JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 273–279
Received: 15 June 2022
Accepted: 22 June 2022

Comparative analysis of Java and Dart programming languages in terms of suitability for creating mobile applications

Analiza porównawcza języków programowania Java oraz Dart pod kątem przydatności do tworzenia aplikacji mobilnych

Łukasz Kozłowski*, Grzegorz Kozieł

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

Abstract
This paper presents the results of a comparative analysis of Java and Dart programming languages in terms of suitability for creating mobile applications. The research was carried out on two proprietary applications with identical functionalities, which were implemented in the analyzed languages. The analysis covers areas such as: CPU load, RAM consumption, battery consumption and program execution time. Additionally, on the basis of proprietary applications, the code structure, number of lines of code and community support were considered. The results of the analysis indicate that it is difficult to clearly determine which language is more efficient, so the choice between Java and Dart should be analytical and best suited to the requirements of a given application.

Keywords: Java; Dart; Android

Streszczenie
W niniejszej pracy przedstawiono wyniki analizy porównawczej języków programowania Java i Dart pod kątem przydatności do tworzenia aplikacji mobilnych. Badania przeprowadzono na dwóch autorskich aplikacjach o identycznych funkcjonalnościach, które zostały zaimplementowane w rozpatrywanych językach. Analiza obejmuje obszary takie jak: obciążenie procesora, zużycie pamięci RAM, zużycie baterii oraz czas wykonania programów. Dodatkowo na podstawie autorskich aplikacji rozpatrzono budowę kodu, liczbę linii kodu i wsparcie społeczności. Rezultaty analizy wskazują, że trudno jednoznacznie określić, który język jest wydajniejszy, dlatego wybór pomiędzy językiem Java, a Dart powinien być analityczny i jak najlepiej dopasowany do wymagań danej aplikacji.

Słowa kluczowe: Java; Dart; Android

*Corresponding author
Email address: lukasz.kozlowski@pollub.edu.pl (Ł. Kozłowski)

1. Wstęp

Dynamiczny rozwój technologii sprawił, że urządzenia mobilne są nierozłącznym elementem codziennego życia dla wielu ludzi. Do funkcjonowania smartfonów niezbędne są aplikacje mobilne, które współcześnie występują niemal w każdej dziedzinie życia. Ze względu na dynamiczny rozwój technologii konieczne jest udoskonalanie aplikacji mobilnych, aby przystosować je do pracy na coraz wydajniejszych smartfonach z różnorodnymi systemami operacyjnymi.

Na dzień dzisiejszy można wyróżnić dwa najpopularniejsze systemy operacyjne przeznaczone na urządzenia mobilne: Android oraz iOS, gdzie ponad 69% urządzeń mobilnych na świecie posiada system Android [1]. Obecnie firmom zależy na pozyskaniu dla swoich projektów mobilnych zarówno użytkowników posiadających system Android jaki iOS. Rozwiązaniem takiego aspektu jest zaprogramowanie aplikacji działającej pod kontrolą obu systemów. Umożliwia to język Dart przy współpracy z narzędziem Flutter. Takie rozwiązanie pozwala zminimalizować koszty oraz zasoby ludzkie, które należałoby przeznaczyć na utworzenie identycznych aplikacji z różnymi bazami kodu w różnych językach. W tym przypadku programista tworzy jedną implementację, a finalnie otrzymuje aplikację działającą na dwóch systemach operacyjnych.

W niniejszym artykule dokonano analizy porównawczej dwóch języków programowania: języka Java, który jest uważany za prekursora wśród języków programowania przeznaczonych dla aplikacji mobilnych oraz język Dart, który oferuje możliwość uruchomienia tego samego kodu na różnych platformach. Przeprowadzone porównanie ukierunkowane było na badania wydajnościowe, którym poddane były dwie identycznie działające autorskie aplikacje mobilne napisane w rozpatrywanych językach programowania. Przy porównywaniu wydajności skupiono się na takich parametrach jak obciążenie procesora, wykorzystanie pamięci RAM, zużycie baterii oraz czas wykonania programów. Dodatkowo pod uwagę wzięto również budowę kodu, liczbę linii z kodem i wsparcie społeczności. Rozważania podjęte w tym artykule mają ułatwić twórcom oprogramowania podjęcie decyzji w momencie, gdy muszą dokonać wyboru technologii, która zostanie użyta do opracowania aplikacji mobilnej.

