

Roope Kivioja

**Avusteisen kommunikaation ohjelmiston toteuttaminen
progressiivisena WWW-sovelluksena**

Tietotekniikan pro gradu -tutkielma

6. kesäkuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Roope Kivioja

Yhteystiedot: roope.kivioja@gmail.com

Ohjaajat: Jonne Itkonen ja Jukka-Pekka Santanen

Työn nimi: Avusteisen kommunikaation ohjelmiston toteuttaminen progressiivisena WWW-sovelluksena

Title in English: Augmentative and Alternative Communication Through a Progressive Web Application

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 49+6

Tiivistelmä: Progressiiviset WWW-sovellukset ovat verrattain tuore, mutta nopeasti suosituksi noussut tapa kehittää ohjelmistoja erilaisiin tarpeisiin. Tässä pro gradu -työssä Ionic-ohjelmistokehyksellä kehitettyä progressiivista WWW-sovellusta käytettiin tutkimaan avusteisen kommunikaation vaatimuksia, haasteita ja erikoispiirteitä sekä ohjelmistoteknisestä, että käytettävyyssnäkökulmasta. Lisäksi tämän suunnittelutieteellisen tutkimuksen myötä pyrittiin saamaan kehittäjille yleistä tietoa progressiivisten WWW-sovellusten rakenteesta ja ominaisuuksista.

Avainsanat: Ionic, progressiiviset WWW-sovellukset, Angular, TypeScript, avusteinen kommunikaatio, autismi, käytettävyys

Abstract: Progressive web applications are a recent, but very popular way to create applications for different needs. In this master's thesis, a progressive web application developed with the Ionic framework was examined to understand the needs of augmentative and alternative communication. This was done from a software engineering and a usability viewpoint. In addition, this master's thesis offers general insights about the structure and features of progressive web applications.

Keywords: Ionic, progressive web applications, Angular, TypeScript, augmentative and al-

ternative communication, autism, usability

Termiluettelo

Ionic	Mobiililaitteille suunniteltujen progressiivisten WWW-sovellusten kehittämiseen tarkoitettu avoimen lähdekoodin ohjelmistokehys.
Angular	Googlen ylläpitämä avoimen lähdekoodin TypeScript-ohjelmistokehys.
TypeScript	Microsoftin ylläpitämä avoimen lähdekoodin ohjelmointikieli.
Avusteinen kommunikaatio	Kommunikointia, jossa puhetta joko tukee tai korvaa jokin tekninen apuväline tai muu keino kuten viittomakieli.
Electron	GitHubin ylläpitämä progressiivisten WWW-sovellusten kehittämiseen tarkoitettu avoimen lähdekoodin ohjelmistokehys.
Apache Cordova	Avoimen lähdekoodin sovelluskehitysympäristö, joka toimii linkityskerroksena WWW-ohjelmistokehysten ja natiivien ohjelmistokehysten välisenä linkityskerroksena.
GPS	Yhdysvaltain puolustusministeriön ylläpitämä satelliittipaikannusjärjestelmä.
MVC-arkkitehtuuri	Ohjelmistoarkkitehtuurityyli, jossa ohjelma on jaettu kolmeen osaan: malli, näkymä ja ohjain/kontrolleri.
Web-komponentit	Sateenvarjotermi, jolla tarkoitetaan neljää WWW-selainten standardia: kustomoitavat elementit, varjo-dokumenttioliomalli, mallit ja HTML-tuonti.
Kustomoitavat elementit	HTML-standardielementtien ulkopuolisia elementtejä, joita voidaan käyttää standardielementtien seassa.
Varjo-dokumenttioliomalli	Piilotettu osa WWW-sivua, jolla on oma eristetty ympäristö skripteille, CSS-tyylitiedostoille ja HTML-elementeille..
HTML-mallit	HTML-palasia, jotka eivät lataudu välittömästi HTML-sivun auetessa, vaan ne voidaan aktivoida myöhemmin.
HTML-tuonti	Tapa ladata HTML-, CSS- ja JavaScript-tiedostoja yhtenäisinä osina.
CSS	Cascading Style Sheets eli WWW-sivujen tyyliohjeet, joilla voidaan säätää sivun ulkoasua.

Creative Commons -lisenssi Tekijänoikeuslisenssi, joka antaa mahdollisuuden levittää ja välittää teosta eteenpäin, tehdä teoksesta kopioita, sekä tehdä siitä johdannaisteoksia. Creative Commons -lisenssiä voidaan kuitenkin rajoittaa lisäehdoilla.

Kuviot

Kuvio 1. Ionic-sovelluksen perusrakenne	9
Kuvio 2. FreeAAC-sovelluksen näkymäpuu.	22
Kuvio 3. FreeAAC-ohjelmiston kommunikointinäkymä.	23
Kuvio 4. FreeAAC-ohjelmiston korttien luontinäkymä.....	24
Kuvio 5. FreeAAC-ohjelmiston korttisyöbolien valintänäkymä.....	25
Kuvio 6. FreeAAC-sovelluksen muistinkäyttö lepotilassa.....	32
Kuvio 7. FreeAAC-sovelluksen muistinkäyttö lepotilassa graafina.	32
Kuvio 8. FreeAAC-sovelluksen muistinkäyttö käytön aikana.....	33
Kuvio 9. FreeAAC-sovelluksen muistinkäyttö käytön aikana graafina.	33

Sisältö

1	JOHDANTO	1
2	TUTKIMUKSEN TAUSTAT.....	2
3	KOMMUNIKOINTISOVELLUKSEN RAKENTAMISEEN TARVITTAVIA TAUS- TATIETOJA JA MENETELMIÄ	4
3.1	Avusteinen kommunikaatio	4
3.2	Progressiiviset WWW-sovellukset	6
3.3	Ionic-ohjelmistokehyksen rakenne ja ominaisuudet	9
3.3.1	Apache Cordova	10
3.3.2	Angular	10
3.3.3	TypeScript	12
3.4	Käytettävyyden laatutekijät	13
3.4.1	Käytettävyys ja avusteinen kommunikaatio	15
4	KOMMUNIKOINTISOVELLUKSEN SUUNNITTELU JA TOTEUTUS.....	18
4.1	Kehitysympäristö ja -työkalut	18
4.2	Rakenne.....	19
4.3	Ohjelmiston näkymät	22
4.4	Käytettävyysnäkökulma	26
5	TOTEUTUNEEN SOVELLUKSEN ARVIOINTI.....	28
5.1	Käytettävyysnäkökulma	34
6	POHDINTA	37
7	YHTEENVETO.....	38
	LÄHTEET	39
	LIITTEET.....	43
A	FreeAAC-ohjelmiston ominaisuusvaatimukset	43
B	FreeAAC-ohjelmiston tiedostot rivimäärittäin	44
C	Muistinkäyttörajat yksittäiselle ohjelmistolle iOS-laitteissa	45
D	Muistinkäyttörajat yksittäiselle ohjelmistolle Android-laitteissa	47

1 Johdanto

Sosiaalinen kanssakäyminen on ihmisen perustarve ja jokaisella ihmisellä on sisäsyntyinen tarve tulla nähdyksi ja huomioduksi. Ihmisillä on useita eri tapoja viestiä tahdollisesti ja tahtomattaan muiden ympärillään olevien ihmisten kanssa. Sosiaalisuus ja yhteistyö ovat ihmislajin määrittävimpiä tekijöitä. Siksi puhekyvyn menettäminen tai ilman sitä koko elämän eläminen on hyvin suuri haaste ihmisen elämässä. Teknologialla kuitenkin voidaan merkittävästi helpottaa tästä vaikeasta ongelmasta kärsivien ihmisten elämää.

Tässä pro gradu -työssä tutkitaan progressiivisia WWW-sovelluksia, avusteista kommunikointia ja niiden yhteensovittamista. Aihepiiriä ei juurikaan ole aiemmin tutkittu, joten suunnittelutieteellinen tutkimus valikoitui sen takia tämän pro gradu -työn tutkimustyyppiksi. Tutkimustietoa kerättiin varsin paljon muiden kuin tietoteknisten julkaisujen aihepiiristä ja sitä jatkojalostettiin tämän tutkimuksen avulla ohjelmistotekniikan käyttötarpeisiin. Varsinaisen tutkimusartefaktin kehittäminen lisäsi työmäärää tuntuvasti, mutta ilman sitä tätä tutkimusta olisi ollut hankala toteuttaa.

Idea tähän pro gradu -työhön syntyi, kun osallistuin Keski-Suomen Autismiliiton kokoontumisen järjestämiseen vuonna 2015. Vapaamuotoisessa kokoontumisessa pääsin näkemään miten autistiset nuoret käyttävät erilaisia apuvälineitä muiden ihmisten kanssa kommunikointiin. Osa nuorista käytti kommunikointiin paperisia apuvälineitä sekä henkilökohtaisen avustajan tukea, mutta osa pystyi kommunikoimaan itsenäisesti tablettitietokoneen avulla. Käyttäjien mielestä avusteisen kommunikaation ohjelmistojen laadussa ja saatavuudessa eri laitteille kuitenkin oli vielä paljon parantamisen varaa.

Toinen tämän pro gradu -työn aiheeseen vaikuttanut tapahtuma oli Adafy Oy:lle tekemäni projekti, jossa käytin Ionic-ohjelmistokehystä. Tuosta projektista minulle jäi vahva tunne, että progressiivisten WWW-sovellusten suosio tulee lähivuosina kasvamaan ja ne ovat käyttökelpoinen tapa toteuttaa yksinkertaisia ohjelmistoja. Viime vuosina näin on myöskin käynyt, sillä erittäin suositut monialustaiset ohjelmistot kuten Visual Studio Code, Slack ja Discord ovat kaikki Electron-ohjelmistokehysten päälle rakennettuja.

2 Tutkimuksen taustat

Pro gradu -työn tavoitteena oli suunnittelutieteellisen tutkimuksen (engl. *design study*) avulla muodostaa kuva progressiivisten WWW-sovellusten ja Ionic-kehiksen nykytilasta, eduista ja haasteista. Lisäksi tutkimuksessa perehdyttiin avusteiseen kommunikaatioon ja avusteisen kommunikaation mobiilisovellusten kehittämiseen sekä ohjelmistoteknisestä, että käytettävyyšnäkökulmasta. Käytettävyystudkimuksessa otettiin huomioon erityisryhmien tarpeet ja etenkin autististen käyttäjien käytettävyyshaasteet.

Suunnittelutieteellisessä tutkimuksessa tutkimuksen kohteena ovat ihmisen tuotokset eli artefaktit. Suunnittelutieteellisessä tutkimuksessa arvioidaan palvelevatko artefaktit niille asetettuja tarkoituksia, ovatko ne hyödyllisiä ja toimivia sekä miten ne vertautuvat muihin artefakteihin. Suunnittelutieteellinen tutkimus on täten soveltavaa tutkimusta.

Tutkielmaa varten luotiin suunnittelutieteellisen tutkimuksen periaatteiden mukaisesti artefakti, **FreeAAC-ohjelmisto**. FreeAAC-ohjelmistolle asetettiin olemassa olevan tutkimuksen perusteella avusteiseen kommunikaatioon liittyviä ohjelmistoteknisiä ja käytettävyystieteellisiä tavoitteita. Näitä täydennettiin yleisillä ohjelmistoteknisillä ja käytettävyyss pohjaisilla laatuvaatimuksilla. Tavoitteiden ja laatuvaatimusten perusteella pyrittiin lopuksi arvioimaan toteutuneen artefaktin ominaisuuksia.

Pro gradu -työn kohderyhmänä olivat sekä kehittäjät, että loppukäyttäjät. Kehittäjien näkökulmasta työssä keskeisiä tarpeita ja tavoitteita oli saada tietoa siitä miten Ionic-ohjelmistokehitys soveltuu kommunikointisovelluksen kehittämiseen, sekä saamaan selville miten progressiiviset WWW-sovellukset rakentuvat ja mitkä ovat niiden etuja ja haasteita. Loppukäyttäjien eli autismin takia avustettua kommunikaatiota tarvitsevien henkilöiden näkökulmasta keskeinen tarve oli selvittää, mitä laatutekijöitä avusteisen kommunikaation ohjelmistoihin liittyy ja miten hyvin nämä laatutekijät pystytään toteuttamaan progressiivisen WWW-sovelluksen avulla.

Tutkimustyyppin mukaisen selvitystyön ja näiden kahden ryhmän tarpeiden perusteella johdettiin joukko tavoitteita. Työssä haluttiin selvittää avusteisen kommunikaation ohjelmiston toiminnallisten ja käytettävyyden laatutekijät lähdemateriaalin perusteella. Näiden laatu-

kijöiden pohjalta muodostettiin joukko vaatimuksia, joiden pohjalta kehitettiin ohjelmisto ja pyrittiin arvioimaan sitä sekä rakenteellisesti, että edellä mainittujen laatutekijöiden ja vaatimusten toteutumiseen. Lisäksi haluttiin selvittää miten hyvin Ionic-ohjelmistokehys yleisesti sopii progressiivisten WWW-sovellusten kehittämiseen.

Pro gradu -työn ulkopuolelle rajattiin varsinainen käyttäjätutkimus sekä vertaileva tutkimus muihin tarjolla oleviin avusteisen kommunikaation ohjelmistoihin. Aikarajoitteiden vuoksi ohjelmistosta toteutettiin vain prototyypiversio, joten käyttäjätutkimuksen toteuttaminen keskeneräisen ohjelmiston pohjalta olisi ollut haastavaa. Lisäksi autististen koekäyttäjien kanssa toiminen vaatii erikoisjärjestelyjä, jotka haluttiin rajata tämän työn ulkopuolelle. Prototyyppi-vaiheessa olevan ohjelmiston vertailu muihin tarjolla oleviin valmiisiin kaupallisiin ohjelmistoihin ei olisi tuottanut riittävän tarkkoja tuloksia johtopäätösten tekemiseen.

3 Kommunikointisovelluksen rakentamiseen tarvittavia taustatietoja ja menetelmiä

Avusteisen kommunikaation ohjelmiston suunnitteluun ja kehittämiseen tarvitaan perustietämystä kommunikaation haasteista kärsivien ihmisten tarpeista. Aihepiiri on monimutkainen ja useasti yksilökohtaisten vaatimusten sävyttämä, mutta olemassa olevan tutkimuksen pohjalta voidaan kuitenkin muodostaa joukko yleisiä vaatimuksia FreeAAC-ohjelmistolle.

3.1 Avusteinen kommunikaatio

On olemassa useita eri sairauksia ja kehityshäiriöitä, joiden johdosta henkilön kyky muodostaa puhetta voi hetkellisesti tai pysyvästi heikentyä. Puheen muodostamisen ongelmista kärsivä henkilö saattaa joutua turvautumaan arkipäivän kommunikaatiossa erilaisiin apuvälineisiin kommunikoidakseen muiden ihmisten kanssa. Yleisesti näitä viestintämenetelmiä kuvaamaan tarkoitettu termi on **puhetta tukeva ja korvaava kommunikaatio** (engl. *Augmentative and Alternative Communication*, lyh. AAC).

Puhetta tukeva ja korvaava kommunikaatio voidaan jakaa kahtia: avustamattomaan ja avusteiseen. **Avustamattomalla puhetta tukevalla ja korvaavalla kommunikaatiolla** tarkoitetaan kommunikaatiota, jossa ei tarvita apuvälineitä. Viittomakieli on yksi esimerkki avustamattomasta puhetta tukevasta ja korvaavasta kommunikaatiosta, mutta myös ihmisen elekieltä voidaan pitää avustamattomana puhetta tukevana ja korvaavana kommunikaationa.

