JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 5 (2017) 208-212
Wysłane: 2017-06-21
Przyjęte: 2017-06-23

Porównanie wydajności platformy Xamarin oraz aplikacji natywnych na przykładzie systemu operacyjnego Android

Ihor Bodia*, Małgorzata Plechawska-Wójcik

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

Streszczenie. W artykule przedstawiono rezultaty analizy porównawczej poszczególnych narzędzi do wytwarzania aplikacji dla systemu operacyjnego Android. Analizę wydajnościową przeprowadzono w oparciu o aplikacje wytworzone przy pomocy poszczególnych frameworków, w szczególności Xamarina oraz natywnych narzędzi wytwarzania oprogramowania. Zdefiniowano kilka kryteriów badawczych, w oparciu o które otrzymano wyniki porównania wydajności i sformułowano wnioski.

Słowa kluczowe: Android; wydajność; porównanie; Xamarin; Java

*Autor do korespondencji.

Adres e-mail: ihor.bodia@pollub.edu.pl

Performance comparison of the Xamarin platform and native applications for Android operating system

Ihor Bodia*, Małgorzata Plechawska-Wójcik

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

Abstract. The article presents the results of comparative analysis of various tools for creating applications for the Android operating system. Performance analysis was based on applications created using individual frameworks, in particular Xamarin and native software development tools. Several research criteria have been defined, based on which results of performance comparison have been derived and formulated conclusions.

Keywords: Android; performance; comparison; Xamarin; Java

*Corresponding author.

E-mail address: ihor.bodia@pollub.edu.pl

1. Wstęp

Praca ta zawiera teoretyczne oraz praktyczne informacje o dość nowym frameworku rozwoju oprogramowania jakim jest Xamarin, którego celem jest ujednolicenie rozwoju oprogramowania. W dzisiejszych czasach technologie informatyczne rozwijają się w szybkim tempie i wciąż pojawiają się nowe rozwiązania, począwszy od nowego języka programowania, kończąc nowymi rozwiązaniami sprzętowymi. Aby deweloperzy korzystający z platformy .NET mogli maksymalnie wykorzystać swoje możliwości w zakresie tworzenia oprogramowania na rynku wieloplatformowych rozwiązań został przedstawiony framework Xamarin. Ten framework jest doskonałym przykładem na to, jak firmy w branży rozwoju oprogramowania mobilnego konkurują między sobą, poprzez zapewnienie twórcom oprogramowania znanych interfejsów dla nieznanych urządzeń fizycznych lub platform mobilnych.

Celem pracy jest realizacja badania polegającego na porównywaniu wydajności działania platformy Xamarin i natywnych technologii budowy aplikacji dla systemu Android. Porównanie zrealizowane będzie w oparciu o wybrane kryteria analizy i wykonane na przykładzie dwóch funkcjonalnie identycznych aplikacji napisanych za pomocą wieloplatformowego frameworka Xamarin i natywnych narzędzi tworzenia aplikacji mobilnych dla systemu operacyjnego Android

W dzisiejszych czasach technologie informatyczne rozwijają się w szybkim tempie i wciąż pojawiają się nowe rozwiązania, począwszy od nowego języka programowania, kończąc nowymi rozwiązaniami sprzętowymi. Aby deweloperzy korzystający z platformy .NET mogli maksymalnie wykorzystać swoje możliwości w zakresie tworzenia oprogramowania na rynku wieloplatformowych rozwiązań został przedstawiony framework Xamarin[1].

Ten framework jest doskonałym przykładem na to, jak firmy w branży rozwoju oprogramowania mobilnego konkurują między sobą, poprzez zapewnienie twórcom oprogramowania znanych interfejsów dla nieznanych urządzeń fizycznych lub platform mobilnych.

Każda platforma ma własną charakterystykę i w związku z tym bardzo trudno jest stworzyć naprawdę uniwersalny interfejs, który pozwalał by w 100% wykorzystać możliwości danej platformy. Wspólnym wyznacznikiem dla takich interfejsów jest wskaźnik przenośności kodu, który określa jaka ilość kodu jest znormalizowana dla wszystkich środowisk i nie wymaga zmian podczas tworzenia aplikacji dla innych platform [2]. Każdy producent takich środowisk programowania wieloplatformowego stara się przybliżyć ten wskaźnik do 100%. Jednym z takich frameworków jest Xamarin, który został powołany w celu ujednolicenia wszystkich platform w jeden interfejs. Jak zostało wspomniane, trudno stworzyć naprawdę uniwersalny interfejs

[3]. Xamarin też ma swoje ograniczenia, przy wytwarzaniu oprogramowania, ponieważ wirtualna maszyna Java nie pozwała korzystać z dynamicznych obiektów takich jak *LINQ* [5].

