Harri Kaikkonen

Natiivi-, web- ja hybridisovelluskehityksen vertailu sekä mobiilisovelluksen kehittäminen

Insinööri, (AMK)
Tieto- ja viestintätekniikka
Kevät 2021

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Kaikkonen Harri

Työn nimi: Natiivi-, web- ja hybridisovelluskehityksen vertailu sekä mobiilisovelluksen kehittäminen

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: Ohjelmistokehitys, mobiilisovellus, natiivi, web sovellus, hybridisovellus

Tämä insinöörityö tehtiin Tieto-Oskari Oy:n toimeksiantona. Tieto-Oskari Oy:llä on entuudestaan kehitetty ELLI-työajanseurantajärjestelmä, joka helpottaa työajan seuraamista nykyaikaisilla metodeilla. ELLI-ohjelmisto kuitenkin kaipasi päivitystä, joten järjestelmän uuden version kehittäminen aloitettiin yhdessä projektitiimin kanssa. Projektiin kuului myös tehdä työajanseurannasta mobiiliversio.

Työn alussa vertailtiin eri mobiilisovelluskehitykseen liittyviä ohjelmointiympäristöjä sekä niiden hyviä ja huonoja puolia eri käyttötarkoituksissa. Projektin alkuvaiheen edetessä pidettiin joka viikko palavereita, joissa suunniteltiin projektin aikataulua sekä etenemisvaiheita. Projektin määritysten ja tavoitteiden suunnittelun jälkeen aloitettiin ohjelmistoon liittyvä ohjelmointityö yhdessä projektitiimin kanssa.

Ohjelmointityöhön liittyi ELLIn lisäksi hallintaohjelma EMANin uuden version kehitys. Opinnäytetyöhön liittyvät työtehtävät projektissa oli kehitysympäristöjen vertailu, ELLI mobiilisovelluksen kehittäminen, siihen liittyvä frontend- sekä backend-ohjelmointi ja mobiilisovelluksen UI- sekä UX-suunnittelu.

Opinnäytetyö toteutettiin keräämällä tietoa internetistä, projektin alussa tehdyistä dokumentaatioista sekä osittain omia havaintoja hyödyntäen. Internetistä poimitut lähteet ovat pääasiassa kehitysympäristöjen tarjoajien dokumentaatioita sekä muutamia asiantuntijoiden kirjoittamia artikkeleita. Insinöörityöstä voi olla hyötyä henkilöille, jotka tarvitsevat tietoa mobiilisovelluksen kehittämiseen.

KAMK • University of Applied Sciences

Abstract

Author(s): Kaikkonen Harri

Title of the Publication: Comparison of Native, Web and Hybrid Application Development and Developing a Mobile Application

Degree Title: Bachelor of Engineering, Information and Communication Technologies

Keywords: software development, mobile application, native, web application, hybrid application

This thesis was commissioned by Tieto-Oskari Oy. Tieto-Oskari Oy has already developed the ELLI working time monitoring system, which facilitates the monitoring of working time with modern methods. However, the ELLI software needed to be updated, so the development of a new version of the system was started together with the project team. The project also included making a new mobile version of ELLI.

At the beginning of the work, the programming environments related to different mobile application development were compared, as well as their pros and cons for different uses. As the initial phase of the project progressed, weekly meetings were held to plan the project schedule and progress phases. After planning the project definitions and goals, the programming work related to the software was started together with the project team.

In addition to ELLI, the programming work included the development of a new version of the management program EMAN. The author's tasks in the project were to compare development environments, to develop the ELLI mobile application, the related frontend and backend programming, and the UI and UX design of the mobile application.

The thesis was carried out by collecting information from the Internet, from the documentation made at the beginning of the project, and partly by utilizing own observations. Sources extracted from the Internet are mainly documentation from development environment providers, as well as a few articles written by experts. Engineering work can be beneficial for people who need knowledge to develop a mobile application.

Alkusanat

Opinnäytetyö sisältää jonkin verran termejä ja käsitteitä, jotka voivat vaatia tietämystä ohjelmistokehityksestä. Opinnäytetyön tarkoitus ei ole avata kaikkia termejä yksityiskohtaisesti, vaan vertailla kehitysympäristöjä ja kertoa mobiilisovelluksen kehittämisestä.

Haluan kiittää opinnäytetyöni ohjaajaa Asko Kinnusta, kielellisestä ohjauksesta Eero Soinista, kaikkia projektissa mukana olleita, sekä eritoten Lauri Väätäistä ja Mikko Luiroa teknisestä avusta projektin yhteydessä.

Sisällys

Symboliluettelo

API	Application Programming Interface, ohjelmointirajapinta
APK	Android Package, Android sovelluksen pakettitiedosto
CD	Continuous deployment, ohjelmiston julkaisemiseen tarkoitettu automatisointi
CI	Continuous integration, menetelmä useiden koodimuutosten yhdistämiseen
CLI	Command Line Interface, komentoliittymä
cross-platform	Sovellus, joka on toteutettu siten, että se toimii usealla eri alustalla
design framework	Muotoilukehys, runko, jonka päälle käyttöliittymää voidaan kehittää
framework	Ohjelmistokehys, runko, jonka päälle sovellus voidaan kehittää
hybridisovellus	Ohjelma tai sovellus, joka toimii web-selaimessa sekä mobiilisovelluksena
hardware	Tietokoneen tai puhelimen fyysiset laitteet
IDE	Integrated Development Environment, ohjelmointiympäristö
JSX	JavaScript XML, syntaksilaajennus JavaScriptiin
natiivi	Laitteen, alustan tai ohjelman oma kehitysympäristö tai ohjelmointikieli
NDK	Native Development Kit, mahdollistaa C ja C++ käytön Androidissa
NodeJS	JavaScript koodin ajoympäristö
NPM	Node Package Manager, JavaScript-ohjelmointikielen paketinhallinta
open web	Avoin verkko kaikille käyttäjille
routing	Reititysalgoritmi, tiedonsiirron osa, jolla valitaan datapakettien määränpää

SDK	Software Development Kit, ohjelmistokehityksen työkalupaketti
SPA	Single-page application, sovellus, joka renderöi yhtä web-sivua dynaamisesti
SQL	Structured Query Language, kyselykieli tietokantajärjestelmiin
UI	User Interface, käyttöliittymä
UX	User Experience, käyttäjäkokemus
web-service	www-sovelluspalvelu, mahdollistaa tietoliikenteen verkon yli

1 Johdanto

Tieto-Oskari Oy on vuonna 1991 perustettu teknologia-alan yritys, joka suunnittelee sekä valmistaa elektronisia laitteita ja ohjelmistoja eri puolille maailmaa. Tämän insinöörityön tavoitteena oli vertailla erilaisia mobiilisovelluskehitykseen liittyviä alustoja, kertoa mobiilisovelluskehityksestä työnantajan projektin yhteydessä ja myöhemmässä vaiheessa kehittää mobiilisovellusta.

Mobiililaitteiden yleistyessä mobiilisovellusten kehityksestä on tullut todella tärkeä osa ohjelmistokehitystä ja tulevaisuudessa yhä enenevissä määrin erilaisia palveluita siirtyy mobiilikäyttöön.

Mobiilisovelluskehityksessä keskeisessä roolissa ovat eri valmistajien mobiililaitteet, käyttöjärjestelmät sekä sovelluksen käyttötarkoitus. Suunnittelussa on otettava huomioon käyttäjät, toimivuus, yhteensopivuus raudan kanssa sekä oikea ohjelmointiympäristö.

Opinnäytetyö koskee Tieto-Oskari Oy:n kehittämää työajanseurantajärjestelmää ELLIä, johon on tarkoitus kehittää uusi versio EMAN-hallintaohjelmasta sekä mobiilileimauksesta.

ELLI työajanseurantajärjestelmä mahdollistaa saldokertymän sekä työajanseurannan helposti ja nykyaikaisella tavalla. Yritysjohto ja taloushallinto saa raportteja eri henkilöiden työajasta erilaisiin tarpeisiin. Kustannusarvioiden tekeminen helpottuu, kun käytössä on todellista tietoa työmääristä. [1]

ELLI on leimauksiin perustuva tarkka ja tasapuolinen työajanseuranta, jolla voi pitää myös lukua myös joustoista ja liukumista. Järjestelmä soveltuu myös vuorotyötä ja epäsäännöllistä työaikoja tekeville yrityksille, koska ELLIn avulla voi suunnitella työntekijöille työvuoroja, jotka voivat poiketa normaaleista säännöllisistä jaksoista. [1]

Järjestelmä erittelee työajat palkkalajeittain. Järjestelmällä voi seurata työaikaa monipuolisesti projekti-, kustannuspaikka-, työvaihe- ja työnumerokohtaisesti. Työnantaja näkee paikallaolijat reaaliajassa, mikä helpottaa työn suunnittelua. [1]

ELLI-työajanseuranta sopii lähes kaikille toimialoille. Järjestelmä on suosittu toimistoissa, tuotantolaitoksissa ja tehtaissa sekä ammattiliitoissa ja kunta-alan palveluntarjoajilla. ELLI työajanseurannassa on runsaat laajennettavuusmahdollisuudet ja raportointityökalut. [1]

Työssä tutkittiin natiivialustojen ja cross-platform kehityksen eroavaisuuksia, hyötyjä ja haittoja mobiiliohjelmoinnissa. Työssä täytyi huomioida toimivuus web-serverin kanssa, ohjelmointikieli,

päivitettävyys, ylläpito ja esimerkiksi toimivuus itse fyysisten laitteiden kanssa. Myös sovelluksen saaminen Googlen Play -kauppaan sekä Applen AppStoreen oli yksi kriteereistä. Tutkimusta aloitettiin tekemään neutraaleista lähtökohdista ja niissä otettiin huomioon vaatimukset, jotka mobiilisovelluksen tulisi täyttää.

