Tampereen yliopisto

Juho Kuusela

BLUETOOTH LOW ENERGY -YHTEYDET ANDROID-SOVELLUSKEHITYKSESSÄ MONIALUSTAISILLA OHJELMISTOKEHYKSILLÄ

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2022

TIIVISTELMÄ

Juho Kuusela: Bluetooth Low Energy -yhteydet Android-sovelluskehityksessä monialustaisilla ohjelmistokehyksillä
Kandidaattitutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikan tutkinto-ohjelma
Toukokuu 2022

Tässä kirjallisuuskatsauksessa selvitetään yleisimpien monialustaisten ohjelmistokehysten soveltuvuutta Android-sovelluksen ohjelmointiin, jonka oleellisena vaatimuksena Bluetooth Low Energy (BLE) -yhteyden käyttö. Vertailua käydään mobiiliohjelmistokehittäjien yleisesti käyttämien ohjelmistokehysten välillä ja pyritään löytämään tarkoitukseen sopivin vaihtoehto. Tavoitteena on löytää selkeitä perusteluja minkä vuoksi jokin ohjelmistokehys sopisi juuri tähän tiettyyn ohjelmistoprojektiin toisia paremmin. Ohjelmistokehyksen monialustaisuus avaa mahdollisuuden laajentaa Android-sovelluksen lisäksi ohjelmistokehitystä myös muille alustoille.

Työn ensimmäisessä vaiheessa tarkastellaan yleisimpien ohjelmistokehysten jakaumaa kehittäjien keskuudessa, ohjelmistokehysten käyttämiä ohjelmointikieliä, ja kunkin ohjelmistokehyksen virallisten laajennuskirjastojen hakemistojen laajuutta. Näiden tekijöiden perusteella valitaan tarkempaan vertailuun kaksi ohjelmistokehystä, joiden osalta selvitetään perustiedot ohjelmistokehyksen kehitys- ja kohdealustoista, BLE-hallintaan tarvittavien laajennosten saatavuudesta, ja näkyvimmistä ohjelmistokehysten yhtäläisyyksistä sekä eroista.

Työssä selviää, että suositusta on vaikea tehdä vain annettujen kahden vaatimuksen perusteella, ja toissijaisiakin arviointipiirteitä tarvitaan lopullisen valinnan tekemiseksi. Toissijaisia arviointipiirteitä ovat muun muassa ohjelmistokehyksen tukemat ohjelmointikielet ja ohjelmistokehyksen muut laajennusmahdollisuudet. Toisaalta myös muut sovelluksen ominaisuudet, kuten käyttöliittymä ja toiminnan tehokkuus ohjaavat valintaa. Myös ohjelmiston jatkokehitys- ja ylläpitotarpeet on huomioitava ohjelmistokehyksen valinnassa, sillä ohjelmistokehyksen vaihtaminen saattaa vaatia suuria muutoksia ohjelmistossa.

Avainsanat: Bluetooth Low Energy, Android, sovellus, ohjelmistokehys, monialustainen ohjelmistokehitys

Sisällysluettelo

1 Johdanto

Moderneilla ohjelmistokehyksillä voi helposti luoda sovelluksia usealle eri alustalle, sillä ohjelmistokehys ikään kuin piilottaa laitteet yleistettyjen rajapintakutsujen taakse. Tämä usein helpottaa myös yhtenäisen käyttöliittymän luomisessa, sillä useissa tapauksissa sovelluskehittäjä ei voi ennalta tietää, millaisella laitteella sovellusta tullaan käyttämään. Monesti myös internetyhteys on oleellisessa osassa sovellusten käytössä, kun tietoja päivitetään laitteille ja laitteilta, erityisesti *esineiden internetiä* (IoT, Internet of Things) käytettäessä.

Bluetooth Low Energy (BLE) on merkittävässä osassa myös *puettavien laitteiden* (wearable devices) langattomissa yhteyksissä, Bluetooth-yhteyksien ollessa muiden muassa älypuhelinten vakio-ominaisuuksia. Standardien noudattamisesta huolimatta eri laitteiden toiminnassa on merkittäviä eroja, jotka vaikuttavat yhteyksien nopeuteen ja laatuun. [1] Sovelluskehittäjän näkökulmasta onkin tärkeää, että kun sovellusta BLE-laitteen käyttämiseen suunnitellaan, on sen toimittava usein hyvinkin erilaisilla laitteilla, ja että sen on toimittava riitettävän luotettavasti normaaleissa käyttötilanteissa. Sovelluksen ja laitteen käyttäjä voi ymmärrettävästi olettaa, että reagointi on lähes välitöntä, jolloin myös viiveet niin yhteyden muodostamisessa kuin käytössä on pidettävä mahdollisimman pieninä [2,3].

Tässä kirjallisuuskatsauksessa perehdytään tarkemmin tuotteistettuihin ohjelmistokehyksiin, ja niiden tarjoamiin mahdollisuuksiin kehittää BLE-yhteyksiä käyttäviä sovelluksia Android-laitteille. Yleisluontoisia ohjelmistokehyksiä BLE-laitteiden yhdistämiseen pilvipalveluihin käyttäen älypuhelinta tiedon välittäjänä on myös esitetty [4,5]. Ohjelmoijan onkin todennäköisesti helppo tarttua jo olemassa olevaan *ohjelmistokehykseen, jolla voi kehittää sovelluksia eri alustoille* (cross platform application development framework), sillä tuotteistettuina niille tarjottu tuki on jo olemassa.

