Ville Pelkonen

Mukautuva ja esteetön käyttöliittymä useaa laitetyyppiä tukevalle sovellukselle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

29.4.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Ville Pelkonen Mukautuva ja esteetön käyttöliittymä useaa laitetyyppiä tukevalle sovellukselle 42 sivua 29.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Lead Developer Juha Pulliainen Lehtori Kari Aaltonen

Insinöörityön tavoitteena oli selvittää mukautuvan ja esteettömän käyttöliittymän ajankohtaiset suunnittelu- ja toteutusmenetelmät. Sen osana toteutettiin käyttöliittymä oppimissovellukseen Windows 10 -alustalle käyttäen Universal Windows Platform (UWP) -sovellusarkkitehtuuria. Lisäksi insinöörityössä tutkittiin adaptiivisen ja responsiivisen suunnittelun periaatteita, esteettömyyden merkitystä, esteettömyyttä koskevaa lainsäädäntöä ja esteettömyyden käytännön toteutusta tietoviestinnän tuotteissa. Insinöörityön tilasi suomalainen digitaalisiin oppimisratkaisuihin erikoistunut yritys. Varsinaisen sovelluksen tilaaja oli monikansallinen suuryritys.

Toteutetun sovelluksen tarkoitus oli toimia oppimisalustana tarjoamalla oppimateriaalia ja pelillistettyjä harjoituksia. Sovellusprojektin tavoitteisiin kuuluivat perustoimintojen ohella transitio Windows 8.1 -sovelluksesta, tarkkojen esteettömyyskriteerien täyttyminen, asiakkaan brändin mukainen ulkoasu, sujuva käytettävyys ja julkaisuvalmistelut Windows Store -jakelupalveluun. Lisäksi sovelluksen tuli toimia älypuhelimilla, tableteilla ja tietokoneilla. Esteettömyyteen kiinnitettiin erityistä huomiota, ja sen toteutumista mitattiin erilaisten ohjelmistojen ja asiakkaan oman tarkistusprosessin avulla. Sovellus luotiin käyttäen C#-ohjelmointikieltä ja XAML-merkintäkieltä.

Insinöörityön lopputuloksena syntyi usealla laitetyypillä toimiva Windows 10 -sovellus, joka täytti vaaditut esteettömyys- ja käytettävyyskriteerit. Sovellus julkaistaan maailmanlaajuiseen jakeluun Windows Storeen vuonna 2016. Projektin päätyttyä asiakasyritys osoitti tyytyväisyytensä ja hyväksyi projektille asetetut tavoitteet saavutetuksi.

Sovelluskehityksen ja käyttöliittymän toteutuksen aikana saatujen kokemusten perusteella esteettömyyden havaittiin olevan hyvä oletusarvo kaikille käyttöliittymille. Oikeudellisista, eettisistä ja taloudellisista syistä sitä ei tulisi laiminlyödä. Kuluttajateknologian monimuotoistuessa esteettömyys voi parantua entisestään, mutta vain jos esteettömyydelle nähdään markkina-arvoa tai sille osoitetaan riittäviä juridisia vaatimuksia. Myös useaa eri laitetyyppiä tukevat sovellukset ja palvelut tulevat yhä ajankohtaisemmiksi. Universal Windows Platform todettiin sovellusarkkitehtuurina erinomaiseksi alustaksi toteuttaa esteettömiä ja adaptiivisia sovelluksia, jotka toimivat usealla erilaisella laitteella. Mukautuvien käyttöliittymien suurimmat haasteet lienevät kuitenkin vielä edessä esimerkiksi virtuaalitodellisuuden, lisätyn todellisuuden ja pieninäyttöisten laitteiden yleistyessä.

Avainsanat	UWP, Windows 10, käyttöliittymä, esteettömyys, mukautuvuus

Author Title Number of Pages Date	Ville Pelkonen Adaptive and accessible user interface for device-agnostic software 42 pages 29 April 2016
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Juha Pulliainen, Lead Developer Kari Aaltonen, Principal Lecturer

The purpose of this thesis was to study modern methods for creating adaptive and accessible user interfaces. The thesis covers a study on the theory of adaptive and responsive user interfaces, the relevance of accessibility, regulations related to accessibility and practical methods for implementing accessibility in Windows 10 applications. The thesis was commissioned by a digital learning agency. The actual client for the project was an international enterprise.

The study consisted of developing a user interface for a Windows 10 learning application utilizing the Universal Windows Platform application architecture. The software is meant to work as a service for anyone to study and practice a defined subject. The application offers theory and exercises for this purpose. Apart from basic functionality, the goals for the application project were a transition from a previous Windows 8.1 application, implementation of a user interface following extensive accessibility standards, a visual style reflecting the client's brand, eloquent usability and preparations for Windows Store publishing. In addition, the application was to support smartphones, tablets, laptops and desktop PCs. Accessibility was tested using the client's own procedures. The application was written in C# and XAML.

An accessible, adaptive and device-agnostic Windows 10 learning application was developed as a part of the study. The application will have its global Windows Store release in 2016. The client was happy with the product and commended the work after the project was finished.

Accessibility was found to be a viable and necessary default for any user interface. It should never be neglected due to legal, ethical and financial reasons. As consumer technology continues to diversify, accessibility may improve further. However, this may require exerting pressure through financial and legal means. Applications adapting to different devices are also likely to proliferate. Universal Windows Platform proved to be an excellent applications architecture for developing accessible and adaptive software that run on various end devices. Greater challenges related to adaptive applications, however, are still in the future as virtual reality, augmented reality and small screen devices become more commonplace.

Keywords	UWP, Windows 10, user interface, accessibility, adaptive

Sisällys

Lyhenteet ja määritelmät

1 Johdanto 1

2 Mukautuvat käyttöliittymät 2

2.1 Adaptiivisuus ja responsiivisuus 2
2.2 Mukautuvuuden merkitys 4

3 Käyttöliittymien esteettömyys 6

3.1 Esteettömyyden merkitys 6
3.2 Lainsäädäntö 8
3.3 Esteettömän käyttöliittymän suunnittelu 9
3.4 Esteettömyyden parhaat käytännöt 10

4 Universal Windows Platform -sovellusarkkitehtuuri 13

4.1 Ominaisuudet 13
4.2 Kehitystyökalut 14
4.3 Sovellusarkkitehtuurin merkitys 15

5 Käyttöliittymän toteutus Windows 10 -sovelluksessa 17

5.1 Sovelluksen vaatimukset 17
5.2 Työtavat ja työkalut 19
5.3 Ratkaisut 20

6 Windows 10 -sovelluksen testaus ja julkaisu 29

6.1 Käyttöliittymän testaus 29
6.2 Windows Store -julkaisu 32

7 Yhteenveto 33

Lähteet 36

Lyhenteet ja määritelmät

C#	Microsoftin Windows-käyttöjärjestelmälle kehittämä tyypitetty olio-ohjelmointikieli.
JAWS	*Job Access With Speech*. Freedom Scientific -yrityksen tuottama, maailman suosituin ruudunlukuohjelma.
JSON	*JavaScript Object Notation*. Avoimen standardin tiedostomuoto tiedonvälitykseen ja -organisointiin.
MVVM	*Model-view-viewmodel.* Sovellusarkkitehtuurimalli, joka on kehitetty organisoimaan sovelluksen eri loogisia osia helpottamaan tapahtumakeskeistä käyttöliittymän ohjelmointia.
UWP	*Universal Windows Platform*. Windows 10- ja Windows 10 Mobile -käyttöjärjestelmiä varten tehty sovellusarkkitehtuuri, joka mahdollistaa usealla laitetyypillä toimivien sovellusten valmistamisen.
XAML	*Extensible Application Markup Language.* Microsoftin kehittämä merkintäkieli, jolla ohjelmoidaan Microsoft Silverlight 1.0-, Windows Presentation Foundation- (WPF) ja UWP-sovellusarkkitehtuurien esitystaso.

1 Johdanto

Insinöörityössä selvitetään, kuinka suunnitella ja toteuttaa esteetön ja päätelaitteen ominaisuuksiin mukautuva käyttöliittymä Windows 10 -sovellukseen. Tavoitteena on toteuttaa käyttöliittymä hyödyntäen Universal Windows Platform -sovellusarkkitehtuuria. Sovellus laaditaan vanhan Windows Phone 8.1 -sovelluksen pohjalta, ja työ laajentaa maailmanlaajuista oppimispalvelua, johon on myös verkkosovellus. Sovellukselle rakennetun käyttöliittymän tulee toimia usealla eri laitetyypillä modernien verkkosivujen tai -palveluiden tavoin. Työ tehdään digitaalisiin oppimisratkaisuihin erikoistuneen Bitville Oy:n tilauksesta asiakasprojektia varten. Asiakas ja sovelluksen tilaaja on monikansallinen suuryritys.

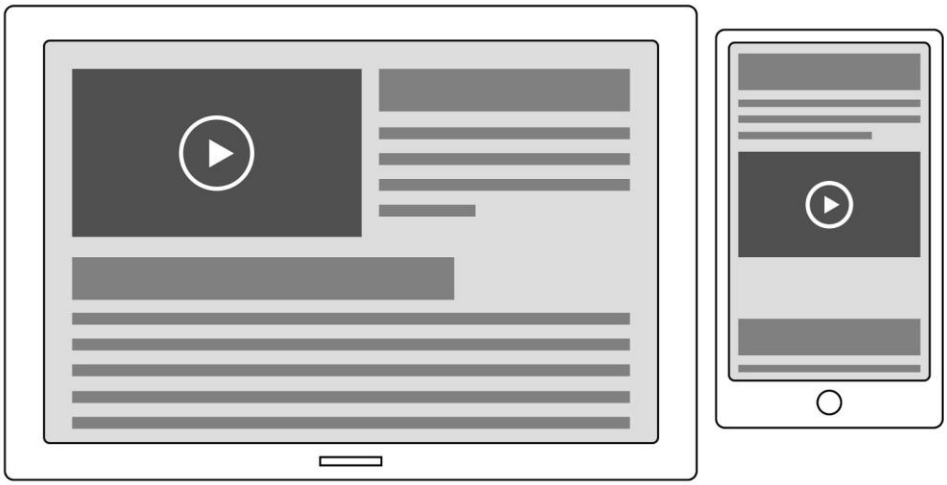
Yhä useammalla on taskussaan älypuhelin tai kotona kosketusnäytöllinen tablettitietokone. Markkinoille on tullut älykelloja, ajotietokoneita ja useita muita erityyppisen näytön omaavia laitteita. Erilaiset laitteet luovat uusia mahdollisuuksia hyödyntää teknologiaa työssä ja jokapäiväisessä elämässä, mutta se tuo sovelluskehittäjille uusia haasteita. On syntynyt tarve luoda sovelluksia, ja niille käyttöliittymiä, jotka toimivat usealla eri päätelaitteella. Ajatus laitteen mukautuvasta käyttöliittymästä on verkkoympäristössä asettunut jo osaksi jokapäiväistä käytettävyys- ja käyttöliittymäsuunnittelua. Sovelluskehityksessä suuntaus ei ole vielä yleistynyt laiteyhteensopivien sovellusten ja niitä tukevien käyttöjärjestelmien vähäisyyden vuoksi.

