JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 6 (2018) 28-33
Wysłane: 2017-09-13
Przyjęte: 2017-09-18

Porównanie wydajności szkieletów aplikacji mobilnych umożliwiających programowanie z wykorzystaniem technologii internetowych

Marcin Martyna*, Jakub Smołka

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Na przestrzeni ostatnich lat powstało wiele szkieletów aplikacji pozwalających na programowanie aplikacji mobilnych. Część z nich bazuje na językach programowania typowych dla tworzenia aplikacji internetowych, takich jak HTML czy JavaScript. W artykule zawarte jest porównanie trzech takich wieloplatformowych środowisk programistycznych jakimi są PhoneGap, NativeScript oraz Appcelerator. W każdym z tych środowisk powstała aplikacja o identycznych funkcjonalnościach przeznaczona na system Android. Zaimplementowane testy pozwoliły sprawdzić która z aplikacji jest najwydajniejsza pod względem czasu jakiego potrzebuje na wykonanie poszczególnych funkcji. Przedstawiono zestawienie i analizę otrzymanych wyników.

Słowa kluczowe: porównanie szkieletów aplikacji; Android; mobilna aplikacja; testowanie aplikacji

*Autor do korespondencji.

Adres e-mail: marcin.martyna850@gmail.com

Efficiency comparison of mobile application frameworks for programming using internet technologies

Marcin Martyna*, Jakub Smołka

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

Abstract. Over the last few years many frameworks allowing programming mobile applications have been created. Some of them are based on programming languages typical for internet application programming - HTML or JavaScript for example. This paper presents a comparison of three cross-platform environments: PhoneGap, NativeScript and Appcelerator. Using each of these environments an application with identical functionalities was created. The application was designed for Android operating system. Implemented tests allowed for determining which one of the application framework is the most efficient with respect to the time needed for executing respective functions. Obtained results are shown and analyzed.

Keywords: framework comparison; Android; mobile application; application testing

*Corresponding author.

E-mail address: marcin.martyna850@gmail.com

1. Wstęp

W świecie gdzie prawie każdy jest użytkownikiem smartfona firmy je produkujące stają przed nie lada wyzwaniem, wyzwaniem o to by to właśnie produkt danej firmy znalazł się na sklepowej półce a następnie trafił do rąk nowego właściciela. W ten sposób firmy konkurują ze sobą tworząc coraz to nowsze, lepsze urządzenia. Smartfony na rynku pojawiły się około dziesięć lat temu i od tamtego czasu systematycznie podbijają serca coraz to nowych użytkowników [1]. Z danych na rok 2016 wynika że na świecie jest 2.1 miliarda tych urządzeń prognoza na obecny rok mówi że będzie ich 2.32 miliarda [2]. Kolejne prognozy wskazują ciągły wzrost tej tendencji [2].

Wraz ze wzrostem mocy obliczeniowych tych urządzeń wzrosły oczekiwania konsumentów odnośnie aplikacji tworzonych na nie. To właśnie dzięki temu postępowi programiści piszący aplikacje na te urządzenia mają coraz to większe możliwości. Artykuł poświęcony jest możliwościom jakie stoją przed programistami, którzy zdecydują się na pisanie aplikacji mobilnych przy pomocy JavaScript czy HTML w szkieletach aplikacji jakimi są PhoneGap, NativeScript czy Appcelerator.

2. Przegląd literatury

Przeglądu literatury dokonano na podstawie dostępnych pozycjach w bazach Scopus, Springer, ScienceDirect oraz BazTech. Pozycje których tematyka stanowiła wsparcie tematu pracy zostały opisane poniżej.

Artykuły [3], [4], [5] stanowiły wprowadzenie do badanych frameworków. Znajdowały się w nich praktyczne przykłady użycia fragmentów kodu źródłowego oraz instrukcje odnośnie interfejsów graficznych poszczególnych frameworków. W artykułach [6], [7], [8] zawarte zostały porównania dotyczące aplikacji tworzonych natywnie oraz tych tworzonych w frameworkach wieloplatformowych. Natomiast artykuły [9], [10], [11] opisują sposoby testowania aplikacji na konkretnych przykładach.

