Mobiilisovelluskehitysympäristöjen vertailu ja soveltuvuus opetuskäyttöön

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, biotalouden koulutus, Forssa

2021

Olli Hoviniemi

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka, biotalouden koulutus Tiivistelmä

Forssa

Tekijä Olli Hoviniemi Vuosi 2021

Työn nimi Mobiilisovelluskehitysympäristöjen vertailu ja soveltuvuus opetuskäyttöön

Ohjaajat Johanna Salmia, Ari Hietala

TIIVISTELMÄ

Mobiilisovelluksen luominen on monivaiheinen prosessi. Nopean kehityksen myötä mobiilisovelluksia on mahdollista tehdä monella eri tavalla. Tässä opinnäytetyössä tutkitaan ja vertaillaan erilaisia mobiilisovellustekniikoita, joita ovat natiivi-, hybridi-, web- ja PWA-tekniikat. Sovellusta suunniteltaessa on hyvä miettiä, mihin tarkoitukseen sovellus tulee.

Opinnäytetyön tarkoituksena on selvittää, millä tekniikalla on nopeinta ja ketterintä luoda sovelluksia opetuskäytössä. Tarkoituksena on tutkia myös sitä, mikä voisi olla tarpeeksi monipuolinen kehitysympäristö aloittelevalle mobiilisovelluskehittäjälle.

Työn tilaajana toimii Hämeen ammattikorkeakoulun tieto- ja viestintätekniikan biotalouden koulutus. Moduuliopinnoissa on aiemmin käytetty sovellusten tekemiseen PWA-tekniikkaa. Opinnäytetyössä selvitetään, onko tämä edelleen hyvä ja nopea tapa tehdä sovelluksia vai korvaako mahdollisesti muu tekniikka tämän lähitulevaisuudessa. Tässä työssä käydään läpi, miten sovellus julkaistaan App Storessa tai Google Play-kaupassa.

Opetuskäyttöön parhaiten sopivat PWA-tekniikka ja low-code ympäristöt. Muihin sovellustekniikoihin verrattuna nämä ovat alhaisen lähtötason kehittäjälle parhain ratkaisu.

Avainsanat Mobiilisovellustekniikat, low-code, PWA, sovelluksen julkaiseminen

Sivut 37 sivua ja liitteitä 1 sivu

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in ICT, Bio and Circular Economy

Forssa

Abstract

Author Olli Hoviniemi

Year 2021

Subject Comparison of mobile application development environments and suitability for educational use

Supervisors Johanna Salmia, Ari Hietala

ABSTRACT

To create mobile applications is a multi-stage process. With the rapid development, it is possible to make mobile applications in many ways. In this thesis, different mobile apps with are native-, hybrid- web- and Progressive Web Application-technology, were compared. The starting point of the work was the idea that, when planned the app, it is good to clarify the purpose the app is planned for. Therefore, the aim of this thesis was to explore, which technique is the fastest and most agile for educational and teaching purposes and what would be the most convenient and versatile development environment for a beginner level developer.

The commissioner of this thesis was HAMK, University of Applied Sciences, the Degree Programme in ICT and Bio and Circular Economy. Earlier, during the module studies, the technology used for application designs was PWA. In this thesis, the goal was to verify whether PWA is a good and fast way to make applications or whether it could be substituted by other technologies in the near future. In addition, instructions on how to publish an app in the App Store or Google Play Store were provided.

The result of the thesis was that PWA technology and low-code environments are best suited for educational use. Compared to other application making techniques, they are the best solution for the beginner level developer.

Keywords Mobile application techniques, low-code, PWA, application publishing

Pages 37 pages and appendices 1 page

Sisällys

1 Johdanto 1
2 Mobiilisovellukset 2
2.1 Historia 2
2.2 Trendit 3
2.2.1 Alustariippumattomuus 3
2.2.2 Puettavat laitteet 4
2.2.3 Lisätty- ja virtuaalitodellisuus 4
2.2.4 Tekoäly 5
2.2.5 5G:n vaikutus mobiilisovelluskehitykseen 6
2.2.6 Beacon-tekniikka 7
2.2.7 IoT:n vaikutus mobiilisovelluskehitykseen 8
2.2.8 Low-code kehitysympäristö 9
3 Mobiilisovellustekniikat 10
3.1 Natiivisovellus 10
3.2 Hybridisovellus 11
3.3 Web-sovellus 11
3.4 Progressiivinen web sovellus (PWA) 12
3.4.1 Lighthouse 13
3.4.2 Hyödyt ja haitat 14
3.4.3 Tuki 14
3.5 Tekniikoiden vertailu 15
3.5.1 PWA vs. natiivisovellus 16
4 Kestävä kehitys mobiilisovelluskehityksessä 17
5 Kehitysympäristöt 17
5.1 Outsystems 18
5.2 Android Studio 20
5.3 AppGyver 22
5.4 Ionic 23
5.5 Xcode 24
6 Sovelluksen esikatselu ja kehitysympäristön ulkoasu 25
6.1 Ulkoasu 25
6.2 Sovelluksen esikatselu 26
7 Julkaiseminen sovelluskaupassa 27

7.1 Sovelluksen julkaiseminen Google Play-kaupassa 27
7.2 Sovelluksen julkaiseminen App Store-kaupassa 28
8 Pohdinta ja johtopäätökset 30
Lähteet 32

Kuvat, taulukot ja kaavat

Kuva 1. Käyttöjärjestelmien markkinaosuudet vuonna 2020 (Statcounter, n.d.) 3
Kuva 2. Aikuisten puettavien laitteiden käyttö Amerikassa (Hindi, n.d.). 4
Kuva 3. Ikea Place-sovellus (Staff, 2019) 5
Kuva 4. Kuinka yritykset käyttävät tekoälyä? (Biswas, 2020) 6
Kuva 5. Datayhteyksien ominaisuudet (Vella, 2019) 7
Kuva 6. Beacon-tekniikan toiminta (Kazan, 2020) 8
Kuva 7. IoT esineiden määrä vuodesta 2015 vuoteen 2025 (PV, 2021) 9
Kuva 8. Verkkosivun rakenne (1Training.org, n.d.) 11
Kuva 9. Esimerkki Lighthouse-raportista 14
Kuva 10. Järjestelmien tuki PWA:lle vuonna 2020 (Poot, n.d.) 15
Kuva 11. Mobiilisovellustekniikoiden ominaisuudet (Cumulations, 2020) 16
Kuva 12. Outsystems käyttöliittymä 18
Kuva 13. Outsystems palvelun rakenne (Outsystems, n.d.-c) 19
Kuva 14. Android Studio käyttöliittymä 21
Kuva 15. Firebase ominaisuudet (Firebase, n.d.) 21
Kuva 16. AppGyver käyttöliittymä 22
Kuva 17. Ionic projekti Visual Studio Codessa 23
Kuva 18. Xcode käyttöliittymä 25
Kuva 19. Outsystems emulaattori 26
Kuva 20. APK-tiedoston lataaminen (Oragui, 2018) 28
Kuva 21. Sovelluksen generointi (Ching, 2019) 29

Liitteet

Liite 1 Aineiston hallintasuunnitelma

1 Johdanto

Hämeen ammattikorkeakoulun tieto- ja viestintätekniikan biotalouden koulutus toimii työn tilaajana. Koulutus on uusi opintotarjonnassa, ja se toteutettiin ensimmäisen kerran vuoden 2017 syksyllä. Koulutuksessa yhdistyy biotalous ja tieto- ja viestintätekniikka. Opintojen aikana opiskelija saa kattavat tietotekniset taidot, joita voi soveltaa bio- ja kiertotalouden teknisissä ratkaisuissa (Hämeen Ammattikorkeakoulu, n.d.).

Tässä opinnäytetyössä tutkitaan mobiilisovelluskehitystä: mistä kaikki on alkanut ja mihin suuntaan kehitys on menossa. Mobiilisovelluksia voi tehdä muutamalla eri tavalla: natiivi-, hybridi- tai web-sovellus, PWA-tekniikka. (Gregory, 2019)

Sovelluskehitystä pyritään muuttamaan siihen suuntaan, että kaikilla on mahdollisuus ohjelmoida sovelluksia osaamisesta huolimatta. Monella on varmasti mielikuvana, että ohjelmointi ja sovellusten tekeminen on vaikeaa ja siihen tarvitaan vankkaa ohjelmointiosaamista. Näin ei kuitenkaan ole. Low-code kehitysalustat ovat yleistyneet ja ne mahdollistavat sovellusten tekemisen ilman ohjelmointikielien osaamista.

Opinnäytetyön tarkoituksena on etsiä opetuskäyttöön soveltuvaa nopeaa ja ketterää tekniikkaa tai kehitysympäristöä. Kehitysympäristöltä odotetaan monipuolisia ominaisuuksia, joita ovat esimerkiksi helppokäyttöisyys ja integrointimahdollisuudet. Työssä tullaan tutkimaan ja vertailemaan erilaisia kehitysympäristöjä.

Sovelluskehityksen viimeinen vaihe on sovelluksen julkaiseminen. Opinnäytetyössä käydään läpi, miten sovellus julkaistaan. Sovelluksen voi julkaista App Storessa, Google Playssa tai Windows Storessa. Sovellusten julkaiseminen käydään läpi vain kahden yleisimmän sovelluskaupan osalta.

Opinnäytetyö pyrkii vastaamaan seuraaviin tutkimuskysymyksiin:

1. Ovatko low-code kehitysympäristöt tulevaisuutta mobiilisovelluskehityksessä?
2. Miten sovellus julkaistaan sovelluskaupassa?
3. Mikä mobiilisovellustekniikka tai -kehitysympäristö on paras opetuskäyttöön?

2 Mobiilisovellukset

Mobiilisovelluksella tarkoitetaan ohjelmistoa, joka ladataan kannettavaan päätelaitteeseen alustakohtaisesta sovelluskaupasta. Yleisesti mobiilisovelluksia on mahdollista ladata suosituimmista sovelluskaupoista, jotka ovat Android Studio, App Store ja Windows Store. App Storesta voi ladata sovelluksia Applen laitteisiin ja Google Play:sta Android-laitteisiin. Mobiilisovellukset ovat joko maksullisia tai ilmaisia. (Sanoma, n.d.)

2.1 Historia

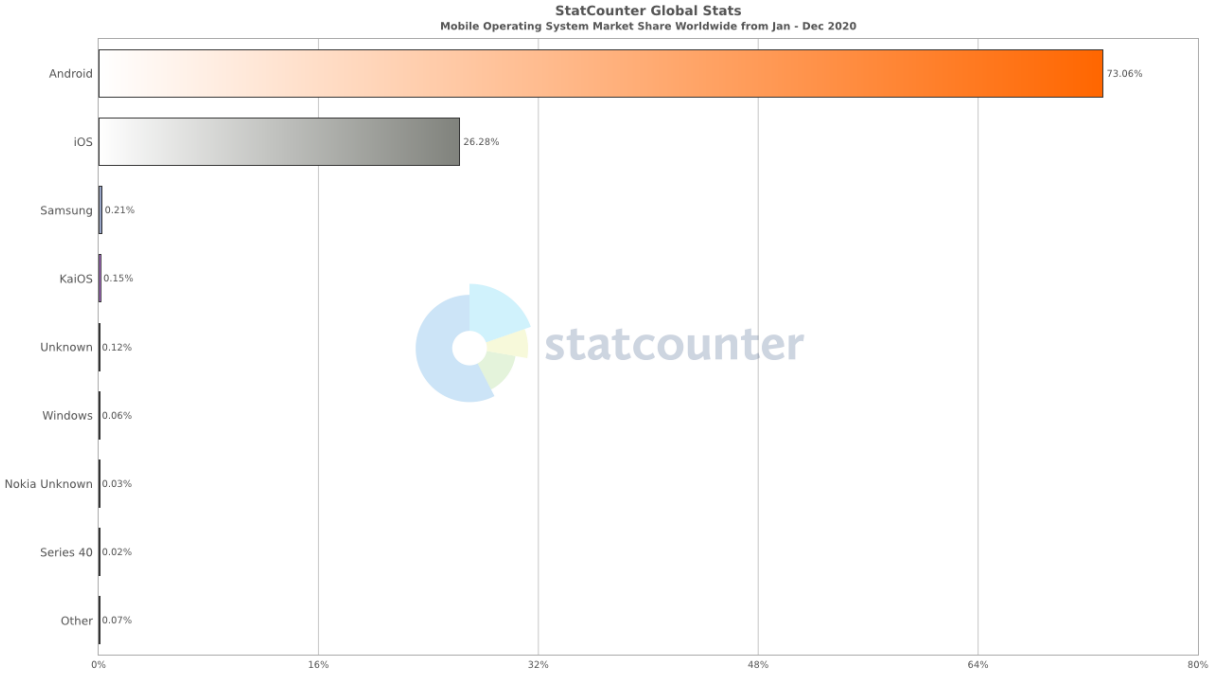
Varsinainen mobiilisovellusten nousu alkoi, kun ensimmäinen kosketusnäyttöpuhelin julkaistiin IBM:n toimesta vuonna 1993. Puhelin sisälsi valmiiksi asennettuja sovelluksia, jotka olivat kalenteri, kello, muistio, sähköposti ja yhteystiedot. Helppokäyttöisyys ja kehittyneemmät sovellukset tekivät tästä kosketusnäyttöpuhelimesta suositun siihen aikaan. (Unifunds, n.d.)

Ennen sovelluskauppoja sovelluksia ladattiin WAP-protokollan avulla (Wireless Application Protocol). WAP on WAP-Forumin kehittelemä protokolla, jonka avulla vanhemmilla laitteilla oli mahdollisuus päästä internet-selaimeen. Vähäinen muisti ja huono resoluutio näytössä johtivat siihen, että selaimet sivut olivat pelkistettyjä normaalista. (Tutorialspoint, n.d.)

Apple julkaisi ensimmäisen omalla iOS-käyttöjärjestelmällä toimivan iPhonen vuonna 2007. Samaan aikaan julkaistiin myös Applen sovelluskauppa App Store. App Storesta käyttäjät pystyivät lataamaan eri käyttötarkoituksiin tarkoitettuja sovelluksia. Hyvin nopeasti iPhonen julkaisemisen jälkeen markkinoille tuli Android-käyttöjärjestelmä. HTC julkaisi ensimmäisenä kuluttajille suunnatun matkapuhelimen pian Androidin julkistamisen jälkeen. Näiden kahden Androidin ja iOSin markkinat lähtivät räjähdysmäiseen kasvuun. Vuonna 2009 Apple saavutti ensimmäisenä miljardin latauksen rajan ja vuosi tämän jälkeen Android ylsi samaan. Sovelluskaupoissa oli vuonna 2011 yli miljoona ladattavaa sovellusta. (Unifunds, n.d.)

Markkinat ovat jakautuneet käytännössä kahden käyttöjärjestelmän välille. Kuvasta nähdään, että Android ja iOS ovat suosituimmat käyttöjärjestelmät. (Kuva 1)

Kuva 1. Käyttöjärjestelmien markkinaosuudet vuonna 2020 (Statcounter, n.d.)



