JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 19 (2021) 107–113
Received: 23 March 2021
Accepted: 09 April 2021

A comparison of mobile applications built with the use of Android and Flutter Software Development Kits based on many criteria

Porównanie aplikacji mobilnych zbudowanych przy zastosowaniu zestawów narzędzi programistycznych Android oraz Flutter z użyciem wielu kryteriów

Damian Gałan, Konrad Fisz*, Piotr Kopniak

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

Abstract
This publication presents a multi-criteria comparison of two mobile applications built with the use of Android and Flutter SDK. The former has been implemented with Kotlin and the latter with Dart. The benchmarking process examines factors such as execution time and CPU usage during data and disk operations. During the analysis, attention was paid to the length and volume the source code, community support and the availability of libraries. The comparative analysis shows that a mobile application written using Android SDK is often not only faster and more efficient, but also has greater community support and the number of libraries available. In addition, an analysis of the source code volume showed that Flutter has more concise but more difficult to navigate code than Android.

Keywords: Android SDK; Flutter; benchmark; comparison

Streszczenie
Niniejsza publikacja przedstawia wielokryterialne porównanie dwóch aplikacji mobilnych zbudowanych przy zastosowaniu Android oraz Flutter SDK. Pierwsza z nich zaimplementowana została w języku Kotlin, natomiast druga w Dart. Proces porównawczy bada takie czynniki jak czas wykonania oraz użycie procesora podczas operacji dyskowych oraz na danych. Dodatkowo podczas analizy zwrócono uwagę na długość oraz objętość kodu, wsparcie społeczności oraz dostępność bibliotek. Analiza wykazała, że aplikacja napisana przy użyciu Android jest nie tylko często szybsza oraz wydajniejsza ale również posiada większe wsparcie społeczności oraz liczbę dostępnych bibliotek. Ponadto analiza objętości kodu źródłowego wykazała, że Flutter posiada kod bardziej zwięzły lecz trudniejszy do nawigowania od Android.

Keywords: Android SDK; Flutter; test wydajnościowy; porównanie

*Corresponding authors
Email addresses: damian.galan@pollub.edu.pl (D. Gałan), konrad.fisz@pollub.edu.pl (K. Fisz)

1. Wstęp

Od wielu lat zaobserwować można bardzo dynamiczny rozwój technologii w dziedzinie urządzeń mobilnych. Kiedyś telefon komórkowy służył jedynie do komunikacji z innym człowiekiem, dzisiaj jest wszechstronnym urządzeniem ułatwiającym codzienne funkcjonowanie. Dzięki możliwości instalacji szerokiej gamy aplikacji pełni on funkcję odtwarzacza audio i wideo, kalendarza, notatnika, budzika, nawigacji i wiele innych. Koncept inteligentnego domu również staje się coraz bardziej powszechny i nikogo nie dziwi już sterowanie urządzeniami AGD lub RTV za pomocą smartfona. Bardzo prężnie rozwijającym się rynkiem jest też segment gier mobilnych, które generują rekordowe przychody i nic nie zapowiada tego by ten trend miał się odwrócić [1].

Mając to wszystko na uwadze producenci telefonów kładą olbrzymi nacisk na to by ich produkty były coraz to bardziej wydajne, a programiści dbają o to by dostępne aplikacje były coraz lepiej zoptymalizowane i dostosowane do potrzeb użytkowników.

Jednakże, z racji istnienia wielu narzędzi pozwalających na implementację danej aplikacji, programiści często stają przed dylematem, która technologia pozwoli na uzyskanie najlepszych rezultatów.

W dzisiejszych czasach najpopularniejszymi systemami operacyjnymi na urządzenia mobilne są Android oraz iOS. Z tych dwóch częściej wybierany jest system Android, z blisko 80% udziałem w całym rynku systemów operacyjnych na urządzenia mobilne [2].

Do tworzenia aplikacji dla systemu Android najczęściej wykorzystuje się zestaw narzędzi programistycznych Android, a sam kod źródłowy najczęściej napisany jest w językach programowania Java lub Kotlin.

Z uwagi jednak na wciąż znaczący udział systemu iOS w rynku, narodziła się idea programowania międzyplatformowych aplikacji mobilnych, które posiadając jedną bazę kodu pozwalają uruchomić kod źródłowy na obu platformach. Takie rozwiązanie pozwala znacznie ograniczyć zasoby ludzkie, potrzebne do wytworzenia, a następnie utrzymania takiej aplikacji mobilnej. Na rynku mamy kilka rozwiązań pozwalających na programowanie międzyplatformowej aplikacji mobilnych, jednak ostatnim czasie coraz większą popularność zdobywa zestaw narzędzi programistycznych Flutter stworzony przez Google [3].