Luvussa 2 käydään läpi ohjelmistokehyksiä, BLE:tä sekä mobiilisovelluskehitystä yleisemmin. Luvussa 3 vertaillaan yleisimpiä monialustakehitystyöhön tuotteistettuja ohjelmistokehyksiä, joita tällä hetkellä ohjelmistokehittäjille on käytettävissä, ja rajataan niitä tarkemmin tutkimuskysymyksen mukaisesti. Luvussa 4 esitellään vertailun tulokset ja tehdään näiden perusteella johtopäätökset. Luvussa 5 esitetään yhteenveto työstä, sekä pohditaan vielä tarkemmin jatkon kannalta oleellisia asioita.

2 Käsitteiden taustoitus

Tässä luvussa perehdytään tarkemmin työn kannalta oleellisiin käsitteisiin. Tämän kirjallisuuskatsauksen otsikko voidaankin jakaa kolmeen oleelliseen termiin, jotka ovat Bluetooth Low Energy, Android-sovelluskehitys ja ohjelmistokehykset. Myöhemmissä aliluvuissa käydään myös lyhyesti tämän tutkielman kannalta oleellisimmat yhdistelmät näistä kolmesta käsitteestä.

2.1 Bluetooth Low Energy

Bluetooth Low Energy (BLE) on Bluetooth SIG, Inc. -organisaation määrittelemä ja ylläpitämä langaton tiedonsiirtostandardi, joka on nimensä mukaisesti suunniteltu matalaenergiseen tiedonsiirtoon. Toisin kuin Bluetooth Classic -standardi, joka perustuu laitteiden *kahdenväliseen* (point-to-point) tiedonsiirtoon, BLE tukee myös *monesta moneen* (many-to-many) tiedonsiirtoa ja *yhteydetöntä* (connectionless) tiedonsiirtoa. [6] BLE määritteleekin kaksi toimintatilaa: *oheislaitetilan* (peripheral mode) matalaenergialaitteille, jotka toimivat pääsääntöisesti tiedon lähettäjinä, ja *keskuslaitetilan* (central mode) niille laitteille, jotka pääsääntöisesti vastaanottavat ja käsittelevät tietoa. Oheislaitetilassa toimiva laite voi lähettää tietoa muodostamatta jatkuvaa yhteyttä, jolloin laitteen virrankulutus voidaan minimoida säätelemällä tiedonsiirron tiheyttä ja viivettä. [7]

2.2 Android-sovelluskehitys

Google tarjoaa sovelluskehitykseen Android-käyttöjärjestelmälle muutamia työkaluja. Käytännössä sovelluksen kääntämiseen ja jakelumuotoon saattamiseen tarvittavat työkalut löytyvät Android SDK:sta (software development kit) Java- ja Kotlin-ohjelmointikielille. Edellä mainittua laajentaa Android NDK (native development kit), joka tuo mahdollisuuden kehittää ohjelmistoja käyttäen myös C- ja C++ -ohjelmointikieliä. Näiden lisäksi Android Studio on tarkoitettu ohjelmistosuunnitteluun, kehitykseen ja testaukseen työpöytäympäristöissä. [8] Android SDK:ta ja Android NDK:ta on mahdollista käyttää myös muilla ohjelmistokehyksillä suunniteltujen sovellusten kääntämiseen Android-alustalle.

On myös huomionarvoista, että Android-alustalle kehitystyökalut ovat vapaasti ladattavissa kaikille kolmelle merkittävimmälle työpöytäkäyttöjärjestelmälle: Windowsille, macOS:lle ja Linuxille. Vastaavasti Xcode, jonka avulla kehitetään Applen eri laitteille sovelluksia, on ladattavissa joko Mac App Store -sovelluksesta (macOS-laitteille) tai

Applen kehittäjäsivustolta [9], käytännössä rajaten ohjelmistokehitystyön vain Applen omille laitteille.

2.3 Ohjelmistokehys

Ohjelmistokehyksen voi määritellä kokonaan tai osittain uudelleen käytettävänä ohjelmistokomponenttien summana, jonka toiminnallisuutta voi kehitysvaiheessa tarkemmin muokata [10]. Niiden tarkoituksena on siis helpottaa ohjelmistokehitystä tarjoamalla kehittäjälle työkaluksi valmiita rakenteita, joita voi käyttää sovelluskehityksen pohjana. Toisaalta ohjelmistokehyksen valinta suunnitteluvaiheessa rajoittaa myös sitä, millaisen lopputuloksen sen avulla saa, ja ohjelmistokehyksen etujen tunnistaminen saattaa vaatia paljon asiaan perehtymistä [11]. Ohjelmistokehyksen vaihtaminen kesken ohjelmiston kehittämisen saattaa vaatia koko ohjelmiston uudelleensuunnittelua ja -kirjoittamista [12].