2.2 Trendit

Teknologia kehittyy vauhdikkaasti ja luo uusia mahdollisuuksia sovelluskehitykseen. Uudet teknologiat ja trendit muokkaavat mobiilisovelluskehitystä tulevaisuudessa. Mobiilisovelluskehitykseen vaikuttavia trendejä ovat tekoäly, 5G, esineiden internet (IoT), AR (Augmented Reality) ja VR (Virtual Reality), alustariippumattomuus ja puettavat laitteet. Low-code kehitysympäristöt ovat yksi mobiilikehityksen trendeistä. (Devathon, 2020) Kaikki nämä trendit täydentävät toisiaan ja mahdollistavat tulevaisuudessa vieläkin vaikuttavampia sovelluksia kuin nyt. (Malhotra, 2020) Beacon-tekniikka on yksi nousevista trendeistä (Hindi, n.d.).

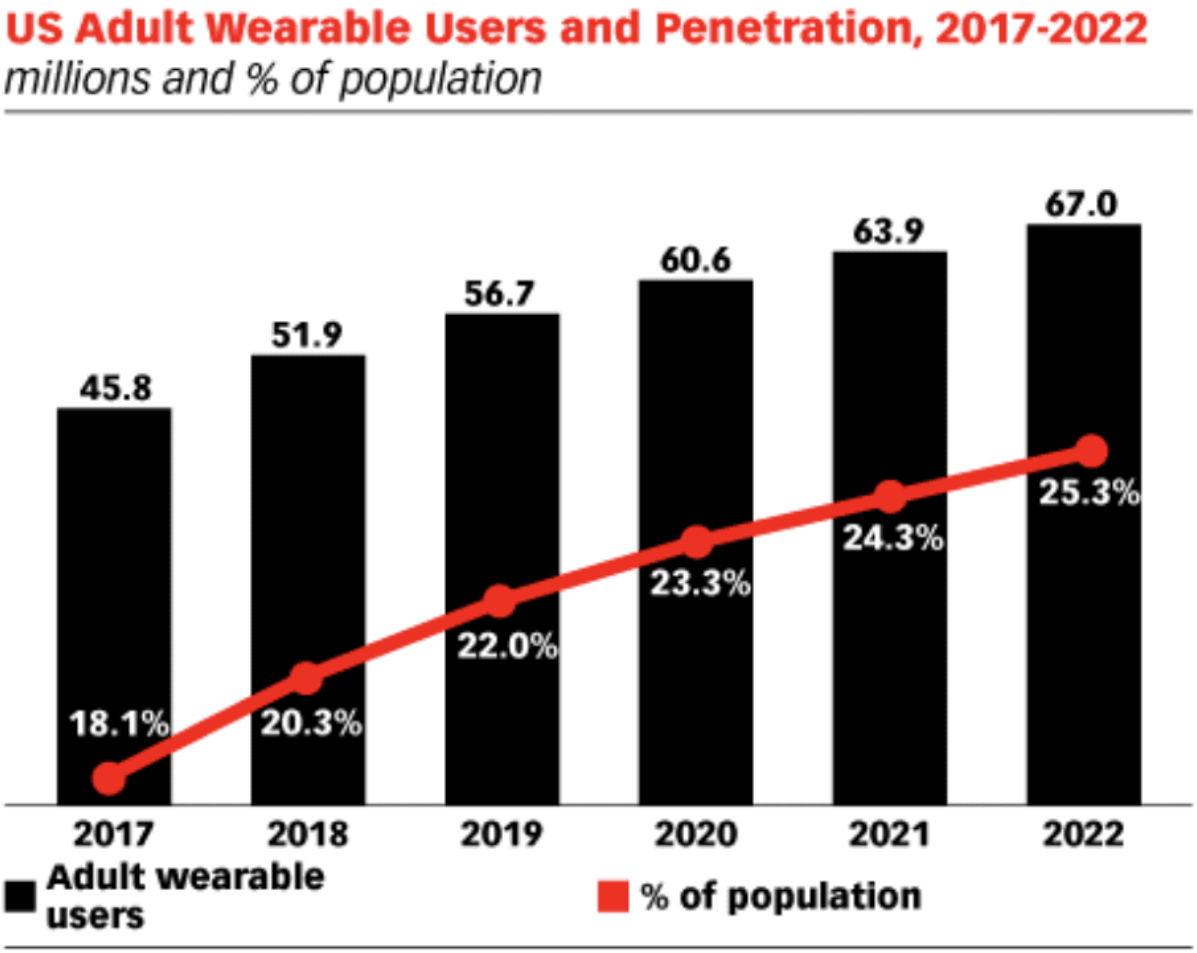
2.2.1 Alustariippumattomuus

Alustariippumattomuus on kasvattanut suosiotaan mobiilisovelluskehityksessä. Alustariippumattomuuden avulla sovelluksen voi luoda usealle eri alustalle säästäen aikaa ja rahaa. Markkinoilla on useita suosittuja työkaluja, joiden avulla voi luoda alustariippumattomia sovelluksia. Näitä ovat esimerkiksi React Native, Flutter, Xamarin ja Unity. (Strizić, 2021)

2.2.2 Puettavat laitteet

Puettavia laitteita on ollut markkinoilla vuosia, joten uudesta asiasta ei ole kyse. Viimeisten vuosien aikana ihmisten kiinnostus on lisääntynyt puettavia laitteita kohtaan. Puettaviksi laitteiksi luokitellaan älykellot, älyvaatteet ja jäljittimet. Kasvu ei ole ollut räjähdysmäistä tällä alueella, mutta markkinat ovat silti kasvaneet tasaisesti. (Kuva 2) Vuonna 2021 on ennustettu, että mobiilisovelluksia tehdään entistä enemmän puettaville laitteille. Tulevien vuosien aikana suurimmat askeleet tapahtuvat tällä alalla. (Hindi, n.d.)

Kuva 2. Aikuisten puettavien laitteiden käyttö Amerikassa (Hindi, n.d.).



2.2.3 Lisätty- ja virtuaalitodellisuus

Lisätty todellisuus (AR) ja virtuaalitodellisuus (VR) ovat olleet suosittuja viihde- ja pelialalla, mutta ne ovat siirtyneet hiljalleen mobiilisovelluskehitykseen ja on samalla yksi kehityksen

trendeistä tällä hetkellä. AR ja VR muuttavat sitä, miten sovelluksia tehdään tulevaisuudessa. VR:n markkina-arvon uskotaan olevan vuonna 2024 jopa 44,7 miljardia dollaria. VR:n avulla sovelluksista pystytään tehdä entistäkin mukaansatempaavampia. VR on otettu käyttöön useammilla teollisuuden aloilla. (Kokki, 2019)

AR-teknologia on lisääntynyt tabletti- ja puhelinalustoilla kehittyneiden prosessorien ja kameroiden ansiosta. Monet suositut sovellukset, kuten Snapchat ja Instagram, tarjoavat mahdollisuuden hyödyntää lisättyä todellisuutta. (Grannell, 2021) Ikea on julkaissut oman versionsa AR-teknologiaa hyödyntävästä sovelluksesta. (Kuva 3) Sovelluksen nimi Ikea Place. Sovelluksessa pystyy tarkastelemaan, miltä Ikean valikoiman tuotteet näyttäisivät omassa kodissa. Sovellus skaalautuu 98 prosentin tarkkuudella huonemittoihin verrattuna. Käyttäjä saa hyvin todenmukaisen kuvan siitä, minkä kokoinen tuote on. (Ayobi, 2017)

Kuva 3. Ikea Place-sovellus (Staff, 2019)

2.2.4 Tekoäly

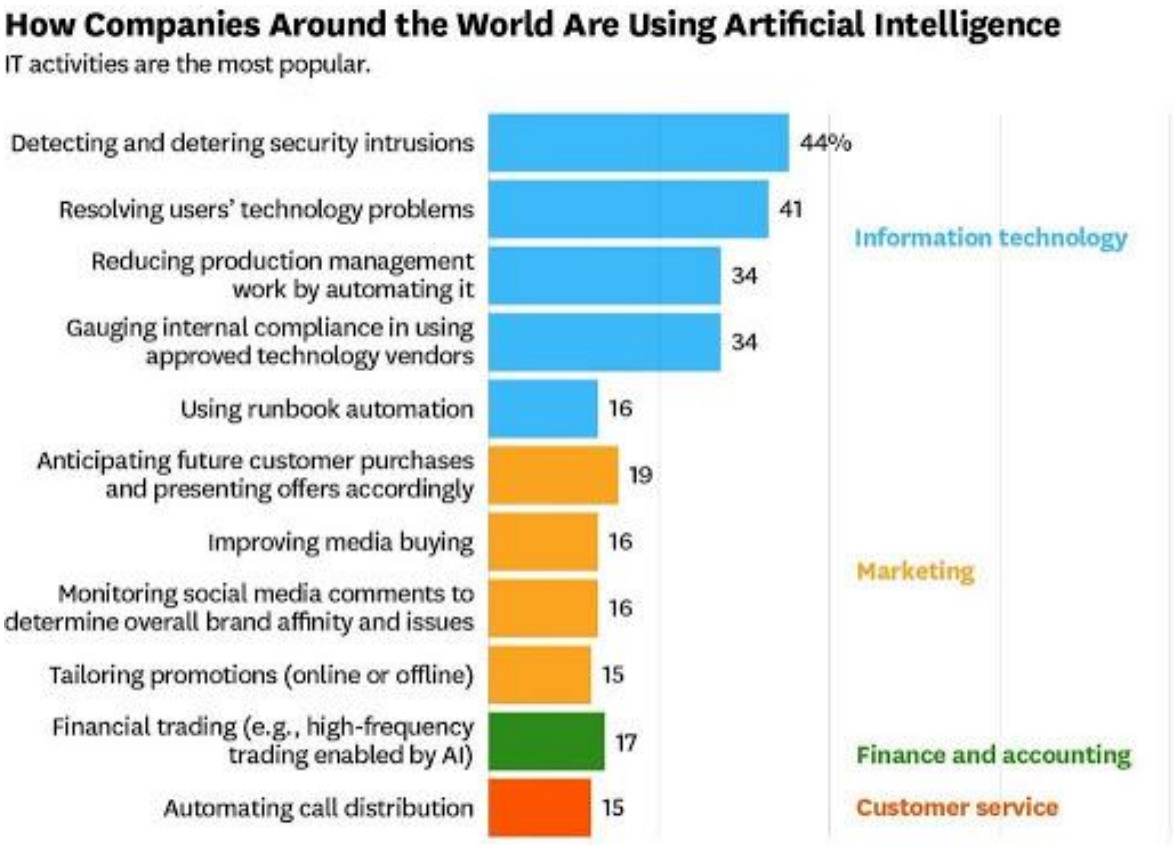
Tekoäly on vakiinnuttanut paikkansa mobiilisovelluskehityksessä. Esimerkiksi Siri ja Alexa ovat ensimmäisiä virtuaalisia tekoälyyn pohjautuvia avustajia. Tekoälyä voidaan hyödyntää

kuvantunnistuksessa, kasvojentunnistuksessa, tunteiden tunnistamiseen, puheentunnistukseen ja ennakoivaan huoltoon. Tekoäly parantaa käyttäjäkokemusta ja tekee sovelluksista entistäkin älykkäämpiä. (Hindi, n.d.)

Chattibotit yleistyvät vuosi vuodelta ja niiden käytön on ennustettu kasvavan vuosittain noin 24 prosenttia. Chattibottien tausta toimii tekoäly. Chattibotit nostavat asiakaspalvelun uudelle tasolle. Chattibotit yrittävät matkia ihmismäisyyttä. (Hindi, n.d.)

Yritykset ympäri maailmaa käyttävät tekoälyä omassa yrityksessä eri tarkoituksiin. (Kuva 4) Tekoälyn käyttö on pitkälti painottunut toistaiseksi teknisten asioiden automatisointiin.

Kuva 4. Kuinka yritykset käyttävät tekoälyä? (Biswas, 2020)



2.2.5 5G:n vaikutus mobiilisovelluskehitykseen

5G on seuraavan sukupolven versio uudesta verkkotekniikasta. 5G käyttää millimetritaajuuksia, joiden ansiosta uudempi verkkotekniikka voi yltää jopa 4–5 kertaa suurempiin nopeuksiin kuin 4G:ssä. (Lounea, n.d.) 5G:n nopeampi yhteys perustuu verkon

viipalointiin. Tämä käytännössä tarkoittaa sitä, että jokainen verkon käyttäjä saa oman häiriö- ja ruuhkavapaan kaistan käyttöönsä. Viipalointi lisää käyttäjien tietoturvaa (DNA, n.d.).

5G tulee muuttamaan mobiilisovelluskehitystä merkittävästi, sillä sen vasteajat, puhutaan myös latenssista, pienenevät merkittävästi 4G:stä. (Kuva 5) Latausajat moninkertaistuvat aikaisemmasta teknologiasta. Nämä seikat vaikuttavat eniten pelaamiseen, AR/VR-teknologioihin ja tietoturvaan. 5G:n avulla sovelluksiin voi tuoda monimutkaisempia ja raskaampia ominaisuuksia, sillä sovellusten suorituskyky paranee (Azorin, 2019)

Kuva 5. Datayhteyksien ominaisuudet (Vella, 2019)

		3G	4G	5G
	Deployment	2004-05	2006-10	2020
	Bandwidth	2mbps	200mbps	>1gbps
	Latency	100-500 milliseconds	20-30 milliseconds	<10 milliseconds
	Average Speed	144 kbps	25 mbps	200-400 mbps

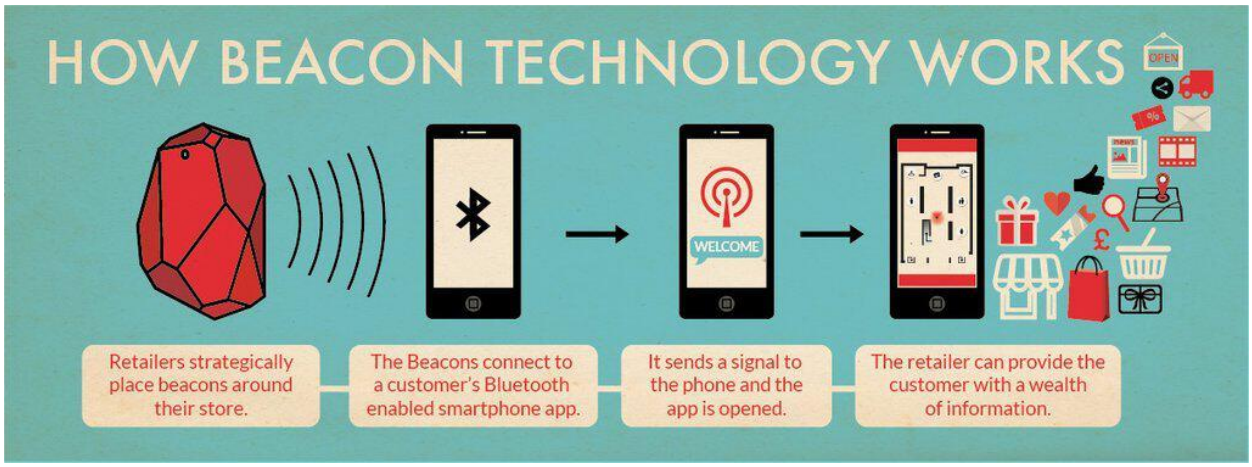
5G:N avulla on mahdollista tulevaisuudessa tehdä kirurginen operaation etänä. 5G on kokonaisuudessaan luotettava verkko. Teollisuudessa 5G:tä pystytään hyödyntämään esimerkiksi automatisoinnissa. (Vella, 2019)

2.2.6 Beacon-tekniikka

Beacon-tekniikkaa käytetään teollisuudessa laajalti sekä esimerkiksi vähittäiskaupoissa ja terveydenhuollossa. Tekniikan avulla voidaan lisätä mihin tahansa mobiilisovelluksiin kehittyneitä toimintoja. Beacon-tekniikasta puhutaan myös majakoina, jotka toimivat mobiilisovellusten kanssa. Ensimmäiset mobiilisovellusmajakat ovat vuodelta 2013, jonka jälkeen tähän päivään saakka tämä teknologia on edistynyt. Majakat toimivat mobiilisovellusten kanssa Bluetoothin kautta, mikäli laite on yhdistettynä lähettävään

laitteeseen. (Kuva 6) Käyttäjän ohittaessa majakan myymälä voi lähettää kohdennettuja ilmoituksia asiakkaalle, myymälä voi tällöin myös seurata asiakkaan toimintoja. Läheisyysmarkkinoinnin etu on mobiilisovelluksen asiakaskokemuksen parantuminen. (Hindi, n.d.)

