JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 6 (2018) 68-72
Wysłane: 2017-11-08
Przyjęte: 2017-11-13

Analiza wydajności aplikacji mobilnych przy zastosowaniu różnych narzędzi programistycznych do ich budowy

Paweł Kotarski*, Kacper Śledź*, Jakub Smołka
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Rozwiązania pozwalające tworzyć aplikacje na różne systemy mobilne, w których istnieje możliwość współdzielenia całego kodu źródłowego bądź dużych jego części cieszą się coraz większą popularnością. W artykule przedstawiono porównanie wydajności aplikacji mobilnych przeznaczonych na system Android stworzonych przy pomocy domyślnych narzędzi tego środowiska oraz rozwiązań oferujących wsparcie dla wielu platform. Autorzy na wybranych przykładach badają wydajność w różnych aspektach działania aplikacji.

Słowa kluczowe: android; wydajność; narzędzia programistyczne

*Autor do korespondencji.

Adresy e-mail: pawel.kotarski1@pollub.edu.pl, kacper.sledz@pollub.edu.pl

Analysis of the impact of development tools used on the performance of the mobile application

Paweł Kotarski*, Kacper Śledź*, Jakub Smołka
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Solutions that allow developers to create application for different mobile systems in which they can share whole source code or large parts of it are becoming more popular. This article compares the performance of Android-based mobile application created with the default and multiplatform tools. Authors on selected examples examine performance in various aspects of the application.

Keywords: android; performance; development tools

*Corresponding author.

E-mail addresses: pawel.kotarski1@pollub.edu.pl, kacper.sledz@pollub.edu.pl

1. Wprowadzenie

Android jest obecnie najpopularniejszym systemem operacyjnym na urządzenia mobilne. Mimo swojej przeważającej pozycji na rynku wciąż istnieją inne, popularne systemy mobilne. W związku z tym niektóre firmy borykają się z problemem tworzenia kilku wersji tej samej aplikacji na różne systemy mobilne, aby dostarczyć swój produkt do jak największej ilości potencjalnych klientów. Wymaga to stworzenia i utrzymywania takich samych programów w różnych językach programowania i pracujących na dedykowanych środowiskach.

W ostatnim czasie powstało wiele bibliotek, które mają pomóc w rozwiązaniu tych problemów. Jednymi z bardziej popularnych są darmowy Apache Cordova oraz komercyjny Xamarin. Ich głównym założeniem jest możliwość wykorzystywania tego samego kodu przez programy tworzone na różne systemy operacyjne. Wiele firm decyduje się na skorzystanie z takich technologii ze względów ekonomicznych.

Niniejszy artykuł ma na celu porównanie wydajności aplikacji działających na systemie Android stworzonych w technologiach oferujących wsparcie dla wieloplatformowości z aplikacjami tworzonymi natywnie. To porównanie ma pomóc w wyborze odpowiedniego narzędzia do tworzenia aplikacji spełniającego wymogi danej organizacji. Dzięki niemu firmy będą mogły podjąć decyzję, czy ewentualne straty wydajnościowe są rekompensowane przez szybsze tempo wytwarzania oprogramowania oraz mniejsze nakłady finansowe.

2. Obiekt badań

Obiektem badań były narzędzia programistyczne służące do tworzenia aplikacji na systemie Android. Badane przez nas narzędzia to: Android SDK i język Java, Android NDK, Xamarin, Apache Cordova.

2.1. Android SDK i język Java

Najpopularniejszą metodą tworzenia aplikacji na system Android jest wykorzystanie języka Java. Jest on wykorzystywany w aplikacjach dostępnych w systemie bezpośrednio po jego zainstalowaniu. Ogromną popularność zyskał dzięki dobrymi proporcjami między łatwości pisania kodu, a wydajnością działania gotowego programu. Rozwiązanie to jest darmowe, ale można je zastosować tylko na telefonach z systemem Android. Widok aplikacji tworzony jest w języku XML natomiast bazowy blok tworzący elementy interfejsu użytkownika reprezentuje klasa View. Wszystkie funkcje systemu udostępniane aplikacji przez system operacyjny dostępne są w postaci klas języka Java[1].