2.4 Ohjelmistokehykset mobiilisovellusten kehittämisessä

Myös mobiilisovelluksia kehitettäessä suunnittelu on tärkeää. Sovelluksia voi kehittää myös ilman ohjelmistokehyksiä käyttäen esimerkiksi alustalle tarjottua SDK:ta, jolloin ohjelmistokehyksen asettamia rajoitteita tai tarjoamia etuja ei ole. Harvoin kuitenkaan mobiilisovellusta kehitetään ilman käyttöliittymää, jolloin jonkinlaisen ohjelmistokehyksen käyttö on yleistä.

Mobiilisovellusten kehittämisessä on kolme selkeää pääasiallista kehityssuuntaa: *web-sovellukset*, *hybridisovellukset* ja *natiivisovellukset*, joskin tällekin jaolle on ehdotettu vaihtoehtoja [13]. Useiden empiiristen tutkimusten mukaan hybridisovelluksen etuna natiivisovelluksiin on nopeampi kehitystyö useammalle alustalle sovellusta toteutettaessa. Natiivisovellukset ovat taas toiminnaltaan nopeampia ja kuluttavat vähemmän resursseja. [14–17] Yhteinen nimittäjä hybridi- ja web-sovelluksille onkin ajonaikaisten tulkattavien kielten käyttö, mikä usein näkyy heikentyneenä suorituskykynä ja suurempana ohjelmakokona natiivisovelluksiin verrattuna.

2.5 Bluetooth Low Energy -yhteydet ohjelmistokehyksissä

Ohjelmistosuunnittelija voi tehdä tiettyjä oletuksia siitä ympäristöstä, jossa sovellusta ajetaan. Määrittelyvaiheessa tiedetään varmuudella mahdollinen käyttöjärjestelmä ja sen mukanaan tuoma tuki laitteistolle. Loppukäyttäjillä oleviin laitteisiin ei voi ohjelmistokehityksessä kuitenkaan vaikuttaa, ja eri alustoilla saman toiminnon toteuttaminen voi-

kin olla hyvin erilaista. Ohjelmistokehyksillä pystytään tätä jonkin verran yksinkertaistamaan, jolloin ohjelmistokehittäjän ei tarvitse erikseen opetella jokaiselle käyttöympäristölle vaadittuja toimenpiteitä.

Fabryq-nimisen ohjelmistokehyksen on tarkoitus kuvata yhtenäinen kehitysympäristö, jossa ohjelmistoja suunnitellaan kokonaisuutena [5]. NetApp-nimisessä sovellusrajapinnassa on tarkoitus yhtenäistää vastaavasti eri valmistajien laitteiden kirjo yhtenäisen rajapinnan taakse, koska käytettyjen laitteiden ominaisuudet ja rajapinnat vaihtelevat merkittävästi [4]. Kummassakin ohjelmistokehyksessä kuvaillaan koko tiedonkulun ketju aina BLE-laitteesta mobiililaitteen kautta pilvipalveluun. Tavoitteena on helpottaa BLE:tä käyttävien sovellusten kehittämistä sulautetuille alustoille luomalla yhtenäinen rajapinta tiedonsiirtoon, jotta sovelluskehittäjä voi keskittyä muihin ominaisuuksiin.

3 Tuotteistetut ohjelmistokehykset mobiilisovelluskehityksessä

Sovelluskehitykseen käytettyjen ohjelmistokehysten suosio määräytyy usein käytettävyyden tai saavutettavuuden perusteella. Käytännössä markkinoilla on kahden ympäristön duopoli, jossa vastakkain ovat pääosin avoimeen lähdekoodiin perustuva Googlen Android, ja Applen hallinnoima suljetun lähdekoodin iOS [13]. Näiden kahden ympäristön määräävästä asemasta markkinoilla johtuen valtaosalla ohjelmistokehyksistä voikin kehittää sovelluksia molemmille alustoille. Taulukko 1 esittää mobiilisovellusten kehittäjien kuuden eniten suosiman ohjelmistokehyksen jakauman.

Taulukko 1. Mobiilisovellusten kehittäjien yleisimmin käyttämät ohjelmistokehykset. [18]

	Flutter	React Native	Apache Cordova	Ionic	Xamarin	Unity
Osuus 2021	42 %	38 %	16 %	16 %	11 %	11 %
Osuus 2020	39 %	42 %	18 %	18 %	14 %	11 %

Tutkimuksessa erottui selvästi kaksi yleisimmin käytettyä ohjelmistokehystä (Flutter ja React Native), joiden osuus oli 40 prosentin tietämillä. Neljää seuraavaksi suosituinta käytti 10 ja 20 prosenttiyksikköä vastanneista, kun taas muiden sovelluskehysten osuus jää korkeintaan kymmeneen prosenttiyksikköön.

3.1 Ohjelmistokehyksissä käytetyt ohjelmointikielet

Vertailtaessa kuutta yleisintä ohjelmistokehystä havaitaan selkeää ryhmittymistä eri ohjelmointikielten osalta, kuten taulukosta 2 nähdään.

Taulukko 2. Käytetyimpien ohjelmistokehysten ohjelmointikielien ja lisenssien vertailu kehitettäessä Android-sovelluksia, kielet kursiivilla ovat toissijaisesti käytettävissä.

