OULUN AMMATTIKORKEAKOULU

Pyry Hofslagare

ALUSTARIIPPUMATTOMAN MOBIILISOVELLUKSEN KEHITYS

ALUSTARIIPPUMATTOMAN MOBIILISOVELLUKSEN KEHITYS

Pyry Hofslagare
Opinnäytetyö
Syksy 2022
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Pyry Hofslagare
Opinnäytetyön nimi: Alustariippumattoman mobiilisovelluksen kehitys
Työn ohjaaja: Pekka Ojala
Työn valmistumislukukausi ja -vuosi: Syksy 2022 Sivumäärä: 31

Opinnäytetyön aiheena oli alustariippumattoman mobiilisovelluksen kehitys. Toimeksiantajalle tehdyn työn tavoitteena oli kehittää iOS-käyttöjärjestelmässä toimiva mobiilisovelluksen demo. Toimeksiantajana toimi oululainen ohjelmistoyritys, missä suoritin ammattiharjoittelun. Opinnäytetyön alussa esiteltiin myös muita teknologioita kuin kehitystyössä käytettyä alustariippumatonta React Nativea. Tutkimusmenetelmänä toimi toiminnallinen tutkimus ja tavoitteena oli synnyttää toiminnallinen tuotos. Tietoperustana opinnäytetyössä oli harjoittelujakson aikana saatu tieto, teknologioitten dokumentaatiot, artikkelit ja luotettavat Internet-lähteet. Lähteinä opinnäytetyössä olivat omat kokemukset kehitystyöstä, verkkojulkaisut ja teknologioiden dokumentaatiot. Tiedonhaussa menetelmänä käytettiin pääsääntöisesti Googlen hakukonetta ja IT-alalla työskenteleviltä saatuja kommentteja sekä ajatuksia aiheesta.

Mobiilisovelluksesta saatiin demoversio valmiiksi ja sitä voidaan käyttää varsinaisen mobiilisovelluksen pohjana. Käyttöliittymän demoversiota on mahdollista myös hyödyntää aivan erilaisen sovelluksen luomiseen kuin mihin se on alun perin tarkoitettu ja tämä nopeuttaa huomattavasti kehittämistä, jos samankaltaisia sovelluksia on suunnitteilla luoda useampia. Opinnäytetyötä voidaan hyödyntää teknologian ja työvälineiden valinnassa.

Asiasanat: Android, iOS, mobiilisovellus, React Native, käyttöliittymä, sovellustestaus, ohjelmointiympäristö

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Pyry Hofslagare
Title of thesis: Developing cross-platform mobile application
Supervisor: Pekka Ojala
Term and year when the thesis was submitted: Autumn 2022
Number of pages: 31

Objective of this thesis was to create user interface for iOS mobile application using React Native technology. Thesis introduces thesis goals, used technology, other technologies, testing the software and final thoughts. The sources in the thesis are my own experiences in development work, online publications, and documentation of technologies. The demo version of the mobile application is ready and can be used as the basis for the actual mobile application.

Keywords: user interface, Android, iOS, mobile application, React Native, integrated development environment

SISÄLLYS

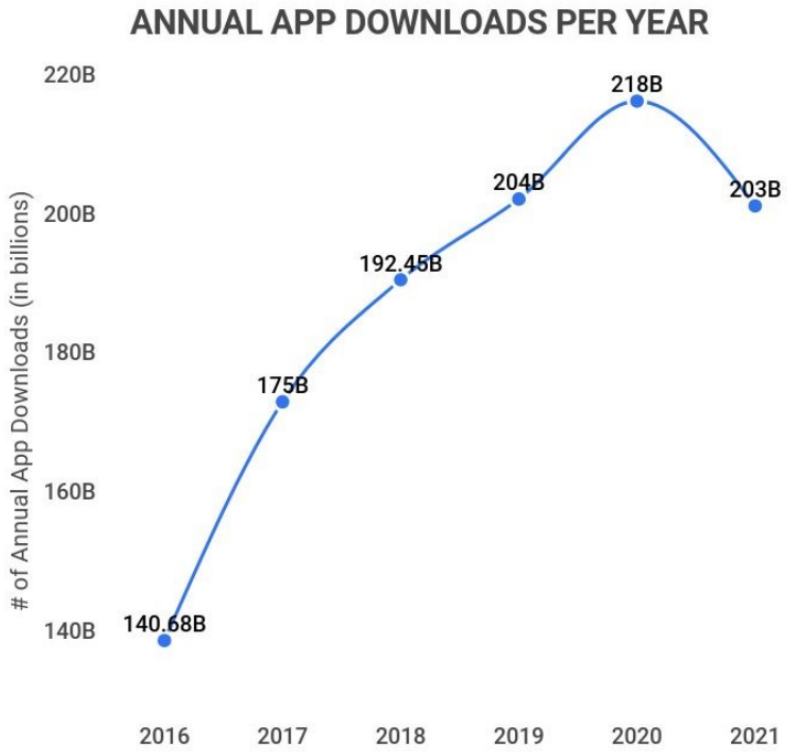
KÄSITTEET

Android	Googlen ylläpitämä mobiililaitteille suunnattu käyttöjärjestelmä
Android Studio	Android-käyttöjärjestelmän virallinen ohjelmointiympäristö kaikille Android-pohjaisille laitteille
Demo	Sovelluksen kehitysversio, mikä toimii visuaalisena esimerkkinä siitä, mitä sovelluksella on tarkoitus tehdä
Front-end	Käyttäjälle näkyvä puoli ohjelmistokehityksestä, esimerkiksi graafinen käyttöliittymä
iOS	Applen mobiililaitteille suunnattu käyttöjärjestelmä
iPhone	iOS-käyttöjärjestelmää käyttävä älypuhelin
macOS	Apple-tietokoneiden käyttöjärjestelmä, esimerkiksi Apple MacBook
Mobiililaite	Moderni pienikokoinen omaava älylaite, kulkee mukana
Simulaattori	Simulaattorilla voidaan simuloida älylaitteita tietokoneella
Solution	Säiliö, jota Visual Studio käyttää projektin järjestämiseen
UI-design	Käyttöliittymäsuunnittelu
UX-design	Käyttäjäkokemussuunnittelu
Unix	Bell Labsin kehittämä käyttöjärjestelmä. Unix-käyttöjärjestelmään perustuu moni moderni käyttöjärjestelmä, kuten macOS
Xcode	Ohjelmointiympäristö iOS- ja macOS-kehitykseen

1 JOHDANTO

Mobiilisovellukset ovat nousseet suosioon digitaalisen siirtymän myötä. Yritykset digitalisoivat prosessejaan kiihtyvään tahtiin (Talentree 2022), ja tämä mahdollistaa yrityksille tiedon siirtymisen nopeasti. Mobiililaitteet ovat ohittaneet internetin sisältöjen selauksessa perinteiset tietokoneet jo vuonna 2016 (MTV Uutiset 2016).

Logistiikkayritykset tarvitsevat mobiilisovelluksia sujuvaan paketin sijainnin ja tilanteen välittämiseen asiakkaalle. Trendi näkyy lähes kaikissa nykypäivän palveluissa; jopa uudenlaista vaihtoehtoa perinteiselle fyysiselle henkilöllisyystodistukselle on kaavailtu mobiilisovelluksen muodossa ja se on tarkoitus ottaa testikäyttöön syksyllä 2013 (Valtiovarainministeriö 2022). Kuviot 1 ja 2 osoittavat, että 2020-luvulla sovelluksia on ladattu jopa 200 miljardia kertaa, ja sovelluskauppojen vuositulot ovat nousseet jo yli 100 miljardin Yhdysvaltain dollarin summiin.



KUVIO 1. Vuosittaiset mobiilisovellusten latausmäärät (Zippia 2022)

GLOBAL APP REVENUE 2016-2021

Year	Total App Revenue (in billions USD)	iOS App Revenue	Google Play App Revenue
2016	$43.50	$28.60	$15
2017	$58.10	$38.50	$21.20
2018	$71.30	$46.60	$24.80
2019	$89	$58.40	$30.60
2020	$111	$72.30	$38.60
2021	$133	$85.10	$47.90

- **By the end of 2022, consumer app spending is expected to reach $170 billion.**

KUVIO 2. Vuosittaiset käyttäjien kuluttamat rahasummat mobiilisovelluksiin Yhdysvaltain dollareissa (Zippia 2022)

Merkittävimmät käyttöjärjestelmät puhelimille ovat iOS ja Android. Puhekielessä kuitenkin puhutaan iPhonesta ja Androidista. Sovellusten kehittäminen on yksilöllistä, koska molemmilla alustoilla on oma käyttöjärjestelmä, ohjelmointiympäristö, ohjelmointikieli sekä julkaisuprosessi.