2. Metodyki porównania środowisk

W aplikacjach które opracowano do testowania wydajności, dodano kilka testów symulujących niektóre obciążenia systemu. W oparciu o nie można uzyskać wyniki dotyczące wydajności i wyciągnąć pewne wnioski. Aplikacje będą posiadały te same zestawy funkcjonalności. Zostaną utworzone one za pomocą języków Java oraz C# w odpowiednich środowiskach Android Studio i Visual Studio dla systemu operacyjnego Android. Pomiar czasu testów aplikacji odbywa się za pomocą natywnych dla używanej platformy klas i metod pomiaru czasu.

Analiza ilości kodu. Analiza porównania kodu wykonana została za pomocą programu „*LocMetrics*" [4]. Ten program pozwala przeprowadzić analizę kodu według różnych kryteriów.

Szybkość generowania graficznego interfejsu użytkownika. Ten test pokazuje prędkość generowania niestandardowych elementów sterowania tworzonych dynamicznie w trakcie działania aplikacji. Podstawowa idea tego testu polega na dodaniu do kontenera *ListView*, elementu podrzędnego *ListViewItem*. Liczba dodawanych elementów jest zdefiniowana przez użytkownika w głównym widoku tego testu. Liczba elementów służąca do generowania interfejsu użytkownika wprowadzona została w liczbie stu tysięcy.

Przetwarzanie złożonych zapytań do bazy danych. Ten test pokazuje prędkość działania i wykonywania operacji *CRUD* (*ang. Create, Update Delete*) w aplikacji. System operacyjny Android pozwala korzystać z lekką bazę danych *SQLite*, która zostanie wykorzystana w tym teście. Dla ułatwienia pracy z tą bazą danych i podążać koncepcji OOP(*ang. Object-oriented programming*) podczas pracy z nią, zostaną wykorzystane ORM-narzędzie *ORMLite*, które są dostępne dla obu platform. Osłona *ORMLite* oferuje obiektowy interfejs pracy z bazą danych *SQLite*. Podczas korzystania z tego narzędzia, należy określić najpierw model tabeli bazy danych, która zostanie wykorzystana w teście.

Analiza pracy pętli z dużą ilością iteracji. Test ten jest dość prosty, sprawdzany jest czasu trwania cyklu z dużą liczbą powtórzeń. Liczba iteracji jest określana przez użytkownika na widoku tego testu, wprowadzając liczby. Głównym celem tego badania jest potwierdzenie lub obalenie hipotezy o tym, że Xamarin nie ustępuje w działaniu aplikacja na telefony narzędzi rozwoju, mimo że Xamarin jest dodatkowym poziomem abstrakcji nad środowiskiem natywnym. Podczas wykonywania tego testu, podjęto decyzję o tym jak testować pętle w warunkach wykonywania stu milionów iteracji.

Porównanie pracy kamery. W tym teście, prowadzone są badania w pracy kamery na różnych platformach. Głównym wskaźnikiem, na którym odbywa się porównania jest liczba klatek na sekundę, które są generowane przez aparat. Główna idea testu polega na wywołaniu kamery z głównego widoku testu, po czym, użytkownik jest zmuszony nagrać film. Po nagraniu wideo, plik filmu jest poddawany przetwarzaniu przez klasą *MediaExtractor*. Klasa ta pozwala wyodrębnić z pliku wideo metadane, wśród których dostępna jest wartość liczby klatek na sekundę. W trakcie badań nagrano trzy pliki wideo za pomocą wywołania kamery z natywnej aplikacji i trzy wideo za pomocą wywołania kamery z aplikacji opartej na frameworku Xamarin.

Analiza możliwości przetwarzania plików. W tym teście sprawdzane są możliwości pracy z plikami i wykonanie pomiarów ich konwersji. Zasada testu opiera się na konwersji plików binarnych do formatu ZIP i odwrotnie. Podczas działania tego testu, system odczytuje binarną zawartość plików, kompresuję i zapisuje pliki w lokalnym repozytorium. Do wykonania tego testu, z góry zostało wygenerowane tysiąc plików binarnych w różnych rozmiarach od zera do tysięcy bajtów. W tym teście, jest pomiar czasu kompresji plików oraz czas dekompresji.

