JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 32 (2024) 199–204
Received: 1 June 2024
Accepted: 3 July 2024

Comparison of performance of Xamarin and Flutter cross-platform frameworks

Porównanie wydajności wieloplatformowych frameworków Xamarin i Flutter

Piotr Kowalczyk*, Mariusz Dzieńkowski

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

Abstract
This article conducts a performance comparative study of two cross-platform development frameworks: Flutter and Xamarin. For this purpose, two test applications running on Android were developed, implementing the same functions, i.e. CRUD operations on the database and sorting an array of objects by the Merge Sort algorithm. Tests were performed on an emulator and on a physical device. The sorted arrays had different sizes, and database operations were performed for different sizes of data sets. The metric used for comparisons was the average implementation time of a given operation. The analysis of the research results showed that Xamarin was definitely more efficient in three CRUD operations, while Flutter performed computational operations, which in this case was sorting, much faster.

Keywords: Flutter; Xamarin; cross-platform frameworks; comparative analysis

Streszczenie
W ramach artykułu przeprowadzono wydajnościowe badania porównawcze dwóch wieloplatformowych szkieletów programistycznych Flutter oraz Xamarin. W tym celu opracowano dwie aplikacje testowe pracujące pod kontrolą systemu Android, realizujące te same funkcje tzn. operacje CRUD na bazie danych oraz sortowanie przez scalanie tablic obiektów. Testy przeprowadzono na emulatorze oraz na urządzeniu fizycznym. Sortowane tablice miały różne rozmiary, a operacje na bazie były realizowane dla różnej wielkości zbiorów danych. Metryką wykorzystaną do porównań był średni czas realizacji danej operacji. Analiza wyników badań pokazała, że Xamarin był zdecydowanie wydajniejszy w trzech operacjach CRUD, natomiast Flutter znacznie szybciej wykonywał operacje obliczeniowe, którą było w tym przypadku sortowanie.

Słowa kluczowe: Flutter; Xamarin; wieloplatformowe szkielety programistyczne; analiza porównawcza

*Corresponding author
Email address: **piotr.kowalczyk2@pollub.edu.pl** (P. Kowalczyk)

1. Wstęp

Z roku na rok coraz większą uwagę zwraca się na potrzeby użytkowników urządzeń mobilnych takich jak telefony czy tablety. Obecnie w Google Play Store jest dostępne ponad 2,5 miliona aplikacji [1], natomiast na urządzenia z systemem iOS jest ich około 2,2 miliona [2]. Taki rozwój różnych systemów mobilnych sprawia, że jest to bardzo dobry rynek dla potencjalnych producentów oprogramowania. Każdy z systemów ma swoją natywną technologię, w której można tworzyć odpowiednie aplikacje. Jednak, jeśli producent chciałby opublikować swoją aplikację na dwa różne systemy, musiałby użyć dwóch różnych technologii do jej implementacji. Proces powstawania takiego oprogramowania mógłby się znacznie wydłużyć i pochłonąć większe koszty. Rozwiązaniem tego problemu jest używanie szkieletów programistycznych, które umożliwiają pisanie aplikacji wieloplatformowych czyli takich, które mogą zostać uruchomione zarówno na urządzeniach z systemem Android, jak i iOS. Szkielet programistyczny jest to specjalne oprogramowanie, które dostarcza różnego rodzaju komponenty oraz biblioteki ułatwiające i przyśpieszające tworzenie aplikacji [3].

W niniejszej pracy zostały porównane dwa szkielety programistyczne Xamarin oraz Flutter. Xamarin jest szkieletem, który powstał w 2011 roku, a 5 lat później został wykupiony przez firmę Microsoft [4]. Używa on platformy .NET, aby uruchamiać programy dla systemów iOS, Android oraz Windows. Flutter natomiast oprócz wyżej wymienionych systemów obsługuje również Linuxa oraz MacOSa.

Celem pracy było porównanie wydajności dwóch szkieletów programistycznych Xamarin i Flutter na urządzeniach mobilnych oraz emulatorach. Dokonano tego przy pomocy dwóch aplikacji testowych, które zostały zaimplementowane do przeprowadzenia badań. Obie aplikacje zawierały te same funkcjonalności: realizowały operacje CRUD na bazie danych oraz sortowały algorytmem MergeSort różnej wielkości tablice. Do porównań wydajności użyto czasu wykonania poszczególnych zadań.