Tietojenkäsittelyn opinnäytetyön aihe pohjautuu toimeksiantajalle tehtyyn sovelluskehitykseen, mikä toteutettiin käyttäen avoimeen lähdekoodiin perustuvaa JavaScript-kirjastoa, React Nativea. Toimeksiantajalle sovellusta kehitettiin iPhonelle ja käytössä tähän oli Apple MacBook. Opinnäytetyön tarkoitus on perehtyä alustariippumattomaan sovelluskehitykseen syvällisemmin, ja luvussa neljä tulee selville, miksi Apple MacBook on mobiilisovelluksia kehittäessä monipuolinen työväline.

2 TAVOITTEET JA TEKNOLOGIAN VALINTA

Opinnäytetyön aihe tuli harjoittelupaikasta jo harjoittelun aikana. Toimeksiantaja ehdotti mobiilisovelluskehitystä. Projekti kiinnosti, koska se antoi hyvän oppimispohjan ja tarttumispinnan ohjelmistokehitykseen. Sovelluskehitykselle asetettiin seuraavia tavoitteita:

- Sovellus kehitetään ensin iPhonelle ja myöhemmin Androidille
- Komponentit ovat uudelleenkäytettäviä
- Käyttöliittymään panostetaan

Toimeksiantajan toiveena oli saada käyttöliittymälle demo, mikä toimisi perustana jatkokehitystä varten. Teknologiavalinnaksi osoittautui React Native, koska toimeksiantajalla on lyöty lukkoon jo aiemmin käyttöön tietyt teknologiat. Toimeksiantaja ei ollut aiemmin kehittänyt React Nativeen pohjautuvia sovelluksia, mutta yrityksellä oli kokemusta Reactista. Pintapuolisena suurimpana erona React on front-end-kehityskirjasto ja React Native on täysi mobiilikehityskirjasto (Radix 2022). Toissijaisena tavoitteena opinnäytetyölle oli tutustua macOS-käyttöjärjestelmään ja sovelluskehitykseen käyttämällä Applen tarjoamia työkaluja.

3 TEKNOLOGIOIDEN VERTAILU

Tässä luvussa vertaillaan teknologioita ja ohjelmointikieliä. Xamarin.Forms ja React Native eivät ole ohjelmointikieliä vaan ohjelmistokehyksiä. Swift ja Kotlin ovat ohjelmointikieliä. Kaikki luvussa esiintyvät teknologiat ovat tavalla tai toisella alustariippumattomia. Alustariippumattomuus ei tarkoita suoraan sitä, että teknologiaa voitaisiin käyttää millä tahansa käyttöjärjestelmällä ja ohjelmistoa kehitettäisiin kaikille alustoille. Alustariippumattomuutta on se, että ohjelmiston toteutuksessa ei ole käyttöjärjestelmään tai laitteistoon liittyviä sidottuja ominaisuuksia. Alustariippumattomuutta on myös ohjelmiston toimivuus ja käyttöliittymän samankaltaisuus laitteesta tai käyttöjärjestelmästä riippumatta (Technopedia 2015).

Swift

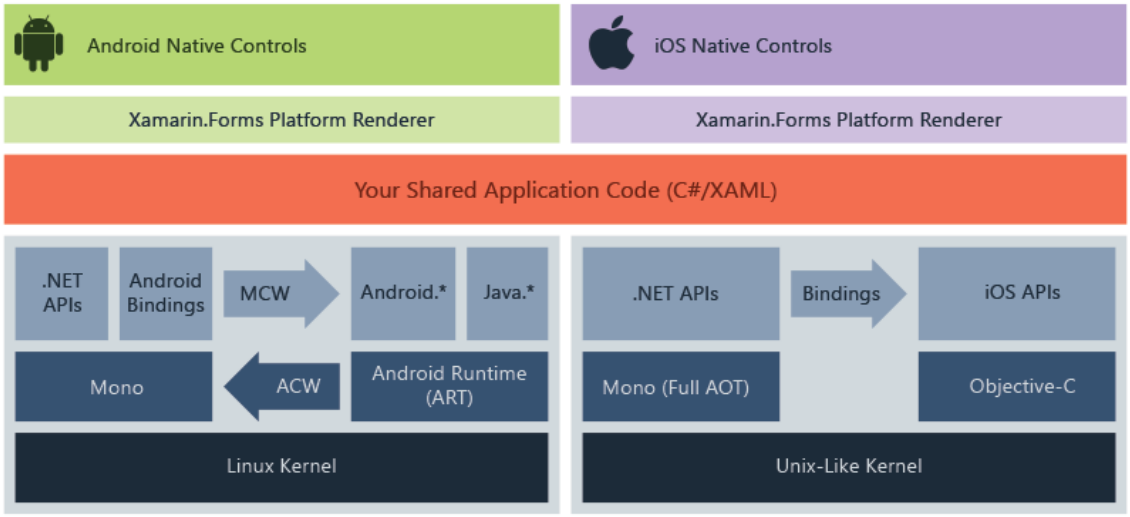
Swift on Applen 2014 julkaisema ohjelmointikieli iOS:lle ja macOS:lle. Apple on korvannut Objective-C:n Swiftillä. Apple on patentoinut Swiftin (The Register 2022), mikä on aiheuttanut epäluottamusta Swiftin käyttöönottoa kohtaan. IBM jopa luopui Swiftin palvelintuen kehitystyöstä patenttipäätöksen seurauksena (I Programmer 2020). Avoimen lähdekoodin Swiftiä voidaan käyttää Mac-tietokoneissa kohdistamaan kehitystyö kaikkiin Apple-alustoihin: iOS, macOS, watchOS ja tvOS (Swift 2022). Apple kertoo Swiftin olevan alustariippumaton, mutta käytännössä tämä ei kuitenkaan pidä täysin paikkaansa. Swift soveltuu Apple-alustojen lisäksi tällä hetkellä Linuxille. (Apple 2022b) Mikäli mobiilisovellusta kehitetään Kotlinilla Androidille ja Swiftillä iPhonelle, silloin koodiydintä ei jaeta ja koodin uudelleenkäytettävyysaste on matalampi kuin alustariippumattomalla teknologialla.

Kotlin

Kotlin on 2011 julkaistu avoimen lähdekoodin staattisesti tyypitetty ohjelmointikieli (InfoWorld 2022). Kotlin on suosittu ohjelmointikieli Android-sovelluskehityksessä, ja vaikka Kotlin on alustariippumaton, Google on vahvistanut sen viralliseksi Android-kehityskieleksi (GeeksForGeeks 2022). Kotlin-projektin luominen onnistuu Android Studiolla, mikä on saatavilla useille käyttöjärjestelmille.

Xamarin.Forms

Xamarin.Forms on avoimen lähdekoodin alusta modernien ja suorituskyvykkäiden sovellusten kehittämiseen iOS-, Android ja Windows-käyttöjärjestelmille .NET-ohjelmistokomponenttikirjastolla. Kuviossa 3 ilmenee Xamarin.Forms-toimintaperiaate. Microsoft on hankkinut omistukseen Xamarinin 2016 (Microsoft 2016). Xamarin tukee virallisesti Mac- ja Windows-tietokoneita, mutta epäviralllisten ohjeiden avulla Xamarin ja Xamarin.Forms ovat saatavilla myös Linux-tietokoneille (Mockit 2020).



KUVIO 3. Xamarin.Forms toimintaperiaate (Microsoft 2022)

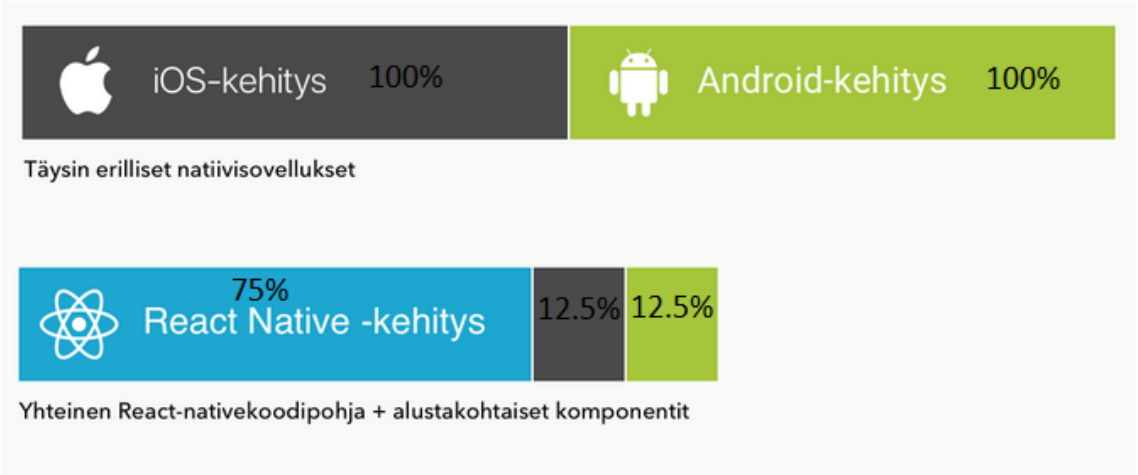
Xamarin.Forms soveltuu sovelluskehittäjille, jos heidän tavoitteenaan on kehittää samanlainen käyttöliittymä kaikilla alustoille, minkä koodin, testauksen ja logiikan voi jakaa kaikkien alustojen kesken. Sovelluskehitys tapahtuu Visual Studio-ohjelmointiympäristössä käyttäen C#-ohjelmointikieltä (Microsoft 2022.)

