Joonas Hopiavuori

Xamarin ja sen soveltuvuus Atk-Palvelu Luhtasaari Oy:n käyttöön

Opinnäytetyö
Kevät 2019
SeAMK Tekniikka
Tietotekniikan tutkinto-ohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Joonas Hopiavuori

Työn nimi: Xamarin ja sen soveltuvuus Atk-Palvelu Luhtasaari Oy:n käyttöön

Ohjaaja: Hilkka Niemelä

Vuosi: 2019 Sivumäärä: 35 Liitteiden lukumäärä: 0

Tässä työssä tutkitaan Xamarin-mobiilisovelluskehitysympäristöä ja miten se sopisi Atk-Palvelu Luhtasaari Oy:n käyttöön. Atk-Palvelu Luhtasaari Oy kehittää SukuJutut-sukuselvitysohjelmistoa PC-tietokoneille ja on kiinnostunut luomaan asiakkailleen mobiiliversion PC-version rinnalle. SukuJuttujen mobiiliversiolla olisi tarkoitus lukea ohjelman PC-versiolla luotuja aineistoja.

Työssä tutustutaan yleisimpiin mobillialustoihin sekä niin kutsuttuun Cross-platform-tekniikkaan ja perehdytään Xamarin-sovelluskehitysympäristöön. Xamarinin asennus osaksi Visual Studiota käydään läpi sekä tutustutaan myös hieman Gedcom-standardiin.

Xamarinin avulla luodaan sovellus joka lukee tekstitiedostosta tekstin, ja näyttää sen sovelluksen näkymässä, sekä käydään läpi haasteita, joita sovellusta luodessa kohdattiin.

Avainsanat: Xamarin, Android, iOS, Mobiililaitteet, SukuJutut

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Author: Joonas Hopiavuori

Title of thesis: Xamarin and How it Would Meet the Needs of Atk-Palvelu Luhtasaari Oy

Supervisor: Hilkka Niemelä

Year: 2019 Number of pages: 35

The thesis covered Xamarin mobile development environment and how it would meet the needs of Atk-Palvelu Luhtasaari Oy. Atk-Palvelu Luhtasaari Oy develops genealogy software for PC, called SukuJutut. The company is interested in developing a mobile version of their genealogy software. The mobile version would only read the genealogy data transferred from the main PC version.

The thesis discussed the most common mobile platforms and Cross-Platform development. Xamarin mobile platform development was studied in more detail and it was researched how to implement Xamarin into Visual Studio. Attention was also paid to the GEDCOM standard.

With Xamarin an Android app was made which reads from a text file. Also some of the difficulties with the program were discussed.

Keywords: Xamarin, Android, iOS, mobile platforms, SukuJutut

SISÄLTÖ

Kuvaluettelo

Käytetyt termit ja lyhenteet

Alusta	Käyttöjärjestelmä esim. mobiililaitteessa.
Android	Googlen kehittämä mobiilikäyttöjärjestelmä.
API	Application Programming Interface. Sovelluskehityksessä käytetty apukirjasto, jonka avulla sovellukseen voidaan liittää toiminnallisuuksia vähemmällä koodilla.
CLI	Command-Line Interface. Käyttöliittymä, jonka avulla käyttäjä antaa rivi kerrallaan komentoja ohjelmalle.
Cross-Platform	Sovellus, joka toimii usealla käyttöjärjestelmällä. Ympäristö, jossa usean käyttöjärjestelmän sovelluksia voidaan kehittää.
Gedcom	Standardi aineistojen siirtoon sukuselvitysohjelmien välillä.
iOS	Applen kehittämä mobiilikäyttöjärjestelmä.
Koontiversio	Lähdekoodista julkaisukelpoiseksi paketiksi käännetty sovellus.
Lähdekoodi	Sovelluskehityksessä kehittäjän luoma koodi. Lähdekoodi täytyy kääntää tietokoneen ymmärtämään muotoon.
Natiivi	Tietylle käyttöjärjestelmälle suunniteltu, sisäänrakennettu ominaisuus.
SDK	Software Development Kit. Ohjekmistotyökalu, jonka avulla sovelluskehitys tiettyyn ympäristöön onnistuu.
Xamarin	Sovelluskehitysympäristö, jonka avulla sovelluksia voidaan luoda niin Android- kuin iOS-alustoille.

1 JOHDANTO

1.1 Työn tausta

Atk-Palvelu Luhtasaari on yritys, joka tuottaa SukuJutut sukuselvitysohjelmistoa PC-tietokoneille. Yritys on kiinnostunut laajentamaan ohjelmistoaan myös mobiililaitteille, pääasiassa Android- sekä iOS-ympäristöihin. Jotta mobiilisovellus pysyisi yksinkertaisena ja helppokäyttöisenä, mobiililaitteille luotava sovellus ei välttämättä kattaisi kaikkia PC-version toimintoja, vaan mobiiliversiolla pyrittäisiin mahdollistamaan PC-versiolla luotujen aineistojen luku. Yritys pohti Xamarin-ympäristön käyttöä, sillä se mahdollistaa sovelluskehityksen Android- ja iOS-ympäristöihin C#-kieltä käyttäen. C#-ohjelmointikieli olisi yrityksen kannalta helpompi ottaa käyttöön, sillä se eroaa vain vähän yrityksen käyttämästä VB.NET-ohjelmointikielestä. Tämä ohjelmointikielten samankaltaisuus johtikin yrityksen mielenkiintoon Xamarin-ympäristöä kohtaan, sillä uuden ohjelmointikielen opettelu voi osoittautua työlääksi.

1.2 Työn tavoite

Työn tavoitteena on tutustua mobiiliohjelmointiin ja Xamarin-tekniikkaan. Xamariniin tutustumisen jälkeen luodaan sovellus, jolla testataan Xamarinin sopivuutta yrityksen käyttötarkoitukseen. Sovelluksen tulisi siis lukea tekstitiedosto ja täyttää tekstitiedostossa olevat tiedot sovelluksen näkymään, selkeästi luettavaksi rakenteeksi. Lopuksi tarkastellaan, sopiiko Xamarin yrityksen suunnittelemaan käyttöön.

1.3 Työn rakenne

Luvussa 2 käydään läpi Android- ja iOS-mobiilialustoja sekä cross-platform ympäristöjä. Luvussa 2.3 on tarkasteltu muutamia cross-platform ympäristöjä. Luku 3 käsittelee Xamarinia, sen historiaa, asennusta osaksi Microsoftin Visual Studio -ohjelmistokehitysympäristöä, Xamarin.Essentials-kirjastoa sekä miten yritys

käyttäisi Xamarinia. Luvussa 4 käydään läpi yrityksen SukuJutut-sukuselvitysohjelmaa sekä Gedcom-standardia. Luku 5 kertoo sovelluskehityksestä yrityksen suunnittelemaan mobiiliversioon SukuJutut-ohjelmistosta. Luvussa 6 on yhteenveto Xamarinista sekä ajatuksia työn etenemisestä.

2 MOBIILIALUSTAT

Usein kun puhutaan mobiililaitteista, ensimmäisenä mieleen tulevat vain älypuhelimet ja tabletit. Mutta mobiililaitteita ovat mm. puhelimet, tabletit, älykellot sekä muut puettavat älylaitteet, jopa sellaiset, joita vasta kehitetään tai joista vielä unelmoidaan. Mobiilisovelluksia voidaan siis tehdä huomattavasti useammalle erilaiselle laitteelle kuin vain puhelimille.

Vuoden 2018 syyskuun markkinajakaumaa eri mobiilikäyttöjärjestelmille on mitattu eri mittareilla. IDC Corporate USA:n mittarin mukaan markkinat jakautuvat seuraavasti: Android-laitteita markkinoilla on noin 86,8 %, iOS-laitteita 13,2 % ja muita laitteita alle 0,1 % (IDC [Viitattu 6.4.2019]). Statcounter-sivusto antaa markkinajakaumaksi: Android-laitteita 76,6 %, iOS-laitteita 20,7 % ja muita laitteita 2,7 % (Statcounter 2019). Voidaan siis päätellä, että Android- ja iOS-järjestelmät hallitsevat markkinoita.

2.1 Android

Android-käyttöjärjestelmä on Googlen kehittämä mobiilikäyttöjärjestelmä. Android tuli markkinoille vuonna 2008. (Kurniawan & Deck 2015, 2.) Sovellusten ohjelmointi Android-alustalle tapahtuu Android Studio -ohjelmistoa käyttäen.

Android on Linuxiin pohjautuva käyttöjärjestelmä, joka toimii ns. multi-user-periaatteella. Tämä tarkoittaa sitä, että jokainen Android-sovellus toimii omana käyttäjänään erillisissä prosesseissa. Täten jokainen sovellus Android-käyttöjärjestelmässä toimii irrallaan ja eristyksissä muista sovelluksista. (Kurniawan & Deck 2015, 1.)

Android käyttää ohjelmointikielenään Javaa. Java on alun perin Sun Microsystemsin kehittämä ohjelmointikieli, mutta myöhemmin Oracle osti Sun Microsystemsin (Burd 2016, 15).

Useat laitevalmistajat ovat tuoneet markkinoille laitteita, joissa on vuosien aikana vaihtunut käyttöjärjestelmäversiot uudempiin, monipuolisempiin versioihin. Koska

kuluttajilla on myös vanhempia laitteita käytössään, tulee sovelluskehittäjän harkita sovellusta tehdessään, mille käyttöjärjestelmän versiolle sovelluksensa tekee. Uudemmat käyttöjärjestelmäversiot tuovat mukanaan ominaisuuksia, joita vanhemmista versioista ei löydy, mutta ne rajoittavat laitteiden määrää, joilla sovellus toimii. (Burd 2016, 15.)