2. Przegląd literatury

W artykule [3] dokonano analizy dwóch wieloplatformowych szkieletów programistycznych: Flutter i Xamarin. Autorzy utworzyli identyczne aplikacje testowe dla systemów Windows i Android. Każda aplikacja oferowała

takie same funkcje jak: obliczanie 30. elementu ciągu Fibonacciego, sortowanie list o rozmiarach pięć i dziesięć tysięcy elementów za pomocą algorytmu scalania, a także operacje CRUD na bazach danych. Każde z tych zadań zostało powtórzone dziesięć tysięcy razy, a do analizy porównawczej wykorzystano średnie czasy wykonania poszczególnych operacji. Na podstawie uzyskanych wyników badań autorzy nie mogli jednoznacznie stwierdzić, która platforma jest lepsza. Testy wydajnościowe frameworków programistycznych wykazały, że Flutter osiągał lepsze wyniki na urządzeniach z systemem Android, podczas gdy na komputerach osobistych wyraźnie przeważał Xamarin. W scenariuszach związanych z sortowaniem tablicy Xamarin uzyskał lepsze wyniki od Fluttera zarówno na urządzeniach mobilnych jak i na komputerze. Badania dotyczące pracy z bazą danych obejmowały pięć scenariuszy. W trzech z nich Flutter osiągnął lepsze wyniki, szczególnie w zadaniach takich jak zapis danych, wyszukiwanie rekordu i modyfikacja rekordu. W dwóch pozostałych scenariuszach, które obejmowały odczyt wszystkich rekordów oraz usunięcie wszystkich rekordów, lepsze wyniki uzyskały aplikacje stworzone za pomocą Xamarina.

W artykule [5] autorzy przedstawiają analizę wieloplatformowych szkieletów programistycznych m.in. Flutter i Xamarin pod kątem funkcjonalności, nakładu pracy, procesu programowania i wydajności. Do badań posłużyła aplikacja wirtualnego portfela, która zawierała interfejs użytkownika, komunikowała się przez sieć oraz miała zaimplementowane różne algorytmy. Zdefiniowano cztery kryteria porównawcze, które podlegały analizie:

- Projekt interfejsu użytkownika – badano jak przyjazne i wydajne są narzędzia do projektowania interfejsu użytkownika.
- Funkcjonalność – sprawdzano czy szkielet programistyczny spełnia założenia, które były wymagane w projekcie, np. kryptografia.
- Wydajność – analizowano użycie procesora oraz pamięci, a także rozmiar wybranych pakietów.
- Obciążenie pracą – dokonywano pomiaru liczby linii kodu napisanego podczas tworzenia projektu.

Na podstawie badań autorzy stwierdzili, że nie ma jednego najlepszego szkieletu programistycznego do tworzenia aplikacji wieloplatformowych. Flutter jest zalecany dla osób, które znają Javę lub C++ lub kiedy muszą przenieść projekt z natywnego szkieletu programistycznego na wieloplatformowy. Natomiast Xamarin jest odpowiedni dla osób, które programują na platformie .NET oraz dla tych, którym zależy na wydajności aplikacji wieloplatformowych.

Artykuł [6] porównuje wydajność platformy Xamarin i aplikacji natywnych na systemach operacyjnych Android i iOS. Przygotowano pięć testów oceniających wydajność aplikacji pod dużym obciążeniem komponentów urządzenia, takich jak procesor i pamięć. Do testów wykorzystano kilka różnych urządzeń. Jedną z głównych zalet tworzenia aplikacji w Xamarinie jest możliwość użycia jednej technologii do stworzenia całego projektu, w przeciwieństwie do aplikacji natywnych, które wymagają odrębnego pisania kodu dla Androida i iOS w różnych technologiach. W efekcie, tworzenie aplikacji natywnych wiąże się z koniecznością opracowania dwóch osobnych projektów, co znacząco wydłuża czas realizacji całego projektu. W badaniu przeprowadzono następujące testy:

- Operacje na bazie danych – wykorzystano silnik SQLite do zbadania operacji dodawania, aktualizacji oraz usuwania danych z bazy.
- Sortowanie danych – zbadano czas potrzebny na posortowanie tablicy zawierającej 50000 elementów.
- Odczytywanie plików i algorytmy kryptograficzne – wykonano szyfrowanie 3000 linii w pliku za pomocą różnych algorytmów kryptograficznych.
- Stosowanie efektów do obrazu – nakładanie na obraz rozdzielczości 4000 x 4000 takich efektów jak skala szarości, rozmycie Gaussa i obrót o 180 stopni.
- Test interfejsu – sprawdzenie wydajności odświeżania danych. Test wykonano 10000 razy na 3 różnych elementach.