3. Obiekt badań

Obiektami badań w pracy zostały dwie aplikacje, które były wytwarzane za pomocą frameworku Xamarin oraz natywnych narzędzi wytwarzania oprogramowania. W aplikacjach które zostały opracowane do testowania wydajności, dodane zostały kilka testów, symulujących niektóre obciążenia systemu, w wyniku działania których można uzyskać wyniki dotyczące wydajności i wyciągnąć pewne wnioski. Aplikacje będą posiadały te same zestawy funkcjonalności. Zostaną utworzone one za pomocą języków Java oraz C# w odpowiednich środowiskach Android Studio i Visual Studio dla systemu operacyjnego Android. Pomiar czasu testów aplikacji odbywa się za pomocą natywnych dla używanej platformy klas i metod pomiaru czasu.

4. Metoda badawcza

Porównanie aplikacji testowych zrealizowane będzie w oparciu o wybrane kryteria analizy i wykonane na przykładzie dwóch funkcjonalnie identycznych aplikacji napisanych za pomocą wieloplatformowego frameworka Xamarin i natywnych narzędzi tworzenia aplikacji mobilnych dla systemu operacyjnego Android.

W pracy postawiono następującą tezę: Platforma Xamarin zapewnia bardziej wydajny sposób budowy aplikacji dla systemu operacyjnego Android niż oprogramowanie natywne.

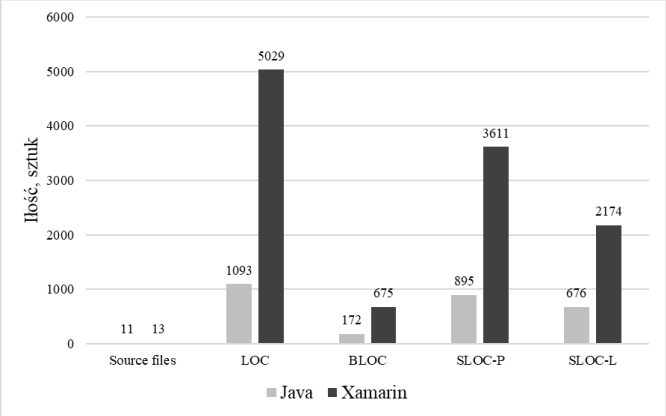
Hipoteza 1: Framework Xamarin generuje więcej kodu dla tego, że nie jest natywnym narzędziem do tworzenia aplikacji do systemu Android.

Hipoteza 2: Xamarin teoretycznie może być wolniejszy w kontekście szybkości generowania graficznego interfejsu użytkownika i przetwarzania multimediów, dla tego, że wywołuje bezpośrednio funkcje Javy.

Aby analiza była bardziej oczywista i jawna, do oceny wyników badań będą wykorzystywane liczby zwane wagami, Każda waga określa częstotliwość wykorzystania danego testu w stosunku do innych testów. Głównym celem danego testu jest ustalenie sumarycznej liczby wag dla każdej platformy. Zakres liczb będzie od 1 – najmniej stosowane do 3 – najwięcej stosowane.

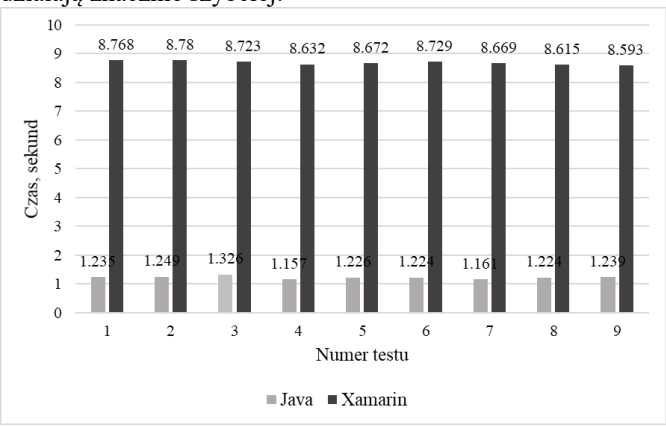
5. Rezultaty badań

Analiza rezultatu pomiarów ilości kodu. Analiza ilości kodu pokazała ogromna różnicę między dwoma projektami. Aplikacja napisana przy pomocy natywnych narzędzi, zajmuje mniej linii kodu oraz ma bardziej zoptymalizowane pliki z punktu widzenia ilości kodu, niż aplikacja napisana w języku C#. Przede wszystkim wynika to z cech składni języków C# i Java. Należy zauważyć, że tworzenie aplikacji wieloplatformowych za pomocą Xamarin, wymaga dodania do zwykłej natywnej aplikacji dodatkowej warstwy abstrakcji, co odbija się na wydajności i liczbie dodatkowo wygenerowanych plików.