Osa digitaalisten sovellusten ja palveluiden käyttäjistä kärsii käyttöä hankaloittavista rajoitteista. Tämä voi johtua synnynnäisestä tai kehittyneestä vammaisuudesta tai vauriosta. Sovellusten käyttöliittymät ovat erityisesti hankalia näkö-, kuulo- ja kehitysvammaisille sekä hermoston häiriöistä kärsiville ihmisille [1]. Esteettömyydellä pyritään tekemään käyttöliittymästä lähestyttävä kaikille ja mahdollistamaan myös tavanomaisesta poikkeavat selaus- tai käyttötavat. Esteettömyyden avulla voidaan tuoda digitaaliset tuotteet ja palvelut lähemmäs ihmisiä, joilla on rajoitteita. Useat valtiot vaativat julkisia tahoja ottamaan esteettömyyden huomioon; joillakin on myös määräyksiä, jotka koskevat yksityisiä ohjelmistokehittäjiä. Esteettömyyden laiminlyöminen voi sulkea pois suuren osan käyttäjistä harventaen potentiaalisia markkinoita, joten esteettömyydellä on myös taloudellista merkitystä. Tarkkaa arviota esteettömyyttä tarvitsevien määrästä on kuitenkin vaikeaa tehdä, sillä valtioiden käsitykset vammaisuudesta eivät ole yhdenmukaisia. [2.]

2 Mukautuvat käyttöliittymät

2.1 Adaptiivisuus ja responsiivisuus

Mukautuvan käyttöliittymäsuunnittelun tarkoitus on mahdollistaa käyttöliittymän sopivuus erikokoisille ja -tyyppisille päätelaitteille [3]. Mukautuvan käyttöliittymäsuunnittelun kaksi suurta suuntausta ovat adaptiivinen ja responsiivinen suunnittelu. Adaptiivisuus ja responsiivisuus ovat tavoitteiltaan hyvin samankaltaisia. Adaptiivisuus perustuu ennalta määriteltyjen käyttöliittymän ominaisuuksien tarjoamiseen päätelaitteeseen sopivaksi, kun taas responsiivisuus pohjautuu yhden käyttöliittymän saumattomaan mukautuvuuteen päätelaitteesta riippumatta. [4.] Kuvassa 1 on havainnollistettu saman käyttöliittymän mukautuvaa asettelua tabletissa ja älypuhelimessa.



Kuva 1. Yksinkertainen esimerkki mukautuvasta käyttöliittymästä eri laitteilla.

Adaptiivinen käyttöliittymä on määritelty useaan versioon tai osaan, joista sopivimmat ovat käytössä päätelaitteesta riippuen. Versio tai ominaisuudet tarjotaan päätelaitteelle sopivaksi laitteen ominaisuuksien, kuten syöttömenetelmän tai näytön koon, perusteella. Käyttöliittymä voi mukautua rakenteeltaan, navigaatioltaan, vuorovaikutuksiltaan tai olla jopa täysin erilainen riippuen päätelaitteesta tai päätelaitteen ominaisuuksista. Adaptiivisuus on jatke progressive enhancement -ajattelulle, joka käsitti pitkälti samat periaatteet. [4.] Adaptiivisuus ei tässä tapauksessa kata muotoutumista käyttöliittymän tilaan.

Adaptiivisuuden täysi hyödyntäminen vaatii laitteen tai sen ominaisuuksien tunnistamisen, mikä ei kaikissa tapauksissa ole mahdollista. Käyttöjärjestelmäympäristössä tämä

on helpompaa, mutta verkkoympäristössä palvelin voi tehdä vain rajallisia arvioita päätelaitteesta käytettävän verkkoselaimen kautta. Adaptiivisuutta voidaan myös käyttää kattoterminä kaikille mukautuville käyttöliittymille, synonyyminä responsiivisuudelle tai osana responsiivisen suunnittelun määritelmää [5].

Adaptiivisuudesta poiketen responsiivisesta käyttöliittymästä on aina vain yksi useaan ympäristöön sopeutuva versio. Responsiivisuus kiteytyy universaalin ratkaisumallin tavoittelemiseen, käyttöliittymään, joka sopii kaikille päätelaitteille ja käyttäjille.

Responsiivinen suunnittelu perustuu joustavaan taulukkomaiseen grid-asetteluun, fluideihin kuviin ja mediakyselyihin [6, s. 9]. Grid koostuu useista laatikkomaisista elementeistä, jotka asettuvat ruudulle suhteessa toisiinsa. Joustavuus tekee elementeistä ympäristöönsä mukautuvat, ja elementtien koko tai sijainti voi vaihdella tilasta riippuen. Näin koko grid-pohjaisesti aseteltu käyttöliittymä asettuu sulavasti päätelaitteen näytön kokoon ja käyttöliittymälle annettuun tilaan. Fluidi kuva asettuu kooltaan sopivaksi sille tarjottuun tilaan suurentuen tai pienentyen. [6, s. 44–46.] Mediakyselyt toimivat rajapyykkimäisten breakpointien avulla. Näytön tai ikkunan mittoihin perustuvan breakpointin saavutettaessa käyttöliittymä mukautuu joustavaa asettelua monimuotoisemmalla tavalla. Esimerkiksi mobiililaitteen pienellä näytöllä voi valikon piilottaa napin painalluksen taakse tilan säästämiseksi. Mediakyselyt toimivat aina päätelaitteella eivätkä esimerkiksi palvelimella. [6, s. 74–79.] Mediakyselyt ovat verkkoselainympäristössä käytettävä Cascading Style Sheets (CSS) -teknologia, mutta sovelluskehityksessä on vastaavia ratkaisuja mukautuvia käyttöliittymiä varten.

Ajatus responsiivisuudesta syntyi verkkosivujen käyttöliittymäsuunnittelua käsittelevästä Ethan Marcotten vuonna 2010 A List Apart -verkkosivulla julkaisemasta artikkelista ja myöhemmin samannimisestä kirjasta, Responsive Web Design. Artikkelista noussut keskustelu synnytti rinnakkaisen ajatuksen adaptiivisista käyttöliittymistä erottaakseen kaksi eri keinoa tarjota laiteyhteensopivia käyttöliittymiä verkkoympäristössä. [7.] Termistö ei kuitenkaan ollut, eikä vieläkään ole, yksiselitteinen.

Adaptiivisuudesta ja responsiivisuudesta saatetaan puhua samana ilmiönä tai toistensa ala- tai yläkäsitteinä [5]. Eri käyttöliittymä- ja verkkosuunnitteluauktoriteetit, kuten Mozilla Developer Network (MDN) ja Microsoft Developer Network (MSDN) määrittelevät adaptiivisuuden ja responsiivisuuden hieman eri tavoin [4; 8]. Responsiivisuus on yleisemmin

käytetty termi, ja nykyään sillä useimmiten tarkoitetaan myös perinteisesti adaptiivisuuteen liittyviä konsepteja. Keskustelu termistöstä on kiivasta verkkoselaimille käyttöliittymiä suunnittelevien ja kehittävien piirissä, mutta sovelluskehittäjien parissa keskustelua ei juuri ole. Sille ei ole tarvetta, koska harva käyttöjärjestelmä on suunniteltu toimimaan usealla eri laitetyypillä ja sovellukset ohjelmoidaan aina käyttöjärjestelmäkohtaisesti. Näkemyserojen vuoksi tässä raportissa puhutaan mukautuvuudesta puhuttaessa päätelaitteille sopeutuvista käyttöliittymistä.

Verkkoympäristössä pelkästään asettelultaan responsiivisten käyttöliittymien ongelmaksi voi koitua ladattavan sisällön, kuten videoiden ja kuvien, määrä. Mobiililaitteilla prosessoinnissa voi kestää odotettua kauemmin ja langattomilla yhteyksillä latausajat voivat vaihdella. Toisaalta jos adaptiivisen käyttöliittymän tapauksessa käyttäjää vaaditaan lataamaan kaikki eri käyttöliittymän versiot päätelaitteesta riippumatta, yksi järkevästi toteutettu responsiivinen ratkaisu voi olla parempi vaihtoehto. [9.]

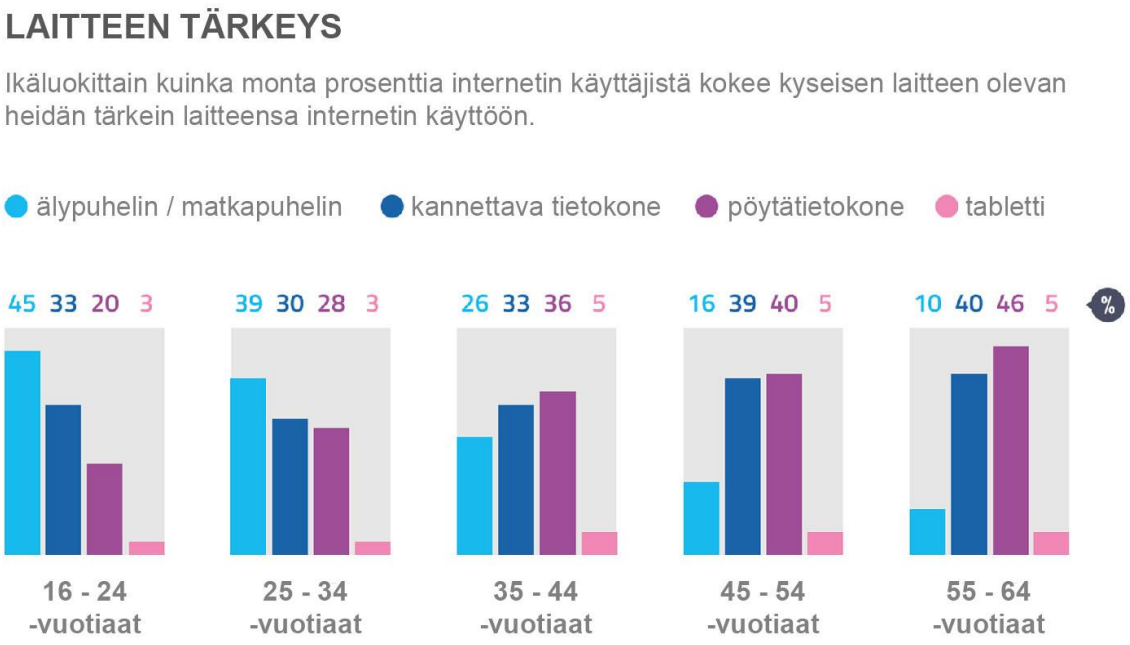
Pelkästään responsiivinen käyttöliittymä voi tarjota sisältöä tai toiminnallisuuksia, jotka eivät toimi optimaalisesti. Käyttöliittymä ei välttämättä huomioi päätelaitteen prosessointitehoa tai arvioi yhteyttä. Adaptiivinen käyttöliittymä voi jo palvelimella määrittää verkkoselaimelle tarjottavan version ja täten parantaa käyttäjäkokemusta jättäen pois soveltumattoman sisällön. Käyttöliittymää suunniteltaessa on tärkeää selvittää, mitkä mukautuvan käyttöliittymän keinot ovat käyttötarkoitukseen soveltuvimmat. Useimmissa tapauksissa responsiivinen suunnittelu ajaa asian ja adaptiivista suunnittelua voi käyttää korostamaan laitteiden erikoisominaisuuksia, jos on tarpeen.

