JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 16 (2020) 293–299
Received: 23 June 2020
Accepted: 7 July 2020

Analysis of the use of Java and C# languages for building a mobile application for the Android platform

Analiza zastosowania języków Java oraz C# do budowy aplikacji mobilnej na platformę Android

Michał Jankowski*, Maria Skublewska-Paszkowska

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

Abstract
Mobile applications for the Android platform can be implemented using Java or C#. The article presents a comparison of the time performance of these languages when sending various text, image and video files in a mobile application. The tests were carried out using two mobile applications with identical functionalities. Based on the collected data, the server application calculated statistics, such as, for example, the time required to send 1 MB of data depending on the file type and size. Based on the results obtained, it was proved that in the case of data transfer via a wireless network, an application written in Java is characterized by greater time efficiency than an analogous application written in C#.

Keywords: mobile applications; REST API; Java; C#

Streszczenie
Aplikacje mobilne na platformę Android można implementować z użyciem języków Java lub C#. Artykuł przedstawia porównanie wydajności czasowej tych języków podczas przesyłania różnych plików typu tekstowego, obrazu i wideo w aplikacji mobilnej. Badania zostały przeprowadzone z użyciem dwóch aplikacji mobilnych o identycznych funkcjonalnościach. Na podstawie zebranych danych aplikacja serwerowa obliczyła statystyki, takie jak na przykład czas wymagany na przesłanie 1 MB danych w zależności od typu oraz rozmiaru pliku. Na podstawie otrzymanych wyników udowodniono, że w przypadku transferu danych poprzez sieć bezprzewodową aplikacja napisana w języku Java cechuje się większą wydajnością czasową niż analogiczna aplikacja napisana w języku C#.

Słowa kluczowe: aplikacje mobilne; REST API; Java; C#

*Corresponding author
Email address: **michal.jankowski1@pollub.edu.pl** (M. Jankowski)

1. Wstęp

Aplikacje mobilne w dzisiejszych czasach towarzyszą ludziom na każdym kroku: czy to bankowość elektroniczna, komunikacja, czy też przeznaczanie czasu na korzystanie z serwisów internetowych lub społecznościowych bądź gier. Narastający trend nowych aplikacji w gwałtownym tempie zaczął rozwijać się po rozpowszechnieniu telefonów komórkowych zdolnych owe aplikacje obsłużyć. Mowa tu konkretnie o telefonach z ekranem dotykowym wyposażonym na przykład w system Android. Nowy trend smartfonów pozwolił utworzyć kolejną gałąź Informatyki w programowaniu, czyli systemy mobilne. Mnogość rozwiązań oraz bezcenna przydatność telefonów komórkowych oraz różnych aplikacji w codziennym życiu zapoczątkowała powstawanie firm specjalizujących się w tworzeniu i rozwijaniu aplikacji mobilnych. Bardzo często wykorzystywane są w nich właśnie języki Java oraz C#.

Celem artykułu jest udowodnienie tezy: Aplikacja mobilna na systemy Android napisana w języku Java oferuje większą wydajność czasową niż analogiczna aplikacja napisana w języku C#. Do przeprowadzenia badania wymagane są elementy bądź czynności, które będzie można zmierzyć odpowiednimi jednostkami miary, takimi jak np. milisekundy. Do tego celu stworzona została aplikacja z funkcjami przesyłania oraz pobierania plików, a także wyświetlaniem statystyk oraz ewentualnym ich pobieraniem. Pozwala to tworzyć raporty oparte o pliki wygenerowane poprzez aplikację i wyświetlać dane na przykład w postaci czytelnych wykresów. Ze względu na konieczność analizy oraz przechowywania zgromadzonych danych stworzona została dodatkowo aplikacja serwerowa. Połączenie pomiędzy aplikacjami mobilnymi, a aplikacją serwerową odbywa się za pomocą sieci bezprzewodowej. Jest to konieczne ze względu na warunki użytkowania telefonów komórkowych – są one połączone rozległymi sieciami, których prędkości transferu danych rosną w coraz szybszym tempie [1].