Xamarin.Formsilla on kattava ekosysteemi kirjastoja, jotka lisäävät sovelluksiin monipuolisia toimintoja. Xamarin.Forms Shell vähentää mobiilisovellusten kehittämisen monimutkaisuutta tarjoamalla perusominaisuudet, joita useimmat sovellukset vaativat. Yhteinen navigointikokemus, URI-pohjainen navigointimalli ja intergoitu hakukäsittelijä ovat esimerkkejä Shellin tarjoamista ominaisuuksista. Xamarin.Forms tarjoaa yhteisen ohjelmistorajapinnan. Material Visual mahdollistaa käyttöliittymän yhtenäisen ulkoasun ja käyttötuntuman sekä iOS:ssä ja Androidissa (Microsoft 2022.) Kuviossa 4 on luotu alkuvaiheissa oleva Xamarin.Forms-sovellus.

KUVIO 4. Xamarinilla toteutettu varhainen versio mobiilisovelluksesta

React Native

React Native on Metan luoma avoimen lähdekoodin UI-ohjelmistokehys. React Native on alusta-riippumaton, ja sovelluksia kehitetään Androidille, iPhonelle ja Windowsille. Myös verkkoselaimella käytettäviä web-sovelluksia voidaan kehittää React Nativella. React Nativella on paljon hyötyjä. Sovellusta voidaan kehittää samanaikaisesti sekä iPhonelle että Androidille. Fraktion blogikirjoituksesta ja kuviosta 5 selviää, että React Nativen uudelleenkäytettävyysaste olisi noin 75 %:in luokkaa (Fraktio 2021).



KUVIO 5. Mobiilisovelluksen toteutuksen työmäärä (Fraktio 2021, muokattu)

Ennen kehitystyön aloittamista toimeksiantajalle minulla ei ollut kokemusta React Nativesta kuin yhdeltä opintojaksolta. Tiesin kuitenkin jo ennestään, että sovellusta olisi mahdollista kehittää myös Androidille samaan aikaan. Tavoitteena oli kuitenkin aluksi luoda sovelluksesta demo iPhonelle, joten kehitystyössä käyttöliittymä kehitettiin vain iPhonelle.

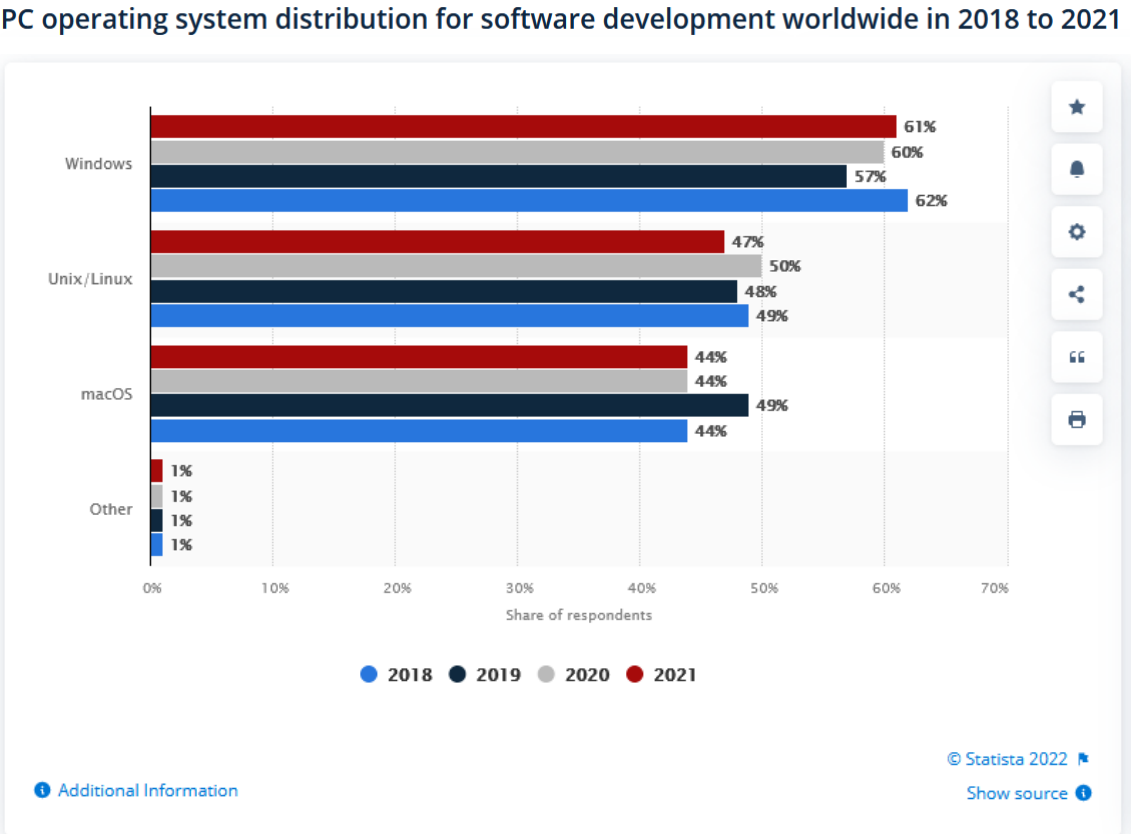
React Native on nopea oppia jopa täysin itsenäisesti. Käyttöliittymän luominen komponenteilla vaatii aluksi jonkinlaista opettelua. Haasteena internetistä löytyvien valmiiden komponenttien käytössä voi olla se, ettei GitHubissa välttämättä kerrota, miten niitä todellisuudessa käytetään. Aloittelijan kannattaa opetella aluksi luomaan omia komponentteja, jotta React Nativen syntaksin tunteminen kasvaa tarpeeksi suureksi.

4 OHJELMOINTIYMPÄRISTÖT

Mobiilisovelluskehityksessä käytetään ohjelmointiympäristöjä. Ohjelmointiympäristöt sisältävät yleisesti koodieditorin, käännösautomaation ja debuggerin. Mobiilikehitys vaatii kuitenkin renderöinnin joko fyysiselle laitteelle tai simulaattorille. iPhone-simulaattori sisältyy Xcode-ohjelmointiympäristöön, ja Android-simulaattori löytyy vastaavasti Android Studiosta.

Koodieditoreina Xcode ja Android Studio ovat hyvin epäsuosittuja. Vuoden 2019 tilastojen mukaan suosituin editori on VS Code (WakaTime 2020).

Käyttöjärjestelmän valinta on monesti kiinni kehittäjän omista mieltymyksistä. iOS-sovellukset ovat tässä poikkeus, ja ne painostavat kehittäjiä käyttämään macOS-käyttöjärjestelmää. Statistan tilastoista kuviossa 6 nähdään, että vuonna 2021 macOS-käyttöjärjestelmän osuus oli 44 % (Statista 2022). Kuluttajien käytössä macOS:n markkinaosuus on vain 16 % (StatCounter 2022).



KUVIO 6. Statistiikkaa käyttöjärjestelmien osuudesta sovelluskehityksessä (Statista 2022)