Kuva 6. Beacon-tekniikan toiminta (Kazan, 2020)



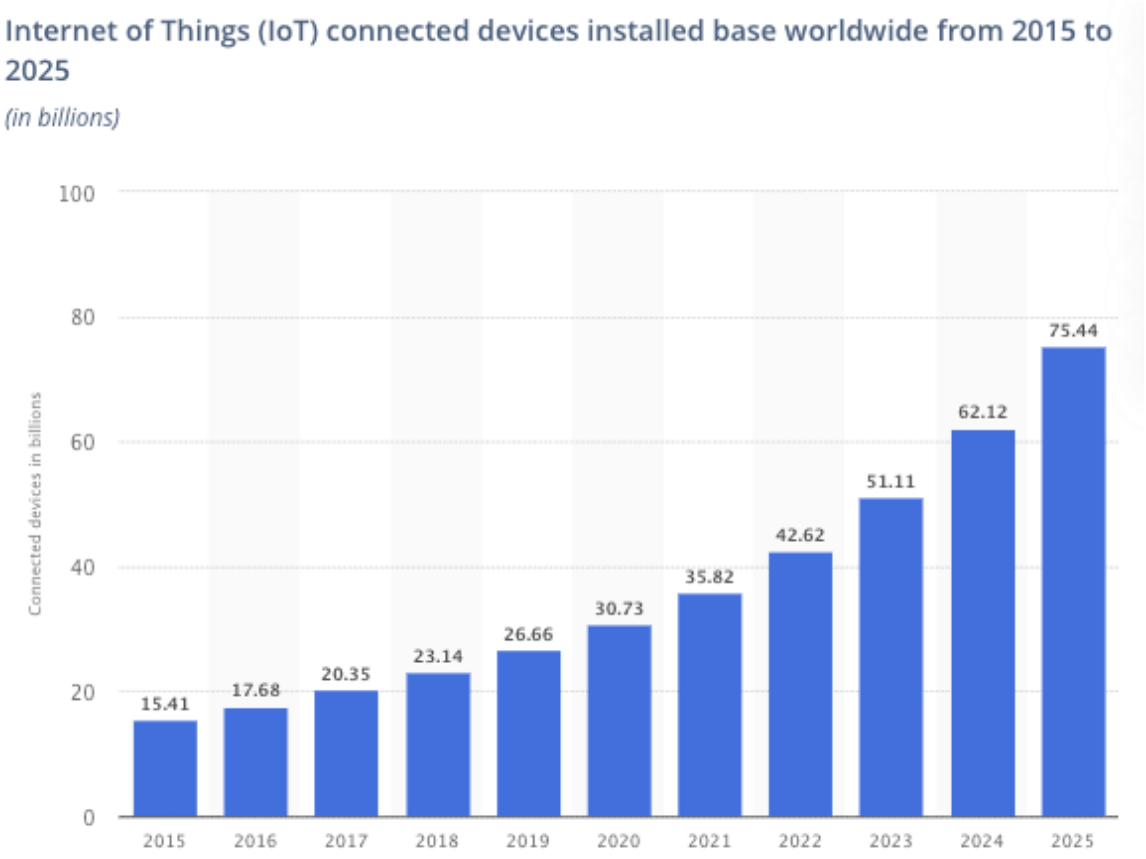
2.2.7 IoT:n vaikutus mobiilisovelluskehitykseen

IoT:lla (Internet of Things) tarkoitetaan esineiden liittämistä internettiin. Liitettäviä esineitä voivat olla auto, jääkaappi yms. Liitetyt laitteet lähettävät ja vastaanottavat erilaista dataa. Ensimmäisen kerran IoT-määritelmää käytti Kevin Ashton vuonna 1999. (Empirica, n.d.)

IoT:n yleistyminen on aiheuttanut nopean nousun. Vuonna 2021 on arvioitu olevan 35,82 miljardia yhdistettyä laitetta. Vuonna 2025 on arvoitu yhdistettyjen laitteiden määrän

nousevan 75,44 miljardiin. Neljän vuoden aikana on ennustettu siis kaksinkertaistumista. (Kuva 7)

Kuva 7. IoT esineiden määrä vuodesta 2015 vuoteen 2025 (PV, 2021)



IoT:n avulla käyttäjä voi hallita yhdistettyjä laitteita sijainnista riippumatta. Tekoäly ja IoT yhdistettyinä luovat käyttäjilleen parhaan mahdollisen käyttökokemuksen. Tekoäly voi ennustaa ja ehdottaa käyttäjälle rutinoituneita toimenpiteitä, kuten esimerkiksi saunan päälle laiton tiettynä päivänä. (Strizić, 2021)

2.2.8 Low-code kehitysympäristö

Low-code kehitysympäristöllä tarkoitetaan kehitysympäristöä, missä kehittäjä tarvitsee vähän tai ei ollenkaan osaamista koodaamisesta. Low-code-työkalut helpottavat kehittäjän työtä vähentämällä itse koodaamista sovelluskehityksessä. Low-code-kehitysympäristöön on valmiiksi luotu elementtejä ja toimintalogiikkoja, mitä yhdistelemällä saadaan aikaan itse sovellus. Tämä on myös jollain tapaa low-code-ympäristön heikkous, sillä monimutkaisia

sovelluksia on lähes mahdoton tai mahdotonta tehdä. Low-coden merkittävin hyöty on sen nopeus. Kehitysprosessi lyhenee ja sovelluksia päästään testaamaan entistä nopeammin. (Tapanainen, n.d.)

Ohjelmistokehittäjistä on valtava pula yritysmaailmassa. Osaavia tekijöitä on todella vaikea löytää, joten sovelluskehittämisestä täytyy tehdä helpompaa ja suoraviivaisempaa. Low-code on kustannustehokas vaihtoehto perinteiseen sovelluskehitykseen verrattuna. Näiden asioiden pohjalta on vähitellen ruvennut syntymään low-code kehitysympäristöt. (Tapanainen, n.d.)

Vaikka tämä tekniikkana on varsin hyvä monessakin mielessä, niin tämän low-coden ei ole tarkoitus korvata olemassa olevia sovelluskehitysmenetelmiä. Low-codea kannattaa käyttää uusissa sovelluksissa, jotka siirtävät ja tuovat tietoa muihin järjestelmiin. (Tapanainen, n.d.)

3 Mobiilisovellustekniikat

Mobiilisovelluksia voi tehdä muutamalla eri tavalla: hybridisovelluksella, natiivisovelluksella, web-sovelluksella ja PWA-tekniikalla. Tässä luvussa käydään läpi, mitä nämä käsitteet tarkoittavat.

3.1 Natiivisovellus

Natiivisovellus ohjelmoidaan alustakohtaisella ohjelmointikielellä. Ohjelmointiympäristö Xcodella tehdään iOS-sovellukset joko Swift- tai Objetive-C-ohjelmointikielellä. Android Studiolla tehdään Android-sovellukset Java- tai Kotlin-kielellä. Natiivisovellus tarjoaa parhaan suorituskyvyn sovellukseen. (Griffth, n.d.) Kehittäjien on mahdollista käyttää kaikkia mobiililaitteen tarjoamia natiiveja ominaisuuksia, kuten GPS (Global Position System), FaceID ynnä muuta sellaista. Natiivin sovelluksen kehittäminen on työlästä ja kallista. Jokaiselle käyttöjärjestelmälle on tehtävä koodi erikseen eikä koodi ole helposti käännettävissä muille järjestelmille. Natiivisovellus on hyvä vaihtoehto, jos tehdään pitkäaikainen sovellus tai peli, joka käyttää montaa eri mobiililaitteen tarjoamaa natiivia ominaisuuttaa. (Sysart, n.d.)

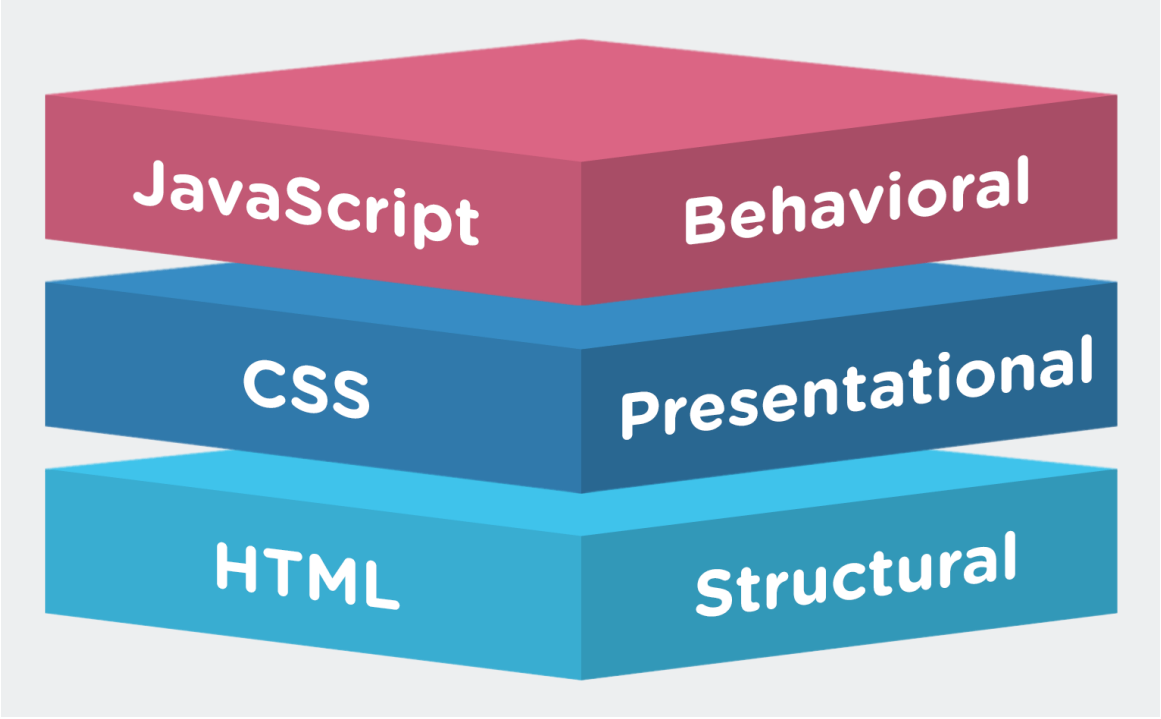
3.2 Hybridisovellus

Hybridisovellus jäljittelee natiivia sovellusta, mutta sitä on vaikea erottaa siitä, sillä se näyttää hyvin samanlaiselta. Hybridisovellus ladataan normaalisti sovelluskaupasta. Hybridisovelluksen luonnissa käytetään verkkotekniikoita (HTML5, CSS) ja sovellus paketoidaan natiiviin kuoreen. Natiivin kuoren avulla hybridisovellukset pystyvät hyödyntämään puhelimen natiiveja ominaisuuksia. HTML (Hypertext Markup Language) ja CSS (Cascading Style Sheets) ovat keskisimmät tekniikat web-kehityksessä. (W3C, n.d.) Hybridisovellus vaatii vain yhden koodin, jota voidaan hyödyntää jokaisella alustalla, mitä on helppo päivittää tarpeen tullessa. (Juntunen, 2019)

3.3 Web-sovellus

Web- eli verkkosovellukset tehdään käyttäen kolmea eri tekniikkaa: Javascript, CSS ja HTML5. (Kuva 8) HTML:n avulla luodaan sivuston sisältö ja rakenne. CSS:n avulla määritetään visuaaliset asiat, kuten fontit ja värit sivulle. Javascript on ohjelmointi- ja komentosarjakieli, jonka avulla sivustosta saadaan tehtyä interaktiivinen. Javascript on monimutkaisempi ohjelmointikieli kuin HTML ja CSS. (1Training.org, n.d.)

Kuva 8. Verkkosivun rakenne (1Training.org, n.d.)



Web-sovellus on verkkosivu, joka avataan selaimessa. Web-sovellus on täysin alustariippumaton ja se voidaan avata millä tahansa päätelaitteella. Erona natiiviin sovellukseen on se, että sovellusta ei ladata puhelimeen vaan se avataan selaimessa. (Redandblue, 2020) Verkkosovellus vaatii jatkuvan yhteyden verkkoon. Hybridisovelluksen tavoin verkkosovellus tehdään yhteen koodiin, jota on helppo muokata sekä päivittää. Verkkosovelluksia on nopeampi ja helpompi rakentaa, mutta ominaisuuksiltaan ne ovat heikompia verrattuna muihin sovellustekniikoihin. (Stevens, 2018)

Kehittäjän kannalta HTML- ja CSS-tekniikat on helppo oppia. Rakenne niissä on selkeä ja johdonmukainen. HTML on yksi yleisimmin käytetyistä ohjelmointikielistä. Tästä on hyvä aloittaa, jos kokemusta ei ole ohjelmoinnista aiemmin. (Thinkful, n.d.)

3.4 Progressiivinen web sovellus (PWA)

PWA-sovellus luodaan, kuten web-sovellus, käyttäen HTML5, CSS ja Javascript:ia. PWA-sovellus eroaa normaalista web-sovelluksesta hieman. PWA-sovellus tarjoaa käyttäjilleen natiivin käyttökokemuksen. PWA-sovelluksesta on mahdollista saada reaaliaikaisia push-ilmoituksia natiivin sovelluksen tavoin. Push-ilmoitukset voivat tulla suoraan sovelluksesta tai palvelimelta. (Lamia, n.d.)