2. Przegląd literatury

W celu zrozumienia konieczności przeprowadzenia badań w obszarze różnic wydajnościowych czasów należy zasięgnąć literatury. Znalezione materiały porównujące języki Java oraz C# zestawiają najczęściej różnice składniowe obu języków [2-5]. Autorzy prac analizują sposób zarządzania pamięcią oraz różnice w składniach bądź metodach języków. Artykuły opisujące różnice wydajnościowe aplikacji to albo czyste dane zebrane za pomocą konkretnej aplikacji [6, 7], albo szerzej opisane przeprowadzone badania z wykorzysta-

niem metod numerycznych (wielokrotne wykonywanie dzieleń, obliczanie wielomianów) [8]. Literatura skupiająca się na systemie Android w niewielkim stopniu opisuje różnice pomiędzy omawianymi językami. Zazwyczaj są to tak, jak wcześniej wspomniane, różnice składniowe [9, 10], czy też badania na plikach lokalnych [11, 12]. Aktualny stan wiedzy autora pracy pozwala stwierdzić, że nie znaleziono badań skupiających się na zagadnieniu omówionym w niniejszej pracy.

3. Języki oraz style architektury programowania

Języki programowania, których wydajność czasowa badana jest w niniejszej pracy to Java oraz C#. Java stosowana jest zarówno w aplikacjach mobilnych jak i desktopowych, do gier, czy też aplikacji komercyjnych [13]. Najbardziej popularną aplikacją do tworzenia oprogramowania kojarzącą się z systemem Android jest Android Studio, które wykorzystuje właśnie ten język [14, 15].

W dalszym etapie do stworzenia aplikacji serwerowej użyty został szkielet aplikacji Spring [16], czyli inaczej platforma pomagająca w budowaniu aplikacji. Spring Framework oparty został na języku Java. Przenośność, wygoda programowania, szybkość tworzenia aplikacji oraz prosta logika zależności - to cechy opisujące Javę, które przemawiają za częstym wybieraniem jej przez programistów.

Drugi język wykorzystany w pracy to C#. Podobnie jak, Java jest językiem obiektowym z hierarchią klas. Zastosowanie C# również sprowadza się do gier i programów komercyjnych, jednak w tym przypadku są to rozbudowane silniki symulacyjne czy też skomplikowane systemy [17]. Warto tu nadmienić, że firma Microsoft mocno skupia się na wykorzystaniu języka C# w swoich narzędziach (Visual Studio, Office). Firma ta również udostępnia tzw. silnik Unity, czyli środowisko do prostego tworzenia gier na przykład na systemy mobilne. Wcześniej wymienione narzędzie Visual Studio, oczywiście przy użyciu różnych bibliotek i programów kompilujących kod źródłowy, czyli translacji języka na taki zrozumiały dla systemu, pozwoliło stworzyć aplikację napisaną w języku C# dla systemu Android [10].

Badanie, które zostało przedstawione w pracy, umożliwiło zestawić dwa bardzo podobne sobie języki programowania cechujące się obiektowością, czyli paradygmatem bazującym na pojęciach klasy i obiektu, jak również sposobem zarządzania pamięcią [18]. Ten ostatni fakt oznacza to, że programista nie musi martwić się o zapewnienie aplikacji wymaganej pamięci sprzętowej.

Dodatkowo, celem komunikacji i wymiany danych pomiędzy użytkownikami a serwerami stosuje się bardzo często tak zwane REST API, czyli w uproszczeniu pewien styl ze zdefiniowanymi regułami [19]. Pozwala on w sposób prosty zaimplementować metody, które będą potem wykorzystywane przez aplikacje systemów mobilnych do wymiany danych. Wcześniej wymieniona platforma Spring w tym przypadku bardzo dobrze pozwala wykorzystać swój potencjał [20].

4. Metoda badań

Badania zostały przeprowadzone z użyciem dwóch środowisk testowych. W tabeli 1 przedstawione zostały specyfikacje środowisk testowych podczas przeprowadzania badań.