	Lisenssi	Tuetut ohjelmointikielet
Apache Cordova [19]	Apache 2	HTML, CSS, JavaScript
Flutter [20]	BSD 3-Clause “New” or “Revised” License	Dart, *Kotlin, Java*
Ionic [21]	MIT	HTML, CSS, JavaScript
React Native [22]	MIT	JavaScript
Unity [23]	Kaupallinen	C#
Xamarin [24]	MIT ja Apache 2	C#, *Objective-C, Java, C, C++*

Apache Cordova, Ionic ja React Native tukeutuvat selvästi verkkoselaimistakin tuttuun JavaScriptiin ensisijaisena ohjelmointikielenään. On siis pääteltävissä, että yhteisen ohjelmointikielen lisäksi nämä kolme mahdollisesti jakavat muutakin yhteistä. JavaScriptillä kirjoitettua ohjelmakoodia ei pääsääntöisesti käännetä ennen sen suorittamista, joten on perusteltua olettaa, että kolme mainittua kehystä tuottavat pääsääntöisesti hybridisovelluksia.

Xamarin käyttää muualta .NET -ohjelmistokehyksestä tuttua C#-kieltä, tukien myös muita samansukuisia kieliä. Flutter puolestaan käyttää Dart-kieltä, mutta tukee myös Android-järjestelmän natiivikieliä Kotlinia ja Javaa. Nämä kaksi ovatkin siis lähempänä natiivisovelluksia, sillä kummankin tapauksessa kaikki ohjelmakoodi käännetään ennen sen suorittamista.

Unity on joukon poikkeus, sillä vaikka se käyttääkin C#-kieltä, sitä ei käytetä perinteiseen ohjelmointiin vaan komentokielen tapaan. Unity on lähtökohtaisesti suunniteltu peliohjelmointiin, tukien myös *täydennetyn todellisuuden* (augmented reality, AR), *yhdistetyn todellisuuden* (mixed reality, MR) ja *tekotodellisuuden* (virtual reality, VR) sovelluksia. [25] Lisäksi Unity on joukon ainoa täysin kaupallisen lisenssin varassa toimiva ohjelmistokehys [23].

3.2 Ohjelmistokehysten laajennettavuus

Ohjelmistokehysten avulla luotujen sovellusten toimintaa laajennetaan yleisesti kirjastoilla tai liitännäisillä, jotka mahdollistavat ohjelmistokehyksen rajojen ylittämisen [12]. Liitännäisen tai kirjaston avulla on helpompi päästä esimerkiksi käsiksi laitteistoon, käyttöjärjestelmän rajapintoihin, tai toiseen ohjelmistokehykseen. Merkittävä osa ohjelmistokehyksistä tukeutuu kehittäjäyhteisön luomiin kirjastoihin ja liitännäisiin, jotka ovat usein myös keskitetysti löydettävissä ylläpitäjän palveluista. Taulukossa 3 on listattuna vertailtujen ohjelmistokehysten viralliset laajennushakemistot.

Taulukko 3. Ohjelmistokehysten viralliset laajennushakemistot 13. huhtikuuta 2022.

	Virallisesti tuetut laajennokset	Laajennosten lukumäärä
Apache Cordova	Cordova Plugins [26,27]	6496; itse sivuston hakupalvelu ei toimi, lukumäärä tarkistettu sivustolta npmjs.com.
Flutter	Dart packages [28]	Noin 24700.
Ionic	Capacitor.js [29] sekä Apache Cordova	Capacitor.js: alle 100.
React Native	React Native Directory [30]	Noin 1000.
Unity	Unity Asset store [31]	Noin 75000, joista ilmaisia alle kymmenesosa.
Xamarin	Open Source Components for Xamarin [32]	Xamarin.Essentials, jonka lisäksi virallisesti Google Play -, Facebook- ja AndroidX-kirjastorajapinnat.

Hakemistojen laajuus vaihtelee suuresti, kuten taulukosta 3 nähdään. Unity Asset Store on selvästi laajin kokonaisuutena, kun taas Xamarin ja Capacitor.js edustavat toista ää-

ripäätä. Capacitor.js on sekä Ionic:in, että Capacitor-yhteisön ylläpitämien laajennosten hakemisto. Xamarin.Essentials puolestaan tarjoaa iOS-, Android-, ja UWP (Unified Windows Platform) -alustoille tuen .NET Framework -kehyksen käyttöön, joka puolestaan on ollut Windows-sovelluskehittäjien saatavilla jo 1990-luvulta lähtien.

3.3 Ohjelmistokehysten yhteenveto

Ohjelmistokehysten laajennettavuus ei ole suoraan luettavissa taulukon 3 numeroista, mutta se voi toimia suuntaa antavana viitteenä kehittäjäyhteisön toiminnasta. On myös hyvä huomioida mahdolliset muut sidokset, kuten jo aiemmin mainittu Xamarin osana .NET-kehitysympäristöä. JavaScript-ohjelmointikieltä pääasiallisesti käyttävien ohjelmistokehysten osalta on myös oleellista huomioida mahdollinen ero suorituskyvyssä, joka muodostuu tiedostojen käytönaikaisesta tulkkaamisesta verrattuna käännettävien ohjelmointikielten käyttöön.