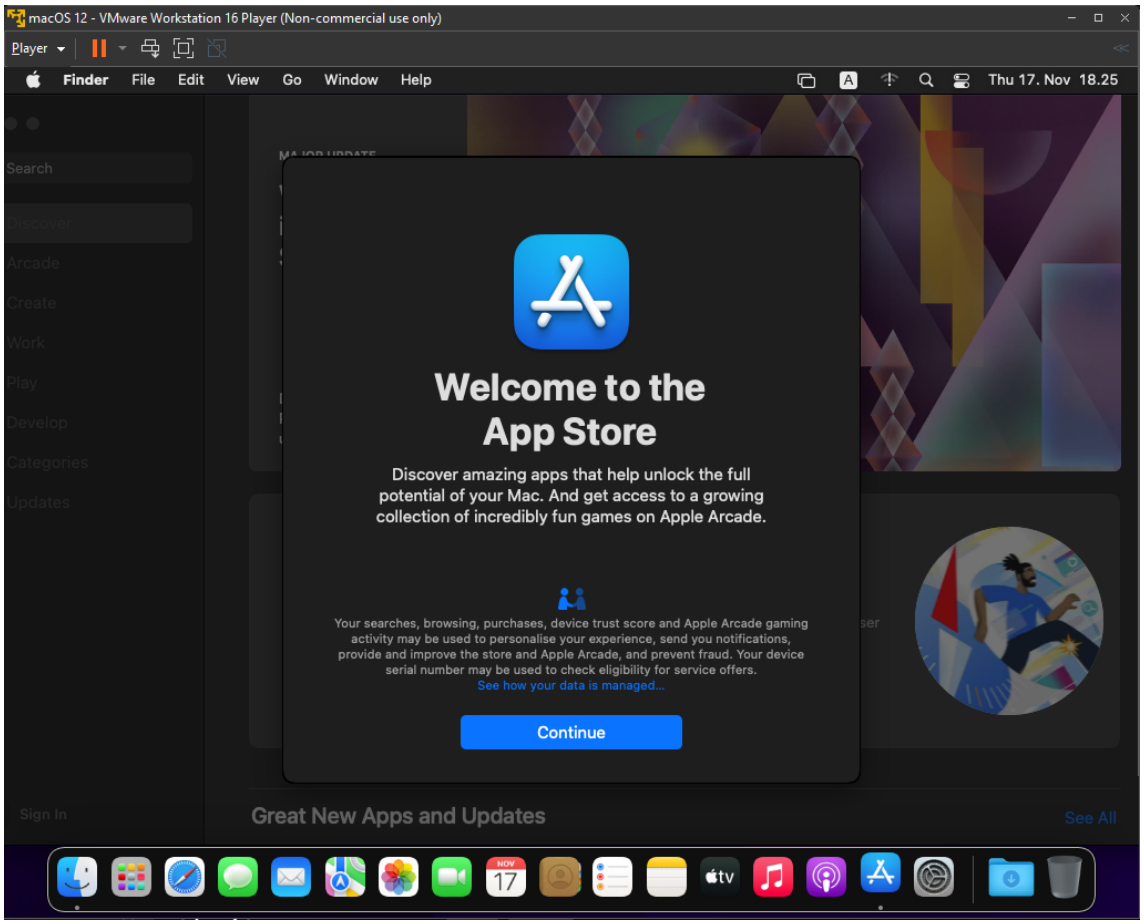
Xcode

Xcoden uusin versio on kirjoitushetkellä Xcode 14. Xcode on Applen virallinen ohjelmointiympäristö, ja se sisältää kaiken, mitä tarvitsee kehitystyötä, testausta ja julkaisua varten. Xcode on suunniteltu monialusteiseksi ja sillä voidaan kehittää sovelluksia iOS:ille, iPadOS:ille, macOS:ille ja tvOS:ille. (Apple 2022a) Xcode on saatavilla vain macOS-käyttöjärjestelmille. Applen MacBook tukee myös Android Studiota, ja tämä lisää kehittäjien suosiota käyttää Unix-pohjaista macOS-käyttöjärjestelmää. Xcodella voidaan luoda Swift-projekteja.

Toimeksiantajalle kehitettiin iPhone-sovellus ja toimeksiantaja halusi selvittää, mitä työvalineitä tarvitsen työskentelyä varten. Selvitin mahdollisuutta käyttää virtuaalikonetta, mikä pitäisi sisällään macOS-käyttöjärjestelmän. Virtuaalikoneen kautta minun onnistui asentaa Xcode-ohjelmisto ilman kaikkia tarvittavia ominaisuuksia. Virtuaalikonetta käytettiin VMware Workstation 16 Player -ohjelmistolla, ja ohjelmisto vaatii tietokoneelta hyvin paljon resursseja. Mobiilisovelluksen kehittämiseen virtuaalikone ei ole soveltuva, syitä tähän on seuraavia:

- Sujuvaan käyttöön järjestelmävaatimuksia ei saavutettu.
- Virtuaalikone kärsi epävakaudesta ja teki uudelleenkäynnistyksiä.
- Xcoden toiminnot eivät toimineet toivotulla tavalla.

Vaikka käytössä olisi tehokas Windows-tietokone, virtuaalikone toimisi vakaasti ja Xcode toimisi toivotusti, löytyy uusi ongelma macOS-käyttöoikeussopimuksista. Applen keskusteluforumilla KMT-käyttäjäprofiili huomauttaa, että käyttäessä macOS-käyttöjärjestelmää virtuaalikoneella olisi VMware-ohjelmistoa käytettävä Mac-tietokoneella. Ilman fyysistä Mac-tietokonetta Xcoden käyttö on macOS-lisenssisopimuksia vastaan. (Apple 2019.)



KUVIO 7. macOS-käyttöjärjestelmä virtuaalikoneella

Sain käyttööni Apple MacBookin M1-prosessorilla, 16 gigatavun keskusmuistilla ja 512 gigatavun kiintolevyllä, mikä vastasi hyvin suositeltuja Xcoden järjestelmävaatimuksia. Vaatimusten saavuttaminen on tärkeää kehitystyön kannalta, koska iPhone-simulaattoria käytetttäessä kehittäjä saattaa joutua käynnistämään simulaattorin useita kertoja uudelleen nähdäkseen tekemät muutokset. Koodari ei halua jäädä useiksi minuuteiksi odottelemaan simulaattorin käynnistymistä.

Android Studio

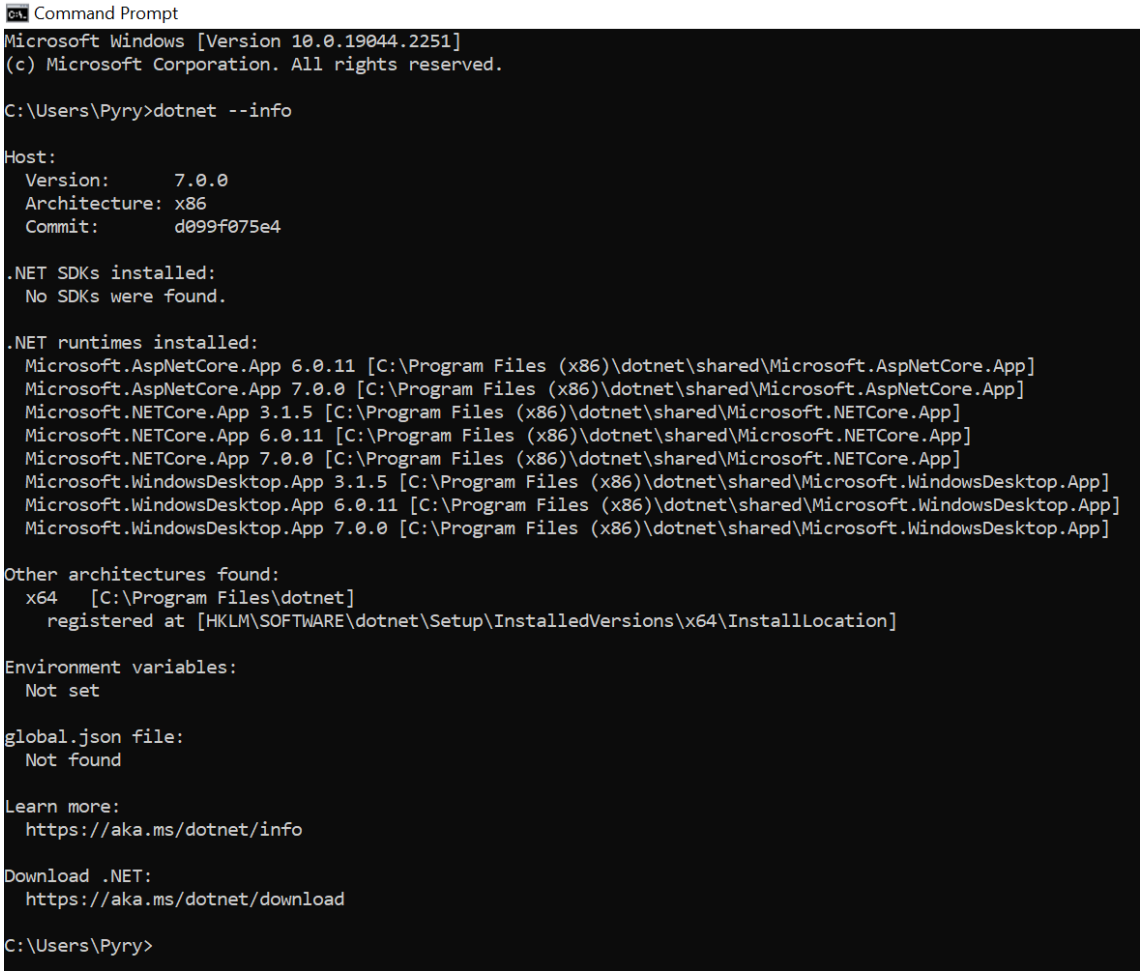
Android Studio on Android-käyttöjärjestelmien virallinen ohjelmointiympäristö ja sisältää kaiken tarvittavan Android-sovellusten kehitystyötä, testausta ja julkaisua varten (Android Studio 2022). Android Studiolla voidaan kehittää sovelluksia Android-laitteiden lisäksi myös Wear OS:lle, Android TV:lle ja Android Autolle. Android Studiolla voidaan luoda Kotlin- ja Java-projekteja.