Tabela 1: Specyfikacja środowisk testowych

	Stanowisko nr 1	**Stanowisko nr 2**
Nazwa urządzenia	Laptop	Komputer stacjonarny
Typ pamięci masowej	SSD	SSD RAID 0
Procesor	2 rdzenie; 4 wątki; 2,2 GHZ	4 rdzenie; 4 wątki; 4,4 GHZ
Pamięć RAM	8 GB	12 GB
Połączenie sieciowe	Wi-Fi	LAN

Ustawienie połączeń sieciowych pozwala zasymulować typowe warunki użytkowania telefonu komórkowego połączonego poprzez sieć. Na stanowisku nr 1 uruchomiona została aplikacja serwerowa, zaś na stanowisku nr 2 aplikacja mobilna. Mierzonymi parametrami w aplikacjach są:

- *czas pobierania pliku;*
- czas wysyłania pliku;
- czas opóźnienia komunikacji pomiędzy aplikacją mobilną a serwerową.

Aplikacja serwerowa otrzymuje powyższe parametry od aplikacji mobilnych i zapisuje je w bazie danych pod odpowiednimi indeksami statystyk. Na podstawie tych danych wyliczane są dodatkowe parametry. Poniżej rozpisane formuły przedstawione są dla wysyłania plików. W przypadku pobierania, owe formuły zamiast wyrażeń oznaczających wysyłanie mają zaimplementowane wyrażenia oznaczające pobieranie.

1. Średnia wartość opóźnienia (ms) – wartość wyrażona w milisekundach, opisująca średni czas potrzebny na otrzymanie odpowiedzi na żądanie przez aplikację mobilną od aplikacji serwerowej Formuła wyliczająca ową wartość jest:
   suma czasów zebranych opóźnień operacji / liczba plików wysłanych

2. Czas przesyłania w przeliczeniu na 1 MB dla danego pliku (ms) - oznacza średni czas zagregowany ze wszystkich operacji pobierania oraz wysyłania podzielony na języki aplikacji źródłowych. Formułą licząca nie jest zwykła średnia lecz dokładniejsza,

mniej podatna na niepewność pomiarową dla plików o niskich rozmiarach:
suma czasów wysyłania wszystkich plików / suma rozmiarów wszystkich plików wysłanych

Dla celów badawczych stworzone zostały pliki z różnymi rozszerzeniami oraz rozmiarami mierzonymi w jednostkach informatycznych określające wymiar informacji danych. Typami i rozszerzeniami plików użytych podczas badania są:
- obraz (jpg);
- wideo (mp4);
- tekst (txt).

W przypadku rozmiarów plików stosowany jest rozmiar binarny, czyli np. 1 kilobajt (KB) równy jest 1024 bajtom (B) [21]. Rozmiary plików dla każdego z rozszerzeń:
- 10 KB;
- 100 KB;
- 10 MB;
- 100 MB.

Ze względów technicznych i konieczności przeprowadzenia badania na plikach naturalnych, to znaczy niewygenerowanych w sztuczny sposób (np. za pomocą poleceń wypełniających plik białymi znakami), rozmiary poszczególnych plików mają nieznaczne odchylenia.

4.1. Scenariusze badawcze

Każdy scenariusz obejmował przeprowadzenie transferu plików w liczbie zależnej od jego rozmiaru:
- 10 KB - 1000 razy;
- 100 KB - 1000 razy;
- 10 MB - 100 razy;
- 100 MB – 10 razy.

Wysyłanie plików

Badane parametry:
- średni czas wysyłania w przeliczeniu na 1 MB dla danego pliku (ms);
- średni czas wysyłania w przeliczeniu na 1 MB dla wszystkich plików (ms);
- średni czas opóźnienia podczas wysyłania danego pliku (ms);
- średni czas opóźnienia podczas wysyłania wszystkich plików (ms).

Scenariusz zrealizowany został dla:
- dwóch aplikacji mobilnych (aplikacja Java oraz C#);
- trzech typów plików (obraz, wideo oraz tekst);
- czterech rozmiarów plików (10 KB, 100 KB, 10 MB oraz 100 MB).

Pobieranie plików

Badane parametry:
- średni czas pobierania w przeliczeniu na 1 MB dla danego pliku (ms);
- średni czas pobierania w przeliczeniu na 1 MB dla wszystkich plików (ms);
- średni czas opóźnienia podczas pobierania danego pliku (ms);
- średni czas opóźnienia podczas pobierania wszystkich plików (ms).