Ohjelmointi Android-ympäristössä tapahtuu useimmiten Java-kielellä. Javalla toteutetaan sovelluksen toiminnallisuudet ja käskyt, mutta käyttöliittymä Android-laitteilla käyttää XML-kieltä. XML on myös laajasti käytetty esimerkiksi verkkosivuilla. Pääosa Android-sovelluksen XML-koodista luodaan automaatisesti Android Studiossa, kun käyttöliittymään tehdään muutoksia. Tämä helpottaa sovelluskehitystä, sillä XML saattaa olla paikoin hieman vaikealukuista ja sekavaa. (Burd 2016, 19.)

Valmiin sovelluksen voi julkaista Googlen Play Storeen. Sovelluksen on kunnioitettava Googlen julkaisukäytäntöjä sekä laatuohjeita. Play Storeen julkaistavien uusien sovellusten täytyy tukea vähintään Android 8.0 -versiota (API level 26). Jotta julkaisu onnistuu, täytyy varmistaa, että sovelluksen Manifest-tiedostoon on tehty tarvittavat muutokset julkaisua varten. Näihin muutoksiin lukeutuvat mm. lokimerkintöjen deaktivoiminen sekä android:debuggable-atribuutin deaktivoiminen. Android:debuggable-atribuutti mahdollistaa kehitysvaiheessa sovelluksen testauksen. (Developers [Viitattu 6.4.2019]; Developers [Viitattu 5.4.2019].)

Manifest-tiedostoon tulee myös lisätä sovelluksen versionumero sekä versionimi. Sovelluksesta täytyy luoda koontiversio, joka on paketti sovelluksen tiedostoista, ja jonka loppukäyttäjä lataa Play Storesta omalle laitteellen. Tämä koontiversio tulee allekirjoittaa sähköisesti. Allekirjoitus ja koontiversion luonti onnistuu Android Studion työkalulla. Julkaistava versio sovelluksesta tulee myös testata perusteellisesti vähintään yhdellä puhelimella sekä yhdellä tabletilla, jotta mahdolliset viat voidaan vielä korjata ennen sovelluksen julkaisua. Mikäli sovellus tarvitsee ulkoista palvelinta toimiakseen, tulee tässä vaiheessa varmistaa, että palvelin on käytettävissä ja turvallinen. Sovelluksen Play Store -sivulle tulee asettaa tiedot sovelluksesta, kuten versionumero, sekä sovelluksen kuvaus ja informatiiviset kuvat. Play Store -sivulle täytyy myös asettaa tieto, mille käyttöjärjestelmäversiolle

tai -versioille sovellus on suunniteltu. Kaupan sivulla voi myös määrittää, mikäli sovellukselle haluaa aluekohtaisia rajoituksia tai minkä hintainen sovellus on. Sovellukselle täytyy myös asettaa ikäraja, jotta mahdollisesti nuorille sopimaton materiaali ei päädy nuorten käyttäjien käsiin, sekä sovellukselle ihanteellisen ikäinen yleisö löytää sovelluksen. Kun edellä mainitut asiat on tehty ja tarkastetu, voi sovelluksen julkaista sovelluskauppaan. (Developers [Viitattu 6.4.2019]; Developers [Viitattu 5.4.2019].)

2.2 IOS

IOS on Applen kehittämä mobiilikäyttöjärjestelmä, joka tuli markkinoille 2007. Applen mobiilisovelluskauppa avattiin 2008 ja samalla mahdollisuus saada oma sovellus markkinoille avautui sovelluskehittäjille. (Feiler 2014, 2.)

Applen iOS-sovellusten luontiin käytetään Xcode-ohjelmointiympäristöä ja Swift-ohjelmointikieltä. Jotta sovelluskehitys iOS-laitteille onnistuisi, tarvitaan Applen Mac-tietokone. (Apple 8.12.2016.)

Applen Swift -ohjelmointikieli julkaistiin vuonna 2014. Swiftin edeltäjä oli Objective-C-kieli, jota käytettiin niin iPhone-puhelinten kuin Mac-tietokoneidenkin ohjelmointikielenä. Swift-kielen 5. versio julkaistiin 2019. Uutta Swift 5 -kielessä on mm. vakaa ABI (Application Binary Interface), jonka avulla esim. sovellus ja kirjastot keskustelevat. Päivityksiä tuli myös kirjastoihin. (Kremenek 25.3.2019.)

Jotta sovelluksen voi julkaista Applen App Storeen, tulee varmistaa, että sovellus noudattaa Applen määräämiä kriteerejä. Sovelluksen tulee toimia uusimmilla käyttöjärjestelmäversioilla (iOS, watchOS), joille sovellus on tehty. Sovelluksen koontiversion täytyy olla tehty iOS12.1 SDK -työkalulla tai uudemmalla ja tukea uusimpia laitteita ja niiden suuria näyttöjä. Sovelluksen toiminta tulee testata hyvin, jotta mahdolliset virheet sovelluksen toiminnassa voidaan korjata ennen julkaisua. Sovelluksen App Store -sivulle täytyy myös lisätä sovelluksen tiedot sekä tuotekuvat sovelluksesta. Vasta kun edellä mainitut kohdat ovat kunnossa, voi sovelluksen lähettää Applen tarkistettavaksi. Apple tarkastaa jokaisen App Storeen lisätyn sovelluksen ennen niiden julkaisemista. Kun sovellus on hyväksytty julkaistavaksi,

voi sovelluksen julkaista App Storeen käyttäjien ladattavaksi. (Apple [Viitattu 5.4.2019].)

2.3 Cross-platform

Usein sovelluskehittäjät julkaisevat sovelluksensa useammalle kuin yhdelle alustalle. Tätä odotetaan mm. yrityksiltä, jotka julkaisevat ja ylläpitävät omia sovelluksiaan, kuten Facebook tai Ilta-Sanomat. Kun samaa sovellusta tehdään usealle käyttöjärjestelmälle, on usein tehokkaampaa käyttää ns. cross-platform-ympäristöä sovellusten ohjelmointiin. Cross-platform-ympäristöt mahdollistavat sovelluskehityksen esim iOS- ja Android-laitteille samanaikaisesti. Cross-platform-ympäristö kääntää jaetun ohjelmakoodin kullekin alustalle sopivaksi, näin pystytään hyödyntämään sama koodi usean alustan sovelluksissa. Ilman jaettua ohjelmakoodia jouduttaisiin jokaiselle alustalle tekemään oma koodinsa ja oma sovelluksensa. (Patil 2016.)

Tunnettuja cross-platform-ympäristöjä ovat mm. Xamarin, Ionic, Appcelerator sekä NativeScript. Seuraavissa luvuissa nämä ympäristöt esitellään lyhyesti.

2.3.1 Xamarin

Xamarin on Mono-ohjelmointiympäristön pohjalta kehitetty mobiiliohjelmointiympäristö, jonka ohjelmointikieli on C#. Xamarinilla voidaan tehdä sovelluksia niin Androidille, iOS-järjestelmälle kuin Windows Phonelle. Xamarin toimii osana Microsoftin Visual Studiota ja mahdollistaa natiivien SDK-työkalujen käytön. Xamarin on suunniteltu niin, että suuri osa ohjelmistokoodista voidaan hyödyntää kaikilla tuetuilla alustoilla, ja vain käyttöliittymät täytyy tehdä alustakohtaisesti (Burns, Dunn, Johnson, Britch & Umbaugh 2017.)

2.3.2 Ionic

Ionic Framework on avoimeen lähdekoodiin perustuva ympäristö, joka käyttää HTML-, CSS- ja JavaScript-kieltä mobiilisovellusten ohjelmointiin. Ionic keskittyy käyttäjäläheiseen ohjelmointiin, eli pääpaino on käyttöliittymässä. Ionic pyrkii web-pohjaisella rakenteellaan mahdollistamaan sujuvan käytön millä tahansa laitteella, joka tukee standardoituja web-teknologioita. Myös visuaalisesti näyttävät ja yksinkertaiset sovellukset ovat Ionicin tavoitteena. Koska Ionic käyttää web-teknologioita, se pystyy käyttämään myös kunkin alustan omia tyylejä sovelluksissaan. Esimerkiksi iOS-laitteilla käytetään Applen iOS design language -tyyliä ja Android-laitteilla Googlen Material Design -tyyliä. Näin jokaisella alustalla sovellukset tuntuvat tutuilta kullakin käyttöjärjestelmällä. (Lucas & Wiegert 23.1.2019.)

2.3.3 Appcelerator

Appcelerator käyttää Titanium-ympäristöä, jossa sovelluskehitys tapahtuu. Appcelerator käyttää ohjelmointikielenään JavaScript-kieltä. Titanium-ympäristö koostuu useasta osiosta, joita ovat mm. Titanium SDK -työkalu, Appcelerator CLI, Hyperloop, Alloy sekä Titanium APIs. Titanium SDK yhdistää sovelluksen lähdekoodin, JavaScript-käännöksen sekä sovelluksen kiinteät elementit paketiksi, joka voidaan suorittaa puhelimella. Appcelerator CLI sisältää työkaluja mobiilisovellusten kehittämiseen sekä käyttöönottoon. Hyperloop mahdollistaa alustakohtaisten API-työkalujen vapaan käytön. Alloy avaa mahdollisuuden sovelluskehitykseen käyttäen Model-View-Controller (MVC) -arkkitehtuuria. Titanium API antaa ohjelmistokehittäjän käyttöön suuren määrän alustakohtaisia API-elementtejä. (Axway [Viitattu 6.4.2019].)