2.2. Android NDK

Android NDK jest zbiorem narzędzi, który umożliwia tworzenie fragmentów aplikacji używając do tego języków natywnych takich jak C i C++. Dla pewnych typów aplikacji taki zabieg może pomóc w ponownym wykorzystywaniu kodu

bibliotek[2]. Pisanie fragmentów aplikacji wrażliwych wydajnościowo w kodzie natywnym może znacznie poprawić szybkość jej działania. Według oficjalnej dokumentacji systemu Android nie zaleca się pisania całej aplikacji przy pomocy Android NDK, ze względu na zwiększenie skomplikowania kodu. Ponadto w pewnych obszarach użycie natywnego kodu nie przynosi zauważalnych zysków wydajnościowych.

2.3. Xamarin

Platforma Xamarin służy do tworzenia aplikacji na różne mobilne systemy operacyjne. Są nimi Android, Windows Mobile oraz iOS. Jest to narzędzie komercyjne, powstałe przez ewolucję projektu Mono, który był alternatywną implementacją .Net Framework dla systemów z rodziny Unix. Językiem odpowiadającym za logikę aplikacji jest C#. Jest on popularnym wyborem dla tworzenia aplikacji desktopowych dla systemu Windows. Wygląd aplikacji można tworzyć w dwojaki sposób: wykorzystując Xamarin Forms lub natywne rozwiązania każdego systemu. Decydując się na Xamarin Forms powstaje jedna wersja widoku w języku XAML. W przypadku tworzenia natywnych widoków trzeba stworzyć osobną wersję widoku dla każdej platformy w domyślnym języku (dla systemu android jest to język XML).

2.4. Apache Cordova

Platforma Apache Cordova jest narzędziem open source, które umożliwia tworzenie aplikacji na wiele systemów mobilnych wykorzystując technologie webowe. Korzystając z tej platformy można pisać aplikację na systemy mobilne takie jak: Andoird, iOS, Windows Moblie, Blackberry. Stworzenie wyglądu aplikacji jest równoznaczne stworzeniu responsywnej strony www dostosowanej do urządzeń mobilnych. Natomiast logika napisana jest w języku JavaScript, w którym można odwoływać się do API systemowego w celu skorzystania z takich funkcjonalności jak GPS, powiadomienia, aparat, akcelerometr poprzez obiekty pośredniczące zapewnione przez platformę Apache Cordova.

3. Procedura i platforma testowa

W niniejszej publikacji przeprowadzone zostały trzy eksperymenty, a następnie została przeprowadzona analiza wyników. Pierwszym eksperymentem był pomiar czasu sortowania tablicy posiadającej 100 000 elementów. Następnym przetestowanym scenariuszem był pomiar czasu potrzebnego na zapis pliku o rozmiarze 10MB. Ostatnim eksperymentem opisanym w tym artykule był pomiar czasu potrzebnego na odczyt pliku o rozmiarze 10MB. Każdy z eksperymentów przeprowadzony został 20 razy aby uniknąć błędów związanych ze skokami obciążenia podzespołów przez system. Na potrzeby testów zostały napisane cztery aplikacje oparte o rozwiązania opisane w rozdziale „2. Obiekt badań”.

3.1. Sortowanie tablicy

Test każdej aplikacji polegał na posortowaniu dwudziestu różnych tablic o rozmiarze stu tysięcy elementów. Wszystkie aplikacje korzystały z tego samego zbioru tablic. Mierzony był czas jedynie sortowania tablicy z pominięciem czasu potrzebnego na załadowanie tablicy do pamięci.

Sortowanie tablic zostało zrealizowane poprzez implementację algorytmu quicksort na poszczególnych platformach. Jest on jednym z najpopularniejszych algorytmów stosowanych w dzisiejszej informatyce do sortowania dużych zbiorów danych. Wykorzystuje zasadę „dziel i zwyciężaj” sortując rekursywnie mniejsze fragmenty podzbioru.

3.2. Zapis dużego pliku

Celem testu było zmierzenie czasu potrzebnego do zapisania w pamięci urządzenia pojedynczego pliku o rozmiarze 10MB. Proces generowania pliku polegał na powieleniu losowego ciągu znaków do momentu uzyskania przez plik oczekiwanego rozmiaru. Aplikacja zapisywała plik sekwencyjnie.

3.3. Odczyt dużego pliku

Ostatni eksperyment polegał na zmierzeniu czasu potrzebnego do sekwencyjnego odczytu pliku o rozmiarze 10MB. Testowanym plikiem był plik tekstowy zawierający losowe znaki.