2 Teoriataustan esittely

Tässä opinnäytetyössä keskeisessä roolissa ovat eri kehitysympäristöt, kuten Android Studio, Xcode ja mobiilisovelluskehitykseen liittyvät ohjelmointityökalut sekä -kirjastot. Opinnäytetyössä mainitaan myös muutama eri ohjelmointikieli, kuten JavaScript, Swift ja TypeScript.

Opinnäytetyön teko alkoi eri kehitysalustojen ja niiden ominaisuuksien vertailulla.

2.1 Yleisesti mobiilisovelluskehityksestä

Moni palveluntarjoaja haluaa nykyään tarjota palvelujaan myös mobiililaitteilla. Näistä hyvinä esimerkkeinä voivat toimia esimerkiksi verkkopankkien mobiiliversiot, sosiaaliset mediat sekä erilaiset suoratoistopalvelut. Teknologian kehittyessä maailmalla mobiililaitteista on tullut yksi tärkeimpiä teknologialaitteita, joita voi omistaa. Mobiilisovellukset siis tuovat huomattavia käyttäjämääriä eri palveluntarjoajille.[2]

Mobiilisovelluskehityksen voi jakaa karkeasti kolmeen ryhmään: natiivikehitys, web-sovelluskehitys sekä hybridikehitys (kuva 1). Natiivikehityksellä tarkoitetaan ohjelmointia, joka suoritetaan eri laitteiden valmistajien tarjoamilla kehitysympäristöillä, kuten Android Studiolla tai Xcodella. Cross-platform kehityksellä tarkoitetaan taas sitä, että ohjelma kehitetään erillisellä ohjelmointiympäristöllä ja joka toimii usealla laitteella. Cross-platformin yleisimmät toteutustavat ovat niin sanotut hybridisovellukset sekä Mobile Web-sovellus, eli serveripuolen teknologialla kehitetty ohjelma, jota useat eri laitteet voivat käyttää.[3]

Kuva 1. Mobiilisovellusten ominaisuuksia

2.2 Natiivisovellukset

Kun puhutaan natiivisovelluksista mobiilikehityksessä, tarkoitetaan sovelluksia, jotka on kehitetty tietyille laitteille niiden valmistajien tarjoamilla työkaluilla. Yleisimmin nämä ovat Android- tai iOS-käyttöjärjestelmiä käyttävät laitteet. Androidin oma kehitysympäristö on Android Studio, ja iOS-laitteille vastaava ympäristö on Xcode. Android Studio käyttää pääasiassa Java- ja Kotlin-ohjelmointikieliä, kun taas Xcode käyttää Swift-kieltä. Nykyään myös esimerkiksi Xamarin Native ja React Native mielletään natiivisovelluskehitykseksi, vaikkakin ne ovat vain kehyksiä tai kirjastoja, jotka sallivat natiiviosien käytön. Xamarinin käyttää C# kieltä, jolla on tehty yksi koodipohja, jota jokainen alusta käyttää. React Native on samankaltainen kuin Xamarin, mutta käyttää pääasiassa JavaScript-kieltä ja React-ohjelmointikirjastoa.

Kehitysympäristöjä on nykyään laaja valikoima, joista valikoitui tähän vertailuun yllä mainitut ympäristöt. Muita mainitsemisen arvoisia ympäristöjä on AppCode iOS/macOS-kehitykseen, NativeScript Android- sekä iOS-alustoille ja Qt. [2]

Natiivisovellusten hyödyt ja haitat voisi summata siten, että natiivisovellusten suorituskyky mobiilaitteen kanssa on tehokkain ja itse sovellus toimii hyvin laitteen raudan (hardware) ja ohjelmointirajapintojen (API) kanssa (kuva 2).

Turvallisuuden kannalta natiivisovellus on turvallisempi kuin esimerkiksi web-sovellus, jonka tietoturva perustuu selaimen tietoturvateknologioihin, kuten JavaScriptiin, HTML:ään sekä CSS:ään, kun taas natiivisovelluksessa käyttäjän data tallennetaan paikallisesti mobiilaitteelle.

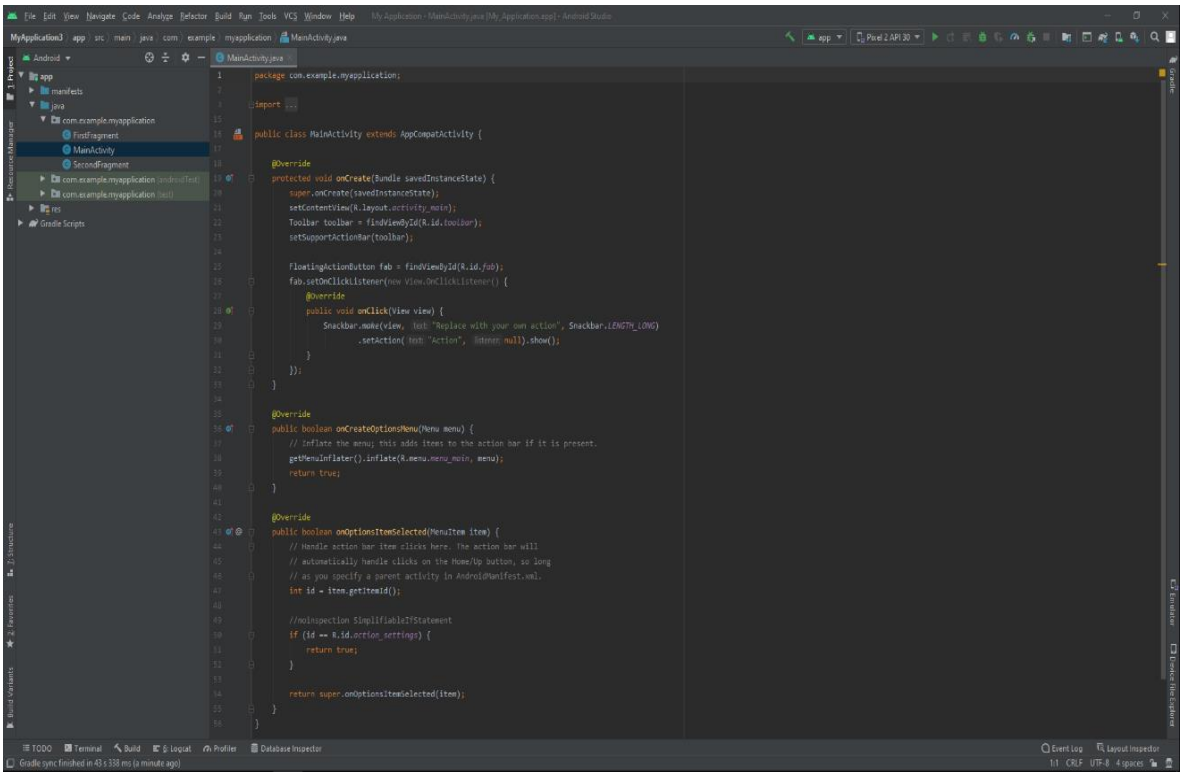
Natiivisovelluksen haittoja on muun muassa se, että natiivisovelluskehitys vaatii monen eri ohjelmointiympäristön sekä ohjelmointikielen osaamista. Hyvinä esimerkkeinä ovat Android Studio, joka käyttää pääasiassa Java- ja Kotlin-kieliä, sekä Xcode, jonka pääasiallinen ohjelmointikieli on Swift. Kehitys on hitaampaa, koska joka laitteelle täytyy tehdä oma sovellus eri ohjelmointikielellä. Tämä saattaa aiheuttaa myös ylimääräisä kuluja työtuntien sekä työntekijöiden palkkojen kanssa. [4]

Kuva 2. Natiivisovelluksen hyötyjä [5]

2.2.1 Android Studio

Android Studio on Googlen kehittämä Android-käyttöjärjestelmien virallinen kehitysympäristö (Integrated Development Environment, IDE) ja se käyttää Java- tai Kotlin-ohjelmointikieliä. Android Studio esiteltiin alun perin vuonna 2013. Vuonna 2014 se siirtyi korvaamaan Eclipse- ohjelman Android-mobiilisovelluskehityksessä.

Android Studio on kehitysalusta, joka perustuu IntelliJ Idea-koodieditoriin ja kehittäjätyökaluihin (kuva 3). Android Studio mahdollistaa useiden hyödyllisten ominaisuuksien käytön, muun muassa useiden eri mobiililaitteiden emulaattorin (kuva 4), joilla voi testata mobiilisovellusten toimintaa virtuaalisesti, C++- ja NDK (Native Development Kit) -tuen, jotka mahdollistavat C- ja C++-koodin käytön Androidissa sekä sisäänrakennetun Google Cloud Platformin, jonka avulla voi käyttää Googlen pilvipalveluita, kuten Googlen Firebase-tietokantaa ja Googlen sovellusmoottoria (App Engine). [6]



Kuva 3. Android Studion perusnäkymä. Vasemmalla on projektin kansioita, ja niiden sisällä työtiedostot.