Swoją premierę Flutter miał w 2015 roku podczas Dart Developer Summit, pod nazwą Sky. Od tamtego czasu prace nad Flutter'em cały czas trwają, a stosunkowo niedawno pokazano możliwość wykonania tego samego kodu na systemy operacyjne Linux, Mac, Windows, Google Fuchsia, oraz w przeglądarce jako aplikacja internetowa, co spowodowało, że zainteresowanie Flutterem znacznie wzrosło.

Niemniej jednak z racji tego, że zestaw narzędzi programistycznych Flutter jest nową technologią, nie można jednoznacznie stwierdzić, czy wykorzystanie wspomnianego narzędzia jest lepszym rozwiązaniem niż sprawdzony już na rynku zestaw narzędzi programistycznych Android. Na samą decyzję wpływu nie powinna mieć jedynie wydajność samej aplikacji ale również kryteria od niej różne takie jak: długość oraz objętość kodu źródłowego, wsparcie społeczności czy dostępność bibliotek. Przedstawiony artykuł podejmuje tematykę porównania wielokryterialnego dwóch funkcjonalnie identycznych aplikacji mobilnych stworzonych z wykorzystaniem zestawów narzędzi programistycznych Android oraz Flutter w oparciu o wyżej wspomniane kryteria oceny.

2. Przegląd literatury

Przed przystąpieniem do wykonania porównania wielokryterialnego zestawów narzędzi programistycznych Android oraz Flutter przeprowadzono badanie literaturowe na temat omawianych narzędzi. Przegląd literatury wykazał, że temat Fluttera nie jest dostatecznie wyczerpany z uwagi na to, że jest on technologią, która jest obecna na rynku od niedawna. Niemniej jednak bazy naukowe są bogate w literaturę na temat porównania mobilnych technolgii natywnych z wieloplatformowymi innymi niż Flutter, co jest istotne z punktu widzenia przedstawionego tematu pracy.

W publikacji G. Kozieła oraz D. Sulowskiego *Analiza porównawcza języków Kotlin i Java używanych do tworzenia aplikacji na system Android* [4] przeprowadzono analizę porównawczą dwóch języków programowania Java oraz Kotlin. W celu porównania obu tych języków wybrano następujące kryteria: obciążenie procesora oraz pamięci RAM, czas kompilacji i wykonania programu. Dodatkowo, języki poddano analizie pod względem struktury kodu, obsługi bazy danych, dostępności bibliotek, popularności oraz wsparcia społeczności. Na potrzeby artykułu stworzona została specjalna aplikacja mobilna, której zadaniem było sortowanie zbiorów liczb z wykorzystaniem kilku algorytmów oraz wyświetlenie prostej animacji. Autorzy artykułu stwierdzają, iż nie ma wyraźnej różnicy pomiędzy językami i nie można wskazać który język programowania jest lepszym wyborem przy tworzeniu aplikacji mobilnych dla systemu Android. Przy większości kryteriów istotną kwestią było to, jaki telefon był używany podczas testów. Zauważalne różnice zostały stwierdzone przy określaniu takich aspektów jak struktura kodu czy popularność. Autorzy artykułu doszli do wniosków, iż język Java jest lepszym wyborem dla początkujących programistów, z powodu większej ilości materiałów pomocniczych. Natomiast Kotlin może być lepszym wyborem dla osób z doświadczeniem w programowaniu aplikacji mobilnych.

Trzy kolejne publikacje, zawierają porównania wielu różnych narzędzi programistycznych do budowy aplikacji mobilnych. W artykule P. Kotarskiego, K. Śledzia oraz J. Smołki *Analiza wydajności aplikacji mobilnych przy zastosowaniu różnych narzędzi programistycznych do ich budowy* [5] zbadano wydajność aplikacji mobilnych przy zastosowaniu różnych narzędzi programistycznych takich Android SDK z wykorzystaniem języka Java, Android NDK, Xamarin oraz Apache Cordova. Dwa pierwsze narzędzia pozwalają na stworzenie aplikacji dla urządzeń wyposażonych z system Android. Kolejne z nich, Xamarin, pozwala na tworzenie aplikacji dla systemu Windows Phone. Apache Cordova pozwala na wykonanie aplikacji na urządzenia mobilne z wykorzystaniem technologii webowych. W tej pracy do zadań zaimplementowanych aplikacji należało: sortowanie tablicy oraz odczyt i zapis danych do pliku. Podczas przeprowadzonych badań, autorzy przeprowadzonej analizy nie mogli jednoznacznie stwierdzić, które z wcześniej wymienionych narzędzi pozwala na stworzenie najwydajniejszej aplikacji.