3. Cel badań

Celem badań było porównanie wydajności frameworków pozwalających na tworzenie mobilnych aplikacji na system Android przy pomocy takich języków programowania jak HTML i JavaScript. Badania mają za zadnie sprawdzić, który ze szkieletów aplikacji radzi sobie najlepiej z poszczególnymi aspektami, tak by osoby poszukujące wieloplatformowego

środowiska programistyczne miały zarys możliwości oraz wydajności porównywanych framework'ów.

4. Szkielety aplikacji

Szkielet aplikacji lub platforma programistyczna (ang. framework) narzędzie, które dostarcza programiście gotowych bibliotek czy komponentów. Znacznie ułatwia pracę i skraca czas jaki programista musi poświęcić nad aplikacją by ją ukończyć. Ponadto istnieją także wieloplatformowe szkielety aplikacji tzw. cross-platform frameworks, które pozwalają na pisanie aplikacji na różne platformy jak np. Android i iOS jednocześnie. Obecnie na rynku istnieje szkielet aplikacji dla prawie każdego popularnego języka programowania. Chcąc napisać aplikację z natywnymi funkcjami na system Android czy iOS programista musi znać dwa różne języki programowania oraz napisać oddzielny kod dla każdej z aplikacji, natomiast w przypadku szkieletów aplikacji wieloplatformowych wystarczy jeden język, w którym napisany kod jest odpowiednio przystosowywany do danej platformy. Najczęściej dzieje się to w taki sposób, że do kodu źródłowego dodawane są odpowiednie wtyczki zawarte w narzędziu przystosowujące aplikację do danego systemu mobilnego. Dla Javy najpopularniejszymi szkieletami aplikacji są Spring MVC, Struts 2 czy Hibernate, a dla JavaScript to Angular, ReactJS czy Vue.js, ale to tylko wierzchołek góry lodowej [12,13]. Narzędzia te z uwagi na coraz to większe potrzeby programistów czy silną konkurencję na rynku są nieustannie ulepszane.

5. Użyte technolgie

W celu przeprowadzenia odpowiednich testów a następnie porównania poszczególnych szkieletów aplikacji niezbędne było wykorzystanie technologii opisanych w kolejnych podpunktach.

5.1. System Android

Jako że docelową platformą aplikacji tworzonych w celach porównawczych jest Android warto wspomnieć o nim kilka słów. W 2005 roku mała firma o nazwie „Android Inc" została kupiona przez znanego na całym świecie giganta branży internetowej „Google", był to początek drogi do stworzenia systemu Android oraz do wejścia firmy Google na rynek urządzeń mobilnych. Kolejnym przełomowym wydarzeniem było wydanie pierwszej wersji tego systemu 21 października 2008 roku, od tamtego dnia Android systematycznie znajdował coraz to nowych konsumentów powoli zdobywając rynek systemów obsługujących urządzenia mobilne [14]. Obecnie open sourceowy Android jest najpopularniejszym systemem mobilnym, prawie 72% na rynku systemów urządzeń mobilnych z pewnością można nazwać miażdżącą przewagą nad konkurencją, drugi na rynku iOS posiada jedynie 18,89% rynku [15]. Należy pamiętać że na przestrzeni lat system ten był nieustannie ulepszany do obecnie znajdującej się na rynku wersji 8.0 zwanej także Oreo.

5.2. NativeScript

Open sourceowy NativeScript jest wieloplatformowym frameworkiem pozwalającym na tworzenie hybrydowych [16] aplikacji mobilnych na urządzenia bazujące na systemach Android i iOS. Należący do bułgarskiej firmy „Telerik" udostępniony został w marcu 2015 roku i nieustanie rozwijany od tamtego czasu. Obecnie dostępny jest w stabilnej wersji 3.0 wydanej w maju 2017 roku. Oparty na licencji Apache 2.0, NativeScript pozwala na tworzenie aplikacji używając zarówno JavaScript jak i TypeScript [17]. Framework ten bezpośrednio wspiera AngularJS aczkolwiek korzystanie z niego nie jest obowiązkowe, użytkownik ma możliwość swobodnego wyboru środowiska programistycznego.

