sciendo

Műszaki Tudományos Közlemények vol. 15. (2021) 61–64.
DOI: Magyar: https://doi.org/10.33895/mtk-2021.15.12
Angol: https://doi.org/10.33894/mtk-2021.15.12

EME MTSZ MTK

Összehasonlító elemzés a natív iOS- és Cross-Platform iOS-alkalmazásfejlesztésről

Comparative Analysis of Native and Cross-Platform iOS Application Development

Kovács Márk,[1] Johanyák Zsolt Csaba[2]

Neumann János Egyetem, GAMF Műszaki és Informatikai Kar, Informatika Tanszék, Kecskemét, Magyarország

[1] *kovacs.mark@gamf.uni-neumann.hu*

[2] *johanyak.csaba@gamf.uni-neumann.hu*

Abstract

Nowadays, mobile applications are developed for more and more areas, providing great help for our everyday lives. When designing a mobile application, the first important decision to make is to choose the targeted platform. Is it only phone or tablet as well? Should the app run on Android or iOS, or should it be available on both mobile operating systems? In the latter case, besides the native development environments, it is worth considering a cross-platform development environment to write the software. This study investigates both the development and performance aspects of some possibilities for iOS application development, namely, native iOS development in Xcode, Xamarin.iOS, and Xamarin.Forms frameworks.

Keywords*: iOS, native, cross platform, Xamarin, Xamarin.iOS, Xamarin.Forms.*

Összefoglalás

Manapság egyre több területen készül mobilalkalmazás, mely nagyban segítheti hétköznapjainkat is. Egy mobilalkalmazás tervezése során először is fontos meghatározni, hogy milyen platformra fej-lesztjük azt. Csak telefonra vagy tabletre is? Android és/vagy iOS-operációs rendszerre? Ha mindkét rendszert megcélozzuk, akkor érdemes átgondolni, hogy melyik fejlesztőkörnyezetben írjuk meg a szoftvert. Abban az esetben, ha több platformra is fejlesztünk, akkor érdemes megvizsgálni a natív fejlesztői környezeteken kívül az úgynevezett Cross-platform-megoldásokat is. Ez a tanulmány az iOS-applikációfejlesztés néhány lehetőségét: a natív iOS-fejlesztést Xcode-ban, illetve a Xamarin.iOS- és Xamarin.Forms-keretrendszerek használatát elemzi, mind fejlesztési, mind teljesítményszempontok figyelembevételével.

Kulcsszavak*: iOS, natív, cross-platform, Xamarin, Xamarin.iOS, Xamarin.Form.*

1. Bevezetés

Egyre több és komplexebb alkalmazásra van igény a dinamikusan bővülő technológiai piacon. A fejlesztőknek számos kihívással kell szembenézniük, amelyeknek különböző megoldásokkal felelhetnek meg. Ha egy alkalmazásnak működnie kell mindkét legnépszerűbb mobiloperációs rendszeren, Androidon és iOS-en is, akkor érdemes átgondolni a rendelkezésre álló lehetőségeket a fejlesztői keretrendszerek tekintetében. Tisztában kell lenni milyen előnyökkel, esetleges hátrányokkal járhat a kiválasztott környezet.

Jelen tanulmány a natív és a keresztplatformos iOS-alkalmazásfejlesztés közötti különbséget vizsgálja az Xcode- és a Xamarin-fejlesztőrendszerek felhasználásával. A következőkben áttekintjük a felhasznált környezeteket és programozási nyelveket.

A Xamarin esetében két lehetőséget is vizsgáltunk. Az egyik a Xamarin.iOS, mellyel csak iOS-operációs rendszert futtató eszközre tudunk

fejleszteni. Felületi szerkesztője nagyban hasonlít az Xcode-ban használt Storyboard szerkesztő felületéhez. A másik lehetőség a Xamarin.Forms, mellyel Android operációs rendszerre is tudunk alkalmazást készíteni.

1.1. Fejlesztőkörnyezetek

Az okostelefonos alkalmazásokat három nagy kategóriába sorolhatjuk a fejlesztés során alkalmazott technológiák és környezetek alapján, ezek a natív alkalmazások, webalkalmazások és a hibrid vagy más néven cross-platform-alkalmazások. A natív alkalmazások esetében csak egy specifikus környezetben, egy bizonyos operációs rendszercsoportra tudunk fejleszteni. Androidra Android Studióval, míg iOS-re Xcode IDE segítségével készíthetünk alkalmazásokat. Ezzel szemben a cross-platform-alkalmazások esetében több operációs rendszerre is fejleszthetünk egyszerre.[1]