Kuva 4. Android Studion virtuaalilaitteen emulaattori

Android Studiolla kehitettyjen sovellusten julkaiseminen Google Play -sovelluskauppaan vaatii Play kaupan kehittäjätunnuksen (Play Developer Account) luomisen, jonka jälkeen sovelluksen julkaiseminen kauppaan on kohtalaisen vaivatonta. Google Play -kaupalla on kolme vaatimusta, otsikko, lyhyt kuvaus sekä pitkä kuvaus. Lisäksi sovellukselle valitaan kategoria, sovelluskuvake ja muutama kuva sovelluksen toiminnoista. Valinnaisia asetuksia ovat esimerkiksi kielimäärittely, kielen kääntäminen sekä hinta, jos sovellus on tarkoitus laittaa myyntiin. Lopuksi sovelluksen tiedoissa täyty olla yhteystiedot, jotta mahdolliset vikailmoitukset ja muut käyttäjien yhteydenotot tulevat oikealle henkilölle, sekä yksityisyyskäytänteet (Privacy Policy), jos kehitetty sovellus tarvitsee henkilökohtaista, tai muuten arkaa dataa käyttäjiltä. [7]

Itse sovelluksen lataaminen kauppaan tapahtuu luomalla Android Studio-projektista Android-sovelluspakettitiedoston (Android package kit, APK), jota Android-käyttöjärjestelmä käyttää asentaakseen sovelluksen. Tätä pakettia käytetään, kun Android Studio-projektista tehdään julkaisu (release). Releasen jälkeen sovelluksen tulisi olla listattuna Google Play -kauppaan, mutta se ei vielä ole ladattavissa käyttäjille. Google Play -kaupassa on esikatselunäkymä, josta näkee varoituksen, jos kehittäjä unohtanut jotain. Samassa näkymässä voi myös tehdä niin sanotun rolloutin,

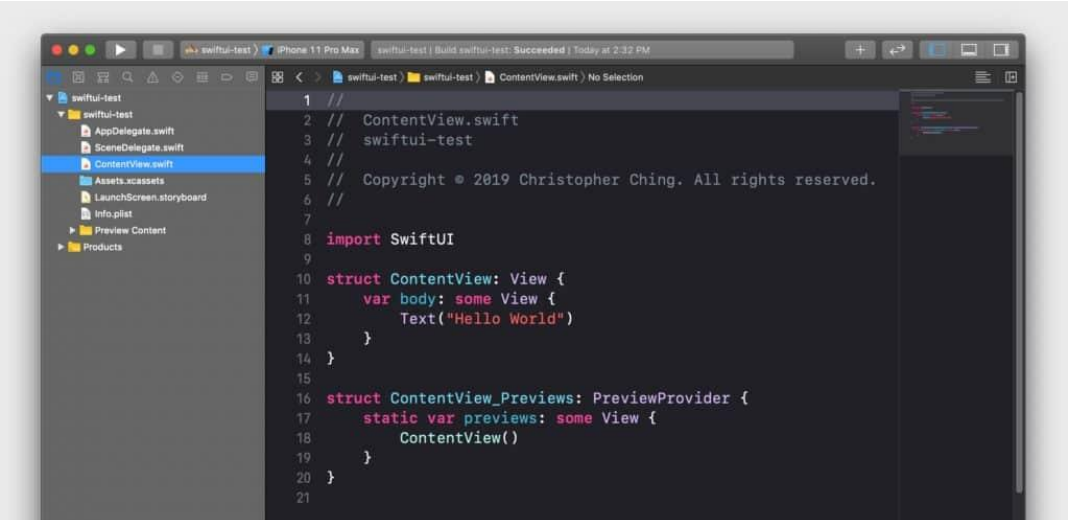
joka julkaisee sovelluksen kaikille käyttäjille valitsemiisi kohdemaihin. Täytyy kuitenkin muistaa, että ilman mainostamista sovellus todennäköisesti päätyy vain pienen käyttäjäryhmän käyttöön.

Muun muassa Netflix, LinkedIn ja WPS Office on kehittänyt sovelluksensa Android Studiolla. [8][9]

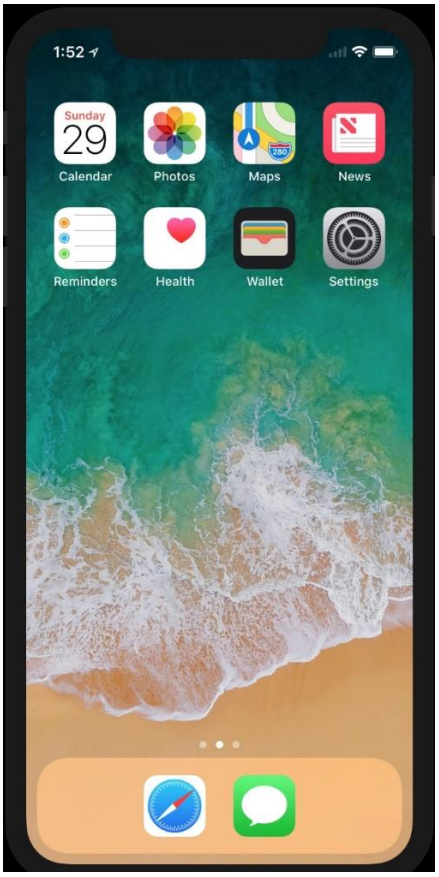
2.2.2 Xcode

Xcode on Applen kehittämä virallinen kehitysympäristö (IDE) Applen laitteille ja käyttöjärjestelmille (kuva 5). Xcode toimii vain ja ainoastaan Applen omalla macOS-käyttöjärjestelmällä. Xcode käyttää pääasiallisena ohjelmointikielenään Switftiä, mutta siitä löytyy tuki monelle muullekin ohjelmointikielelle, kuten Python, C, AppleScript, C++, Objective-C, Objective-C++ ja Java. Xcoden voi myös asentaa virtuaalitietokoneelle, jossa käyttöjärjestelmänä on macOS. Täytyy kuitenkin muistaa, että ilman virallista Applen valmistamaa laitetta voi olla haastavaa saada sovellus Applen AppStoreen.

Kuten Android Studiolla, Xcodella on käyttäjäystävällinen ja selkeä käyttöliittymä. Ohjelma sisältää kattavan valikoiman dokumentaatioita sekä työkaluja, joiden käyttö mahdollistaa toimivan ja graafisesti näyttävän sovelluksen luomisen nykyaikaisilla menetelmillä. Xcodella on mahdollista kehittää mobiilisovellusten lisäksi macOS-ohjelmia, Java-appletteja (selaimessa suoritettava Java-prosessi) [10] sekä komentoriviohjelmia (Unix). Kuten Android Studiossa, myös Xcodessa on mahdollisuus käyttää emuloitua virtuaalilaitetta koodin testaamiseen (kuva 6).



Kuva 5. Xcode-perusnäkymä. Näkymä on hyvin samankaltainen kuin Android Studiossa [11]

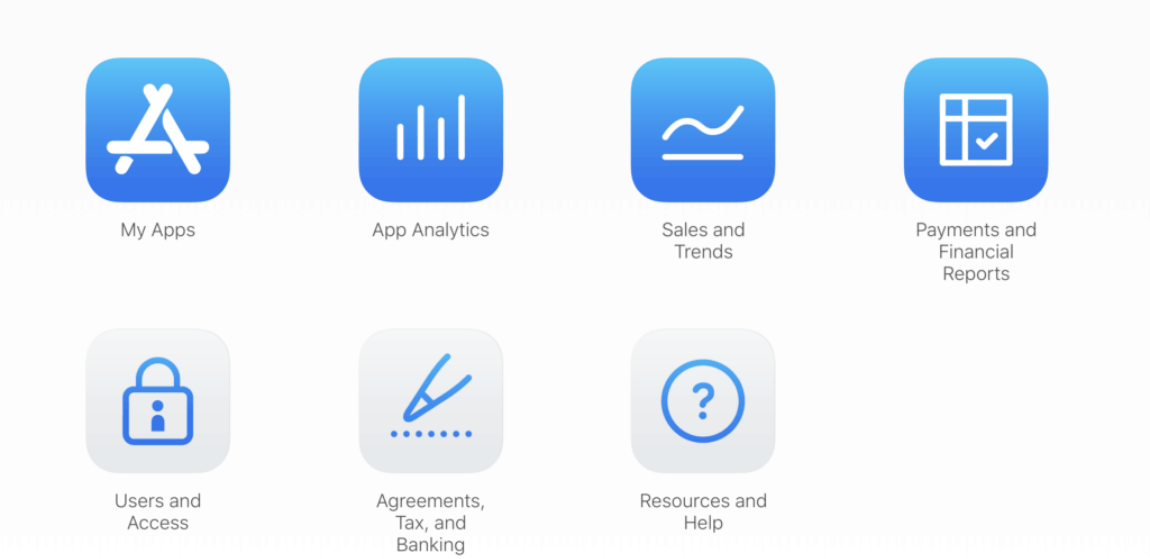


Kuva 6. Xcoden virtuaalilaite

Sovelluksen julkaiseminen Applen AppStoreen on hankalampi prosessi, kuin esimerkiksi sovelluksen saaminen Google Play -kauppaan, koska Apple on tarkempi sovelluksista, jotka pääsevät App Storeen. Tällä Apple varmistuu siitä, että sovellukset ovat varmasti laadukkaita, yhteensopivia Applen laitteiden kanssa ja että ne palvelevat käyttäjiä Applen standardien mukaisesti.

Mobiilisovelluksen julkaisemisen ensimmäinen vaihe on ilmoittautua Applen kehittäjäohjelmaan (Apple Developer Program), jonka kautta on mahdollista jakaa sovellusta AppStoren kautta jopa miljardeille käyttäjille ympäri maailmaa. Apple Developer Programissa on useita ominaisuuksia, kuten beta-testaus jopa 10,000 testaajalla, sekä sovelluksen analysointityökalu, jolla voi tarkastella sovelluksen tuottoja tai esimerkiksi käyttäjämääriä. Apple Developer Program maksaa 99 Yhdysvaltain dollaria (noin 83 €) vuodessa ja jäsenyys loppuu automaattisesti, jos vuosittaista maksua ei makseta. [12]

Apple Developer Programiin liittymisen jälkeen pääsee käsiksi Apple Store Connect -nimiseen nettisivuun, joka toimii hallintasovelluksena sovelluksen kehittäjän ja AppStoren välillä (kuva 7).