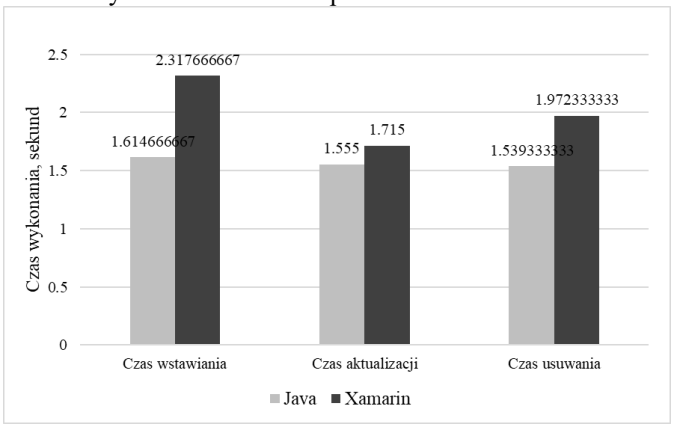
Rys. 1. Porównanie metryk kodu

Analiza rezultatów dynamicznego generowania interfejsu użytkownika. Podczas przeprowadzania testu tworzenia graficznego interfejsu użytkownika, okazało się, że dynamiczne generowanie interfejsu użytkownika szybciej odbywa się w natywnych środowiskach wykonawczych z systemem Android. Można przypuszczać, że dzieje się tak dlatego, że natywne środowisko wymaga na etapie kompilacji informacji na temat kontrolowanych obiektów. Różnica okazała kolosalna, aplikacja na podstawie frameworka Xamarin generuje elementy graficzne użytkownika sześć-siedem razy wolniej niż środowisko natywne. W wyniku badań, należy zauważyć, że mechanizmy natywnych narzędzi działają znacznie szybciej.



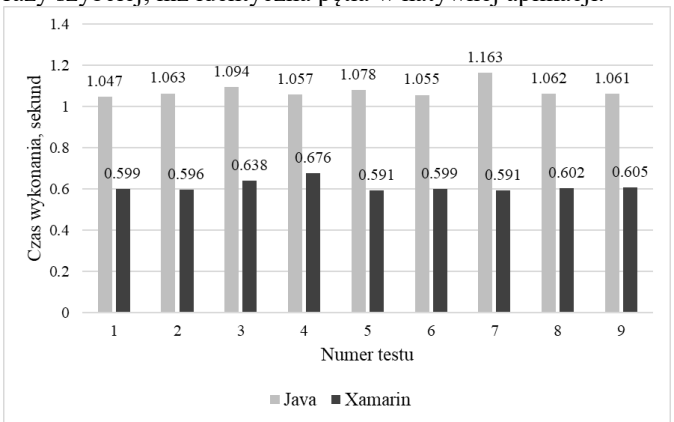
Rys. 1. Porównanie wyników testowania generowania graficznego interfejsu użytkownika

Porównanie rezultatów działania bazy danych. W teście z badaniem wydajności wbudowanych baz danych została sprawdzona szybkość wykonywania zapytań do bazy danych. Warunki testu były takie same w obu aplikacjach - wykorzystano obiektowe narzędzie *SQLite* do bezpiecznego korzystania z narzędzi bazy danych. W wyniku przeprowadzonych testów, okazało się, że natywne środowisko działa szybciej, choć różnica nie jest duża. Wyniki testów nie były nieoczekiwane, jak wspomniano wcześniej, nad natywnym środowiskiem wykonania jest środowisko wykonania Mono [3], która wymaga dodatkowych nakładów czasu procesora.



Rys. 2. Porównanie wyników testowania *CRUD* operacji

Analiza wyników testu pętli z dużą ilością iteracji. Analiza działania pętli z dużą liczbą iteracji przedstawiła niejednoznaczne wyniki. Działanie pustej pętli w aplikacji na bazie frameworka Xamarin okazało się szybsze, niż w zwykłej aplikacji napisanej przy pomocy języka Java, pomimo tego, że liczba iteracji była sto milionów. Biorąc pod uwagę to zachowanie, można przypuszczać, że wynikało ono z działania kompilatora .NET który optymalizuje kod na etapie kompilacji w wyniku czego puste wywołanie cyklu zostało zoptymalizowane. Ostatecznie, wyniki były takie, że pętla w aplikacji na bazie frameworka Xamarin działa o dwa razy szybciej, niż identyczna pętla w natywnej aplikacji.