2.2 Mukautuvuuden merkitys

Mukautuvan käyttöliittymän eduiksi lasketaan ylläpidon helppous, käytettävyys, pitkäikäisyys ja parempi näkyvyys. Kun käyttöliittymästä on vain yksi versio, joka on tehty toimimaan erikokoisilla näytöillä, vain yhtä tuotosta on tarve päivittää. Responsiivinen käyttöliittymä voi tukea myös uusia markkinoille tulevia laitteita niiden tarkkuudesta tai näytön koosta riippumatta. Tällöin päivityksen tarve rajautuu lähinnä visuaalisen ilmeen päivitykseen eikä tarvitse huolehtia käyttöliittymän yhteensopivuusongelmista. [10.]

Laaja laitetuki auttaa huomattavasti käyttäjien tavoittamisessa. Kun palvelusta tai verkkosivusta on olemassa vain yksi versio ja linkki, se parantaa hakukoneoptimointia.

Google on myös suosinut mobiiliystävällisiä, tarkoittaen muun muassa responsiivisia, verkkosivuja jo huhtikuusta 2015 [11]. Vuoden 2015 lopussa internetin käyttäjistä 87 % omisti älypuhelimen, lähes yhtä moni kuin pöytätietokoneen tai kannettavan tietokoneen [12]. Kuten kuvasta 2 näkyy, alle 35-vuotiaat internetin käyttäjät kokevat äly- tai matkapuhelimen olevan tärkein laite internetin selaamiseen. Käyttäjäkunnan maksimoimiseksi on olennaista, että mikä tahansa digitaalinen palvelu tai tuote, jolla on mahdollisuus toimia usealla eri alustalla, hyödyntää tätä voimavaraa.



Kuva 2. Internetin käyttäjien tärkeimmäksi koetut laitteet ikäluokittain vuoden 2015 viimeisellä vuosineljänneksellä [12].

Natiivisovelluksissa, eli sovelluksissa, jotka on tehty käyttöjärjestelmälle asennettavaksi, käyttöliittymän mukautuvuus ei ole niin suuri haaste kuin verkkoselaimelle tarjottavassa sisällössä. Koska käyttöjärjestelmät on usein valmistettu tietyntyyppisiä laitteita varten, natiivisovellusten alustoista voidaan tehdä tarkkoja oletuksia eivätkä oletetut päätelaitteet usein vaihtele suuresti ominaisuuksiltaan. Natiivisovelluksen käyttöliittymä ladataan yleensä vain kerran osana sovellusta, joten vaihteleva internetyhteyden saatavuus ei ole ongelma. Käyttäjä voi ladata sovelluksen silloin, kun hyvä yhteys on saatavilla.

Nykyään on mahdollista tehdä natiivisovelluksia, jotka toimivat usealla laitetyypillä. Tällaisia sovelluksia voidaan valmistaa esimerkiksi Windows 10 -käyttöjärjestelmää varten kehitetyn Universal Windows Platform -sovellusarkkitehtuurin avulla. Päätelaitteissa voi

olla suuriakin eroja koon, syöttömenetelmän, käyttötarkoituksen, katseluetäisyyden tai muiden ominaisuuksien puolesta. Mukautuvien käyttöliittymien haasteet ovat näissä tapauksissa ilmeiset.

Mukautuvat käyttöliittymät ja erilaiset laitteet ovat yleistyneet niin paljon, että käyttöliittymä kannattaa suunnitella laajalla laiteyhteensopivuudella riippumatta suuremman työmäärän tuomista lisäkustannuksista. On selvää, että mukautuvat käyttöliittymät ovat tulleet jäädäkseen ja yhdelle laitteelle kohdistettujen käyttöliittymien suunnittelu on todennäköisesti käytännöllistä vain muista paljon poikkeavien laitteiden, kuten älykellojen, kanssa.

3 Käyttöliittymien esteettömyys

3.1 Esteettömyyden merkitys

Esteettömän käyttöliittymän tarjoama tieto ja toiminnot ovat saatavilla käyttäjille, joilla on normaalia käyttöä rajoittavia vammoja tai vaurioita. Esteettömyys on osa laajempaa käsitettä saavutettavuudesta, tai universaalista suunnittelusta, ja se merkitsee tuotteen tai palvelun lähestyttävyyttä kaikille ihmisille kyvyistä tai taustoista riippumatta. Esteettömyyttä ja saavutettavuutta käytetään myös toistensa synonyymeinä. [13.] Tässä raportissa esteettömyydellä viitataan nimenomaan rajoittuneiden henkilöiden huomioimiseen ja saavutettavuudella yleiseen lähestyttävyyteen ja käytettävyyteen käyttäjien taustoista riippumatta.

Esteettömyyden huomioiminen käyttöliittymissä hyödyttää kognitiivisesti, motorisesti, näöltään ja kuuloltaan vammaisia. Heihin kuuluvat muun muassa sokeat, värisokeat, heikkonäköiset, kuurot, liikuntaesteiset, ikäihmiset, kehitysvammaiset ja epileptikot. [1.]

World Health Organisationin (WHO) vuonna 2014 päivitetyn tiedotteen mukaan 285 miljoonalla ihmisellä maailmassa on näköhaittoja. Heistä 39 miljoonaa on sokeita. [14.] Lisäksi 8 % maailman miehistä ja 0,5 % naisista on puna-vihersokeita [15]. Skandinaviassa värisokeus on vielä yleisempää vahvan kaukasialaisen perimän vuoksi. Jopa 10–11 % skandinaavisista miehistä on värisokeita. [16.]

Vuonna 2012 WHO:n julkaiseman tiedotteen mukaan 360 miljoonalla ihmisellä, eli 5,3 %:lla koko maailman väestöstä, on kuulovamma [17, s. 2]. Sekä näkö- että kuulovammaisuuden tapauksissa ikäihmisillä ja taloudellisesti heikommin toimeentulevien valtioiden kansalaisilla on huomattavasti suurempi osuus kuin muilla [16; 17, s. 7–8, 12]. EU:n jäsenmaiden kansalaisista 80:tä miljoonaa rajoittaa jokin vamma [18]. Useiden tutkimusten perusteella jopa 20 % maailman väestöstä on jollain tapaa rajoittunut. Vaikka internetin ja tietotekniikan käyttöön vaikuttavien rajoitteiden osuus on varmasti pienempi, joka kymmenennen tai kahdennenkymmenennen ihmisenkin saavutettavuuden laiminlyöminen voi olla epätoivottavaa. [1.] Microsoftin vuonna 2003 tilaaman tutkimuksen mukaan jopa 57 % Yhdysvaltain 18–64 -vuotiaista tietokoneen käyttäjistä hyötyy esteettömyydestä. Näistä 40 prosenttiyksiköllä on lievä rajoite ja 17 prosenttiyksiköllä merkittävä rajoite tietokoneen käytössä. [19, s. 11–13.]

Täydellisesti saavutettavan käyttöliittymän tai minkä tahansa palvelun tuottaminen on mahdotonta. On aina jokin ihmisryhmä, jolla ei ole työkaluja tai mahdollisuuksia käyttää hyvinkin suunniteltua käyttöliittymää. Se ei tarkoita, etteikö täydellisyyteen voisi pyrkiä. Mahdollisimman monen käyttäjäkunnan huomioiminen tarjoaa mahdollisuuden kasvattaa luottamusta erityisryhmien kanssa. Esteettömyys on ainutlaatuisen tärkeää henkilöille, joilla on rajoitteidensa vuoksi vaikeuksia jokapäiväisessä elämässä ja useimpien palveluiden käyttäminen on hankalaa. Jopa seitsemän kymmenestä Euroopan unionin jäsenmaan kansalaisesta kokee esteettömyydellä olevan merkittävä positiivinen vaikutus ikäihmisille ja vammaisille [20, s. 9].

Yrityksillä ja suunnittelijoilla on kolme selkeää pääsyytä tavoitella esteettömyyttä. Esteettömyyden avulla tavoitetaan laajempi käyttäjäkunta, esteettömyyden laiminlyöminen voi olla eettisesti kyseenalaista ja joidenkin valtioiden laki voi vaatia esteettömyyttä julkaistuilta ohjelmistoilta tai verkkosisällöiltä rangaistuksen uhalla.

Sovelluskehityksessä esteettömyyden ei tulisi olla vain lisäominaisuus tai harkinnanvarainen kuluerä. Esteettömyyden huomioiminen ei ole pelkästään moraalinen kysymys; sillä on myös taloudelliset ja oikeudelliset perusteet. Toki tasa-arvon ja saavutettavuuden nimissä eettisen periaatteen tulisi riittää. Esteettömyys, ja saavutettavuus yleisesti, on tehty melko helpoksi toteuttaa alustojen kehittäjien varmistaessa automatisoinnin ja tarjotessa selkeät ohjeistukset käyttöönottoon. Käyttöliittymän suunnittelijalta ja kehittäjältä vaaditaan kuitenkin huolellisuutta ja osaamista esteettömän lopputuotteen rakentamiseen. Saavutettavuutta pidetään yhtenä erittäin hyvänä käyttöliittymän laadun mittarina.

3.2 Lainsäädäntö

Suomen perustuslain kuudes pykälä yhdenvertaisuudesta vaatii vammaisten ihmisten huomioimista [21]. Käytännössä tämä toteutuu maankäyttö- ja rakennuslaissa, joka vaatii esimerkiksi julkisten tilojen olevan saavutettavia liikuntarajoitteisille henkilöille. Suomen vuonna 2007 allekirjoittama YK:n vammaisten henkilöiden oikeuksia koskeva yleissopimus sisältää myös artiklan nimeltään Esteettömyys, joka ohjaa toimenpiteisiin kansalaisten yhdenvertaisuuden saavuttamiseksi rakennetussa ympäristössä [22]. Tätä ei ole kuitenkaan vielä vuoden 2016 maaliskuuhun mennessä ratifioitu. Suomen laissa ei kuitenkaan ole yleispäteviä vaatimuksia sovellusten tai käyttöliittymien esteettömyydestä.

Yhdysvalloissa julkisilla tahoilla on tiukat vaatimukset sähköisten palveluiden esteettömyydestä vuonna 1998 muutetun Section 508 of the Workforce Rehabilitation Act of 1973:n vuoksi [23]. Vuonna 1996 määrätty Section 255 of the Communications Act vaatii myös yksityisten tahojen tuottamien tietoliikennetuotteiden ja -palveluiden olevan esteettömiä. Täten valmistajien tulee varmistaa, että lopputuotteet on suunniteltu, kehitetty ja koottu esteettömiksi ja käytettäviksi vammaisille henkilöille siinä määrin, kuin on mahdollista. [24.]