Avustettu puhetta tukeva ja korvaava kommunikaatio tarkoittaa puolestaan kommunikaatiota, jossa käytetään jotain apuvälinettä. Apuvälineenä voi olla esimerkiksi valokuvia, kommunikaatiotaulu tai elektroninen laite. Tämän perusteella avustettu puhetta tukeva ja korvaava kommunikaatio voidaan jakaa edelleen kahdeksi ryhmäksi: matalan teknologian ja korkean teknologian puhetta tukevaan ja korvaavaan kommunikaatioon. Käyttäjät eivät välttämättä käytä vain yhtä edellä mainituista tyypeistä. (Sigafos ja Drasgow 2001)

Puhetta tukevaa ja korvaavaa kommunikaatiota voidaan tarjota puheen muodostamisen ongelmista kärsiville myös tietoteknisten sovellusten avulla, joiden kautta käyttäjä kirjoittaa jo-

ko suoraan tekstiä tai kommunikoi valitsemalla symboleja. Esimerkiksi autistisilla käyttäjillä on tyypillisesti hyvin henkilö- ja yksityiskohtaisia tarpeita puhetta tukevalle ja korvaavalle kommunikaatiolle, joten tietoteknisten sovellusten suhteellisen helppo muokattavuus puoltaa niiden käyttöä perinteisempien puhetta ja kommunikointia korvaavien apuvälineiden sijaan. Puhetta tukevat ja korvaavat kommunikointisovellukset käyttävät yleisimmin symboleja. Usein symbolit ryhmitellään kortteihin, joita on helppo vaihtaa tilanteeseen sopivaksi. Autisteille tyypillistä on vahva visuaalis-avaruudellinen hahmotuskyky, joten tutkimuksen mukaan piirroksiin ja valokuvaan liitetyt merkitykset ovat tälle ryhmälle luonteva tapa kommunikoida. Vaughnin ja Hornerin tutkimuksessa (Vaughn ja Horner 1995) Karl-nimisen autistisen koehenkilön haastava käytös ja aggressio vähenivät, kun pelkästään verbaalisesti annettujen ruokavaihtoehtojen rinnalle tuotiin kuvat ruoka-annoksista. Symbolipohjaiselle puhetta tukevalle ja korvaavalle kommunikaatiolle on siis sekä tutkimuspohjaista näyttöä että käytännön kokemuksiin perustuvaa kannustetta.

Puhetta tukevaan ja korvaavaan kommunikointisovellukseen voidaan liittää myös symboleita lukeva ääniominaisuus. Ääniominaisuus mahdollistaa kommunikoinnin näköyhteyden ulkopuolelle, vähentää symbolien tulkitsijan läsnäolon pakollisuutta ja helpottaa pidempien viestien rakentamista puhetta tukevassa ja korvaavassa kommunikointisovelluksessa. (Nunes 2008)

Ääniominaisuuden lisäämisellä avustetusta puhetta tukevasta ja korvaavasta kommunikaatiosta voidaan tehdä **monimodaalista** (engl. *multimodal*). Monimodaalisuus tarkoittaa usean eri kommunikaatiotavan kautta tapahtuvaa kommunikaatiota. Monimodaalinen kommunikaatio voi tapahtua eri tapojen kautta yhtäaikaaisesti tai peräkkäin. Puhetta tukevasta ja korvaavasta kommunikaatiosta kannattaa tehdä monimodaalista useista eri syistä. Ensinnäkin suurin osa kaikesta kommunikaatiosta on monimodaalista, sillä toiselle ihmiselle puhuttaessa on tavallista selkeyttää sanomaa ilmein ja elein. Toiseksi puhetta tukevaa ja korvaavaa kommunikaatiota käyttävä henkilön tulee useasti kommunikoida muiden puhetta tukevaa ja korvaavaa kommunikaatiota käyttävien henkilöiden kanssa, jolloin vaihtoehtoja on hyödyntää. Kolmas merkittävä syy on se, että eri puhetta tukevissa ja korvaavissa kommunikaatiotavoissa on vahvuuksia ja heikkouksia, joten eri kommunikaatiotapoja sekoittamalla voidaan korvata yksittäisen kommunikaatiotavan heikkouksia. (Sigafos ja Drasgow 2001)

Puhetta tukevaan ja korvaavaan kommunikaatioon liittyy useita haasteita. Koska syyt ja tarpeet puhetta korvaavalle kommunikaatiolle ovat hyvin moninaisia, yhtenäisiä käytäntöjä on vaikea muodostaa ja tarve henkilökohtaiselle räätälöinnille on suurta. Lisäksi puhetta tukevaa ja korvaavaa kommunikaatiota tarvitsevien henkilöiden määrä ei ole väestötasolla kovin suuri: arviolta 1,3 % yhdysvaltalaisista ei kykene kommunikoimaan puheen välityksellä (Beukelman ja Mirenda 2013). Toisessa tutkimuksessa Iso-Britannian kansalaisista 0,5 % arvioitiin jonkinlaisten avusteisen kommunikaation tarpeessa oleviksi (Enderby ym. 2013). Tämän takia puhetta tukevia ja korvaavien kommunikointisovellusten tuottamiseen on hyvin vähän kaupallista kannustetta.

3.2 Progressiiviset WWW-sovellukset

Progressiiviset sovellukset (engl. *Progressive Web Applications*, lyh. *PWAs*) ovat selaimessa ajettavia WWW-sovelluksia, joiden ulkoasu määrittyy alustakohtaisesti niin, että niiden ulkoasu on mahdollisimman yhdenmukainen laitteen natiivien sovellusten kanssa. Selainpohjaisuuden takia progressiiviset sovellukset pystyvät käyttämään tarjolla olevia sovellusympäristön resursseja joustavasti sen sijaan, että ne itse määrittäisivät vaatimukset toiminnalleen. Termi on verrattain tuore ja vakiintumaton, sillä esimerkiksi Ionic-kehiksen dokumentaatiossa käytetään myös termiä **hybridisovellus** (engl. *hybrid application*).

Progressiivisten sovellusten ilmeisin hyöty on alustariippumattomuus. Samaa sovellusta voidaan käyttää kaikissa ympäristöissä, jotka tukevat moderneja WWW-selaimia. Eri versioita samasta sovelluksesta ei tarvitse kehittää ja ylläpitää erikseen, joten progressiivisten sovellusten avulla voidaan säästää kalliita työresursseja. Hyvä esimerkki progressiivisten sovellusten käyttöä tukevasta markkinatilanteesta on nykyisten älytelevisioiden kirjava tarjonta, sillä valmistajien omien käyttöjärjestelmien lisäksi muunmuassa Apple, Amazon ja Roku kehittävät televisioon liitettäviä laitteita, joiden avulla käyttäjä voi ajaa erilaisia sovelluksia. Näiden kaikkien laitteiden tukeminen olisi hyvin vaikeaa perinteisten sovellusten avulla. (Frankston 2018)

Tällä hetkellä erityisesti Google panostaa progressiivisten sovellusten kehittämiseen Chrome-selaimen kehitysympäristön yhteydessä. Googlen (Google 2018) mukaan progressiiviset so-

vellukset tarjoavat perinteisiin WWW-sovelluksiin verrattuna enemmän luotettavuutta, käytettävyyttä ja monipuolisempaa sisältöä. Yrityksen mukaan luotettavuus paranee, sillä progressiiviset sovellukset pystyvät tarjoamaan sisältöä myös ilman verkkoyhteyttä. Google myös väittää, että progressiivisten WWW-sovellusten kohdalla käytettävyyttä parantaa nopeammin käyttäjän komentoihin vastaava käyttöliittymä ja sovellusmaisuus puolestaan parantaa käyttäjän immersiota.

Googlen Chrome-selain on rakennettu avoimen lähdekoodin Chromium-selaimen päälle. Yksi merkittävimmistä Chromium-selaimeen pohjautuvista progressiivisten WWW-sovellusten kehitykseen tarkoitetuista kehyksistä on Electron. Electron on Microsoftin nykyisin omistaman GitHubin kehittämä ja ylläpitämä. Esimerkkejä Electron-sovelluksista ovat Discord, Slack ja Visual Studio Code. Microsoftin päätös muuttaa Edge-selain Chromium-pohjaiseksi saattoi johtua ainakin osittain progressiivisten WWW-sovellusten vaatimista ominaisuuksista kuten tehokkaammasta muistinhallinnasta.

Twitter, AliExpress ja Lancôme ottivat vuonna 2017 käyttöön progressiiviset sovellukset ja julkaistut tulokset ovat olleet positiivisia AppInstitute-sivuston raportin mukaan. Twitter onnistui uuden progressiivisen sovelluksensa ansiosta lisäämään sivupäivityksiä 65% käyttäjäsessiota kohden, lisäämään lähetettyjen Twitter-viestien määrää 75% ja vähentämään käytön lopettamista 20%. Lisäksi raportin mukaan uusi sovellus käytti vähemmän kuin 3% natiivin sovelluksen vaatimasta muistitilasta ja vähensi datan käyttöä 70%. Datan käytön määrän vähentyminen on erityisen merkittävää, koska Twitter arvioi, että vuonna 2017 45% sen sisällöstä ladattiin 2G-verkon läpi. (AppInstitute 2017)

Raportin mukaan AliExpress ja Lancôme ovat molemmat verkkokauppoja, joilla on ollut ongelmia mobiiliverkkokauppojensa tehokkuuden kanssa. Progressiiviseen sovellukseen siirtymällä AliExpress lisäsi uusien asiakkaiden myyntitapahtumien määrää 104%:lla ja Lancôme 17%:lla. Lisäksi AliExpress tuplasi sivunlatausten määrän sekä kasvatti sessioiden pituutta 74%:lla. Lancôme puolestaan onnistui lisäämään iOS-sessioiden määrää 53%:lla. Edellä mainitut tulokset eivät kuitenkaan ole suoraan yleistettävissä yleisemmin progressiivisten sovellusten vaikutuksiin, sillä niiden otoskoko on erittäin pieni. (AppInstitute 2017)

Progressiivisten WWW-sovellusten käyttö ei kuitenkaan ole ongelmatonta. Ensinnäkin, kos-

ka progressiivisten WWW-sovellusten hyödyntämät teknologiset kehitysaskleet ja niiden käyttöön kannustavat taloudelliset tekijät ovat verrattain tuoreita, vaaditaan kehittäjiltä paljon uusien asioiden opettelua sekä vakiintumattomien työkalujen ja kehysten käyttämistä kehitystyössä. Toinen merkittävä haaste voi olla tiedon tallentaminen, sillä selaimen toimintaan pohjautuvalle sovellukselle ei ole varattu tietoturvan takia samoja oikeuksia kuin natiiveille sovelluksille. Kolmas ongelma voi tulla eteen käytettävyydessä, sillä selaimessa toimivan sovelluksen vaatimat resurssit saattavat hidastaa sovelluksen toimintaa. (Divante 2018)

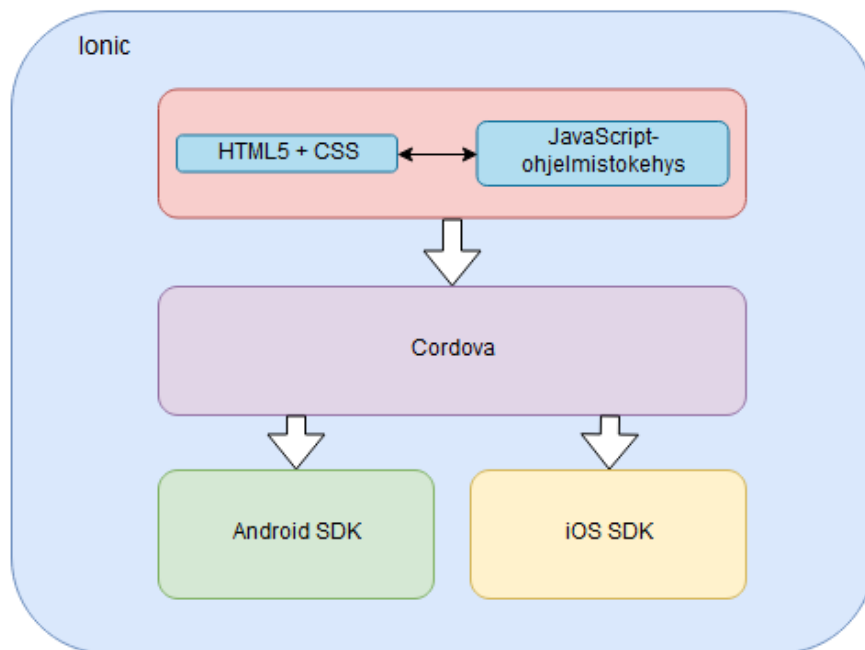
Myös progressiivisten sovellusten keskusmuistinkäyttöä on kritisoitu. Progressiiviset WWW-sovellukset vaativat perinteisiä sovelluksia enemmän muistia, koska niitä ajetaan selainympäristössä, jossa pelkästään sovelluksen lisäksi täytyy käyttää resursseja myös selainalustan tarpeisiin. On vaikea arvioida, kuinka suuri osa muistinkäytöstä johtuu sovelluksen toteutuksesta ja kuinka suuri osa itse ohjelmistokehyksestä, mutta hyvin yksinkertaisissakin testeissä on saatu tuloksia, jotka viittaavat moninkertaiseen keskusmuistinkäyttöön natiiveihin sovelluksiin verrattuna (O'Kelly 2017).

Suuremman muistinkäytön lisäksi myös energiankulutus saattaa olla korkeampi progressiivisissa WWW-sovelluksissa. Ohjelmistoanalytiikkatyökalua tarjoavan Greenspectorin mukaan (Pichon 2017) Twitterin progressiivisen WWW-sovelluksen energiankulutus valmiustilassa on 12-25% korkeampi asetuksista riippuen. Greenspectorin motiivit saattavat kuitenkin olla ristiriidassa puolueettoman ohjelmistoanalyysin kanssa, joten tulokset eivät ole välttämättä täysin luotettavia. Voidaan kuitenkin olettaa, että ainakin lisääntynyt muistinkäyttö aiheuttaa myös lisääntynyttä energiankulutusta ja täten haittaa mobiililaitteiden pitkäaikaista käyttöä.

Progressiiviset WWW-sovellukset sopivat siis parhaiten tilanteisiin, joissa vähän muistia vaativan sovelluksen halutaan tukevan useita eri alustoja, mutta joissa tavallinen WWW-sivu ei riitä. Avusteisen kommunikaation sovelluksien vaatimukset osuvat hyvin yhteen progressiivisten WWW-sovellusten vaatimusten kanssa, sillä kuvakorttien näyttäminen ei ole muistiintensiivistä, mutta toisaalta taas avusteisen kommunikaation sovellusten halutaan olevan helposti muovattavissa henkilökohtaisiin tarpeisiin sopiviksi sekä toimivan useilla eri alustoilla.