W badaniach dotyczących operacji na bazie danych w systemie Android, Xamarin uzyskał zdecydowanie gorsze wyniki w porównaniu do rozwiązań natywnych, które okazały się znacznie szybsze od aplikacji wieloplatformowych. Jednakże Xamarin osiągnął lepsze wyniki niż natywne aplikacje na iOS, choć różnica nie była duża. W algorytmach sortowania natywne rozwiązanie na Androida również okazało się zdecydowanie lepsze. Natomiast w przypadku systemu iOS, lepiej poradziło sobie rozwiązanie natywne. Najprawdopodobniej wynika to z faktu, że dla kodu C# konieczne jest uruchomienie dodatkowej maszyny wirtualnej. W przypadku algorytmów kryptograficznych na systemie Android wyniki były silnie uzależnione od urządzenia, na którym przeprowadzano testy. Na jednym smartfonie lepsze wyniki osiągnęło rozwiązanie natywne, podczas gdy na innych lepszy okazał się Xamarin. Natomiast na systemie iOS, natywne rozwiązanie zdecydowanie przewyższało wydajnością Xamarin. W przypadku stosowania efektów obrazu dla obu systemów operacyjnych zdecydowanie wydajniejszy był Xamarin. Wynika to z faktu, że zawiera on wiele bibliotek, które posiadają zaimplementowane w sposób bardziej efektywny algorytmy niż rozwiązania natywne. W teście jakości interfejsu lepszą wydajność mają rozwiązania natywne pracujące zarówno na systemie Android jak i iOS. Przyczyną tego stanu rzeczy jest to, że aplikacja wieloplatformowa musi przekształcać cały interfejs, w taki sposób, aby był kompatybilny z natywnymi wzorami. Zrealizowane testy także nie wskazują jednoznacznie, czy lepiej pisać jedną aplikację za pomocą szkieletu wieloplatformowego, czy też tworzyć dwie oddzielne aplikacje przy użyciu natywnych technologii. Jednak na podstawie przeprowadzonych badań można zauważyć, że aplikacje stworzone za pomocą platformy Xamarin są tylko nieznacznie mniej wydajne niż te same aplikacje napisane w technologiach natywnych.

3. Metodyka badawcza

3.1. Opis stanowiska badawczego

W celu przetestowania aplikacji na urządzeniach mobilnych z systemem Android zostały wykorzystane dwa urządzenia: Emulator Pixel 5 oraz urządzenie fizyczne Samusng Galaxy M12. Specyfjkacja obu urządzeń została przedstawiona w Tabeli 1.

Opracowano dwie aplikacje testowe – jedną za pomocą technologii Xamarin, a drugą Flutter. Aplikacje te zawierały te same funkcjonalności, które realizowały operacje CRUD oraz sortowanie tablicy metodą przez scalanie. Obie współpracowały z bazą danych, która została wygenerowana w pamięci aplikacji (ang. InMemory).

Tabela 1: Konfiguracja emulatora i smartfona

Urządzenie	Pixel 5	Samsung Galaxy M12
System operacyjny	Android 11	Android 13
Procesor	Qualcomm Snapdragon 765G Taktowanie procesora: 2,4 GHz Liczba rdzeni: 8	Samsung Exynos 850 Taktowanie procesora: 2,0 GHz Liczba rdzeni: 8
Pamięć RAM	8 GB	4 GB

3.2. Scenariusze badawcze

W celu właściwego przeprowadzenia badania, w pierwszej kolejności zostały dobrane kryteria porównawcze. Następnie zostały przygotowane odpowiednie scenariusze badawcze, które umożliwiły przetestowanie wydajności szkieletów programistycznych. Wskaźnikiem, który został wybrany do porównywania był średni czas wykonania poszczególnych aplikacji. Badania zostały przeprowadzone wg następujących scenariuszy testowych:

- Sortowanie tablicy metodą przez scalanie (ang. Merge Sort)
- Zapis do bazy danych
- Odczyt z bazy danych
- Usuwanie z bazy danych
- Aktualizowanie bazy danych