2. Przegląd literatury

Rozpatrując budowanie aplikacji mobilnych należy wziąć pod uwagę, że istnieje możliwość programowania

takich aplikacji na dwa sposoby - w sposób natywny oraz wieloplatformowy. Aplikacja napisana natywnie charakteryzuje się tym, że jest dostępna na jedną platformę. Podejście wieloplatformowe umożliwia wykorzystanie jednego kodu do pracy na wielu platformach [2]. Aspekt ten zbadali: P. Kotarski, K. Śledź i J. Smołka w publikacji [3]. Elementy procesu badawczego odnosiły się do narzędzi informatycznych służących do generowania aplikacji przeznaczonych dla systemu Android. [3]. W swoich eksperymentach autorzy zaimplementowali programy w różnych technologiach, aby sprawdzić ich wydajność dotyczącą czasu posortowania tablic z elementami liczbowymi oraz zapisu i odczytu dużego pliku w urządzeniu fizycznym. Podsumowując dokonane wyniki badań twórcy zauważają, że rozwiązania międzyplatformowe nie wykazują właściwości, które mogłyby przyczynić się do ich zdeklasowania w porównaniu z metodami natywnymi. Ponadto stwierdzają, że oba rozwiązania są do siebie zbliżone pod względem wydajnościowym. W pracy nie stwierdzono jednoznacznie, które narzędzie jest bardziej efektywne dla programistów [3].

Jako drugie źródło wiedzy w analizowanym temacie oparto się na artykule D. Gałana, K. Fisza i P. Kopniaka [4]. Praca przedstawia konfrontację dwóch aplikacji, z których pierwsza została napisana w języku Kotlin przy wykorzystaniu zestawu narzędzi Android, zaś druga w języku Dart z zastosowaniem Flutter SDK [4]. Jako środowisko programistyczne posłużyło narzędzie Android Studio [5]. W badaniach autorzy biorą pod uwagę takie aspekty jak czas wykonywania operacji i obciążenie procesora urządzenia mobilnego, które mierzone są podczas przeprowadzania opracowanych scenariuszy testowych. Są to odpowiednio: sortowanie zbioru liczbowego, zapisywanie i odczytywanie danych z lokalnej bazy danych oraz zapisywanie i odczytywanie znaków z pliku tekstowego [4]. Ponadto autorzy analizują również biblioteki będące do dyspozycji programistów jak i objętość kodu źródłowego oraz wsparcie społeczności. Na podstawie przeprowadzonych eksperymentów twórcy pracy dochodzą do wniosku, że aplikacja zaimplementowana w języku Kotlin przy użyciu zestawu narzędzi Android SDK jest wydajniejsza, a ponadto posiada optymalny kod źródłowy i większe wsparcie społeczności oraz bibliotek niż aplikacja napisana za pomocą zestawu narzędzi Flutter SDK [4].

W publikacji [6] G. Kozieła oraz D. Sulowskiego wykonano badanie porównawcze języków programowania Java i Kotlin. W związku z badaniami zaimplementowano autorską aplikację mobilną oddzielnie dla każdego języka. Jej funkcjonalność polegała na sortowaniu zbiorów liczbowych przy użyciu wybranych algorytmów oraz wyświetleniu animacji. W analizie porównawczej skupiono się na takich aspektach jak czasy wykonania i kompilacji, obciążenie pamięci RAM jak też procesora. Poza tym do analizy włączono strukturę kodu, dostępność baz danych, a także wsparcie wspólnoty społeczności programistycznej. Należy dodać, że analiza postawionych kryteriów odbywała się podczas obserwacji projektowania, implementacji oraz działania stworzonej aplikacji. Otrzymane wyniki pozwoliły autorom pracy na wysunięcie wniosku, że zarówno jeden jak i drugi język nie wyróżnia się znacząco, aby stwierdzić, który z nich jest wydajniejszy przy rozpatrywaniu tworzenia aplikacji mobilnych działających na urządzeniach z systemem Android.

Spośród znalezionych publikacji za najbardziej dopasowaną do niniejszej pracy uznano publikację *Java and Dart programming languages: Conceptual comparison* autorstwa A. M. Hassana [7]. Autor w swojej pracy dokonuje omówienia języków programowania Dart oraz Java w celu określenia różnic oraz ich wspólnych cech. W artykule rozpatrywano właściwości dotyczące składni, typów danych, dziedziczenia, operatorów logicznych, instrukcji warunkowych oraz innych związanych z procesem kodowania aplikacji. Dodatkowo wskazano rozbieżność tych języków w zakresie używanych kompilatorów oraz dostępnych bibliotek. Poza tym przedstawiono także fragmenty kodu dla obu technologii. Z rozważań autora wynika, że język Dart ma ogromne możliwości programistyczne i może być równie atrakcyjny jak Java [7].

Z dotychczasowych rozważań wynika, że zarówno Java jak i Dart mają swoje atuty w konkretnych zastosowaniach. Należy zwrócić uwagę, że na popularności zyskują narzędzia wieloplatformowe wśród, których jest Flutter korzystający z języka Dart [8]. Dlatego rodzi się pytanie czy język Dart używany w rozwiązaniu dostarczającym aplikacje mobilne na Android i iOS ze wspólnego kodu jest wydajniejszy niż tradycyjne podejście z użyciem Javy i uzyskaniem aplikacji tylko na jedną platformę. Dodatkowo nie znaleziono publikacji dokonujących analizy tejże problematyki i z tego względu w niniejszym artykule podjęto się jej zbadania.