5.3. Appcelerator

Kolejny framework, Titanium Appcelerator to następne wieloplatformowe środowisko programistyczne dostępne dla każdego użytkownika. Choć pojawił się już w grudniu 2008 roku to dopiero w czerwcu 2009 roku i w 2012 roku wyszło wsparcie do takich platform jak Android czy iOS. Appcelerator pozwala nam na tworzenie aplikacji na wcześniej wymienione platformy używając jednego kodu bazowego. W przypadku Appceleratora jedynym dostępnym językiem programowania jest JavaScript. Kolejną wspólną cechą tych środowisk jest licencja (Apache 2.0). W przypadku Appceleratora użytkownik do dyspozycji ma oddzielne środowisko, w którym ma możliwość edytowania kodu źródłowego. W Appceleratorze istnieje także możliwość zakupienia wersji rozszerzonej dającej dodatkowe możliwości np. czat i wsparcie email, współpraca z innymi twórcami czy dostęp do usługi RMAD (ang. Rapid Moblie App Development) [18], to tylko część możliwości jakie twórcy przygotowali swoim klientom więcej informacji można znaleźć na stronie producenta [19].

5.4. PhoneGap

Należący do fundacji Apache (pełna nazwa Apache Software Fundation) PhoneGap jest wieloplatformowym szkieletem aplikacji bazującym na licencji Apache 2.0. PhoneGap nie różni się od wcześniej wymienionych programów pod względem języków jakie są używane do tworzenia aplikacji, tak jak poprzednicy bazując na jednym kodzie można stworzyć aplikacje na Androida i iOSa jednocześnie. Szukając informacji na temat tego środowiska można natknąć się na pewne nieścisłości, otóż PhoneGap stworzony został przez firmę Nitobi, która w 2011 roku została kupiona przez Adobe Systems. Następnie kod źródłowy PhoneGap został przekazany obecnemu właścicielowi, aczkolwiek by mogło to się odbyć firma Adobe musiała zmienić nazwę produktu z uwagi na ochronę własności intelektualnej [20]. Apache Callback taka była pierwsza propozycja nowej nazwy, niestety nie przyjęła ona się zbyt dobrze i społeczność szybko ją zmieniła. Obecnie formalna nazwa tego frameworka to Apache Cordova - Cordova od nazwy ulicy na jakiej znajdowała się siedziba firmy Nitobi w Vancouver [20]. Według autora [20] pomimo

różnych nazw, które są jedyną różnicą jest to wciąż ten sam framework i nie prędko to się zmieni.

5.5. Urządzenie testowe

Testy przeprowadzone zostały na urządzeniu Asus ZenFone 2 ZE551ML o następujących parametrach:

Model	Asus ZenFone 2 (ZE551ML)
Wyświetlacz	5.5 cala, IPS LCD Pojemnościowy, 16 mln kolorów 1920x1080 pikseli
Procesor	Intel Atom Z3580 (4x 2.3GHz)
Pamięć	4 GB pamięci RAM 32 GB na dane, slot microSD
System	Android 5.0 KitKat
Komunikacja	Wi-Fi 802.11 ac Bluetooth, NFC 3G, 4G, dual-SIM
Aparat	13 MP z tyłu i 5 MP z przodu
Bateria	3000 mAh
Wymiary i waga	152,5 x 77,2 x 10,9 mm 170 g

Rys 1. Parametry urządzenia na którym odbyły się testy [21]

Więcej informacji na temat danego urządzenia można znaleźć na stronie producenta [21].

6. Metodyka badawcza

W każdym ze środowisk wykonano aplikację implementującą ten sam zestaw testów. W każdym z testów umieszczono funkcje mierzące czas wykonania się danego testu. Funkcja swreset() odpowiada za reset wartości pomiaru, reset ten następuje za każdym razem gdy dany test jest uruchamiany. Funkcja startstop() rozpoczyna pomiar czasu a display() podaje wartość w jakim test się wykonał.