Scenariusz zrealizowany został dla:
- dwóch aplikacji mobilnych (aplikacja Java oraz C#);
- trzech typów plików (obraz, wideo oraz tekst);
- czterech rozmiarów plików (10 KB, 100 KB, 10 MB oraz 100 MB).

5. Wyniki badań

5.1. Wysyłanie plików

Przeważająca liczba niższych wartości znajduje się w kolumnie oznaczonej językiem Java (tabela 2). Oznacza to, że w przypadku wysyłania plików aplikacja mobilna w języku Java odznacza się lepszą wydajnością. Również różnice procentowe wartości bardzo często przekraczają 10%. Można więc stwierdzić, że różnica w wydajności porównywanych aplikacji jest znacząca.

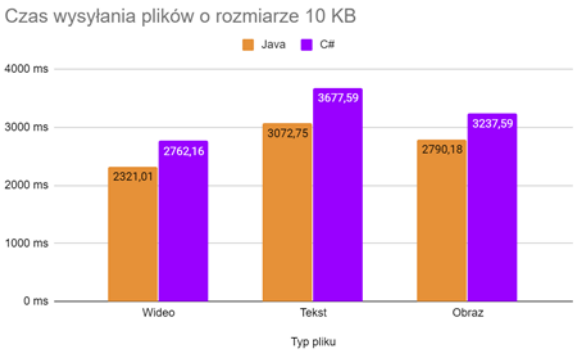
Tabela 2: Dane dotyczące wysyłania plików

	Wysyłanie				
	Rozmiar pliku	Typ pliku	Java	C#	Różnica procentowa
Czas przesyłania w przeliczeniu na 1 MB dla danego pliku [ms]	10 KB	Wideo	2321,01	2762,16	15,97%
		Tekst	3072,75	3677,59	16,45%
		Obraz	2790,18	3237,59	13,82%
	100 KB	Wideo	402,65	493,57	18,42%
		Tekst	524,18	549,69	4,64%
		Obraz	495,35	557,18	11,10%
	10 MB	Wideo	202,42	198,97	1,70%
		Tekst	196,86	204,97	3,96%
		Obraz	201,17	196,03	2,56%
	100 MB	Wideo	190,56	215,06	11,39%
		Tekst	187,7	205,78	8,79%
		Obraz	189,9	209,39	9,31%

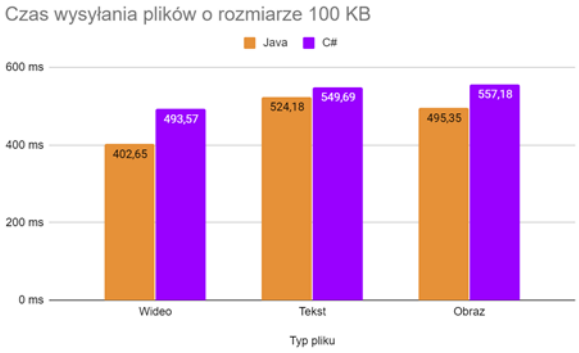
Wykresy przedstawione na rysunkach od 1 do 4 pozwolą przeanalizować różnice pomiędzy typami plików dla danych rozmiarów. Tło komórek oznaczone kolorem zielonym oznacza wartość niższą, przy porównywaniu obu języków. Dodatkowo, w celu ułatwienia porównania wartości dodana została dodatkowa kolumna oznaczająca różnice procentową pomiędzy wartością maksymalną a minimalną.

Na rysunkach od 1 do 4 widać, że przypadku przesyłania pliku o najmniejszych rozmiarach, najwydajniejsze jest przesyłanie plików wideo. Elementy o rozmiarach 10 oraz 100 KB wykazują najniższe czasy przesyłania z wyraźnymi różnicami dla aplikacji napisanej w języku Java. Rozmiar plików rzędu 10 MB wykazuje znacznie niższe czasy przesyłania przemawiające za aplikacją C#. W tym też zakresie dominuje obraz jako plik z najbardziej korzystną wartością czasu transferu. Pliki o rozmiarze 10 MB są najbardziej wydajne w przeliczeniu na czas wymagany do przesłania 1 MB danych.