3. Metodyka badawcza

Na potrzeby badań zostały utworzone dwie autorskie aplikacje mobilne zaimplementowane w języku Java - przy użyciu standardowego zestawu narzędzi Android SDK oraz w języku Dart, którego używa narzędzie Flutter. Obie aplikacje posiadają takie same funkcjonalności, które posłużyły jako scenariusze do wykonania testów. Działanie aplikacji polegało na wykonaniu operacji po naciśnięciu odpowiedniego przycisku. Zaimplementowane operacje to wstawianie, edytowanie, odczytywanie i usuwanie tysiąca rekordów z nielokalnej bazy danych, a ponadto sortowanie tysiąca i dziesięciu tysięcy dodatnich liczb losowych.

Wyniki obciążeniowe były odczytywane podczas działania aplikacji z daną operacją przy wykorzystaniu środowiska programistycznego Android Studio. Środowisko to posiada wbudowane funkcje do monitorowania zachowania aplikacji działającej na urządzeniu fizycznym lub emulatorze, które określane są jako moduł Android Profiler. Do pomiaru czasów wykonania programów zastosowano wbudowane metody oferowane przez biblioteki badanych technologii. Należy jeszcze uwzględnić, że dla każdego z aspektów wydajnościowych wykonano po 100 prób.

W badaniu dotyczącym objętości kodu została rozpatrzona składnia języków Java i Dart, liczba linii kodu potrzebnych do napisania takiej samej autorskiej aplikacji w jednym jak i drugim języku. Kryterium, które obejmuje wsparcie społeczności polegało na zbadaniu liczby wpisów na portalach internetowych poświęconych społeczności programistycznej danego języka programowania.

Parametry urządzenia mobilnego wykorzystanego w badaniu przedstawiono w Tabeli 1. Dodatkowo należy wspomnieć, że telefon współpracował z bezprzewodową siecią - Wi-Fi o prędkości do 300 Mb/s przy pobieraniu i do 50 Mb/s przy wysyłaniu danych. Transmisja danych została wyłączona, aby urządzenie korzystało tylko z sieci bezprzewodowej.

Tabela 1: Parametry stanowiska badawczego

Urządzenie mobilne	Samsung Galaxy A52
Procesor	Qualcomm Snapdragon 720G, Ośmiordzeniowy
Pamieć RAM	6 GB
Pamięć ROM	128 GB
System	Android 11
Bateria	4500 mAh

Przed przystąpieniem do testów telefon został sformatowany i pozbawiony czynników, które mogłyby wpłynąć na wynik badania. Oprogramowanie na smartfonie zostało zaktualizowane do najnowszej wersji, aby wyeliminować sytuację niekontrolowanych połączeń z Internetem podczas wykonywania badań. W każdym z testów bateria była naładowana do poziomu nie mniejszego niż 85 %. Dodatkowo z urządzenia zostały usunięte wszystkie aplikacje, które prowadziłyby do nieoczekiwanych oddziaływań na jego zasoby.

4. Porównanie cech wybranych języków

Aplikacja napisana przy użyciu języka Java składa się ze zbioru klas. Klasy posiadają pola do przechowywania danych, które są określane mianem zmiennych, a co więcej posiadają także metody, aby wykonywać operacje na danych. Tworzenie klasy następuje zazwyczaj w osobnym pliku, który posiada taką samą nazwę jak klasa, ale nie jest to regułą. W Javie podczas deklaracji zmiennych należy zdefiniować rodzaj danych jakie będą przechowywane. Należy dodać, że od wersji Java 10 pojawił się typ specjalny var, dzięki któremu nie ma obowiązku podawania typu danych, ponieważ kompilator sam określi typ danej zmiennej.

Poznając język Dart należy pamiętać, że wszystko co można umieścić w zmiennej jest obiektem, a obiekt jest instancją klasy [9]. W tym języku klasy także posiadają zmienne oraz metody do pracy nad nimi. Instancje klasy można deklarować z określeniem typu danych, lecz nie ma takiego przymusu, gdyż od momentu powstania ten język posiada typ var, który pozwala na zadeklarowanie zmiennych bez określania typu.

Konstruktor w Javie posiada nazwę identyczną jak nazwa klasy i strukturę podobną do metod wraz z argumentami, które zostaną przypisane do obiektów klasy. Analogicznie jest w przypadku języka Dart, ponieważ domyślnie dla klasy również dostępny jest konstruktor bezargumentowy o tej samej nazwie co stworzona klasa. Konstruktor może przyjmować parametry w celu zainicjalizowania zmiennych klasy przy użyciu własnych wartości. W języku Java konstruktor nie może zwracać wartości. Natomiast w języku Dart istnieje taka możliwość przy użyciu konstruktora fabrykującego. Utworzenie takiego konstruktora jest możliwe przy użyciu słowa kluczowego factory przed nazwą konstruktora. Wywołanie takiego konstruktora jest identyczne jak w przypadku zwykłych konstruktorów.