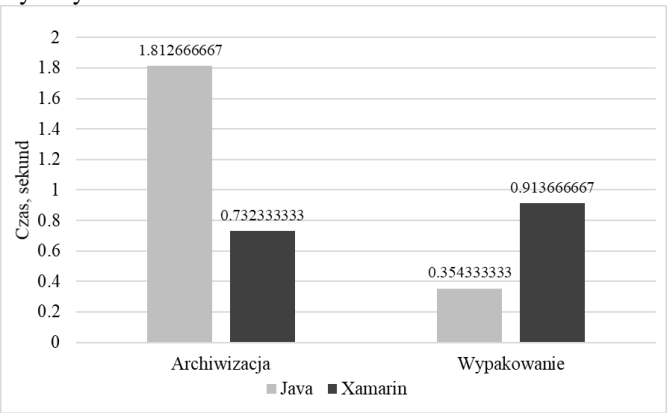


Rys 3. Porównane wyniki testowania czasu działania pustej pętli

Analiza wyników pracy kamer. Podczas porównywania pracy kamer została stworzona przez dwa dość proste skrypty do testowania. Aby rozpocząć test, użytkownik musiał nagrać film, a później system analizował go. Kryterium, według którego była porównana wydajność zostało oparte o liczbę klatek na sekundę każdego nagranego wideo. W efekcie końcowym okazało się, że używana platforma nie ma wpływu na pracę kamery. Należy przypuszczać, że wynika to z faktu, że w tym teście zostały wykorzystane mechanizmy intencji, co oznacza, że obiekt

kamery jest częścią serwisu, który dostarcza informacji o kamerze zainstalowanej na danym urządzeniu. Na tej podstawie można stwierdzić, że środowisko wykonania, z którego została wywołana kamera w żaden sposób nie wpływa na jej wydajność, tak jak ma to miejsce w przypadku wywołania obiektu kamery, która jest wykorzystywana przez zewnętrzną aplikację systemową.

Badanie rezultatów testu przetwarzania plików. Podczas badania możliwości środowisk uruchomieniowych oraz pracy z przetwarzaniem plików, został porównany czas konwersji plików binarnych do formatu ZIP i odwrotnie. Przede wszystkim, warto zauważyć, że natywne narzędzie programistyczne nie zawiera interfejsu, który oferowałby wszystkie niezbędne funkcje do pracy z kompresją plików w formacie ZIP. Wyniki porównania są niejednoznaczne. Czasy archiwizacji okazały się dwa razy lepsze niż aplikacja bazująca na frameworku Xamarin, podczas gdy czas rozpakowania okazał się lepszy w natywnych narzędziach wykonywania.



Rys. 4. Porównanie wyników czasu przetwarzania plików

Po przeprowadzeniu wszystkich testów, porównań i analiz należy zauważyć, że wyniki okazały się mieszane. Przede wszystkim warto zauważyć, że obie hipotezy były potwierdzone, a mianowicie, w przypadku tworzenia aplikacji dla systemu operacyjnego Android za pomocą Xamarin, generowane jest znacznie więcej kodu niż przy opracowywaniu natywnymi narzędzi oraz to że aplikacja na bazie frameworka Xamarin działają wolniej niż zwykła natywna aplikacja. Na podstawie wyższych analiz poszczególnych testów została stworzona podsumowująca tabela 5.1.

Jak widać z rezultatów sumarycznych wag, natywne narzędzie wytwarzania aplikacji Java, ma 80 punków, w ten czas jak narzędzie Xamarin ma 60. Każdy z systemów mógł otrzymać maksymalnie 100 punktów. Składowe testy, których wyniki były na końcu sumowane mają różną wartość użytkową. Dla przykładu badanie z analizą działania bazy danych jest jednym z najważniejszych i można było zyskać tutaj 20 punktów. Z kolei test pracy kamery nie jest, aż tak istotny dlatego maksymalny wynik jaki system mógł uzyskać to 5.