Visual Studio

Visual Studio on Microsoftin kehittämä sovelluskehitysympäristö. Visual Studiossa voidaan käyttää useita ohjelmointikieliä. Microsoftin C# on .NET-alustalle kehitetty ohjelmointikieli ja sopii täydellisesti käytettäväksi Visual Studiossa. Microsoft myös tarjoaa koodieditoria Visual Studio Code, mikä taas sopii muihin kieliin, kuten PHP, paremmin.

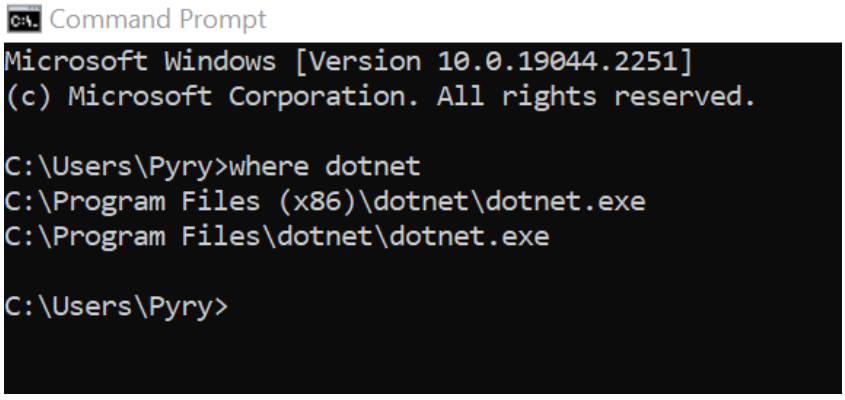
Visual Studiossa onnistuu koodata, kehittää, debugata ja analysoida kaikkea, mikä kuuluu sovelluskehitykseen. Visual Studio on saatavilla Windowsille, Linuxille ja Macille. Visual Studiossa on solutioneja ja projekteja. Yhteen solutioniin voi sisältyä useita projekteja, mutta jokaisen projektin täytyy kuitenkin kuulua johonkin solutioniin. Myös emulaattori sisältyy ohjelmistoon. Alustariippumaton ohjelmointiympäristö mahdollistaa siis myös sovelluskehityksen iPhonelle ilman, että käytössä on Applen Mac-tietokone.

Xamarin-sovelluksia kehitetään Visual Studiossa. Windows-käyttäjillä on ilmennyt Xamarinin ja Visual Studion käyttöönotossa ongelmia. Xamarin käyttää .NET-frameworkkia ja asennus ei välttämättä onnistu kuten dokumentaatiossa tapahtuu. Xamarin-sovellusta käynnistäessä Visual Studio ilmoittaa, ettei löydä SDK:ta. Tämä johtuu siitä, että käyttöjärjestelmä yrittää käyttää 32-bittistä asennusta 64-bittisen sijaan. Kyseisen ongelman kartoittaminen tapahtuu kirjoittamalla komentoriville kysely”dotnet –info”. Kuviossa 8 esiintyvällä dotnetillä viitataan .NET-sovelluskehykseen. Komentorivi ilmoittaa, ettei .NET SDK:ta löyty. Samalla kuviosta ilmenee järjestelmän yrittävän käyttää 32-bittistä asennusta. Ongelma on siinä, että Xamarinia varten asensin 64-bittisen version.

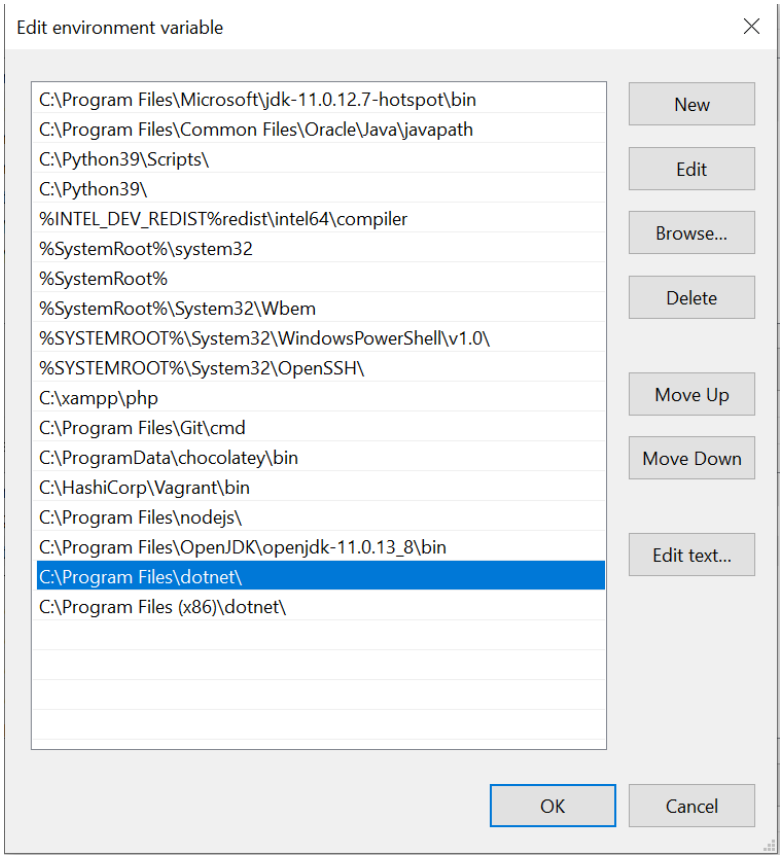


KUVIO 8. Komentorivi kertoo, ettei löydä .NET SDK:ta

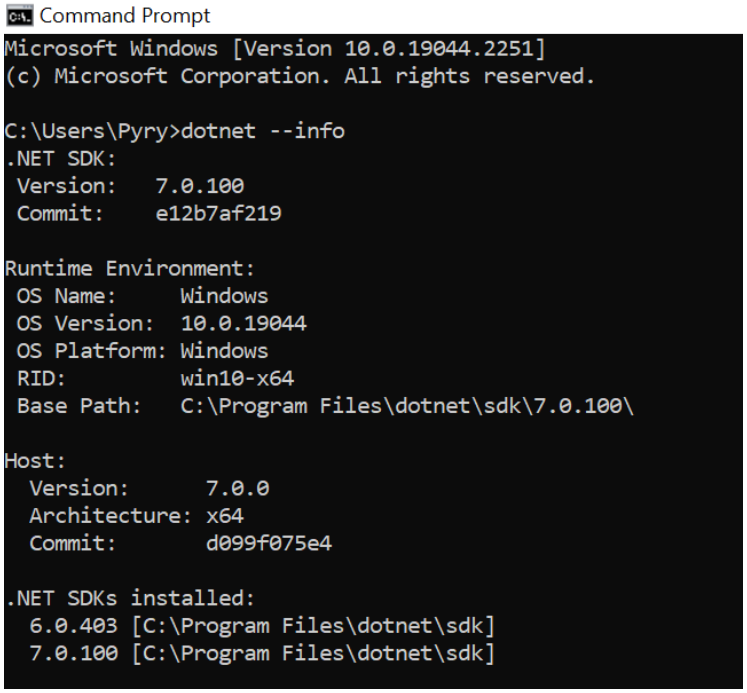
Komennolla ”where dotnet” (kuvio 9) komentorivi kertoo, että 32-bittinen versio on ennen 64-bittistä versiota. Ongelman löydyttyä sen voi korjata muuttamalla ympäristömuuttujien järjestystä siten, että haluttu 64-bittinen .NET-asennus on järjestyksessä ensin. Kuviossa 10 esiintyy Windows-käyttöjärjestelmän ympäristömuuttujat ja kuviossa 11 todetaan muutosten ratkaisevan ongelman.



KUVIO 9. Komentorivi kertoo 32-bittisen asennuksen olevan valittuna ennen 64-bittistä.



KUVIO 10. Ympäristömuuttujien järjestyksen muuttaminen.