W pozostałych dwóch z trzech wspomnianych publikacji dokonano porównania aplikacji wykonanych przy użyciu Xamarina, Androida, iOS oraz postawiono pytanie czy zalecane jest użycie technologii wieloplatformowych w tworzeniu aplikacji mobilnych. W pierwszej z nich *The comparison of native apps performance on iOS (Swift) and Android with cross-platform application - Xamarin: student project* [6] autorzy stwierdzają, że aplikacja napisana w wieloplatformowej technologii Xamarin może być tak samo wydajna jak aplikacja natywna napisana w Android lub iOS i wybór stosownej technologii powinien być uzależniony od funkcjonalności wytwarzanej aplikacji mobilnej. Ponadto autorzy zaznaczają, że Xamarin Forms jest także w trakcie aktywnego rozwijania, więc nie jest on produktem całkowicie gotowym w środowisku produkcyjnym. Analiza wyników jakie zostały otrzymane w publikacji *Performance analysis of native and cross-platform mobile applications* [7] pokazały natomiast, iż w większości przypadków lepsze wyniki pod względem wydajności osiągały aplikacje natywne niż ta zaimplementowana w technologii Xamarin. Ważnym faktem stwierdzonym w obu publikacjach jest to, że Xamarin Forms znacznie upraszcza proces wytwarzania oprogramowania i pozwala skrócić czas potrzebny na wytworzenie aplikacji mobilnej, jednak wiedza na temat natywnych technologii jest konieczna by go używać.

Wszystkie z opisanych wyżej publikacji pozwalają wyciągnąć wniosek, iż przeprowadzanie badań porównujących różne technologie do tworzenia aplikacji mobilnych jest istotnym zagadnieniem. Takie badania pozwalają zgłębić wiedzę dotyczącą zagadnień wydajnościowych jak i poza wydajnościowych jakie mogą mieć wpływ przy tworzeniu aplikacji na urządzenia mobilne oraz mogą pomóc przy wyborze z wykorzysta-

niem jakiej technologii tworzenie takich aplikacji jest najbardziej optymalne. Dodatkowo, nie znaleziono publikacji która traktowałaby o porównaniu narzędzia Flutter z innym narzędziem, co było powodem dla którego autorzy niniejszego artykułu podjęli się tejże analizy.

3. Metoda badawcza

W celu przeprowadzenia badań wydajności utworzono dwie funkcjonalnie identyczne aplikacje mobilne przeznaczone na systemy Android przy zastosowaniu zestawów narzędzi programistycznych Android oraz Flutter.

Na potrzeby porównania wydajności tych dwóch zestawów narzędzi programistycznych przygotowano następujące scenariusze testowe:

- Sortowanie 10000, 100000 oraz 1000000 liczb naturalnych algorytmem Dual-Pivot Quicksort.
- Zapisanie do lokalnej bazy danych 100, 500 oraz 1000 rekordów.
- Odczytanie z lokalnej bazy danych 100, 500 oraz 1000 rekordów.
- Zapisanie 1000, 10000 oraz 100000 znaków alfanumerycznych do pliku tekstowego.
- Odczytanie 1000, 10000 oraz 100000 znaków alfanumerycznych z pliku tekstowego.

3.1. Środowisko badawcze

Do zbadania obciążenia procesora podczas operacji sortowania danych wykorzystano moduł Android Profiler oraz narzędzie DevTools. Natomiast do zbadania czasu wykonania scenariuszy testowych opisanych w punkcie 3 wykorzystane zostały dwie funkcjonalnie identyczne aplikacje testowe monitorujące oraz zapisujące do bazy danych czasy wykonania poszczególnych operacji.

Szczegółowa specyfikacja urządzenia fizycznego na którym uruchomiono aplikacje testowe przedstawiona została w Tabeli 1.