MVC-arkkitehtuuria käytetään lähdekoodin organisoimiseen ja selkeyttämään lähdekoodin osioita sekä niiden tehtäviä (Codecademy [Viitattu 6.4.2019]). Appceleratorilla luodut sovellukset tuntuvat tutuilta kaikilla alustoilla, sillä Appcelerator käyttää kunkin käyttöjärjestelmän omia API-elementtejä (Axway [Viitattu 6.4.2019]).

2.3.4 NativeScript

NativeScript perustuu Apache 2.0 -lisenssin avoimeen lähdekoodiin (NativeScript [Viitattu 28.4.2019]). Ohjelmointikieliä, joita NativeScript käyttää, ovat JavaScript, Angular sekä TypeScript. Näillä kielillä on mahdollista luoda natiiveja sovelluksia Android- ja iOS-laitteille. NativeScript antaa mahdollisuuden käyttää kunkin alustan natiiveja API-rajapintoja. (NativeScript, [Viitattu 29.4.2019].)

NativeScript tarjoaa verkkopohjaisen NativeScript Playground –palvelun, jossa voi tutustua NativeScriptiin ja sillä ohjelmointiin. Tarjolla on myös CLI-työkalut, joilla sovelluskehitys onnistuu paikallisesti omalla tietokoneella, ilman tarvetta verkkopalveluun. (NativeScript, [Viitattu 29.4.2019].) NativeScript CLI -työkalut ovat tarjolla Windows-, Mac- ja Linux-käyttöjärjestelmille (NativeScript [Viitattu 27.4.2019]).

3 XAMARIN

Xamarin on ohjelmistokehitysympäristö, joka mahdollistaa jaetun ohjelmakoodin hyödyntämisen mobiilisovelluksissa. Xamarinin ohjelmointikieli on C# (Burns, Dunn, Johnson, Britch & Umbaugh 2017).

Xamarin sallii alustakohtaisten SDK-työkalujen käytön C#-komennoilla. IOS-alustalla Xamarin.iOS antaa käyttöön Applen CocoaTouch SDK -työkalun, jolloin esimerkiksi UIKit-kirjastoa käyttämällä saadaan käyttöön kaikki iOS-käyttöliittymän hallintakomennot. Xamarin.Android mahdollistaa Androidin käyttöliittymäkomentojen käytön, ja Android.Views-kirjastolla voidaan hallita Androidin käyttöliittymää. (Burns, Dunn, Johnson, Britch & Umbaugh 2017.)

Xamarinin etuna on, että Xamarin.Android ja Xamarin.iOS tarjoavat yhtenäisen tavan kirjoittaa C#-koodia, jota voidaan käyttää kaikilla alustoilla. Sovelluksen toiminnallisuudet, kuten tietokantojen hallinnan ja verkonhallinnan, voi toteuttaa Xamarinilla niin, että komennot ja toiminnallisuudet kirjoitetaan vain kerran, ja kaikki alustat käyttävät tätä koodia. Käyttöliittymät on kuitenkin luotava erikseen jokaiselle alustalle. (Burns, Dunn, Johnson, Britch & Umbaugh 2017.)

Xamarin toimii osana Microsoftin Visual Studiota, sitä voidaan käyttää niin Mac- kuin PC-tietokoneilla. On kuitenkin muistettava, että iOS-alustalle koontiversion luonti täytyy tehdä Mac-tietokoneella ja Android- sekä Windows-alustoille koontiversiot on luotava Windows-tietokoneella. (Burns, Dunn, Johnson, Britch & Umbaugh 2017.)

3.1 Historiaa

Xamarin perustettiin vuonna 2011. Xamarinin perustajat olivat työskennelleet Mono-ohjelmointiympäristön parissa ennen Xamarinin perustamista (MindChemistry 2013). Mono-ohjelmointiympäristö on cross-platform-ympäristö, joka pohjautuu Microsoftin .NET-ympäristöön (Mono [Viitattu 30.4.2019]).

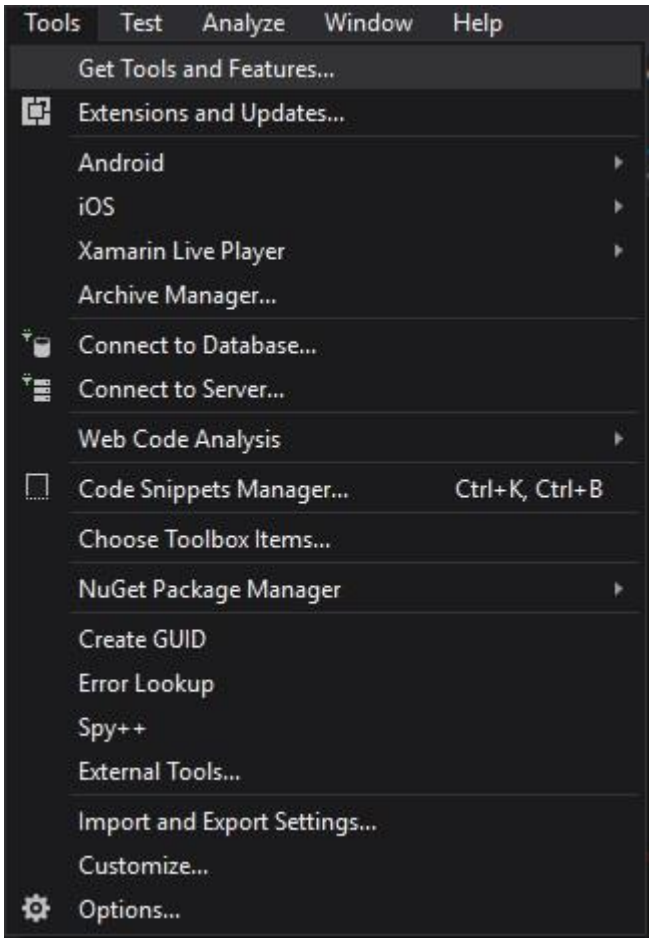
Aluksi Xamarinin ohjelmistot olivat vuosimaksullisia. Ohjelmistot oli jaettu neljään tilausvaihtoehtoon. Nämä vaihtoehdot olivat Starter, Indie, Business ja Enterprise.

Starter-versio oli ilmainen käyttää, mutta rajoituksena oli mm. sovelluksen koontiversion maksimikoko, kolmannen osapuolen API-rajapintojen käytön mahdottomuus eikä Starter-versiota voinut käyttää Visual Studiossa. Indie-versiossa poistuivat rajoitukset koontiversion maksimikoosta sekä kolmannen osapuolen API-rajapintojen käytöstä. Rajoitus Visual Studion käytöstä pysyi kuitenkin Indie-versiossakin. Indie-versiossa oli myös rajoitus yrityksen kokoon, mikäli yrityksessä oli yli 5 henkilöä, täytyi yrityksen valita Business- tai Enterprise-versio. Business- ja Enterprise-versioissa oli vähemmän rajoituksia kuin Indie-versiossa. Enterprise-versioon kuului myös tukipalvelu. (Smith 2014, 8,9.)

Vuonna 2016 Xamarin siirtyi Microsoftin alaisuuteen (Gutherie 2016). Samalla Xamarin liitettiin osaksi Visual Studiota. Tämän liitoksen johdosta Xamarin on ilmainen käyttää kaikille Visual Studion käyttäjille. Visual Studion Professional- ja Enterprise-lisenssit ovat maksullisia. Community-lisenssi on yksityiskäyttöön, avoimen lähdekoodin projekteihin, opetuskäyttöön sekä pienille yrityksille. Se on tämän työn kirjoitushetkellä ilmainen. (Friedman 2016.)

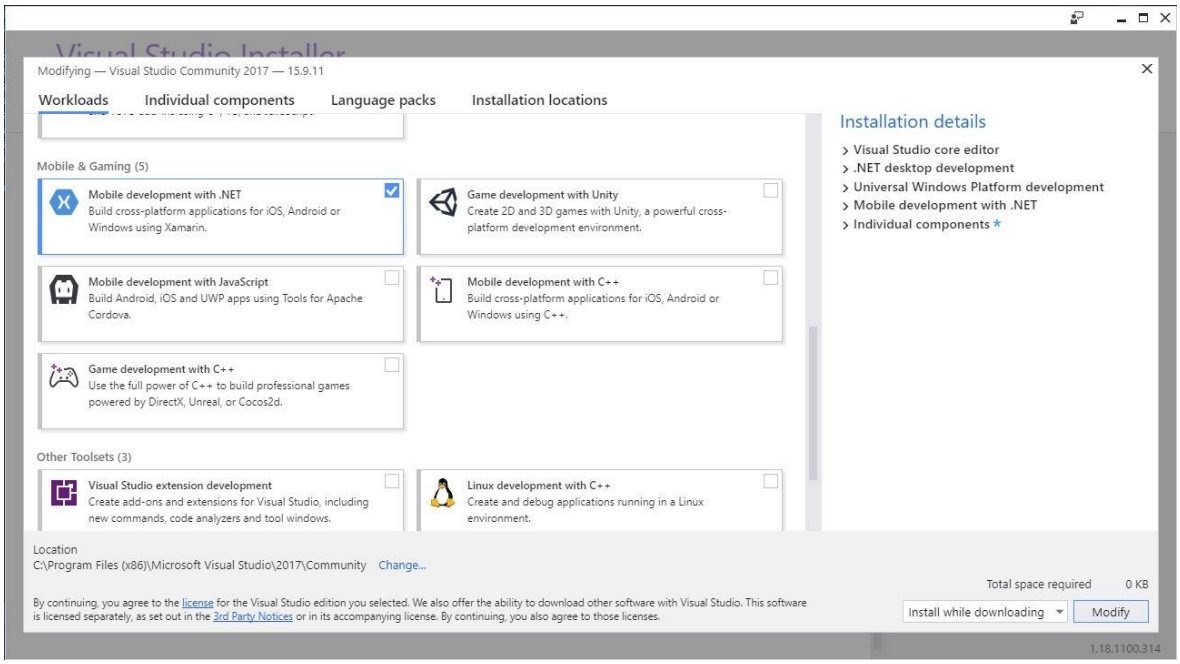
3.2 Asennus Visual Studioon

Xamarinin asennus osaksi Visual Studiota onnistuu helposti käyttäen Visual Studio Installer -ohjelmaa. Visual Studio Installer käynnistetään valitsemalla Visual Studiossa Tools → Get Tools and Features… (kuva 1).