Tämän selvityksen tarkoituksena ei kuitenkaan ole löytää kaikkein tehokkainta ohjelmistokehystä juuri tähän käyttötarkoitukseen, vaan vertailla yleisimmin käytettyjen ohjelmistokehysten soveltuvuutta juuri Bluetooth Low Energy -yhteyksiä käyttävän sovelluksen kehittämiseen. Tähän mennessä on selvitetty, että Flutter ja React Native ovat yleisimmin käytetyt kaksi ohjelmistokehystä, ja kummassakin on mahdollisuuksia laajentaa toiminnallisuutta liitännäisillä. Seuraavassa luvussa käydäänkin tarkemmin läpi näiden kahden ohjelmistokehyksen ominaisuuksia aiemmin määriteltyjen ehtojen mukaan, jotka kiteytettynä ovat tämän kirjallisuusselvityksen otsikossa.

4 Valittujen ohjelmistokehysten vertailu

Flutter ja React Native ovat aiemmin mainitun kyselytutkimuksen mukaan kaksi käytetyintä ohjelmistokehystä mobiilikehittäjien keskuudessa, ja kumpaakin käyttää noin 40 prosenttia kaikista tutkimukseen vastanneista. [18] Nämä eivät kuitenkaan poissulje toistensa käyttöä, ja joissain tapauksissa voikin olla mielekästä käyttää useampia ohjelmistokehyksiä toisiaan tukevien ominaisuuksien vuoksi. On mahdollista toteuttaa sovellus loppukäyttäjälle näkymättömiltä osilta esimerkiksi Flutterilla tai Xamarinilla, ja luoda käyttöliittymä tälle sovellukselle React Nativella tai Ionicilla. Selkeyden vuoksi tässä työssä kuitenkin keskitytään vain ensin mainittujen kahden keskinäiseen vertailuun.

4.1 Ohjelmistokehyksen käyttöönotto ja muut kohdealustat

Kummankin ohjelmistokehyksen taustalla vaikuttaa suuri monikansallinen yhtiö, jolla on merkittävä asema tietotekniikan ja tietoliikenteen alalla, mikä on nähtävissä laajana tukena erilaisille laitteille ja käyttöjärjestelmille. Molemmat ohjelmistokehyksien kehitysympäristöt ovat asennettavissa niin Windows-, macOS-, kuin Linux-käyttöjärjestelmille. Lisäksi Flutter tukee asennusta ChromeOS-laitteille, joissa on Linux-beetaominaisuus asennettuna. React Nativen tarjoama Expo-työkalu puolestaan mahdollistaa ”kehitystyön aloittamisen minuuteissa”, ja Snack-työkalu sallii React Native -ympäristön kokeilun ilman ohjelmistoasennuksia tietyillä verkkoselaimilla. [33,34]

Kuten luvussa 2.2 todettiin, Apple on rajoittanut ohjelmistokehityksen omille alustoilleen, ja tämä on nähtävissä myös ohjelmistokehysten käyttöönotoissa. iOS-ohjelmistokehitys on molemmilla ohjelmistokehyksillä mahdollista, mikäli kehitysalustana on macOS-tietokone, ja sille asennettu Xcode-kehitysympäristö. Android-ohjelmistokehitys vaatii myös Android Studion asennuksen ohjelmistokehyksen omien ohjelmistovaatimusten lisäksi. Kumpikin ohjelmistokehys asennetaan merkittävin osin komentoriviltä, mikä ei todennäköisesti ole ongelma ohjelmistokehittäjälle.

React Native tukee ohjelmistokehitystä kahdelle merkittävimmälle mobiilialustalle, mutta myös selainsovelluksille; Metan partnerit sekä React Native -yhteisö tukevat myös muita alustoja [35], kuten myös Flutter [36]. Mikäli siis myöhemmässä vaiheessa on tarpeen tehdä esimerkiksi työpöytä- tai iOS-sovellus käyttäen samaa koodipohjaa, on tämä teoriassa mahdollista vaihtamatta ohjelmistokehystä, tai tuomatta lisäksi muita ohjelmistokomponentteja.

4.2 BLE-laitteiston saavutettavuus

Android tukee BLE-yhteyksiä käyttöjärjestelmätasolla ohjelmoitaessa suoraan Javalla tai Kotlinilla [37]. Koska kumpikaan ohjelmistokehyksistä ei kuitenkaan suoraan tue ohjelmointia kummallakaan kielellä, järjestelmärajapintakutsut tulee sovittaa jonkin välissä olevan ohjelmakomponentin avulla. Käytännössä tämä hoidetaan sovittamalla natiivikielellä kirjoitettu ohjelmakirjasto tai -moduuli ohjelmistokehyksen käyttämään ohjelmointikieleen. Valmiiksi paketoidut liitännäiset, joita edellisen luvun taulukossa 3 listattiin, ovat tapa välttää alustakohtaista ohjelmointia käyttäessä ohjelmistokehystä.