Tabela 1: Parametry techniczne urządzenia testowego

Model	Motorola G8 Power
Procesor	Qualcomm Snapdragon 665
Taktowanie procesora	2000 MHz
Liczba rdzeni	8
Pamięć RAM	4 GB
Pamięć ROM	64 GB
System operacyjny	Android 10

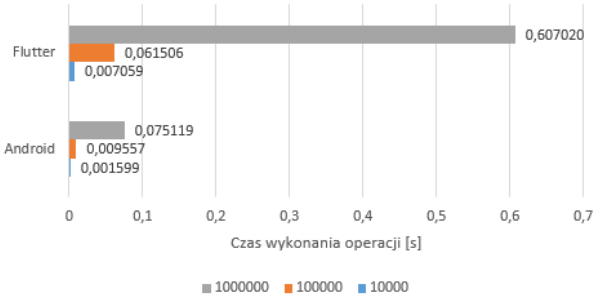
4. Analiza wyników badań wydajnościowych

W punkcie 4 zamieszczono wyniki przeprowadzonych badań wydajnościowych. Wszystkie czasy badań zostały przedstawione w sekundach.

Wykresy prezentują średnią trzydziestokrotnego wykonania poszczególnych eksperymentów na podstawie scenariuszy testowych opisanych w punkcie 3.

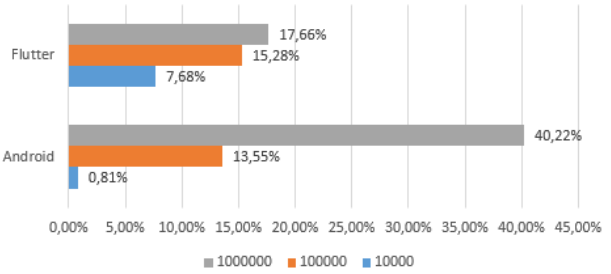
4.1. Sortowanie danych

Rysunek 1 przedstawia średni czas sortowania 10000, 100000 oraz 1000000 liczb naturalnych. Na wykresie można zauważyć, że aplikacja zbudowana w Androidzie wykonała te operacje szybciej od aplikacji wykonanej we Flutterze. Znaczącą różnicę widać przy każdej liczbie sortowanych liczb. Przy tych próbach aplikacja napisana w Androidzie okazuje się około siedmiokrotnie szybsza.



Rysunek 1: Zestawienie wyników pomiaru czasu wykonania operacji sortowania elementów.

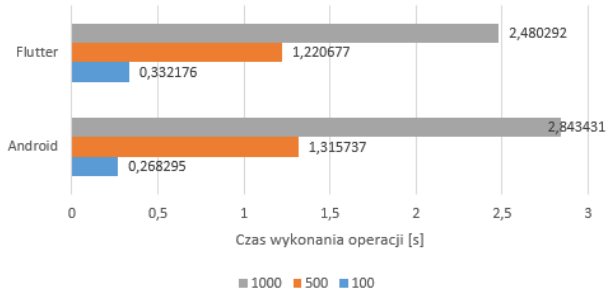
Na Rysunku 2 zamieszczono wyniki porównania obciążenia procesora podczas operacji sortowania elementów w strukturze danych. Na przedstawionym wykresie można zauważyć, że to Android mniej obciążył procesor podczas operacji sortowania. Warto podkreślić jednak dużą przewagę po stronie Fluttera podczas sortowania 1000000 liczb. W tym przypadku Flutter poradził sobie znacznie lepiej.



Rysunek 2: Zestawienie wyników pomiaru użycia procesora podczas wykonania operacji sortowania elementów.

4.2. Zapis do bazy danych

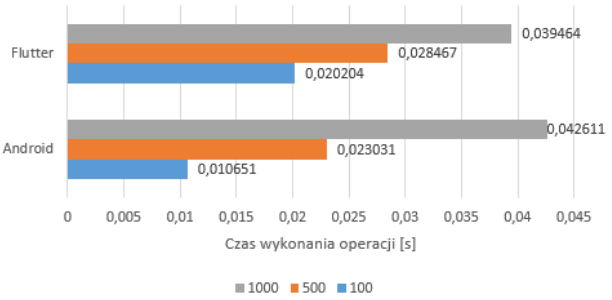
Na Rysunku 3 zaprezentowano średni czas zapisu danych do lokalnej bazy danych wykorzystującej silnik baz danych SQLite. Na podstawie przedstawionego wykresu, można wywnioskować, że aplikacja zaimplementowana przy użyciu Flutter SDK lepiej radziła sobie od aplikacji zaimplementowanej w Androidzie przy zapisie większej ilości danych. Android natomiast charakteryzuje się szybszym zapisem danych do bazy przy niewielkiej ilości.



Rysunek 3: Zestawienie wyników pomiaru czasu wykonania operacji zapisu do bazy danych.