3.4. Platforma testowa

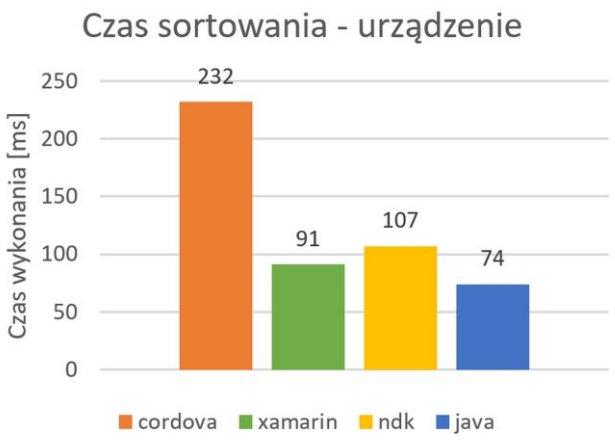
Wszystkie testy zostały przeprowadzone na urządzaniu fizycznym oraz emulatorze systemu Android działającym na komputerze klasy PC bazującym na specyfikacji telefonu LG Nexus 5X. Testowanym urządzeniem fizycznym był telefon Xiaomi Redmi 3 Pro, którego parametry sprzętowe zostały przedstawione w Tabeli 1.

Tabela 1. Parametry techniczne Xiaomi Redmi 3 Pro

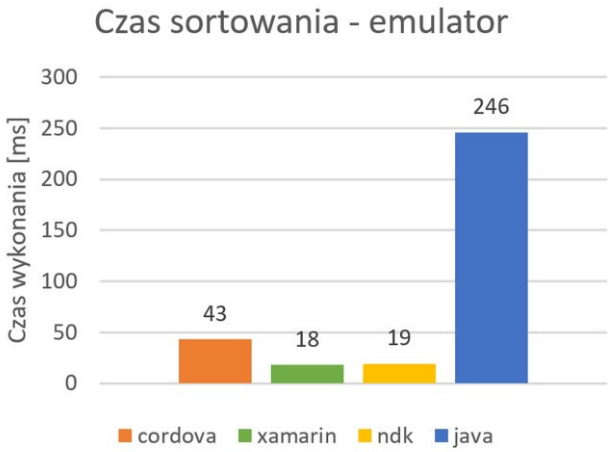
Procesor	Qualcomm Snapdragon 616
Taktowanie	1,50 GHz
Liczba rdzeni	8
Pamięć RAM	3 GB
Pamięć ROM	32 GB
System operacyjny	Android 7.1.2 Nougat

4. Wyniki badań

Poniżej znajdują się wyniki przeprowadzonych badań. Wszystkie czasy badań zostały przedstawione w milisekundach. Wykresy prezentują medianę dwudziestokrotnego wykonania eksperymentu. Wykresy przedstawiają kolejno wyniki badań sortowania tablicy posiadającej sto tysięcy elementów na urządzeniu oraz emulatorze, zapis pliku o rozmiarze 10MB na urządzeniu oraz emulatorze i odczyt pliku o rozmiarze 10MB na urządzeniu oraz emulatorze dla aplikacji stworzonej przy użyciu rozwiązania Apache Cordova, Xamarin, Android NDK i języka C++ oraz Android SDK i języka Java.