PWA-sovellukseksi voidaan sanoa verkkosovellusta, jos siinä on seuraavat ominaisuudet: service worker, web app manifest ja HTTPS-protokolla. HTTPS (Hypertext transfer protocol secure) käyttää SSL-varmennetta (Secure sockets layer), joka tekee turvallisen yhteyden selaimen ja palvelimen välille. (Harnish, 2020) Web app manifest on käytännössä yksinkertainen JSON-tiedosto, joka sisältää PWA:n perustiedot. (IQUII, 2019) Service worker on yksi PWA-sovelluksen tärkeimmistä asioista. JavaScript-pohjainen Service worker mahdollistaa sovelluksen toimimisen verkottomassa tilassa. Se sieppaa verkkopyyntöjä, noutaa resursseja, toimittaa Push-viestejä ja tallentaa tarvittavat tiedot verkkosivusta välimuistiin. Kun sovellus avataan seuraavaan kerran, service worker palauttaa välimuistissa olevat tiedot käyttäjälle. HTTPS-protokolla luo turvallisuutta PWA-sovelluksen käyttämiseen, sillä sovellus seuraavaa verkkoliikennettä jatkuvasti. Vääriin käsiin tietojen ei ole hyvä joutua. (Haldar, 2018)

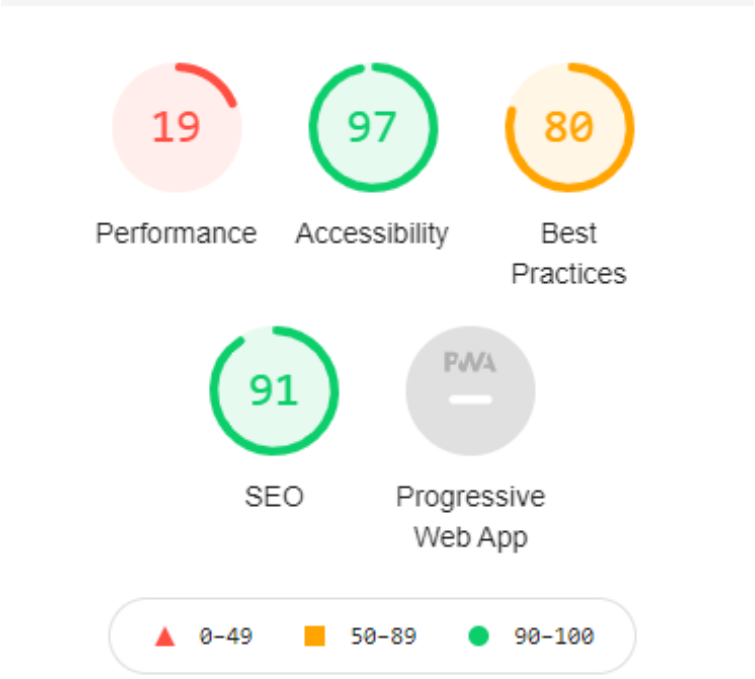
PWA-sovelluksen voi lisätä puhelimen kotivalikkoon. iOS-järjestelmällä tämä toimii siten, että jaa-painikkeesta aukeaa valikko, josta valitaan lisää kotivalikkoon. Android-järjestelmällä selain ilmoittaa, jos PWA-sovellus on saatavilla kyseisellä verkkosivulla. Kun sovellus on lisätty kotivalikkoon, se näyttää aivan natiivilta sovellukselta. (Mantco, n.d.)

3.4.1 Lighthouse

Google on kehittänyt avoimeen lähdekoodiin perustuvan työkalun nimeltä Lighthouse. Lighthousen avulla PWA-sovelluksen voi optimoida paremmaksi käytettävyyden, saavutettavuuden ja löytyvyyden osalta. Lighthousen saa selaimiin ladattavana lisäosana. (Sormunen, 2020)

Raportti antaa tiedot suorituskyvystä, saavutettavuudesta, parhaista käytännöistä, SEO:sta ja PWA-mittarista. (Kuva 9) Suorituskyky kertoo siitä, kuinka hyvän ja nopean käyttökokemuksen PWA-sovellus antaa käyttäjilleen. Saavutettavuus kertoo PWA-sovelluksesta sen, kuinka hyvin se palvelee eritysasemassa olevia henkilöitä. Parhaat käytännöt kertovat PWA-sovelluksessa olevista virheistä ja tietoturvariskeistä. SEO (Search Engine Optimation) tarkoittaa suomennettuna hakukoneoptimointia. Se kertoo, kuinka hyvin PWA-sovellus on löydettävissä. (Sormunen, 2020)

Kuva 9. Esimerkki Lighthouse-raportista



3.4.2 Hyödyt ja haitat

PWA:n etuna ovat sen halvat kustannukset. Käyttäjien on helppo löytää sovellukset, sillä niitä ei tarvitse etsiä sovelluskaupasta. PWA pystyy hyödyntämään mobiililaitteen natiiveja ominaisuuksia. Jos sivusto tarjoaa PWA-sovellusta, sen voi lisätä puhelimen kotivalikkoon muiden sovelluksien joukkoon. (Alajoki, 2020)

PWA:ssa on muutamia heikkouksia. Sovelluskauppojen sovelluksista voi pyytää maksua, mutta PWA-sovelluksesta ei voi. Suorituskyky on heikompi verrattuna natiiviin sovellukseen. iOS-laitteilla ei ole mahdollisuutta saada Push-ilmoituksia. Tämä ongelma korjaantuu varmasti ajan myötä. (Alajoki, 2020)

3.4.3 Tuki

Suuret yritykset ovat viimeisten vuosien aikana kehittäneet tukea PWA:lle. Mukana kehittämisessä ovat Apple, Samsung, Microsoft ja Google. Tällä hetkellä lähes kaikki selaimet tukevat PWA:n offline-ominaisuuksia. Tällä hetkellä Androidilla on parhaimmat tuet PWA:lle

selaimissa (Kuva 10). Apple on lähtenyt PWA:n kehittämiseen hieman muita yrityksiä myöhemmin, ja tämä näkyy kehitteillä olevissa ominaisuuksissa. (Poot, n.d.)

Kuva 10. Järjestelmien tuki PWA:lle vuonna 2020 (Poot, n.d.)

3.5 Tekniikoiden vertailu

Natiivi- ja hybridisovellus tarjoavat pääsyn laitteen ominaisuuksiin. Hybridisovellus tarvitsee ’plugineita’ toimiakseen. Plugin tarkoittaa ohjelmistolaajennusta, jonka avulla voidaan parantaa ohjelman ominaisuuksia. (Computer Hope, 2020) Web-sovellus tarjoaa rajoitetut

mahdollisuudet päästä laitteen ominaisuuksiin. Suorituskyvyltään natiivisovellus on vakain, mutta myös web-sovellus ja hybridisovellus tarjoavat keskinkertaisen tai korkean suorituskyvyn. Ohjelmointikieli natiivissa sovelluksessa on alustakohtainen, kun web-sovellus ja hybridisovellus luodaan verkkotekniikoiden avulla. Samalla hybridi- ja web-sovellukset ovat alustariippumattomia. Käyttäjäkokemus on suorituskyvyn tavoin paras natiivisovelluksessa. Natiivin sovelluksen koodia ei voida käyttää uudelleen, mutta muut tekniikat mahdollistavat sen. (Kuva 11)

Kuva 11. Mobiilisovellustekniikoiden ominaisuudet (Cumulations, 2020)

Ominaisuus	**Natiivisovellus**	**Hybridisovellus**	**Web-sovellus**
Pääsy laitteeseen	Täysi	Täysi (pluginien kanssa)	Rajoitettu
Suorituskyky	Korkea	Keskinkertaisesta korkeaan	Keskinkertaisesta korkeaan
Ohjelmointikieli	Alustakohtainen	HTML, CSS, Javascript	HTML, CSS, Javascript
Monialusta tuki	Ei	Kyllä	Kyllä
Käyttäjä kokemus	Korkea	Keskinkertaisesta korkeaan	Keskinkertaisesta korkeaan
Koodin uudelleen käyttö	Ei	Kyllä	Kyllä

3.5.1 PWA vs. natiivisovellus

PWA:ta ja natiivia sovellusta verrataan tässä kappaleessa, koska ne näyttävät käyttäjälle hyvin samalta päätelaitteessa. Kappaleessa käydään näiden kahden tekniikan eroja toisiinsa verrattuna.

PWA-sovellus on halvempi tehdä kuin natiivisovellus. Natiiviin sovellukseen tarvitaan enemmän erillisiä koodeja, jos sovellus aiotaan julkaista useammalla kuin yhdellä alustalla. PWA-sovelluksen tekeminen on nopeampaa ja päivittäminen helpompaa. PWA-sovelluksen julkaiseminen on helpompaa kuin natiivin sovelluksen, sillä sitä ei tarvitse julkaista sovelluskaupoissa. PWA-sovelluksen voi hakukoneoptimoida, sillä se toimii samalla tavailla kuin muutkin verkkosivustot. Natiivisovelluksessa tietoturva on luotettavammalla tasolla kuin PWA-sovelluksessa. PWA-sovellus käyttää kuitenkin HTTPS-protokollaa, joka on turvallinen käyttäjälle. Natiiviin sovellukseen luotetaan enemmän, sillä sovelluksen on läpäistävä turvallisuusvaatimukset. Molemmat tekniikat ovat suorituskyvyltään hyviä, mutta natiivisovellus on PWA-sovellusta suorituskykyisempi. (Luong, 2021)

4 Kestävä kehitys mobiilisovelluskehityksessä

Kestävällä kehityksellä tarkoitetaan sitä, että nykyhetken taloudelliset, ympäristölliset ja sosiaaliset tarpeet pystytään täyttämään ilman, että seuraavat sukupolvet kärsivät siitä. (Team Compile, 2019) ICT-ala luo paljon uusia ympäristöhyötyjä, kun tuotteet korvataan palveluilla, liikkuminen vaihdetaan etäyhteyksiin ja teollisia tuotantoprosesseja tehostetaan. ICT-alan ratkaisut, kuten viestintäverkot, datakeskukset ja älylaitteet, tarvitsevat sähköä ja materiaaleja ja täten tuottavat paljon kasvihuonepäästöjä. ICT-ala kokonaisuudessaan käyttää 4–10 prosenttia koko maailman sähköenergiasta. Tämä määrä on merkittävän suuri. Älykkäillä laitteilla voidaan varmistaa, että ympäristönormit toteutuvat. ICT-alan vaikutus ilmastonmuutokseen on suuri, vaikka se toistaiseksi synnyttää positiivisia ja negatiivisia vaikutuksia ympäristöön. (Liikenne- ja viestintäministeriö, 2020) Suuret pilvipalveluiden tuottajat yrittävät jatkuvasti pienentää omaa hiilijalanjälkeään. Yritykset, kuten Amazon ja Google, ovat kertoneet käyttävänsä uusiutuvaa energiaa sähkön tuottoon. (Team Compile, 2019)

Sovellusten kehittäminen tapahtuu sähköisesti, joten se ei synnytä varsinaisesti jätettä ympäristöön. Ohjeet ovat sähköisessä muodossa, joten niitä ei tarvitse tulostaa. Sovellusten kehittäminen ja ylläpito vaativat sähköä toimiakseen. Low-code-kehitysympäristöt ovat pitkälti pilvipalveluja, joten myös niiden ylläpito vaatii sähköä. Älylaitteiden nopea kehitys on johtanut siihen, että älylaitteita uusitaan nopeassa syklissä ilman sen välttämätöntä uusimisen tarvetta. Älylaitteissa kierrätysaste on hyvin pieni. (Liikenne- ja viestintäministeriö, 2020). Alustariippumattomat sovellukset auttavat siinä, että älylaitteita ei välttämättä tarvitsi vaihtaa niin usein.

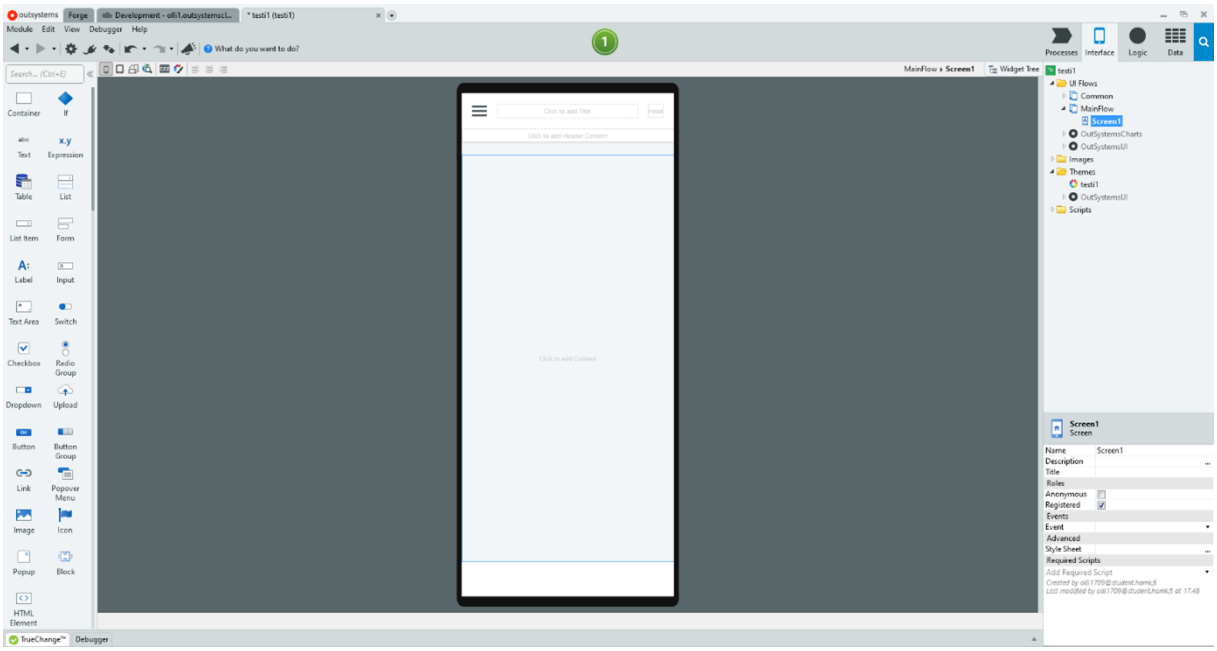
5 Kehitysympäristöt

Tässä kappaleessa tutustutaan muutamaan erilaiseen kehitysympäristöön ja niiden tarjoamiin lisäosiin. Tutkittavat kehitysympäristöt ovat Outsystems, Android Studio, Xcode, AppGyver. Ionic on oikeastaan sovelluskehys ja sitä avataan myös hieman oman otsikon alla. Näistä AppGyver ja Outsystems ovat ns. low-code kehitysympäristöjä. Xcodessa ja Android Studiossa pystyy luomaan natiiveja sovelluksia. Tässä luvussa avataan kehitysympäristöjä tarkemmin ja jokaista kehitysympäristöä kommentoidaan soveltuvuudeltaan.