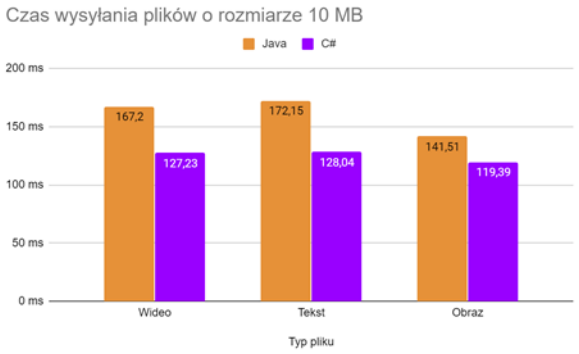
Jednym z wyznaczników rzetelności badania jest współczynnik korelacji Pearsona [22, 23]. Współczynniki korelacji bliższe liczbie 1 bądź -1 oznaczają korelację coraz silniejszą, zaś bliższe 0 oznaczają korelację coraz słabszą. Danymi, których korelację badano były wartości czasów przesyłania w przeliczeniu na 1 MB dla danego pliku w zestawieniu Java oraz C#. W przypadku wysyłania plików, współczynnik ten wynosi 0,9997224976, czyli jest bardzo bliski wartości jeden. Fakt ten oznacza, że wyniki badania nie zostały zaburzone w żaden sposób, nie występują również znaczące odchylenia pojedynczych wartości, co miałoby wpływ na analizę wyników.



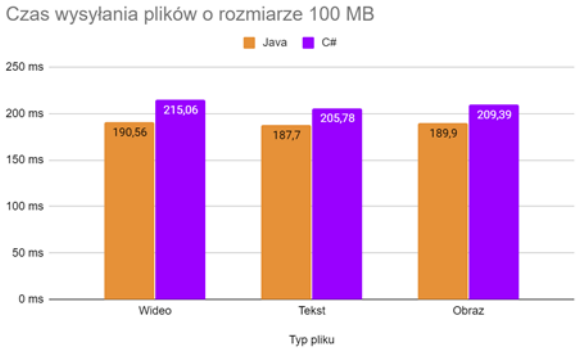
Rysunek 1: Czas wysyłania plików o rozmiarze 10 KB



Rysunek 2: Czas wysyłania plików o rozmiarze 100 KB



Rysunek 3: Czas wysyłania plików o rozmiarze 10 MB



Rysunek 4: Czas wysyłania plików o rozmiarze 100 MB

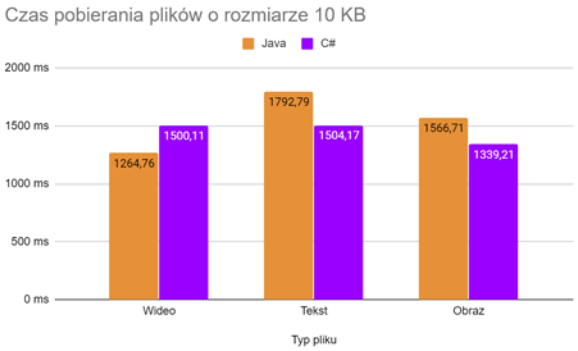
5.2. Pobieranie plików

Zagregowane dane wskazują na zróżnicowanie wyników i wyraźny kontrast dla poszczególnych badanych rozmiarów (tabela 3). Zakres danych dla 10 oraz 100 KB wykazuje większą wydajność w przypadku aplikacji napisanej w języku C#. Pozostałe badane pliki, czyli te o rozmiarach 10 i 100 MB przemawiają natomiast za aplikacją C#, jednakże, z niewielkimi różnicami, rzędu około 6%.

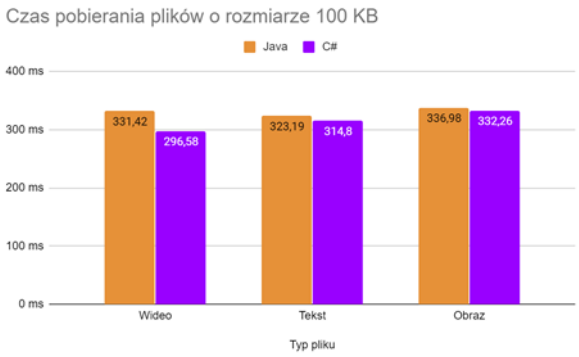
Tabela 3: Dane dotyczące pobierania plików