3.3 Ionic-ohjelmistokehityksen rakenne ja ominaisuudet

Ionic on yksi suosituimpia progressiivisten sovellusten kehittämiseen tarkoitetuista ohjelmistokehyksistä (Belitsoft 2019). Electronin ollessa työasemille suunnattu kehys, Ionic keskittyy mobiililaitteiden vaatimukseen. Ionic on avointa lähdekoodia ja hyödyntää Apache Cordova -ympäristöä sekä suosituimpia frontend-ohjelmistokehyksiä. Aiemmin Ionic tuki vain Angular-ohjelmistokehystä, mutta Ionicin versiosta 4 lähtien kehittäjällä on mahdollisuus valita itse käyttämänsä frontend-kehys. Ionic on MIT-lisenssin alainen ja täten avointa lähdekoodia. Angular-ohjelmistokehys perustuu TypeScript-ohjelmointikieleen. Ionicin perusrakenne on kuvattuna kaaviossa 1. Ohjelmistokehityksenä Ionicin päämäärä on tarjota kehittäjälle oikealta näyttävä ja visuaalisesti toimiva sovellus. Sen ei ole tarkoitus korvata tyypillisiä JavaScript-kirjastoja, vaan se toimii niiden tukena. (Ionic 2019)



Kuvio 1. Ionic-sovelluksen perusrakenne.

Ionicia ylläpitää ja kehittää Ben Sperry ja Max Lynchin perustama samanniminen yritys. Avoimen lähdekoodin projektina Ionic-kehityksen kehitykseen ja ylläpitoon pääsee osallistumaan GitHub-sivuston välityksellä kuka tahansa. Ionicin viimeisin pääversio on 4.0.0. Viime aikoina Ionicin kehitystyössä painoarvoa on annettu erityisesti Ionic-kehityksen ja Angularin välisten riippuvuuksien vähentämiseen, jotta Ionicia voitaisiin käyttää muiden kehitysten ku-

ten Reactin ja Vue.js:n yhteydessä.

Ionic koostuu komponenteista. Ionicin komponentit ovat uudelleenkäytettäviä käyttöliittymäelementtejä, jotka toimivat sovelluksen käyttöliittymän rakennusosina. Niiden avulla sovellus näyttää yhtenäiseltä ja käyttöliittymän kehitystyö nopeutuu. Komponentit koostuvat HTML:stä, CSS:stä ja JavaScriptistä. Esimerkkejä tällaisista komponenteista ovat muunmuassa painikkeet, välilehdet ja erilaiset listat. Ionicin toteutuksessa on pyritty tukemaan näiden komponenttien mahdollisimman joustavaa räätälöintiä (Ionic 2019).

Ionicin ulkoasu perustuu teemoihin. Myös teemat lisäävät sovelluksen ulkoasun yhtenäisyyttä. Teemat myös mukautuvat eri alustojen ulkoasustandardien mukaisiksi kehittäjän niin halutessa. Esimerkiksi Android- ja iOS-mobiilikäyttöjärjestelmille suunnattujen sovellusten ulkoasu poikkeaa toisistaan, jos käytetään Ionicin oletusteemoja. Teemojen käyttäminen parantaa sovelluksen käytettävyyttä tekemällä sen ulkoasusta ennustettavamman ja tutumman loppukäyttäjälle. Ionicin oletusteemat noudattavat Applen iOS-designperiaatteita (Apple 2019) sekä Googlen Material Design -määrittäjiä (Google 2019).

3.3.1 Apache Cordova

Apache Cordova (aikaisemmin PhoneGap) on alunperin Nitobin kehittämä sovelluskehitysympäristö mobiililaitteille. Adobe osti Nitobin vuonna 2011 ja myöhemmin uudelleenjulkaisi Apache Cordovan avoimena lähdekoodina. Apache Cordova toimii WWW-sovelluskehityksen ja natiivien sovelluskehityksen välisenä linkityskerroksena (Waranashiwar ja Ukey 2018). Apache Cordovan avulla Ionic ja muut Apache Cordovan päälle rakennetut ohjelmistokehykset voivat toimia syvemmällä tasolla kuin tyypillinen WWW-sovellus, sillä niille tarjoutuu rajapinta suoraan natiiveihin sovelluskehityksiin. Näin progressiivisille WWW-sovelluksille tarjoutuu mahdollisuus käyttää muunmuassa mobiililaitteen kameraa ja GPS-paikannusta.

3.3.2 Angular

Angular on Googlen Angular Teamin ylläpitämä TypeScript-pohjainen käyttöliittymäkehys. Se on jatkoa aiemmin ilmestyneelle AngularJS-kehykselle. Alkuperäinen AngularJS-kehys julkistettiin vuonna 2010, ja se oli ensimmäinen suosittu ohjelmistokehys dynaamisten

HTML-sivujen kehittämiseen mahdollistaen tehokkaamman yhden sivun WWW-sovellusten rakentamisen. Angular julkaistiin vuonna 2016, ja se on kokonaan uudelleenohjelmoitu versio AngularJS:stä.

AngularJS perustuu MVC-arkkitehtuuriin, kun taas uudempi Angular on komponenttipohjainen. Komponenttipohjainen arkkitehtuuri pyrkii tarjoamaan paremman uudelleenkäytettävyyden, luettavuuden, testattavuuden ja ylläpidettävyyden. Ohjelma jaetaan itsenäisiin komponentteihin, joita voidaan käyttää useasti ja niiden itsenäisyys helpottaa yksikkötestaamista. Itsenäiset komponentit ovat huomattavasti helpommin ymmärrettävissä ja niiden korvaaminen on joustavampaa, mikä parantaa ylläpidettävyyttä. (AltexSoft 2018)

Angularin komponenttipohjainen rakenne perustuu kolmeen hiljattain ilmestyneeseen teknologiaan: Web-komponentteihin (engl. *Web Components*), JavaScriptin ES2015-standardiin ja TypeScript-ohjelmointikieleen.

Web-komponentit on sateenvarjotermi, jolla tarkoitetaan neljää WWW-selaimissa yleistävää standardia: kustomoitavat elementit (engl. *custom elements*), varjo-dokumenttioliomalli (engl. *shadow DOM*), mallit (engl. *templates*) ja HTML-tuonti (engl. *HTML imports*).

Kustomoitavat HTML-elementit ovat HTML-standardielementtien ulkopuolisia elementtejä, joita voidaan käyttää standardielementtien seassa. Kustomoitava HTML-elementti irroittaa komponentin sivun muista osista, joten se mahdollistaa komponentin eristämisen.

Varjo-dokumenttioliomalli on piilotettu osa WWW-sivua, jolla on oma eristetty ympäristö skripteille, CSS-tyylitiedostoille ja HTML-elementeille. Varjo-dokumenttioliomallin elementit ja tyylit eivät vaikuta varjo-dokumenttioliomallin ulkopuolisiin alueisiin ja vastavuoroisesti muut sivun elementit ja tyylit eivät vaikuta varjo-dokumenttioliomallin alaisiin osiin. Tätä eristettyä aluetta voidaan täten käyttää komponentin piirtämiseen.

Mallit ovat HTML-palasia, jotka eivät lataudu välittömästi HTML-sivun auetessa, vaan ne voidaan aktivoida JavaScriptillä myöhemmin. Malleista on useita toteutuksia eri kehityksissä, mutta Web-komponentit standardisoivat mallien rakentamisen ja tarjoavat suoran tuen niiden hyödyntämiselle selaimessa. Malleja käyttämällä varjo-dokumenttioliomalliin piilotetusta sisällöstä voidaan tehdä dynaamista.

Viimeinen Web-komponenttien osa on **HTML-tuonti**. HTML-tuonnin avulla HTML-, CSS- ja JavaScript-tiedostoja voidaan ladata yhtenäisinä osina. Angular ei käytä HTML-tuontia, vaan se käyttää JavaScriptin moduulilatausta. (Arora ja Hennessy 2018)

3.3.3 TypeScript

TypeScript on avoimen lähdekoodin ohjelmointikieli, jota ylläpitää Microsoft. TypeScript kääntyy suorituvaiheessa JavaScriptiksi, ja se on tarkoitettu tukemaan JavaScript-ohjelmien kehitystä parantamalla JavaScriptin ominaisuuksia. TypeScript sisältää ES 2015 -standardin mukaiset ominaisuudet sekä lisäksi se tarjoaa ohjelmoijan käyttöön tyypit ja koristeilijat. Angular-ohjelmointikehityksen ohjelmointiin käytetään TypeScript-ohjelmointikieltä.

TypeScriptin on tarkoitus tarjota Microsoftin .NET-ympäristöön tottuneille ohjelmoijille olionlähtöisempi lähestymistapa JavaScript-kehitykseen (Maharry 2013). JavaScript on suosittu ohjelmointikieli, mutta varsinkin ohjelmiston lähdekoodin määrän kasvaessa sen heikkoudet alkavat tulla esiin. TypeScript pyrkii puuttumaan näihin ongelmiin tarjoamalla ohjelmoijalle moduulijärjestelmän, luokat, rajapinnat ja staattisen tyyppityksen (Bierman, Abadi ja Torgeresen 2014).

Ohjelmointikielen **modulaarisuudella** viitataan yleisesti itsenäisistä osista kasattaviin suurempiin kokonaisuuksiin. TypeScriptissä modulaarisuus on toteutettu siten, että muuttujat, funktiot, luokat ja muut vastaavat ohjelmointikielen perusrakenteet ovat olemassa vain moduulien sisällä, ellei niitä erikseen esitellä muille moduuleille (Microsoft 2019c). Tämä vähentää ohjelman eri osien välistä riippuvuutta toisistaan, mikä helpottaa muutosten tekemistä ohjelmistoon.

Perinteisesti JavaScript on käyttänyt uudestikäytettävien komponenttien rakentamiseen funktioita ja prototyyppipohjaista perintää. Iso osa nykyohjelmoijista ei kuitenkaan ole tottunut käyttämään edellä mainittuja keinoja, vaan nykyisin suosituin lähestymistapa uudelleenkäytettävyyteen on luokkien käyttäminen. Myös JavaScriptin oliotukea on pyritty parantelemaan sen viimeisimmissä versioissa. (Microsoft 2019a)

Uudelleenkäytettävyyden lisäksi TypeScriptin luokkien avulla saadaan aikaan modulaarisia komponentteja, joita on helpompi ylläpitää ja skaalata. Virheiden löytämistä helpottaa se,

että luokkien avulla ohjelmakoodin rajoista saadaan selkeämpiä. Skaalautuvuutta varten täytyy monesti pystyä korvaamaan vanhoja komponentteja uusilla ja luokkia käyttämällä myös tämä on helpompaa.

Rajapintoja käytetään kuvaamaan luokkien ominaisuuksia. TypeScriptissä rajapintoja käytetään tyyppien nimeämiseen ja niiden avulla voidaan tehdä olioiden välisiä sopimuksia ohjelmoijan itse laatiman ohjelmistokoodin sisällä sekä myös ohjelmoijan oman ohjelmistokoodin ja muiden kehittämien kirjastojen välillä. (Microsoft 2019b) Esimerkiksi Swagger-niminen työkalu generoi .NET-palvelinkoodista valmiita rajapintoja, joita TypeScript-pohjaisen WWW-sovelluksen on helpompi käyttää. Rajapinnat ovat luonnollinen osa luokkiin perustuvaa ohjelmointia. Niiden avulla luokan toteutus voidaan erottaa sen ulkopuolelle näkyvistä osista ja jakaa luokkia ryhmiin. Kuten luokkienkin kohdalla, rajapintojen käyttäminen voi parantaa ohjelman ylläpidettävyyttä, skaalautuvuutta ja osien uudelleenkäytettävyyttä.

JavaScriptistä poiketen TypeScript on staattisesti tyypitetty kieli. Tyypitys tarkoittaa sitä, että muuttujien ja olioiden tyypit täytyy määritellä ohjelmakoodissa. Dynaamisesti tyypitetyissä kielissä kuten JavaScriptissä muuttujien tyypit päätellään käännöksen aikana. Staattisen tyypityksen avulla ohjelmoijan on helpompi nähdä suoraan, että ohjelman tieto on oikean tyyppistä sitä käsitellessä. Iso osa tyyppivirheistä jää tällöin kiinni jo ennen kääntämistä kun taas JavaScriptillä ohjelmoidun ohjelman kohdalla näin ei käy. Toisaalta vahva tyypitys aiheuttaa lisätyötä tyyppimuunnosten takia.

3.4 Käytettävyyden laatutekijät

Kaikkiin ihmisen tuottamiin esineisiin ja asioihin liittyy tiedostettu tai tiedostamaton suunnitteluprosessi. Vaikka ihmiset ovat suorittaneet aktiivista suunnittelua esihistoriallisista ajoista saakka, on tietoisten suunnitteluprosessien tutkimus verrattain tuoretta. Nykyinen suunnittelun kenttä voidaan jakaa karkeasti kolmeen eri osaan: teollinen suunnittelu, käytettävyyssuunnittelu ja kokemussuunnittelu (Norman 2013). Jokapäiväisessä kommunikaatiossa avustavan sovelluksen suunnittelussa tulee olla erityisen kiinnostuneita käytettävyyssuunnittelusta.

Hyvin suunniteltu sovellus on miellyttävä käyttää ja ohjaa käyttäjää käyttämään sovellus-

ta oikealla tavalla. Tämä on erityisen tärkeää sovelluksissa, joita on tarkoitus käyttää arkisissa tilanteissa, jotka vaativat reaaliaikaista reagointia muiden ihmisten kommunikointiin. Avustettua kommunikaatiota käyttävillä ryhmillä on myös omia käytettävyystarpeita, joiden huomioimista varten täytyy ymmärtää käytettävyyttä tavallista laajemmassa kontekstissa.

Ohjelman käyttöliittymän **käytettävyys** (engl. *usability*) on laadullinen määre, joka voidaan jakaa viiteen laadulliseen osa-alueeseen: opittavuus (engl. *learnability*), tehokkuus (engl. *efficiency*), muistettavuus (engl. *memorability*), virhealttius (engl. *errors*) ja tyydyttävyyys (engl. *satisfaction*). (Nielsen 2012)

Opittavuus tarkoittaa sitä miten helppo käyttöliittymää on oppia käyttämään. Toisella tapaa ilmaistuna, opittavuus tarkoittaa ohjelmiston kykyä opettaa käyttäjälleen oikea tapa käyttää ohjelmistoa. Opittavuutta vastaava termi on **löydettävyys** (engl. *discoverability*). Opittavuutta lisääviä ominaisuuksia ovat muunmuassa muistettavuus, loogisuus, toistettavuus ja yhdenmukaisuus. Opittavuudeltaan hyvä ohjelma ilmaisee käyttäjälleen tehokkaasti mitä toiminnallisuksia se sisältää, mitä eri toiminnot tarkoittavat ja kuinka eri toiminnallisuksia käytetään. (Shamsuddin ym. 2012)

Käyttöliittymän **tehokkuus** määrittyy siitä miten nopeasti käyttäjät voivat suorittaa ohjelman toimintoja sen jälkeen kun ohjelman käyttöliittymää on opittu käyttämään. Se on määritelmä tai mitta sille, miten helposti ja nopeasti haluttu toiminto voidaan suorittaa tuttua käyttöliittymää käyttämällä. Tehokkuutta voidaan suoraan mitata esimerkiksi kulunutta aikaa tai välivaiheiden määrää mittaamalla.

Muistettavuus on määre, jonka avulla arvioidaan miten nopeasti käyttäjä muistaa toiminnot ohjelman käyttämisen lopettamisen ja uudelleen aloittamisen jälkeen. Se mittaa sitä, miten helppo käyttäjän on käyttää ohjelmistoa sen jälkeen, kun ohjelmiston edellisestä käyttökerasta on kulunut aikaa. Muistettavuutta on hankala mitata suoraan tyypillisten käyttäjätutkimusmetodien avulla, mutta esimerkiksi Affordable Usability -sivuston (Usability 2011) mukaan sitä voidaan WWW-sovellusten yhteydessä tutkia erilaisten verkkosivuanalytiikka-työkalujen avulla.