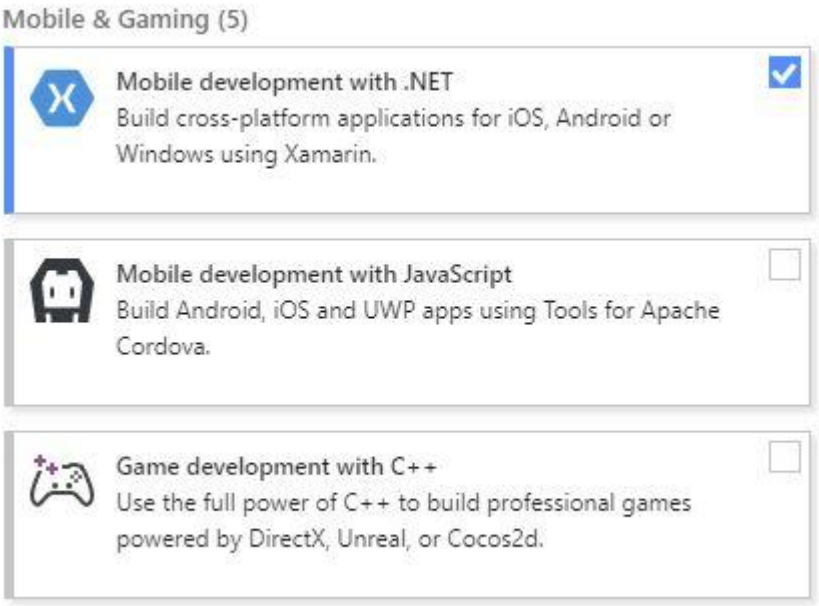


Kuva 1. Get Tools and Features

Visual Studio Installerissa (kuva 2) valitaan Workloads-välilehdeltä, Mobile & Gaming -kohdasta, Mobile development with .NET (kuva 3).

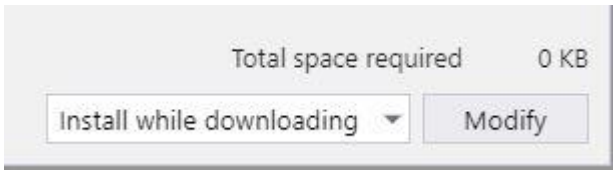


Kuva 2. Visual Studio Installer



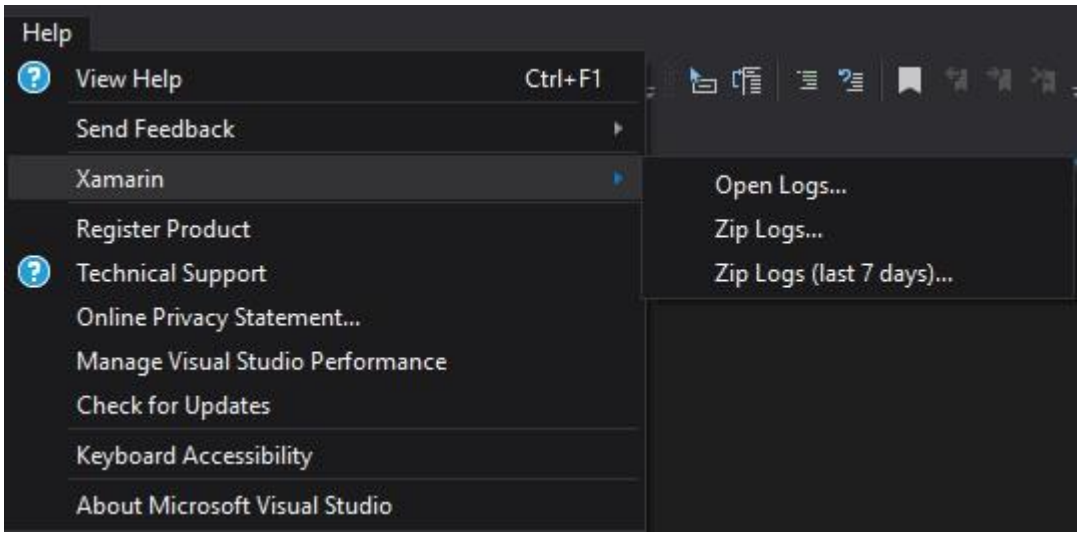
Kuva 3. Mobile development with .NET

Tämän jälkeen Visual Studio Installerin oikeassa alakulmassa näkyy Total space required -kohdassa, miten paljon tilaa Xamarinin asennus tarvitsee kiintolevyltä. Xamarinin asennus alkaa painettaessa Modify-painiketta (kuva 4). Kuvassa 4 Total space required näyttää 0 KB, sillä Xamarin oli jo asennettu osaksi Visual Studiota kuvaa otettaessa.



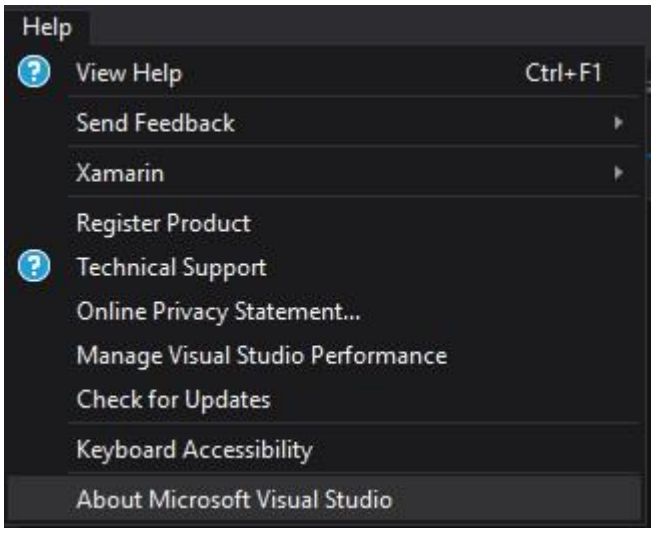
Kuva 4. Total space required

Onnistuneen asennuksen voi tarkistaa avaamalla Visual Studiossa Help-valikon, jossa tulisi nyt näkyä Xamarin yhtenä valikon kohdista (kuva 5).