	Pobieranie				
	Rozmiar pliku	Typ pliku	Java	C#	Różnica procentowa
Czas przesyłania w przeliczeniu na 1 MB dla danego pliku [ms]	10 KB	Wideo	1264,76	1500,11	15,69%
		Tekst	1792,79	1504,17	16,10%
		Obraz	1566,71	1339,21	14,52%
	100 KB	Wideo	331,42	296,58	10,51%
		Tekst	323,19	314,8	2,60%
		Obraz	336,98	332,26	1,40%
	10 MB	Wideo	216,4	237,08	8,72%
		Tekst	225,88	236,18	4,36%
		Obraz	226,83	242,05	6,29%
	100 MB	Wideo	236,45	271,74	12,99%
		Tekst	234	218,65	6,56%
		Obraz	236	249,38	5,37%

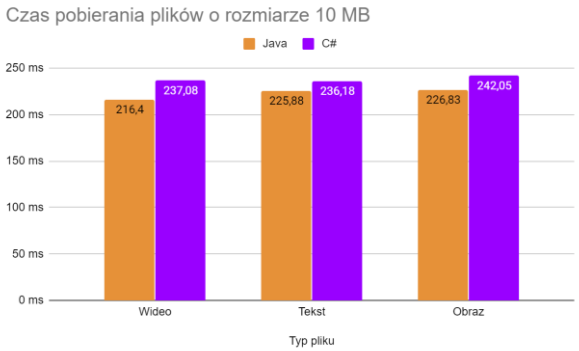
Wykresy na rysunkach od 5 do 8 ukazujące czas pobierania 1 MB danych w zależności od typu dla podanego rozmiaru pliku wykazują unormowane dane z odchyleniami w przypadku pliku tekstowego o rozmiarze 10 KB. Największą wydajność dla pobierania, w odróżnieniu od badania dotyczącego wysyłania plików, wykazują zarówno pliki o rozmiarze 10 MB jak i 100 MB. Niemniej jednak, różnice pomiędzy wartościami są niewielkie.



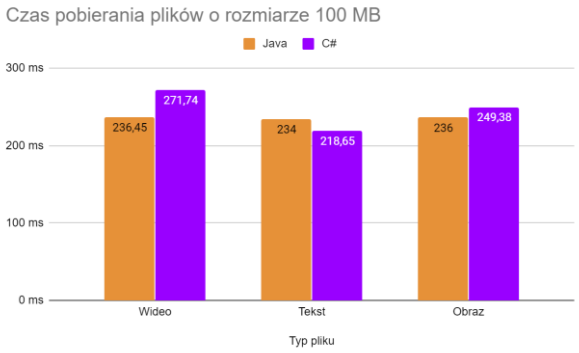
Rysunek 5: Czas pobierania plików o rozmiarze 10 KB



Rysunek 6: Czas pobierania plików o rozmiarze 100 KB



Rysunek 7: Czas pobierania plików o rozmiarze 10 MB



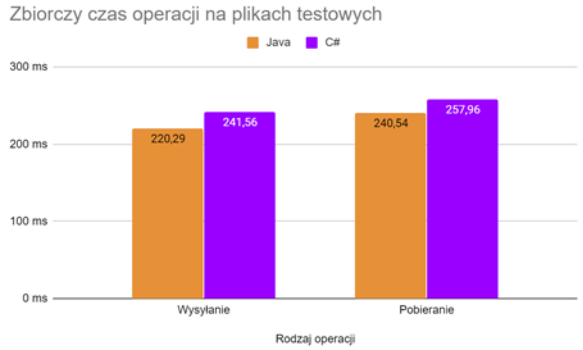
Rysunek 8: Czas pobierania plików o rozmiarze 100 MB

Wartość współczynnika korelacji Pearsona obliczona została na podstawie wyników z tabeli 3 dla języków Java oraz C#. Współczynnik ten dla pobierania plików, tak jak w przypadku wysyłania, jest bardzo bliski liczbie 1 i wynosi około 0,9774876905. Oznacza to silne powiązanie danych dla mierzonych wartości dzięki czemu badanie można uznać za rzetelne, bez występujących znaczących odchyleń dla pojedynczych wartości.