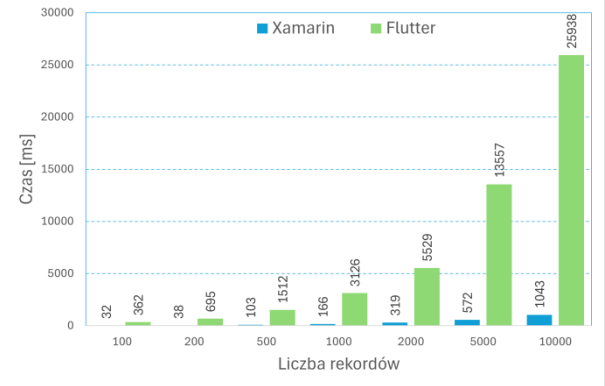
Dla każdej operacji związanej z komunikacją z bazą danych badano czas wykonania dla następującej liczby rekordów: 100, 200, 500, 1000, 2000, 5000 i 10000. Natomiast tablice, które były sortowane przez algorytm Merge Sort posiadały następującą liczbę elementów: 10000, 20000, 50000, 100000, 200000, 500000 oraz 1000000.

4. Wyniki badań

W trakcie prowadzenia badań każdy ze scenariuszy został powtórzony 10 razy, a uzyskane wyniki uśredniono. W dalszej części artykułu przedstawione zostaną wyniki dotyczące operacji wykonywanych na bazie danych (Rysunki 1–8), a następnie wyniki sortowania tablicy algorytmem Merge Sort (Rysunki 9 i 10).

4.1. Operacje CRUD – dodawanie danych do bazy (testy na emulatorze)

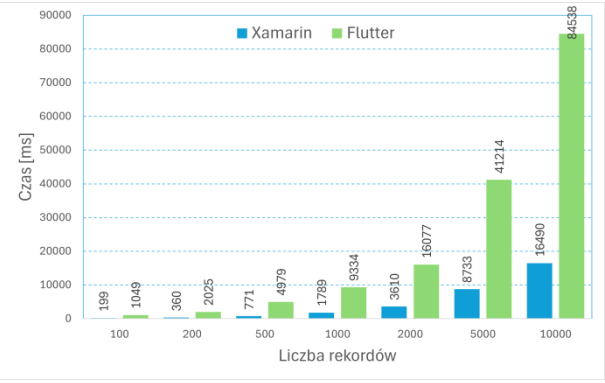
Pierwszy scenariusz badawczy to dodawanie danych do bazy. Na Rysunku 1 widać, że aplikacja zbudowana na bazie szkieletu Xamarin zdecydowanie lepiej radzi sobie w tej operacji niż aplikacja oparta na Flutterze. Przy jednoczesnym dodawaniu bardzo dużej liczby rekordów (10000) Xamarin jest o prawie 25 razy szybszy od Fluttera.



Rysunek 1: Średni czas dodawania danych do bazy danych.

4.2. Operacje CRUD – dodawanie danych do bazy (testy na urządzeniu fizycznym)

Analizując ten sam przypadek operacji CRUD, tym razem dla urządzenia fizycznego, można stwierdzić, że Xamarin i w tym przypadku jest wyraźnie szybszy (Rysunek 2). Jednak różnica między tymi szkieletami nie jest aż tak duża jak w badaniach zrealizowanych na emulatorze. Niezależnie od wielkości dodawanego pakietu Xamarin był wydajniejszy od 4 do 6 razy. W przypadku jednoczesnego dodawania 10000 rekordów szkielet Microsofu był szybszy 5 razy niż szkielet Flutter. Spowodowane to było dłuższym czasem nawiązywania połączenia z bazą danych aplikacji zbudowanej na bazie Fluttera. Należy w tym miejscu zaznaczyć, że każda operacja dodawania rekordu oznaczała osobne połączenie z bazą danych.

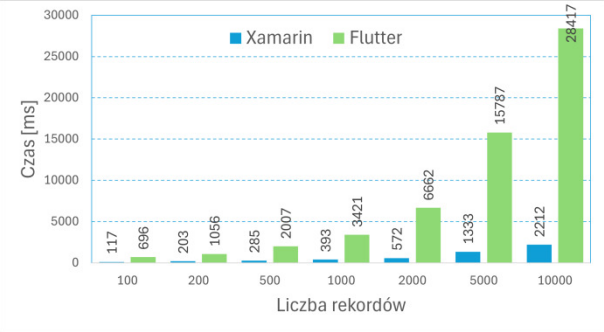


Rysunek 2: Średni czas dodawania danych do bazy danych.