1.1.1. Xcode

Az Xcode az Apple által fejlesztett IDE, melynek segítségével készíthetünk natív alkalmazásokat mindegyik rendszerére, iOS-re, iPadOS-re, watchOS-re, macOS-re vagy akár tvOS-re. A legtöbbször iOS-re vagy iPadOS-re szokás alkalmazásokat fejleszteni. Amikor Xcode-ban fejlesztünk, rendelkezésünkre áll egy Interface Builder a felület kialakításához, amiben egy Storyboard található. Ebben grafikusan tudjuk kialakítani a felületünket. Az Apple korábban az Objective C programozási nyelvet támogatta, azonban 2014-től saját programnyelvet fejlesztett ki Swift néven. [2]

1.1.2. Visual Studio 2019, Xamarin

Manapság az egyik lenépszerűbb fejlesztőkörnyezet a Visual Studio, mellyel sokféle szoftvert készíthetünk több platformra is. Alapvető programozási nyelvei a C-típusú nyelvek, ezen belül is a C#, mely .NET-alapokra épül.

A mobilalkalmazások készítéséhez a Microsoft által favorizált keretrendszer a Xamarin, amely több platformon működő alkalmazások fejlesztésére is alkalmas. A Xamarin egy nyílt forrású eszköz, amelyet a Microsoft által 2016-ban felvásárolt Xamarin nevű cég fejleszt. Kétféle lehetőséget biztosít a cross-platform-alkalmazások készítésére. [3]

Az egyik a Xamarin Native, amellyel az Android és az iOS SDK-k felhasználásával C# nyelven tudunk alkalmazásokat készíteni. Az egyik tesztprogramunk egyik alrendszere a Xamarin.iOS segítségével készült. Itt az Xcode-ban használt Interface Builderhez sokban hasonló módon tudjuk alakítani a felületet. Ez azok számára lehet egy kedvező, akik nem szeretnek valamilyen okból az Xcode-ban Swift nyelven fejleszteni. Emellett Mac-gépen fejlesztve az alkalmazást, a Visual Studio felajánlja a felület Xcode-ban történő elkészítésének lehetőségét is. [4, 5]

A másik lehetőség a Xamarin.Forms. Ez egy platformfüggetlen megoldást tesz lehetővé, amellyel lehetőségünk van általában kevesebb programozással mindkét platformra fejleszteni. Itt minden megcélzott platformhoz a fejlesztőrendszer generál egy külön projektet, és emellett külön készül egy projekt azon kódrészeknek, amelyek mindkét platform számára azonosak. Azokat a platformspecifikus részeket, amelyeket nem tudunk a közös kódba írni, a külön projektekben szerkeszthetjük. A felületet az Android Studiótól és az Xcode-tól eltérően nem tudjuk grafikusan kialakítani, hanem az XAML nyelv segítségével kell leírnunk azt. [6, 7]

1.2. Programozási nyelvek

1.2.1. Swift

Az Apple által fejlesztett és használt Swift programozási nyelv egyre népszerűbb az alkalmazásfejlesztők körében is. Nemcsak mobil alkalmazások fejlesztésére használható, hanem akár asztali vagy felhőalapú szolgáltatásokat is készíthetünk vele. A Swift egy erősen típusos nyelv, ebből az is következik, hogy biztonságos. Megtalálhatók benne a C-típusú nyelvek elemei is. A Swiftben megírt kód kimondottan könnyen átlátható és olvasható, a nyelv viszonylag hamar elsajátítható. [4, 8]

1.2.2. C#

A Microsoft által a .NET-hez fejlesztett C# egy általános célú, magas szintű objektumorientált nyelv, ami számos rokon vonást mutat a C++-szal és a Javával. Használata kényelmes és gyors alkalmazásfejlesztést tesz lehetővé. A C# is erősen típusos. Bár jelenleg csak nyolcadik helyen áll az IEEE Spectrum mobilfejlesztésre használt nyelvek ranglistáján [9], de határozott előnye a listában előtte szereplő nyelvekkel szemben, hogy támogatja mind a négy nagy platformcsaládra (web, asztali, mobil és beágyazott) történő fejlesztést.

2. Összehasonlítások, tesztek

2.1. Tesztalkalmazás bemutatása

Vizsgálatunk során készítettünk egy viszonylag egyszerű alkalmazást mindhárom környezetben: Xcode-ban, Xamarin.iOS-ben, illetve Xamarin.