React Native Directory -hakemiston [30] käyttö osoittautui työlääksi, sillä haku ei toimi niin sanottuna kokosanahakuna, vaan hakutulos voi löytyä myös keskeltä sanaa. Hake-

mistosta kuitenkin löytyi selvitystyötä tehdessä kaksi liitännäistä [38,39], jotka molemmat tukevat sekä Android-, että iOS-alustoja. Kumpikin liitännäisistä tarjoaa nopealla tutkimisella riittävän tuen BLE-yhteydellä oheislaitteiden (peripheral) hakemiseen ja yhdistämiseen. Kummassakaan ei kuitenkaan ole tukea oheislaitteena toimimiselle, esimerkiksi vertaistyyppisiin yhteyksiin kahden samankaltaisen laitteen välillä.

Dart Packages -hakemistosta [28] hakiessa ilmeni jonkun verran sama ongelma kuin React Native Directory -hakemistosta hakiessa edellä. Tällä kertaa kuitenkaan ongelma oli tavallaan käänteinen, sillä liitännäisistä löytyi myös huomattavasti erikoistuneempia paketteja tiettyjen laitteiden tai tietyn tyyppisten yhteysmuotojen käyttöön. Myös oheislaitetilaa tukevia paketteja oli saatavilla, ja osa liitännäisistä lupasi tukea Android- ja iOS-alustojen lisäksi myös muita käyttöjärjestelmiä.

4.3 Ohjelmistokehysten yhteiset ominaisuudet ja merkittävimmät erot

Dart-ohjelmointikieli, jota Flutter ensisijaisesti käyttää, on suunniteltu käännettäväksi, mutta käännöksen ajankohta ja kohde riippuu ohjelmistokehityksen vaiheesta ja kohdealustasta. Web-sovelluksen tapauksessa Dart-kielinen ohjelmakoodi käännetään JavaScript-kielelle, joka on puolestaan myös React Nativen pääasiallinen ohjelmointikieli. [40] Kumpikin ohjelmointikieli on suunniteltu tukemaan myös asynkronista ohjelmointia [41,42], jonka vuoksi ne soveltuvat erilaisten yhteyksien hallinnointiin.

Kumpikin ohjelmistokehyksistä tukee hyvin laajentamista liitännäisten avulla, ja erot todennäköisesti näkyvät lähinnä valmiin sovelluksen toimintanopeudessa. Dart-ohjelmakoodi käännetään ennen suorittamista, ja tämän lisäksi tarvitaan pienikokoinen Dart Runtime -moduuli, joka hoitaa yleisimmät ohjelman suorittamiseksi vaaditut toiminnot [40]. Toiminnallisuutta laajentavat ohjelmakoodikirjastot siis käännetään ja linkitetään osaksi ajettavaa binäärikoodia. Myös React Nativen laajennokset käännetään ja linkitetään osaksi ohjelmakoodia, mutta tämän lisäksi suoritusympäristön pitää tulkita ja esittää kääntämätön JavaScript-koodi ajonaikaisesti. React Nativen etu onkin, että pelkän JavaScript-koodin muuttaminen ei vaadi välttämättä koko ohjelman uudelleenkääntämistä. Toisaalta JavaScript-tulkin pitää olla valmiina kaikkiin mahdollisiin tilanteisiin, kun taas etukäteen käännetty Dart-koodi voidaan tuotannossa pitää mahdollisimman optimaalisena juuri ohjelman suoritusta varten.

Käytännössä suurta eroa ei siis pelkästään näillä vertailutermeillä saada aikaiseksi. Kummankin ohjelmistokehyksen vahvuutena on laaja levinneisyys, ja käytetyt ohjel-

mointikielet tuskin aiheuttavat suurta ongelmaa ohjelmistokehittäjille. Dart vaikuttaa hyvin samankaltaiselta kieleltä kuin Android-järjestelmän natiivikieli Java, kun taas JavaScript on hyvin laajalle levinnyt seittisovellusten yhteydessä.

Kun pidetään mielessä lähtövaatimuksena ollut Android-ohjelmistokehitys BLE-yhteyksille, merkittävimmät erot löytyvät kuitenkin näiden tekijöiden ulkopuolelta. Mikäli sovelluksen tulisi tarjota mahdollisesti muuttuvaa sisältöä suurempia määriä, tällöin React Native saattaa soveltua paremmin, sillä JavaScript-koodia ei tarvitse erikseen kääntää, ja sovellus tällöin toimii lähemmin niin sanottuna hybridisovelluksena. Toimintatehokkuutta kaipaava kääntyy näistä kahdesta ennemmin Flutterin suuntaan. Dart-kielinen ohjelmakoodi käännetään ennen suoritusta ja sovellus toimii enemmän kuin natiivisovellus, vaikkei olekaan kirjoitettu varsinaisella natiivikielellä.

5 Yhteenveto ja jatkokehitysehdotukset

Tässä kirjallisuuskatsauksessa perehdyttiin tuotteistettuihin ohjelmistokehyksiin, ja niiden tarjoamiin mahdollisuuksiin kehittää BLE-yhteyksiä käyttäviä sovelluksia Android-laitteille. Aluksi määriteltiin valittavaksi vertailuun ohjelmistosuunnittelijoille tehdyn kyselytutkimuksen mukaan yleisimmin työssään käyttämiä ohjelmistokehyksiä [18]. Alkuperäisestä joukosta rajattiin tarkempaan vertailuun kaksi käytetyintä ohjelmistokehystä. Pelkästään vertailemalla tutkimuskysymyksessä määriteltyjä ominaisuuksia ei kuitenkaan saavutettu suurta eroa tarkempaan vertailuun valittujen kahden suosituimman ohjelmistokehyksen välillä.