4.3. Operacje CRUD – edytowanie danych w bazie (testy na emulatorze)

Wyniki badania operacji edytowania danych dla testów zrealizowanych na emulatorze zostały przedstawione na Rysunku 3.

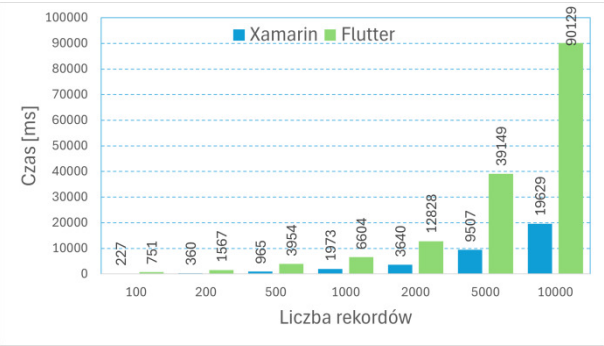
Wyniki pokazują, że również w tym przypadku aplikacja napisana za pomocą technologii Xamarin przewyższała wydajnością konkurencyjną platformę. Tym razem różnica między tymi szkieletami była nieco mniejsza i wynosiła podczas dodawania dużego pakietu danych (10000 rekordów) 13 razy.



Rysunek 3: Średni czas edycji danych w bazie danych.

4.4. Operacje CRUD – edytowanie danych w bazie (testy na urządzeniu fizycznym)

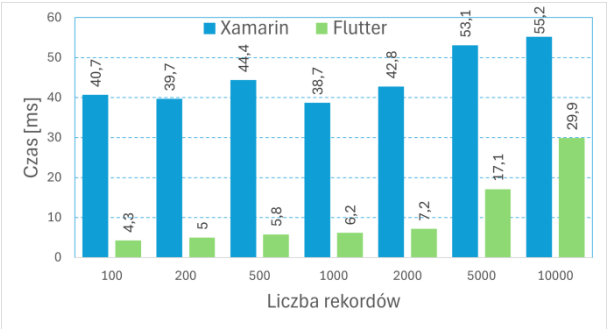
Rezultaty testów zrealizowanych na urządzeniu fizycznym edytowania danych, przedstawione na Rysunku 4 także uwidaczniają lepszą wydajność aplikacji utworzonej za pomocą szkieletu programistycznego Xamarin. Framework ten, niezależnie od przypadku był 3,3–4,6 razy szybszy. Znacznie gorszy wynik dla aplikacji zbudowanej na bazie Fluttera, podobnie jak w teście dodawania danych na urządzeniu fizycznym, był skutkiem tego, że tutaj również każda operacja to osobne połączenie z bazą danych.



Rysunek 4: Średni czas edytowania danych w bazie danych.

4.5. Operacje CRUD – odczyt danych z bazy (testy na emulatorze)

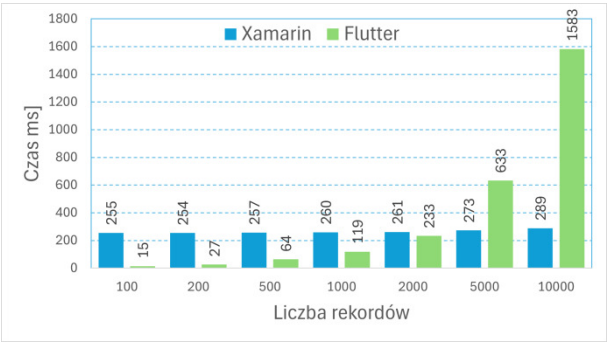
W teście odczytu danych z bazy przeprowadzonym na emulatorze lepszym okazał się szkielet Flutter (Rysunek 5). Przy odczycie małej liczby rekordów (100) był on szybszy o prawie 10 razy od Xamarina. Znacznie mniejsza różnica w wydajności wystąpiła przy odczycie największej liczby rekordów (10000) i wyniosła niecałe 2 razy.



Rysunek 5: Średni czas odczytu danych z bazy danych.