Gettery i settery to metody, które pozwalają pobierać i inicjować wartości pól klas. W Dart, aby zdefiniować metodę pobierającą wartość pola klasy - getter, należy użyć słowa kluczowego get, które umieszczane jest po określeniu typu danych zwracanych, ale przed nazwą metody zwracającej dane pole klasy. Identycznie wygląda sytuacja podczas tworzenia metody inicjalizującej pola klasy - setter, ale tutaj słowem kluczowym jest set. Metoda ustawiająca setter posiada jeden parametr i nie zwraca żadnej wartości. Natomiast w Javie settery i gettery mogą mieć dowolną nazwę i nie potrzebują modyfikatora określającego set lub get. Wystarczy pamiętać, że metoda ustawiająca nic nie zwraca, posiada parametr i przypisuje go do pola klasy, a metoda pobierająca zawsze zwraca jakąś wartość i posiada typ jak zmienna zwracana.

4.1. Struktura projektów

Aplikacja zaimplementowana w języku Java została skompilowana w trybie debug, a w języku Dart w trybie profile. Do analizy struktury projektów autorskiej aplikacji napisanej w Java i Dart wykorzystano wtyczkę Statistic dla środowiska Android Studio [5]. Za pomocą tego dodatku sprawdzono m.in. liczbę plików, rozmiar projektów i liczbę linii z kodem. Narzędzie Statistic posiada możliwość filtrowania danych, które mają zostać poddane analizie między innymi po rozszerzeniu plików. Na potrzeby tej pracy skorzystano z filtrowania plików z rozszerzeniem .dart dla aplikacji napisanej w języku Dart oraz .java dla aplikacji korzystającej z Javy. W tym przypadku były to pliki napisane przez programistę. Dodatkowo dla aplikacji zaimplementowanej w języku Java poddano analizie pliki z rozszerzeniem .xml, które definiują graficzny interfejs aplikacji. W języku Dart interfejs graficzny jest definiowany w kodzie Dart. W Tabeli 2 zaprezentowano uzyskane wyniki.

Tabela 2: Statystyka plików Java, Xml oraz Dart

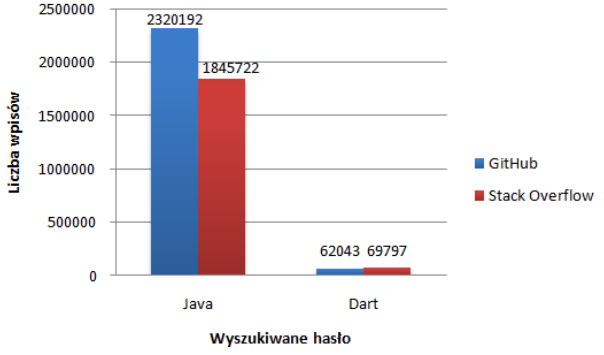
Kryterium	**Typ plików**		
	Dart	Java	XML
Liczba plików	9	7	3
Liczba linii z kodem	525	420	279
Rozmiar plików na dysku [KB]	28	32	20
Rozmiar całego projektu aplikacji po instalacji [MB]	47,56	13,35	

W statystyce ukierunkowanej tylko na dane z rozszerzeniami badanych języków to jest .java oraz .dart wynika, że liczba plików języka Dart jest o 2 większa w tym projekcie niż dla projektu zaimplementowanego w języku Java. Ponadto liczba linii zawierających kod jest większa o sto dla autorskiej aplikacji w języku Dart. Różnica w rozmiarze rozpatrywanych plików jest znikoma, ponieważ wynosi 4 KB i w tym przypadku to pliki języka Java zajmują więcej miejsca na dysku. Rozpatrując tworzenie graficznego interfejsu użytkownika widać, że dla aplikacji zaimplementowanej w języku Java niezbędne są dodatkowe pliki xml, w których zdefiniowany jest wygląd aplikacji. W języku Dart wygląd aplikacji jest zdefiniowany w kodzie plików z rozszerzeniem .dart. Dodatkowo należy zwrócić uwagę, że aplikacja zaimplementowana w języku Dart przy użyciu Fluttera posiada prawie cztery razy większy rozmiar niż autorska aplikacja w języku Java.

Analiza statystyki dla struktury projektu aplikacji napisanej przy współpracy Darta i Fluttera pokazuje, że prawie we wszystkich rozpatrywanych kryteriach ów projekt przyjmuje znacząco większe wartości niż projekt Java z Android SDK. Jednakże ze względu na to, że aplikacja zaimplementowana w języku Java potrzebuje dodatkowo plików definiujących wygląd to statystyka plików projektów jest korzystniejsza dla aplikacji w języku Dart. Można przypuszczać, że większy rozmiar aplikacji zaimplementowanej w języku Dart jest spowodowany wieloplatformowymi możliwościami Fluttera, który potrzebuje większej liczby plików oraz bibliotek, aby projekt napisany w jednym języku mógł działać na dwóch systemach to jest Android i iOS.