On melko epätodennäköistä, että lopputulos olisi kovin erilainen, mikäli vertailua käytäisiin useamman yleisluontoisen ohjelmistokehyksen välillä. Erityisesti tiettyyn käyttötarkoitukseen suunnitellun ohjelmistokehyksen kuten fabryq [5] tai NetApp [4] vertailu yleiskäyttöiseen ohjelmistokehykseen voisi tuoda erilaisia tuloksia. Käytännön toteutusta kummallekaan kahdesta esitetystä ohjelmistokehyksestä ei tämän kirjallisuusselvityksen piirissä löytynyt. Pelkästään kirjallisuuden perusteella on vaikea tehdä päätelmiä soveltuvuudesta, sillä teoreettisen työkalun käytännön toimintaa on vaikea suhteuttaa konkreettiseen ongelmaan. Valinta tehdäänkin todennäköisesti muiden ominaisuuksien, kuin pelkästään teknisten vaatimusten perusteella. [12] Ohjelmistokehittäjä oletettavasti haluaa käyttää ennemmin valmiita työkaluja, ja työkalut tulee valita ratkaistavan ongelman mukaan, jotta lopputulos saavutetaan tavoitteiden mukaisesti.

Tässä työssä myös yksinomaan keskityttiin ohjelmistokehitykseen Android-laitteelle, mutta ohjelmistokehittäjän tulee todennäköisesti ratkoa muitakin ongelmia ohjelmiston elinkaaren aikana. Usein kehitystyötä tehdään samanaikaisesti myös BLE-oheislaitteelle, ja tällöin voi olla tarvetta työstää sovellusta ilman valmista laitetta. Tällaisen tilanteen helpottamiseksi Frohner ja muut ehdottavat toisen Android-laitteen käyttöä simuloimaan sulautettua laitetta [43].

Myös BLE-yhteyksien langattomuus tuo oman osansa loppukäyttäjien mahdollisesti kokemiin ongelmiin. Eri valmistajien laitteiden välillä voi esiintyä merkittäviäkin toimintaeroja periaatteessa saman standardin laitteiden välillä. Tipparaju ja muut esittääkin useita erilaisia ratkaisutapoja yleisimpiin yhteyden muodostamisen ja käytönaikaisiin ongelmiin [1]. Myös tietoturvallisuus on huomioitava, esimerkiksi *väliintulohyökkäyksiltä* (man-in-the-middle attack) on suojauduttava. Yaseen ja muiden julkaisussa esitetään konkreettisia ratkaisuja yleisimpien tietoturvaongelmien ratkaisuun sulautetun järjestelmän ja isäntälaitteen välisissä BLE-yhteyksissä [44].

Lähdeluettelo

1. Tipparaju VV, Mallires KR, Wang D, Tsow F, Xian X. Mitigation of Data Packet Loss in Bluetooth Low Energy-Based Wearable Healthcare Ecosystem. Biosensors. 23. syyskuuta 2021;11(10):350.

2. Miller RB. Response time in man-computer conversational transactions. Teoksessa: In Proc AFIPS Fall Joint Computer Conference [Internet]. 1968. Saatavissa: https://doi.org/10.1145/1476589.1476628

3. T'Jonck K, Pang B, Hallez H, Boydens J. Optimizing the Bluetooth Low Energy Service Discovery Process. Sensors. 2021;21(11):3812.

4. Kim H, Ahn M, Hong S, Lee S, Lee S. Wearable Device Control Platform Technology for Network Application Development. Mobile information systems. 2016;2016:1–20.

5. McGrath W, Etemadi M, Roy S, Hartmann B. fabryq: using phones as gateways to prototype internet of things applications using web scripting. Teoksessa ACM; 2015. s. 164–73. (EICS '15).

6. Bluetooth SIG, Inc. Bluetooth Technology Overview [Internet]. Bluetooth® Technology Website. 2022 [viitattu 20. maaliskuuta 2022]. Saatavissa: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

7. Gomez C, Oller J, Paradells J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. Sensors. syyskuuta 2012;12(9):11734–53.

8. Google Developers. Android Developers [Internet]. Android Developers. 2022 [viitattu 20. maaliskuuta 2022]. Saatavissa: https://developer.android.com/

9. Apple. Xcode | Apple Developer Documentation [Internet]. 2022 [viitattu 17. huhtikuuta 2022]. Saatavissa: https://developer.apple.com/documentation/xcode

10. Johnson RE. Frameworks = (components + patterns). Commun ACM. 1. lokakuuta 1997;40(10):39–42.

11. Fayad M, Schmidt DC. Object-oriented application frameworks. Communications of the ACM. lokakuuta 1997;40(10):32–8.

12. Myllärniemi V, Kujala S, Raatikainen M, Sevón P. Development as a journey: factors supporting the adoption and use of software frameworks. Journal of Software Engineering Research and Development. 4. kesäkuuta 2018;6(1):1–22.

13. Nunkesser R. Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development. Teoksessa: 2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft). 2018. s. 214–8.