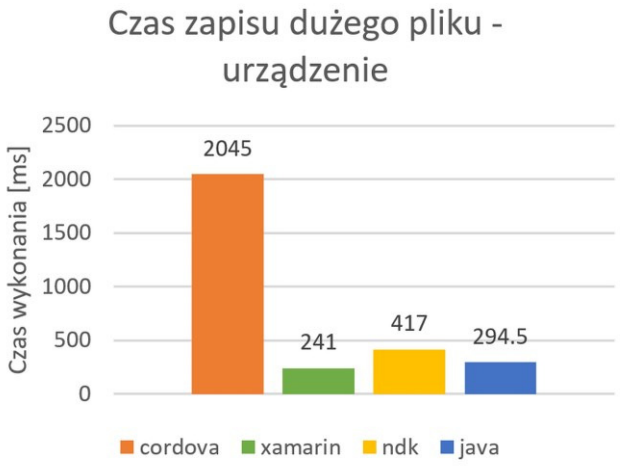


Rys. 1. Szybkość sortowania tabeli na urządzeniu

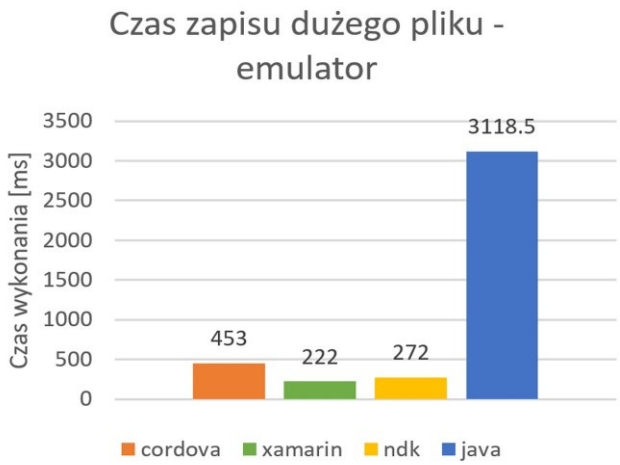


Rys. 2. Szybkość sortowania tabeli na emulatorze

Rysunek 1 oraz rysunek 2 przedstawiają wyniki badań sortowania tablic elementów. Na urządzeniu fizycznym z tym zadaniem najszybciej poradziła się aplikacja napisana w języku Java, natomiast najgorszy wynik osiągnęła aplikacja napisana na platformę Apache Cordova. Na symulatorze najlepiej poradziły sobie aplikacje napisane w technologiach Xamarin i Android NDK a najgorzej w języku Java.

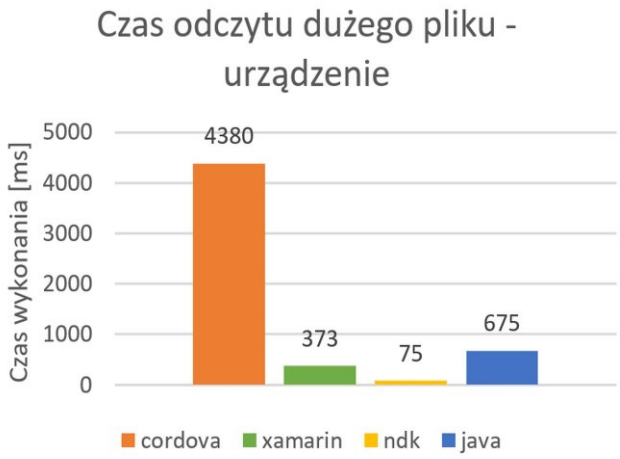


Rys. 3. Szybkość zapisu dużego pliku na urządzeniu

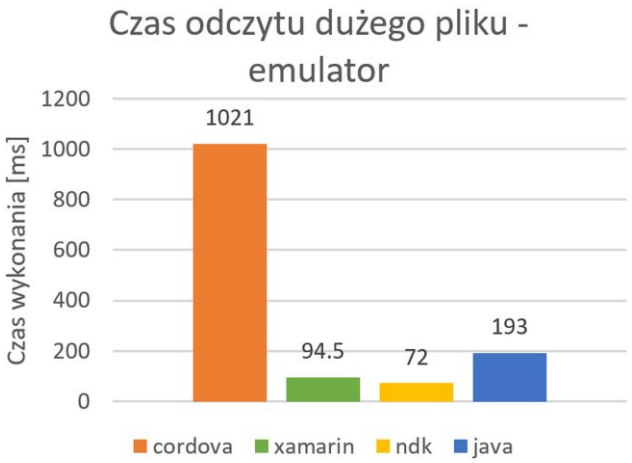


Rys. 4. Szybkość zapisu dużego pliku na symulatorze

Rysunek 3 oraz rysunek 4 przedstawiają wyniki badań zapisu dużego pliku. Na urządzeniu fizycznym najszybciej poradziła sobie aplikacja w technologii Xamarin, natomiast najgorszy wynik osiągnęła aplikacja napisana na platformę Apache Cordova. Na symulatorze najlepiej poradziła sobie aplikacja napisana w technologii Xamarin, a najgorzej w języku Java.



Rys. 5. Szybkość odczytu dużego pliku na urządzeniu



Rys. 6. Szybkość odczytu dużego pliku na symulatorze

Rysunek 5 oraz rysunek 6 przedstawiają wyniki badań odczuty dużego pliku. Na urządzeniu fizycznym z takim zadaniem najszybciej poradziła sobie aplikacja napisana z wykorzystaniem android NDK, natomiast najgorszy wynik osiągnęła aplikacja napisana na platformę Apache Cordova. Na symulatorze kolejność wyników jest identyczna.