Kuva 7. App Store Connectin näkymä [13]

Applella on tarkat vaatimukset, jotka sovelluksen tulee täyttää ennen kuin se hyväksytään AppStoreen. Aluksi täytyy testata, ettei sovelluksessa ole bugeja eikä se kaadu. Lisäksi sovelluksen tiedot sekä kehittäjän yhteystiedot tarkistetaan, että ne pitävät paikkansa. [13]

Mobiilisovelluksen täytyy olla turvallinen siten, että sovelluksen käyttäjien mobiililaitteet eivät vaurioidu ohjelmallisesti tai fyysisesti, eikä sovellus saa sisältää haitallista materiaalia, kuten tarkoituksella iljettävää, vastenmielistä, laitonta tai muuten vain huonon maun vastaista sisältöä. [13][14]

Sovellukselle täytyy myös suorittaa erilaisia testauksia, ennen kuin sen voi lähettää AppStoreen tarkastettavaksi. Ensimmäiseksi täytyy varmistaa, että sovellus on varmasti toimiva, eikä se sisällä bugeja tai kaadu sovellusta käytettäessä. Tähän testiin sisältyy myös väliaikaisten ja kehitysvaiheessa tarvittujen koodien poistaminen sekä backendin toimivuus. Applen tarkastukseen tarvitsee myös demokäyttäjän, jolla tarkastajat voivat käyttää sovellusta kokonaisuudessaan sekä kuvia sovelluksen käyttöliittymästä sekä toiminnasta. Kehittäjän täytyy olla varma toimivuudesta tässä vaiheessa, koska Applen tarkastuvaiheessa beta-testaus pitää olla suoritettu ja sovellus tulee takaisin korjattavaksi, jos se kaatuilee tai siinä on huomattavia ohjelmointivirheitä. [13][14]

Jos mobiilisovellus on tarkoitus laittaa AppStoreen myyntiin, eikä vain ilmaisjakeluun, täytyy kehittäjällä olla dokumentaatio bisnesmallista. Applen täytyy tietää tarkasti, miten rahastaminen suoritetaan sovelluksen myynnissä sekä mahdollisissa sovelluksen sisäisissä myyntitilanteissa,

joissa käyttäjä voi ostaa esimerkiksi lisää sisältöä sovellukseen. Näiden tietojen puuttuminen voi pitkittää tarkastusta tai jopa aiheuttaa tarkastuksen hylkäämisen. [13][14]

Apple on myös hyvin tarkka sovelluksen UI- ja UX-suunnittelusta sekä sovelluksen laillisuudesta. Sovelluksen täytyy olla helppokäyttöinen, viimeistelty ja sen pitää täyttää laillisuusvaatimukset kaikissa sijainneissa, jonne sovellusta jaetaan. Sovellus ei myöskään saa kannustaa rikoksiin tai muuhun haitalliseen toimintaan.[13][14]

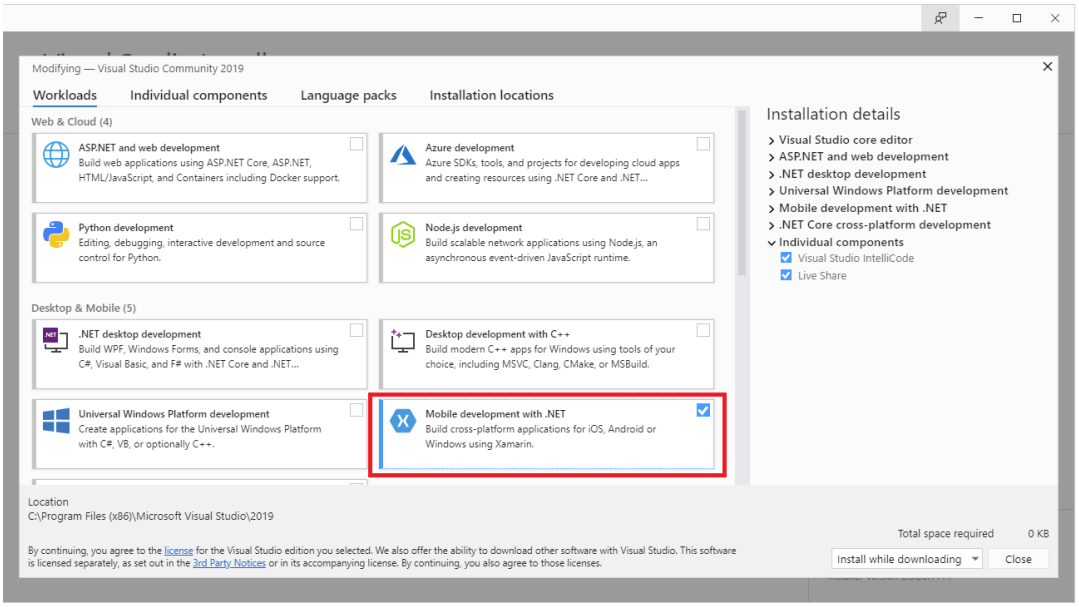
2.2.3 Xamarin.Native

Xamarin on Microsoftin tytäryhtiö, jonka pääkonttori sijaitsee San Franciscossa. Xamarinin on laajennus .NET ohjelmointialustalle, ja sen ohjelmointikielenä on C#. Laajennus tuo .NET alustaan uusia työkaluja sekä kirjastoja, jotka mahdollistavat sovelluskehityksen esimerkiksi Android-, iOS- ja Windows-käyttöjärjestelmille.[15]

Xamarinin voi asentaa Visual Studioon lisäosana (kuva 9) ja Xamarin.Nativen työkalulla voi kehittää natiivisovelluksia, joiden backendissä pyörii sama C# -kielellä tehty koodi, mutta UI- ja UX-ominaisuudet täytyy tehdä joka laitteelle erikseen. Jaettu C# koodi siis mahdollistaa jaetun ohjelmalogiikan, tietokantayhteydet, web-palveluiden (web service) kutsumisen ja backend-yhteyden. Xamarin.Nativella kehitetyn sovelluksen jaetun koodin osuus on noin 75 %.[15]

Xamarin.Nativen jaetun koodin ansiosta ohjelmiston päivittäminen on helpompaa kuin erikseen eri alustoille kehitettyjen ohjelmistojen päivitys, koska kehittäjän täytyy päivittää vain yksi ohjelmistokoodi. Tosin Xamarinilla tehdyt natiiviosat täytyy päivittää erikseen joka laitteelle, mutta se on silti huomattavasti pienempi työ, kuin koko koodipohjan päivittäminen eri laitteille.

Xamarin.Nativella kehitetyt sovellukset voi listata Google Play -kauppaan tai Applen AppStoreen, jos sovellus täyttää kauppojen vaatimukset. Kauppojen vaatimuksista kerron niiden tutkimisosuudessa, kohdat 2.2.1 ja 2.2.1 [16]



Kuva 8. Xamarinin käyttöönotto Visual Studio -ohjelmointiympäristössä asennuksen yhteydessä [19]

2.3 Web-sovellukset

Web-mobiilisovellus on web-teknologioilla kehitetty web-sivu, jota käytetään puhelimessa selaimen kautta. Web-kehitys suoritetaan pääsääntöisesti HTML-, CSS- ja JavaScript-kielillä, joilla kehitetään web-sivu, joka sisältää sovelluksen tarvitsemat toiminnot (kuva 10).

Mobiili web-sovelluksen kehittäminen onnistuu millä tahansa koodieditorilla, mutta esimerkiksi Visual Studion tai Sublimen käyttö on suositeltavaa, koska niihin on helppo ottaa käyttöön tarvittavat JavaScript-kirjastot, joita yleensä web-sovelluskehityksessä käytetään. Jos haluaa käyttää web-sovelluksen kehittämiseen Java-ohjelmointikieltä, on hyviä vaihtoehtoja koodieditoriksi Eclipse ja Netbeans, koska niissä on käytännöllisiä työkaluja web-sovelluskehitykseen. Eclipsen vahvuuksia on suurten projektien hallinta, debuggaus-ominaisuudet sekä analysointityökalut, kun taas Netbeansin vahvuuksiin kuuluu eri tietokantojen tuki, kuten esimerkiksi Java DB, MySQL ja Oracle tietokannoille.

Web-mobiilisovellusten hyötyjä on muun muassa yhteensopivuus eri laitteiden kanssa, koska eri koodipohjia ei tarvita. Tämän ansiosta kehityksessä tulevat kulut ovat pienemmät, kuin esimerkiksi natiivisovelluksissa, joissa tarvitsee eri ohjelmointikielin osaamista sekä tämän takia mahdollisesti enemmän ohjelmoijia.

Web-mobiilisovelluksen päivittäminen on myös helpompaa, koska ei tarvitse tehdä päivityksiä useisiin eri koodeihin. Riittää, kun päivitykset tekee yhteen versioon.

Tämän tyyppisellä mobiilisovelluksella on kuitenkin myös huonot puolensa. Web-mobiilisovellus ei esimerkiksi voi käyttää mobiililaitteiden ominaisuuksia tehokkaasti. Näitä ominaisuuksia on muun muassa kamera, sijainti sekä bluetooth.