5.1 Outsystems

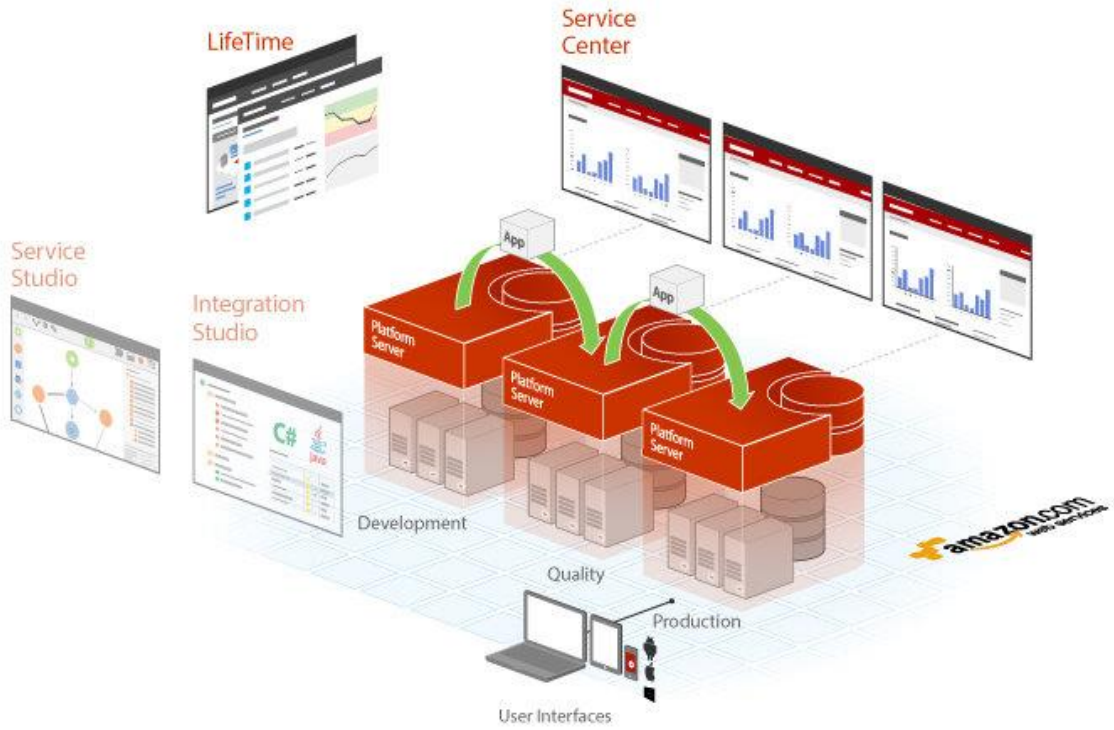
Outsystems on pilvipohjainen low-code kehitysympäristö mobiili- ja verkkosovellusten tekemiseen. (Kuva 12) .NET on ohjelmointikieli, jonka avulla Outsystems-järjestelmä on luotu. Käyttäjän ei kuitenkaan tarvitse osata tätä ohjelmointikieltä luodakseen sovelluksia tässä ympäristössä. Outsystems on maailmanlaajuisesti käytetty kehitysympäristö. (Valuga, n.d.)

Kuva 12. Outsystems käyttöliittymä



Nämä kaikki palvelut kuuluvat Outsystems-tilaukseen. (Kuva 13) Service Studio on varsinainen kehitysympäristö, missä sovellukset luodaan. Service Studiossa on mahdollista tehdä web- tai mobiilisovelluksia. Service Studio auttaa kehittäjää jatkuvalla virheenkorjauksella. (Outsystems, n.d.-a) Integration Studion avulla pystyy hallitsemaan ja luomaan laajennuksia sovellukseen. (Outsystems, n.d.-b)

Kuva 13. Outsystems palvelun rakenne (Outsystems, n.d.-c)



Outsystems toimii applikaationa selaimen kautta, joka mukautuu käytettävän laitteen koon ja ominaisuuksien mukaan. Selaimen kautta käytettävyyden ansiosta laitteelle ei tarvitse ladata erillisiä ohjelmia. Ilman internet-yhteyttä applikaatio ei toimi ilman kolmannen osapuolen komponentteja. Tavallisen web-applikaation taustalla pyörii monia toisiaan tukevia tekniikoita, jotka mahdollistavat kehityksen. Outsystems tarjoaa esimerkiksi työkaluja käyttöliittymien, tietokantojen, REST- ja SOAP-web-palveluiden käyttämiseen sovelluksissa. Kaikkiin Outsystemsin työkaluihin pääsee käyttäen Outsystems Serviceä. (Salo, 2017)

Kehitysympäristöön vaaditaan kirjautumista, jotta applikaation kehittämisen voi aloittaa. Kehitysympäristö sisältää serverille luotuja applikaatioita sekä luomaan uusia sellaisia, mutta myös asentamaan Forgesta muiden käyttäjien luomia applikaatioita. Forge on komponenttivarasto, johon Outsystems tai käyttäjät voivat lisätä omia laajennuksia. (Salo, 2017)

Outsystems-jäsenyys on helposti kaikkien käyttäjien saatavilla. Kehitysympäristönä Outsystems on helppokäyttöinen ja monipuolinen. Tämä ympäristö sopisi hyvin opetuskäyttöön.

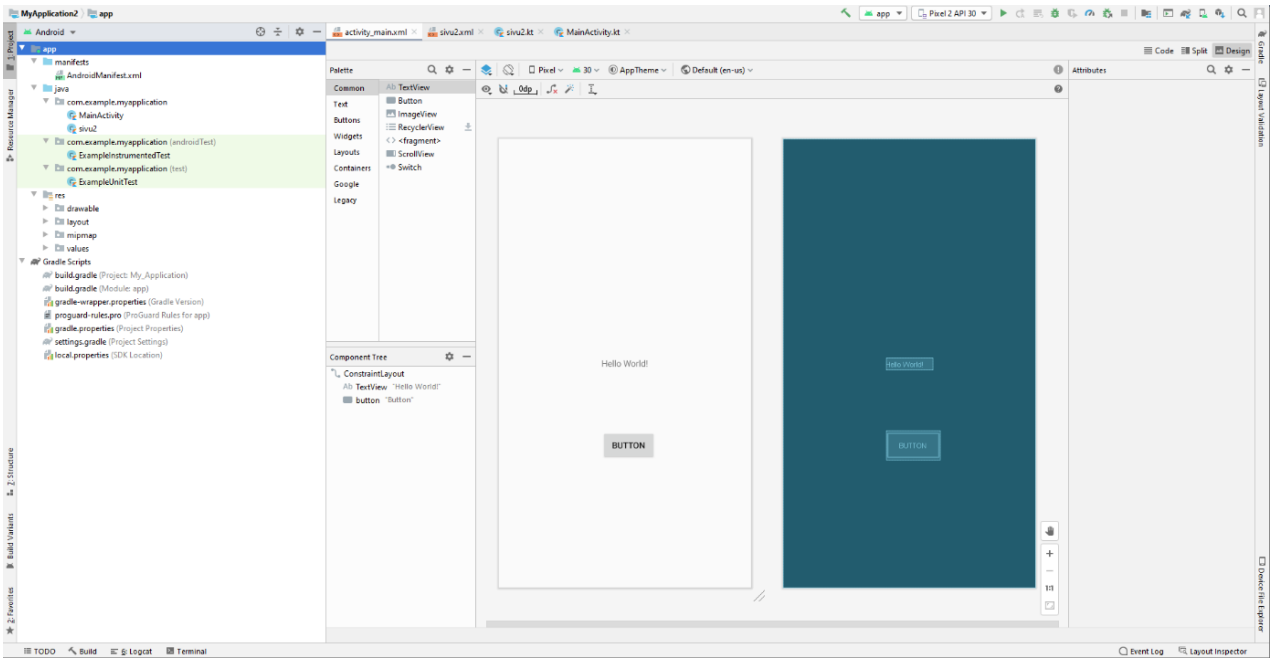
5.2 Android Studio

Android Studio on integroitu kehitysympäristö Android-sovellusten kehittämiseen. (Kuva 14) Sovellusten kehittäminen perustuu IntelliJ IDEA:n. Android studio tarjoaa yhä enemmän toimintoja, jotka lisäävät tuottavuutta luodessa Android-sovellusta. Android Studiossa on enemmän ominaisuuksia, jotka parantavat sovelluksen tuottavuutta. Ominaisuuksia ovat muun muassa nopea emulaattori. Ympäristö mahdollistaa kehittämisen kaikille Android-laitteille ja monipuoliset testaustyökalut. (Developers, n.d.-a)

Kotlin on noussut Android Studion ykköskielen asemaan. Kehittäjän on kirjotettava vähemmän koodia ja ylläpidettävän koodin määrä on pienempi verrattuna Javaan. Java ohjelmointikielenä ei ole häviämässä mihinkään, vaan Google tukee myös sitäkin jatkossa. (Kolehmainen, 2019)

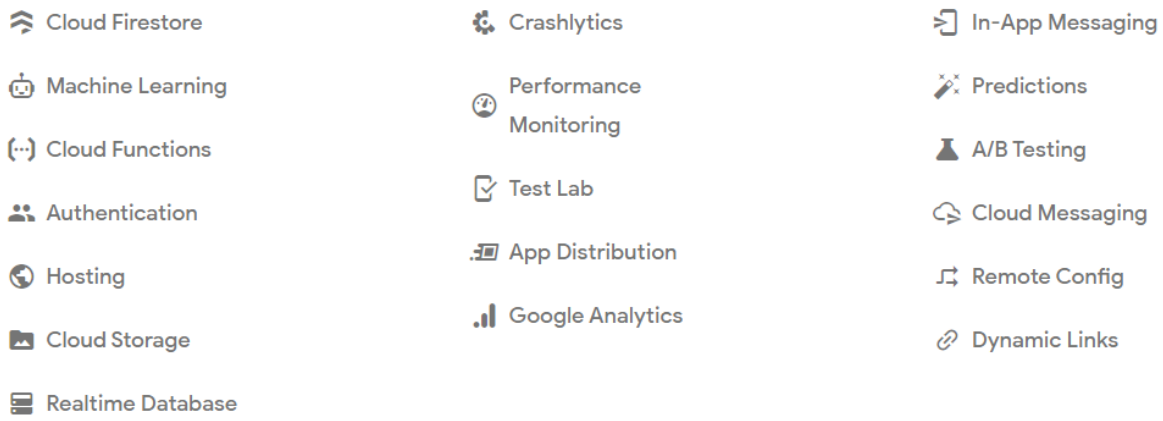
Sovelluksen simulointi tapahtuu Android Emulaattorin avulla. AVD-managerin avulla voi asentaa mieleisen emulaattorin Android Studioon. Emulaattori tarjoaa käytännössä kaikki samat ominaisuudet kuin oikea puhelinkin. Emulaattorin avulla voi simuloida, vaikka saapuvia puheluita tai tekstiviestejä. (Developers, n.d.-b)

Kuva 14. Android Studio käyttöliittymä



Firebase on Googlen kehittämä työkalu, jonka avulla pystyy tehostamaan ja ylläpitämään Android Studiossa luotuja sovelluksia. Firebase tarjoaa paljon sovelluksia tehostavia ja parantavia ominaisuuksia. (Kuva 15)

Kuva 15. Firebase ominaisuudet (Firebase, n.d.)

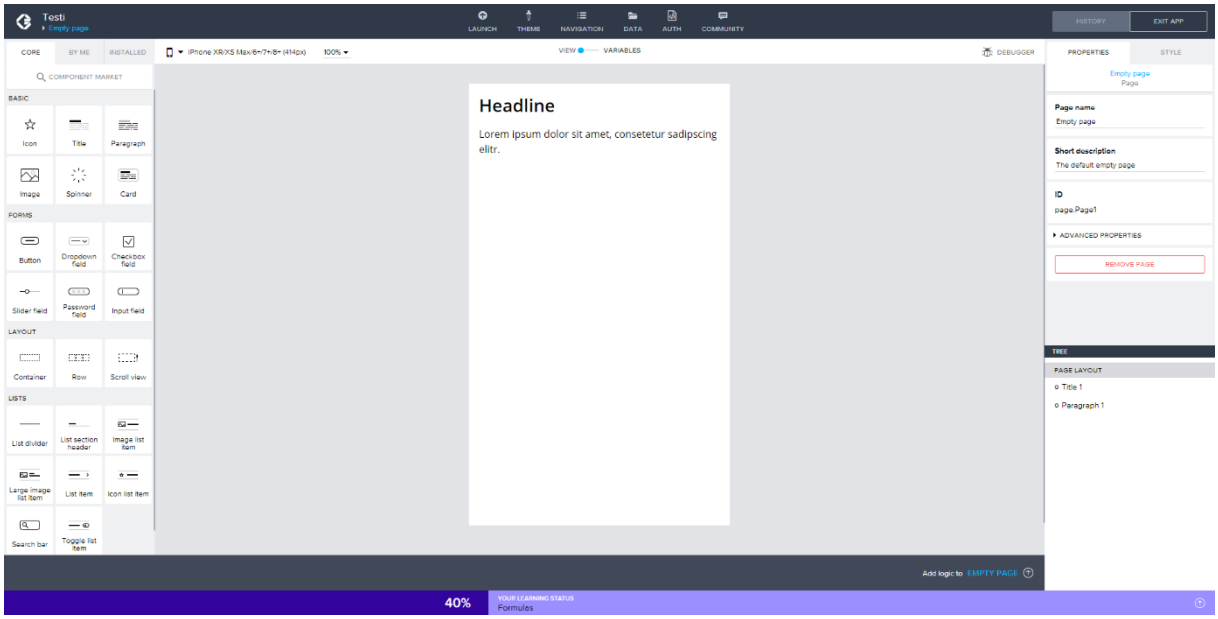


Android Studion avulla on mahdollista luoda hyvinkin vaikuttavia sovelluksia. Ohjelmointikielen opettelu vie merkittävän määrän aikaa, joten siinä mielessä se ei optimaalisin tapa tehdä sovelluksia opetustilanteissa.

5.3 AppGyver

AppGyver on low-code kehitysympäristö, jonka avulla voi tehdä sovelluksia monille eri alustoille. (Kuva 16) AppGyver luo sovellukset käyttäen React Nativea. React Native on Javascript-kehys, jonka avulla voi luoda natiivin tuntuisia sovelluksia. (O'Reilly, n.d.)