4.2. Wsparcie społeczności

W niniejszej pracy badanie dotyczące wsparcia społeczności obejmuje analizę popularności języków Java i Dart na wybranych portalach społeczności informatycznej. Podczas nauki bądź pracy programiści napotykają wiele problemów dotyczących języków, bibliotek oraz innych narzędzi programistycznych, którymi dzielą się na forach internetowych w celu zwiększenia szansy na rozwiązanie danego problemu i podzielenia się nim z innymi członkami społeczności. Aby ocenić popularność badanych języków zestawiono liczbę wpisów zawierających nazwą danego języka programowania z dwóch najbardziej popularnych serwisów społeczności to jest GitHub oraz Stack Overflow [10].



Rysunek 1: Popularność języków programowania.

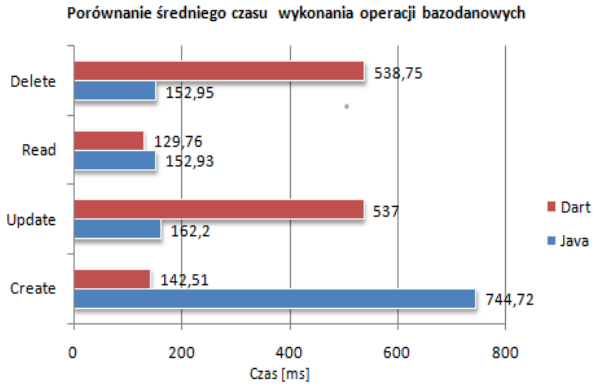
Analizując zebrane dane przedstawione na Rysunku 1 można zauważyć, że większą popularność osiąga Java. Badania popularności zostały przeprowadzone w kwietniu 2022 roku i dla Javy w serwisie GitHub odnotowano 2320192 wpisów, a w StackOverflow 1845722. Natomiast wyniki dla języka Dart są wyraźnie mniejsze i są to odpowiednio 62043 zapytania na portalu GitHub oraz 69797 dla StackOverflow [11, 12]. Interpretując uzyskane dane należy zwrócić uwagę na fakt, iż język Java miał swój początek 31 lat temu w 1991 roku [13]. Jest on zatem znacznie starszy niż język Dart, który pojawił się dopiero w 2011 roku [14]. Ze względu na swój młody wiek Dart nie jest jeszcze tak dobrze rozpowszechniony jak Java, która obecnie jest już widoczna niemal we wszystkich gałęziach przemysłu, biznesu i wielu innych kluczowych sektorach.

Warto dostrzec, że język Dart od 2015 roku jest ściśle rozwijany pod kątem aplikacji mobilnych w platformie Flutter [15] co może przyczynić się do wzrostu popularności tego języka wśród programistów tworzących aplikacje mobilne. Co więcej w 2021 roku Flutter, w którym wykorzystywany jest język Dart był najczęściej wybierany spośród narzędzi do tworzenia aplikacji mobilnych [8].

5. Wyniki wydajnościowe

5.1. Czas wykonania programów

Na Rysunku 2 przedstawiono zestawienie średnich czasów wykonania operacji, którymi są wstawianie rekordów do bazy danych - create, edytowanie - update, odczytanie - read i usuwanie - delete.

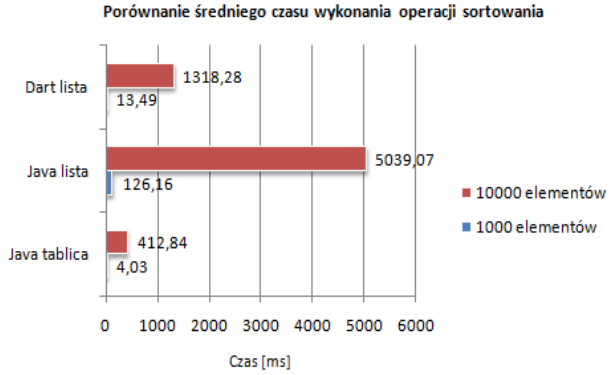


Rysunek 2: Porównanie średniego czasu wykonania operacji bazodanowych.

Aplikacja zaimplementowana w języku Java potrzebuje najwięcej czasu na wykonanie operacji wstawiania rekordów do bazy danych. Jest to wartość o 602,21 milisekundy dłuższa niż wynik dla tej samej funkcjonalności zaimplementowanej w języku Dart. Ostatnią operacją, w której Java osiąga lepsze wyniki czasowe nad językiem Dart jest proces odczytywania danych. Różnica jest znikoma, ponieważ wynosi zaledwie 23,17 milisekund na korzyść języka Dart. W pozostałych dwóch doświadczeniach: delete i update dla

języka Dart średni czas wykonania jest wyraźnie większy niż dla języka Java.