Web-mobiilisovelluksilla on rajoitteita offline-toiminnoissa. Siinä missä natiivisovellus voi toimia mobiililaitteessa täysin offline-tilassa, web-sovellus on toiminnaltaan riippuvainen internetyhteydestä, lukuun ottamatta välimuistiin tallennettuja tietoja, joita voi käyttää offline-tilassa. Näillä tiedoilla tosin on mahdollista käyttää sovellusta vain hyvin rajallisesti. Täytyy kuitenkin muistaa, että natiivisovelluksessakin voi olla toimintoja, jotka vaativat internet yhteyden. [17]

Kuva 9. Web-mobiilisovellus [18]

2.4 Hybridisovellukset

Hybridisovellus on kuin natiivisovellus, mutta se on kehitetty web-teknologioilla (HTML, CSS, JavaScript). Hybridisovellus pyörii puhelimen natiiviosassa, joka käyttää Webview -kirjastoa heijastaen sovelluksen selainmoottorin kautta näytölle. Mobiililaite ei siis käytä varsinaista selainta, vaan näyttää web-sivun natiivisovelluksen kautta. Tästä syystä hybridisovellus mahdollistaa laitteen toimintojen käytön, joiden käyttö ei ole mahdollista tavallisissa web-sovelluksissa, kuten kameran sekä esimerkiksi paikallisen tietokannan käytön (kuva 11). [19]

Hybridisovelluksen kehittäminen on kustannustehokasta verrattuna esimerkiksi natiivikehitykseen. Ohjelmistokehittäjien täytyy luoda vain yksi projekti Android- ja iOS-käyttöjärjestelmille, joten eri ohjelmointikielien osaamista ja usean projektin luomisen sijaan tarvitaan vain yhtä kieltä sekä projektia. Tämä mahdollistaa henkilöresurssien tehokkaan käytön. Kehityskustannuksia laskee myös se, että itse sovelluksen kehittäminen on ajallisesti nopeampaa.

Hybridisovelluksissa on toki myös huonoja puolia, esimerkiksi suorituskyky voi olla heikompi kuin natiivisovelluksessa. Jos esimerkiksi kehittää samanlaisen sovelluksen sekä natiivina, että hybridinä, on todennäköistä, että natiivisovellus toimii laitteella paremmin. Sovelluksen täytyisi kuitenkin olla melko raskas ja sisältää monia toimintoja, jotta eron näkisi. Kuitenkin nykyajan mobiililaitteiden tehon huomioon ottaen ero on niin pieni, että käyttäjä ei todennäköisesti edes huomaisi sitä.

Sovelluksen päivittäminen ja uusien ominaisuuksien lisääminen voi olla hitaampaa kuin natiivisovelluksissa, koska aina kun Apple tai Google julkaisee uuden ominaisuuden alustoilleen, natiiviympäristöt saavat ominaisuuden käyttöönsä nopeammin kuin hybridisovelluksen kehittäjät. [20]

Muun muassa Instagram, Gmail, Twitter ja Amazon Appstore on kehitetty hybridimenetelmällä. Suosittuja kehitysympäristöjä hybridisovelluskehitykseen on muun muassa React, Ionic sekä Angular. [21][22]

Kuva 10. Hybridisovelluksen ja natiivisovelluksen ero [23]

2.4.1 ReactJs

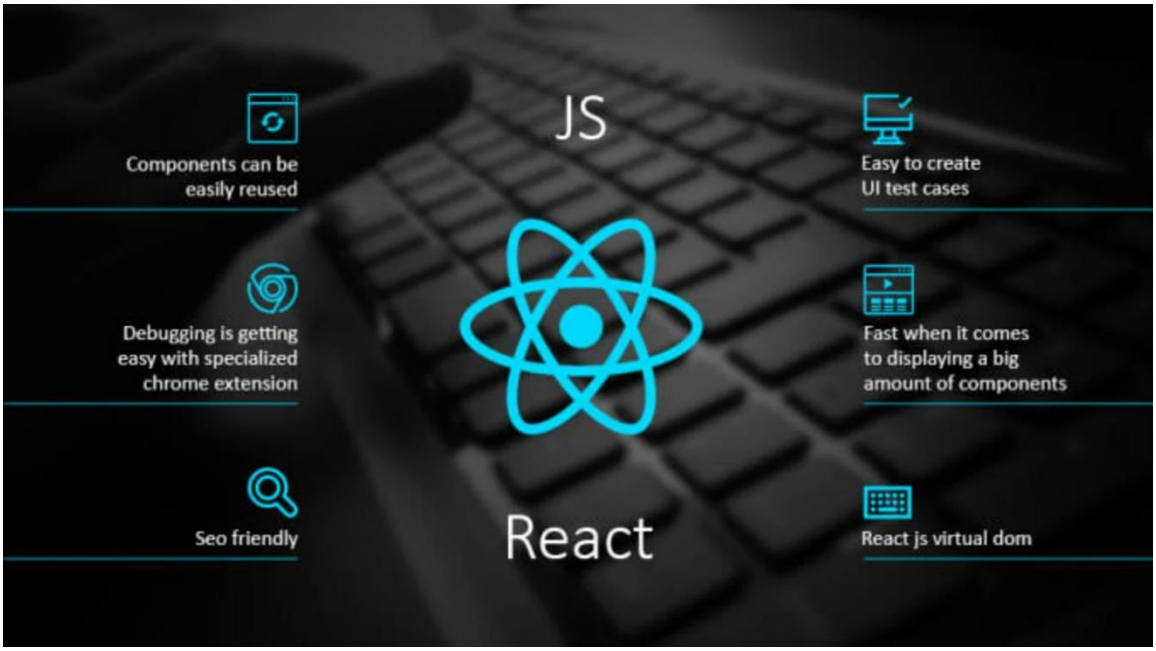
React on avoimen lähdekoodin JavaScript -kirjasto, jota käytetään web-sivujen visuaalisen kerroksen hallintaan. Reactin vahvuuksia on mahdollisuus kehittää suuriakin sovelluksia, joissa data liikkuu vaivattomasti sekä sillä on mahdollista luoda sulavasti toimiva ja skaalautuva käyttöliittymä. [25]

React käyttää normaalin JavaScript mallin sijasta JavaScript XML-mallia (JSX). JSX on laajennus JavaScript-ohjelmointikielelle ja käytännössä se muuttaa HTML-tagit React elementeiksi ja näin ollen mahdollistaa HTML-koodin kirjoittamisen Reactilla. Täytyy kuitenkin muistaa, että pelkästään JavaScriptillä ohjelmointikin on mahdollista. [25] [29]

Reactin vahvuuksia on komponenttipohjainen tiedostorakenne sekä debuggaus- ja testausominaisuudet. Tavallisen JavaScriptin käyttö tekee Reactista helposti lähestyttävän ja siinä on matala oppimiskynnys varsinkin, jos HTML- ja CSS-ohjelmointi on entuudestaan tuttua. [25]

Mobiilisovelluskehityksessä Reactin vahvuus on Reactista löytyvät natiiviohjelmointimahdollisuudet, joista kerrotaan enemmän React Native osuudessa kohdassa 2.4.2. Hybridimallissa kuitenkin on se ero natiivikehitykseen, että hybridimallissa kehitetään web-sivu, joka heijastetaan mobiililaitteiden natiivisovelluksella käyttäen esimerkiksi Android Studio webview-ominaisuutta, joka käyttää laitteen selainmoottoria, mutta ei varsinaista selainta, kuten pelkät web-sovellukset. Tämä siis mahdollistaa sen, että Reactilla luodaan yksi web-sivu ja se heijastetaan eri laitteiden natiivisovellusten kautta käyttäjälle, mutta se mahdollistaa myös laitteiden natiiviosien käytön. [25]

Kuten React Nativen käytössä, Reactin käytön voi aloittaa paketinhallintajärjestelmällä(NPM), jonka kautta React-projektin voi luoda. NPM:ä tarvitaan myös build-ympäristön asettamiseen. Toinen tapa aloittaa Reactin käyttö on tuoda (import) ReactJs -kirjasto HTML-koodiin (kuva 12). [26]



Kuva 11. ReactJS ominaisuuksia [27]

2.4.2 React Native

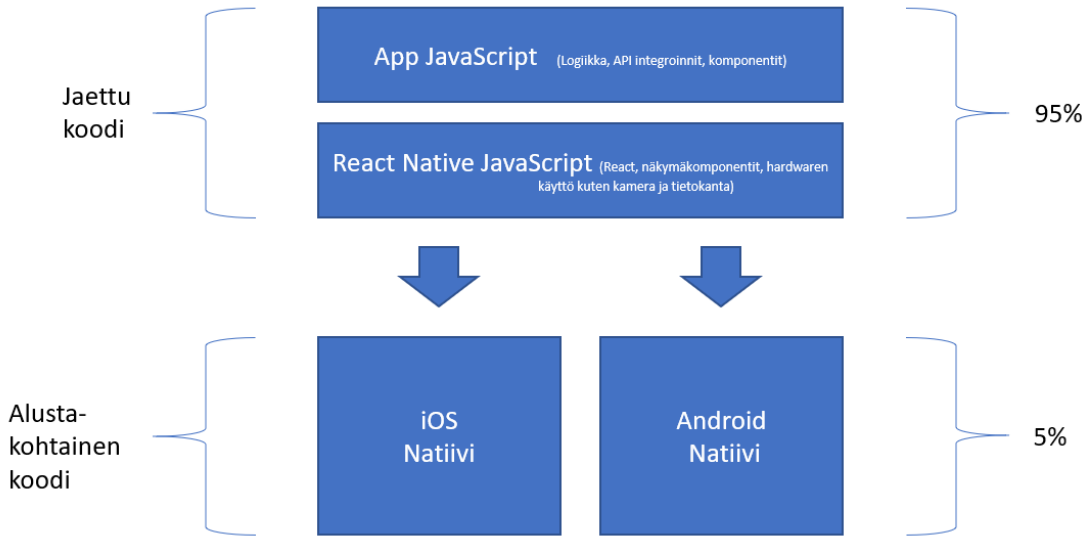
React Native on Facebookin julkaisema ohjelmointikehys Reactille (framework), joka mahdollistaa mobiilikehityksen usealle eri alustalle. Ohjelmointi tapahtuu JavaScript -kielellä ja React Native mahdollistaa melkein koko koodipohjan jakamisen esimerkiksi Android- ja iOS-käyttöjärjestelmille (kuva 13).