Ison-Britannian tasa-arvoa ajava Equality Act of 2010 -määräys vaatii kaikkia maassa julkaistuja tietoliikennetuotoksia ottamaan käytettävyydessä huomioon vammaiset riippumatta siitä, ovatko tuotokset yksityisten tahojen, julkisten tahojen vai hyväntekeväisyysjärjestöjen tekemiä [25].

Euroopan unioni on julkaissut vaatimuksia jäsenmaidensa julkisille tahoille esteettömyyden huomioimiseen, mutta ne eivät ole lainopillisesti määrääviä. Suurin osa jäsenmaiden julkisista tahoista ei kuitenkaan seuraa suosituksia. Julkisia tahoja koskevien suositusten lisäksi EU on laatinut ohjeistukset myös yksityisille tahoille. European Accessibility Act on kokoelma ohjeita, joiden tarkoitus on helpottaa eri alojen yksityisten palveluiden ja tuotteiden esteettömyyttä EU:n jäsenmaissa tarjoamalla yhtenevät vaatimukset kaikille. [26.] European Accessibility Act ei ole lainopillisesti määräävä, mutta tutkimukset viittaavat esteettömyyden huomioimisella olevan taloudellista hyötyä etenkin pienille ja keskisuurille yrityksille. Tarjoamalla yhteiset yhtenäiset standardit unioni arvioi voivansa auttaa välttämään jopa 20 miljardin euron edestä kuluja vuoteen 2020 mennessä jäsenmai-

den hallintojen ja yritysten kesken. 78 % väestöstä uskoo yhteisten standardien helpottavan yritysten toimintaa muissa EU-maissa. EU:n pyrkimys onkin tässä tapauksessa edistää esteettömyyttä markkinoiden kautta. [20.]

Suomen valtiovarainministeriö on myös julkaissut kattavat ohjeistukset julkishallinnon verkkopalveluita varten. Verkkopalvelujen laatukriteeristö käsittelee hyvin myös esteettömyyttä. Kriteerit eivät kuitenkaan ole lainopillisesti pitävät. [27.]

Yhdysvaltoihin ja Isoon-Britanniaan julkaistavien ohjelmien ja tietoviestinnän palveluiden kehittäjien tulee olla erityisen huolellisia esteettömyyden toteutumisesta valtioiden lainsäädännön vuoksi. Monissa muissa valtioissa, kuten Itävallassa, Ranskassa, Suomessa ja Tanskassa, julkisten palveluiden tuottajilla on vastaavia vaatimuksia. Esteettömyyden laiminlyömisestä voi pahimmillaan seurata oikeustoimenpiteitä. Kaikki ihmiset huomioiva yhteiskunta laittaa tämän vastuun kehittäjien harteille. Jokaisella käyttöliittymäsuunnittelijalla ja -kehittäjällä olisi tästä syystä hyvä olla ainakin perusosaaminen esteettömyyden periaatteista ja toteutustavoista. Esteettömyys kuuluu vakaasti osaksi palveluntuottajan ammattietiikkaa.

3.3 Esteettömän käyttöliittymän suunnittelu

Esteettömyyttä tarvitsevalla käyttäjällä on usein käytössään laite tai ohjelma, joka mahdollistaa pääsyn tietoon ja toimintoihin verkko- tai käyttöjärjestelmäympäristössä tavanomaisesta käyttötavasta poiketen. Tällaisia ovat muun muassa tekstiä ääneen lukevat ruudunlukuohjelmat, kuten Microsoft Narrator, Apple VoiceOver, Googlen Android-käyttöjärjestelmälle kehittämä TalkBack tai JAWS. Ruudunlukuohjelma lukee näytöllä esiintyvän aktiivisen sovelluksen tekstin ja elementit järjestyksessä ja antaa tarvittaessa metatietoa sisällöstä ja interaktiivisista osista.

Ruudunlukuohjelmien moitteeton toiminta vaatii, että käyttöliittymä on tehty huolellisesti esteettömyysstandardeja seuraten. Tämä vaatii ammattitaitoa. Esteettömän käyttöliittymän toteutukseen kuuluu tavallista kattavampi testaus ja osaavan henkilöstön rekrytointi tai koulutus, jos sellaista ei vielä ole. Itse käytännön työ ei vaadi juuri enempää resursseja esteellisen lopputuotteen valmistamiseen verrattuna. Esteettömyys voi jopa tuoda säästöjä vähentämällä ylläpito-, päivitys- ja eri tuoteversioiden valmistuskuluja. Verkkoympäristössä esteettömyys auttaa myös hakukoneoptimoinnissa. [28.]

Käyttöliittymän esteettömyyttä koskevia standardeja tarjoavat usein esteettömyyttä edistävät organisaatiot, käyttöjärjestelmien kehittäjät ja suuret sovelluskehitystä tekevät yritykset. Web Content Accessibility Guidelines (WCAG) 2.0 on yleinen verkkoselaimella tarjottavaa sisältöä koskeva esteettömyysstandardi. Sen on kirjoittanut maailmanlaajuinen verkkostandardeja kehittävä ja ylläpitävä organisaatio World Wide Web Consortium (W3C) osana Web Accessibility Initiative (WAI) -liikettä. [29.] Tämän lisäksi muun muassa Microsoftilla, Applella ja Googlella on käytettävyys- ja esteettömyyssuositukset natiivisovelluksia varten.

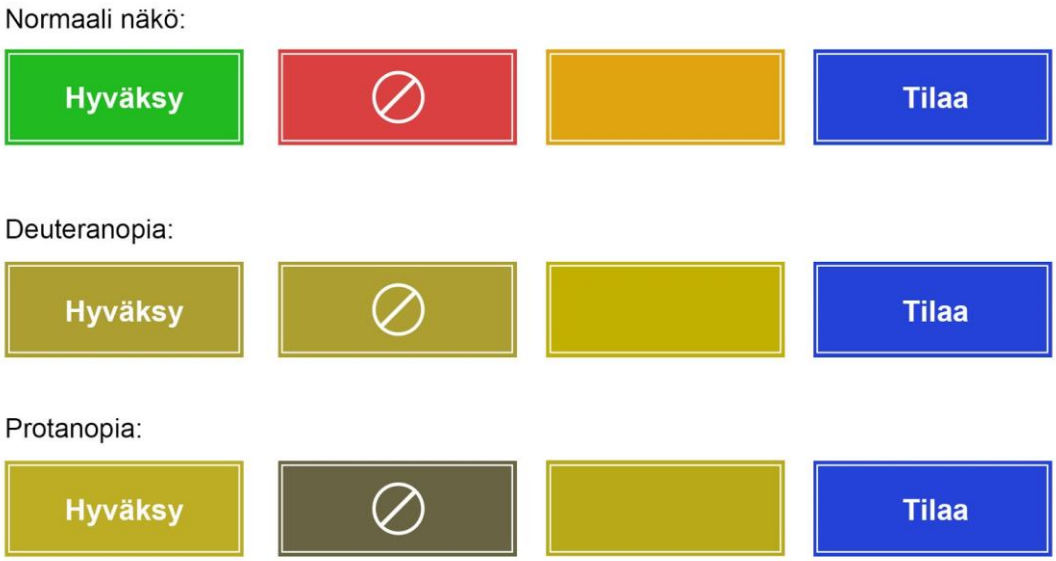
Yleinen harhaluulo on, että esteetön käyttöliittymä, verkkosivu tai sovellus ei näytä modernilta tai lähestyttävältä. Tämä ei ole totta. Esteettömyys perustuu tekniseen saavutettavuuteen ja hyvien graafisten ja teknisten käytäntöjen huomioimiseen. Esteettömyysohjeet suosittelevat rakentamaan käyttöliittymän järkeväksi ja johdonmukaiseksi, mistä on hyötyä kenelle tahansa käyttäjälle. [30.] Edes värisokeuden huomioiminen suunnittelussa ei rajaa käytettäviä värejä, kunhan osioissa on myös väristä poikkeavia merkitseviä yksityiskohtia. Myös kontrastin, kuten muidenkin esteettömyyttä tukevien parhaiden käytäntöjen, hyvä käyttö tekee graafisesta ulkoasusta selkeän ja toimivan.

3.4 Esteettömyyden parhaat käytännöt

Esteettömyysstandardeissa usein mainittuja huomiokohtia ovat sisältö, väri, kontrasti, typografia, asettelu, syöttömenetelmä, informaatiohierarkia, animaatio, videoiden tekstitykset ja semantiikka, eli käyttöliittymän osien sisäiset merkitykset. Käyttöliittymät toimivat usein samoin periaattein päätelaitteesta tai jakelukanavasta riippumatta, vaikka toteutustapa voi teknisesti ollakin eriävä. Useita standardien ohjeistuksia voidaan soveltaa universaalisti.

Pelkällä värillä ei ole hyvä osoittaa sovelluksen tai palvelun tilaa, koska ei voida luottaa käyttäjän kykyyn erottaa värejä toisistaan. Väri-indikaatiota voi tukea lisäämällä tekstiä, kuvakkeen tai muodon korostuksen. Suurin osa värisokeista, noin kolme neljästä, on puna-vihersokeita, joten erityisesti punaisen ja vihreän värin käyttöä tulee harkita tarkoin. Noin puolella värisokeista puuttuu kokonaan yksi kolmesta eri valon väriä aistivasta silmän tappisolutyypistä. He eivät kykene lainkaan aistimaan punaista tai vihreää valoa, ja heillä värisokeus on varsin rajoittava tekijä. Muilla värisokeilla on kaikki silmän tappisolutyypit, mutta jotkin niistä toimivat heikosti. Valtaosa värisokeudesta johtuu perimästä,

mutta esimerkiksi alkoholimyrkytys voi johtaa näön heikkenemiseen ja myös värisokeuteen. [31.] Kuvassa 3 on esimerkki, kuinka vihersokeudesta, eli deuteranopiasta, ja punasokeudesta, eli protanopiasta, kärsivät henkilöt voivat nähdä eri värillä esitetyt elementit.



Kuva 3. Esimerkki vihreästä, punaisesta, keltaisesta ja sinisestä elementistä normaalin näön omaavan, vihersokean (deuteranopia) ja punasokean (protanopia) silmin.

Käyttöliittymän tekstin ja taustan kontrasti on tärkeää luettavuudelle. Esimerkiksi Microsoft suosittelee 4,5:1-valoisuuskontrastia tekstin ja taustan välillä selkeyden saavuttamiseksi. Fontin on oltava kyllin isoa, ja fontin koon suurentamisen on hyvä olla mahdollista, jotta teksti on kaikkien luettavissa. Fontin suurennusmahdollisuuksia suunnitellessa on hyvä kiinnittää huomiota siihen, että käyttöliittymän rakenne pysyy eheänä kaikissa käyttötapauksissa. Yleensä käyttöjärjestelmässä tai verkkoselaimessa on sisäänrakennettu suurennusominaisuus, joten selkeän kirjaintyypin ja sopivan kontrastin käyttäminen ovat riittäviä toimenpiteitä luettavuuden varmistamiseksi. Teksti tulee esittää tekstille sopivassa muodossa aina kun mahdollista. Jos tekstiä on esimerkiksi kuvan sisällä, ruudunlukuohjelmat eivät osaa sitä lukea. Tekstiä tyylitellessä on muistettava myös värisokeita koskevat rajoitukset. [32.]