KUVIO 11. Komentorivi löytää .NET SDK:n

5 KÄYTTÖLIITTYMÄN SUUNNITTELU

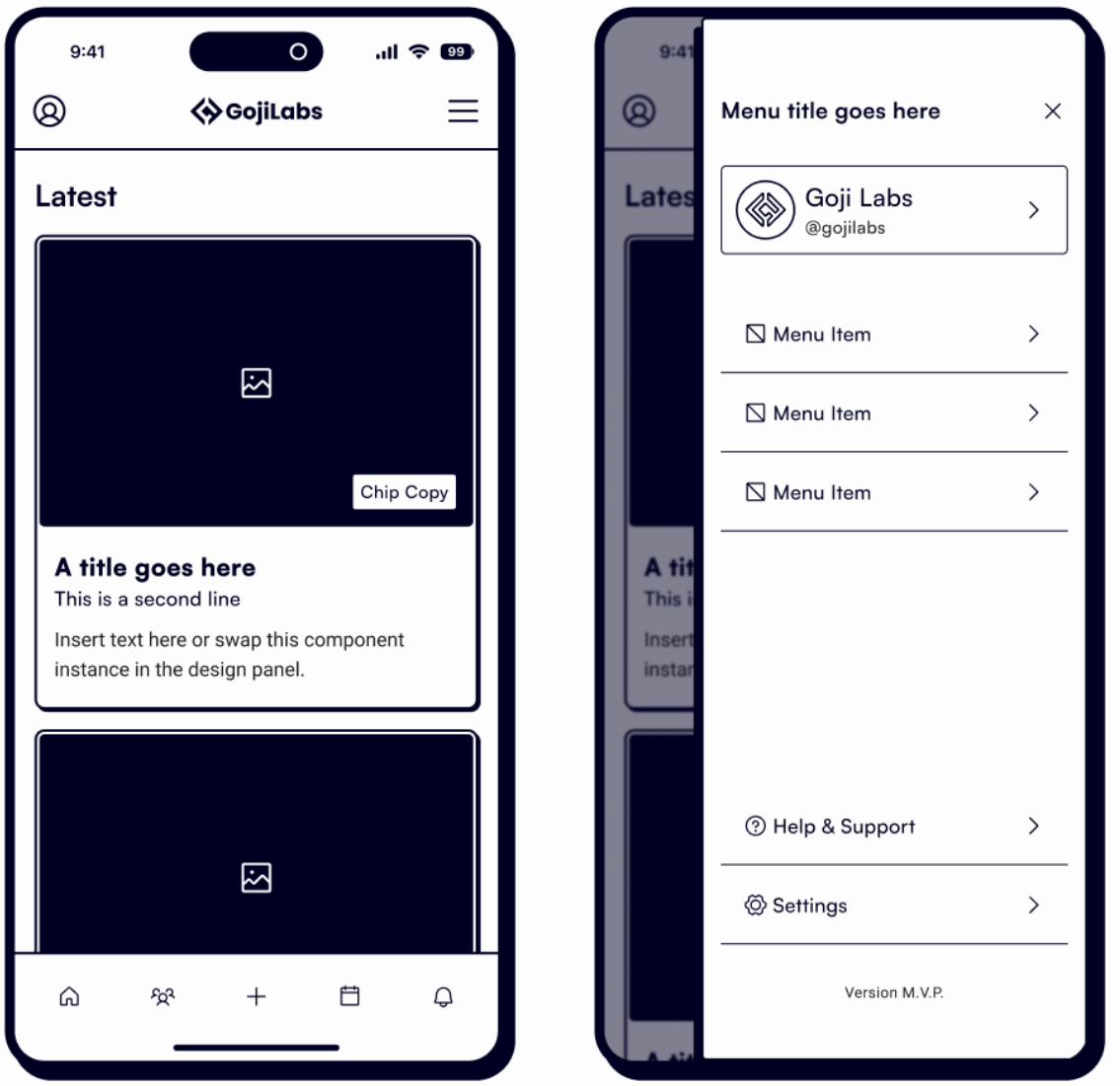
Mobiilisovelluksessa käyttöliittymä on se, millä käyttäjä käyttää sovellusta. Toimeksiantajalla oli selvä käsitys käyttöliittymästä. Suunnitteluvaiheessa luodaan ensin visuaalisia kuvia käyttöliittymästä ilman ohjelmointivaihetta. Käyttöliittymää eli UI:ta voidaan suunnitella esimerkiksi seuraavilla ohjelmistoilla:

- Adoben ohjelmistot
- Figma
- Canva.

Ohjelmiston valinta riippuu paljon henkilökohtaisesta mieltymyksestä sekä yrityksen käytännöistä. Figman ja Canvan etuina on ilmaisversion mahdollisuus. Ammattikäyttäjä kuitenkin vaatii maksullisen version säilyttääkseen työhistorian ja voidakseen luoda useamman luonnoksen. Käyttöliittymän esittelyssä näytetään Figman UI-design-kuvioita. Toimeksiantaja antoi käyttöliittymän eli UI:n suunnitelman Adobe XD -tiedostona, mutta toimeksiannon käyttöliittymää ei käytetä esimerkkinä liikesalaisuuden vuoksi. Adobe XD toimi mainiona työkaluna kehitystyössä, koska sieltä pystyi lataamaan sovellukseen tulevat kuvakkeet ja näki tarkat elementtien mitat. Lisäksi käyttöliittymästä sai hyvän käsityksen, kun Adobe XD -tiedostossa pystyi liikkumaan eri näkymissä suoraan painamalla käyttöliittymässä olevia kuvakkeita.

Figma on käyttöliittymäsuunnitteluun tarkoitettu työkalu. Figma toimii verkkoselaimella ja myös työpöytä- ja mobiilisovelluksina. Adobe ilmoitti 15. syyskuuta 2022 hankkivansa Figman 20 miljardilla Yhdysvaltain dollarilla (Forbes 2022).

Figmalla voidaan luoda oma UI-työkalupakki, mistä löytää kuvakkeiden ulkoasut, fontin koon ja kaikkea muuta käyttöliittymään kuuluvaa tarvittavaa materiaalia kuten värikoodit. Figmasta löytää myös valmiita pohjia (kuvio 12), mihin voi lisätä omaa sisältöä nopeasti ja säästää aikaa suunnittelussa.



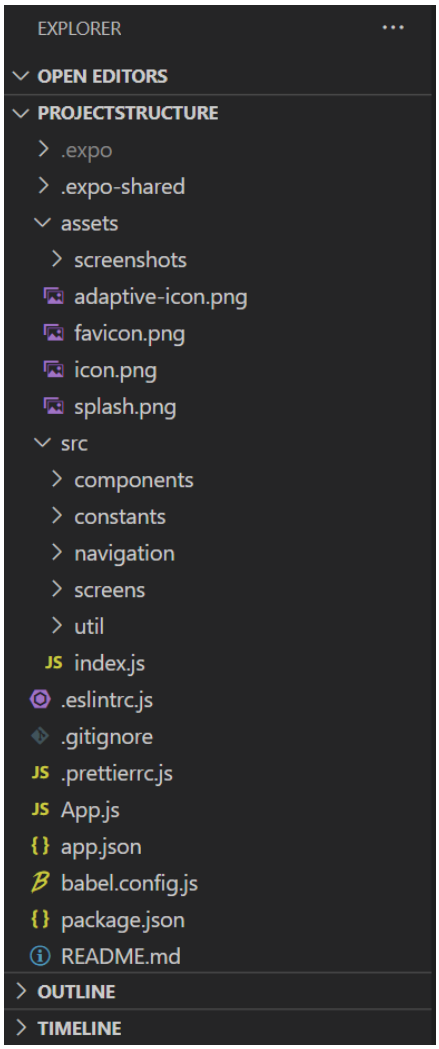
KUVIO 12. Figmalla toteutettu käyttöliittymän esimerkki (Figma 2022)

UX-design on käsitteenä lähellä UI-designia. UX tulee englannin kielen termistä ”User experience” eli käyttäjäkokemus. Puhuttaessa näistä kahdesta asiasta asiat risteävät usein ja aina ei saa selkeää käsitystä, kummasta asiasta milläkin hetkellä todellisuudessa puhutaan. UX-design ja UI-design ovat vahvasti yhteyksissä toisiinsa, mutta eivät kuitenkaan sama asia. Żaneta Lenczewskan blogikirjoitukseen perustuva vertailu (kuvio 13) käsitteiden välillä kuvastaa keskeisiä eroja ja yhtäläisyyksiä. (Pagepro 2022.)

UX-design	UI-design
Ihmislähtöinen lähestymistapa tuotteen suunnitteluun	Ihmislähtöinen lähestymistapa tuotteen esteettisen kokemuksen suunnitteluun
Prosessi sellaisten tuotteiden rakentamiseen, mitkä tarjoavat erinomaisen käyttökokemuksen	Prosessi, jossa metallikehyksiä muutetaan houkutteleviksi ja helppokäyttöisiksi käyttöliittymiksi
Keskittyy käyttäjiin ja heidän matkaansa tuotteen läpi	Keskittyy visuaalisiin ja interaktiivisiin elementteihin
Tavoitteena on parantaa käyttäjävuorovaikutusta ja tuotteiden käyttökokemusta	Tavoitteena on luoda houkutteleva, helppokäyttöinen ja skaalautuva tuote

KUVIO 13. UX- ja UI-designien eroavaisuudet ja yhteneväisyydet

Ohjelmistorakenteen suunnittelu on vaihe ennen varsinaiseen kehitystyöhön lähtemistä. Kehittäjän on tärkeää löytää oikeat näkymät, komponentit, kuvakkeet ja navigaatio vaivattomasti. Komponenttien on hyvä olla erillään omassa kansiossaan. Jos komponentin rakentaa jonkin näkymän eli screenin sisään, komponentin uudelleenkäyttö menee hyvin vaikeaksi. Tuttu englanninkielinen iskulause ”Write once, run anywhere” eli kirjoita kerran, suorita missä tahansa, on hyvä muistisääntö sovelluksen suunnittelussa. Ei ole mitään järkeä kirjoittaa koodia useita kertoja, jos sitä voidaan käyttää helposti uudelleen luomalla siitä komponentti. Ohjelmistorakenne vaihtelee myös teknologioittain. Xamarin.Forms esimerkiksi tekee kehittäjälle alustavan, selkeän rakenteen jo projektia luotaessa.