Teza, która zdefiniowana dla danej pracy została podważona Framework Xamarin nie pozwala tworzyć bardziej wydajnych aplikacji, jak ma to miejsce w przypadku zwykłej aplikacji z dodatkiem środowiska wykonania Mono. W niektórych przypadkach środowisko to wymaga jednak czasu procesora, z czego wynika wolniejsze działania aplikacji, co zostało udowodnione w danej pracy.

Tabela 1. Podsumowanie punktów poszczególnych testów

Nazwa testu	Punkty		
	Xamarin	Java	Wartość maksymalna
Analiza ilości kodu	5	15	20
Szybkość generowania interfejsu użytkownika	5	15	20
Działanie bazy danych	12	17	20
Pętla z dużą ilością iteracji	18	13	15
Analiza działania kamery	5	5	5
Przetwarzanie plików	15	15	20
Suma wag:	60	80	100

6. Wnioski

W tej pracy było zbadano jedno z narzędzi dla wieloplatformowego wytwarzania oprogramowania. Framework Xamarin na pewno jest wartym uwagi rozwiązaniem dla branży rozwoju oprogramowania mobilnego, ponieważ pozwala tworzyć produkty, nie zagłębiając się w szczegóły konkretnych platform z punktu widzenia wytwarzania oprogramowania.

Po realizacji badania i po otrzymaniu wyników można wyciągnąć kilka wniosków na temat możliwości zastosowania tego frameworka. Z punktu widzenia programowego interfejsu, który zapewnia jednolity dostęp do wszystkich głównych platform programowania, jest to doskonałe narzędzie, który ma szeroki wachlarz możliwości, dokumentację i dużą ilość materiałów. Niemniej jednak warto zauważyć, że ten framework okazał się gorszy z punktu widzenia wydajności w odnoszeniu do natywnych narzędzi działania aplikacji. Sądząc po wynikach, można powiedzieć, że w czasie działania aplikacji na bazie frameworka Xamarin może czasami dość znacznie ustępować natywnym narzędzi, co może być istotne dla niektórych aplikacji.

Można powiedzieć, że zastosowanie tego frameworka tak samo może się różnić w zależności od wymagań aplikacji. Jeśli aplikacja nie pracuje na dużej ilości danych i nie wymaga dynamicznego generowania złożonego interfejsu użytkownika, to ten framework może być zastosowany do realizacji aplikacji mobilnych. W pozostałych przypadkach, framework ten nie jest znacznie gorszy od natywnego środowiska uruchomieniowego. Warto również zauważyć, że przy wszystkich wadach badanego frameworku opisanych wyżej, koszt wytwarzania aplikacji za pomocą narzędzia Xamarin jest znacznie mniejszy.

Literatura

[1] M. Reynolds, Xamarin Mobile Application Development for Android, Packt Publishing Ltd, 2014.

[2] D. Hermes, Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals, Apress, 2015.

[3] M. Reynolds, Xamarin Essentials, Packt Publishing Ltd, 2014.

[4] R. Meier, Professional Android 4 Application Development, John Wiley & Sons, 2012.

[5] M. Dave, A. Grant and S. Komatineni, Pro Android 5, Apress, 2015.

[6] Z. Mednieks, L. Dornin, M. Nakamura and Blake Meike, Programming Android, 2nd Edition, O'Reilly Media, Incorporated, 2012.

[7] J. Masner, P. Šimek, J. Jarolímek and I. Hrbek, "Mobile Applications for Agricultural Online Portals – Cross-platform or Native Development," June 2015. [Online]. Available: https://www.researchgate.net/publication/283878658_Mobile_Applications_for_Agricultural_Online_Portals_-_Cross-platform_or_Native_Development. [04.06.2017].

[8] V. Tunalı and S. Z. Erdogan, "Comparison of Popular Cross-Platform Mobile Application Development Tools," October 2015. [Online]. Available: https://www.researchgate.net/publication/282816272_Comparison_of_Popular_Cross-Platform_Mobile_Application_Development_Tools. [04.06.2017].

[9] D. Hermes, "Mobile Development Using Xamarin," January 2015. [Online]. Available: https://www.researchgate.net/publication/300266365_Mobile_Development_Using_Xamarin. [04.06.2017].

[10] L. A. B. Prieto, Z. Komínkova-Oplatková, R. T. Frías and J. L. E. Hernández, "A time performance comparison of particle swarm optimization in mobile devices," January 2016. [Online]. Available: https://www.researchgate.net/publication/309361796_A_time_performance_comparison_of_particle_swarm_optimization_in_mobile_devices. [04.06.2017].