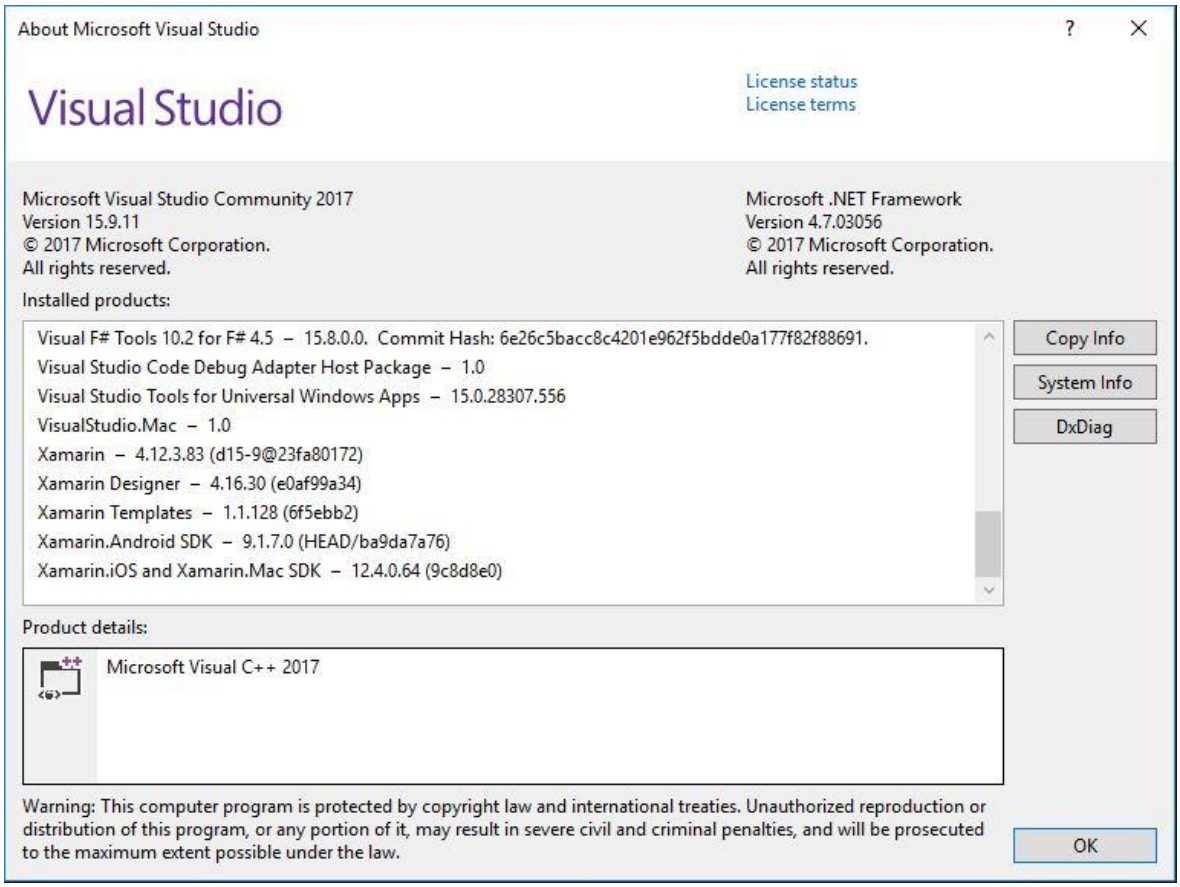


Kuva 5. Xamarin Help-valikossa

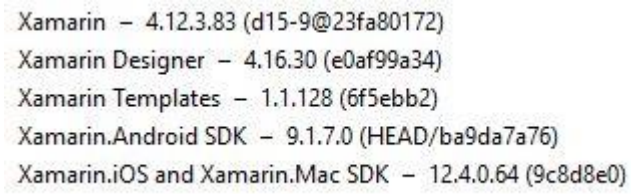
Onnistuneen asennuksen voi tarkistaa myös avaamalla Help-valikosta kohdan About Microsoft Visual Studio (kuva 6). Tämä avaa ikkunan (kuva 7), jossa on listattuna kaikki Visual Studioon asennetut osat. Listassa pitäisi nyt olla Xamarinin asennetut komponentit (kuva 8).



Kuva 6. Help-valikon About Microsoft Visual Studio



Kuva 7. About Microsoft Visual Studio



Kuva 8. Xamarinin asennetut komponentit

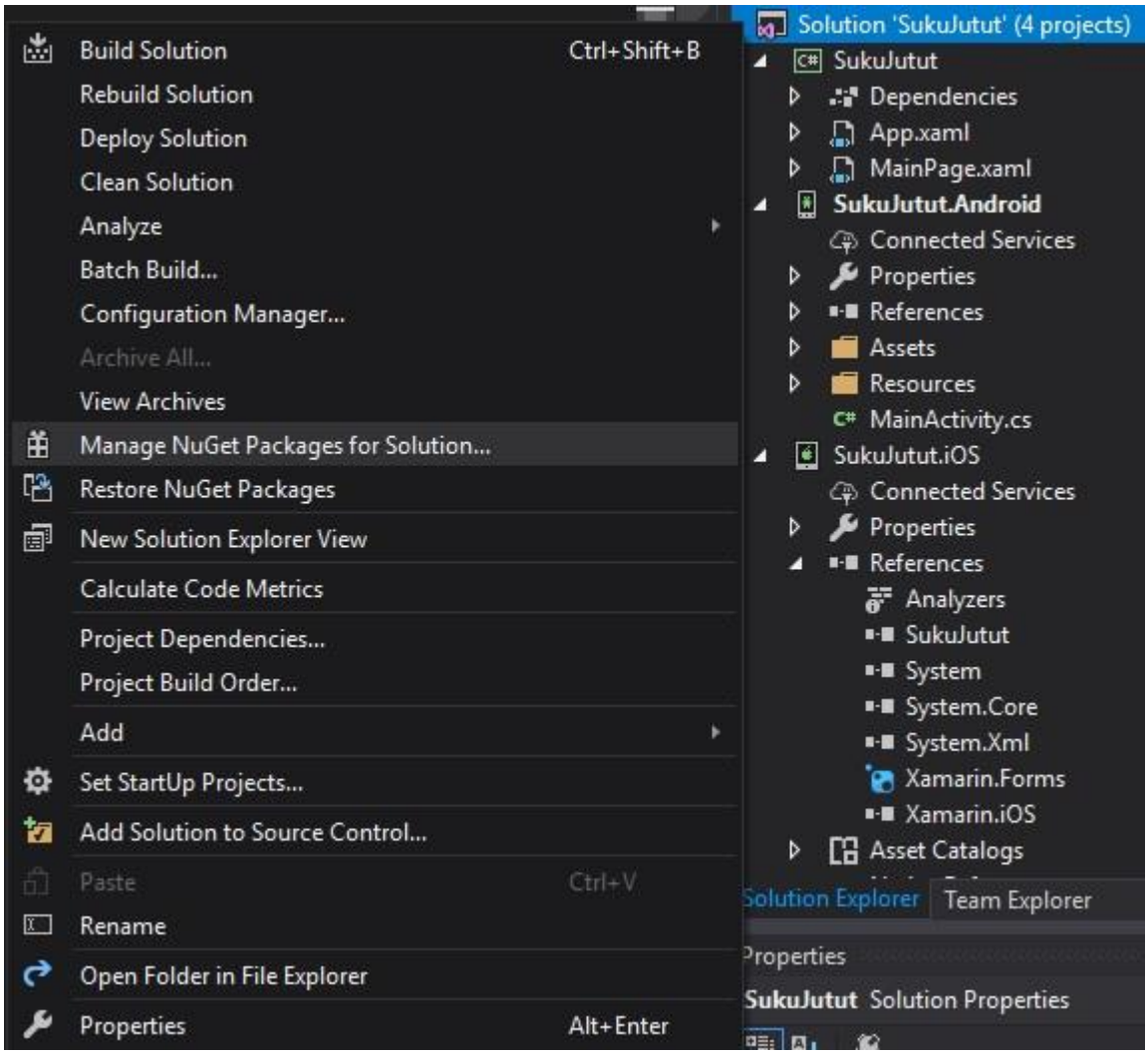
3.3 Xamarin.Essentials

Tammikuussa 2019 Microsoft julkaisi uuden API-kirjaston, Xamarin.Essentials, jonka avulla sovelluskehittäjät pääsevät käyttämään kunkin alustan natiiveja API-rajapintoja C#-ohjelmointikielellä. Xamarin.Essentialsin avulla eri alustojen natiiveja API-rajapintoja voidaan käyttää samalla koodilla, kuin ne olisivat samanlaiset jokaisella alustalla. Tämä mahdollistaa entistäkin laajemman lähdekoodin jakamisen alustojen välillä. Suurin ero API-kirjaston käytössä alustojen välillä on

käyttöoikeuksien hallinta. Jokaisella alustalla on omat vaatimuksensa käyttöoikeuksien käsittelyssä, mitkä sovelluksen on täytettävä. (Simone 2019.) Xamarin.Essentials sisältää yli 30 natiivia API-rajapintaa, joilla voidaan käyttää mm. paikannusta, sähköpostia, kiihtyvyysanturia, salamavaloa, tärinämoottoria ja monia muita (Montemagno, Dunn, Malcolm, Petzold & Umbaugh 2019).

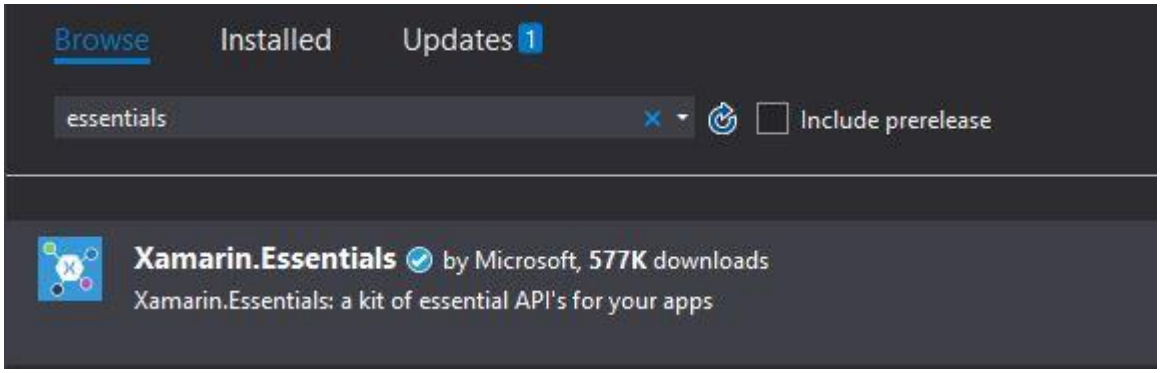
Laajasta kirjastosta huolimatta, Xamarin.Essentials ei tuo sovellukseen ylimääräistä sisältöä, joka kasvattaisi sovelluksen kokoa, sillä sovelluksen koontiversiota luodessa Xamarin poistaa käyttämättömät API-työkalut, jättäen vain sovelluksessa käytetyt työkalut (Montemagno 2018).

Xamarin.Essentialsin liittäminen osaksi sovellusta onnistuu helposti Visual Studion NuGet-paketinhallintasovelluksella. Projektin ollessa auki valitaan Visual Studiossa, Solution Explorerissa, projektin nimi ja napautetaan tätä hiiren oikealla painikkeella. Tästä valitaan Manage NuGet Packages for Solution (kuva 9).