Kuva 16. AppGyver käyttöliittymä



AppGyver on täysin ilmainen kehittäjille ja yrityksille, joiden liikevaihto on alle 10 miljoonaa dollaria. Ilmaistilaukseen sisältyy sovellusten julkaiseminen sovelluskaupoissa tai verkossa. (AppGyver, n.d.-a)

AppGyverin käyttö perustuu drag and drop-toimintoihin. Näiden toimintojen avulla on helppo luoda käyttöliittymä sovellukselle. Visuaaliset elementit ovat helposti muokattavissa. Sovelluksen toiminta logiikka luodaan täysin visuaalisesti AppGyver:issa. (AppGyver, n.d.-b)

Component Marketplace on paikka, mistä käyttäjät voivat ladata erilaisia komponentteja, teemoja tai toimintoja. AppGyver on itse luonut Marketplaceen komponentteja, mutta myös AppGyverin käyttäjillä on mahdollisuus lisätä teoksiaan sinne. (AppGyver, n.d.-c)

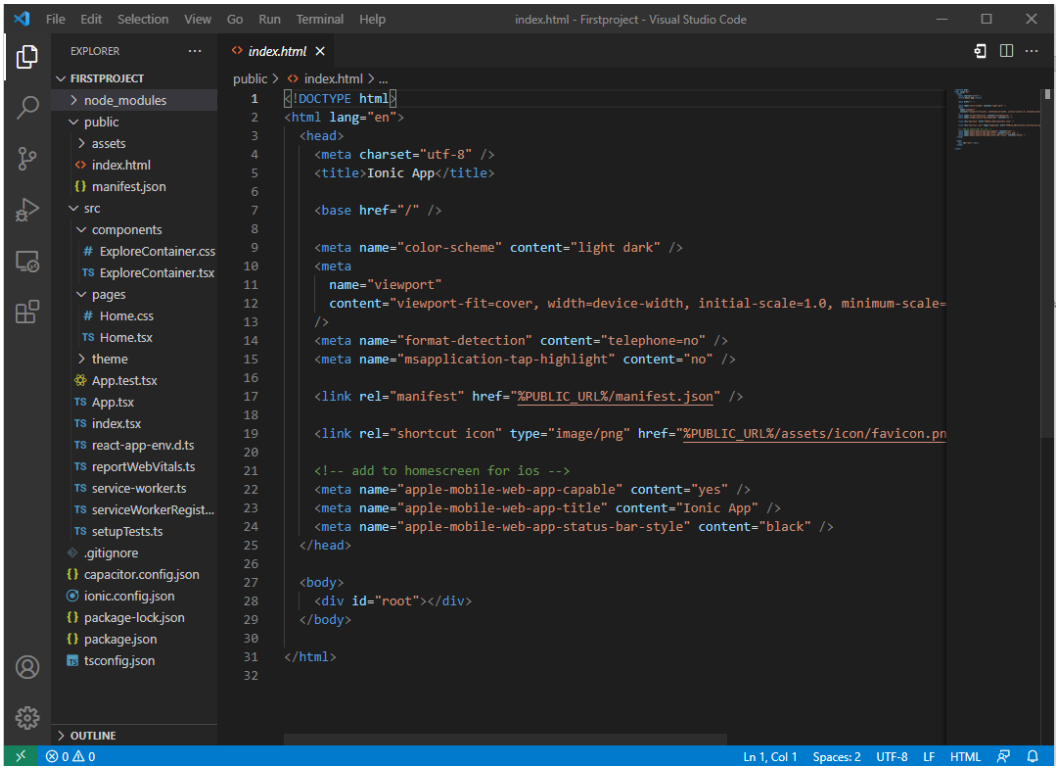
AppGyver on selainpohjainen kehitysympäristö, joten se on helposti kaikkien käyttäjien saatavilla. Ohjelmointi osaamista ei tarvita juuri lainkaan, että kehitysympäristöllä saa luotua sovelluksen. AppGyver on hyvä valinta kaiken tasoisille käyttäjille.

5.4 Ionic

Ionic on lähtöisin vuodesta 2012, kun natiivien sovellusten tekeminen verkkotekniikoilla oli vielä alkutekijöissä. Tarkoituksena oli luoda helppo ja parempi tekniikka käyttäjilleen luoda sovelluksia. Ionic on nykyään suosituin alusta mobiilisovelluskehityksessä sen moninaisen toimivuutensa ansiosta. (Ionic, n.d.-a)

Ionic projektin rakenne on selkeä. (Kuva 17) Visual Studio Code on yleisesti käytetty ympäristö Ionic projekteissa. Visual Studio tarjoaa tuen ES-6-syntakseille ja TypeScript-tuen. Visual Studio on kaikille alustoille saatavilla, mikä lisää suosiota käyttäjien keskuudessa. (JavaTpoint, n.d.)

Kuva 17. Ionic projekti Visual Studio Codessa



Ionic on HTML5-ohjelmistokehys, jolla luodaan hybridisovelluksia. Hybridisovellukset ovat käytännössä pieniä verkkosivuja, jotka käyttävät selainliittymää ja pääsevät natiiviin kerrokseen. Hybridisovelluksen saa luotua, kun HTML5-kehys paketoidaan Cordova- tai PhoneGap-kääreeseen. (Ionic, n.d.-b)

Ionicin avulla ohjelmoidut sovellukset sopivat parhaiten yksinkertaisiin käyttöliittymäratkaisuihin. Natiiviin sovellukseen verrattuna Ionic on hitaampi ja käyttöliittymä eroaa natiivista. Ionic on helppokäyttöinen väline ja yksi parhaista välineistä toteuttaa web-tekniikoilla sovellus sovelluskauppaan. (JAMK, n.d.)

Ionic Native on kirjasto, jonka avulla mobiililaitteen natiiveihin ominaisuuksiin päästää täysin käsiksi. Lisäosat ovat täysin ilmaisia ja kaikkien saatavilla. Ionicin avulla on mahdollisuus tehdä myös PWA-sovelluksia. (JAMK, n.d.)

Ionic on hyvin monipuolinen sovelluskehys. Ionicilla on laaja käyttäjäkunta, joten paljon erilaisia demoja on kehittäjien saatavilla. Sitä on hyvä käyttää, jos on aikomusta luoda hybridisovelluksia. Ionic sopii kaiken tasoisille käyttäjille.

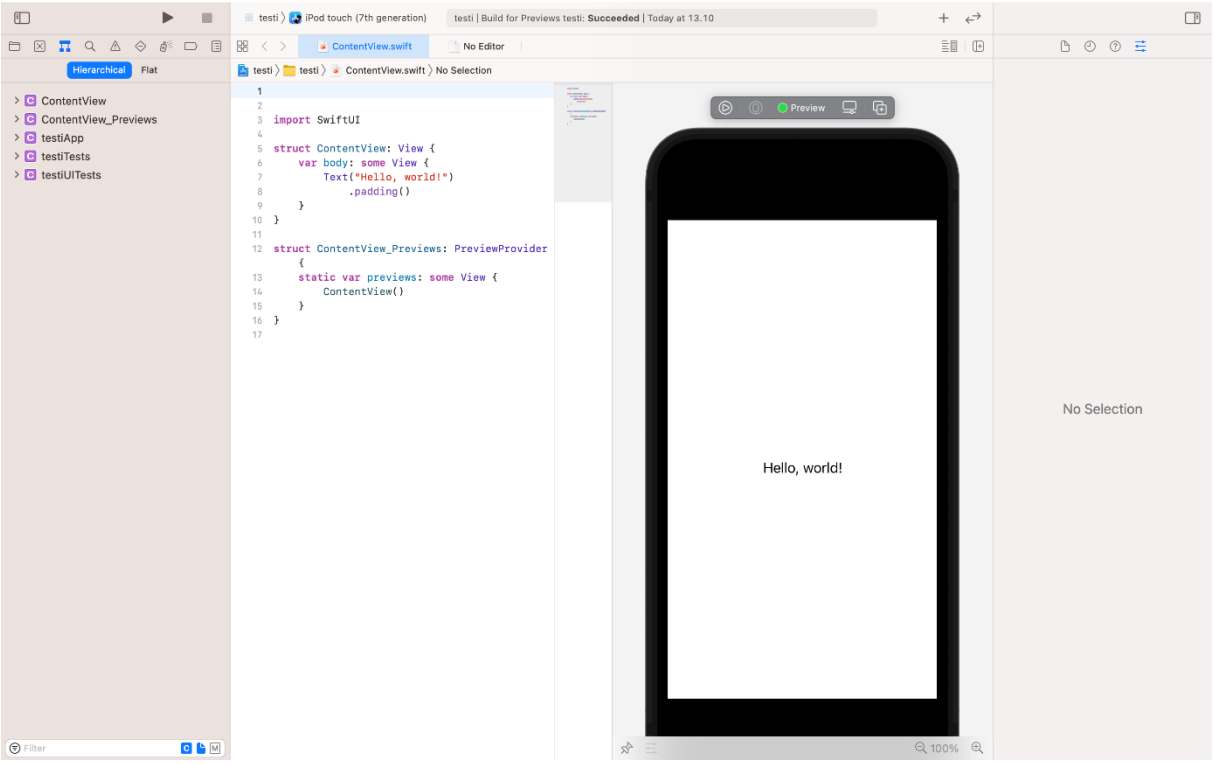
5.5 Xcode

Xcode on Applen luoma IDE (Integrated development environment), missä voi luoda ohjelmistoja iOS:lle, iPadOs:lle, watchOS:lle ja tvOS:lle. (Kuva 18) Xcodella tehdyt sovellukset ovat tuettuja Applen puolesta. iOS-sovelluksia on mahdollista myös tehdä kolmannen osapuolten ratkaisuilla, mutta Apple ei tue tällöin niitä. Xcode sisältää kaikki tarvittavat työkalut, joita tarvitaan sovelluksen luomiseen. Xcode sisältää tekstieditorin, kääntäjän ja koontijärjestelmän. Xcodesta on mahdollista julkaista valmis sovellus suoraan App Storeen. (Chung, n.d.)

Xcode käyttää ohjelmointikielenään Swiftiä ja Objetive-C:tä. Swift on moderni ohjelmointikieli ja sen on tarkoitus korvata Objetive-C-ohjelmointikieli. Swift-ohjelmointikielen oli tarkoitus tuoda tehokkuutta ja selkeyttä ohjelmointiin Xcodessa. Kun Xcodessa käytettäviä ohjelmointikieliä verrataan keskenään, niin Swift on helpommin

opittavissa. Swift-ohjelmointikieli sopii myös ohjelmoinnista vähän tietävälle kehittäjälle. (Hynninen, n.d.)

Kuva 18. Xcode käyttöliittymä



Xcode on ulkoasultaan siisti ja miellyttävä. Xcode on ladattavissa vain App Storesta, joten se asettaa laiterajoituksia siten, että se vaatii Applen tietokoneen sovellusten kehittämiseen. Swift on hyvä ohjelmointikieli, jolla on hyvät tulevaisuuden näkymät.

6 Sovelluksen esikatselu ja kehitysympäristön ulkoasu

Tässä kappaleessa vertaillaan edellä mainittujen kehitysympäristöjen ulkoasua ja sovellusten esikatselua. Sovelluksen esikatselu helpottaa sovelluksen kehittämistä. Ulkoasulla on vaikutusta kehitysympäristön mielekkyyteen ja käytettävyyteen.

6.1 Ulkoasu

Käyttöliittymät muistuttavat hyvin paljon toisiaan low-code kehitysympäristöissä. Vasemmassa reunassa on hyvin usein valikko, josta voi vetää valmiita elementtejä

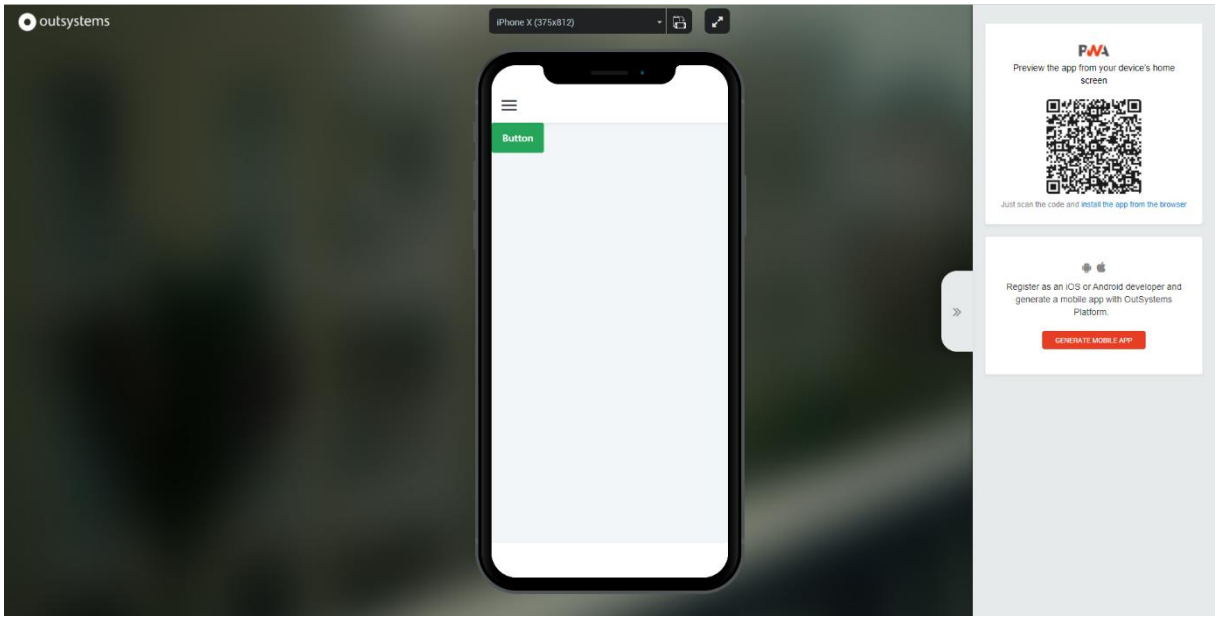
sovellukseen. Tyyliasetusten muuttaminen onnistuu Outsystemsissa ja AppGyver:issa ympäristöissä oikealta. Käyttöliittymiä ei pysty muokkaamaan haluamansa näköiseksi.

Natiivit kehitysympäristöt Xcode ja Android Studio puolestaan oletusasetuksilla näyttävät käyttöliittymältään hyvin samalta. Vasemmasta reunasta löytyvät sovellukseen liittyvät kansiorakenteet. Keskellä on yleisesti itse koodi. Android Studiossa käyttöliittymää pystyy kuitenkin muokkaamaan siten, että sieltä voi piilottaa tai siirtää ikkunoita haluamalla tavalla. Xcoden ulkoasua ei oikeastaan pysty muuttamaan, vaan osat ovat niin sanotusti staattisia.

6.2 Sovelluksen esikatselu

Sovelluksen esikatselu tapahtuu AppGyver:issa itse selaimesta tai puhelimeen ladattavasta sovelluksesta, joka on nimeltään AppGyver Preview. AppGyver Previewiin kirjaudutaan samalla tilillä kuin selain versioon. Sovelluksessa näkyvät kaikki käyttäjätiliin liitetyt projektit. Service Studiossa 1-click-publishin avulla saadaan testata sovellusta. Outsystems tarjoaa selaimessa aukeavan emulaattorin, jossa voi kokeilla sovelluksen toimivuutta. PWA-sovelluksen saa puhelimeen kokeiltavaksi QR-koodin avulla. (Kuva 19)

Kuva 19. Outsystems emulaattori



Android Studiossa sovelluksen esikatselu toimii erillisesti ladattavalla Android-emulaattorilla tai suoraan fyysisessä Android-laitteessa. Android-emulaattori täytyy ensiksi asentaa

Android Studiossa, jotta sitä voi käyttää esikatselussa. Xcodesta emulaattori aukeaa automaattisesti, kun painetaan vasemmasta yläreunasta Play-näppäintä.