Przykład 1. Ukazanie metod odpowiedzialnych za pomiar czasu na przykładzie pobierania lokalizacji GPS w framework'u PhoneGap

```
var Latitude = undefined;
var Longitude = undefined;
function getMapLocation() {
swreset();
startstop();
navigator.geolocation.getCurrentPosition(onMapSuccess,
onMapError, {enableHighAccuracy: true});
}
function onMapSuccess(position) {
Latitude = position.coords.latitude;
Longitude = position.coords.longitude;
document.stpw.output.value = "Lokalizacja " + Latitude + " "
+ Longitude;
display();
}
function onMapError(error) {
document.stpw.output.value = "Odczytanie lokalizacji nie
powiodło się";
}
```

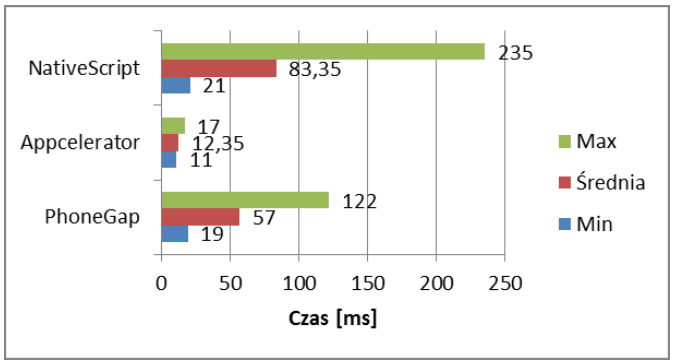
Podczas opracowywania testów wzorowano się na pracy [22]. Oto spis wszystkich testów:

1) Generowanie 100 000 liczb z przedziału [0-9]
2) Pobieranie listy kontaktów
3) Sortowanie listy 10 000 elementów
4) Zapis dużego pliku
5) Wielokrotny zapis małego pliku
6) Wielokrotny odczyt małego pliku
7) Odczyt dużego pliku
8) Odczyt bieżącej pozycji urządzenia poprzez GPS
9) Konwersja obrazu do odcieni szarości
10) Odtworzenie pliku audio o długości 1s

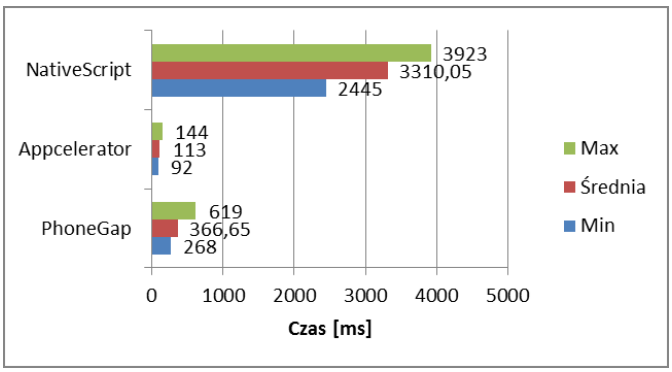
Każda z powyższych funkcji została przetestowana trzykrotnie, raz w każdej z aplikacji wykonanych w różnych frameworkach. Wszystkie testy odbyły się na jednym urządzeniu a w tle nie było uruchomionych żadnych innych aplikacji. Miało to na celu nie dopuszczenie do sytuacji gdzie dane funkcje miały by lepszy czas jedynie ze względu na to, że w tle działały inne aplikacje spowalniające czas wykonania poszczególnych testów. Funkcje badano pod względem czasu jakiego potrzebują na wykonanie, a następnie wyniki zestawiono w tabeli. Rezultaty z tabel posłużyły do stworzenia wykresów przedstawiających minimalne, średnie oraz maksymalne czasy pomiarów poszczególnych frameworków. Należy zaznaczyć, że w przypadku Appceleratora nie udało się zaimplementować testu konwersji obrazu do odcieni szarości.