KUVIO 14. Esimerkki käyttöliittymän varhaisesta ohjelmistorakenteesta

6 SOVELLUKSEN TESTAAMINEN

Ohjelmistotestaus on ohjelmistokehityksen tärkeä osa-alue, jossa tarkastetaan sovellukselle asetetut vaatimukset ja varmistetaan, että sovelluksessa ei ole ohjelmistovirheitä. Testaus sisältää ohjelmiston komponenttien suorittamisen manuaalisilla tai automaattisilla työkaluilla yhden tai useamman valitun ominaisuuden arvioimiseksi. Ohjelmistotestauksen tarkoituksena on löytää virheitä ja puitteita. Isoissa ohjelmistoyrityksissä on testaukseen erikoistunutta henkilöstöä. Testaajat työskentelevät muiden tiimiläisten kanssa projektissa, ja testaajalta saa arvokasta palautetta sovelluksen toimivuudesta.

Virheiden löytämisen lisäksi testitulokset antavat varmistuksen, että koodi on kirjoitettu oikein ja projektiin voidaan ruveta lisäämään uusia ominaisuuksia. Testien dokumentointi helpottaa ymmärtämään koodin toiminnallisuutta, jos projektiin liittyy uusia tiimiläisiä ja he eivät ole nähneet koodia ennestään. Testattavan koodin kirjoittaminen on osa ohjelmistosuunnittelua ja mahdollistaa, että koodi on useissa pienissä moduuleissa pitkän kokonaisen useita koodirivejä sisältävän tiedoston sijaan. Ohjelmistotestauksessa testitaulukko voidaan suunnitella dokumentointityökalulla esimerkiksi Wordillä tai Excelillä. Esimerkkitapauksessa (kuvio 17) kirjoitetaan testitapauksen kuvaus, testivaiheet, mahdollinen testidata, odotettu lopputulos ja onnistuiko testi vai ei (Guru99 2022b.)

```
describe('Form', () => {
    it('funktion kutsuminen lähetyspainikkeen painamisen jälkeen.', () => {
      const onSubmit = jest.fn();
      const { getByPlaceholderText, getByText } = render(<Form onSubmit={onSubmit} />);

      fireEvent.changeText(getByPlaceholderText('Käyttäjätunnus'), 'pyry');
      fireEvent.changeText(getByPlaceholderText('Salasana'), 'salasana');
      fireEvent.press(getByText('Lähetä'));

      expect(onSubmit).toHaveBeenCalledTimes(1);

      //
      expect(onSubmit.mock.calls[0][0]).toEqual({
        username: 'pyry',
        password: 'salasana',
      });
    });
  });
```

KUVIO 15. Esimerkki testaamisesta koodilla

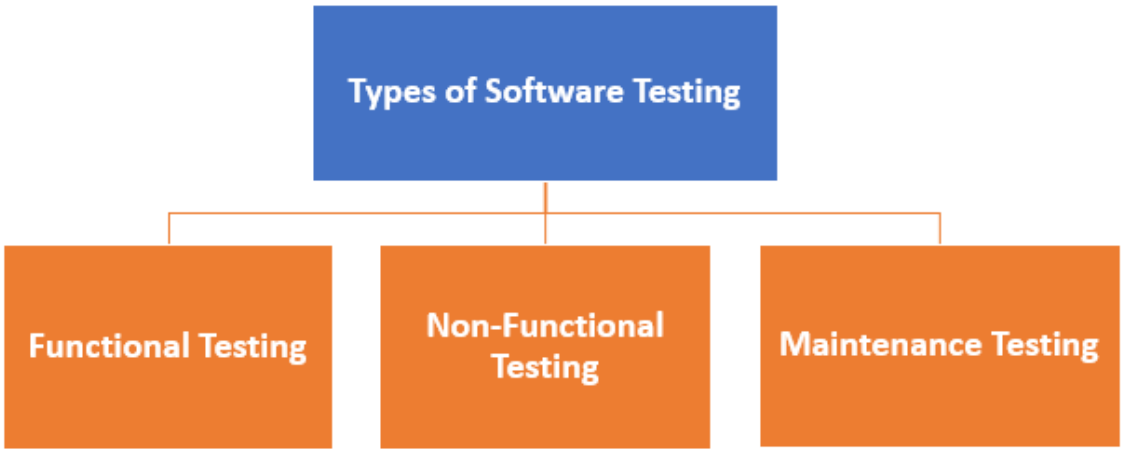
Ohjelmistotestauksen hyödyt ja edut:

- Kustannustehokas: Yksi ohjelmistotestauksen tärkeimmistä eduista. IT-projektin testaaminen ajoissa auttaa säästämään rahaa pitkällä aikavälillä. Vikojen havaitseminen ohjelmistotestauksen aikaisemmassa vaiheessa maksaa vähemmän.
- Turvallisuus: Tietoturvan varmistaminen on tärkeä testauksen etu, koska ihmiset etsivät luotettavia tuotteita. Se auttaa poistamaan riskit ja ongelmat ennen sovelluksen julkaisua.
- Tuotteen laatu: Olennainen vaatimus tuotteelle on sen laadunvarmistus. Testaus varmistaa, että laadukas tuote toimitetaan asiakkaille.
- Asiakastyytyväisyys: Minkä tahansa tuotteen päätavoitteena on tarjota asiakkailleen tyytyväisyyttä. UI/UX-testaus takaa parhaan käyttökokemuksen.

Ohjelmistotestauksen tyypit

Tyypillisesti testaus luokitellaan kolmeen kategoriaan (kuvio 16):

- Toiminnallinen testaus
- Ei-toiminnallinen testaus tai suorituskykytestaus
- Ylläpito (regressio ja ylläpito) (Guru99 2022a.)



Kuva 16. Testityypit ohjelmistosuunnittelussa (Guru99 2022a)

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	1. Go to site `http://demo.guru99.com` 2. Enter UserId 3. Enter Password 4. Click Submit	Userid = guru99 Password = pass99	User should Login into an application	As Expected	Pass
TU02	Check Customer Login with invalid Data	1. Go to site `http://demo.guru99.com` 2. Enter UserId 3. Enter Password 4. Click Submit	Userid = guru99 Password = glass99	User should not Login into an application	As Expected	Pass

KUVIO 17. Testitapaus kirjautumisesimerkistä (Guru99 2022b)

Testaustasoja voi olla useita. Toiminnallinen testaus varmistaa, että ohjelmisto toimii toivotulla tavalla ja täyttää asetetut vaatimukset toiminnallisuutensa puolesta. Toiminnallinen testaus voidaan tehdä loppukäyttäjän näkökulmasta käyttöliittymää käyttäen. Ei-toiminnallisessa testauksessa käydään läpi muuta kuin näkyvää toiminnallisuutta liittyen suorituskykyyn ja resurssien käyttöön.

Regressiotestaus pitää sisällään toiminnallisen testauksen ja ei-toiminnallisen testauksen. Regressiotestauksessa tarkistetaan, että ohjelmistoon tehdyt muutokset eivät ole aiheuttaneet ongelmia aiemmin toimiviksi todettuihin toiminnallisuuksiin. Käytettävyystestauksessa tutkitaan, kuinka hyvin ohjelmisto toimii ja onko se helppokäyttöinen sekä ovatko värit, kontrastit ja siirtymät hyväksyttäviä. Käytettävyystestauksessa ohjelmistolle voidaan esimerkiksi asettaa vaatimus, ettei erilaiseen näkymään siirtyminen saa kestää kuin korkeintaan 400 millisekuntia.

Hyväksymistestaus on testauksen osa-alue, jossa ohjelmiston loppuasiakas tai asiakkaan edustaja tarkastaa valmiin tai lähellä valmistumisvaihetta olevan tuotteen. Hyväksymistestaaja tarkistaa, vastaako tuote vaatimuksia. Tutkiva testaus on vähemmän suunnitelmallista testausta. Tutkivan testauksen aikana dokumentoidaan päiväkirjaan tehtyjä asioita ja mahdollisesti löytyneitä löydöksiä. Tutkivan testauksen painopistealueet sovitaan etukäteen (Itewiki 2022.)