Yleisesti asettelun tulee olla selkeä ja tarjota kullekin sovelluksen tilalle, kuten aktiiviselle sivulle, relevanttia informaatiota yleisestä yksityiskohtaiseen tarkentuvassa järjestyksessä. Tämä auttaa ruudunlukuohjelmien käyttäjiä ymmärtämään sisältöaiheen varhaisessa vaiheessa lukuprosessia. Videoissa on hyvä olla tekstitykset, jotta myös kuurot

voivat ymmärtää puheen ja videon sanoman. Animaation on parhaiden käytäntöjen mukaan oltava käyttöliittymää tukeva eikä hallitseva. Se ei saa missään nimessä johtaa mahdollisiin epilepsiakohtauksiin ilman selkeää ennakkovaroitusta ja mahdollisuutta poistua sisällöstä, tai mielellään ollenkaan.

Sisältöä tukevissa elementeistä, kuten napeista ja kuvista, on löydyttävä tarvittavat metatiedot, jotta erilaiset apuohjelmat pystyvät välittämään tiedon vuorovaikutteisuudesta tai sisällöstä käyttäjälle. Esimerkiksi kuvilla on hyvä olla määritetty tekstikuvaus, jonka ruudunlukuohjelma voi lukea ääneen sokealle käyttäjälle kuvan sijaan.

Joissain tapauksissa on syytä kiinnittää huomiota itse sisältöön. Selkokielinen, lähestyttävä ja hyvin jäsennelty teksti on saavutettavuuden kannalta tärkeää vammaisille, joilla on vaikeuksia ymmärtää kieltä. Liiallisen monimutkaisia termejä tulee välttää, ja tekstiosioiden on hyvä olla johdonmukaisessa järjestyksessä. Sisällön ymmärrettävyyden tärkeys tulee parhaiten esille julkisissa palveluissa. Esimerkiksi Suomen eduskunnan esteettömyystyöryhmän [33] raportin mukaan eduskunnan viestinnän tulee olla ymmärrettävää kaikille. Se suosittelee edellä mainittuja ohjeita etenkin tietoliikenneviestintää koskien.

Sokea, heikkonäköinen tai motorisista häiriöistä kärsivä ei välttämättä voi käyttää hiirtä ja visuaalista ulkoasua käyttöliittymän selaamiseen. Vaihtoehtoinen selaustapa on verkkoselaimilla ja natiivisovelluksissa toimiva kohdistus, jota myös ruudunlukuohjelmat hyödyntävät paljon. Kohdistus osoittaa, mikä elementti on milloinkin aktiivisena. Yleensä kohdistus toimii vain interaktiivisissa elementeissä, kuten linkeissä tai napeissa. Tavallisin tapa hyödyntää kohdistusta on näppäimistön avulla, mutta kohdistusta voidaan ohjata myös esimerkiksi äänikomennoilla, kaukosäätimellä tai peliohjaimella. Jotta kohdistus käyttöliittymän selaamisessa toimii oikein, riittää usein kunkin alustan elementtien semantiikan oikeaoppinen käyttäminen sovellusta tai sivua rakennettaessa. Tärkeää on huolehtia, että kohdistuksen siirtojärjestys on looginen ja intuitiivinen.

Käyttöliittymän esteettömyyttä tulee testata säännöllisesti eri kehitysvaiheissa. Paras tapa testata käyttöliittymän esteettömyyttä on antaa se kokeiltavaksi sellaisille henkilöille, jotka esteettömyyttä tarvitsevat. Testauksessa kannattaa ottaa huomioon erilaiset persoonat, kuten aloittelevat ruudunlukuohjelmien käyttäjät ja testausammattilaiset, jotka ymmärtävät taustalla olevaa teknologiaa. Kannattaakin testata käyttöliittymiä erilaisilla persoonilla. Erilaisten testaajien avulla voi tulla esille ongelmia, joita ei muuten olisi tullut

ajatelleeksi. Käyttöliittymiä on hyvä testata myös itse ruudunlukuohjelmia tai pelkkää näppäimistöä käyttäen. Tarvittaessa näytön voi vaikka laittaa kiinni testauksen ajaksi. Kuten kaikkea käyttöliittymätestausta, myös esteettömyyden testausta on hyvä tehdä jo aikaisessa vaiheessa projektia, vielä kun puutteet ovat helposti korjattavissa. [34.] Jos sovelluksen tai verkkosisällön rakenne on alusta alkaen rakennettu huonosti, voi esteettömyyden lisääminen olla vaivalloista.

4 Universal Windows Platform -sovellusarkkitehtuuri

4.1 Ominaisuudet

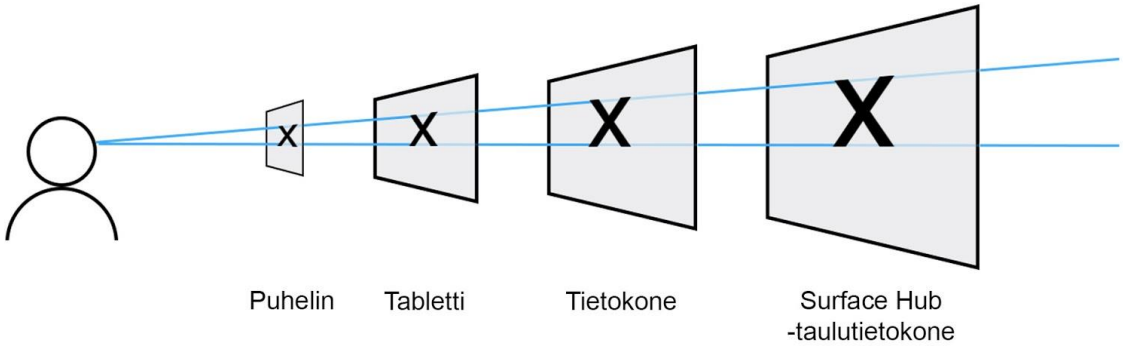
Universal Windows Platform (UWP) on sovellusarkkitehtuuri, joka mahdollistaa useita laitetyyppejä tukevien sovellusten tuottamisen. UWP-sovellus voi toimia millä tahansa laitteella, johon on asennettu Windows 10 -käyttöjärjestelmä. Arkkitehtuuri julkaistiin osana Windows 10- ja Windows 10 Mobile -käyttöjärjestelmiä heinäkuussa 2015. Sen avulla tehdyt sovellukset eivät ole yhteensopivia aiempien käyttöjärjestelmäversioiden kanssa. [35.]

Toisin kuin muut natiivisovellukset, UWP-sovellus kohdistetaan laitetyypille eikä käyttöjärjestelmälle. Sovellusarkkitehtuurin keskeinen sovellusohjelmointirajapinta (Application Programming Interface, API) on yhteinen kaikille Windows 10 -laitteille ja täten pelkästään keskeistä rajapintaa käyttävä sovellus toimii kaikilla laitetyypeillä. Lisäksi sovellusarkkitehtuuri tukee useita ohjelmointirajapintoja, jotka mahdollistavat eri laitetyyppien ominaisuuksien hyödyntämisen. UWP-sovellusten tukemia laitetyyppejä ovat tietokoneet, tabletit, älypuhelimet, Xbox-pelikonsolit, Surface Hub -taulutietokoneet ja IoT (Internet of Things) -laitteet, kuten älykellot tai kodinkoneet. Sovelluksille on myös yksi yhteinen jakelukanava Windows Store. [36.]

Universal Windows Platform -sovellusarkkitehtuuri tukee ohjelmointikieliä C++, C#, Visual Basic ja JavaScript. Sovelluksen esitystason voi toteuttaa käyttämällä HTML- tai XAML-merkintäkieliä tai DirectX -ohjelmointirajapintakokoelmaa, riippuen ohjelmointikielestä. HTML toimii JavaScriptin kanssa, XAML muiden kielten kanssa ja DirectX on vaihtoehtoinen esitystaso C++-kielelle. Koodi on lähtökohtaisesti jaettua kaikille alustoille,

ellei sovellukseen kuulu laitetyyppejä erikseen tukevia laajennuksia. Sovellus voi laitekohtaisesti käyttää esimerkiksi älypuhelimen kameraa, Xbox-pelikonsolin ohjainta tai useaa kosketuspistettä tukevan kosketusnäytön toimintoja. [36.]

UWP-sovellusarkkitehtuurissa on panostettu huomattavasti käyttöliittymän laiteyhteensopivuuteen. XAML- ja HTML-merkintäkielet perustuvat responsiiviseen joustavaan grid-asetteluun. Käyttöliittymä skaalautuu kuhunkin laitteeseen sopivaksi effective pixel -teknologialla, joka muuttaa elementtien kokoa ottaen huomioon näytön fyysisen koon ja laitteen katseluetäisyyden tyypillisessä käytössä. Kuvassa 4 havainnollistetaan effective pixel -skaalausta. On tosin huomioitava, että skaalaus on tarkka ainoastaan, jos käyttöliittymä on aseteltu 4-pikseliselle ruudukolle. Sovellusarkkitehtuurissa on keinoja asettaa elementtikohtaisia visuaalisia tyylejä näytön leveydestä, näytön korkeudesta tai tunnistetusta laitteesta riippuen. Kaikkien asettelumääreiden tulee olla neljällä jaollisia. Määreitä voi myös muokata dynaamisesti ohjelmointikielen avulla. Windows 10:ssä jokaisella laitetyypillä on omanlaisensa ikkunakehys, jossa usein on takaisin-painike ja muita sovelluksen hallintaan universaalisti yhteensopivia kontrolleja. [37.]



Kuva 4. Universal Windows Platform -sovelluksen effective pixel -skaalaus ja kuinka se vaikuttaa elementtien kokoon eri laitteilla perustuen näytön kokoon ja katseluetäisyyteen [37].

4.2 Kehitystyökalut

UWP-sovelluskehitys vaatii Windows 10 -käyttöjärjestelmän, Visual Studio 2015 -kehitysohjelman ja Microsoft-ohjelmistokehittäjän tilin. Lisäksi kaikilla testilaitteilla, mukaan lukien kehitysympäristöön käytettävällä tietokoneella, tulee antaa lupa testaukseen asetusten kautta. Kehittäjätilin aktivoimisen voi tehdä oman Microsoft-tilin asetuksissa. [38.]

Visual Studio 2015 sisältää useita kehitysapuja, muun muassa visuaalisen Design-näkymän, useita testaustyökaluja ja reaaliaikaisen kielentarkistuksen. Lisäksi se kääntää ohjelmointikielen ja paketoi valmiin sovelluksen. Ohjelmaan voi myös ladata lisäosia, kuten GitHub-integraation, tai valmiita sovelluspohjia. Visual Studio 2015 Community on kehitystyökalun ilmaisversio, joka sisältää kaiken tarvittavan UWP-sovelluskehitystä varten henkilökohtaisia tai korkeintaan miljoonan dollarin liikevaihdon omaavien pienyritysten kaupallisia projekteja varten. Ohjelman maksullisessa versiossa on joitain lisäominaisuuksia ja sallivammat käyttöehdot.