7. Wyniki badań

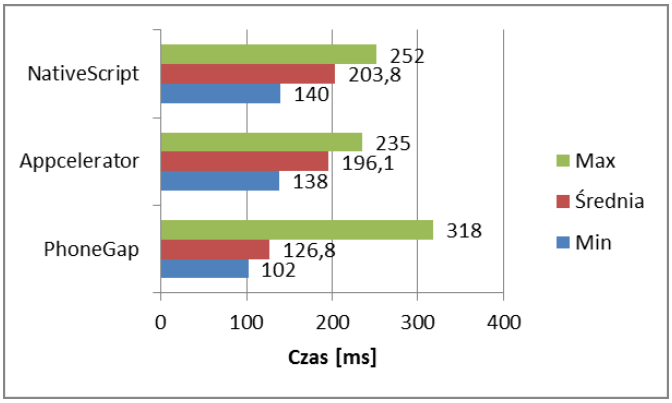
Do każdego z testów wykonano dwadzieścia pomiarów. Wyniki umieszczono w tabeli, a następnie na ich podstawie stworzone zostały wykresy znajdujące się na rysunkach 2-11. Min oznacza minimalny czas pośród wszystkich wyników w jakim test został wykonany. Średnia stanowi średnią obliczoną ze wszystkich pomiarów danego narzędzia. Max jest czasem najdłuższym w jakim zadanie zostało wykonane. Wartości liczbowe na osi poziomej każdego z wykresów oznaczają czas w milisekundach.



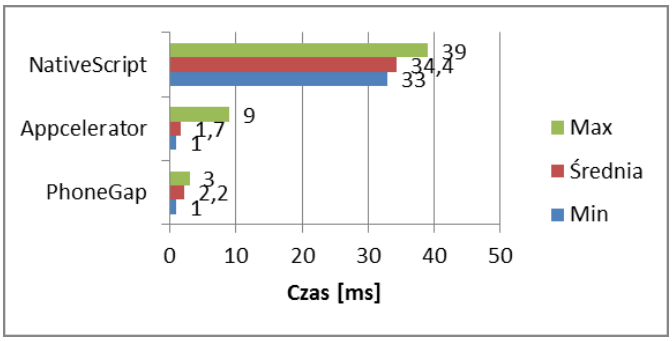
Rys 2. Wyniki testu generowania 100 000 liczb z przedziału (0;9)



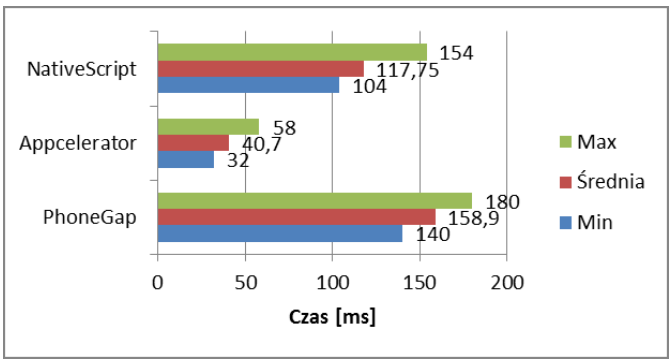
Rys 3. Wyniki testu pobierania kontaktów z urządzenia



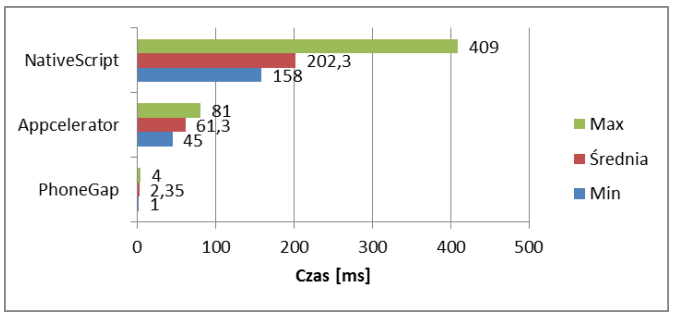
Rys 4. Wyniki testu sortowania tablicy 10 000 elementów



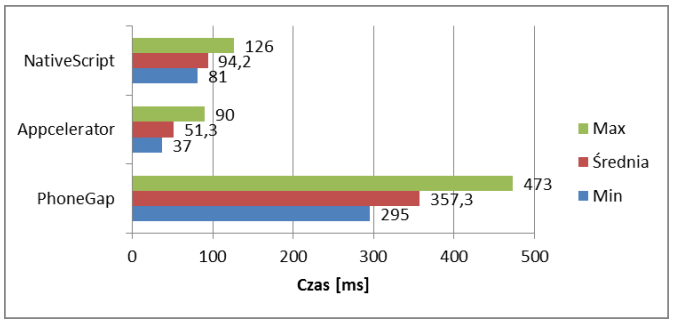
Rys 5. Wyniki testu odczytu z dużego pliku