7 POHDINTA

Opinnäytetyön toiminnallisen osuuden tavoitteena oli kehittää iOS-käyttöjärjestelmälle mobiilisovelluksen demo React Native -teknologialla. Demon päätarkoitus oli toimia visuaalisena esityksenä, joka selkiyttää sovelluksen käyttötarkoitusta. Mobiilisovellukseen haluttiin saada käyttöliittymä, mihin on panostettu huolellisesti, ja jatkokehitystä varten siellä on tarvittavat painikkeet ja navigaatiot valmiina. Sovelluksessa oli iOS-käyttöjärjestelmälle tarkoitettuja ominaisuuksia ja jatkokehityksessä projektin vastuulleen ottavan tiimin täytyy kehittää vastaavia toimintoja Androidille. Sovelluksen tavoitteena oli saada käyttöliittymä näyttämään samalta kuin se näyttää suunnitelmissa ja lisätavoitteena oli saada ylimääräisiä toimintoja. Ensisijainen tavoite toteutui ja toissijainen tavoite jäi toteutumatta, mikä aiheutti itselleni jonkinasteista tyytymättömyyttä sovelluksen lopputulokseen. Kehitin mobiilisovellusta itsenäisesti ja sain arvokasta konsultaatiota yrityksen työntekijältä perustuen hänen React-osaamiseensa. Pelkkä React-tietämys ei kuitenkaan ollut avuksi monessa mobiilikehitykseen liittyvässä asiassa, ja olisi ollut suuri lisähyöty, jos konsultaatiota antava henkilö olisi pystynyt osallistumaan koodin kirjoittamiseen.

Opinnäytetyö on teoriapainotteinen, koska allekirjoitin salassapitosopimuksen ja en voinut käyttää mobiilisovellukseen käytettyä materiaalia. Opinnäytetyössä esiteltiin alustariippumattoman mobiilisovelluksen kehitystä välillä viitaten toiminnalliseen osuuteen. Teoreettisessa osuudessa esiteltiin aluksi syitä teknologian valintaan ja myös omia intressejä ohjelmointikokemuksen saamisessa. Opinnäytetyö etenee loogisesti vaihe vaiheelta siten, että opinnäytetyön luettua mobiilikehitystä harkitseva pystyy tekemään valinnan käytettävistä teknologioita ja työkaluista ennen projektin aloitusta. Eniten hyötyä opinnäytetyöstä on sellaiselle, jolla on jo ohjelmistokokemusta ja joka haluaa lähteä kehittämään mobiilisovellusta.

Projekti oli haastava ja vaati paljon päivittäistä panostusta. React Native aiheutti projektin alussa ongelmia, koska uudessa Apple MacBook -tietokoneessa oli uusi M1-siru, mille ei ollut vielä tullut tukea React Nativen puolesta, ja React Nativen nettisivuilla asiasta tiedotettiin vasta myöhemmin. Projektin aloittamiselle oli kuitenkin käytettävissä vaihtoehtoinen tapa, ja projekti myöhemmin muutettiin Expo Cli:stä React Native Cli -teknologiaan, kun M1-sirun ongelma korjattiin React Nativen puolelta. Haasteita myös aiheutti se, että tämä oli ensimmäinen isompi työ React Nativella, ja en ollut tietoinen kaikesta, mitä on ja mitä ei ole mahdollista toteuttaa.

Loppupäätelmänä olen tyytyväinen toimeksiantona tehtyyn työhön sekä opinnäytetyöhön. Sain paljon ohjelmistokokemusta lyhyessä ajassa, ja erilaisten teknologioiden käyttöönotto ja opettelu on tulevaisuudessa paljon helpompaa kuin ennen projektin aloittamista. Myös tehtäviin kuluvan ajan hahmottaminen luonnistuu paremmin, ja osaan arvioida, meneekö ohjelmoitavassa asiassa tunteja vai päiviä.

LÄHTEET

Android 2022. Android Studio. Hakupäivä 14.11.2022. https://developer.android.com/studio

Apple 2019. Using Virtual Mac is acceptable? Hakupäivä 7.9.2022. https://developer.apple.com/forums/thread/118085

Apple 2022a. Xcode 14. Hakupäivä 14.11.2022. https://developer.apple.com/xcode/

Apple 2022b. Swift. Hakupäivä 1.12.2022. https://developer.apple.com/swift/

Figma 2022. A Mobile Lo-fi UI Kit by Goji Labs. Hakupäivä 14.11.2022. https://www.figma.com/community/file/1169305666571254524

Forbes 2022. Adobe's $20 Billion Takeover Of Figma Makes Cofounders Billionaires. Hakupäivä 20.11.2022 https://www.forbes.com/sites/iainmartin/2022/09/15/adobes-20-billion-takeover-of-figma-makes-cofounders-billionaires/

Fraktio 2021. React Native: kaikki mitä olet aina halunnut kysyä. Hakupäivä 10.11.2022. https://www.fraktio.fi/blogi/react-native-kaikki-mita-olet-aina-halunnut-kysya

GeeksForGeeks 2022. Top Programming Languages for Android App Development. Hakupäivä 10.11.2022. https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/

Guru99 2022a. What is Software Testing? Definition. Hakupäivä 28.11.2022. https://www.guru99.com/software-testing-introduction-importance.html

Guru99 2022b. How to Write Test Cases in Software Testing with Examples. Hakupäivä 28.11.2022. https://www.guru99.com/test-case.html

I Programmer 2020. IBM Stops Work on Server-Side Swift. Hakupäivä 10.11.2022. https://www.i-programmer.info/news/98-languages/13369-ibm-stops-work-on-server-side-swift.html

InfoWorld 2022. What is Kotlin? The Java alternative explained. Hakupäivä 10.11.2022. https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html.

Itewiki 2022. Laadunvarmistus ja ohjelmistotestaus. Hakupäivä 28.11.2022. https://www.ite-wiki.fi/opas/laadunvarmistus-ja-ohjelmistotestaus/.

MTV Uutiset 2016. Historiallinen käänne: Internetiä selattiin enemmän mobiililaitteilla kuin tietokoneilla Hakupäivä 7.11.2022. https://www.mtvuutiset.fi/artikkeli/historiallinen-kaanne-internetia-selattiin-enemman-mobiililaitteilla-kuin-tietokoneilla/6146978.

Microsoft 2016. Microsoft to acquire Xamarin and empower more developers to build apps on any device. Hakupäivä 29.11.2022. https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/.

Microsoft 2022. What is Xamarin.Forms? Hakupäivä 27.11.2022. https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms.

Mockitt 2020. Can Xamarin Run on Linux and Everything about Xamarin Linux. Hakupäivä 10.11.2022. https://mockitt.wondershare.com/develop/xamarin-linux.html.

Pagepro 2022. UX and UI Design in Mobile App Development Process. Hakupäivä 16.11.2022. https://pagepro.co/blog/ux-and-ui-design-in-mobile-app.

Radix 2022. React vs React Native: Which One to Choose and Why? Hakupäivä 9.11.2022. https://radixweb.com/blog/react-vs-react-native

The Register 2019. Apple: Trust us, we've patented parts of Swift, and thus chunks of other programming languages, for your own good. Hakupäivä 10.11.2022. https://www.theregister.com/2019/01/26/apples_swift_patents/.

Solita 2022. Mobiilikehitys. Hakupäivä 29.11.2022. https://www.solita.fi/mobiilikehitys/.

StatCounter 2022. Desktop Operating System Market Share Worldwide. Hakupäivä 14.11.2022. https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202208-202209-bar.

Statista 2022. PC operating system distribution for software development worldwide in 2018 to 2021. Hakupäivä 14.11.2022. https://www.statista.com/statistics/869211/worldwide-software-development-operating-system/.

Swift 2022. About Swift. Hakupäivä 29.11.2022 https://www.swift.org/about/.

Talentree 2022. Mitkä ovat mobiilisovelluksen edut? Hakupäivä 7.11.2022. https://talentree.fi/softa/mitka-ovat-mobiilisovelluksen-edut/.

Technopedia 2015. Cross Platform. Hakupäivä 15.12.2022. https://www.techopedia.com/definition/17056/cross-platform.

Valtiovarainministeriö 2022. Digitaalinen henkilöllisyystodistus. Hakupäivä 7.11.2022. https://vm.fi/digitaalisen-henkilollisyyden-hanke.

WakaTime 2020. WakaTime 2019 Programming Stats. Hakupäivä 14.11.2022. https://wakatime.com/blog/40-wakatime-2019-programming-stats.

Zippia 2022. 40 Fascinating Mobile App Industry Statistics [2022]: The Success Of Mobile Apps In The U.S. Hakupäivä 7.11.2022. https://www.zippia.com/advice/mobile-app-industry-statistics/.