Kuten Reactissa, React Native mahdollistaa monipuolisten mobiili UI-kirjastojen käytön kehityksessä. React Nativen vahvuuksiin kuuluu myös laaja käyttäjäkunta, joten internetistä löytyy valtava määrä ohjeita, esimerkkikoodeja sekä dokumentaatiota. [25]

React Nativen asennukseen vaaditaan NodeJS -ajoympäristö (JavaScript runtime environment), sekä sen paketinhallintajärjestelmä NPM:n (Node Package Manager). NodeJS mahdollistaa webserverien sekä nettityökalujen (networking tools) luomisen, sekä JavaScript -koodin suorittamisen suoraan palvelimella selaimen ulkopuolella. NPM puolestaan on komentorivi asiakasohjelma (command line client), jolla voi keskustella rekisterin kanssa, ajaa komentoja komentorivillä sekä se mahdollistaa JavaScript -moduulien käyttöönoton, jotka löytyvät rekisteristä. Myös React Native projektin luomien tapahtuu komentorivin kautta. [25][26]

React Native -projektin editoinnin voi aloittaa millä tahansa koodieditorilla, joista käytetyimmät on Visual Studio sekä Xcode. Projektin luomisen yhteydessä NPM luo tarvittavat työtiedostot projektin aloittamiseen ja kehitystyön jatkuessa ohjelmoija voi muokata näitä tiedostoja sekä lisätä muita tarvittavia tiedostoja. Kun kehitystyö on siinä pisteessä, että sovellusta voi testata, voi kehittäjä käyttää esimerkiksi Android Studion emuloitua virtuaalilaitetta testaamiseen. [25][26]

React Native sovelluksen voi julkaista Google Play -kauppaan ja Applen AppStoreen, kunhan Googlen ja Applen vaatimukset sovellukselle täyttyvät.



Kuva 12. React Native ohjelmistoarkkitehtuuri

2.4.3 Angular

Angular on Googlen kehittämä tyylikehys (design framework) ja kehitysympäristö, jolla voi kehittää monimutkaisiakin yhden sivun sovelluksia (SPA), mikä tarkoittaa sitä, että sovellus tai nettisivu keskustelee käyttäjän kanssa päivittämällä sivulla näkyvää dataa dynaamisesti web-serveriltä sen sijaan, että se lataisi aina uuden sivun.

Angular on kehitysympäristö, joka käyttää ohjelmointikielenään TypeScriptiä. Angularin ominaisuuksiin kuuluu komponenttipohjainen ohjelmointikehys, joka mahdollistaa skaalautuvien web-

sovelluksien kehittämisen. Angulariin on myös laaja valikoima integroituja kirjastoja, joiden ansiosta ohjelmistokehittäjällä on käytössä suuri valikoima ominaisuuksia, kuten reititys (routing), lomakkeiden hallinta (forms management) sekä asiakkaan ja serverin välinen kommunikaatio. [30.]

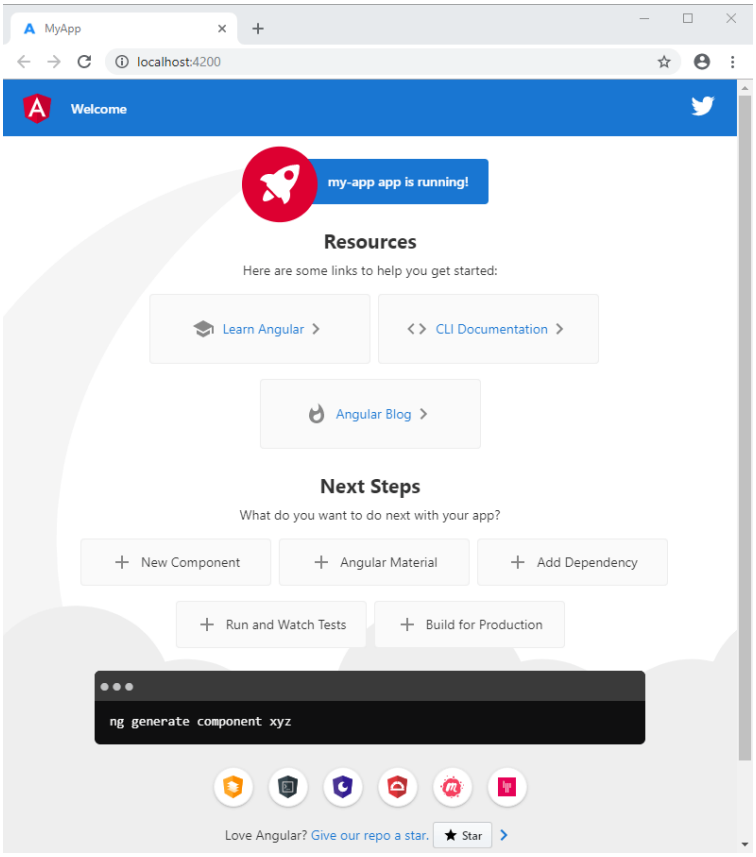
Angular on suunniteltu siten, että sillä kehitetyt web-sivut sekä sovellukset on mahdollisimman helppoa päivittää, joten uusien ominaisuuksien käyttöönottaminen on näin ollen helpompaa. Suuri ja monipuolinen käyttäjäkunta sisältää yli 1.7 miljoonaa henkilöä [30.], joten internetissä on laaja valikoima kirjastoja sekä sisällöntuottajia.

Angularin käyttöönotto on hyvin samankaltainen Reactin kanssa. Aluksi täytyy asentaa Node.js, sekä NPM. Käytännössä NPM ohjaa Angularin käyttöä ja sillä voi asentaa Angularin komentorivin (Angular CLI), luoda uuden työalustan ja projektin sekä asentaa kirjastoja.

Kun Angular CLI on asennettu, sitä voi käyttää komentorivin kautta. Angularia ohjataan ng -alkuisilla komennoilla. Esimerkiksi *ng new* -komento luo uuden projektin, *ng generate* -komento luo tai muokkaa projektin tiedostoja, ja *ng serve* -komento luo sovelluksesta koontiversion (build) sekä käynnistää sen palveluna, joka käynnistää itsensä uudelleen aina kun tiedostoja muokataan (kuva 14, kuva 15).

```
npm install -g @angular/cli
ng new my-app
cd my-app
ng serve --open
```

Kuva 13. Komentorivi komennot, joilla asennetaan Angular CLI, sekä luodaan projekti. *npm install -g @angular/cli* -komento asentaa Angular CLI:n, jossa *-g* -lippu tarkoittaa globaalia asennusta järjestelmääsi. *Ng new my-app* komento luo uuden projektin, *cd my-app* vaihtaa tiedostosijainnin kansioon, jonne uusi projekti luotiin. *Ng serve –open* komento luo sovelluksesta koontiversion, käynnistää serverin ja tekee nämä kaksi operaatiota uudelleen, jos tiedostoja muokataan. --open lippu käynnistää selaimen automaattisesti. [30]



Kuva 14. Angularin vakiosivu projektin luomisen jälkeen [31]

2.4.4 Ionic

Ionic on ilmainen avoimen lähdekoodin UI-työkalu, jolla voi kehittää laadukkaita mobiili- ja työpöytäsovelluksia käyttäen JavaScriptiä ja web-teknologioita, kuten HTML ja CSS. Ionic toimii yhteen suosittujen ohjelmointikehyksien kanssa, kuten Reactin ja Angularin, mutta sitä voi myös käyttää itsenäisesti. Kuten React ja Angular, Ionicilla kehitetyissä sovelluksissa on taustalla yksi koodipohja, jota eri laitteet käyttävät.

Ionic on rakennettu siten, että se suoriutuu hyvin uusimmilla mobiililaitteilla ja se käyttää tehokkaasti laitteiden hardwarea. Se on myös suunniteltu siten, että se toimii näyttävästi kaikilla laitteilla sekä alustoilla. Valmiit komponentit, fontit ja vaikuttava kantateema mahdollistaa kehityksen nykyaikaisen tyylikkäästi. [32]

Ionic on natiivi- ja web-tehostettu, mikä tarkoittaa sitä, että Ionic emuloi natiivisovellusten UI-ohjeistusta, sekä käyttää natiivi ohjelmistokehityspaketteja (SDK, software development kit). UI-