Kuva 9. Manage NuGet Packages

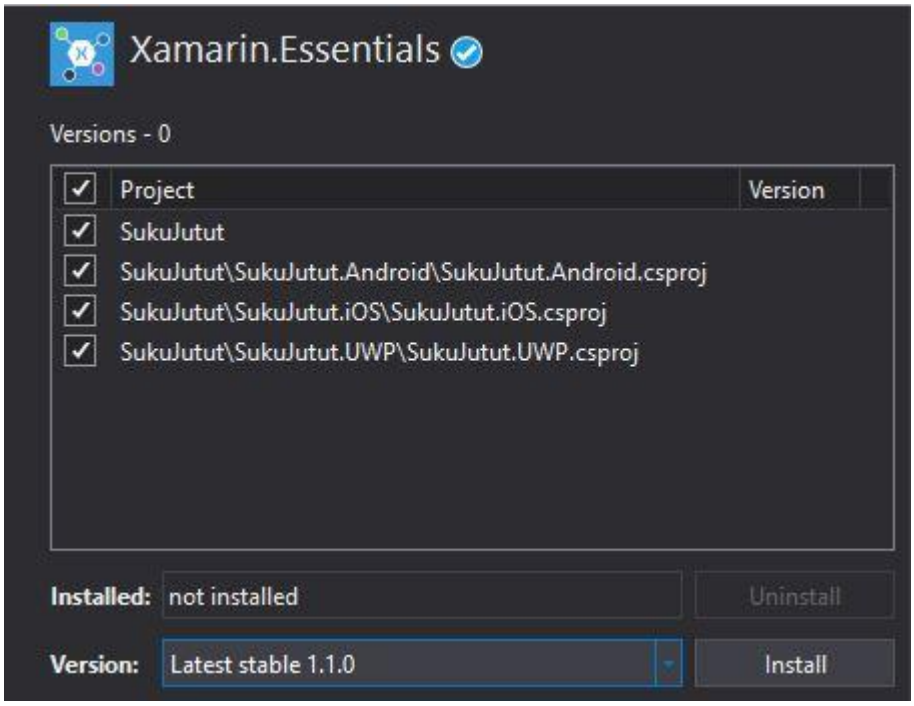
Tämän jälkeen aukeaa NuGet-paketinhallintaikkuna, jonka hakukentän avulla voidaan etsiä Xamarin.Essentials-paketti (kuva 10).



Kuva 10. Xamarin.Essentials

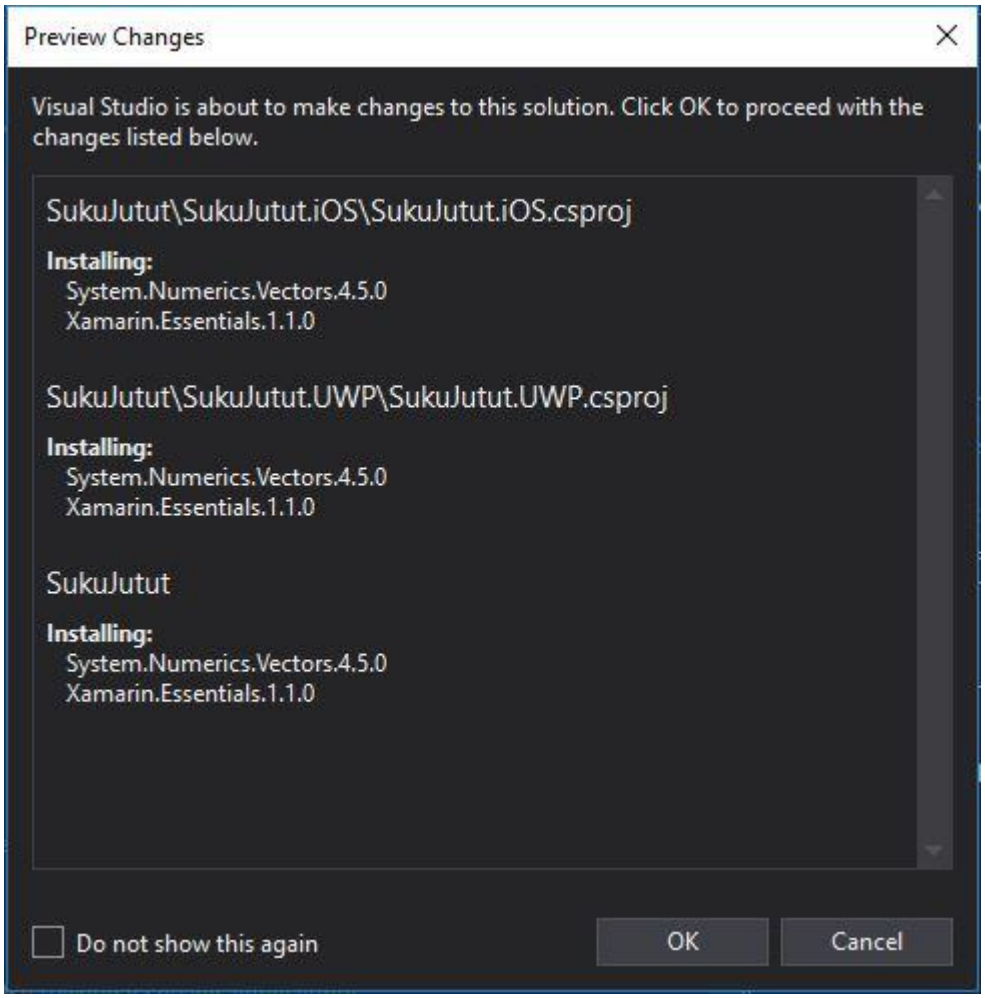
Cross-Platform-sovellusta luodessa valitaan kaikki alustat (kuva 11). Mikäli sovellusta luodaan vain yhdelle alustalle, ei valintaa tarvitse tehdä. Asennettavan

version Xamarin.Essentailsista voi myös määrittää tässä kohdassa, on kuitenkin suositeltavaa käyttää viimeisintä vakaata versiota.



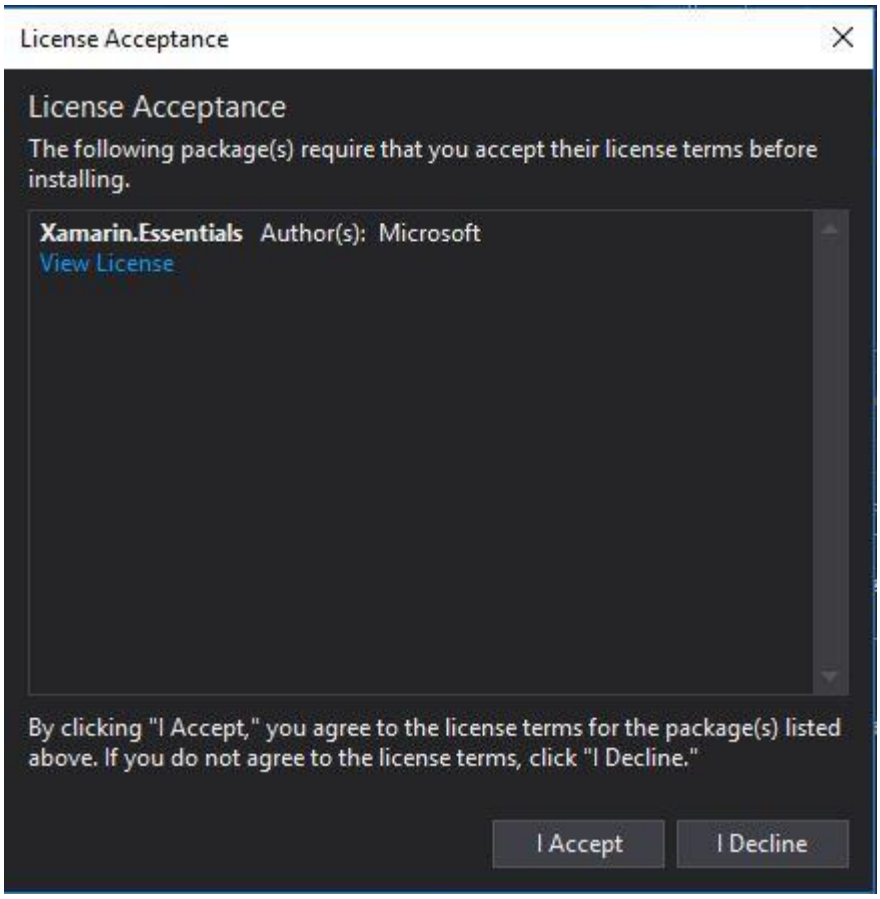
Kuva 11. Xamarin.Essentials-asennus

Kun asennuksen aloittaa Install-painikkeesta, Visual Studio kysyy jatketaanko asennusta (kuva 12). Asennusta jatketaan OK-painikkeella.



Kuva 12. Xamarin.Essentials-asennuksen vahvistus

Jotta asennus voidaan viedä päätökseen, täytyy lisenssisopimus lukea sekä hyväksyä (kuva 13).



Kuva 13. Xamarin.Essentials-lisenssisopimus

Kun lisenssisopimus on hyväksytty, asentaa Visual Studio Xamarin.Essentialsin osaksi projektia.

3.4 Xamarin yrityksen käytössä

Atk-Palvelu Luhtasaari Oy kiinnostui Xamarinista, kun Xamarin siirtyi Microsoftin alaisuuteen. Mahdollisuus maksuttomaan sovelluskehitykseen, jonka takana on suuri ja tunnettu ohjelmistoyritys, sekä ohjelmointikieli, joka on lähellä yrityksessä käytettyä ohjelmointikieltä, houkuttelivat tutustumaan tarkemmin tähän vaihtoehtoon.