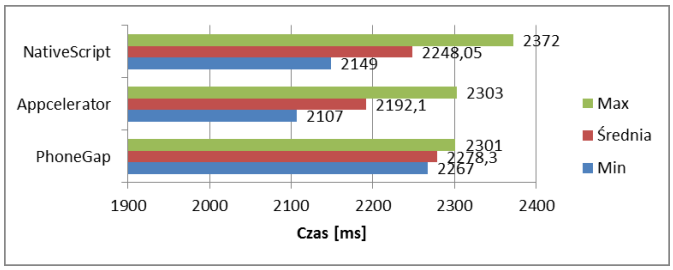
Rys 6. Wyniki testu wielokrotnego odczytu małego pliku



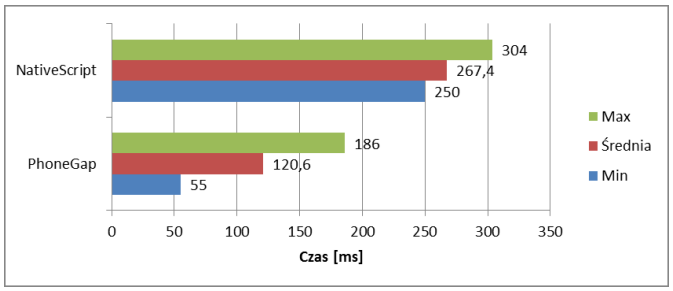
Rys 7. Wyniki testu zapisu dużego pliku



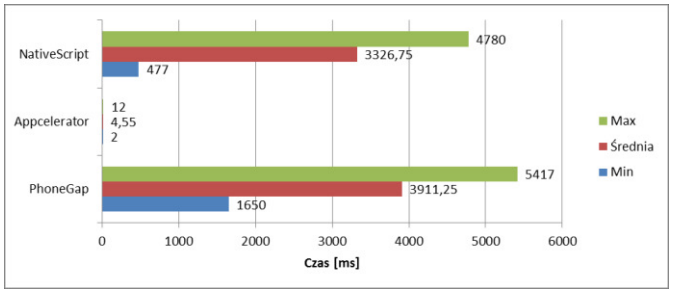
Rys 8. Wyniki testu wielokrotnego zapisu małego pliku



Rys 9. Wyniki testu odtwarzania pliku audio



Rys 10. Wyniki testu konwersji obrazu do odcieni szarości



Rys 11. Wyniki testu pobierania lokalizacji z GPS

8. Wnioski

Na podstawie danych zebranych z pomiarów i zestawionych na rysunkach 2-11 oraz w tabeli 1, można zauważyć znaczną przewagę Appceleratora nad pozostałymi frameworkami.

Tabela 1. Porównanie badanych frameworków

Kryterium porównawcze	Appcelerator	PhoneGap	NativeScript
Czas budowy aplikacji	1min 16s	3min 28s 7s przy uruchamianiu w przeglądarce	1min 35s przy pierwszym uruchomieniu oraz po dodaniu wtyczki. 25s w następnych uruchomieniach
Sposób budowy aplikacji	W środowisku, aplikacja jest tworzona bezpośrednio na urządzenie lub emulator	Na stronie producenta, aplikacja jest możliwa do pobrania poprzez zeskanowanie kodu QR po zbudowaniu projektu. Możliwość uruchomienia aplikacji w przeglądarce jednak nie wspiera to natywnych funkcji	W wierszu poleceń poprzez wykonanie polecenia „tns run android” wcześniej przechodząc do lokalizacji projektu za pomocą polecenia „cd”
Zintegrowane środowisko graficzne	Tak	Tak	Nie
Licencja	Apache 2.0	Apache 2.0	Apache 2.0
Obsługiwane języki programowania	JavaScript	JavaScript oraz HTML	JavaScript oraz TypeScript
Plik na bazie którego budowany jest interfejs graficzny	XML	HTML	XML
Wtyczki	Do pobrania na stronie producenta	Do pobrania na stronie producenta	Do pobrania na stronie producenta
Sposób dodania wtyczki w projekcie	Skopiowanie pliku zip z wtyczką do projektu a następnie dodanie modułu w pliku tiapp.xml	Odniesienie się poprzez „<script type="text/javascript" src=lokalizacja wtyczki"></script>” w pliku html	Dodanie w wierszu poleceń za pomocą polecenia „tns plugin add nazwa wtyczki”