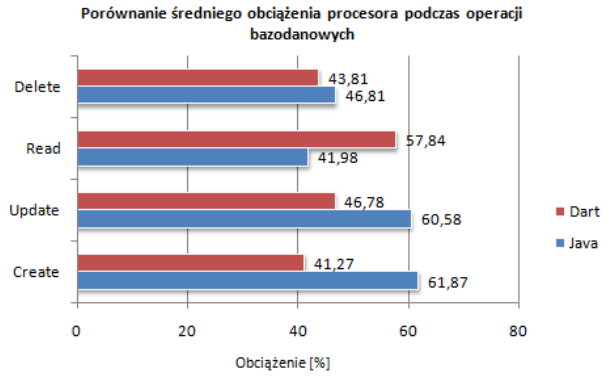
Kolejny scenariusz testowy dotyczył pomiaru czasu wykonania funkcji zaimplementowanej do sortowania zbiorów liczbowych przy wykorzystaniu algorytmu sortowania bąbelkowego. Zbiory danych zawierały odpowiednio 1000 oraz 10000 losowych dodatnich elementów typu int. Ich zawartość była identyczna zarówno dla aplikacji napisanej w języku Java jak i Dart. Dane przechowywane były w plikach, a przed sortowaniem następowało ich wczytanie i zapisanie liczb do odpowiednich struktur danych, to jest: dla języka Dart zbadano listę, a dla Javy tablicę oraz listę, ponieważ język Dart nie posiada oddzielnej struktury zwanej tablicą.



Rysunek 3: Porównanie średniego czasu sortowania liczb.

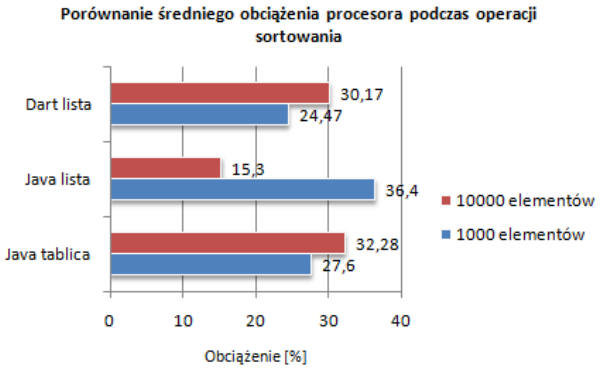
Porównując średnie czasy wykonania operacji sortowania algorytmem bąbelkowym zaprezentowane na Rysunku 3 można zauważyć, że sortowanie listy danych w języku programowania Java jest najmniej efektywne, ponieważ do zrealizowania tego zadania potrzeba 126,16 milisekund dla tysiąca elementów i aż 5037,07 milisekund dla dziesięciu tysięcy elementów co daje ponad 5 sekund. Java zdecydowanie szybciej poradziła sobie podczas sortowania zbioru liczbowego zawartego w tablicy, gdzie średni czas wykonania wynosił dla tysiąca elementów zaledwie 4,03 milisekund i tylko 412,84 milisekundy dla dziesięciu tysięcy. Język Dart podczas sortowania listy uzyskał wynik średnio o 3 razy większy niż Java podczas sortowania tablicy.

5.2. Obciążenie procesora



Rysunek 4: Porównanie średniego obciążenia procesora podczas operacji bazodanowych.

Na Rysunku 4 wyszczególniono średnie obciążenia procesora wszystkich wykonanych kryteriów testowych dotyczących operacji bazodanowych. Na jego podstawie można dostrzec, że aplikacja napisana w języku Java obciąża bardziej procesor podczas operacji create, update oraz delete niż taka sama aplikacja utworzona w języku Dart. Natomiast podczas operacji read Java osiągnęła mniejszy wynik obciążenia procesora niż język Dart.

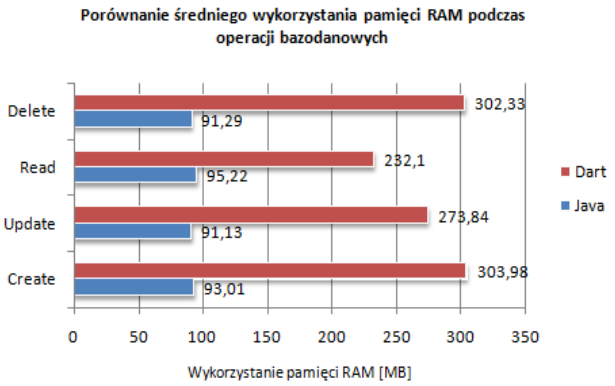


Rysunek 5: Średnie obciążenie procesora podczas sortowania.