Forms-ban. A program egy listát tartalmaz, ami 20 db. előre definiált elemből áll, melyeket törölhetünk, illetve egy sor kiválasztása után megjeleníthetünk egy Alert ablakban. Új elem hozzáadása egy szövegbeviteli mező és egy nyomógomb segítségével lehetséges. Célunk a teljesítmény összehasonlítása három alkalmazás tekintetében.

A felületet meglehetősen gyorsan ki tudtuk alakítani, és mindhárom alkalmazás esetében hasonló végeredményt kaptunk, amit az 1. és a 2. ábra mutat be. Az 1. ábra jól szemlélteti az alkalmazás felületét és működését Xcode-ban. Példánkban a lista ételek neveit tárolja el.

A Xamarin.iOS és a natív iOS esetében könnyen ki lehetett alakítani egy szinte teljesen megegyező elrendezésű felületet, miközben a Xamarin.Forms esetében az XAML nyelv használta miatt kicsit eltérő felületet kapunk.

2.2. Teszteléshez használt szoftver és eszköz

A teszteléshez az Xcode tesztelő programját használtuk, az Instruments szoftvert. Ebben több lehetőség áll rendelkezésre a teszteléshez. Le tudjuk mérni az alkalmazás indítási idejét, illetve a használat során látható és rögzíthető a fejlesztett alkalmazás hardverigénye. Nyomon követhető a processzorigény, ami az alkalmazás gyorsaságát nagy mértékben befolyásolja.

A tesztelés során öt főbb funkció működését vizsgáltuk:

– eindulás,
– görgetés,
– sor kiválasztása,
– sor törlése,
– új elem hozzáadása.

A tesztkészülék egy iPhone XS 64 GB-os, 2×2,5 GHz-es és 4×1,59 GHz-es hatmagos készülék volt. Az előbbi felsorolás szerinti sorrendben történt meg mindhárom applikáció tesztelése.

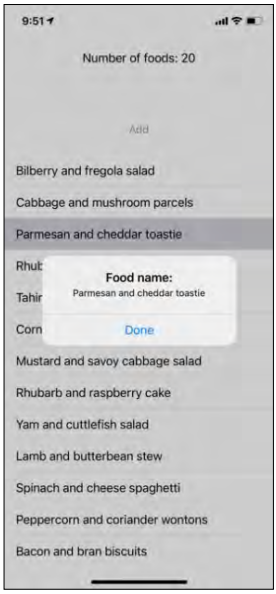
2.3. Tesztelések eredményei

A tesztelések eredményeit táblázatokkal szemléltettük. Az egyes műveletekhez szükséges minimális és maximális processzorigényeket, illetve a felhasznált magok számát jelöltük.

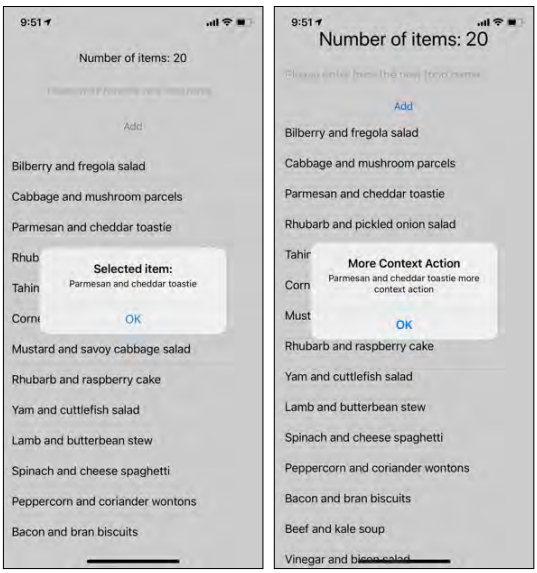
2.3.1. Xcode-ban fejlesztett alkalmazás

Az alkalmazás 384 ms alatt indult el, processzorigénye az 1. táblázatban látható.

Az alkalmazás futása során, ha előtérben volt, de nem használtuk egyetlen funkcióját sem, akkor nem használt fel mérhető processzor-erőforrást.



1. ábra. *Xcode-ban készült alkalmazás*



2. ábra. *Xamarin.iOS-ben (bal oldal) és Xamarin. Forms-ban (jobb oldal) írt alkalmazás felülete*

1. táblázat. *Natív iOS alkalmazás processzorigénye*

Funkció	Min %	Max %	CPU mag
elindulás	10	170	6
görgetés	10	120	5
kiválasztás	10	110	5
törlés	10	110	5
hozzáadás	10	120	6

2. táblázat. *Xamarin.iOS-alkalmazás processzorigénye*

Funkció	Min %	Max %	CPU mag
elindulás	10	120	6
görgetés	10	90	4
kiválasztás	10	110	5
törlés	10	110	4
hozzáadás	10	130	6