W 7 na 10 testów narzędzie to okazało się najwydajniejsze osiągając wyniki nawet kilku czy kilkunastokrotnie lepsze od konkurentów. Na podstawnie dokonanych pomiarów oraz danych z rysunków 2-11 wynika że Appcelerator jest szybszy od drugiego pod względem wydajności PhoneGap o 267%. Wynik uzyskano poprzez porównanie zsumowanych ze sobą średnich wartości każdego środowiska ze wszystkich testów z wyjątkiem konwersji obrazu do odcieni szarości. W porównaniu z NativeScript, Appcelerator jest o 359% szybszy. Tak znaczne różnice wynikają głównie ze znacznych różnic w czasach pomiarów w testach, takich jak odczyt lokalizacji z GPS czy pobranie kontaktów. Drugim pod względem wydajności jest framework PhoneGap który od najgorszego NativeScript jest o 36% bardziej wydajny. Przy obliczaniu tego wyniku uwzględnione zostały średnie czasy w teście konwersji obrazu do odcieni szarości.

Literatura

[1] Podróż w czasie: Era G1, czyli pierwszy smartfon z Androidem o wielu imionach, http://komorkomania.pl/2465,podroz-w-czasie-htc-dream-era-g1 [01.06.2017]

[2] Number of smartphone users worldwide from 2014 to 2020 (in billions, https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ [26.05.2017]

[3] Sarah Allen, Vidal Graupera, Lee Lundrigan: PhoneGap

[4] S. Hanschke, H. Heitkötter, T. A. Majchrzak: Evaluating Cross-Platform Development Approaches for Mobile Applications.

[5] S. Allen, V. Graupera, L. Lundrigan: Titanium Mobile.

[6] Wenliang Du, Xing Jin, Tongbo Luo, Lusha Wang: Fine-Grained Access Control for HTML5-Based Mobile Applications in Android.

[7] Euler Horta Marinho, Rodolfo Ferreira Resende: Native and Multiple Targeted Mobile Applications.

[8] Antonio Cicchetti, Manuel Palmieri, Inderjeet Singh: Comparison of cross-platform mobile development tools.

[9] B. Dookheea, V. Hurbungs, Y. K. Suttroogun: A Framework to Reduce the Testing Time of a Mobile Application Using an Automation Tool.

[10] Marcelo Medeiros Eler, Andre Takeshi Endo, Davi Bernardo Silva,: An analysis of automated tests for mobile Android applications.

[11] Henning Heitkötter ,Tim A. Majchrzak, Benjamin Ruland, Till Weber: Comparison of Mobile Web Frameworks.

[12] 7 Best Java Frameworks for 2016 https://www.romexsoft.com/blog/7-best-java-frameworks-for-2016/ [01.06.2017]

[13] 5 Best JavaScript Frameworks in 2017 https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282.

[14] A complete history of Android http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327/3 [01.06.2017]

[15] Mobile operating systems' market share worldwide from January 2012 to December 2016 https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ [01.06.2017]

[16] Aplikacja mobilna – Hybryda czy Natywna? Różnice https://www.gmi.pl/blog/aplikacja-mobilna-hybryda-natywna/

[17] Strona producenta TypeScript https://www.typescriptlang.org/ [14.08.2017]

[18] Rapid mobile app development (RMAD) http://whatis.techtarget.com/definition/rapid-mobile-app-development-RMADrapid [02.06.2017]

[19] PhoneGap, Cordova, and what’s in a name? http://phonegap.com/blog/2012/03/19/phonegap-cordova-and-whate28099s-in-a-name/ [02.06.2017]

[20] Asus ZenFone ZE551ML https://www.asus.com/pl/Phone/ZenFone_2_ZE551ML/ [03.06.2017]

[21] Strona producenta Appcelerator zakładka cnnik https://www.appcelerator.com/pricing/ [6.09.2017]

[22] Bartłomiej Matacz: Efficiency Analysis of Mobile Application Frameworks .