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)

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].

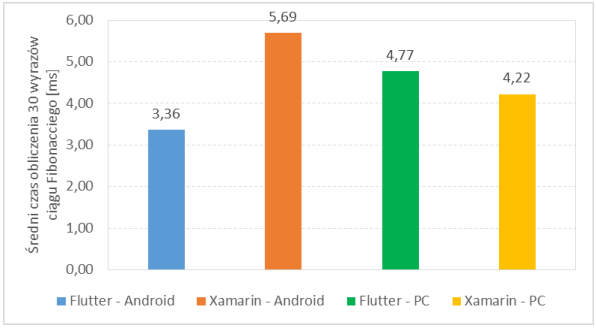
3.3. Kryteria porównawcze i scenariusze testowe

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.

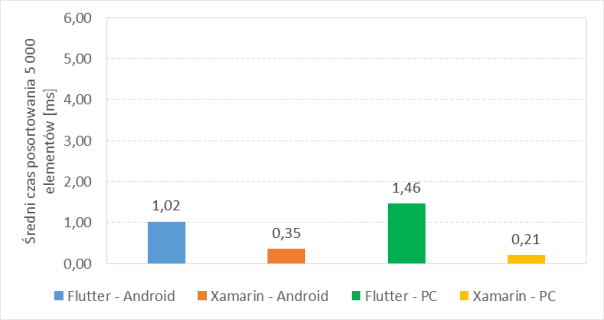


Rysunek 1: Średni czas obliczenia trzydziestego wyrazu ciągu Fibonacciego.

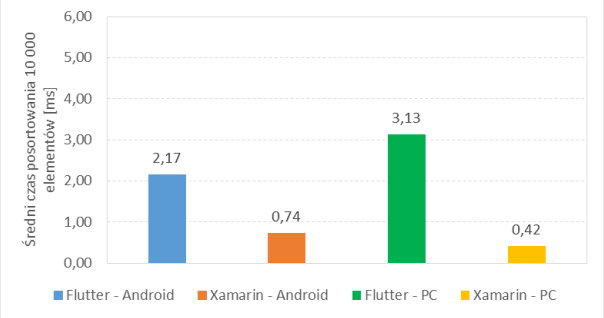
Scenariusz, w którym wyliczano trzydziesty wyraz ciągu miał za zadanie sprawdzić, jak aplikacje bazujące na danym szkielecie programistycznym radzą sobie z zadaniami liniowymi. Na Rysunku 1 widoczna jest duża różnica między wynikami dla aplikacji działają-

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.

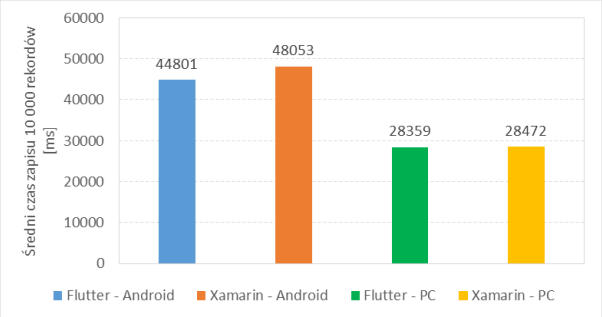


Rysunek 2: Średnie czasy sortowania listy 5 000 elementów algorytmem MergeSort.



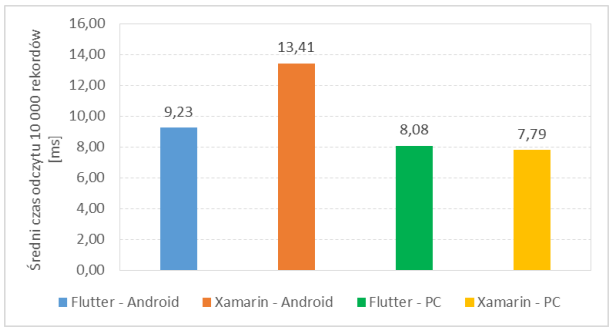
Rysunek 3: Średnie czasy sortowania listy 10 000 elementów algorytmem MergeSort.

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.



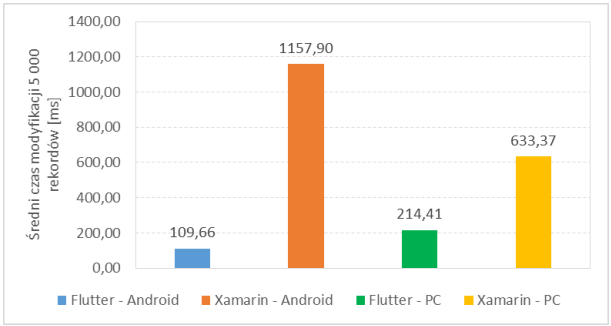
Rysunek 4: Średnie czasy wykonania 10 000 operacji zapisu rekordów do bazy danych.

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.



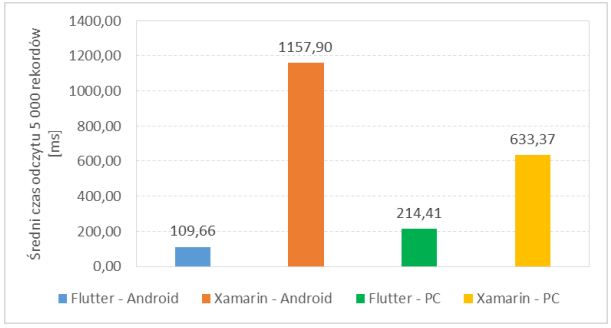
Rysunek 5: Średnie czasy odczytu pierwszych 10 000 rekordów z bazy danych.

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.



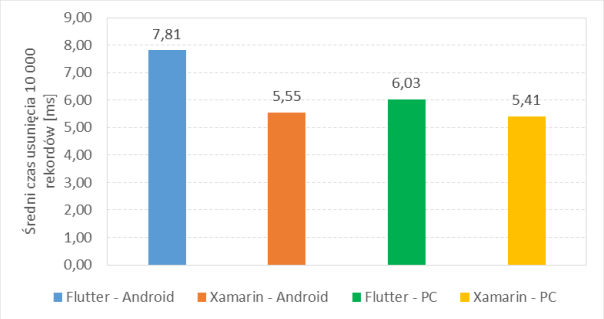
Rysunek 6: Średnie czasy modyfikacji 5 000 losowo wybranych rekordów z bazy danych.

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].