Tabela 4: Dane ogólne transferu plików dla badania nr 1

	Wartości średnie danych zebranych w badaniu			
	Rodzaj operacji	Java	C#	Różnica procentowa
Czas przesyłania w przeliczeniu na 1 MB dla danego pliku [ms]	Wysyłanie	220,29	241,56	8,81%
	Pobieranie	240,54	257,96	6,75%
Wartość opóźnienia [ms]	Wysyłanie	0,07	0,25	72,00%
	Pobieranie	0,07	0,13	46,15%

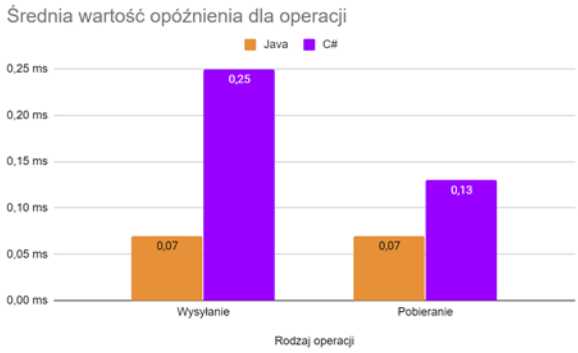
Dane przedstawione w tabeli 4 reprezentują dane uzyskane podczas przeprowadzania całego badania. Od razu zauważyć można dominację koloru zielonego w kolumnie oznaczonej językiem Java. Fakt ten oznacza wartości niższe, a więc bardziej korzystne dla każdego zagregowanego parametru. W przypadku czasu przesyłania plików w przeliczeniu na 1 MB, różnica wartości wynosi odpowiednio dla wysyłania oraz pobierania 8,81% i 6,75%.



Rysunek 9: Zebrany czas operacji transferu plików

Wbrew informacjom uzyskanym z wykresów i tabel w poprzednich punktach, ogólna wydajność, w przypadku wysyłania danych poprzez sieć przemawia za korzystaniem z aplikacji napisanej w języku Java (rysunek 9). Różnice wydajności pomiędzy operacjami wysyłania oraz pobierania są bardzo niewielkie (około 2%).

Z rysunku 10 wynika, że uśrednione wartości opóźnień dla operacji wykonanych na plikach jasno wskazują na komunikację pomiędzy aplikacją Java a aplikacją serwerową jako dużo bardziej wydajną. Współczynnik korelacji obliczony został na podstawie wartości średnich zebranych czasów przesyłania dla wszystkich badanych plików z wykorzystaniem aplikacji Java oraz C# i wynosi on 0,9998774792. Oznacza to silny związek powiązania wartością a co za tym idzie ukazuje to wiarygodność danych i nieznaczne odchylenia w przypadku pojedynczych wartości.



Rysunek 10: Czas opóźnień dla transferu plików

6. Wnioski

Teza zakłada większą wydajność czasową przesyłania danych aplikacji Java niż aplikacji C#. Przeprowadzone badania wykazały słuszność tezy. Przypomnieć należy, że badania wykonane zostały w warunkach typowego użytkowania aplikacji mobilnych, czyli połączenia poprzez sieć bezprzewodową. Wyniki przeprowadzonych badań wykazały około 7-8% przewagę w wydajności aplikacji Java nad aplikacją C#. W badaniach dodatkowo zostały zestawione różne typy oraz rozmiary plików. Badanie pierwsze wykazało największą wydajność przesyłania plików o rozmiarach 10 MB dla wysyłania oraz 10 MB i 100 MB dla pobierania. Jeżeli chodzi o typy plików, to trudno wyłonić taki, który wyróżniałby się większą wydajnością w porównaniu do pozostałych.