Yritys olisi halukas toteuttamaan PC-sovelluksensa rinnalle mobiilisovelluksen, jolla olisi mahdollista lukea PC-versiolla luotuja tiedostoja.

4 SUKUJUTUT JA GEDCOM

4.1 SukuJutut

SukuJutut on Atk-Palvelu Luhtasaari Oy:n luoma sukututkimusohjelma, joka toimii kaikilla Windows-versioilla XP-versiosta uudempiin. SukuJutut-ohjelmaa voi myös käyttää Mac-tietokoneilla Crossover-ohjelman avulla sekä Linuxissa esim. Wine-emulaattoria käyttäen. SukuJutut on kehitetty Visual Basic -ohjelmointikielellä. Ohjelmaa on kehitetty vuodesta 1993. (SukuJutut [Viitattu 29.4.2019].)

SukuJutut on helppokäyttöinen ohjelma, joka sopii niin harrastelijoille kuin ammattilaisillekin. Ohjelmassa on monipuoliset mahdollisuudet tulostuksiin ja henkilöhakuihin. Ohjelmassa on myös työkalu sukukirjojen tekoon sekä monia esipolvikaavioita, joita ovat mm. ympyränmuotoinen esivanhempoien taulu sekä html-esipolvikaavio perhekaavoineen, jotka voidaan tulostaa myös RTF-muodossa. SukuJutut ohjelmassa on yhdeksän eri kielivaihtoehtoa. (SukuJutut [Viitattu 28.4.2019].)

Suku- ja henkilötietojen siirtäminen toisesta sukututkimussovelluksesta onnistuu Gedcom-standardin avulla. Gedcomin avulla voidaan myös siirtää aineistoja sukututkijalta toiselle. (SukuJutut [Viitattu 28.4.2019].)

4.2 Gedcom

Gedcom (GEnealogical Data COMmunications) on sukuselvitysohjelmien käyttämä standardi aineistojen siirtoon. Gedcom-standardin on luonut Myöhempien Aikojen Pyhien Jeesuksen Kristuksen kirkko. (Gedcom 1999.)

Sukuselvitysohjelmalla voidaan luoda Gedcomin avulla .ged-tiedosto ohjelmaan kerätystä aineistosta. Gedcom-tiedosto on tekstitiedosto, joka sisältää henkilöiden tiedot ja sukulaisuussuhteet. Gedcom-tiedostossa data on järjestetty hierarkiaan niin, että jokaisella rivillä on hierarkiaa vastaava numero, viite tiedosta sekä valinnainen arvo, joka on esimerkiksi henkilön nimi. (Gedcom 1993.)

Viite Gedcom-tiedostossa kertoo, mitä tyyppiä rivin tieto on. Hierarkianumeron ansiosta rivillä voi olla alirivejä. Rivi ja sen alirivit muodostavat kokonaisuuden, joka käsittelee rivin viitteellä määriteltyä asiaa. Rivit ja alirivit muodostavat tietueita. Uuden tietueen alkua merkitään hierarkianumerolla 0 (nolla). (Gedcom 1993.)

5 SOVELLUKSEN TOTEUTUS XAMARINILLA

5.1 Tavoite ja tarkoitus

Atk-Palvelu Luhtasaari Oy on kiinnostunut tuomaan asiakkailleen mobiiliversion SukuJutut-ohjelmastaan. Mobiiliversiolla olisi tarkoitus lukea SukuJutut-tietokoneohjelmalla luotuja Gedcom-tiedostoja. Sovelluksella ei siis olisi mahdollista muokata aineistoja, ainakaan tässä vaiheessa. Tämä siksi, koska SukuJutut-ohjelman tietokanta on tallennettu ohjelman kansioihin paikallisesti. Mikäli samaa aineistoa muokkaa useassa laitteessa, hankaloituu aineistojen yhdistäminen takaisin yhtenäiseksi aineistoksi.

SukuJutut-mobiilisovelluksen keskeisiä tavoitteita on lukea Gedcom-tiedosto mobiililaitteen muistista ja järjestää tiedoston sisältö sovelluksen näkymään tarkasteltavaksi. Gedcom-tiedoston valinta tai haku laitteen tallennusmediasta helpottaisi myös sovelluksen käyttöä. Laitteen käyttöliittymään luodaan kentät, jotka täytetään henkilötiedoilla, jotka Gedcom-tiedostosta luetaan.

SukuJutut-mobiiliversion prototyyppi tehdään Android-alustalle, sillä niin yrityksellä kuin työn tekijälläkin on mahdollisuus testata sovellusta Android-laitteella. IOS-alustalle sovellusta ei vielä toteuteta, koska iOS-alustalle sovellusta luodessa testauksen tulisi tapahtua Mac-tietokoneella, jota ei ollut saatavilla sovellusta luodessa.

5.2 Toteutus

SukuJutut-mobiilisovelluksen toteutus Android-alustalle aloitettiin suunnittelemalla sovelluksen rakennetta. Sovelluksen olisi tarkoitus päästä lukemaan puhelimen tallennusmediasta tekstitiedosto ja näyttää tekstitiedoston sisältö sovelluksen näkymässä. Sovelluskehitys aloitettiin luomalla Visual Studiolla Xamarin.Android-projekti ja selvittämällä, miten Android-alustalla tiedostojen luku onnistuu.

Android on käyttöjärjestelmänä toteutettu niin, että tallennusmedia on jaettu sisäiseen (internal) ja ulkoiseen (external) tallennusmediaan. Sisäinen

tallennusmedia tarkoittaa sovelluksen omaa osiota tallennusmediasta, jonne muilla sovelluksilla tai käyttäjillä ei ole pääsyä. Ulkoinen tallennusmedia taas tarkoittaa kaikkea muuta, jaettua, tallennusmediaa. Ulkoista tallennusmediaa voi siis olla puhelimen sisäinen tallennusmedia tai SD-kortti. (Developers [Viitattu 29.4.2019].)

Tämä ulkoisen ja sisäisen tallennusmedian jako ihmetytti mobiiliohjelmointiin tutustuvaa sovelluskehittäjää, etenkin ulkoisen median liittyminen sekä laitteen sisäänrakennettuun tallennusmediaan että SD-korttiin.

Ensimmäisessä testivaiheessa sovellukseen oli määritelty kiinteä osoite tekstitiedoston polulle. Tämä tarkoittaa sitä, että tekstitiedoston täytyi olla juuri oikeassa paikassa puhelimen tallennusmediassa, jotta sovellus löytää tekstitiedoston. Tämä ei tietenkään voi jäädä lopulliseen sovellukseen, sillä se rajoittaa liikaa tiedostosijaintia ja loppukäyttäjälle tämä hankaloittaisi sovelluksen käyttöä.

MainActivity.cs-tiedosto sisältää sovelluksen toiminnalliset osat. Tämä tiedosto sisältää OnCreate()-luokan, joka suoritetaan sovelluksen käynnistyessä. Sovelluksen testivaiheessa toiminnallisuudet luotiin suoraan MainActivity.cs-tiedoston OnCreate()-luokan sisään. Vaihtoehtoisesti OnCreate()-luokkaan olisi voitu määritellä kutsu, joka olisi suorittanut erillisen luokan, joka sisältää sovelluksen toiminnallisen koodin.

Jotta sovellus pääsisi lukemaan laitteen tallennusmediaan, tuli tälle määritellä AndroidManifest.xml-tiedostoon oikeus lukea ulkoista tallennusmediaa. Sovelluksen näkymään lisättiin TextView-elementti, johon tekstitiedoston sisältö näytetään. TextView-elementtiin viitattiin OnCreate()-luokassa elementin id-arvolla, jotta sovellus tunnistaisi oikean elementin, jota käyttää.

Tämän jälkeen luotiin tarkistus, joka varmistaa, että sovelluksella on oikeus lukea laitteen tallennusmediaa. Tarkistus luotiin käyttämällä ContextCompat.CheckSelfPermission()-komentoa. Mikäli sovelluksella ei ollut tarvittavia oikeuksia, ilmoitti se puuttuvista oikeuksista. Jos oikeudet tallennusmediaan olivat kunnossa, sovellus haki tekstitiedoston laitteen tallennusmediasta, luki tiedoston sisällön ja näytti sen TextView-elementissä.

Sovelluksen testaaminen tapahtui käyttämällä Android-puhelinta. Vaihtoehtoisesti testauksen olisi voinut suorittaa Android-emulaattorilla, mutta tiedoston käsittelystä johtuen, tekstitiedosto oli yksinkertaisempaa siirtää puhelimen tallennusmediaan kuin emulaattoriin.

Ensimmäinen testi puhelimella antoi virheilmoituksen puuttuvista oikeuksista. Jotta sovellus pääsisi tallennusmediaan lukemaan tiedoston, tuli puhelimen sovellusasetuksista käydä sallimassa tälle sovellukselle tallennusmedian lukuoikeudet. Kun oikeudet oli puhelimessa sallittu, testattiin sovellusta uudestaan, ja sovellus luki tekstitiedoston sekä näytti tiedoston sisällön näkymän TextView-elementissä.

Käytettävyyden parantamiseksi sovellukseen tulisi lisätä liitännäinen, joka kysyy tarvittavat oikeudet, jollei niitä ole. Tämä helpottaisi loppukäyttäjän kokemusta, sillä oikeuksien määrittäminen laitteen sovellusasetuksista hankaloittaa sovelluksen käyttöönottoa.