4.3. Odczyt z bazy danych

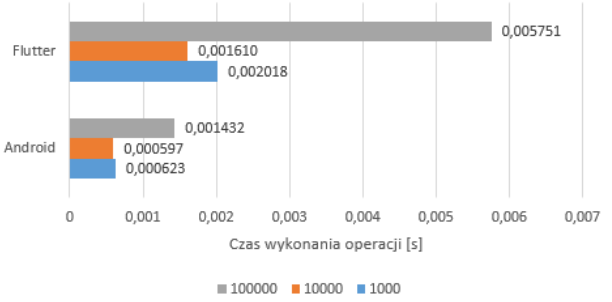
Wykres na Rysunku 4 przedstawia średni czas odczytu danych z lokalnej bazy danych opartej o silnik baz danych SQLite. Podobnie jak przy zapisie do bazy, podczas odczytu danych z bazy danych Flutter również okazał się dużo szybszy od Androida przy większej ilości danych. Przewaga Androida widoczna jest przy zapisie 100 elementów.



Rysunek 4: Zestawienie wyników pomiaru czasu wykonania operacji odczytu z bazy danych.

4.4. Zapis do pliku

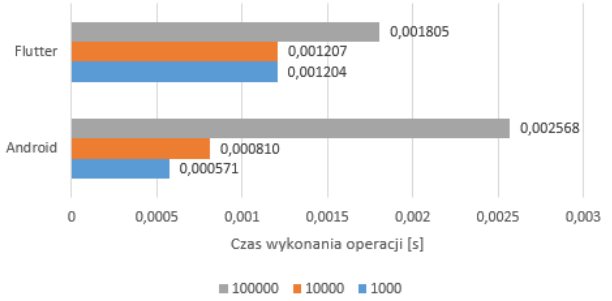
Wykres na Rysunku 5 prezentuje średni czas zapisu 1000, 10000, 100000 znaków alfanumerycznych do pliku tekstowego. Przy tym badaniu Android lepiej poradził sobie wykazując się mniejszymi średnimi czasami zapisu. Dużą przewagę Androida widać szczególnie przy zapisie każdej ilości znaków. Średnio Android wykonuje operację dwa razy szybciej.



Rysunek 5: Zestawienie wyników pomiaru czasu wykonania operacji zapisu do pliku.

4.5. Odczyt z pliku

Rysunek 5 przedstawia średni czas odczytu 1000, 10000 oraz 100000 znaków alfanumerycznych z pliku tekstowego. Z wykresu widać, że Android średnio szybciej przeprowadził badaną operację. Największa różnica widoczna jest przy 1000 i 10000 znakach. Podczas odczytu 100000 znaków to Flutter wykonał te operacje średnio w krótszym czasie lecz jego przewaga nad Androidem nie jest znacząca.



Rysunek 6: Zestawienie wyników pomiaru czasu wykonania operacji odczytu z pliku. odczytu z pliku.

5. Analiza wyników badań poza wydajnościowych

Przy tworzeniu aplikacji mobilnych istotną kwestią jest dobór odpowiedniej technologii, która pozwoli na uzyskanie jak najlepszych wyników pod względem wydajności. Jednakże, przy doborze technologii nie mniej ważne są kwestie różne od wydajności, która mają istotny wpływ na szybkość oraz jakość wytwarzanego oprogramowania. Dlatego podczas porównania Androida oraz Fluttera zbadano również cechy różne inne niż wydajność, tzn.:

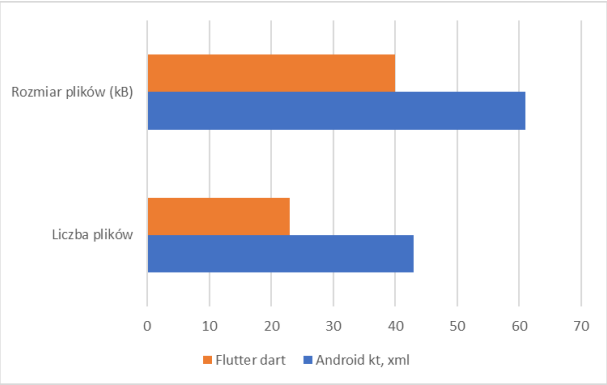
- liczba linii kodu źródłowego,
- dostępność bibliotek,
- wsparcie społeczności.