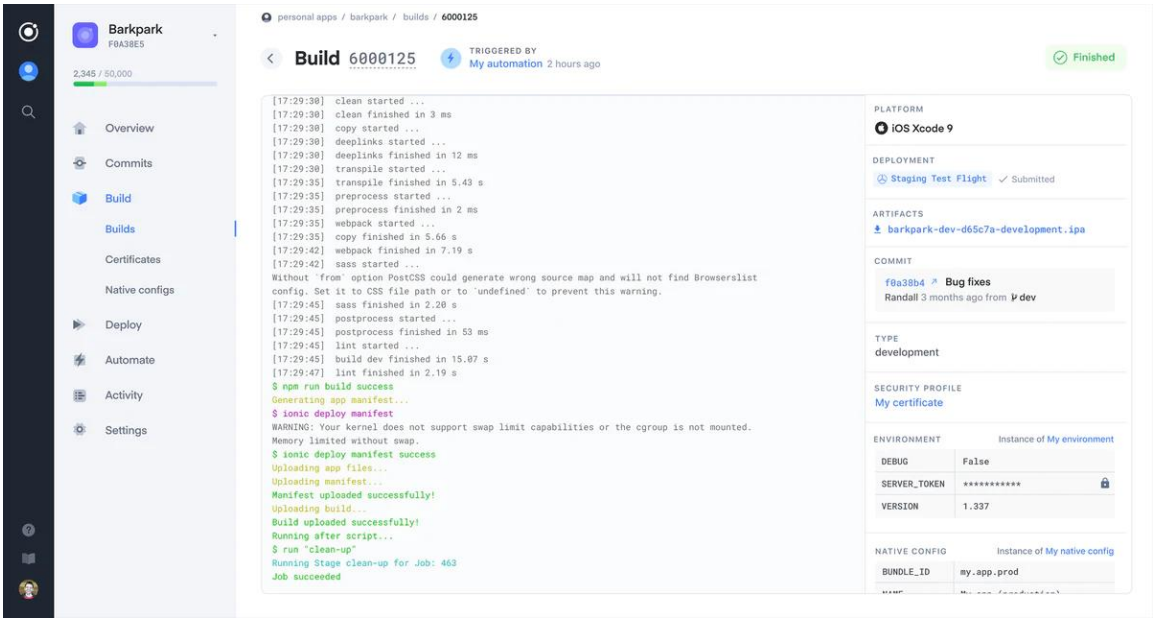
standardien, sekä laitteiden ominaisuuksien lisäksi sovellukseen yhdistyy avoin web (open web) [31.], jotka yhdessä luovat tehokkaan sekä joustavan sovelluksen.

Aikaisemmat versiot Ionicista oli sidottu tiukasti Angulariin, mutta version 4.x kohdalla Ionic ohjelmointikehystä päivitettiin siten, että se toimii itsenäisenä web-komponenttikirjastona, johon sisältyy uusimpia JavaScript kehyksiä. [32.]. Ionic web-komponentteja voi siis nykyään käyttää useimpien frontend-ohjelmointikehyksien kanssa, kuten React ja Vue.

Yksi päätavoitteista Ionicin muuttamisessa ohjelmointikehyksestä web-komponentiksi oli se, että muilta ohjelmointiympäristöiltä ei vaadita niin paljon resursseja Ionicin tueksi, vaan Ionic voi toimia itsenäisemmin web-sivuilla. [33]

Ionicilla on oma CLI (Command Line Interface), joka on komentorivityökalu. Ionic CLI:n saa asennettua käytössä olevalle laitteelle globaalisti NPM:n avulla. Tällä työkalulla voi ajaa Ionic-komentoja, kuten *ionic build,* joka luo koontiversion sovelluksesta. [33]

Ionic CLI mahdollistaa myös sovelluksen koonnin ja käyttöönoton pilven kautta, mutta tämä vaatii Appflow jäsenyyden. Appflow on jatkuva integrointi (CI, continuous integration) ja jatkuva käyttöönotto (CD, continuous deployment) alusta Ionic kehittäjille (kuva 16). Se mahdollistaa ohjelmistokehittäjien luoda koontiversioita ja jakaa sovellusta helposti sekä nopeasti. [35]



Kuva 15. Appflow näkymä [36]

2.5 Yhteenveto mobiilikehitysympäristöistä

Mobiilisovelluskehitykseen on saatavilla laaja valikoima upeita työkaluja, jotka palvelevat ohjelmistokehittäjää hyvin, jos niiden ominaisuuksiin perehtyy ja mihin tarkoitukseen mikäkin ohjelmointikehys tai alusta on optimaalinen. Opinnäytetyötä tehdessä kävi ilmi, että jo pelkästään valmistajien dokumentaatioita selaamalla pääsee syvälle eri ohjelmointityökalujen maailmaan, joten oikean ympäristön valinta on tehty siltä osin helpoksi. Toki eri projektien päämäärät voivat rajata vaihtoehtoja.

Tieto-Oskari Oy:n projektissa tärkeimpiä vaatimuksia oli yhteneväisyys Android- ja iOS-laitteiden välillä, sekä Google Play -kaupan ja Applen AppStoren vaatimusten täyttäminen. Tiettyjen toimintojen kehittäminen rajoitti kehitysympäristö valintaa, kuten offline-tilassa toiminta, käytettävyys mobiililaitteiden lisäksi myös esimerkiksi tietokoneella sekä kielen vaihtaminen englannin ja suomen välillä.

Alkuvaiheessa oli vielä epäselvää, millä ympäristöllä mobiilisovellusta lähdetään kehittämään. Aluksi käytettiin Android Studiota ja sillä kehitettiin karkea beta-versio sovelluksesta, jossa pääasiassa oli eri toimintojen alustava testaus. Vaatimusmäärittelyjen kehittyessä myös ohjelmointiympäristö sekä ohjelmointikieli alkoivat hahmottua tarpeitamme vastaaviksi.

Päädyimme käyttämään projektissa Angularia, jonka ohjelmointikielenä on TypeScript. Angularin käyttö mahdollisti tarvittavien toiminnallisuuksien kehittämisen ja hybridisovellus vastasi ideoitamme sekä vaatimuksia, joita sovelluksen olisi täytettävä. Toinen vaihtoehto oli React ja se olisi varmasti ollut yhtä pätevä vaihtoehto. Lähdimme kuitenkin pohdintojen jälkeen toteuttamaan projektin Angularilla, koska saimme siihen myös kattavan koulutuksen.

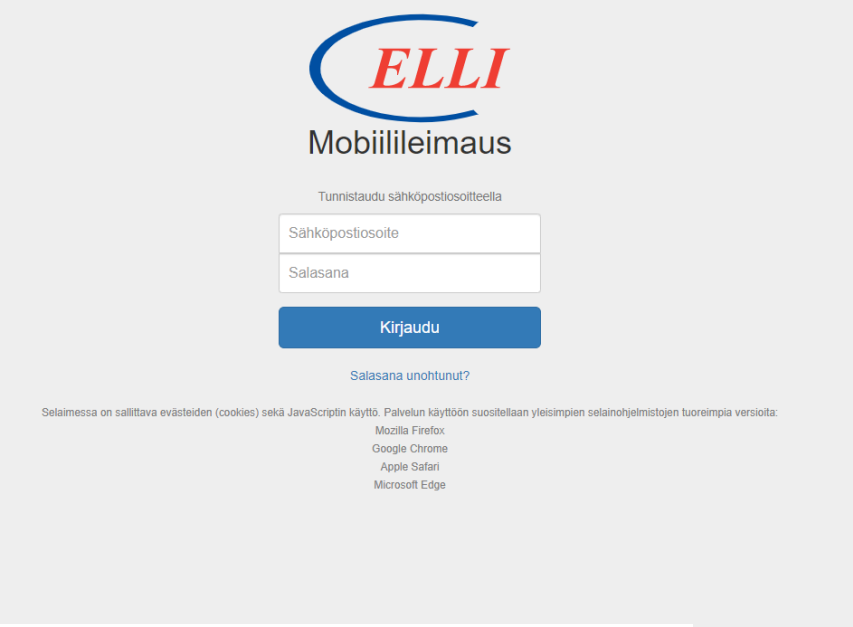
2.6 Mobiilisovelluksen kehittäminen

Tieto-Oskari Oy:n projekti ELLI työajanseurantajärjestelmän uudesta versiosta alkoi syksyllä 2020 ja jatkuu ainakin kesään 2021 asti (kuva 17). Projektin alussa pidettiin palavereita, joissa keskityttiin ohjelmiston suunnitteluun, vaatimusmäärittelyihin, ohjelmointiympäristöihin sekä ohjelmointikieliin. Työtehtäviin kuului myös selvittää eri vaihtoehtoja mobiilisovelluksen kehittämiseen ja kertoa niistä palavereissa. Projektille luotiin myös alustava aikataulu.

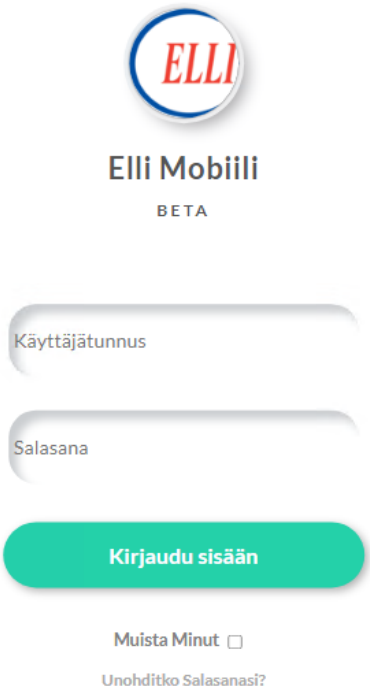
Mobiilisovelluksen lisäksi samaan aikaan alettiin kehittämään uutta versiota EMAN-hallintaohjelmasta, joka käyttää samaa backendiä mobiilisovelluksen kanssa. Opinnäytetyöhön liittyviin työtehtäviin kuului mobiilisovelluksen kehittäminen, sen UI- ja UX-suunnittelua sekä hieman backend kehitystä. Aluksi Angular oli ympäristönä tuntematon, mutta projektitiimi sai neljän päivän mittaisen koulutuksen, joka antoi hyvät valmiudet lähteä kehittämään sovellusta.

Alkukankeuden jälkeen sovelluksen kehittäminen lähti sujumaan hyvin. Uusia ominaisuuksia syntyi joka viikko ja kehityksen edetessä saatiin ideoita sekä kehityssuuntia myös käyttöliittymään. Käyttöliittymä ja ulkoasu ei kuitenkaan ollut tärkeysjärjestyksessä ensimmäisenä, vaan aluksi keskityttiin sovelluksen toiminnallisuuteen sekä backendin toimivuuteen. Oma työni backendin parissa oli tehdä muutamia tietokantakyselyitä, sekä muokata hieman SQL- tietokantaa. Pääasiassa keskityin frontendiin.