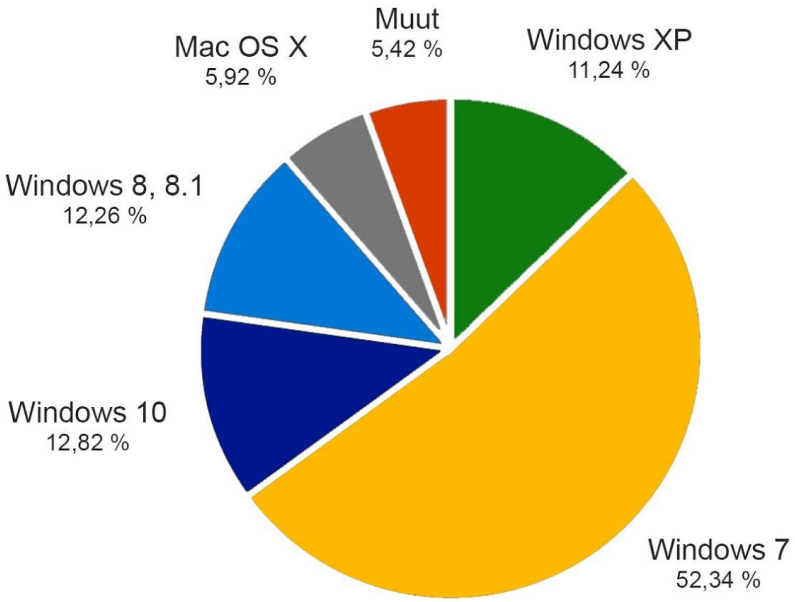
UWP-sovelluskehitys ei rajaa muita työkaluja Visual Studion lisäksi. Käyttöliittymäsuunnitteluun ja prototyyppien tekemiseen voi käyttää haluamiaan palveluita tai ohjelmia, kuten Adobe Illustrator, Adobe Photoshop tai PowerPoint. Microsoft tarjoaa kattavat tyylisuositukset käyttöliittymän suunnittelua varten ja suositusten mukaisia valmiita resursseja Photoshop- ja PowerPoint-ohjelmille. Versionhallintaan voi käyttää myös vapaasti valittavaa palvelua, kuten Git tai Apache Subversion (SVN).

Windows 10 -käyttöjärjestelmälle suunnattujen sovellusten toimiessa useilla eri laitteilla, käyttäjillä on mahdollisuus yhdistää palvelunsa yhden laite- ja sovellustarjoajan piiriin. Esimerkiksi Continuum-teknologialla Windows 10 Mobile -älypuhelin voi yhdistää tietokoneeseen ja hyödyntää tähän liitettyjä oheislaitteita, kuten hiirtä, näppäimistöä ja näyttöä [39]. Käyttäjän laitteiden yhtenevyys vaatii tietysti sitoutumista Microsoftin tuotteisiin ja niiden tukemiin ohjelmiin.

4.3 Sovellusarkkitehtuurin merkitys

Mikään muu sovellusarkkitehtuuri ei tue usealle laitteelle mukautuvia sovelluksia samalla tavalla kuin Universal Windows Platform. Applen iOS- ja tvOS-käyttöjärjestelmille on mahdollista tehdä Universal-sovelluksia, jotka toimivat iPhone-puhelimilla, iPad-tableteilla, iPod Touch -medialaitteilla ja Apple TV -älytelevisioilla [40; 41]. Nämä eivät kuitenkaan tue tietokoneille tarkoitettua OS X -käyttöjärjestelmää. Apple on Microsoftin suurin kilpailija laiteyhteensopivissa sovelluksissa, sillä mikään muu käyttöjärjestelmäkehittäjä ei hallitse yhtä suurta tuettujen laitteiden kirjoa tai käyttäjäkuntaa. Googlen Android-käyttöjärjestelmää ei ole pöytä- tai kannettaville tietokoneille ja Linux-käyttöjärjestelmiä ei esiinny mobiililaitemarkkinoilla. Erilaisia laitteita tukevien sovellusten mahdollistamat käyttötapaukset ovat kuitenkin harvassa vahvasti jakautuneiden markkinoiden vuoksi.

Kuten kuvasta 5 näkyy, vuoden 2016 maaliskuussa Windows 10:n markkinaosuus kaikista tietokoneiden käyttöjärjestelmistä on vain 12,8 %, mutta sen voidaan olettaa nousevan ajan myötä [42]. Microsoft tarjoaa ilmaisen käyttöjärjestelmäpäivityksen Windows 7- ja Windows 8.1 -laitteille heinäkuun 2016 loppuun mennessä. On todennäköistä, että moni käyttäjä päivittää käyttöjärjestelmänsä vielä ennen aikarajan umpeutumista. Näiden käyttöjärjestelmien yhteenlaskettu osuus tietokoneiden käyttöjärjestelmistä on 62,9 %, reilusti yli puolet. Kaiken lisäksi Windows 10 on Microsoftin Jerry Nixonin [43] mukaan yrityksen viimeinen erillinen käyttöjärjestelmäversio. Tämän vuoksi asennusten voidaan olettaa lisääntyvän tasaisesti ajan myötä. Lisäksi käyttöjärjestelmälle tehtyjen sovellusten eliniän voidaan odottaa olevan aiemmille versioille ohjelmoituja sovelluksia huomattavasti pidempi.



Kuva 5. Pöytätietokoneisiin asennettavien käyttöjärjestelmien markkinaosuudet maaliskuussa 2016 [42].

Microsoftin tuottamilla käyttöjärjestelmillä on erittäin huono asema mobiililaitemarkkinoilla. Vuoden 2015 toisen vuosineljänneksen lopulla vain 2,6 % mobiililaitteissa oli Windows Phone -käyttöjärjestelmä [44]. Vuoden lopussa julkaistut älypuhelimet Lumia 950 ja Lumia 950 XL kuitenkin toivat kilpailuun mukaan UWP-sovelluksia tukevan Windows 10 Mobile -käyttöjärjestelmän. Tämä ei silti välttämättä riitä merkittävän markkinaosuuden takaamiseksi. UWP-sovelluskehitys pysyy todennäköisesti vahvana ainakin tietokoneympäristössä, mutta jää vielä nähtäväksi, kuinka paljon laajasta laitetyyppien tuesta on käytännössä hyötyä.

5 Käyttöliittymän toteutus Windows 10 -sovelluksessa

Insinöörityön osana toteutettiin käyttöliittymä Windows 10 -sovellukselle käyttäen Universal Windows Platform -sovellusarkkitehtuuria. Käyttöliittymän tavoitteena oli tukea älypuhelimia, tabletteja, kannettavia tietokoneita ja pöytätietokoneita. Sovellus tehtiin aiemman Windows 8.1 -sovelluksen pohjalta, mutta käyttöliittymä toteutettiin kokonaan uudelleen. Lisäksi muuta pinnan alla piilevää tekniikkaa päivitettiin sovellusrajapintojen muutosten vuoksi. Projektin arvioitu toteutusaika oli noin kymmenen viikkoa. Tähän ei laskettu valmiin sovelluksen julkaisuvalmisteluihin ja hyväksyttämiseen kuluvaa aikaa. Sovellus toteutettiin digitaalisiin oppimisratkaisuihin erikoistuneen Bitville Oy:n toimesta asiakasprojektina. Asiakkaana toimi monikansallinen suuryritys.

5.1 Sovelluksen vaatimukset

Sovelluksen tarkoitus oli toimia oppimisalustana kenelle tahansa, joka haluaa opiskella sovelluksen tarjoamaa aihepiiriä. Käyttäjäkertomukset kuvaavat eri käyttötapauksia, mitä sovelluksessa voi tehdä. Valmiin sovelluksen käyttäjäkertomuksiin kuuluivat seuraavat asiat:

- sovelluksen käyttöönottoa varten opastuksen saaminen
- kirjautuminen sisään ja tilin luominen palveluun Microsoft Account –tunnuksella
- harjoituksen valitseminen useasta eri aihepiiristä ja kategoriasta
- edetessä vaikeustasoltaan nousevien pisteytettyjen harjoitusten tekeminen, niistä palautteen saaminen ja harjoitusten yhteydessä olevien vinkkien paljastaminen
- henkilökohtaisen kehityksen seuraaminen aihepiiri- ja kategoriakohtaisesti pisteitä keräämällä ja tasoja läpäisemällä
- kategorioiden lataaminen laitteelle, ladatun tiedon poistaminen ja ladattujen kategorioiden sisältämien harjoitusten tekeminen offline-tilassa
- kategoriaan liittyvän teorian opiskelu harjoitusten aikana
- aihepiirikohtaisten uutisten lukeminen
- käyttäjien ja käyttäjäryhmien vertailu pisteittäin kuluvan kuukauden, edellisen kuukauden ja palvelun olemassaolon ajalta

- käyttäjäryhmien luominen, ryhmiin kutsuminen, niihin liittyminen ja niiden sisäinen viestikeskustelu
- oman tilin julkisuuden, sukupuolen ja roolin asettaminen ja halutessa tilin poistaminen palvelun tietokannasta
- sovelluksen lataaminen Windows Store -palvelusta.

Sovelluksen tuli olla myös yhteensopiva olemassaolevan back end -arkkitehtuurin, eli käyttäjälle näkymättömien pääosin palvelimella suoritettavien toimintojen, kanssa. Sovelluksen julkaisun tekee lopulta asiakas, mutta valmis sovellus viimeistellään julkaisua varten asiakkaan puolesta.

Käyttöliittymää varten asetettiin hyvin tiukat ja kattavat esteettömyysvaatimukset pohjautuen asiakasyrityksen omaan yli 180-sivuiseen ohjeistukseen. Suurin osa ohjeiden sisällöstä perustui WCAG 2.0 -standardiin ja Section 508 -vaatimuksiin. Koska sovellus julkaistaisiin maailmanlaajuiseen jakeluun, oli tärkeää perehtyä avainmarkkinoiden, kuten Yhdysvaltojen, lainsäädäntöön esteettömyydestä. Asiakasyrityksen antama ohjeistus käsitteli esteettömyyttä kyllin syvällisesti täyttääkseen lain määräämät vaatimukset kaikkialla sitä noudatettaessa. Käyttöliittymään ei sisältynyt videota, ääntä tai monimuotoista animaatiota, joten esteettömyys koski lähinnä staattista ja interaktiivista sisältöä.