Virhealttiudella tarkoitetaan käyttäjien tekemien virheiden määrää ja niiden laatua sekä kuinka helppoa niistä selviäminen on. Käyttäjien tekemien virheiden määrän minimointi on

yksi käytettävyyssuunnittelun tärkeimmistä tavoitteista. Don Normanin (Norman 2013) mukaan käyttäjää ei juuri koskaan pitäisi syyttää tekemistään virheistä vaan suurin osa käyttövirheistä johtuu huonosta suunnittelusta.

Tyydyttävyyys kertoo, miten tyytyväinen käyttäjä on käyttöliittymään. Tyydyttävyyys mittaa sitä miten hyvin käyttöliittymä ottaa huomioon käyttäjän tarpeet ja miten miellyttävää käyttö on. Tyydyttävyyttä ymmärtääkseen täytyy tietää, että käyttöliittymä ja käyttäjäkokemus ovat kaksi eri asiaa. Käyttäjäkokemus ja sen tyydyttävyyys tai tyydyttämättömyys on seurausta useasta eri tekijästä.

Visuaalinen ulkoasu on osatekijä käyttöliittymän tyydyttävyyttä arvioidessa, mutta myös halutuilla ominaisuuksilla ja niiden toimivuudella on merkitystä tyydyttävyyteen. Jos käyttäjä ei löydä haluamiaan toimintoja, osa halutuista ominaisuuksista puuttuu tai ne on hankalta löytää, ohjelmiston tyydyttävyyys on matala. Tyydyttävyyys on viidestä edellä listatusta käytettävyystekijästä kaikista subjektiivisin.

On olemassa muitakin laadullisia määreitä, kuten **käyttökelpoisuus** (engl. *utility*), jolla määritellään ohjelman tarjoamien eri ominaisuuksien määrää ja vertaamista haluttuihin ominaisuuksiin. Käytettävyyttä ja käyttökelpoisuutta yhtä aikaa tarkistelemalla saadaan aikaan kuva ohjelman **hyödyllisyydestä** (engl. *usefulness*). (Nielsen 2012)

3.4.1 Käytettävyyys ja avusteinen kommunikaatio

Avusteiseen kommunikaatioon liittyy omia käytettävyyshaasteita. Tarve avusteiseen kommunikaatioon voi johtua useista eri syistä, joten käyttäjäkirjo ja käyttäjien henkilökohtaisten tarpeiden määrä on suuri. Jo pelkästään tämän havainnon perusteella voidaan olettaa ohjelmiston muokattavuuden olevan tärkeää.

CP-vammaisten avusteista kommunikaatiota tutkineessa tutkimuksessa (Clarke ja Wilkinson 2005) yksi merkittävä käytettävyystekijä on avusteista kommunikaatiota käyttävän henkilön hidas toiminta. Avusteista kommunikaatiota käyttävä henkilö ei välttämättä pysty reagoimaan nopeasti eri keskusteluaiheisiin tai muodostamaan riittävän paljon kommunikaatiota, jolloin keskustelukumppani joutuu arvaamaan, mitä avusteista kommunikaatiota käyttävä henkilö yrittää sanoa. Käytettävyyden näkökulmasta on myös huomioitava, että käyttäjällä

on riittävästi aikaa suorittaa valintoja käyttöliittymässä ilman, että näkymä vaihtuu tai käyttöliittymäelementit muuttuvat.

Comunicador-nimistä espanjankielisille avusteisen kommunikaation käyttäjille tarkoitettua ohjelmistoa käsittelevässä tutkimuksessa (Cagigas, Olalla ja Sanchez 2005) nostetaan esiin kontekstiriippuvaisten valintojen tärkeys. Koska kommunikointi on usein vaivalloista avusteisten kommunikaation sovellusten kautta, avusteisen kommunikaation sovelluksessa on hyvä olla mahdollisuus ryhmitellä esivalmistellut sanat tai kuvat tilanteiden mukaan. Esimerkiksi työ-, harrastus- ja arkitilanteissa tarvitaan usein hyvin erilaisia sanavarastoja. Kontekstin valitsemisen lisäksi sanoja tai kuvia täytyy myös pystyä ryhmittelemään ja järjestelemään kontekstiryhmien sisällä.

Autismi vaikuttaa monella tapaa ihmisen kykyyn havainnoida ympäristöä ja sen myötä kykyyn käyttää sovelluksia. Iso-Britannian The National Autistic Society listaa verkkosivullaan (The National Autistic Society 2018) autistisille ihmisille sopivien verkkosivujen visuaalisen toteutuksen päävaatimukset. Listauksessa painotetaan erityisesti visuaalisen ilmeen selkeyttä, staattisuutta ja yksiselitteisyyttä. Lisäksi koekäyttäjien roolia korostetaan.

Ilmaiseksi tarjolla olevia avusteisen kommunikaation mobiilisovelluksia tutkineen tutkimuksen (Khan, Tahir ja Raza 2014) mukaan autistisilla käyttäjillä on ominaisuus- ja käytettävyystarpeita, joita tarjolla olleissa sovelluksissa ei ole. Tutkimuksen mukaan mahdollisuus kuvien ottamiseen sekä äänen tallentamiseen on erittäin hyödyllinen ominaisuus avusteisia kommunikaatiosovelluksia käyttäville autisteille. Kuvat eri sijainneista ja ihmisistä auttavat päivittäisessä kommunikaatiossa huomattavasti.

Kommunikaatiosovelluksessa tulisi myös olla mahdollisuus säätää ohjelman asetuksia hallintapaneelin kautta. Hallintapaneelin tulisi olla salasanasuojattu, jotta käyttäjä ei pääse vahingossa poistamaan tärkeitä asetuksia tai kuvia sovelluksesta. Yhteen näyttöön ei saa mahdollistaa liian paljon informaatiota, sillä autismiin liittyy usein aistiyliherkkyyttä (National Autistic Society 2019), mikä heikentää autistisen henkilön kykyä ymmärtää isoja määriä informaatiota kerrallaan. Tämän takia esimerkiksi kommunikaatiosovelluksen korteilla olevien kuvien määrää tulisi pystyä säätämään sovelluksen hallintapaneelisti.

Kommunikaatiosovellusten ääniominaisuus voi auttaa käyttäjää oppimaan sovelluksen käyt-

töä nopeammin. Äänisyntetisaattoreita Martin-nimisen pojan avulla tutkineessa tutkimuksessa (Schlosser 1999) havaittiin, että Martin oppi muodostamaan uusia lauseita helpommin, kun avusteisen kommunikaation sovellus luki kirjoitetun tekstin ääneen. Tutkimuksen luotettavuutta kuitenkin heikentää se, että tutkimuksessa tutkittiin vain yhden koehenkilön oppimistuloksia.

Symbolien käyttöä sovelluksien käyttöliittymissä tekstin korvikkeena pidetään yleisesti hyvänä tapana säästää tilaa, vähentää lukemisen aiheuttamaa kognitiivista kuormaa sekä kiertää tarvetta tekstien kääntämiseen useille eri kielille. Autistisilla henkilöillä esiintyy kuitenkin useasti hahmotusongelmia, jotka haittaavat symbolien merkityksen ymmärtämistä. Autististen henkilöiden on keskimäärin haastavaa ymmärtää abstraktien ja vähän ikonisuutta sisältävien symbolien merkitystä. Esimerkiksi palloa esittävä symboli on autisteille helppo ymmärtää, mutta abstraktien asioiden kuten *mene*-verbin yhdistäminen nuoli-symboliin voi olla haastavaa. (Kozleski 1991)

Yksi jonkin verran tutkittu seikka on autismin vaikutus ihmisen suosikkiväreihin. Grandgeorgen ja Masatakan (Grandgeorge ja Masataka 2016) mukaan autistiset lapset vaikuttavat pitävän vihreästä väristä. Keltaista ja ruskeaa tulisi välttää. Edellä mainitun tutkimuksen otoskoko oli kuitenkin pieni, joten tuloksia ei voi yleistää. Grandgeorgen ja Masatakan mukaan lapset pitävät erityisesti pääväreistä. // TODO: Lihota

4 Kommunikointisovelluksen suunnittelu ja toteutus

Tutkielmaa varten toteutettiin FreeAAC-niminen avusteisen kommunikaation ohjelmisto. FreeAAC-ohjelmiston ominaisuusvaatimukset johdettiin edellisissä kappaleissa mainittujen ominaisuuksien perusteella:

// TODO: Tee tästä listasta hyvännäköinen

1. *Edellisissä kappaleissa lainattujen tutkimusten perusteella kommunikointisovelluksen tulisi olla korttipohjainen.* (Luku 3.1, s. 4)
2. *Korttien tulisi koostua kuvista ja symboleista.* (Luku 3.1, s. 4)
3. *Kuvia tulisi olla mahdollista valita valmiista kuvapankista.* (Luku 3.4.1, s. 15)
4. *Kuvia tulisi olla mahdollista ottaa itse laitteen kameralla.* (Luku 3.4.1, s. 16)
5. *Korttien kokoa tulisi olla mahdollista säätää käyttäjäkohtaisesti.* (Luku 3.4.1, s. 16)
6. *Kommunikaationäkymässä pitäisi olla mahdollisuus valita eri kortteja nopeasti ja helposti kontekstiin sopivaksi.* (Luku 3.4.1, s. 16)
7. *Ohjelmiston pitäisi pystyä lukemaan symboleilla ja kuvilla kirjoitettu viesti ääneen.* (Luku 3.4.1, s. 16)
8. *Kortteja ja muita asetuksia pitäisi päästä hallitsemaan hallintapaneelistä. Hallintapaneelin tulisi olla salasanaalla lukittava, jotta erityisryhmään kuuluva käyttäjä ei vahingossa säätäisi korttien tai muiden ohjelmiston osien asetuksia väärinlaisiksi.* (Luku 3.4.1, s. 16)

4.1 Kehitysympäristö ja -työkalut

Ionic-pohjaisten ohjelmistojen toteuttamiseen voidaan teoriassa käyttää melkein mitä tahansa tekstin kirjoittamiseen ja muokkaamiseen sopivaa työkalua sekä komentorivipohjaisia kääntäjiä. Käytännössä kuitenkin Ionic-pohjaisen ohjelmiston kehitystyötä helpottaa oikeiden työkalujen valinta. Tässä kappaleessa luetellaan FreeAAC-ohjelmiston kehitystyössä

oleelliset työkalut sekä kehitysympäristön toimivuuteen vaikuttavat taustaohjelmat.

Ionic CLI (Ionic Command Line Interface, *suom. Ionic-komentorivikehoite*) on Ionicin tarjoama konsolisovellus, jolla voidaan nopeasti generoida, asentaa ja räätälöidä Ionic-sovellukseen liittyviä tiedostoja.

Ionic-sovellusta voidaan ajaa paikallisena testiversiona neljällä eri tapaa: selaimessa, iOS- tai Android-simulaattorilla, mobiililaitteen selaimessa tai itsenäisenä sovelluksena puhelimessa. FreeAAC-ohjelmistoa testattiin pääosin selainversiona. Alustariippumattomuutta on osana mahdollistamassa Node.js -ympäristö.

Node.js on alustariippumaton runtime-ympäristö palvelinpuolen JavaScript-koodin suorittamiseen. Lähes kaikki JavaScript-pohjaiset kirjastot nojaavat Node.js:ään ja myös Ionic vaatii Node.js:n asennuksen. Node.js Package Manager eli **npm** on paketinhallintajärjestelmä JavaScript-kirjastoille. FreeAAC-ohjelmiston tarvitsemat ulkopuoliset paketit asennettiin npm:ää käyttäen.

FreeAAC-ohjelmiston kehittämiseen käytettiin **Visual Studio Code** -nimistä ohjelmistoympäristöä. Visual Studio Code on Microsoftin kehittämä ja ylläpitämä avoimen lähdekoodin Electron-pohjainen ohjelmistoympäristö. Angularin käyttämä TypeScript-ohjelmointikieli on myös Microsoftin kehittämä, joten Visual Studio Coden TypeScript-tukeen on panostettu. Visual Studio Coden tukena käytettiin myös liitännäisiä kuten ESLint-analyysityökalua.

ESLint on koodianalyysityökalu, joka käy läpi ohjelmistoympäristössä olevaa ohjelmakoodia ja auttaa ohjelmoijaa ohjelmoimaan koodianalyysityökaluun valittujen asetusten mukaisesti. ESLint voidaan helposti integroida Visual Studio Codeen laajennoksena, jolloin ohjelmistokehittäjä näkee koodianalyysityökalun tuottamat huomautukset reaaliaikaisesti.

4.2 Rakenne

FreeAAC noudattaa Ionic-sovelluksien oletusrakennetta. Ionic-sovelluksissa kansiorakenteen ylimmälle tasolle sijoitetaan projektin asetuksia sisältävät tiedostot. Projektin asetustiedostoista ohjelmistokehitysprosessin kannalta tärkein on `package.json`. Tiedostossa listataan kaikki ohjelmiston käyttämät ulkopuoliset kirjastot, ja Node.js:n paketinhallinta-

järjestelmä npm lataa sen perusteella oikeat versiot jokaisesta kirjastosta Ionic-sovelluksen käytettäväksi.

Seuraavalla kansiotasolla Ionicin oletusrakenteessa projekti jakautuu kahteen pääkansioon: `Resources` ja `Src`. Ohjelmiston pikakuvakkeiden ja käynnistysruudun tarvitsemat media-tiedostot ovat `Resources`-kansiossa.

`Src`-kansiossa sijaitsevat ohjelmiston lähdekoodia sisältävät tiedostot. `Src`-kansion päällä sijaitsevat `index.html`-, `manifest.json`- ja `service-worker.js`-tiedostot. `Index.html` on ensimmäinen HTML-tiedosto, jonka käynnistytvä Ionic-sovellus lukee. Se sisältää Ionic-sovelluksen juurikomponentin ja linkkejä sovelluksen yleisesti vaatimiin resursseihin kuten tyylitiedostoihin.

`Src`-kansiota alemmalla tasolla on Ionic-sovelluksen oletusrakenteessa viisi eri kansiota: `Assets`, `Classes`, `Pages`, `Providers` ja `Theme`. **// TODO: Lihota**

`Assets`-kansioon ohjelmiston kehittäjä voi sijoittaa kuva- ja äänitiedostoja, joita ohjelman toiminnallisuudet vaativat. FreeAAC-sovelluksen kuvakirjasto sijaitsee kansiossa.

`Classes`-kansiossa sijaitsevat ohjelmiston käyttämät luokat. FreeAAC-ohjelmistossa kaksi käytössä olevaa luokkaa ovat `Card` ja `WordSymbol`. `Card`-luokka vastaa kommunikointisovelluksissa käytettäviä kortteja. Se sisältää tiedon kortin nimestä, sen koosta ja kortin sisältämistä kuvista ja symboleista. `WordSymbol`-luokka taas on representaatio korteilla olevien kuvien ja symbolien tietomallista. Se sisältää kuvan tai symbolin nimen sekä viittauksen kuvatiedostoon.

Ohjelmiston näkymät sijaitsevat `Pages`-kansiossa. FreeAAC-ohjelmistossa on seitsemän eri näkymää: `Home`, `Main`, `CardCreate`, `CardDelete`, `Info`, `Options` ja `SelectSymbolModal`. Angular-ohjelmistokehystä käyttävässä Ionic-sovelluksessa näkymät ovat HTML-tiedostoja, joihin on **sidottu** (engl. *bind*) **komponenttiedostossa** olevia muuttujia ja aliohjelmia. Lisäksi kansioon sijoitetaan yleensä **moduulitiedosto**, jossa sijaitsevat viittaukset tarvittaviin ulkopuolisiin kirjastoihin sekä muihin moduuleihin ja tyylitiedosto, jolla ohjelmiston ulkoasu voidaan säätää näkymäkohtaisesti.