14. Bosnic S, Papp I, Novak S. The development of hybrid mobile applications with Apache Cordova. Teoksessa: 2016 24th Telecommunications Forum (TELFOR). 2016. s. 1–4.

15. Lim SH. Experimental Comparison of Hybrid and Native Applications for Mobile Systems. IJMUE. 31. maaliskuuta 2015;10(3):1–12.

16. Olajide O, Abiodun A, Okunola A, Gabriel T, Michael S. Performance Evaluation of Native and Hybrid Android Applications. CAE. 25. toukokuuta 2018;7(16):1–9.

17. Que P, Guo X, Zhu M. A Comprehensive Comparison between Hybrid and Native App Paradigms. Teoksessa: 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN). 2016. s. 611–4.

18. JetBrains s.r.o. Miscellaneous Tech - The State of Developer Ecosystem in 2021 Infographic [Internet]. JetBrains: Developer Tools for Professionals and Teams. 2021 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://www.jetbrains.com/lp/devecosystem-2021

19. The Apache Software Foundation. Apache Cordova [Internet]. 2022 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://cordova.apache.org/

20. Google. Flutter - Build apps for any screen [Internet]. 2022 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://flutter.dev/

21. Ionic. Cross-Platform Mobile App Development [Internet]. Ionic Framework. 2022 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://ionicframework.com/

22. Meta Open Source. React Native · Learn once, write anywhere [Internet]. 2022 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://reactnative.dev/

23. Unity Technologies. Unity Real-Time Development Platform | 3D, 2D VR & AR Engine [Internet]. 2022 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://unity.com/

24. Microsoft, .NET Foundation. Xamarin | Open-source mobile app platform for .NET [Internet]. Microsoft. 2022 [viitattu 15. maaliskuuta 2022]. Saatavissa: https://dotnet.microsoft.com/en-us/apps/xamarin

25. Unity Technologies. Unity - Manual: Unity User Manual 2020.3 (LTS) [Internet]. 2022 [viitattu 20. maaliskuuta 2022]. Saatavissa: https://docs.unity3d.com/Manual/

26. npm, Inc. keywords:ecosystem:cordova - npm search [Internet]. 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://www.npmjs.com/search?q=keywords:ecosystem:cordova

27. The Apache Software Foundation. Plugin Search - Apache Cordova [Internet]. 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://cordova.apache.org/plugins/

28. Google. Dart packages [Internet]. Dart packages. 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://pub.dev/

29. Ionic Open Source. Capacitor Plugins - Capacitor [Internet]. Capacitor: Cross-platform native runtime for web apps. 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://capacitorjs.com/docs/plugins

30. React Native Directory [Internet]. React Native Directory. 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://reactnative.directory

31. Unity Technologies. Unity Asset Store - The Best Assets for Game Making [Internet]. 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://assetstore.unity.com/

32. Open Source Components for Xamarin [Internet]. Xamarin; 2022 [viitattu 13. huhtikuuta 2022]. Saatavissa: https://github.com/xamarin/XamarinComponents

33. Google. Flutter documentation | Flutter [Internet]. 2022 [viitattu 18. huhtikuuta 2022]. Saatavissa: https://docs.flutter.dev/

34. Meta. Setting up the development environment · React Native [Internet]. 2022 [viitattu 18. huhtikuuta 2022]. Saatavissa: https://reactnative.dev/docs/environment-setup

35. Meta. Out-of-Tree Platforms · React Native [Internet]. 2022 [viitattu 19. huhtikuuta 2022]. Saatavissa: https://reactnative.dev/docs/out-of-tree-platforms

36. Google. Supported platforms [Internet]. 2022 [viitattu 19. huhtikuuta 2022]. Saatavissa: https://docs.flutter.dev/development/tools/sdk/release-notes/supported-platforms

37. Google Developers. android.bluetooth.le | Android Developers [Internet]. 2022 [viitattu 19. huhtikuuta 2022]. Saatavissa: https://developer.android.com/reference/android/bluetooth/le/package-summary

38. react-native-ble-manager [Internet]. Innove; 2022 [viitattu 21. huhtikuuta 2022]. Saatavissa: https://github.com/innoveit/react-native-ble-manager

39. dotintent/react-native-ble-plx [Internet]. intent; 2022 [viitattu 21. huhtikuuta 2022]. Saatavissa: https://github.com/dotintent/react-native-ble-plx

40. Google. Dart overview [Internet]. 2022 [viitattu 18. huhtikuuta 2022]. Saatavissa: https://dart.dev/overview.html

41. MDN. async function - JavaScript | MDN [Internet]. [viitattu 21. huhtikuuta 2022]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

42. Google. Asynchronous programming: futures, async, await [Internet]. [viitattu 21. huhtikuuta 2022]. Saatavissa: https://dart.dev/codelabs/async-await.html

43. Frohner M, Urbauer P, Sauermann S. Bluetooth Low Energy Peripheral Android Health App for Educational and Interoperability Testing Purposes. Studies in health technology and informatics. 2017;236:336-.

44. Yaseen M, Iqbal W, Rashid I, Abbas H, Mohsin M, Saleem K, ym. MARC: A Novel Framework for Detecting MITM Attacks in eHealthcare BLE Systems. Journal of medical systems. 2019;43(11):1–18.