Oppimissovelluksen käyttöliittymän esteettömyydessä oli asiakasyrityksen standardiin pohjautuen huomioitava seuraavat asiat:

- Sovelluksen navigointiin oli oltava useita eri tapoja. Navigoinnin tuli toimia osoituksen lisäksi myös kohdistuksen avulla ainakin näppäimistöä käyttäen. Osoitukseen voi käyttää esimerkiksi hiirtä tai kosketusnäyttöä ja kohdistukseen näppäimistön lisäksi äänikomentoja, peliohjainta tai kaukosäädintä. Kohdistus oli oltava selkeästi erotettavissa ja kohdistuksen siirron järjestyksen tuli olla looginen.
- Nopeita välähdyksiä tuli välttää; yli kolmea välähdystä sekunnissa ei saanut missään vaiheessa esiintyä.
- Käyttöliittymän oli pysyttävä eheänä ja toimivana fontin kokoa suurennettaessa.
- Teksti tuli esittää aina mahdollisuuksien mukaan teksti- eikä kuvamuodossa. Tekstin ja taustan kontrastin tuli olla vähintään 4,5:1.
- Kuvilla tuli olla korvaava tai selittävä teksti ruudunlukuohjelmia varten.
- Elementtien toiminnallisuutta tai merkitystä ei saanut esittää pelkän värin avulla.

- Elementtien, jotka vaativat käyttäjän syöttämää tietoa, tuli olla selitettyjä sanallisesti.
- Sisällön järjestyksen tuli olla looginen ja jokaisella selvästi erillisellä osiolla tuli olla selkeä informatiivinen otsikko. Sisällön oli oltava selattavissa ja käytettävissä ruudunlukuohjelman avulla.
- Linkkien oli oltava selkeästi erotettavissa ja niiden sisältämien tekstien informatiivisia.
- Toiminnallisten elementtien, toistuvien rakenteiden ja navigaation tuli olla visuaalisesti yhteneviä ja erotettavia toisistaan.

Valmiin sovelluksen tuli läpäistä asiakasyrityksen omat testit, jotka keskittyivät yksityiskohtaisesti sovelluksen toimivuuteen ja esteettömyyteen. Testit perustuivat yrityksen omaan esteettömyysstandardiin ja UWP-sovellusten tekniikkaan.

Sovelluksen jatkamassa palvelussa esiintyvä haasteosio jätettiin toistaiseksi vaatimuksista pois. Haasteiden tarkoituksena olisi tarjota ajoitettuja, esimerkiksi viikon auki olevia, tehtäviä käyttäjille. Käyttäjät kilpailisivat parhaasta suoritusajasta, ja parhaan ajan saanut voisi voittaa palkintoja. Haasteet olisivat aina erityisiä tapahtumia, jotka asiakasyritys järjestäisi. Toistaiseksi haasteita on voinut suorittaa vain palvelun verkkosovelluksen kautta, mutta uusien haasteiden julkaisuun mennessä olisi tarkoitus päivittää myös natiivisovellus tukemaan niitä.

5.2 Työtavat ja työkalut

Kehitysympäristönä käytettiin Visual Studio 2015 -ohjelmistoa ja versionhallintaan yksityistä GitHub-tietovarastoa. Testilaitteina toimivat Hewlett Packard EliteBook Folio 9740m -kannettava tietokone, Microsoft Lumia 640 ja Lumia 950, joihin kaikkiin on asennettu Windows 10 -käyttöjärjestelmä.

Sovelluksen toteutti nelihenkinen tiimi Bitville Oy:stä. Tiimiin kuului kolme ohjelmistokehittäjää ja graafinen suunnittelija. Projektin tuotantoprosessi seurasi ketterän kehityksen periaatteita. Asiakasyrityksen kanssa pidettiin parin viikon välein katsauskokouksia, joissa seurattiin työn etenemistä, kysyttiin palautetta asiakkaalta ja sovittiin lyhyen aikavälin tavoitteista. Suuren aikaeron vuoksi kokoukset olivat usein iltamyöhään, ja tämän vuoksi niihin osallistui vain projektipäällikkö ja projektitiimin ulkopuolinen Bitvillen edustaja.

Työtehtävät sovelluksen kehityksessä jaettiin avoimesti osaamisen ja kunkin henkilön aikataulujen mukaan. Projektitiimin jäsenet tekivät samanaikaisesti myös muita projekteja. Kolmella ohjelmistokehittäjällä oli suuntaa-antavat roolit front end -ohjelmoijana, eli sovelluksessa itsessään ajettavien käyttäjälle näkyvien toiminnallisuuksien laatijana, back end -ohjelmoijana ja projektipäällikkönä. Varsinaiset tehtävät jakautuivat kuitenkin tilanteen mukaan, eikä selkeää jaottelua ollut. Graafinen suunnittelija loi suunnitelman käyttöliittymälle kehittäjien tukemana ja tarjosi konsultaatiota visuaalisesta ilmeestä ja käytettävyydestä läpi projektin.

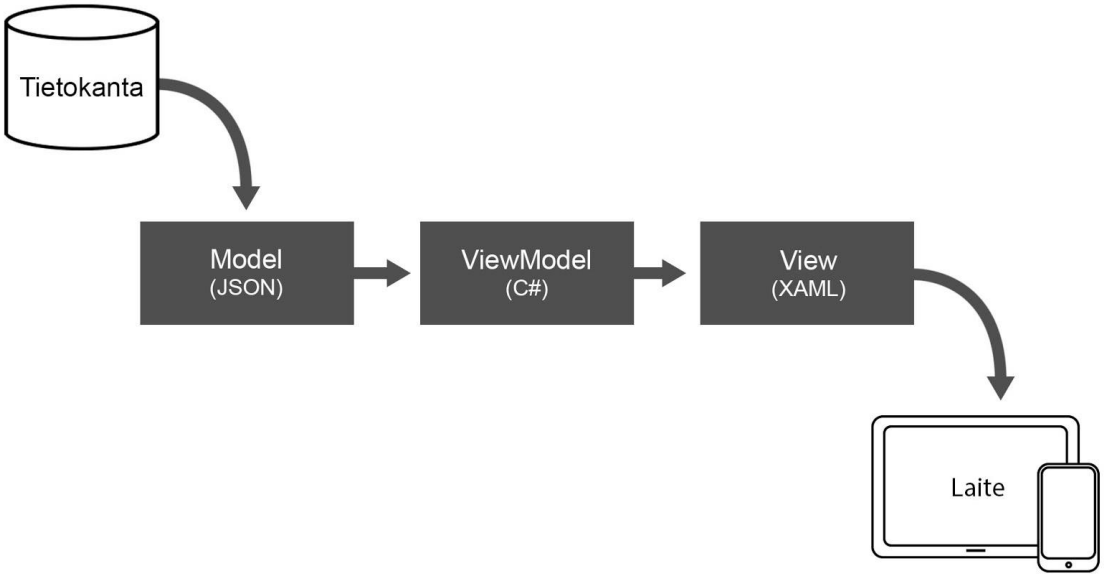
Selkeitä välivaiheita projektissa olivat käyttöliittymä- ja prototyyppiversioiden valmistaminen ja esittäminen asiakkaalle. Näitä seuraavien vaiheiden tavoitteet ja sovelluksen kehityssuunta päätettiin yhdessä asiakkaan kanssa versioista saatujen kommenttien perusteella. Jäykkää aikataulua ei ollut projektin aikarajaa lukuun ottamatta; välivaiheita sovittiin tarvittaessa.

5.3 Ratkaisut

Insinöörityössä toteutettu oppimissovellus ohjelmoitiin käyttäen C#-ohjelmointikieltä, XAML-merkintäkieltä ja model-view-viewmodel-arkkitehtuurimallia. Nämä nähtiin luontevimmiksi valinnoiksi tekijöiden osaamisen ja valmiina olleen Windows 8.1 -sovelluksen koodin suhteen. Käyttäjän laitteelle näkymätön back end -osuus on isännöity Microsoft Azure -pilvipalvelimella.

Sovelluksessa käytetty MVVM (model-view-viewmodel) on arkkitehtuurimalli, jonka Microsoftin sovellusarkkitehdit Ken Cooper ja Ted Peters kehittivät vuonna 2005. MVVM:n tarkoitus on mahdollistaa tapahtumakeskeinen ohjelmointi, sovelluksen osien erottelu niiden tarkoituksen mukaan ja helppo tiedon linkittäminen näkymään. Se on jatkokehitetty yleisestä model-view-controller (MVC) -arkkitehtuurimallista. [45.]

MVVM-arkkitehtuurimalliin kuuluu kolme osaa: model, view ja viewmodel. Model edustaa sovelluksen käyttämää tietoa, view edustaa käyttöliittymää ja viewmodel yhdistää nämä kaksi. Viewmodel välittää tietoa model-osalta view-osalle ja hallinnoi näkymän toimintoja. Tiedon linkittämisen ja sovelluksen organisoinnin lisäksi arkkitehtuuri helpottaa ohjelmointiprosessin hallintaa ja interaktiivisten ominaisuuksien toteuttamista. [45.] Kuvassa 6 on havainnollistettu arkkitehtuurimallia.



Kuva 6. MVVM-arkkitehtuurimallin toiminta ja käytetyt kielet eri tasoilla sovelluksessa.

Oppimissovelluksen käyttöliittymä on adaptiivinen ja responsiivinen. Tämä toteutettiin antamalla kaksi eri breakpointia 640 ja 1008 pikselin näytön- tai ikkunanleveyksille. Microsoft [46] suosittelee breakpointeja 720 ja 1024 pikselin leveyksille, mutta tässä tapauksessa suosituksista tehtiin poikkeus parempaa älypuhelinten vaakanäkymän tukea varten. Käyttöliittymässä on pienemmille mobiililaitteille kapeampi yleisnäkymä ja yli 640 pikseliä leveille näytöille laajempi vaakanäkymä, joka vaihtuu adaptiivisesti käyttöliittymälle annetun tilan mukaan. Vielä laajemmassa, yli 1008-pikselisessä näkymässä navigaatio ja sivupalkissa esiintyvä lisäsisältö, kuten ryhmäkeskustelu, ovat koko ajan näkyvissä. Navigaation avaava nappi, jota merkitään kolmella allekkain esitetyllä vaakaviivalla, avaa kapean navigaatiopalkin tarkemmat selostukset. Mobiilinäkymässä navigaatio on piilotettu kokonaan ja avattavissa nappia painamalla. Tämä SplitView-niminen navigaatiorakenne on UWP-sovellusarkkitehtuurista valmiina. Mobiililla sivupalkin sisältö on löydettävissä navigaation avulla täysikokoisina sisältösivuina. Navigaation ja esityshierarkian muutos estää lisäsisällön ja harjoitusten samanaikaisen tekemisen, mutta mahdollistaa selkeämmän käyttöliittymän pieninäyttöisillä laitteilla.

Käyttöliittymän adaptiivisuuteen käytettiin kahta eri elementtiä: VisualStateManager kietoo yhteen käyttöliittymän tyylimääritelmät ja AdaptiveTrigger vaihtaa näitä breakpointien avulla. Yhdessä ne toimivat mediakyselyiden tavoin antaen sovellukselle eri visuaalisen tilan näytön tai sovellusikkunan koosta riippuen. Breakpointien tarkoitus on suunnata käyttöliittymä tietokoneelle, tabletille tai älypuhelimelle. Kannettavien tietokoneiden

skaala asettuu joko tabletti- tai tietokonenäkymän piiriin. Kuvassa 7 esitetään sovelluksen käyttöliittymän rakennetta eri breakpointien sisällä.