Na koniec należy wspomnieć, że pomimo starannych przygotowań środowisk testowych w sposób jednakowy i uniezależnienia ich od ewentualnych błędów pomiarowych to jednak pomiary obarczone są pewnymi drobnymi odchyleniami. Wprawdzie zbadane współczynniki korelacji nie wykazują różnic zależności, to jednak ograniczona zasobami sprzętowymi liczba próbek badawczych oraz wpływ niezależnych czynników na wykonywanie testów takich jak np. zmienne taktowania procesorów, wpływ temperatury na wydajność komponentów, specyfika sieci bezprzewodowej, czyli zdolność do tracenia pakietów i obniżenia prędkości połączeń, czy też zmienne obciążenie procesora funkcjami systemowymi. Pomimo powyższych trudności, których nie sposób zapobiec w warunkach domowych, zdaniem autora badanie zostało przeprowadzone w sposób należyty, rzetelny i z jak największą dokładnością.

7. Podsumowanie

Celem niniejszej pracy było porównanie w zakresie transferu plików dwóch analogicznych aplikacji w językach Java oraz C#. Wymagane było stworzenie aplikacji

serwerowej wraz z bazą danych oraz aplikacji mobilnych w dwóch językach programowania. Połączenie podobnych ze względu na obiektowość języków programowania umożliwiło poznanie oraz nauczenie się nowych rozwiązań programistycznych. Pozwoliło to także na podejście do tematu z dwóch różnych stron i ujawniło, że pomimo różnych środowisk programistycznych, wiele metod ma bardzo podobną składnię oraz zasadę działania. Jednakże, mimo tych podobieństw, otrzymane wyniki nieznacznie się różniły względem początkowych założeń.

Literatura

[1] Sieć 5G, https://www.plus.pl/news/art-8353-nadchodzi-kolejna-generacja-sieci-komorkowej-5g [05.06.2020].

[2] Java vs C#, https://www.educba.com/java-vs-c-sharp/ [05.06.2020].

[3] I. Alkadi, G. Alkadi, H. Etheridge, A Comparative Analysis Of C# And Java As An Introductory Programming Language For Information Systems Students, Review of Business Information Systems (RBIS) 10 (2011) 37-40.

[4] Porównanie składniowe Java oraz C#, https://www.softwaretestinghelp.com/csharp-vs-cpp-vs-java/ [05.06.2020].

[5] Porównanie składniowe Java oraz C#, https://www.guru99.com/java-vs-c-sharp-key-difference.html [05.06.2020].

[6] Porównanie wydajnościowe Java oraz C#, http://www.bentuser.com/article.aspx?ID=323&AspxAutoDetectCookieSupport=1 [05.06.2020].

[7] Porównanie wydajnościowe Java oraz C# https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/csharp.html [05.06.2020].

[8] Porównanie wydajnościowe Java oraz C# (artykuł), http://www.itu.dk/~sestoft/papers/numericperformance.pdf [05.06.2020].

[9] A. Stasiewicz, Android. Podstawy tworzenia aplikacji, Helion, 2013.

[10] A. Kempa, T. Staś, Wstęp do programowania w C#: Łatwy podręcznik dla początkujących, Tomasz Staś, 2014.

[11] M Reynolds., Xamarin Mobile Application Development for Android, Packt Publishing Ltd, 2014.

[12] I. F. Darwin, Android. Receptury, Helion, 2013.

[13] Zastosowanie języka Java, https://jaki-jezyk-programowania.pl/technologie/java/ [05.06.2020].

[14] Ranking IDE, https://pypl.github.io/IDE.html [05.06.2020].

[15] Android Studio, https://developer.android.com/studio [05.06.2020].

[16] Definicja Spring, https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html [05.06.2020].

[17] Zastosowanie C#, https://testuj.pl/blog/10-pytan-programisty-temat-jezyka-csharp/ [05.06.2020].

[18] F. Friesen, Learn Java for Android Development, Apress, 2013.

[19] B. Burke, RESTful Java with JAX-RS, "O'Reilly Media, Inc.", 2010.

[20] F. Gutierrez, Wprowadzenie do Spring Framework dla programistów Java, Helion, 2015.

[21] Jednostka informacji, https://pl.wikipedia.org/wiki/Jednostka_informacji [05.06.2020].

[22] Współczynnik korelacji Pearsona, https://pogotowiestatystyczne.pl/slowniczek/korelacja-pearsona/ [05.06.2020].

[23] Użyteczność korelacji Pearsona w badaniach, http://zsi.tech.us.edu.pl/~nowak/smad/SMAD_korelacje.pdf [05.06.2020].