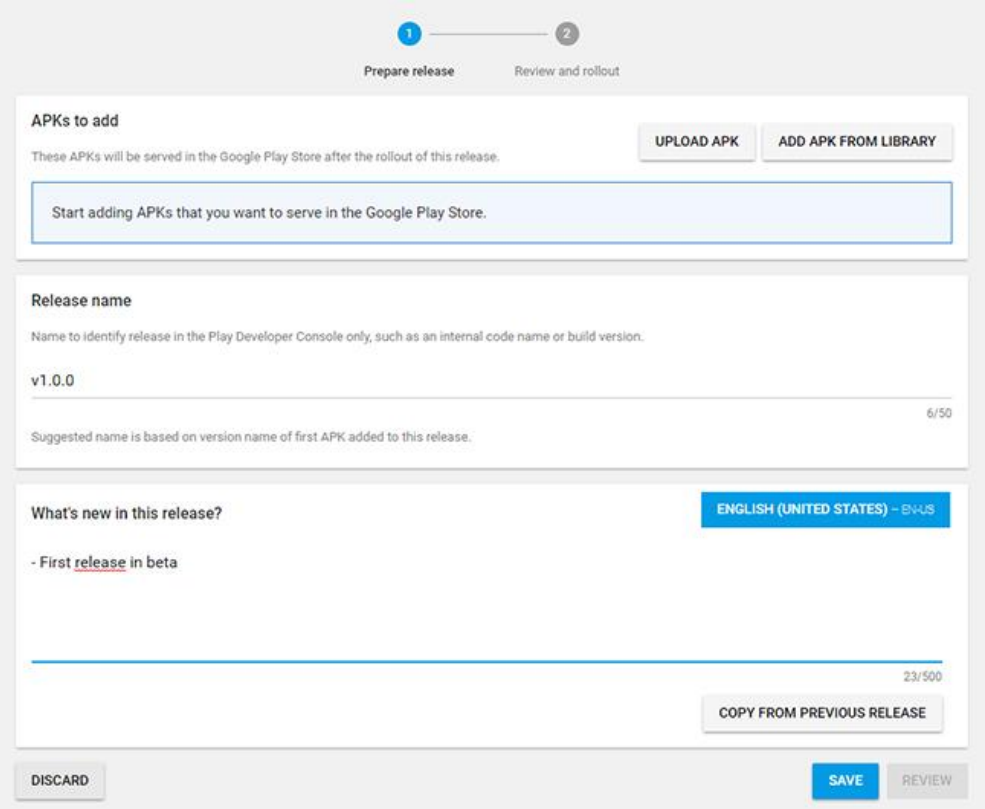
7 Julkaiseminen sovelluskaupassa

Sovelluksen julkaiseminen on osa sovelluskehitystä. Tässä kappaleessa käydään läpi sitä, miten sovellus julkaistaan alustakohtaisessa sovelluskaupassa. Sovelluksen julkaiseminen on rajattu kahteen suosituimpaan sovelluskauppaan eli App Store ja Google Play-kauppa. Julkaistavat sovellukset tarvitsevat molemmissa sovelluskaupoissa digitaalisen allekirjoituksen, ilman tätä sovellusta ei ole mahdollista julkaista. Allekirjoitus takaa, että kehittäjät ovat luotettavia. (Codemagic, 2020)

7.1 Sovelluksen julkaiseminen Google Play-kaupassa

Sovellusten julkaiseminen Google Play-kaupassa vaatii kehittäjätilin, joka pitää luoda ensimmäisenä. Kehittäjän täytyy hyväksyä jakelusopimus, jotta voi siirtyä maksuvaiheeseen. Kehittäjätilin avauksessa veloitetaan kertaluontoinen 25 dollarin maksu. Jos aikoo myydä sovelluksen sisäisiä palveluita tai sovellus on maksullinen, niin silloin täytyy luoda kauppiastili. Täydennettävät tilitiedot on hyvä antaa rekisteröitymisen alkuvaiheessa, mutta se onnistuu myöhemminkin. Täysi rekisteröityminen saattaa kestää jopa 2 päivää.
Google Play Consolesta voi luoda uuden sovelluksen. Siinä kohtaa sovellukselle määritetään kieli ja otsikko. Tiedot ovat vaihdettavissa myöhemmin. Sovelluksen otsikko ja lyhyt sekä täysi kuvaus sovelluksesta on syötettävä Google Play Consoleen. Sovelluskuvaus näkyy Google Play:n käyttäjille sovellussivulla. Kuvankaappaukset ja laitekuvat lisätään graafiset resurssit kohdassa. Sovellus täytyy seuraavaksi luokitella. Kehittäjän on lisättävä yhteystiedot, mistä käyttäjät saavat tukea sovelluksen käyttöön liittyen. Jos sovellus kerää henkilöihin liittyvää dataa, niin silloin täytyy luoda tietosuojakäytäntö, josta selviää miten sovellus kerää, jakaa ja käyttää dataa. Sovelluksen APK-tiedosto lisätään Google Play Consoleen. (Kuva 20) APK-tiedosto (Android Package Kit) on Androidin käyttämä tiedostomuoto, jolla jaetaan ja asennetaan sovellukset. (Oragui, 2018)

Kuva 20. APK-tiedoston lataaminen (Oragui, 2018)



Google Play voi poistaa sovelluskaupastaan luokittelemattomia sovelluksia, joten on hyvä tehdä sovellukselle sisältöluokitus. Sisällön väärentäminen saattaa johtaa sovelluksen poistamiseen sovelluskaupasta. Sovellus täytyy määritellä maksulliseksi tai ilmaiseksi. Kun kaikki edellä mainitut asiat on suoritettu, niin sovelluksen voi lähettää arvioitavaksi. (Oragui, 2018)

7.2 Sovelluksen julkaiseminen App Store-kaupassa

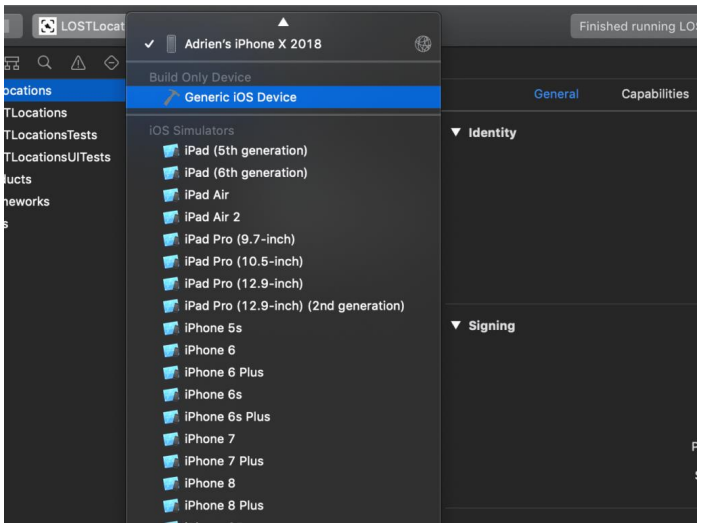
Apple tarjoaa App Store Review Guideline-listan, jonka avulla voi varmistaa, että sovellus vastaa vaadittavia ehtoja. Guideline-listan avulla Apple tarkastaa sovellukset. Apple mainitsee dokumentissa, että se on elävä. Tällä tarkoitetaan sitä, että uudet sovellukset voivat luoda uusia sääntöjä listalle. Jos järjestelmään yrittää huijata eri tavoilla, kuten käyttäjätietojen varastaminen tai kopioida muiden töitä, tämä johtaa automaattisesti sovelluksien poistoon sekä kehitysohjelmasta erottamiseen. (Apple Developer, n.d.)

Sovelluksen julkaisussa on viisi kohtaa, joiden mukaan täytyy edetä. Ensimmäisenä on rekisteröityminen Apple Developer-ohjelmaan. Se maksaa vuodessa 99 dollaria. Vuosimaksun maksamatta jättäminen johtaa julkaistujen sovellusten poistoon. Vuosimaksulla saa etuja mm. testityökalut, sovellusanalytiikka ja pääsy Apple-ohjelmiston beetaversioihin. Tämän jälkeen yhdistetään tili App Store Connectiin. Omaa toimintaa voi hallita ja seurata App Store Connectissa. Sovellus on tässä kohtaa hyvä testata, että siinä ei ole virheitä tai bugeja. Sovelluksen hylkäyksen syy voi olla siinä. (Ching, 2019)

App Store Connectissa omat sovellukset-valikosta löytyy +-merkki, jota painamalla päästään syöttämään sovelluksen tietoja. Täytettäviä tietoja on nimi, sovelluksen kieli, Bundle-ID sekä SKU-numero. Apple vaatii itselleen testaukseen tilitietoja, jos sovelluksessa on mahdollista kirjautua. Tietosuojakäytännöille on lisättävä URL-osoite. Sovelluksen hinta päätetään, onko ilmainen vai maksullinen. Sovellukseen voi asettaa haluamansa julkaisupäivän, jolloin sovellus julkaistaan, jos se on päässyt läpi Applen tekemästä tarkastuksesta. (Ching, 2019)

Sovelluksesta on lisättävä näyttökuvia ja laitekuvia. Mikäli sovellus julkaistaan monelle eri Applen laitteelle, niin jokaisesta on lisättävä laite- ja sovelluskuvat. Sovellukset ladataan App Storeen suoraan Xcodesta tai Application loaderin avulla. Xcodesta simulaattori valikosta valitaan Generic iOS Device. (Kuva 21)

Kuva 21. Sovelluksen generointi (Ching, 2019)



Xcode luo sovelluspaketin (.ipa), missä voi kestää hetki tiedostokoosta riippuen. Ipa-paketti on Applen käyttämä tiedostomuoto, jota käytetään sovellusten jakamiseen ja lataamiseen.

Automaattisesti paketin luonnin jälkeen aukeaa arkisto luoduista sovelluksista. Valikosta valitaan julkaistava sovellus, sitten jaa sovellus. Xcode kysyy, miten sovellus halutaan julkaista. Muita jako tapoja ovat asennus suoraan laitteeseen (Ad Hoc), yrityksen sisäinen jako (Enterprise) tai oman tiimin sisäinen jako (Development). Jos Xcode ei löydä vikaa sovelluksesta, niin se on mahdollista ladata App Storeen. Seuraavana vaiheena on sovelluksen lähettäminen Applelle tarkastettavaksi. Sovelluksen hyväksymisestä julkaisija saa sähköpostiviestin. (Ching, 2019)

8 Pohdinta ja johtopäätökset

Sovelluksen suunnittelussa on tärkeä miettiä, mihin käyttötarkoitukseen sovellus tulee ja mitä osaamista kehittäjältä löytyy. Sovelluksen tekemiseen ei ole oikeaa ja väärää tekniikkaa, mutta tässä työssä oli tarkoituksena selvittää sopiva kehitystekniikka opetuskäyttöön, johon parhaiten tähän tutkimisen perusteella sopivat PWA-tekniikka ja low-code-kehitysympäristöt. Opetuksessa on tärkeä lähteä liikkeelle helpoista ja yksinkertaisista asioista, jotta opiskelijat saavat onnistumisen kokemuksia tekemästään työstä. Näissä kahdessa tavassa tämä toteutuu hyvin.

Natiivisovelluksen tekeminen rajaa käyttäjiä, jos ei ole mahdollista tehdä useille alustoille sovellusta. Swift-ohjelmointikieli on saanut paljon kehuja kehittäjiltä, mutta täytyy muistaa, että silloin täytyy olla tietokoneessa macOS-käyttöjärjestelmä. Tämä vaatii paljon resursseja ja osaamista opiskelijoilta, joten tämä ei ole paras mahdollinen ratkaisu. Hybridisovellus jää hieman PWA-sovelluksen varjoon ominaisuuksiltaan ja hybridisovelluksen tekemiseen vaaditaan vähän osaamista natiiveista ohjelmointikielistä. Web-sovellus ei ole yhtä monipuolinen kuin PWA-sovellus. PWA-sovellus tarjoaa natiivin ja web-sovelluksen parhaat puolet sekä se on täysin alustariippumaton. Tulevaisuudessa yritykset ottavat enemmän käyttöön low-code-ympäristöjä, jotta sovelluskehitystä saadaan vieläkin nopeammaksi. Siinä mielessä low-code-kehitysympäristöjen hallinnasta on hyötyä tulevaisuudessa. Low-code-kehitysympäristöt kehittyvät ominaisuuksiltaan vauhdikkaasti.

Minulla ei ollut juurikaan ennen opinnäytetyötä kokemusta sovelluskehityksestä. Olen suorittanut kuuden opintopisteen kurssin, missä kehitettiin muutama yksinkertainen sovellus Android Studiolla. Silloin koin itse, että kehittäminen on haastavaa. Olen suorittanut

myös verkkotekniikoiden perusteet. Natiiveja ohjelmointikieliä on mahdollista oppia käyttämään, mutta aloittelevalle kehittäjälle se ei ole välttämättä paras ja helpoin ratkaisu. Tämän päivän teknologiat mahdollistavat sovellusten tekemisen muillakin tavoilla. Opinnäytetyön aikana loin sovelluksen low-code ympäristössä harjoitus mielessä ja sain suunniteltua toimivia ominaisuuksia sovellukseen muutamassa päivässä. Eniten kehityksessä innosti se, että sovelluksen sai hyvin nopeasti toimimaan ja sitä pääsi heti testaamaan suoraan puhelimessa.

Ehdin hieman tutkia muitakin kuin valitsemiani kehitysympäristöjä opinnäytetyön aikana. Muita mielestäni hyviä low-code-kehitysympäristöjä ovat Bubble.io ja Thunkable. Opinnäytetyön loppupuolella ohjaani ehdotti Thunkable-kehitysympäristöä. Tutustuin itse ympäristöön hyvin nopeasti, ja se vaikutti hyvin monipuoliselta ja yksinkertaiselta käyttää. Logiikka luodaan samaan tyyliin kuin Scratch-sovelluksessa. Integraatiomahdollisuuksista löytyi muun muassa Firebase.

Jatkokehitysideana voisi olla jonkin sovelluksen luominen ehdottamallani tekniikoilla. Vaihtoehtoisesti toinen opiskelija voisi tutkia tarkemmin muita kehitysympäristöjä, joita en itse tutkinut tämän opinnäytetyön aikana syvällisemmin. Opinnäytetyöni on pääasiassa painottunut tekniikoiden ja kehitysympäristöjen tutkimiseen, joten tämä voisi olla hyvä aihe toiselle opiskelijalle.

Nykyinen sähkönkulutuksen taso pyritään huomiomaan paremmin ICT-alalla, jotta tulevaisuudesta voidaan tehdä kestävämpi. Laitteiden uusiminen on nopeaa, mikä ei ole kestävää. Toisaalta tekniikan kehittyminen vaatii uusia laitepäivityksiä, jolloin laitteita päivitetään herkemmin. Vanhoista laitteista saa monessa paikassa hyvitystä, kun ostaa uuden laitteen. Tällöin kierrätys tapahtuu oikein. Vanhat laitteet menevät jälleenmyyntiin vain, jos laitteen kunto sallii sen.