`Theme`-kansiossa sijaitsevat ohjelmiston ulkoasuun vaikuttavat SCSS-tiedostot. Ionicin ole-

tusprojektissa se sisältää `variables.scss`-tiedoston, joka asettaa projektille Ionicin oletusteeman. Kehittäjä voi itse vapaasti säätää ohjelmiston päätason ulkoasua tätä kautta.

`Providers`-kansioon sijoitetaan ohjelmiston palveluluokat. **Palvelumalli** (engl. *service pattern*) on suunnittelumalli, jossa useiden eri luokkien käyttämiä palveluita sijoitetaan samaan luokkaan yhdeksi palveluksi. Näin keskeisten palveluiden toteutus voidaan irroittaa itse luokkien toteutuksesta. Angularissa palveluluokat tuodaan luokkien käytettäväksi riippuvuusinjektiolla.

Riippuvuusinjektio (engl. *dependency injection*) on suunnittelumalli, jossa luokalle annetaan toinen luokka tai staattinen aliohjelma, joka tarjoaa luokan käyttöön uusia ominaisuuksia. Riippuvuusinjektiossa luokka tai staattinen aliohjelma annetaan toiselle luokalle sen sijaan, että luokka itse hakisi luokan tai staattisen aliohjelman. Angular-ohjelmistokehyksessä riippuvuusinjektio toteutetaan tuomalla toinen luokka `import`-lauseella mukaan luokan lähdekoodiin ja julistamalla riippuvuusinjektio luokan konstruktorissa.

FreeAAC-ohjelmiston tiedostojen tekstirivien prosenttiosuudet FreeAAC-ohjelmistossa ovat: TypeScript 57,6 %, HTML 28,0 %, CSS 11,7 % ja JavaScript 2,7 %.

FreeAAC-ohjelmistossa ylivoimaisesti eniten rivejä sisältävät `package.json`- ja `package-lock.json`-tiedostot. `Package.json` on JSON-tiedosto, joka sisältää NodeJS-alustan automaattisesti generoiman kuvauksen ohjelmiston perustiedoista ja riippuvuuksista npm-paketteihin versioineen. `package-lock.json`-tiedosto taas käytännössä pyrkii varmistamaan, että jokaisesta npm-paketista on käytössä oikea versio ja paketit ovat ehjiä. Sen tarkoitus on varmistaa, että jokainen asennettu ohjelmisto on mahdollisimman samankaltainen ja toimiva. Tämä edesauttaa alustariippumattomuutta.

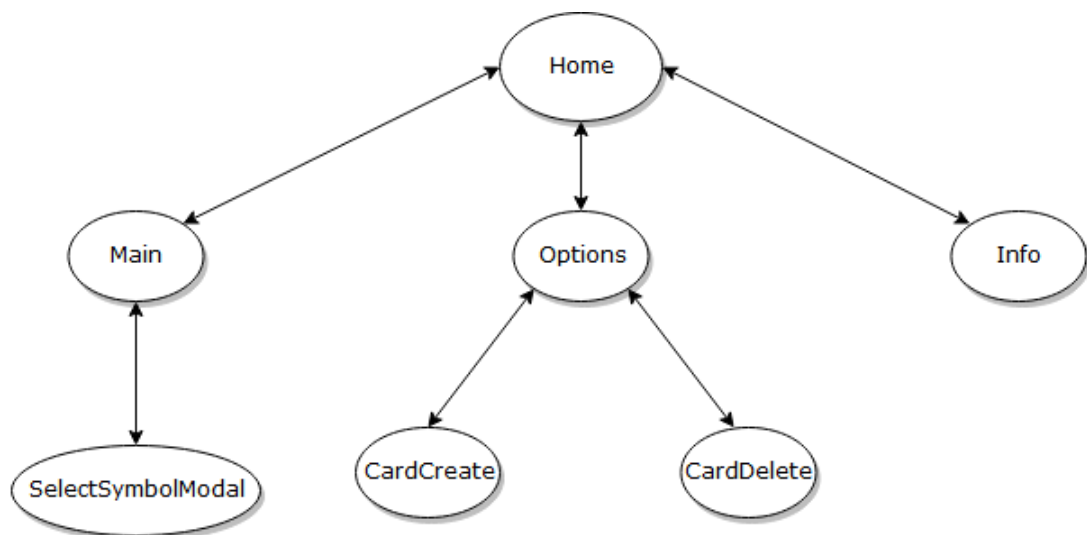
Toinen merkittävän määrän rivejä sisältävä generoitu tiedosto on `config.xml`, joka on Apache Cordovan käyttämä koko ohjelmiston laajuinen asetustiedosto. Asetustiedostossa on lähinnä Android- ja iOS-käyttöjärjestelmissä ajettavien ohjelmistoversioiden räätälöimiseen tähtääviä asetuksia.

Ohjelmiston ainoa JavaScript-tiedosto on `service-worker.js`. Se on oleellinen osa progressiivisen WWW-sovelluksen toimintaa, sillä se mahdollistaa toimintoja, jotka ovat ai-

emmin olleet mahdollisia vain natiiveille sovelluksille. `service-worker.js` on itsenäinen komentosarja, joka voi suorittaa toimintoja, joihin ei tarvita käyttäjän aktiivista osallistumista. Tällaisia ovat muunmuassa Internetiin yhteydessä olevasta sovelluksesta palvelimelle lähtevien kutsujen perille menemisen varmistaminen ja push-ilmoitusten näyttäminen.

4.3 Ohjelmiston näkymät

Ionicin näkymiä käsitellään pinona. Ionic-sovelluksessa näkymästä toiseen siirrytään lisäämällä uusi näkymä pinon päällimmäiseksi käyttämällä Ionicin `NavController`-kontrolleriluokkaa. Näin näkymäsiirtymien historia säilyy ja käyttäjä voi palata edellisiin näkymiin helposti painamalla mobiililaitteen edellinen-painiketta. Tällöin viimeksi pinoon lisätty näkymä poistetaan pinosta.

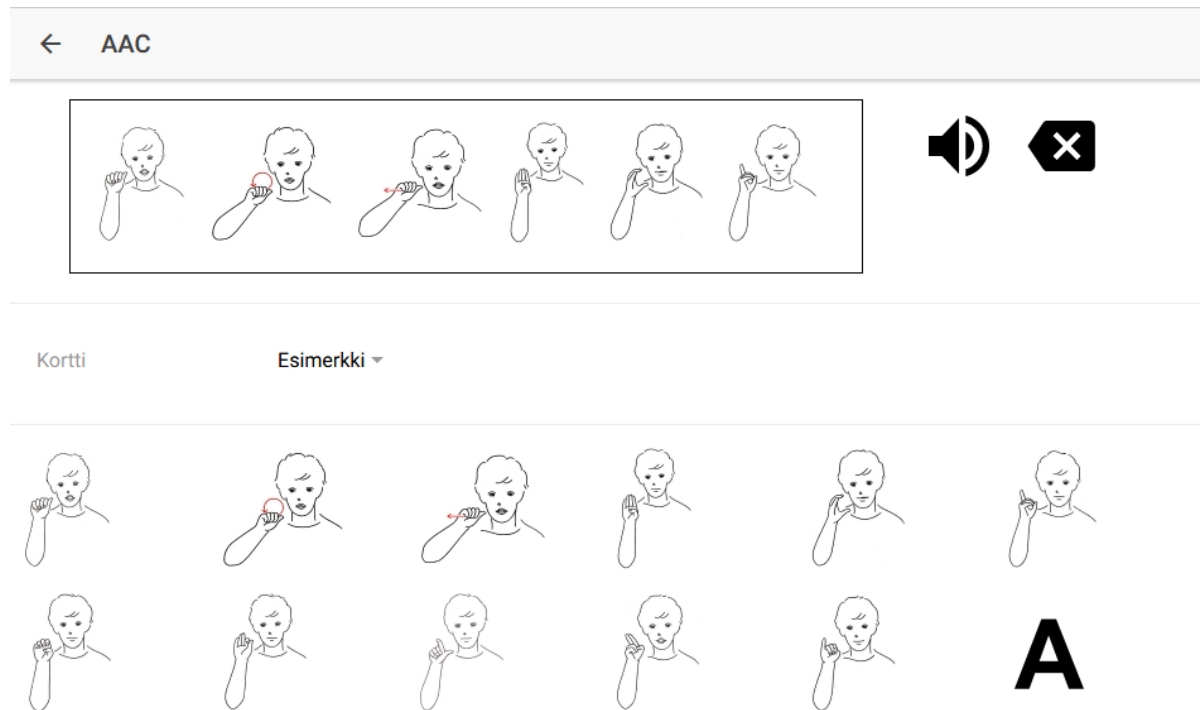


Kuvio 2. FreeAAC-ohjelmiston näkymäpuu.

FreeAAC:n Ionic-näkymissä sisältö on jaettu kahden eri päätason HTML-elementin sisälle: `ion-header` ja `ion-content`. `Ion-header`-lohkossa sijaitsevat näkymän ylälaitaan tuleva teksti, joka kertoo missä näkymässä ollaan tällä hetkellä. Lisäksi lohkoon voidaan sijoittaa valikkotoimintoja. `Ion-content`-lohkossa sijaitsee näkymän varsinainen pääsisältö.

Home-näkymä on FreeAAC-ohjelmiston ensimmäinen näkymä ja se sisältää ohjelmiston päävalikon. Päävalikossa sijaitsevat painikkeet, joita klikkaamalla käyttäjä pääsee keskuste-

lunäkymään (Main), asetusnäkyymään (Options) tai infonäkymään (Info). Home-näkymän komponenttitiedostossa ei ole muita kuin näkymän vaihtamiseen liittyviä aliohjelmia.



Kuvio 3. FreeAAC-ohjelmiston kommunikointinäkyymä.

// TODO: Yhteinen tapa käsitellä näkymien nimiä

Ohjelmiston tärkein näkyymä on kommunikointinäkyymä eli Main-niminen näkyymä, jossa sijaitsevat ohjelmiston varsinaiset kommunikointityökalut. Main-näkyymä on kuvassa 3. Main-näkymän komponenttitiedosto sisältää viestilaatikon päivittämiseen sekä korttien tietojen lataamiseen tarvittavat aliohjelmat. Korttien lataamiseen käytetään CardDataProvider-luokkaa, joka on injektoitu mukaan komponenttiin sen konstruktorissa.

Options-näkyymä on kokoomanäkyymä ohjelmiston hallintaa varten. Options-näkymän kautta käyttäjä pääsee siirtymään korttien luontinäkyymään (CardCreate) ja korttien poistonäkyymään (CardDelete).







Kortteja luodaan ohjelmiston CardCreate-näkyymässä. CardCreate-näkyymä on kuvassa 4. CardCreate-näkyymässä kortille voidaan valita nimi, koko ja liittää siihen kuvia ku-







←

Luo uusi kortti

Kortin nimi

Koko 12 ▾

TALLENNNA

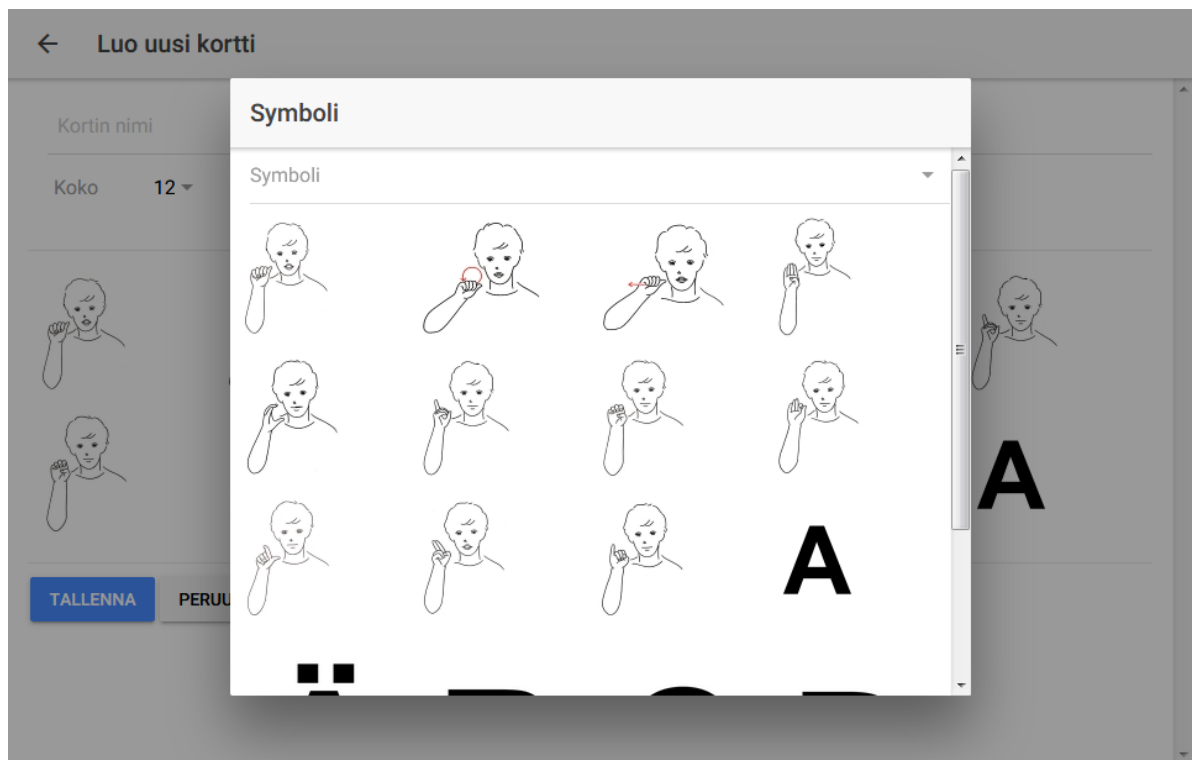
PERUUTA

Kuvio 4. FreeAAC-ohjelmiston korttien luontinäkö.

vakirjastosta. Kaikki edellä mainitut kentät ovat pakollisia ja käyttäjälle näytetään toast-virheilmoitus, jos jokin tieto puuttuu. Toast-virheilmoitukset käyttävät Ionicin `ToastController`-luokkaa. Kuvia valitaan erillisen ponnahdusikkunan kautta ja Ionic-sovelluksessa ponnahdusikkunoiden näyttämisestä vastaa `ModalController`-luokka. Korttien tallentaminen tapahtuu `CardDataProvider`-luokan kautta.

Kuvien valintaa varten aukeava ponnahdusikkuna sisältää `SelectSymbolModal`-näytteen. `SelectSymbolModal`-näytteen sisältävä ponnahdusikkuna on kuvassa 5. Näytössä kuvan voi valita eri kategorioista. Kattegoria valitaan pudotusvalikosta. `SelectSymbolModal`-näytteen komponenttiedosto sisältää kuvan tallentamiseen ja ponnahdusikkunan sulkemiseen liittyvät aliohjelmat. FreeAAC-ohjelmiston kuvakirjasto koostuu Papunet-sivuston tarjoamasta Creative Commons -lisenssin alaisesta kuvapankista.