5.1. Liczba linii kodu źródłowego

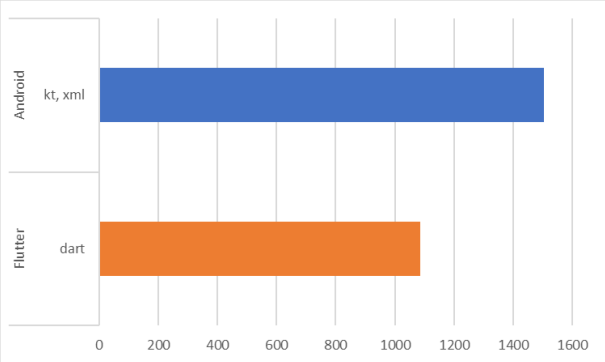
Do zbadania objętości kodu źródłowego posłużono się aplikacjami testowymi zaimplementowanymi w Android oraz Flutter SDK. Były to te same aplikacje, których użyto w porównaniu wydajnościowym. Badanie przeprowadzone zostało z użyciem wtyczki o nazwie *Statistic* dostępnej dla zintegrowanego środowiska programistycznego *Android Studio*.

Na potrzeby wykonania tego porównania analizie poddano pliki o rozszerzeniu:

- .dart - kod źródłowy aplikacji zaimplementowanej we Flutter SDK, który odpowiada zarówno za logikę jak i warstwę prezentacji
- .kt - kod źródłowy aplikacji zaimplementowanej w Android SDK odpowiadający za logikę
- .xml - kod źródłowy aplikacji zaimplementowanej w Android SDK odpowiadający za warstwę prezentacji oraz konfigurację



Rysunek 7: Porównanie rozmiaru oraz liczby plików zawierających kod źródłowy aplikacji testowych.

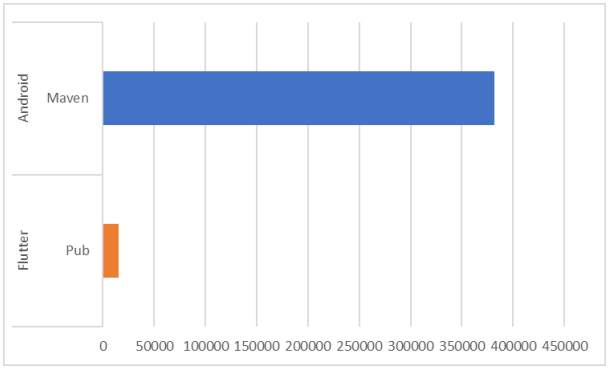


Rysunek 8: Porównanie liczby linii kodu.

Na podstawie analizy, której wyniki zamieszczono na wykresach zamieszczonych na Rysunku 7 oraz Rysunku 8 stwierdzono, że to Flutter posiada bardziej zwięzły kod. Wynikać to może z faktu, że zarówno logikę jak i warstwę prezentacji poszczególnych widoków zakodowano w tych samych plikach. W Androidzie natomiast istnieje jasny podział na kod odpowiadający za warstwę logiczną oraz prezentacji. Z praktycznego punktu widzenia może być to postrzegane jako zaleta ponieważ dzięki temu istnieje bardziej klarowny podział odpowiedzialności pomiędzy plikami źródłowymi. Podział ten ułatwia programiście poruszania się po kodzie źródłowym, szczególnie w zaawansowanych projektach programistycznych.

5.2. Dostępność bibliotek

W celu określenia liczby bibliotek dostępnych dla technologii Flutter wykorzystano repozytorium *Pub*, które jest menedżerem pakietów dla języka programowania Dart. Zawiera on biblioteki wielokrotnego użytku i pakiety dla programów Flutter, AngularDart i ogólnych programów Dart. Natomiast do zbadania dostępność bibliotek dla Android użyto repozytorium *Maven*, w którym przechowywane są wszystkie pliki z rozszerzeniem .jar bibliotek, wtyczki lub inne artefakty specyficzne dla projektów opartych na technologii Java.

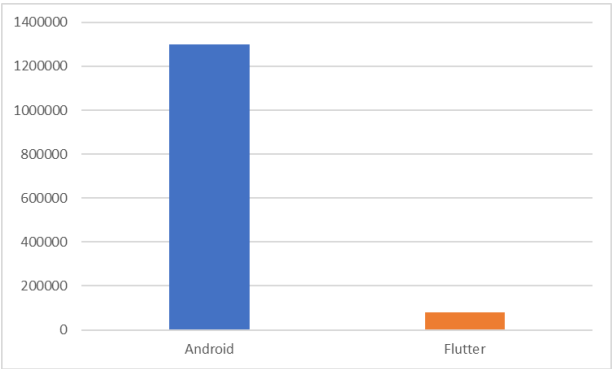


Rysunek 9: Porównanie liczby dostępnych bibliotek.