Średnie obciążenia procesora podczas sortowania dwóch zbiorów liczbowych dla struktur danych w języku Java odpowiednio tablicy i listy oraz dla języka Dart listy prezentuje Rysunek 5. Analizując to zestawienie można zauważyć, że aplikacja zaimplementowana w języku Dart uzyskała najlepszy wynik podczas sortowania zbioru danych o małej liczbie elementów - tysiąc. Natomiast biorąc pod uwagę drugi zbiór liczb zawierający dziesięć tysięcy elementów widać jednoznacznie, że lista w języku Java w najmniejszym stopniu obciąża procesor podczas operacji z większą liczbą danych. Warto zauważyć, że podczas sortowania listy w języku Dart zarówno dla tysiąca jak i dziesięciu tysięcy elementów średnio uzyskano wyniki mniej obciążające procesor niż dla tablicy w języku Java, a to mogłoby tłumaczyć, dlaczego w języku Dart tablice zostały zastąpione listami.

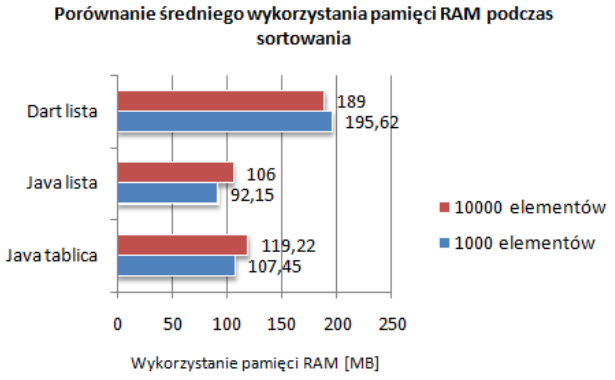
5.3. Wykorzystanie pamięci RAM

Rysunek 6 przedstawia średnie wykorzystanie pamięci RAM przez badane aplikacje podczas wykonywania operacji bazodanowych to jest create - wstawiania rekordów do bazy danych, update - edytowania, read - odczytania oraz delete - usuwania. Jednoznacznie widać, że język Java bezkonkurencyjnie osiągnął lepsze wyniki bez względu na wykonywaną operację. Różnica pomiędzy średnimi wykorzystania pamięci w danych operacjach wynosiła maksymalnie 4 megabajty. Aplikacja zaimplementowana w języku Dart wypadała gorzej podczas wykonywania procesów bazodanowych. Operacja wstawiania rekordów do bazy danych generowała największe wykorzystanie pamięci RAM, a najmniejsze w operacji odczytywania danych, lecz różnica w tym przypadku była niewielka, gdyż wynosiła zaledwie 71,88 megabajtów.



Rysunek 6: Porównanie średniego wykorzystania pamięci RAM podczas operacji bazodanowych.

Ostatnie doświadczenie jakie przeprowadzono w celu sprawdzenia wykorzystania pamięci RAM dotyczyło sortowania zbiorów liczbowych. Rysunek 7 prezentuje średnie wykorzystanie pamięci RAM dla całego eksperymentu dotyczącego sortowania danych i widać, że struktura danych zwana listą w języku Java wykorzystywała średnio najmniej pamięci RAM, a różnica liczebności zbioru nieznacznie wpłynęła na zwiększenie wykorzystania pamięci. Minimalnie większe wykorzystanie pamięci odnotowano dla tablicy w tym języku, średnio o 10 megabajtów więcej niż dla listy. Zarówno proces sortowania listy jak i tablicy w języku Java wraz ze wzrostem liczby elementów w zbiorze wykazywał zwiększone wykorzystanie pamięci RAM. Sytuacja jest odwrotna w przypadku listy dla języka Dart, ponieważ podczas sortowania liczniejszego zbioru średnie wykorzystanie pamięci było mniejsze niż dla mniejszego zbioru elementów.



Rysunek 7: Porównanie średniego wykorzystania pamięci RAM podczas sortowania.

5.4. Zużycie baterii

Przy wykorzystaniu narzędzia Android Profiler określenie wykorzystania baterii było możliwe poprzez trzy poziomy: Low, Medium oraz High, gdzie Low oznacza najmniejsze wykorzystanie, a High największe w danym momencie pracy aplikacji na urządzeniu fizycznym.

Wyniki uzyskane z przeprowadzonych scenariuszy testowych wskazują, że podczas działania autorskich aplikacji bez względu na rodzaj testu bateria była wykorzystywana na stałym poziomie Medium i nie wykazywała tendencji do zmiany tej wartości.