Kortteja voidaan poistaa erillisessä `CardDelete`-näytössä. `CardDelete`-näytö koostuu pudotusvalikosta, josta voidaan valita mikä kortti halutaan poistaa sekä vahvistuspainikkeesta. Korttilistaus ladataan `CardDataProvider`-luokan avulla ja myös poistokomento kulkee samaa reittiä pitkin.



Kuvio 5. FreeAAC-ohjelmiston korttisyöbolien valintänäkömä.

Info-näkymässä listataan FreeAAC-sovellukseen liittyviä tietoja. Info-näkymä on staattinen eikä siinä ole toiminallisuuksia.

FreeAAC:lla on kaksi palveluluokkaa: `CardDataProvider` ja `ImageDataProvider`. `CardDataProvider`-olion tehtävä on tarjota näkömäkomponenteille tietoa olemassa olevista korteista. `ImageDataProvider`-olio huolehtii ohjelmiston kuvakirjaston eri kuvien toimittamisesta näkömäkomponenteille.

Ionic tarjoaa ohjelmistokehittäjän käyttöön `Ionic Storage` -nimisen tallennuspalvelun. `Ionic Storage` valitsee tallennustilan dynaamisesti ajoympäristöstä riippuen. Natiivissa kontekstissa `Ionic Storage` suosii `SQLite`-tietokantaa, kun taas WWW-sovelluksessa tai progressiivisena WWW-sovelluksessa ensisijainen tallennustila on `IndexedDB`-tietokanta. Jos `IndexedDB`:n käyttö ei jostain syystä ole mahdollista, yrittää `Ionic Storage` seuraavaksi käyttää `WebSQL`-tietokantaa tai selaimille varattua `LocalStorage`-tallennustilaa.

`Ionic Storage`en monimutkaisuus ja dynaamisuus heijastelee hyvin progressiivisten WWW-

sovellusten tallennusongelmia. Jokaisella edellä mainituista tallennusmahdollisuuksista on omat vahvuutensa ja heikkoutensa. Tämän takia tiedon tallentaminen massamuistiin jätettiin FreeAAC-ohjelmiston toteutuksen ulkopuolelle.

4.4 Käytettävyyšnäkökulma

// TODO: Pura tämä luku ja liitä näkymistä kertovaan osuuteen ja arvioivaan osuuteen

FreeAAC-ohjelmiston käytettävyyksivaatimukset johdettiin yleisten suositusten perusteella sekä edellisissä kappaleissa mainittujen ominaisuuksien perusteella.

1. *FreeAAC-ohjelmiston tulisi noudattaa yleisiä käytettävyyksivaatimuksia, mutta myös sen lisäksi ottaa huomioon erityisryhmien, erityisesti autististen käyttäjien, käytettävyyksivaatimukset huomioon.* (Luku 3.4.1, s. 15)

2. *FreeAAC-ohjelmiston värityksen tulisi olla pääväriihin perustuva, selkeä ja sen näkymissä tulisi olla riittävästi kontrastia.* (Luku 3.4.1, s. 17)

3. *Abstraktien symbolien käyttämistä tulisi välttää, sillä autistiset käyttäjät eivät välttämättä ymmärrä niiden merkitystä samalla tavalla kuin erityisryhmiin kuulumattomat käyttäjät.* (Luku 3.4.1, s. 17)

4. *FreeAAC-ohjelmiston käytettävyyttä arkielämän nopeaa reagointia vaativissa tilanteissa lisäisi mahdollisimman perinpohjainen mahdollisuus räätälöintiin.* (Luku 3.4.1, s. 15)

5. *Käyttäjien mahdollisten motoristen haasteiden vuoksi näkymien tulisi olla pääosin staattisia ja niissä ei saisi olla nopeaa reagointia vaativia tapahtumia.* (Luku 3.4.1, s. 15)

6. *Käyttäjällä tulisi olla mahdollisuus asemoida elementtejä vapaasti ruudulla.* (Luku 3.4.1, s. 15)

FreeAAC-sovellus käyttää Ionicin oletuselementtejä ja oletusteemaa. Ionicin mukaan oletuselementit ja oletusteema noudattavat Applen iOS-designperiaatteita sekä Googlen Material Design -määrittäjiä.

Valikkonäkymien lisäksi käytettävyyksratkaisuja on tehty erityisesti kommunikaationäkymäs-

sä (Main) ja korttien asetusnäkymissä (CardCreate ja CardDelete). Kommunikaatio-näkymä koostuu kolmesta eri komponentista: viestilaatikko, korttivalinta ja symbolivalinta. Viestilaatikkossa näkyvät käyttäjän valitsevat symbolit järjestyksessä. Käyttäjä voi poistaa symboleita yksi kerrallaan viimeisestä symbolista alkaen. Symbolikortti valitaan pudotus-valikosta, josta löytyvät käyttäjän kortinluonti-näkymässä tekemät kortit. Sivun alalaidassa näkyvät kortin symbolit, joista käyttäjä voi valita haluamansa symbolit viestilaatikkoon.

5 Toteutuneen sovelluksen arviointi

// TODO: Alusta tämä luku paremmin

Rivimäärien käyttäminen ohjelmiston arviointiin ei ole täysin ongelmaton, mutta eri tiedostojen rivimääriä analysoimalla voidaan saada perustason tietämystä ohjelmiston rakenteesta ja oleellisista osista. Edes ohjelmakoodin rivin määrite ei ole täysin yksiselitteinen asia, sillä yksi ohjelmakoodin looginen osa voidaan rivittää eri ohjelmointiperiaatteiden perusteella useille riveille. Lisäksi ohjelmakoodin rivimäärään vaikuttavat muutkin luettavuusmääritteet kuten tyhjien rivien käyttö sekä kommenttien määrä. FreeAAC-ohjelmiston rivien kokonaismäärä laskettiin ilman tyhjien tai kommenttirivien vähentämistä kokonaismäärästä.

Suurin osa riveistä sijaitsee ohjelmistokehittäjän itse laatimissa mallitiedostoissa ja näitä hallinnoivissa komponenttiedostoissa. Näissä tiedostoissa olevien rivien määrä on 783. Jos 7528 riviä sisältävä `package-lock.json`-tiedosto otetaan pois laskuista, on ohjelmiston kokonaisrivimäärä 1320 ja ohjelmistokehittäjän pääasiallisesti tuottamat tiedostot sisältävät noin 59,32% ohjelmiston kokonaisrivimäärästä.

Koska rivimäärien laskeminen ei anna kovin hyvää kuvaa ohjelmistosta, on ohjelmistojen analysoimiseen kehitetty edistyneempiä menetelmiä. Halsteadin monimutkaisuusmetriikat ovat tapoja mitata ohjelman ominaisuuksia. Maurice Howard Halstead pyrki niiden avulla muodostamaan tavan mitata ohjelman ominaisuuksia vertailukelpoisesti. Halsteadin monimutkaisuusmetriikoiden muodostamista varten tarvitaan neljä perusmuuttujaa: uniikkien operaattorien määrä (n_1), uniikkien operandien määrä (n_2), operaattorien kokonaismäärä (N_1) ja operandien kokonaismäärä (N_2).

Näiden neljän perusmuuttujan pohjalta voidaan johtaa useita eri monimutkaisuusmetriikoita:

Metriikka	Merkitys	Kaava
n	Sanasto	$n_1 + n_2$
N	Koko	$N_1 + N_2$
V	Määrä	$N * \log_2 n$
D	Haastavuus	$n_1/2 + N_2/n_2$
E	Työmäärä	$V * D$
B	Virheet	$V/3000$
T	Testausaika	E/k

Perusmuuttujien arvot FreeAAC-ohjelmistolle:

Tiedosto	n_1	n_2	N_1	N_2
MainPage	65	55	213	92
CardCreatePage	62	53	192	88
SelectSymbolModalPage	27	20	50	22
HomePage	26	16	56	31
OptionsPage	24	16	39	14
CardDataProvider	23	15	55	63
ImageDataProvider	22	14	52	16
CardDeletePage	21	17	42	20
Card	17	22	55	63
WordSymbol	16	14	47	16
InfoPage	13	10	26	10

Monimutkaisuusmetriikoiden arvot FreeAAC-ohjelmistolle:

Tiedosto	<i>n</i>	<i>N</i>	<i>V</i>	<i>D</i>	<i>E</i>	<i>B</i>	<i>T</i>
MainPage	120	305	1471.17...	54.36...	79978.03...	0.49...	4443s
CardCreatePage	115	280	1314.33...	51.47...	67651.01...	0.44...	3758s
SelectSymbolModalPage	47	72	277.73...	14.85...	4124.28...	0.09...	229s
HomePage	42	87	301.97...	25.19...	7605.86...	0.10	423s
Card	39	118	290.70...	24.34...	7075.83...	0.10...	393s
CardDeletePage	38	62	220.41...	12.35...	2722.75...	0.07...	151s
OptionsPage	38	62	220.41...	12.35...	2722.75...	0.07...	151s
CardDataProvider	38	118	288.64...	48.30...	13941.12...	0.10...	775s
ImageDataProvider	36	68	268.84...	12.57...	3379.65...	0.09...	188s
WordSymbol	30	63	230.62...	9.14...	2108.56...	0.08...	117s
InfoPage	23	36	117.61...	6.50...	764.48...	0.04...	42s

Sanastometriikan (*n*) perusteella FreeAAC-ohjelmiston kaksi sanastoltaan selkeästi laajinta tiedostoa ovat MainPage ja CardCreatePage. Muut tiedostot ovat laajuudeltaan suunnilleen samankokoisia keskenään. Ainoastaan InfoPage-tiedosto on selkeästi muita pienempi.

Kokometriikka (*N*) noudattaa pitkälti samoja linjoja, mutta Card- ja CardDataProvider-tiedostojen koko on suhteellisesti suurempi sanastometriikkaan verrattuna kuin muiden tiedostojen. Tämä kertoo näissä tiedostoissa olevien luokkien muista suuremmasta toisteisuudesta. Ne sisältävät useita pieniä metodeja, jotka ovat kytkeytyneet korttien käsittelyyn ja yksittäisen kortin tietojen tarjoamiseen muille luokille.

Testausyritys Verifysoftin mukaan (lainaus) tiedoston määrämetriikan (*V*) tulisi olla vähintään 100 ja korkeintaan 8000. Kaikki FreeAAC-ohjelmiston tiedostot pysyvät näiden rajojen sisällä. Tiedostojen keskinäisissä määrämetriikoissa ei ole mitään edellisistä metriikoista poikkeavaa.

Haastavuusmetriikka (*D*) pyrkii mittaamaan tiedoston virhealttiutta. Laajimpien MainPage- ja CardCreatePage-tiedostojen lisäksi korkea haastavuusmetriikka on myös CardDataProvider-tiedostolla. // TODO: Lihota

Tiedostojen työmäärät (E) noudattavat samoja linjoja kuin haastavuusmetriikka. Vaikka CardDataPro ei ole sanastoltaan tai kooltaan yhtä suuri kuin kaksi suurinta tiedostoa, sisältää se kuitenkin sellaisia rakenteita, jotka nostavat sen kuitenkin yhtä merkittäväksi osaksi ohjelmaa.

Virhemetriikka (B) on arvio tiedostossa esiintyvien ohjelmistovirheiden määrästä. Verifysoftin mukaan tiedoston virhemetriikan arvon tulisi olla alle 2. Jokainen FreeAAC-ohjelmiston tiedosto alittaa tämän arvon.

Viimeinen tässä pro gradu -työssä tarkasteltava metriikka on testausaika (T). Sillä pyritään arvioimaan tiedostoon testaamisen kuluvaa aikaa sekunneissa. Testausaikametriikka tuottaa muiden metriikoiden mukaisia arvoja. Testausajan laskemiseen käytetään arvoa k , joka määrittyy monen tekijän perusteella. Arvoon vaikuttavat asiat kuten testaajien kokemus ja tiedoston kriittisyys ohjelman toimivuuden kannalta, joten tämän pro gradu -työn puitteissa testausaikametriikasta ei juurikaan saada lisäarvoa FreeAAC-ohjelmiston arviointiin.

FreeAAC-ohjelmisto ei ylitä Halsteadin monimutkaisuusmetriikoiden ohjearvoja ja eri tiedostojen arvot eri metriikoiden kohdalla ovat johdonmukaisia keskenään. FreeAAC-ohjelmistotäten noudattaa Halsteadin monimutkaisuusmetriikoiden perusteella hyvän ja ehjän ohjelmiston rakennetta. Halsteadin monimutkaisuusmetriikoiden todistusvoima ei kuitenkaan ole aukoton ja parempia tuloksia saataisiin aikaan vertaamalla FreeAAC-ohjelmiston monimutkaisuusmetriikoiden arvoja muiden vastaavien ohjelmistojen arvoihin.

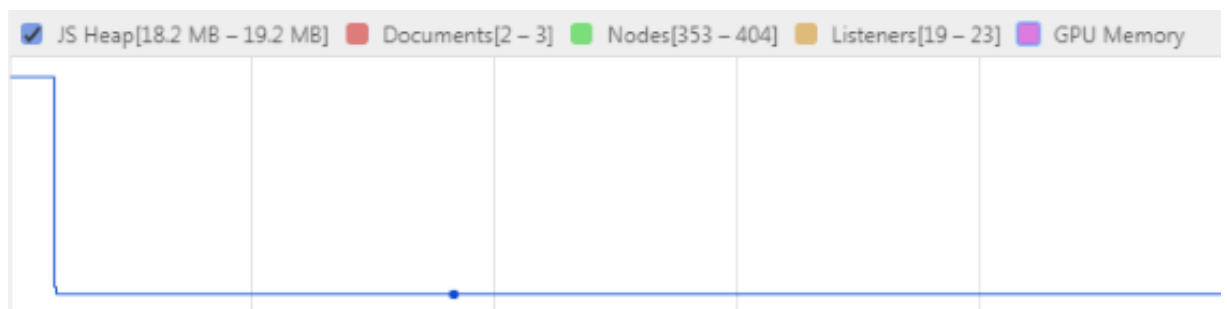
Vertaileva tutkimus muihin avusteisen kommunikaation sovelluksiin on kuitenkin rajattu tämän pro gradu -työn ulkopuolelle, sillä suurin osa avusteisen kommunikaation sovelluksista on kehitetty kaupalliseen käyttöön ja avoimen lähdekoodin sovelluksia ei juurikaan ole tarjolla. // TODO: prototyypiohjelmisto jne.

Progressiivisten WWW-sovellusten muistinkäyttöä arvioitiin kappaleessa 3.2. FreeAAC-ohjelmiston muistinkäyttöä mitattiin Firefox-selaimen kehitystyökaluilla ottamalla mittauksia sekä ohjelman alkutilassa 6 sekä käytön aikana 8. Lisäksi muistinkäyttöä seurattiin samalla tavoin Chromium-selaimen kehitystyökaluilla ohjelman alkutilassa 7 sekä käytön aikana 9.

Korkein mitattu muistinkäyttö rasiuksessa oli 37MB. Liitteissä C ja D on listattu joukko yleisten tablettitietokoneiden ja matkapuhelinten muistinkäyttörajoja ohjelmistoille. Ainoat

Bytes	Count	Total Bytes	Total Count	Group
3 789 168 22%	1 0%	3 789 168 22%	1 0%	#document
2 563 144 15%	42 327 22%	2 563 144 15%	42 327 22%	js:Shape
2 038 912 12%	29 830 16%	2 038 912 12%	29 830 16%	Function
1 911 536 11%	22 695 12%	1 911 536 11%	22 695 12%	Object
1 906 624 11%	49 143 26%	1 906 624 11%	49 143 26%	strings
1 547 728 9%	5 910 3%	1 113 552 6%	4 423 2%	JSScript
981 184 5%	9 550 5%	789 640 5%	8 343 4%	js:LazyScript
779 376 5%	14 986 8%	779 376 5%	14 986 8%	js:ObjectGroup
568 080 3%	5 796 3%	568 080 3%	5 796 3%	Array
560 144 3%	6 229 3%	560 144 3%	6 229 3%	js:Scope
208 096 1%	2 060 1%	208 096 1%	2 060 1%	Call
162 656 1%	221 0%	162 656 1%	221 0%	js:jit:JitCode
31 264 0%	977 1%	31 264 0%	977 1%	js:BaseShape

Kuvio 6. FreeAAC-sovelluksen muistinkäyttö lepotilassa.