Tätä kirjoittaessa, projekti on edelleen kesken ja sitä on tarkoitus jatkaa ainakin kesäkuun loppuun saakka. Mobiilisovelluksen toiminnot ovat melkein valmiit, jonka jälkeen alamme ohjelmoimaan ulkoasua lopulliseen muotoonsa (kuva 18).



Kuva 16. Vanha ELLI-mobiililleimauksen etusivu



Kuva 17. Uusi ELLI-mobiililleimauksen etusivu

3 Pohdinta ja yhteenveto

Opinnäytetyön alussa tietämykseni mobiilikehityksestä rajoittui lähinnä Android Studioon, mutta työn edetessä tutuksi tuli muutkin natiivikehitykseen liittyvät ympäristöt sekä web-sovellus ja hybridisovellusympäristöt. Myös CSS- ja HTML-ohjelmointi olivat itselleni haastavia, koska en ollut työskennellyt niiden parissa aiemmin. Projekti oli myös ensimmäinen suurempi ohjelmointikokonaisuus, jossa olen mukana, joten olen oppinut ajattelemaan ohjelmointia ja ohjelmistokehitystä laajempana kokonaisuutena.

Työnantajani projektissa päädyttiin käyttämään Angularia, joka vastasi tarpeitamme serveri- ja web-puolen kannalta sekä sillä pystyi kehittämään joustavan ja toimivan käyttöliittymän mobiililaitteille. Angular ja TypeScript onkin tullut hyvin tutuksi projektin edetessä.

Projekti eteni kaiken kaikkiaan hyvin ja lopputulos vastaa tähän mennessä projektin alkuvaiheessa laatimaamme vaatimusmäärittelyä. Alussa tehty tutkimus mobiilisovellusten kehitysympäristöistä sujui hyvin ja siitä saimme tarpeeksi laajan kuvan eri kehitysvaihtoehdoista, jonka pohjalta pystyimme valitsemaan kehitysympäristön. Sain myös hyvän käsityksen siitä, mitä kaikkea ohjelmistokehitysprojektiin sisältyy työelämässä.

Haasteita projektissa oli muun muassa uusi ohjelmointiympäristö ja kieli, mutta alussa olleen Angular koulutuksen jälkeen pääsimme vauhtiin. Joidenkin ominaisuuksien tekeminen typescriptillä osoittautui haastavaksi, mutta lopulta saimme ne toimimaan halutulla tavalla. Tämä johtui osittain kokemuksen puutteesta Angular ja typescript ympäristön parissa.

Typescript ohjelmointikielenä on hyvin monipuolinen ja joustava, joten sen kanssa työskentely oli hyvin palkitsevaa. Tiedostorakenne on selkeä, ohjelmointisyntaksit nopeasti ymmärrettäviä ja kasvavan suosion takia internetissä on miltei loputon määrä käyttäjien laatimia ohjeita sekä ratkaisuja ongelmatilanteisiin. Huomasin, että jos koodissa ilmeni ongelmia, pikaisella netin selauksella siihen todennäköisesti löytyi ratkaisu.

Vaikka projektissa oli viikoittaisia palavereita, välillä oli tunne, että työ voisi edistyä tehokkaamminkin. Parannettavaa oli esimerkiksi vaatimusmäärittelyjen ja rautalankamallien päivittämisessä, koska projektin edetessä tuli vastaan tilanteita, joissa haluttuja ominaisuuksia ei voinut tai kannattanut tehdä alkuperäisen suunnitelman mukaan. Toki täytyy muistaa, että uutta ohjelmistoa kehittäessä on melkein mahdotonta laatia alussa koko projektin kestävä suunnitelma ja aikataulu, koska ulkoisia muuttujia ja uusia huomioita tulee jatkuvasti työn edetessä.

Lähteet

1. Tieto-Oskari Oy:n nettisivujen tuotetiedoista, https://www.tieto-oskari.fi/tuotekategoria/tyoajanseuranta/ Haettu 11.5.2021

2. Erin Gilliam Haije, Top 20 Mobile Development Tools: An Overview, https://mopinion.com/mobile-development-tools-an-overview/ Haettu 11.5.2021

3. Jim Cowart, what is a Hybrid Mobile App, https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app- Haettu 11.5.2021

4. Clearbridge Mobile, https://clearbridgemobile.com/benefits-of-native-mobile-app-development/ Haettu 11.5.2021

5. Martha Jones, Top Advantages of Native Mobile App Development for Business! https://martha-7987.medium.com/top-advantages-of-native-mobile-app-development-for-businesses-ab5c016b2e94 Haettu 11.5.2021

6. Android Studion dokumentaatio, overview, https://developer.android.com/studio/intro Haettu 11.5.2021

7. David Oragui, How to Publish an App on Google Play: A Step-by-Step Guide, https://medium.com/@the_manifest/how-to-publish-an-app-on-google-play-a-step-by-step-guide-80f9f533e370 Haettu 11.5.2021

8. Googlen dokumentaatio, Play Console Help, Create and set up your app, https://support.google.com/googleplay/android-developer/answer/9859152?hl=en#zippy=%2Cproduct-details%2Cgraphics Haettu 11.5.2021

9. Sumanta Banerjee, is there any famous apps which created in Android Studio, https://www.quora.com/Is-there-any-famous-apps-which-created-in-android-studio Haettu 11.5.2021

10. Tuntematon kirjoittaja, Java Applet Basics, https://www.geeksforgeeks.org/java-applet-basics/ Haettu 11.5.2021

11. Chris Ching, Xcode Tutorial for Beginners, https://codewithchris.com/xcode-tutorial/ Haettu 11.5.2021

12. Apple dokumentaatio, https://developer.apple.com/programs/ Haettu 11.5.2021

13. Chris Ching, How to Submit Your App to the App Store, https://codewithchris.com/submit-your-app-to-the-app-store/ Haettu 11.5.2021

14. Apple dokumentaatio, App Store Review Guidelines, https://developer.apple.com/app-store/review/guidelines/ Haettu 11.5.2021

15. Microsoft dokumentaatio, What is Xamarin? https://docs.microsoft.com/fi-fi/xamarin/get-started/what-is-xamarin Haettu 11.5.2021

16. Nigel Ferrissey, When to Use Xamarin.Forms vs Xamarin Native, https://www.telerik.com/blogs/when-to-use-xamarin-forms-vs-xamarin-native Haettu 11.5.2021

17. Tania H., Mobile App vs Mobile Website: Pros and Cons of Each Approach in 2019, https://rubygarage.org/blog/mobile-app-vs-mobile-website Haettu 11.5.2021

18. Tuntematon kirjoittaja, Differences between mobile app and website development, https://www.oclocksoftware.com/blog/differences-between-mobile-app-and-website-development Haettu 11.5.2021

19. Chris Griffith, What is Hybrid App Development, https://ionic.io/resources/articles/what-is-hybrid-app-development Haettu 11.5.2021

20. Sagar Sharma, Pros & Cons of Hybrid Mobile Apps Development, https://www.credencys.com/blog/pros-cons-of-hybrid-mobile-apps-development/ Haettu 11.5.2021

21. Thomas Wilfred, List of Top Hybrid Apps, https://codersera.com/blog/list-of-top-hybrid-apps/ Haettu 11.5.2021

22. Tuntematon kirjoittaja, Top 10 Hybrid Mobile App Development Frameworks, https://clever-solution.com/top-10-hybrid-mobile-app-development-frameworks/ Haettu 11.5.2021

23. Tuntematon kirjoittaja, Native App vs Hybrid App vs Web App https://www.amiwebsolutions.com/mobile-app-development/native-hybrid-web-apps/ Haettu 11.5.2021

24. Rohit Joseph Marcus, Xamarin Native vs Xamarin.Forms: How to Choose, https://www.logiticks.com/blog/xamarin-native-vs-xamarin-forms-how-to-choose/ Haettu 11.5.2021

25. Hardik Shah, ReactJs vs React Native – Key Difference, Advantages, and Disadvantages, https://www.simform.com/reactjs-vs-reactnative Haettu 11.5.2021

26. Nkenganyi Clovis, How To Learn ReactJS in 2021, https://dev.to/clovissocial/how-to-learn-reactjs-in-2021-389d Haettu 11.5.2021

27. React Native dokumentaatio, React Native – Overview, https://www.tutorialspoint.com/react_native/react_native_overview.htm Haettu 11.5.2021

28. Nitin pandit, What And Why React.js, https://www.c-sharpcorner.com/article/what-and-why-reactjs/ Haettu 11.5.2021

29. ReactJS dokumentaatio, Introducing JSX, https://reactjs.org/docs/introducing-jsx.html Haettu 11.5.2021

30. Angular dokumentaatio, https://angular.io/docs Haettu 11.5.2021

31. Angular dokumentaatio, What is Angular, https://angular.io/guide/what-is-angular Haettu 11.5.2021

32. Angular dokumentaatio, Setting up the local environment and workspace, https://angular.io/guide/setup-local Haettu 11.5.2021

33. Ionic dokumentaatio, Ionic Framework, https://ionicframework.com/docs Haettu 11.5.2021

34. Mark Surman, What is the open web and why is it important?, https://www.yearofopen.org/november-open-perspective-what-is-open-web/what-is-the-open-web-and-why-is-it-important-submitted-by-mark-surman-executive-director-of-the-mozilla-foundation/ Haettu 11.5.2021

35. Ionic dokumentaatio, Welcome to Appflow, https://ionic.io/docs/appflow Haettu 11.5.2021