6. Wnioski

Analizując wyniki otrzymane ze zrealizowanych badań można spostrzec, że wybór języka programowania do tworzenia aplikacji mobilnych pomiędzy językiem Java, a Dart jest istotnym elementem, który może przyczynić się do zwiększenia popularności finalnej aplikacji. Analizując czasy wykonania programów ze scenariuszy testowych dotyczących operacji bazodanowych można zauważyć, że programy napisane w języku Java charakteryzują się większą wydajnością. Z kolei język Dart odznaczył się mniejszym obciążeniem procesora podczas tych eksperymentów. Rozpatrując poziom wykorzystania pamięci RAM należy zwrócić uwagę, że aplikacja zaimplementowana w języku Java wyraźnie osiągnęła lepsze wyniki, a co za tym idzie aplikacja mogła wykonywać zadania ze zwiększoną szybkością. Natomiast aplikacja napisana w języku Dart znacząco wykorzystywała pamięć RAM, nawet do trzech razy więcej niż bliźniacza aplikacja utworzona w języku Java. Prawdopodobnie zwiększone zapotrzebowanie na pamięć RAM mogło przyczynić się do wydłużenia czasów wykonania programów dla języka programowania Dart.

Rozważając aspekt sortowania danych również można zauważyć znaczące różnice, które dotyczą wydajności porównywanych języków programowania podczas pracy na różnych strukturach danych. Należy zwrócić uwagę, że język Java dla funkcji sortującej dane umieszczone w tablicy osiągnął najmniejszy czas wykonania, gdzie proces ten w języku Dart dla listy osiągnął czasy większe około 3 razy. Największy czas wykonania sortowania uzyskała struktura danych zwana listą w języku Java, ale jeśli chodzi o obciążenie procesora i wykorzystanie pamięci RAM można stwierdzić, że w najmniejszym stopniu obciążała urządzenie mobilne.

Pomimo tego, że autorskie aplikacje posiadały identyczne funkcjonalności to większy rozmiar osiągnęła aplikacja zaimplementowana w języku Dart.

Aby jednoznacznie stwierdzić, który język programowania jest efektywniejszy należałoby wykonać dodatkowe badania z uwzględnieniem większej liczby urządzeń posiadających różne podzespoły i systemy. Warto zwrócić uwagę, że język Dart przy współpracy z narzędziem Flutter przy jednej bazie kodu dostarcza programiście aplikację finalną działającą pod systemem Android oraz IOS. Wieloplatformowość tej technologii pozwala na zaoszczędzenie czasu oraz środków przeznaczonych do utworzenia potencjalnej aplikacji. Takiej zalety nie posiada język Java z zestawem narzędzi Android. Dlatego jest to ważny aspekt podczas rozpatrywania przydatności porównywanych języków programowania do tworzenia aplikacji mobilnych. Podsumowując dokonane analizy można stwierdzić, że wybór pomiędzy językiem Java, a Dart powinien być analityczny i najbardziej pasujący do wymagań danego projektu.

Literatura

[1] Popularność mobilnych systemów operacyjnych, https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009, [12.10.2021].

[2] Cechy Aplikacji natywnych i wieloplatformowych, https://www.uptech.team/blog/native-vs-cross-platform-app-development, [11.12.2021].

[3] P. Kotarski, K. Śledź, J. Smołka, Analiza wydajności aplikacji mobilnych przy zastosowaniu różnych narzędzi programistycznych do ich budowy, Journal of Computer Sciences Institute 6 (2018) 68-72. https://doi.org/10.35784/jcsi.642

[4] D. Gałan, K. Fisz, P. Kopniak, Porównanie aplikacji mobilnych zbudowanych przy zastosowaniu zestawów narzędzi programistycznych Android oraz Flutter z użyciem wielu kryteriów, Journal of Computer Sciences Institute 19 (2021) 107-113. https://doi.org/10.35784/jcsi.2614

[5] Android Studio, https://developer.android.com/studio [11.12.2021].

[6] D. Sulowski, G. Kozieł, Analiza porównawcza języków Kotlin i Java używanych do tworzenia aplikacji na system Android, Journal of Computer Sciences Institute 13 (2019) 354-358. https://doi.org/10.35784/jcsi.1332

[7] A.M. Hassan, JAVA and DART programming languages: Conceptual comparison, Indonesian Journal of Electrical Engineering and Computer Science 17 (2020) 845-849. http://doi.org/10.11591/ijeecs.v17.i2.pp845-849

[8] Popularność platform mobilnych na świecie, https://www.statista.com/statistics/869224/worldwide-software-developer-workinghours/, [11.01.2022].

[9] Dokumentacja języka programowania Dart, https://dart.dev/guides/language/language-tour, [10.05.2022].

[10] Ranking serwisów społeczności informatycznej, https://www.closeriq.com/blog/2020/06/top-developer-communities [11.04.2022].

[11] Serwis Stack Overflow, https://stackoverflow.com/, [11.04.2022].

[12] Serwis Git Hub, https://github.com/, [11.04.2022].

[13] C.S. Horstmann, Java. Tom I - Podstawy, Helion, 2016.

[14] Wprowadzenie do języka programowania Dart, https://en.wikipedia.org/wiki/Dart_(programming_language)#cite_note-12, [11.05.2022].

[15] Wprowadzenie do technologii Flutter, https://en.wikipedia.org/wiki/Flutter_(software)#cite_note-6, [22.04.2022].