4.6. Operacje CRUD – odczyt danych z bazy (testy na urządzeniu fizycznym)

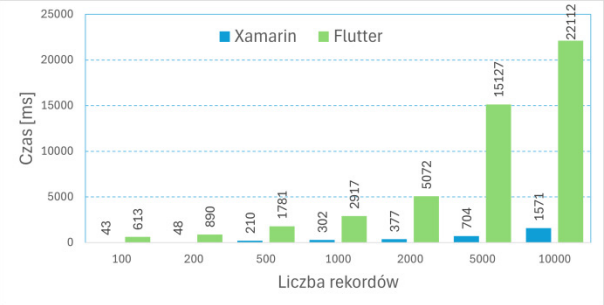
Na urządzeniu fizycznym podczas odczytu danych wyniki nie są już tak jednoznaczne. Odczyt małej liczby rekordów z bazy danych był szybciej realizowany na aplikacji opartej na Flutterze. Podczas pobierania dużych pakietów danych (powyżej 5000) szybszym szkieletem był Xamarin. Wyniki przedstawione na Rysunku 6 przy pobieraniu 10000 rekordów na raz były ponad 5 razy lepsze dla frameworka Microsoftu. Flutter podczas pobierania mniejszych zestawów danych (100–2000 rekordów) osiągnął lepsze wyniki, ponieważ pobranie danych było realizowane przy pomocy jednej operacji połączenia z bazą danych. Ograniczenie liczby operacji, skutkujące zmniejszeniem liczby osobnych połączeń z bazą danych, przyczyniłoby się do poprawy wydajności szkieletu Flutter.



Rysunek 6: Średni czas odczytu danych z bazy danych

4.7. Operacje CRUD – usuwanie danych z bazy (testy na emulatorze)

Testy usuwania różnej wielkości zestawów danych z bazy pokazały, że i w tym przypadku najlepszą wydajność miała aplikacja zaimplementowana za pomocą Xamarina, która dla pakietu składającego się z 10000 rekordów była 14 razy szybsza niż aplikacja Fluttera (Rysunek 7).

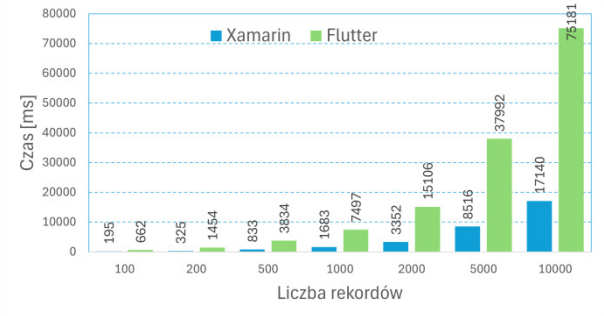


Rysunek 7: Średni czas usuwania danych z bazy danych.

4.8. Operacje CRUD – usuwanie danych z bazy (testy na urządzeniu fizycznym)

Wyniki tego samego testu wykonywanego na urządzeniu fizycznym pokazuje ten sam obraz – Xamarin był szkieletem wydajniejszym podczas usuwania różnej wielkości zbiorów danych (Rysunek 8).

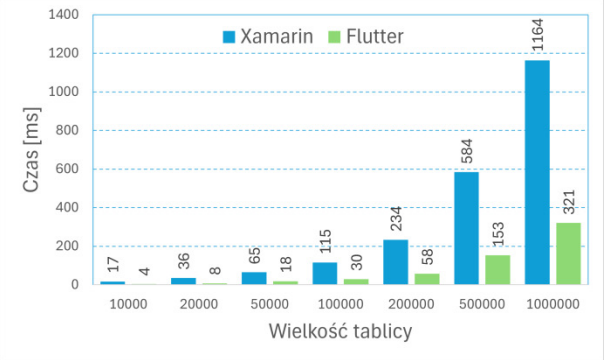
Różnica ta jest mniejsza, bo dla dużego zbioru danych (10000) framework ten był ponad 4 razy szybszy niż Flutter. Także w tym przypadku, ponownie Xamarin okazał się wydajniejszym szkieletem, ponieważ w tym zadaniu również wymagana była duża liczba połączeń do bazy danych.



Rysunek 8: Średni czas usuwania danych z bazy danych.