5. Analiza wyników

W wynikach testów sortowania tablicy elementów oraz zapisu dużego pliku można zauważyć podobny stosunek wydajnościowy poszczególnych platform wobec siebie. Widoczna tam jest również znacząca różnica czasu wykonania testu przez aplikacje napisaną w oparciu o technologię Java pomiędzy urządzeniem fizycznym a emulatorem. Taka różnica nie jest zauważalna przy teście odczytu dużego pliku.

Podczas każdego eksperymentu przeprowadzanego na fizycznym urządzeniu najwolniejszą aplikacją okazywał się program stworzony przy pomocy platformy Apache Cordova. Różnica wydajności pomiędzy wspomnianą aplikacją a pozostałymi testowanymi aplikacjami była często mocno zauważalna oraz wynosiła wielokrotność najlepszego wyniku. Kolejność technologii w rankingu testu odczytu dużego pliku jest identyczna na urządzeniu fizycznym oraz na emulatorze.

Na podstawie przeprowadzonego testu sortowanie tablicy można zauważyć, że użytkownik korzystający z aplikacji stworzonych w oparciu o technologie: Xamarin, Android NDK oraz język Java nie zauważy znaczących różnic wydajnościowych podczas wykonywania podstawowych obliczeń przez aplikację. W wypadku odczytu dużego pliku widać znaczące różnice pomiędzy wynikami wszystkich platform. Lecz oprócz aplikacji napisanej w oparciu o rozwiązanie Apache Cordova te różnicę wydajnościowe nie powinny przeszkadzać w codziennym użytkowaniu aplikacji.

5.1. Sortowanie tablicy

W wynikach badania sortowania tablicy na urządzeniu można zauważyć znaczne różnice czasowe pomiędzy aplikacją napisaną w oparciu o platformę Apache Cordova, a aplikacjami napisanymi z wykorzystaniem innych technologii. Czasy sortowania tablic aplikacji napisanych w oparciu o Xamarin, Android NDK i język Java są porównywalne i te aplikacje działają około 250% szybciej niż aplikacja korzystająca z technologii webowych. Najszybsza na urządzeniu okazała się aplikacja napisana w języku Java.

Natomiast na emulatorze wyniki prezentują się zupełnie inaczej. Widać znaczny spadek wydajności aplikacji napisanej w języku Java. Jest ona prawie 13 razy mniej wydajna od aplikacji pisanych w technologiach Xamarin i Android NDK. Stosunek wydajności aplikacji napisanej w Apache Cordova do aplikacji napisanych w Android NDK i Xamarin jest prawie identyczny w wynikach testów przeprowadzonych na urządzeniu fizycznym i wynikach testów przeprowadzonych na symulatorze. Najszybszymi okazały się aplikacje napisane z wykorzystaniem technologii Xamarin i Android NDK otrzymując prawie identyczne czasy z różnicami rzędu pojedynczych milisekund.

5.2. Zapis dużego pliku

W teście zapisu dużego pliku najlepszy wynik na urządzeniu osiągnęła aplikacja napisana z wykorzystaniem narzędzi Xamarin. Podobne, ale nieco gorsze rezultaty uzyskały kolejno programy napisane z użyciem Android SDK w języku Java, oraz Android NDK i języka C++. Aplikacja stworzona z wykorzystaniem Apache Cordova zapisywała plik o wielkości 10 MB najdłużej, z czasem około 2 sekund. Dla porównania pozostałe aplikacje radziły sobie z tym zadaniem w czasie poniżej pół sekundy.

Na testach dokonanych na emulatorze najszybszą platformą był Xamarin. Pokonał on drugi w kolejności Android NDK o ok. 20%. W danym teście trzecie miejsce zajęła platforma Apache Cordova z wynikiem ok. dwa razy gorszym od pierwszego miejsca. Najdłużej test wykonywał się na aplikacji napisanej w języku Java. Czas wykonania był prawie siedmiokrotnie dłuższy od czasu wykonania podobnej aplikacji napisanej w oparciu o technologie webowe i wynosił ponad trzy sekundy.