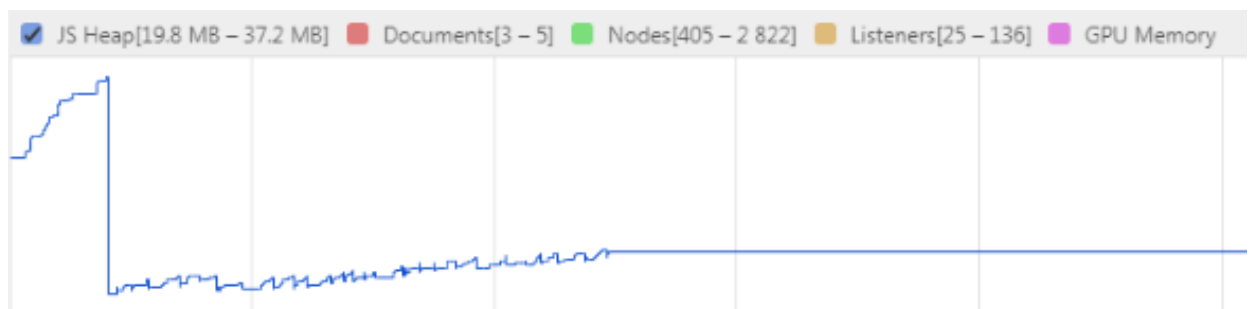
Kuvio 7. FreeAAC-sovelluksen muistinkäyttö lepotilassa.

listatut laitteet, joiden muistinkäyttörajan FreeAAC-ohjelmisto ylittää ovat HTC:n valmistamia matkapuhelimia vuosilta 2010 ja 2011. Muistinkäyttö nousee prosentuaalisesti varsin ison määrän käytön aikana, mutta kokonaismäärä ei aiheuta ongelmia millekään nykyaikaiselle mobiililaitteelle.

Muistinkäyttökäyriä tutkimalla nähdään, että ohjelman käynnistyessä muistinkäyttö nousee hetkellisesti, mutta ensimmäisen roskienkeruun jälkeen muistinkäyttö putoaa heti käynnistyksen jälkeistä tilannetta huomattavasti matalammaksi. Näin käy sekä ilman käyttäjän syötettä, että aktiivisen käytön aikana. Ilman käyttäjän syötteitä FreeAAC-ohjelmiston muistinkäyttö ei nouse ollenkaan. Ohjelmistossa ei ole toimintoja, jotka käyttäisivät muistia silloin kun käyttäjä ei aktiivisesti käytä ohjelmistoa. Tämä on tärkeää, sillä avusteisen kommunikation sovellusta käytetään useasti vain hetkellisesti. Käytön aikana muistinkäyttö nousee hietaasti ja 5 minuutin testikäytön aikana ei saatu aikaan suuria hyppäyksiä ylöspäin. FreeAAC-

Inspector Console Debugger Style Editor Performance Memory Network >>									
Record call stacks View: Aggregate Group by: Type Filter									
19.05.19 klo 22.05.31	X	Bytes	Count	Total Bytes	Total Count	Group			
17.18 MB	Save	7 841 616 29%	6 567 3%	5 788 352 21%	4 941 2%	JSScript			
		4 139 712 15%	10 0%	4 139 712 15%	10 0%	#document			
19.05.19 klo 22.12.15	X	2 704 232 10%	43 945 21%	2 704 232 10%	43 945 21%	js:Shape			
27.10 MB	Save	2 597 856 10%	30 696 14%	2 597 856 10%	30 696 14%	Object			
		2 116 928 8%	30 998 15%	2 116 928 8%	30 998 15%	Function			
		1 945 688 7%	50 176 24%	1 945 688 7%	50 176 24%	strings			
		1 643 256 6%	1 753 1%	1 643 256 6%	1 753 1%	js:jit:JitCode			
		890 832 3%	9 458 4%	783 296 3%	8 290 4%	js:LazyScript			
		880 144 3%	9 016 4%	880 144 3%	9 016 4%	Array			
		817 472 3%	15 704 7%	817 472 3%	15 704 7%	js:ObjectGroup			
		613 376 2%	6 894 3%	613 376 2%	6 894 3%	js:Scope			
		263 584 1%	2 743 1%	263 584 1%	2 743 1%	Call			

Kuvio 8. FreeAAC-sovelluksen muistinkäyttö käytön aikana.



Kuvio 9. FreeAAC-sovelluksen muistinkäyttö käytön aikana.

ohjelmiston muistinkäyttö vaikuttaa tasaiselta.

Angular-ohjelmistokehyksen sekä TypeScript-ohjelmointikielen avulla ohjelmistosta saatiin helposti tehtyä modulaarinen, minkä pitäisi parantaa ohjelman ylläpidettävyyttä ja helpottaa sen jatkokehitystä. Ylläpidettävyyttä ja jatkokehitettävyyttä ei kuitenkaan ole helppo mitata ja niiden arviointi ei tämän pro gradun -työn kannalta ole kovin mielekästä.

Näkymien HTML-formaatti ei juurikaan aiheuttanut ohjelmistoteknisiä haasteita tai käytettävyysongelmia. Ainoastaan symbolien vapaa asemointi kortilla osoittautui sen verran haastavaksi tehtäväksi, että sitä ei toteutettu. HTML-kieli on alunperin tarkoitettu dokumenttien hierarkkiseksi ja rakenteelliseksi kuvauskieleksi, joten HTML-elementtien kelluvaa asemointia toisiinsa nähden ei ole helpoin toteuttaa sen avulla. Kelluvien elementtien toteuttaminen on kyllä mahdollista oikeanlaisten CSS-määritysten avulla, mutta tämä vaatisi erillisen asemointipalvelun rakentamista ohjelmistoon sekä CSS-luokkien lisäämistä ja poista-

mista reaaliajassa. Lisäksi korttien asetusten tallentaminen muuttuisi huomattavasti haastavammaksi. Perinteisessä mobiilisovelluksessa elementtien vapaa asemointi olisi todennäköisesti helpompaa. // TODO: Lähteitä ja muista lisätä ominaisuusvaatimuksiin!

Kenties haastavin osuus FreeAAC-ohjelmiston toteuttamisessa oli tietojen tallentaminen. Kuten luvussa 4.3 todettiin, progressiivisille WWW-sovelluksille ei ole luontevaa tallennustilaa, sillä selainten oikeuksien määrä on hyvin tarkasti rajoitettua tietoturvasyistä. Kuvat voidaan liittää mukaan sovellukseen, mutta korttien asetusten tallentaminen ei ole yksinkertaista. Natiivisovelluksilla on suora pääsy rajattuun massamuistitilaan, mutta WWW-sovelluksissa tiedot tulee tallentaa muualle. Täten massamuistiin liittyvät operaatiot jätettiin tämän pro gradu -työn ulkopuolelle. // TODO: Korjaa hieman aiempaa kerrottua vastaavaksi.

5.1 Käytettävyyšnäkökulma

Käytettävyyden perusvaatimusten osalta FreeAAC-ohjelmiston toteutuksessa ei ilmennyt suuria haasteita. Ionicin tarjoama oletusteema on tehty yleisten käytettävyystvaatimusten mukaisesti käyttämällä erityisesti Googlen ja Applen mobiilikäyttöjärjestelmien käytettävyysohjeita, joten suurin osa ohjelmistosta pystyy käyttämään sitä suoraan ilman muutoksia. Tämä nopeuttaa ohjelmiston kehittämistä ja parantaa ohjelmiston ulkoasun käytettävyyttä sekä yhtenäisyyttä.

FreeAAC-sovelluksen elementit ovat riittävän isoja, jotta niiden valitseminen ja toiminnallisuuden ymmärtäminen on helpompaa kun ohjelmistoa ajetaan pieninäyttöisillä mobiililaitteilla. // TODO: Lähde käytettävyysohjeista? Suuret elementit myös rajoittavat yhteen näkymään mahtuvien ominaisuuksien määrää, mikä auttaa pitämään näkymät riittävän yksinkertaisina käyttäjälle.

FreeAAC-ohjelmiston opittavuutta helpottaa se, että näkymät muistuttavat paljolti toisiaan. FreeAAC-ohjelmistossa ei ole moodeja eikä samoja toimintoja voi suorittaa usealla eri tavalla. Tämä lisää sekä opittavuutta, että muistettavuutta.

Tehokkuus on määre, jota parhaiten voidaan mitata suorittamalla käyttäjätutkimus, jossa käyttäjälle annettuun tehtävään käytettyä aikaa sekä siinä ilmenneitä haasteita pyritään mit-

taamaan. Koska tutkielmassa ei suoritettu käyttäjätutkimusta vaan suunnittelutieteellinen tutkimus, on tehokkuutta tutkittava muiden keinojen avulla. Yksi tapa arvioida tehokkuutta on muodostaa lista yleisimpiin toimintoihin vaadittavien välivaiheiden määristä ja arvioida tehokkuutta niiden perusteella.

Toiminto	Välivaiheiden määrä
Kommunikointinäkymään siirtyminen	1
Viestin kirjoittaminen	$3 + (x * \text{symbolien määrä})$
Kortin luonti	$5 + (x * \text{symbolien määrä})$
Kortin poisto	3

Välivaiheiden määrästä ei voi suoraan muodostaa täyttä kuvaa ohjelmiston tehokkuudesta tai muista laadullisista tekijöistä, mutta niitä voidaan verrata psykologisiin tutkimuksiin ihmisten lyhytkestoisesta muistista. Yksi kaikkien aikojen eniten lainatuista psykologian tutkimuksista (Gorenflo ja McConnell 1991) on George A. Millerin *The Magical Number Seven, Plus or Minus Two* (Miller ym. 1956), jonka keskeiset havannot liittyvät ihmisen kykyyn muistaa tiettyjä määriä perusalkioita kerrallaan.

Millerin mukaan ihminen pystyy muistamaan 7 ± 2 perusalkiota lyhytkestoista muistia käyttämällä. Lisäksi Millerin mukaan ihminen käsittelee näitä alkioita kahden tai kolmen alkion ryppäissä. Yksikään FreeAAC-ohjelmiston perustoiminto ei ylitä tätä Millerin laiksikin kutsuttua perusoletusta. Täten FreeAAC-ohjelmiston perustoimintojen käyttö on ainakin teoreettisella tasolla ihmisen lyhytkestoisen muistin rajojen sisällä. Tämä parantaa ohjelmiston tehokkuutta, muistettavuutta ja vähentää käyttäjän tekemien virheiden määrää.

FreeAAC-ohjelmiston virhealttiutta vähentää myös se, että käyttäjän syötemahdollisuuksia on rajattu. Käyttäjä ei yleensä voi päätyä tilanteisiin, jossa tapahtuu jotain peruuttamaton. Ainoastaan symbolin poistaminen kortilta tai kortin poistaminen kokonaisuudessaan voi aiheuttaa käyttäjälle tilanteen, jossa menetetään paljon tehtyä työtä. Nämä ominaisuudet on kuitenkin piilotettu erilliseen salasanasuojattuun näkymään, joten riski peruuttamattomaan virheeseen on minimoitu.

Tyydyttävyyttä on hankala mitata ilman koekäyttäjiä, mutta yleisten laatutekijöiden arviointi ja huomioon ottaminen ohjelmiston toteutuksessa lisää tyydyttävyyttä. Voidaan esimerkiksi

olettaa, että ohjelmiston yhtenäinen ulkoasu lisää opittavuuden ja tehokkuuden lisäksi myös tyydyttävyyttä.

FreeAAC-ohjelmistolle listattiin myös joukko erityisryhmien käytettävyyksvaatimuksia. Etenkin autististen käyttäjien tarpeet pyrittiin ottamaan huomioon.

Sovelluksen käyttämiseen ei tarvita erityisryhmille mahdollisesti motorisesti haastavia eleitä kuten pyyhkäisyjä. Erityisesti autisteilla on usein ongelmia hienomotoristen liikkeiden suorittamisessa, vaikka autististen käyttäjien yksilökohtaiset erot ovat tässäkin suuria (Provost, Heimerl ja Lopez 2007).

FreeAAC-ohjelmiston valikot on toteutettu perinteisellä tavalla painikkeita hyödyntämällä. Monessa mobiilisovelluksessa navigointi tapahtuu niin sanotun *hampurilaisvalikon* avulla, mutta autistisille käyttäjille tarkoitetussa ohjelmistossa abstraktien symbolien käyttö voi olla ongelmallista, joten ohjelmistossa ei käytetä tätä yleistä ratkaisua.

FreeAAC:n ulkoasusta tehtiin mustavalkoinen, jotta symbolit ja käyttäjän mahdollisesti ottamat kuvat erottuvat paremmin muista elementeistä.

FreeAAC-ohjelmiston näkymissä ei ole toimintoja tai elementtejä, joiden käyttäminen vaatii nopeaa reagointia. Näkymistä on pyritty rakentamaan mahdollisimman yksinkertaisia ja tämän esimerkiksi korttien luontinäkymä ja korttien poistonäkymä on erotettu toisistaan.

// TODO: korttien vaihtaminen ja korttien koko

6 Pohdinta

Pro gradu -työn tärkein tulos oli se, että olemassa olevan tutkimuksen pohjalta luotujen vaatimusten noudattaminen ei tuottanut ongelmia. Progressiiviset WWW-sovellukset, ja tarkemmin rajattuna Ionic-kehys, sopii avusteisen kommunikaation sovelluksen kehittämiseen sekä ohjelmistoteknisestä, että käytettävyyšnäkökulmasta. Lisäksi saatiin kattavasti tietoa siitä, mitä laatutekijöitä avusteisen kommunikaation ohjelmiston kehittämiseen liittyy.

Suunnittelutieteellisen tutkimuksen käyttäminen progressiivisten WWW-sovellusten ohjelmistoteknisen puolen tutkimiseen oli toimiva tapa saada aikaan perustietoa tutkimuksessa esitetyn käyttötapauksen vaatimuksista ja niiden toteuttamisen haasteista. Progressiiviset WWW-sovellukset ovat edelleen niin tuore ilmiö, että olemassa olevaa tutkimusta oli haastavaa löytää. Tämän takia artefaktin kehittäminen oli tärkeää, sillä sen avulla saatiin paljon ensikäden perustietoa progressiivisten WWW-sovellusten toiminnasta ilman nojaamista olemassa olevaan tutkimukseen.

Käytettävyystudiumuksen näkökulmasta suunnittelutieteellisen tutkimusmetodin käyttäminen oli haastavampaa. Oli tärkeää ottaa huomioon käytettävyystekijät artefaktin suunnitteluvaiheessa, mutta varmempien johtopäätösten vetäminen valmiista artefaktista olisi kenties vaatinut käyttäjätutkimuksen tekemistä. Tästä syystä valmiin artefaktin tutkiminen käytettävyyšnäkökulmasta jäi ehkä hieman pintapuoliseksi.