Ennen oikeuksia kysyvää liitännäistä, työssä perehdyttiin, miten käyttäjä voisi itse määritellä tiedoston sijainnin. Monia vaihtoehtoja kokeiltiin, joista yksi onnistui valitsemaan kansion muttei tiedostoa. Löydettiin kuitenkin Xamariniin FilePicker-liitännäinen, jonka avulla tiedoston valitseminen tallennusmediasta onnistuu. Kun liitännäinen asennettiin NuGet-paketinhallintasovelluksen kautta, ja koodiin lisättiin liitännäisen komennot, antoi Visual Studio kuitenkin virheilmoituksen koodista. Virheilmoitusta yritettiin korjata niin Visual Studion ehdotusten kuin Xamarinin keskustelupalstojenkin avulla, mutta mikään ei korjannut ongelmaa, vaan lähinnä syntyi uusia vikoja.

Yhdessä yrityksen edustajan kanssa päätettiin, että mikäli sovellusta ei saataisi valmiiksi, voisi tämä työ toimia pohjana jatkokehitykselle. Sovellus ei tämän työn puitteissa valmistunut, vaan jatkokehitystä tarvitaan.

6 YHTEENVETO

Xamarin on monipuolinen ja hyvä cross-platform-sovelluskehitysympäristö, joka mahdollistaa laajan lähdekoodin jakamisen eri alustojen välillä. Etenkin Xamarin.Essentials tuo Xamariniin paljon kirjastoja laitteen ominasuuksien, kuten paikannuksen, hallintaan.

Xamarin ei kuitenkaan ole kovin aloittelijaystävällinen ympäristö. Kokemusta mobiiliympäristöjen ohjelmointiin on hyvä olla, jotta Xamariniin tutustuminen helpottuu.

Uskon että suurimmat ongelmat sovelluksen kanssa johtuivat siitä, ettei mobiilialustojen sovelluskehityksestä ollut kovin paljon aiempaa kokemusta.

Atk-Palvelu Luhtasaari Oy:n käyttöön Xamarin olisi sopiva, sillä lähdekoodin jakaminen yleisimpien mobiilialustojen välillä helpottaa sovelluskehitystä ja nopeuttaa sovelluskehitystä eri alustoille.

LÄHTEET

Apple. 8.12.2016. Jump Right In. [Verkkosivu]. Apple Inc. [Viitattu 6.4.2019]. Saatavissa: https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift

Apple. Ei päiväystä. Submit your apps today. [Verkkosivu]. Apple Inc. [Viitattu 5.4.2019]. Saatavissa: https://developer.apple.com/app-store/submissions

Axway. Ei päiväystä. Titanium Platform Overview. [Verkkosivu]. Axway. [Viitattu 6.4.2019]. Saatavissa: https://docs.axway.com/bundle/Titanium_SDK_allOS_en/page/titanium_platform_overview.html

Burd, B. 2016. Java Programming for Android Developers for Dummies. [Verkkokirja]. Hoboken: John Wiley & Sons, Incorporated. [Viitattu 6.4.2019]. Saatavana ProQuest Ebook Central –palvelusta. Vaatii käyttöoikeuden.

Burns, A., Dunn, C., Johnson, J., Britch, D. & Umbaugh, B. 2017. Part 1 – Understanding the Xamarin Mobile Platform. [Verkkosivu]. Microsoft Corporation. [Viitattu 29.4.2019]. Saatavissa: https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform

Codecademy. Ei päiväystä. MVC: Model, View, Controller. [Verkkosivu]. Codecademy. [Viitattu 6.4.2019]. Saatavissa: https://www.codecademy.com/articles/mvc

Developers. Ei päiväystä. Launch checklist. [Verkkosivu]. Google developers. [Viitattu 5.4.2019]. Saatavissa: https://developer.android.com/distribute/best-practices/launch/launch-checklist.html

Developers. Ei päiväystä. Publish your app. [Verkkosivu]. Google Developers. [Viitattu 6.4.2019]. Saatavissa: https://developer.android.com/studio/publish

Developers. Ei päiväystä. Data and file storage overview. [Verkkosivu]. Google Developers. [Viitattu 29.4.2019]. Saatavissa: https://developer.android.com/guide/topics/data/data-storage

Feiler, J. 2014. IOS App Deevelopment for Dummies. [Verkkokirja]. Hoboken: John Wiley & Sons, Incorporated. [Viitattu 6.4.2019]. Saatavana ProQuest Ebook Central –palvelusta. Vaatii käyttöoikeuden.

Friedman, N. 2016. Xamarin for Everyone. [Verkkosivu]. Microsoft Corporation. [Viitattu 9.4.2019]. Saatavissa: https://devblogs.microsoft.com/xamarin/xamarin-for-all/

Gedcom. 1993. THE GEDCOM STANDARD. [www-dokumentti]. Family History Department, The Church of Jesus Christ of Latter-day Saints. [Viitattu 29.4.2019]. Saatavissa: https://chronoplexsoftware.com/gedcomvalidator/gedcom/gedcom-5.3.pdf

Gedcom. 1999. THE GEDCOM STANDARD. [www-dokumentti]. Family History Department, The Church of Jesus Christ of Latter-day Saints. [Viitattu 29.4.2019]. Saatavissa: http://phpgedview.sourceforge.net/ged551-5.pdf

Guthrie, S. 2016. Microsoft to acquire Xamarin and empower more developers to build apps on any device. [Verkkosivu]. Microsoft Corporation. [Viitattu 9.4.2019]. Saatavissa: https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/

IDC. Ei päiväystä. Smartphone Market Share. [Verkkosivu]. IDC Corporate USA. [Viitattu 6.4.2019]. Saatavissa: https://www.idc.com/promo/smartphone-market-share/os

Kremenek, T. 25.3.2019. Swift 5 Released!. [Verkkosivu]. Swift. [Viitattu 6.4.2019]. Saatavissa: https://swift.org/blog/swift-5-released

Kurniawan, B. & Deck, P. 2015. Android application development: a beginner's tutorial. [Verkkokirja]. Brossard: Brainy Software. [Viitattu 6.4.2019]. Saatavana ProQuest Ebook Central –palvelusta. Vaatii käyttöoikeuden.

Lucas, E. & Wiegert, C. 23.1.2019. What is Ionic Framework?. [Verkkosivu]. Ionic. [Viitattu 6.4.2019]. Saatavissa: https://ionicframework.com/docs/intro

MindChemistry. 2013. Xamarin. [Verkkosivu]. Mind Chemistry Inc. [Viitattu 9.4.2019]. Saatavissa: http://www.mindchemistry.com/Library/DocumentLibrary/SoftwareDevelopers/TechnologiesForMindChemistryProducts/Xamarin#.XMcuy2gzaUl

Mono. Ei päiväystä. About Mono. [Verkkosivu]. Mono. [Viitattu 30.4.2019]. Saatavissa: https://www.mono-project.com/docs/about-mono/

Montemagno, J. 2018. Accessing Native Features the Cross-Platform Way with Xamarin.Essentials. [Verkkosivu]. Microsoft Corporation. [Viitattu 29.4.2019]. Saatavissa: https://devblogs.microsoft.com/xamarin/accessing-native-features-xamarin-essentials/

Montemagno, J., Dunn, C., Malcolm, N., Petzold, C. & Umbaugh, B. 2019. Xamarin.Essentials. [Verkkosivu]. Microsoft Corporation. [Viitattu 29.4.2019]. Saatavissa: https://docs.microsoft.com/fi-fi/xamarin/essentials/index?WT.mc_id=docs-xamarinblog-jamont

NativeScript. Ei päiväystä. System Requirements. [Verkkosivu]. NativeScript. [Viitattu 27.4.2019]. Saatavissa: https://docs.nativescript.org/start/general-requirements

NativeScript. Ei päiväystä. How is NativeScript Licensed?. [Verkkosivu]. NativeScript. [Viitattu 28.4.2019]. Saatavissa: https://www.nativescript.org/faq/how-is-nativescript-licensed

NativeScript. Ei päiväystä. Getting Started With NativeScript. [Verkkosivu]. NativeScript. [Viitattu 29.4.2019]. Saatavissa: https://docs.nativescript.org/start/introduction

Patil, R. 2016. Pros and Cons of Cross-Platform Mobile App Development. [Verkkosivu]. InfoQ. [Viitattu 29.4.2019]. Saatavissa: https://www.infoq.com/articles/mobile-cross-platform-app-development

Simone, S. 2019. Xamarin.Essentials Streamlines Cross-Platform Access to Native iOS, Android, and UWP Features. [Verkkosivu]. InfoQ. [Viitattu 29.4.2019]. Saatavissa: https://www.infoq.com/news/2019/01/xamarin-essentials-native-apis

Smith, W. 2014. Learning Xamarin Studio. [Verkkokirja]. Birmingham: Packt Publishing. [Viitattu 29.4.2019]. Saatavanva EBSCOhost-palvelusta. Vaatii käyttöoikeuden.

Statcounter. 2019. Mobile Operating System Market Share Worldwide. [Verkkosivu]. Statcounter. [Viitattu 6.4.2019]. Saatavissa: http://gs.statcounter.com/os-market-share/mobile/worldwide

SukuJutut. Ei päiväystä. SukuJutut. [Verkkosivu]. Atk-Palvelu Luhtasaari Oy. [Viitattu 28.4.2019]. Saatavissa: https://sukujutut.fi/sukujut/esitteet/esite2.htm

SukuJutut. Ei päiväystä. SukuJutut. [Verkkosivu]. Atk-Palvelu Luhtasaari Oy. [Viitattu 29.4.2019]. Saatavissa: https://sukujutut.fi/sukujut/index.htm