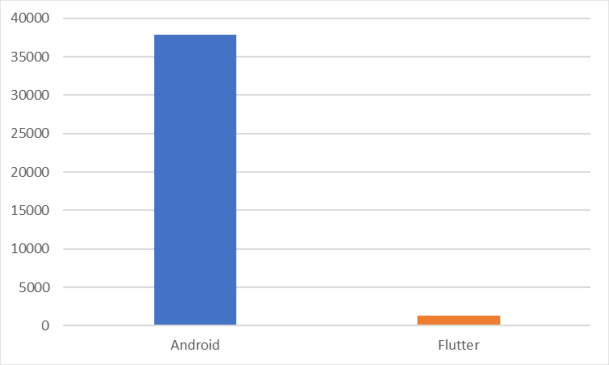
Wyniki przeprowadzonego porównania pod kątem liczby bibliotek przedstawiono na wykresie na Rysunku 9. Na ich podstawie widać, że repozytorium Maven, z którego korzysta Android jest znacznie bardziej liczne. Pod uwagę wziąć należy jednak fakt, że w przypadku Androida trudno jest określić jaki procent całości dostępnych bibliotek znajdzie swoje zastosowanie podczas wytwarzania aplikacji mobilnej przy użyciu Android SDK ponieważ w repozytorium Maven znajdują się biblioteki dla wszystkich technologii opartych o wirtualną maszynę Java. Niemniej jednak, nawet mając ten fakt na uwadze przewaga po stronie Androida wydaję się być na tyle znacząca, że to właśnie technologię Android można uznać za tę z większą liczbą dostępnych bibliotek.

5.3. Wsparcie społeczności

Do analizy wsparcia społeczności wykorzystano dane dostępne na serwisie społecznościowym *Stack Overflow*, na którym programiści zadają pytania dotyczące szeroko pojętego wytwarzania oprogramowania oraz na serwisie internetowym *Reddit*. Serwis ten prezentuje linki do różnorodnych informacji, które ukazały się w Internecie. Dane zebrane na Rysunku 10 oraz 11 przedstawiają te zapytania użytkowników, które zostały oznaczone pojęciem „Android” oraz „Flutter” w wyżej wymienionych serwisach.



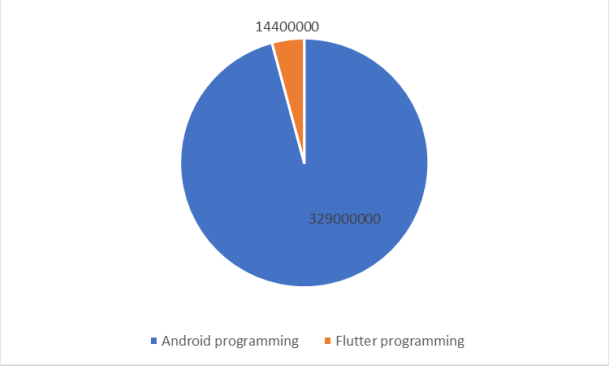
Rysunek 10: Porównanie liczby wyników na Stack Overflow.



Rysunek 11: Porównanie liczby wyników na Reddit.

Na podstawie otrzymanych wyników widać, że na zapytanie „Android" otrzymujemy znacznie więcej wyników niż na zapytanie „Flutter". Różnica na korzyść Androida jest bardzo znacząca ponieważ wynosi około 1,2 mln na Stack Overflow oraz 36,5 tys. na Reddit. Należy mieć jednak na uwadze fakt, iż w przypadku pojęcia „Android" wyszukiwarki pokazują nie tylko informacje o samym programowaniu w Android SDK, ale również wątki związane z systemem operacyjnym Android. Niestety obecnie żaden z wymienionych portali nie posiada na tyle zaawansowanej funkcji filtracji, która pozwalałaby by precyzyjne zawęzić tematykę prezentowanych wyników.

Oprócz sprawdzenia liczby zwracanych informacji na temat badanych zestawów narzędzi programistycznych na Stack Overflow oraz Reddit, autorzy postanowili zbadać ilość wyników zwracanych przez wyszukiwarkę Google odpowiednio dla pojęć „Flutter programming" oraz „Android programming". Wyniki z przeprowadzonego badania przedstawiono na wykresie kołowym na Rysunku 12.