Pro gradu -työn vertaaminen olemassa olevaan tutkimukseen on haastavaa myös siitä syystä, että hyvin harva avusteista kommunikaatiota tutkivista tutkimusta lähestyi asiaa tietoteknisestä näkökulmasta. Ainoa selkeästi olemassa olevaa avusteisen kommunikaation ohjelmistoa tutkinut tutkimus, joka tämän pro gradu -työn aineistonkeruussa onnistuttiin löytämään, oli Comunicador-nimistä espanjankielisille käyttäjille suunnattua ohjelmistoa tutkinut tutkimus, jonka tuloksia käsiteltiin luvussa 3.4.1.

– Pohdintaluvussa harjoitetaan itsenäistä ajattelua. Sen voi periaatteessa kirjoittaa ilman lähdekirjallisu-

7 Yhteenveto

FreeAAC-ohjelmiston prototyyppi sisältää suurimman osan tarpeellisista ominaisuuksista ja siitä pystyttiin toteuttamaan toimiva sekä mahdolliseen koekäyttöön sopiva versio tutkielman aikarajoitteiden puitteissa. Tämän perusteella voidaan todeta, että ainakin perustasolla progressiiviset WWW-sovellukset vaikuttavat lupaavalta ja riittävän kypsältä teknologialta yksinkertaisten mobiilisovellusten toteuttamiseen.

Tämän pro gradu -työn pohjalta syntyi useita mahdollisia jatkotutkimuksen aiheita. Yksi FreeAAC-ohjelmiston toteutuksen ulkopuolelle jäänyt perusominaisuus oli massamuistitilan käyttö käyttäjän asetusten ja luotujen korttien pysyvään tallennukseen. Tämän ominaisuuden karsimista perusteltiin Ionic-ohjelmistokehyksen massamuistitallennuksen monimutkaisuudella ja vaihtoehtojen paljoudella. Voisi siis olla kannattavaa tutkia progressiivisten WWW-sovellusten massamuistitallennusvaihtoehtojen hyviä ja huonoja puolia erilaisissa tilanteissa.

Toinen tutkimuksen aikana esiin noussut puute oli eri erikoisryhmien parissa tehtyjen käytettävyydestutkimuksien vähyys. Erityisesti modernien mobiililaitteiden ja kosketusnäyttöjen käytettävyyttä ei ole juurikaan ehditty tutkimaan näiden erityisryhmien näkökulmasta. Esimerkiksi käytettävyyden laatutekijöitä arvioidessa jouduttiin käyttämään yleisiä laatutekijöitä. Ei tiedetä esimerkiksi miten autismi vaikuttaa eri laatutekijöiden arviointiin.

Progressiivisten WWW-sovellusten muistinkäyttö oli yksi tässä tutkimuksessa tarkasteltu asia. Muistinkäyttö ei osoittautunut ongelmaksi tässä nimenomaisessa käyttötapauksessa, mutta tutkimuksen aihepiiriin ei kuulunut suuria määriä muistia vaativien progressiivisten WWW-sovellusten tutkiminen. FreeAAC-ohjelmiston toiminnallisuudet eivät vaadi suuria määriä laskentatehoa tai massiivisia muistiallokaatioita, joten täyttä kuvaa Ionic-ohjelmistokehyksen tai progressiivisten WWW-sovellusten suorituskyvystä ei saatu.

Yleisesti ottaen oli yllättävää miten pieni määrä ongelmia FreeAAC-ohjelmiston kehitystyössä ilmeni ja miten helposti suurin osa laadullisista vaatimuksista täyttyi. FreeAAC-ohjelmiston kehittämistä on tarkoitus jatkaa myös tämän pro gradu -työn jälkeen ja toivottavasti jonain päivänä ohjelmistosta on iloa muussakin kuin tutkimuskohteena.

Lähteet

AltexSoft. 2018. “The Good and the Bad of Angular Development”. Viitattu 29. joulukuuta 2018. <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/>.

AppInstitute. 2017. “A Beginner’s Guide to Progressive Web Apps”. Viitattu 3. heinäkuuta 2018. <https://appinstitute.com/a-beginners-guide-to-progressive-web-apps/>.

Apple. 2019. “Human Interface Guidelines”. Viitattu 11. toukokuuta 2019. <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>.

Arora, Chandermani, ja Kevin Hennessy. 2018. *Angular 6 by Example*. 3. painos. Livery Place, 35 Livery Street, Birmingham, UK: Packt Publishing.

Belitsoft. 2019. “React Native vs Xamarin vs Ionic”. Viitattu 6. kesäkuuta 2019. <https://belitsoft.com/react-native-development/react-native-vs-xamarin-vs-ionic>.

Beukelman, D. R., ja P. Mirenda. 2013. *Augmentative and alternative communication: Supporting children and adults with complex communication needs*. 4. painos. Baltimore: Paul H. Brookes Publishing Co.

Bierman, Gavin, Martin Abadi ja Mads Torgersen. 2014. “Understanding TypeScript”. *Lecture Notes in Computer Science* 8586:257–281. doi:978-3-662-44202-9_11.

Cagigas, Sira E. Palazuelos, Marisa M. Dominguez Olalla ja Jose L. Martinez Sanchez. 2005. “Graphic Communicator with Optimum Message Access for Switch Users”. *Assistive Technology: From Virtuality to Reality : AAATE 2005*: 207–211.

Clarke, Michael, ja Ray Wilkinson. 2005. “The Usability of AAC in Conversation: A Case Study of Interaction between a Child using AAC and her Classmate”. *Assistive Technology: From Virtuality to Reality : AAATE 2005*: 3–7.

Divante. 2018. “PWA Design Challenges”. Viitattu 30. kesäkuuta 2018. <https://divante.co/blog/pwa-design-challenges/>.

Enderby, Pam, Simon Judge, Sarah Creer ja Alexandra John. 2013. “Beyond the anecdote: Examining the need for, and provision of, AAC in the United Kingdom”. *White Rose Research Online*.

Frankston, Bob. 2018. “Progressive Web Apps”. *IEEE Consumer Electronics Magazine* 16 (3): 106–107. doi:10.1109/MCE.2017.2776463.

Google. 2018. “Progressive Web Apps”. Viitattu 26. kesäkuuta 2018. <https://developers.google.com/web/progressive-web-apps/>.

———. 2019. “Material Design”. Viitattu 11. toukokuuta 2019. <https://material.io/design/>.

Gorenflo, Daniel W., ja James V. McConnell. 1991. “The Most Frequently Cited Journal Articles and Authors in Introductory Psychology Textbooks”. *Teaching of Psychology* 18 (1): 8–12. doi:10.1207/s15328023top1801_2.

Grandgeorge, Marine, ja Nobuo Masataka. 2016. “Atypical Color Preference in Children with Autism Spectrum Disorder”. *Frontiers in Psychology* 7. doi:10.3389/fpsyg.2016.01976.

Ionic. 2019. “Ionic Documentation”. Viitattu 4. tammikuuta 2019. <https://ionicframework.com/docs>.

Khan, Sehrish, Mutahira Naseem Tahir ja Arif Raza. 2014. “Usability issues for smartphone users with special needs — Autism”. *IEEE Xplore*. doi:10.1109/ICOSST.2013.6720615.

Kozleski, Elizabeth B. 1991. “Visual symbol acquisition by students with autism”. *Exceptionality: A Special Education Journal* 2 (4): 173–194. doi:10.1080/09362839109524782.

Maharry, Dan. 2013. *TypeScript Revealed*. 1. painos. New York City, NY: Apress.

- Microsoft. 2019a. “TypeScript Documentation - Classes”. Viitattu 17. tammikuuta 2019. <https://www.typescriptlang.org/docs/handbook/classes.html>.
- . 2019b. “TypeScript Documentation - Interfaces”. Viitattu 20. tammikuuta 2019. <https://www.typescriptlang.org/docs/handbook/interfaces.html>.
- . 2019c. “TypeScript Documentation - Modules”. Viitattu 17. tammikuuta 2019. <https://www.typescriptlang.org/docs/handbook/modules.html>.
- Miller, George A., Jochen Braun, Christof Koch ja Joel L. Davis. 1956. “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information”. *Psychological Review* 101 (2): 343–352. doi:10.1037/h0043158.
- National Autistic Society. 2019. “Sensory differences”. Viitattu 6. kesäkuuta 2019. <https://www.autism.org.uk/about/behaviour/sensory-world.aspx>.
- Nielsen, Jakob. 2012. “Usability 101: Introduction to Usability”. Viitattu 2. maaliskuuta 2019. <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>.
- Norman, Don. 2013. *The Design of Everyday Things*. 3. painos. New York City, NY: Basic Books.
- Nunes, Débora R. P. 2008. “AAC Interventions for Autism: a Research Summary”. *International Journal of Special Education* 23 (2): 17–26.
- O’Kelly, Rory. 2017. “Electron Memory Usage Compared to Other Cross-Platform Frameworks”. Viitattu 21. helmikuuta 2019. <http://roryok.com/blog/2017/08/electron-memory-usage-compared-to-other-cross-platform-frameworks/>.
- Pichon, Michaël. 2017. “Is Twitter Lite really that light for your battery life?” Viitattu 4. toukokuuta 2019. <https://greenspector.com/en/articles/2017-11-21-is-twitter-lite-really-that-lite-for-your-battery-life/>.
- Provost, Beth, Sandra Heimerl ja Brian R. Lopez. 2007. “Levels of Gross and Fine Motor Development in Young Children with Autism Spectrum Disorder”. *Physical Occupational Therapy In Pediatrics* 27 (3): 21–36. doi:10.1080/J006v27n03_03.

Schlosser, Ralf. 1999. "Comparative efficacy of interventions in augmentative and alternative communication". *Augmentative and Alternative Communication* 15:56–68. doi:<https://doi.org/10.1080/07434619912331278575>.

Shamsuddin, Nurul Afqah, Shahida Sulaiman, Sharifah Mashita Syed-Mohamad ja Kamal Zuhairi Zamli. 2012. "Improving learnability and understandability of a Web application using an action-based technique". *2011 Malaysian Conference in Software Engineering*. doi:10.1109/MySEC.2011.6140678.

Sigafoos, Jeff, ja Erik Drasgow. 2001. "Conditional Use of Aided and Unaided AAC: A Review and Clinical Case Demonstration". *Focus On Autism And Other Developmental Disabilities* 16 (3): 152–161.

The National Autistic Society. 2018. "Designing Autism-friendly Websites". Viitattu 28. kesäkuuta 2018. <https://www.autism.org.uk/professionals/others/website-design.aspx>.

Usability, Affordable. 2011. "Usability Goals: Memorability of a Website". Viitattu 16. maaliskuuta 2019. <http://www.affordableusability.com/usability/memorability.html>.

Waranashiwar, Juilee, ja Manda Ukey. 2018. "Ionic Framework with Angular for Hybrid App Development". *International Journal of New Technology and Research* 4 (5): 1–2.

Vaughn, Bobbie, ja Robert Horner. 1995. "Effects of Concrete Versus Verbal Choice Systems on Problem Behavior". *Augmentative and Alternative Communication* 11 (2): 89–92. doi:10.1080/07434619512331277179.

Liitteet

A FreeAAC-ohjelmiston ominaisuusvaatimukset

Lähdemateriaalista johdetut ominaisuusvaatimukset FreeAAC-ohjelmistolle:

Ominaisuus	Toteutettu
Kommunikaatiokortit.	Kyllä
Kuvien ja symbolien valitseminen kortilta viestiin.	Kyllä
Viestin äänisynteesi.	Ei
Korttien luominen ja poistaminen.	Kyllä
Korttien koon määrittäminen.	Kyllä
Kortin vaihtaminen.	Kyllä
Kuvien valitseminen korteille valmiista kirjastosta.	Kyllä
Kuvien valitseminen korteille kuvia ottamalla.	Ei
Hallintapaneeli.	Kyllä
Lukitusominaisuus hallintapaneeliin.	Ei
Korttien ja asetusten tallentaminen laitteen massamuistiin.	Ei

B FreeAAC-ohjelmiston tiedostot rivimäärittäin

Tiedosto	Rivien lukumäärä
package-lock.json	7528
cardcreate.ts	112
variables.scss	88
config.xml	86
app.module.ts	73
package.json	66
image-data.ts	53
main.ts	52
index.html	50
main.html	41
cardcreate.html	40
card.ts	39
.gitignore	36
app.component.ts	36
card-data.ts	36
app.scss	32
selectsymbolmodal.ts	32
service-worker.js	31
home.ts	29
carddelete.ts	28
selectsymbolmodal.html	28
tsconfig.json	28
options.ts	27
wordsymbol.ts	26
options.html	21

Tiedosto	Rivien lukumäärä
info.html	20
home.html	19
app.html	18
.editorconfig	16
info.ts	14
cardcreate.module.ts	13
carddelete.html	13
carddelete.module.ts	13
info.module.ts	13
options.module.ts	13
selectsymbolmodal.module.ts	13
manifest.json	12
tslint.json	13
README.md	8
ionic.config.json	8
main.ts	8
cardcreate.scss	3
carddelete.scss	3
home.scss	3
info.scss	3
options.scss	3
selectsymbolmodal.scss	3
main.scss	0
Yhteensä	8848

C Muistinkäyttörajat yksittäiselle ohjelmistolle iOS-laitteissa

Tablettitietokoneet lihavoitu.

Malli	Muistinkäyttöraja
iPad1	127MB
iPad2	275MB
iPad3	645MB
iPad4	585MB
iPad Mini	297MB
iPad Mini retina	696MB
iPad Air	697MB
iPad Air 2	1383MB
iPad Pro 9.7"	1395MB
iPad Pro 10.5"	3057MB
iPad Pro 12.9" (2015)	3058MB
iPad Pro 12.9" (2017)	3057MB
iPad Pro 11.0" (2018)	2858MB
iPad Pro 12.9" (2018)	4598MB
iPhone4	325MB
iPhone4s	286MB
iPhone5	645MB
iPhone5s	646MB
iPhone6	645MB
iPhone6+	645MB
iPhone6s	1396MB
iPhone6s+	1392MB
iPhoneSE	1395MB
iPhone7	1395MB
iPhone7+	2040MB
iPhone8	1364MB

iPhone X	1392MB
iPhone XS	2040MB
iPhone XS Max	2039MB
iPhone XR	1792MB

D Muistinkäyttörajat yksittäiselle ohjelmistolle Android-laitteissa

Tablettitietokoneet lihavoitu.

Malli	Muistinkäyttöraja
Samsung Galaxy Tab GT-P1000	48MB
Samsung Galaxy Tab 8.9 GT-P7300	64MB
Samsung Galaxy Tab 10.1 GT-P7500	64MB
Samsung Galaxy Tab 3 10.1 GT-P5200	96MB
Acer Iconia A500	48MB
Kindle Fire HD 7"	48MB
Asus Transformer Prime TF201	48MB
Nexus 10	192MB
HTC Wildfire	16MB
HTC Wildfire S	20MB
HTC Salsa	20MB
HTC Desire	32MB
HTC Desire S	32MB
Samsung Galaxy S GT-I9000	48MB
Samsung Galaxy R GT-I9103	64MB
Samsung Galaxy Y GT-S5360	64MB
Samsung Galaxy Note N7000	64MB
Samsung Galaxy S3 GT-I9300	64MB
Samsung Galaxy S4 GT-I9505	128MB
Google Galaxy Nexus	96MB
Google Nexus 4	192MB
Google Nexus 5 GT-I9300	192MB
Samsung Galaxy S6 SM-G920W8	256MB