3. táblázat. *Xamarin.Forms-alkalmazás processzorigénye*

Funkció	Min %	Max %	CPU mag
elindulás	50	130	6
görgetés	10	100	6
kiválasztás	10	130	6
törlés	10	110	4
hozzáadás	10	120	6

2.3.2. Xamarin.iOS-ben fejlesztett alkalmazás

Az alkalmazás processzorigénye a **2. táblázat**-ban látható. 2,4 másodperc alatt indult el, ami meglehetősen magasnak számít, ekkora alkalmazásnál. Hasonlóan a natív alkalmazáshoz, itt sem igényelt szinte semmilyen processzor-erőforrást nyugalmi állapotban.

2.3.3. Xamarin.Forms-ban fejlesztett alkalmazás

Az alkalmazás processzorigénye a **3. táblázat**-ban látható. 665 ms alatt indult el, ami a natív iOS-alkalmazáshoz viszonyítva jónak mondható. Nyugalmi állapotban is volt egy állandó 10%-os processzorigény.

3. Összefoglalás, következtetések

Összességében mindhárom alkalmazás jól teljesített, azonban néhány szempontból mégis eltérnek.

A fejlesztés szempontjából a natív és a Xamarin. iOS nagyon közel áll egymáshoz. Hasonlóan megvalósíthatók a funkciók mindkettőben. Xamarin. Forms esetében a felületet XAML-ben kell megírni, és a tesztelés során használt Visual Studio 16.9.1-ben csak utána láthatjuk, hogy miként jelennek meg az egyes vezérlők. Ez a vizuális tervezést kedvelőknek hátrány lehet.

Az indításakor a natív és a Xamarin. Forms-alkalmazások nyújtották a legjobb teljesítményt, míg a Xamarin.iOS sokkal gyengébben teljesített. A natív alkalmazás, habár a legtöbb erőforrás használatával, de a leggyorsabban indult el. Mindhárom alkalmazás esetében a legnagyobb erőforrásigényt az indítás jelentette. A memória tekintetében nem történt jelentős fogyasztás egyik esetében sem.

A Xamarin.Forms nyugalmi állapotban is egy minimális, állandó 10%-os processzorterhelést eredményezett. A tesztalkalmazás funkciói tekintetében mindegyik különböző értékekkel teljesített, azonban összességében hasonlóan teljesítettek.

Köszönetnyilvánítás

Köszönettel tartozunk a kutatás támogatásáért, amely az EFOP-3.6.1-16-2016-00006 „A kutatási potenciál fejlesztése és bővítése a Pallasz Athéné Egyetemen” pályázat keretében valósult meg. A projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával, a Széchenyi 2020 program keretében valósul meg.

Szakirodalmi hivatkozások

[1] Molenda, D., Skublewska-Paszkowska M.: *Analysis of the Possibility of Shortening the Time of Creating a Mobile Application for Android and iOS Systems Using Xamarin Technology*. Journal of Computer Sciences Institute, 12. (2019) 226–231. https://doi.org/10.35784/jcsi.493

[2] Swift, About Swift. (letöltve: 2021. február 26.) https://swift.org/about/

[3] Prajapati M., Phadake D., Poddar A.: *Study on Xamarin cross-platform framwork*. International Journal of Technical Research and Applications, 4/4. (2016) 13–18.

[4] Vishal K., Kushwaha A. S.: *Mobile Application Development Research Based on Xamarin Platform*. 4th International Conference on Computing Sciences (ICCS), Jalandhar, India, 2018, 115–118. https://doi.org/10.1109/ICCS.2018.00027

[5] Ebone A., Tan Y., Jia X.: *A Performance Evaluation of Cross-Platform Mobile Application Development Approaches*" IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft), Gothenburg, Sweden, 2018, 92–93.

[6] Pawel G., Maria S.-P., Edyta L., Jakab S.: *Performance Analysis of Native and Cross-Platforms Mobile Applications*, IAPGOŚ 2/2017, (2017) 50–53. https://doi.org/10.5604/01.3001.0010.4838

[7] Altersoft, The Good and The Bad of Xamarin Mobile Development, 2020. (letöltve: 2021. 02. 28.) https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/

[8] Bilberg D.: *Comparing Performance between React Native and Natively Developed Smartphone Applications in Swif., A Comparative Analysis and Evoluation of the React Native Framework.* https://www.diva-portal.org/smash/get/diva2:1215717/FULLTEXT01.pdf

[9] IEEE *Spectrum Interactive: The Top Programming Languages*. [Megtekintve: 2021.03.14.] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020