Rysunek 12: Porównanie liczby wyników w wyszukiwarce Google.

Poddając analizie powyższe wyniki, stwierdzono, że to dla frazy „Android programming" wyszukiwarka Google zwraca znacznie więcej wyników.

Biorąc pod uwagę również poprzednie badanie dotyczące liczby wyników na Stack Overflow oraz Reddit stwierdzono, że przewaga na korzyść Android jeżeli chodzi o to kryterium porównawcze jest znacząca i to w przypadku wytwarzania oprogramowania za pomocą Android SDK programista może liczyć na większe wsparcie społeczności.

6. Wnioski

Celem przeprowadzonych badań było porównanie wielokryterialne aplikacji mobilnych zbudowanych przy zastosowaniu zestawów narzędzi programistycznych Android oraz Flutter.

Pierwszym z badanych kryteriów było obciążenie procesora podczas sortowania danych. W przeprowadzonym badaniu stwierdzono, iż aplikacja zbudowana za pomocą Androida w większości przypadków, mniej obciąża procesor podczas tej operacji. Podczas sortowania danych zmierzono również czasy, które Android oraz Flutter potrzebowały na jego przeprowadzenie. Android szybciej posortował dane w strukturze danych, niezależnie od jej wielkości.

Drugim kryterium był zapis oraz odczyt danych z lokalnej bazy danych. Po przeanalizowaniu wyników zaobserwowano, badania nie pozwalają na wysunięcie jednoznacznego wniosku. Przy małej ilości danych to Android charakteryzuje się krótszym czasem, natomiast Flutter lepiej poradził sobie z większą ilością danych.

Aplikacja zaimplementowana w Android SDK okazała się również bardziej wydajna w większości badań zapisu oraz odczytu danych z pliku tekstowego. Biorąc pod uwagę to oraz poprzednie przeprowadzone badanie dotyczące zapisu oraz odczytu z lokalnej bazy danych, nie jest możliwe jednoznaczne stwierdzenie, która z badanych technologii szybciej zapisuje lub odczytuje dane zapisane w pamięci lokalnej urządzenia.

Kolejne badania dotyczyły cech innych niż wydajność. Bazując na analizie kodu źródłowego aplikacji testowych stwierdzono, że Flutter posiada kod bardziej zwięzły lecz trudniejszy do nawigowania od programu natywnego ze względu na brak klarownego podziału pomiędzy warstwą logiki a prezentacji. Łatwość poruszania się po kodzie źródłowym może mieć kluczowe znaczenie szczególnie dla zaawansowanych projektów programistycznych dlatego pomimo mniejszej objętości kodu to Android zdaniem autorów prezentuje bardziej przemyślane podejście do podziału obowiązków pomiędzy plikami od Flutter.

Dwa kolejne kryteria różne od wydajności dotyczyły liczby dostępnych bibliotek oraz wsparcia społeczności. Android SDK okazał się narzędziem z dużo większym wsparciem społeczności programistycznej, co z kolei mogło mieć pewne przełożenie na liczbę dostępnych bibliotek zewnętrznych, które w większości są tworzone a następnie publikowane przez samą społeczność.

Na podstawie przeprowadzonych badań można stwierdzić, iż aplikacja zaimplementowana z wykorzystaniem zestawu narzędzi programistycznych Android jest często nie tylko podobnie wydajna lub nawet wydajniejsza ale również posiada lepiej zorganizowany kod źródłowy, większe wsparcie społeczności oraz liczbę dostępnych bibliotek zewnętrznych od aplikacji zbudowanej przy użyciu Flutter SDK. Mając to wszystko na uwadze stwierdzono, że to Android SDK jest obecnie bardziej zalecanym zestawem narzędzi programistycznych do wytwarzania oprogramowania na systemy operacyjne Android od Flutter SDK.

Literatura

[1] M. A. Khaled, R. S. Tariq, S. Khaled, Mobile Gaming Trends and Revenue Models, Springer International Publishing (2016).

[2] M. Naldi, Concentration in the mobile operating systems market, Universit‘y of Rome Tor Vergata (2016).

[3] M. Napoli, Beginning Flutter: A Hands On Guide to App Development, John Wiley & Sons, 2019.

[4] 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

[5] 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

[6] D. Dobrzański, W. Zabierowski, The comparison of native apps performance on iOS (Swift) and Android with cross-platform application - Xamarin: student project, International Journal of Microelectronics and Computer Science 8 (2018) 112-116.

[7] 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. https://doi.org/10.5604/01.3001.0010.4838