Moirangthem, B. Laikhuram, J. Laishram, T. D. Singh, Comparative Analysis of Python and Java for Beginners, Int. Res. J. Eng. Technol 7(8) (2020) 4384-4407.

[8] H. Singh, Speed Performance Between Swift and Objective-C, Int. J. Eng. Appl. Sci. Technol 1(10) (2016) 185-189.

[9] D. Sulowski, G. Kozieł, Comparative Analysis of Kotlin and Java languages Used to Create Applications for the Android System, Journal of Computer Sciences Institute 13 (2019) 354-358, https://doi.org/10.35784/jcsi.1332.