5.3. Odczyt dużego pliku

Odczyt dużego pliku na urządzeniu trwa od 75 milisekund do 4.3 sekundy. Najlepsze osiągi ma aplikacja napisana z wykorzystaniem Android NDK, a najgorsze wyniki osiągnął program napisany w technologiach webowych. Aplikacje stworzone przy pomocy rozwiązania Xamarin oraz języka Java odczytują pojedynczy plik o rozmiarze 10 MB w ok. 0.5 sekundy. Jest to zauważalna różnica dla przeciętnego użytkownika systemów mobilnych.

Na wynikach testów na emulatorze można zauważyć mniejszą różnicę pomiędzy wynikami. Niechlubną ostatnią pozycję podczas tego testu osiągnęło rozwiązanie stworzone przy pomocy Apache Cordova. Jego wydajność była ponad 5 razy mniejsza od plasującej się na trzeciej pozycji aplikacji napisanej w języku Java. Różnica w wydajności pomiędzy rozwiązaniami stworzonymi w oparciu o platformę Xamarin i Android NDK nie jest zauważalna z punktu widzenia przeciętnego użytkownika.

6. Wnioski

Na podstawie przeprowadzonych na potrzeby artykułu testów można wysnuć następujące wnioski:

- Platforma Apache Cordova w dużym stopniu odstaje wydajnościowo od pozostałych opisanych technologii. Najprawdopodobniej wynika to ze sposobu działania technologii webowej. Logika aplikacji opiera się na języku JavaScript, który jest parsowany i kompilowany w momencie uruchomienia. Ponadto przy korzystaniu z zasobów zewnętrznych silnik platformy Apache Cordova musi przenosić dane pomiędzy warstwą natywną a warstwą webową.
- Podczas testów zaobserwowano znaczącą różnicę w czasie wykonania między aplikacją napisaną w języku Java uruchamianą na urządzeniu fizycznym a emulatorze. Często był to spadek w rankingu względem innych technologii. Może to wynikać z niższej wersji systemu Android na emulatorze niż testowanym urządzeniu oraz gorszej optymalizacji środowiska uruchomieniowego

Android (ART) języka Java na emulatorze niż na urządzeniu.

- Na podstawie przeprowadzonych testów nie można jednoznacznie stwierdzić które narzędzie pozwala tworzyć najwydajniejsze aplikacje. Najwydajniejszą technologią na urządzeniu dla kolejnych scenariuszy testowych: sortowania tablicy, odczytu dużego pliku, zapisu dużego pliku są kolejno: język Java, Android NDK oraz Xamarin.

Uzyskane wyniki pozwalają pokazać, że istnieją rozwiązania służące do tworzenia wieloplatformowych aplikacji mobilnych, które nie ustępują wydajnościowo rozwiązaniom natywnym. Dzięki temu można szybciej tworzyć aplikacje mobilne nie tracą na wydajności ich działania. Ponadto dzięki zastosowaniu takich rozwiązań zmniejsza się czas i koszt utrzymywania aplikacji dzięki współdzieleniu dużych części kodu pomiędzy platformami.

Literatura

[1] View | Android Developers, https://developer.android.com/reference/android/view/View.html [30.06.2017]

[2] Android NDK | Android Developers, https://developer.android.com/ndk/index.html [30.06.2017]

[3] Batyuk L., Schmidt A. D., Schmidt H. G., Camtepe A., Albayrak S. , Mobile Wireless Middleware, Operating Systems and Applications – Workshops, Springer, 2009, 381-392

[4] Yadav R. K., Bhadoria R. S., Performance analysis for Android runtimes environment, IEEE, 2015

[5] Gerasimov V. V., Bilovol S. S., Ivanova K. V., Comparative analysis between Xamarin and PhoneGap for .Net, 2015

[6] Ptitsyn P. S., Radko D. V., ARPN Journal of Engineering and Applied Sciences, vol. 11, no. 19, 2016, 11300-11307

[7] Corral L., Sillitti A., Succi G., Mobile multiplatform development: An experiment for performance analysis, Procedia Computer Science, 2012

[8] Delia L., Galdamez N., Thomas P., Corbalan L., Pesado P., Multi-Platform Mobile Application Development Analysis, Research Challenges in Information Science (RCIS), 2015

[9] Lin C.M., Lin J.H., Dow C.R., Wen C.M., Benchmark Dalvik and Native Code for Android System, IEEE, 2011

[10] Peppers J., Xamarin Crossplatform Application Development, Packt Publishing, 2014

[11] Cadmen R.K., Apache Cordova in Action, Manning Publications, 2015