Lähteet

1Training.org. (n.d.). *How can you use HTML, CSS and JavaScript for Web Development?* Noudettu osoitteesta https://www.1training.org/blog/html-css-javascript-web-development/

Alajoki, J. (2020). *Mikä on progressiivinen verkkosovellus (PWA) ja mitä etuja se tarjoaa?* Noudettu osoitteesta https://www.evermade.fi/fi/artikkeli/mika-on-progressiivinen-verkkosovellus-pwa-edut/

AppGyver. (n.d.-a). *Pricing*. Noudettu osoitteesta https://www.appgyver.com/pricing

AppGyver. (n.d.-b). *Introducing Composer Pro*. Noudettu osoitteesta https://www.appgyver.com/

AppGyver. (n.d.-c). *Marketplace*. Noudettu osoitteesta https://docs.appgyver.com/overview/the-app-builder/left-sidebar/marketplace

Apple Developer. (n.d.). *App Store Review Guidelines.* Noudettu osoitteesta https://developer.apple.com/app-store/review/guidelines/#before-you-submit

Ayobi, A. (21.9.2017). *IKEA Launches Augmented Reality Application*. Noudettu osoitteesta https://www.architectmagazine.com/technology/ikea-launches-augmented-reality-application_o

Azorin, P. (8.9.2019). *How 5G technology will influence software development*. Noudettu osoitteesta https://medium.com/@guestposts_92864/how-5g-technology-will-influence-software-development-eef0bd239a0e

Biswas, P. (16.3.2020). *Mobile App Development – Where is the Future Headed?* Noudettu osoitteesta https://www.business2community.com/mobile-apps/mobile-app-development-where-is-the-future-headed-02292884

Ching, C. (31.10.2019). *How To Submit Your App To the App Store.* Noudettu osoitteesta https://codewithchris.com/submit-your-app-to-the-app-store/#apple-developer-program

Chung, E. (n.d.). *What is Xcode and why do I need it?* Noudettu osoitteesta https://www.zerotoappstore.com/what-is-xcode-and-why-do-i-need-it.html

Codemagic. (11.11.2020). *How to code sign & publish iOS apps.* Noudettu osoitteesta https://blog.codemagic.io/how-to-code-sign-publish-ios-apps/

Computer Hope. (6.2.2020). *Plugin*. Noudettu osoitteesta https://www.computerhope.com/jargon/p/plugin.htm#spelling

Cumulations. (9.3.2020). *Hybrid vs Native Mobile App Development: which one you should choose in 2020.* Noudettu osoitteesta https://www.cumulations.com/blog/hybrid-vs-native-mobile-app-development/

Devathon. (15.12.2020). *Low code vs. No code: Which is better for web and app development?* Noudettu osoitteesta https://medium.com/@devathon_/low-code-vs-no-code-which-is-better-for-web-and-app-development-3710aaab395f

Developers. (n.d.-a). *Android Studio*. Noudettu osoitteesta https://developer.android.com/studio/intro

Developers. (n.d.-b). *Run apps on the Android Emulator*. Noudettu osoitteesta https://developer.android.com/studio/run/emulator?gclid=EAIaIQobChMIybLfnf_Q8AIVBQWiAx0grwu_EAAYASAAEgJ1MvD_BwE&gclsrc=aw.ds

DNA. (n.d.). *5G-verkko*. Noudettu osoitteesta https://www.dna.fi/5g-verkko

Empirica. (n.d.). *Mika on IoT?* Noudettu osoitteesta https://www.empirica.fi/iot/

Firebase. (n.d.). *Firebase helps you build*. Noudettu osoitteesta https://firebase.google.com/

Grannell, C. (2021). *Best AR apps in 2021: Augmented reality comes to your phone*. Noudettu osoitteesta https://www.tomsguide.com/round-up/best-ar-apps

Gregory, S. (25.11.2019). *What is the Difference Between Web Apps, Native Apps, Hybrid Apps and Progressive Web Apps?* Noudettu osoitteesta https://hackernoon.com/what-is-the-difference-between-web-apps-native-apps-hybrid-apps-and-progressive-web-apps-py19n2gdi

Griffth, C. (n.d.). *What is Hybrid App Development?* Noudettu osoitteesta https://ionic.io/resources/articles/what-is-hybrid-app-development#h-what-is-a-native-mobile-app

Haldar, M. (1.8.2018). *What is a PWA and why should you care?* Noudettu osoitteesta https://blog.bitsrc.io/what-is-a-pwa-and-why-should-you-care-388afb6c0bad

Harnish, B. (18.12.2020). *What Is HTTPS: The Definitive Guide to How HTTPS Works*. Noudettu osoitteesta https://www.semrush.com/blog/what-is-https/?kw=&cmp=US_SRCH_DSA_Blog_Core_BU_EN&label=dsa_pagefeed&Network=g&Device=c&utm_content=484311161407&kwid=dsa-1057183187995&cmpid=11778642910&agpid=113114221383&BU=Core&extid=167385686699&adpos=&gclid=CjwKCAjw1uiE

Hindi, D. (n.d.). *15 Mobile App Development Trends of 2021*. Noudettu osoitteesta https://buildfire.com/mobile-app-development-trends/

Hynninen, T. (n.d.). *Opi tuntemaan Swift: Applen uusi ohjelmointikieli*. Noudettu osoitteesta https://teemuhynninen.fi/2014/07/opi-tuntemaan-swift-applen-uusi-ohjelmointikieli/

Hämeen Ammattikorkeakoulu. (n.d.). *Tieto- ja viestintätekniikka, biotalous, insinööri (AMK).* Noudettu osoitteesta https://www.hamk.fi/amk-tutkinto/biotalous-insinoori-amk/

Ionic. (n.d.-a). *The dev-friendly app platform for building cross-platform apps with one codebase, for any device, with the web.* Noudettu osoitteesta https://ionicframework.com/what-is-ionic?__hstc=13779304.b18b897b8d0ffb8a749a9dcb44470275.1616677490220.1620816869233.1621250202481.8&__hssc=13779304.1.1621250202481&__hsfp=3352249704&_gl=1*xvki6g*_ga*MTEzNTI1MzkwLjE2MTY2Njk4MjU.*_ga_REH9TJF6KF*MTYyMTI1MD

Ionic. (n.d.-b). *Chapter 1: All About Ionic*. Noudettu osoitteesta https://ionicframework.com/docs/v1/guide/preface.html

IQUII. (4.3.2019). *Progressive Web App (PWA): what they are, pros and cons and the main examples on the market*. Noudettu osoitteesta https://medium.com/iquii/progressive-web-app-pwa-what-they-are-pros-and-cons-and-the-main-examples-on-the-market-318f4538c670

JAMK. (n.d.). *Mobiilikehitys web-tekniikoilla*. Noudettu osoitteesta https://tiko.jamk.fi/~tuito/websk20/mobwebsk/index.html#32

JavaTpoint. (n.d.). *Ionic Editors*. Noudettu osoitteesta https://www.javatpoint.com/ionic-editors

Juntunen, T. (30.7.2019). *Miten valitsen uuden sovelluksen teknologian?* Noudettu osoitteesta https://www.tecinspire.com/news/miten-valitsen-uuden-sovelluksen-teknologian/

Kazan, S. (7.6.2020). *How does Beacon Technology work? – Simple explanation with examples*. Noudettu osoitteesta https://iotdunia.com/how-beacon-technology-works/

Kokki, M. (21.6.2019). *Virtual Reality Mobile App Development- Future is Immersive*. Noudettu osoitteesta https://medium.com/@maegancook1/virtual-reality-mobile-app-development-future-is-immersive-solution-analysts-a10f5b15ddc1

Kolehmainen, A. (8.5.2019). *Google kehottaa käyttämään kotlinia – Android-java sai naulan arkkuunsa?* Noudettu osoitteesta https://www.tivi.fi/uutiset/google-kehottaa-

kayttamaan-kotlinia-android-java-sai-naulan-arkkuunsa/3818e838-3eae-4ca6-a879-e17cae6d89c4

Lamia. (n.d.). *Push-ilmoitukset: mitä ne ovat ja mihin niitä käytetään?* Noudettu osoitteesta https://lamia.fi/blog/push-ilmoitukset

Liikenne- ja viestintäministeriö. (15.6.2020). *ICT-alan kaksi puolta: ICT-ala kuluttaa energiaa ja materiaaleja, mutta vie myös kohti hiilineutraalia yhteiskuntaa*. Noudettu osoitteesta https://kestavakehitys.fi/-/10184/ict-alan-kaksi-puolta-ict-ala-kuluttaa-energiaa-ja-materiaaleja-mutta-vie-myos-kohti-hiilineutraalia-yhteiskuntaa

Lounea. (n.d.). *5G-Nyt? Huomenna? Milloin? Mitä?* Noudettu osoitteesta https://www.lounea.fi/5g-nyt-huomenna-milloin-mita

Luong, I. (4.3.2021). *PWA vs Native App and how to choose between them.* Noudettu osoitteesta https://www.magestore.com/blog/pwa-vs-native-app-and-how-to-choose-between-them/

Malhotra, M. (6.10.2020). *An Overlook to the Future of Mobile Application Development.* Noudettu osoitteesta https://www.valuecoders.com/blog/technology-and-apps/future-of-mobile-application-development/

Mantco. (n.d.). *Progressiivinen web-sovellus.* Noudettu osoitteesta https://mantco.fi/progressiivinen-web-sovellus-pwa/

Oragui, D. (15.6.2018). *How to Publish an App on Google Play: A Step-by-Step Guide.* Noudettu osoitteesta https://themanifest.com/mobile-apps/how-publish-app-google-play-step-step-guide

O'Reilly. (n.d.). *Chapter 1. What Is React Native?* Noudettu osoitteesta https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

Outsystems. (n.d.-a). *Service Studio Overview*. Noudettu osoitteesta https://success.outsystems.com/Documentation/11/Getting_started/Service_Studio_Overview

Outsystems. (n.d.-b). *Integration Studio*. Noudettu osoitteesta https://success.outsystems.com/Documentation/11/Reference/Integration_Studio

Outsystems. (n.d.-c). *OutSystems Platform*. Noudettu osoitteesta https://www.outsystems.com/enterprise-apaas/

Poot, A. (n.d.). *The state of PWA support on mobile and desktop in 2020*. Noudettu osoitteesta https://simplabs.com/blog/2020/06/10/the-state-of-pwa-support-on-mobile-and-desktop-in-2020/

PV, G. (15.4.2021). *The Future of Mobile App Development: 5 Trends for 2021.* Noudettu osoitteesta https://www.hakunamatatatech.com/our-resources/blog/mobile-app-development-trends

Redandblue. (12.4.2020). *Web sovellus tuo verkkosivustoille natiivinsovelluksen ominaisuudet.* Noudettu osoitteesta https://redandblue.fi/fi/web-sovellus-tuo-verkkosivustolle-nativiivisovelluksen-ominaisuudet/

Salo, P. (2017). *Web-sovelluksen rakentaminen ja elinkaaren hallinta*. Noudettu osoitteesta https://www.theseus.fi/bitstream/handle/10024/136275/Salo_Pekka.pdf?sequence=1&isAllowed=y

Sanoma. (n.d.). *Mikä on sovellus?* Noudettu osoitteesta https://www.sanoma.fi/mitateemme/tietosuoja/tuotekohtaiset-tarkennukset/mobiilisovellukset/

Sormunen, M. (17.6.2020). *VERKKOPALVELUN NOPEUDEN TESTAAMINEN GOOGLE LIGHTHOUSE -TYÖKALULLA.* Noudettu osoitteesta https://www.crasman.fi/blogi/verkkopalvelun-nopeuden-testaaminen-google-lighthouse-ty%C3%B6kalulla

Staff, B. (27.9.2019). *IKEA's AR App Now Lets You Preview Multiple Furniture Items*. Noudettu osoitteesta https://beebom.com/ikea-ar-app-preview-multiple-furniture-items/

Statcounter. (n.d.). *Mobile Operating System Market Share Worldwide* . Noudettu osoitteesta https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202001-202012-bar

Stevens, E. (3.4.2018). *What is the difference between a mobile app and a web app?* Noudettu osoitteesta https://careerfoundry.com/en/blog/web-development/what-is-the-difference-between-a-mobile-app-and-a-web-app/

Strizić, M. (14.1.2021). *Mobile App Development Trends to Know*. Noudettu osoitteesta https://decode.agency/mobile-app-development-trends-to-know/

Sysart. (n.d.). *Natiivi, hybridi, vai React Native? Mobiilisovellustyypit vertailussa*. Noudettu osoitteesta https://sysart.fi/blog/2017/11/30/natiivi-hybridi-vai-react-native-mobiilisovellustyypit-vertailussa/?gclid=Cj0KCQjw-

LOEBhDCARIsABrC0TkLkdMVaN93zlwoqJd8-oBi9ig3nJDZXKn2xWDfRpU-dFV9xF7woygaAvl8EALw_wcB

Tapanainen, T. (n.d.). *Mitä on low-code ja kuka sitä tarvitsee?* Noudettu osoitteesta https://www.ecraft.com/fin/blog/2018/2/15/mita-on-low-code-ketteraa-sovelluskehitysta

Team Compile. (11.6.2019). *Mitä on kestävä ohjelmistokehitys?* Noudettu osoitteesta https://compile.fi/mita-on-kestava-ohjelmistokehitys/

Thinkful. (n.d.). *How Hard is it to Learn HTML?* Noudettu osoitteesta https://www.thinkful.com/blog/how-hard-is-it-to-learn-html/

Tutorialspoint. (n.d.). *WAP - Introduction*. Noudettu osoitteesta https://www.tutorialspoint.com/wap/wap_introduction.htm

Unifunds. (n.d.). *History of Mobile Application Development*. Noudettu osoitteesta https://unifunds.com/history-of-mobile-application-development

Valuga. (n.d.). *What is OutSystems?* Noudettu osoitteesta https://www.valuga.nl/what-is-outsystems/

Vella, H. (15.5.2019). *5G vs 4G: what is the difference?* Noudettu osoitteesta https://www.raconteur.net/technology/5g/4g-vs-5g-mobile-technology/

W3C. (n.d.). *HTML and CSS*. Noudettu osoitteesta https://www.w3.org/standards/webdesign/htmlcss

Liite 1: Aineiston hallintasuunnitelma

Tulen keräämään havainnointiainestoa opinnäytetyön aikana. Opinnäytetyössä tullaan havainnoimaan eri sovellusalustojen toimintaa. Havainnointiaineisto tullaan keräämää, siten että tutkin eri mobiilisovellusalustoja ja kirjaan niistä huomioita toimivuudesta sekä julkaisuprosessista ylös itselleni. Kerään havainnoinnin yhteydessä näyttökuvia mobiilisovellusalustoista. Opinnäytetyön alussa mietin mahdollista kyselyn järjestämistä alan yrityksille, mutta se on vielä hieman kysymysmerkki tässä kohtaan.

Opinnäytetyö ei tule sisältämään mitään arkaluontoista tietoa, joten säilytys tapahtuu henkilökohtaisessani OneDrive kansiossa sekä tallennan myös varmuuskopion omaan ulkoiseen kovalevyyn. Kaikki opinnäytetyöhön liittyvä aineisto yms. on kerätä yhteen paikkaan/kansioon selkeyden vuoksi.

Kerätystä aineistosta voi olla mahdollisesti jotain hyötyä tulevaisuudessa, joten en itse näe syytä niiden tuhoamiselle opinnäytetyön valmistuttua. Mahdolliselle jatkokäytölle ei pitäisi olla esteitä, sillä opinnäytetyössä ei tule olemaan yritysten tai henkilöiden tietoja.