4.9. Sortowanie tablicy algorytmem Merge Sort (testy na emulatorze)

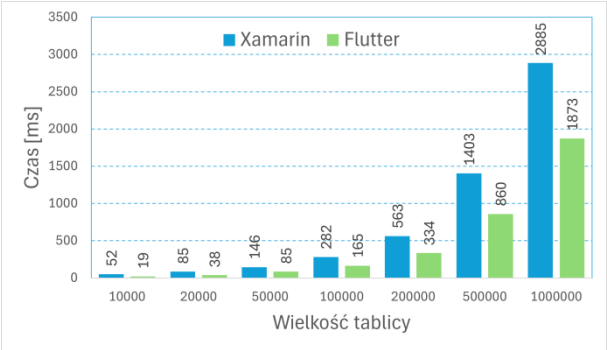
Ostatnie badanie dotyczyło sortowania tablicy liczb całkowitych z zakresu 1–1000000 przy pomocy algorytmu Merge Sort (Rysunek 9). Testy przeprowadzone na symulatorze dały wyniki, które jednoznacznie pokazują, że Flutter był zdecydowanie wydajniejszy od Xamarina. We wszystkich testach z różnymi zbiorami danych szkielet ten był od 3,4 do 4,5 razy szybszy.



Rysunek 9: Średni czas wykonania algorytmu sortowania.

4.10. Sortowanie tablicy algorytmem Merge Sort (testy na urządzeniu fizycznym)

Wyniki tego samego testu, ale przeprowadzonego na urządzeniu fizycznym przedstawia Rysunek 10.



Rysunek 10: Średni czas wykonania algorytmu sortowania.

Wykres pokazuje, że i tym razem lepsza okazała się aplikacja utworzona za pomocą szkieletu programistycznego Flutter. Jednak przy dużych rozmiarach tablic przewaga Fluttera była wyraźnie mniejsza niż w badaniu na emulatorze (1,5: 3,6). Na podstawie uzyskanych wyników można wysnuć wniosek, że Flutter podczas wykonywania złożonych operacji obliczeniowych okazał się wydajniejszym rozwiązaniem.

5. Wnioski

Porównanie wydajności technologii Flutter i Xamarin było głównym celem tych badań. Zostały utworzone dwie aplikacje testowe, które posiadały te same funkcjonalności i za pomocą których dokonano pomiarów czasu.

Po przeprowadzeniu analizy danych można wyciągnąć następujące wnioski:

- W trzech z czterech operacji na bazie danych: w dodawaniu, edycji i usuwaniu w testach przeprowadzonych zarówno na emulatorze jak i na urządzeniu fizycznym, zdecydowanie szybszym szkieletem programistycznym okazał się Xamarin.
- Jedynie w operacji pobierania/odczytu danych z bazy realizowanej na emulatorze wydajniejszym szkieletem okazał się Flutter.
- W operacji odczytu danych wykonywanej na urządzeniu fizycznym dla małych zbiorów danych szybszym frameworkiem okazał się Flutter. Natomiast dla dużych zestawów danych (od 5000 rekordów) wydajniejszym był Xamarin.
- W teście sortowania zrealizowanym zarówno na emulatorze jak i na urządzeniu fizycznym lepszą wydajność uzyskał Flutter.

Na podstawie uzyskanych wyników i przeprowadzonych analiz trudno jest jednoznacznie orzec, która technologia jest wydajniejsza. Warto jednak zauważyć, że jeśli głównym celem będzie wykonywanie działań na bazie danych to warto rozważyć użycie szkieletu programistycznego Xamarin. Jeśli operacje na bazie będą dotyczyć jedynie operacji odczytu danych, to w przypadku używania urządzenia fizycznego przy pracy na małych

zbiorach danych korzystniej będzie użyć Fluttera, a przy pracy z dużymi zbiorami Xamarina. Natomiast, gdy aplikacja będzie realizowała złożone operacje obliczeniowe lepszym wyborem będzie platforma Flutter.

Literatura

[1] Liczba dostępnych aplikacji w Google Play Store https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, [10.11.2023].

[2] Liczba aplikacji w Apple App Store https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, [20.11.2023].

[3] M. Uciński, M. Dzieńkowski, A comparative analysis of performance of Flutter and Xamarin development frameworks, Journal of Computer Sciences Institute 25 (2022) 366-370.

[4] G. Versluis, Xamarin Forms Essentials. First Steps Toward Cross-Platform Mobile Apps, Apress, Berkeley, CA, 2017.

[5] D. You, M. Hu, A comparative study of cross-platform mobile application development, Wellington, New Zealand, 2021.

[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 (3) (2017) 122-126.