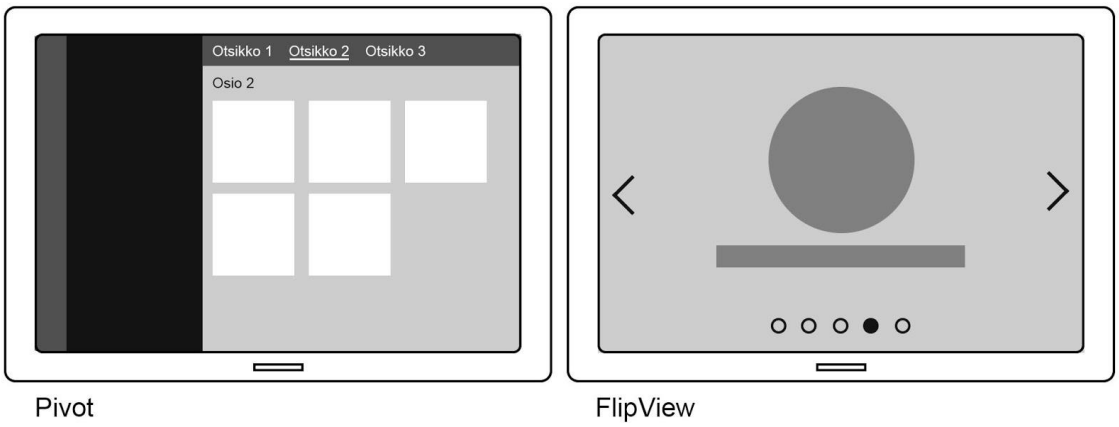
Kuva 7. Oppimissovelluksen käyttöliittymän rakenne kahdessa eri breakpointissa.

Sivupalkki ja varsinainen sisältöosio on jaettu navigaatiohierarkkisesti erillisiksi. Sisältöosion harjoituksia ja sivupalkkia voi selata autonomisesti toisistaan riippumatta. Käyttäjä pystyy esimerkiksi kysymään neuvoa ryhmäkeskustelussa harjoituksia tehdessään. Koska harjoitukset ovat sovelluksen pääsisältö, Windows 10 -kehyksissä, kuten Surface-tabletin alalaidassa tai sovellusikkunan ylänurkassa, esiintyvä takaisin-painike ohjaa pääsisällön navigaatiohistoriaa. Sivupalkki- ja sisältöosiot ovat eroteltu Frame-elementtien avulla. Frame hallitsee kunkin osion sisälle asetettujen Page-sivukehyksien navigaatiota ja tilaa itsenäisesti, joten osiot ovat toisistaan täysin irrallisia. MVVM-arkkitehtuurimallin avulla koko sovelluksen sisältämä tieto on keskitetty model-osalle, joten jokainen sovelluksen osio pysyy synkronissa toisiinsa nähden. Esimerkiksi harjoituksista saadut pisteet päivittyvät automaattisesti sivupalkin sisältöön ja sovellus näyttää koko ajan reaaliaikaista tietoa.

Sovelluksen sisäiseen navigaatioon käytettiin kolmea erilaista ratkaisua. Pääosin sisältöä navigoitaessa sisältö vaihdettiin kokonaan napin tai linkin painalluksesta. Ryhmänäkymän sisällä käytettiin poikkeavasti välilehtimäistä Pivot-rakennetta, ja uuden

käyttäjän ensin näkemässä opastusosiossa käytettiin karusellimaista FlipView-navigaatiota. Pivot ja FlipView ovat molemmat UWP-sovellushierarkian sisältämiä valmiita ratkaisuja.

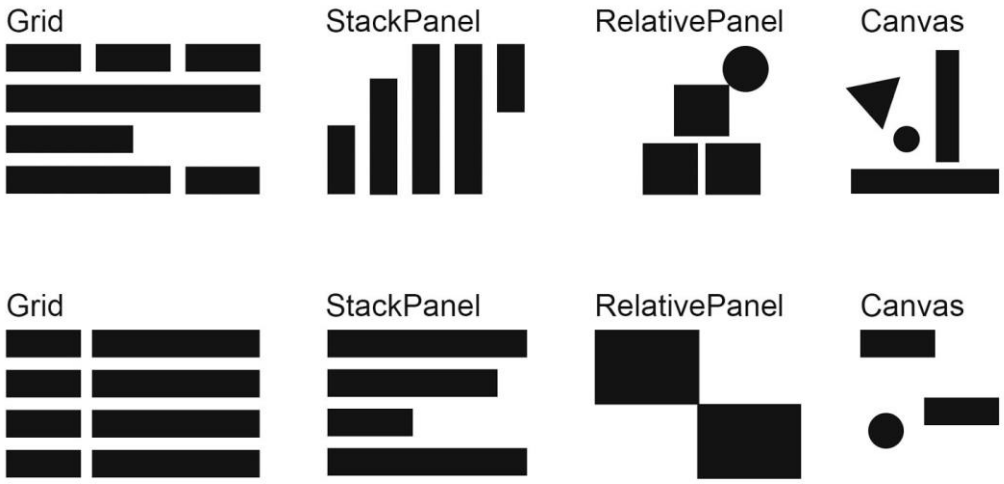
Pivot-navigaatiossa osion ylä- tai alalaidassa on välilehtien tapainen kohdistettava ja osoitettava rivi interaktiivisia otsikoita. Kohdistetun rivin selaaminen nuolinäppäimillä tai yksittäisen otsikon painaminen siirtää Pivot-elementin sisään määriteltyjä PivotItem-sivuja vaakatasossa näyttäen aina yhden sivun kerrallaan. Sivuttain selaaminen toimii mobiililaitteilla myös sormen pyyhkäisyllä. FlipView-navigaatio on monella tapaa samanlainen, mutta sitä kontrolloidaan sivuttais- tai ylä- ja alanappeja painamalla. Mobiilissa osioiden vaihtaminen onnistuu myös pyyhkäisemällä. FlipView-navigaation ongelmana on selaustilan merkitseminen. UWP ei tarjoa valmiiksi indikaattoreita navigaation havainnollistamiseksi. Tämän vuoksi oppimissovellukseen lisättiin rivi pieniä ympyröitä indikoimaan aktiivista osiota ja osioiden kokonaismäärää. Tämän koettiin parantavan käyttäjäkokemusta etenkin uudelle käyttäjälle. Kuvassa 8 on havainnollistettu Pivot- ja FlipView-navigaatioita ja niiden eroavaisuuksia.



Kuva 8. Pivot- ja FlipView-navigaatiorakenteet havainnollistettuina.

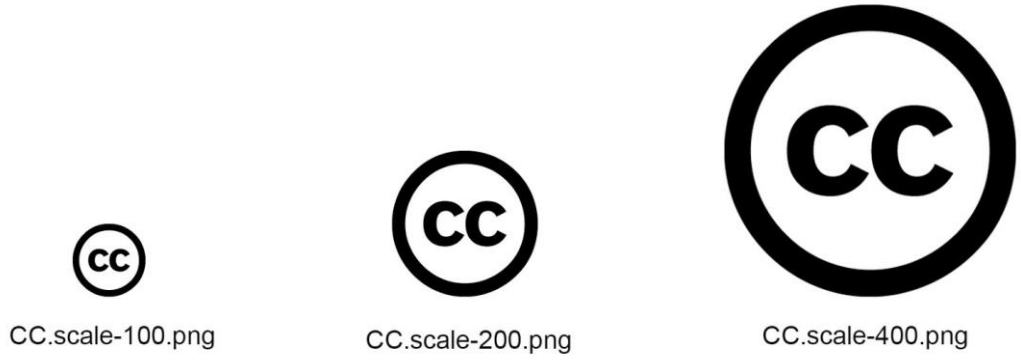
UWP-sovelluksen käyttöliittymä on automaattisesti responsiivinen käytettäessä sen tarjoamia valmiita XAML-asetteluelementtejä. Grid-, Canvas-, RelativePanel- ja StackPanel-elementit täyttävät käyttöliittymän tarpeet kukin eri tavoin. Ruudukoittava Grid- ja annetun sisällön vaaka- tai pystysuunnassa pinoava StackPanel-elementit mahdollistavat joustavan taulukkomaisen asettelun. Gridille voi antaa erikseen rivejä tai sarakkeita. RelativePanel-elementin voi sijoittaa vapaasti suhteessa sen isäntäelementtiin, ja Canvas toimii vapaana kenttänä, jonka sisälle voi laatia elementtejä koordinaattien avulla. [47.]

Elementeille annettavien määreiden, kuten leveyden tai marginaalien, avulla voi asettelua muovata hienovaraisesti. Kuvassa 9 on havainnollistettu eri elementtien asettelun ominaisuuksia. Oppimissovelluksessa yleistason asetteluun käytettiin pääosin Grid- ja StackPanel-elementtejä.



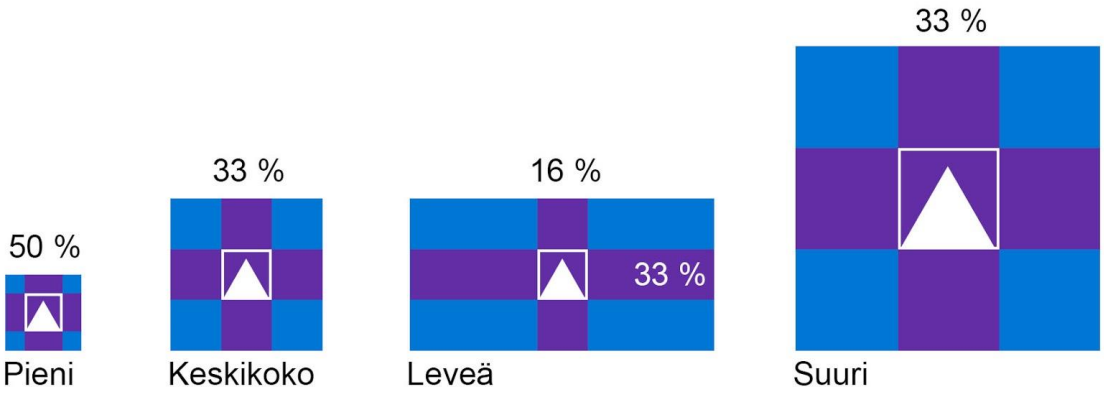
Kuva 9. Grid-, StackPanel-, RelativePanel- ja Canvas-elementtien toiminta.

Sovelluksessa esiintyviä kuvia, kuten logoa ja räätälöityjä kuvakkeita, varten tarjottiin kolme erikokoista versiota. UWP-sovelluksen yhteydessä ladataan aina näytön tarkkuuteen sopivin versio kuvasta. Tätä kutsutaan adaptiiviseksi skaalaukseksi (engl. adaptive scaling). Automaattista kuvaversion valintaa voi käyttää tallentamalla tiedostonimeen ennen tiedostopäätteen määrittävää osaa merkkisarjan .scale-200, .scale-400 tai .scale- jollain muulla numeroarvolla. Esimerkiksi yhden kuvatiedoston nimi voisi olla AppLogo.scale-300.png. Merkkisarjassa esiintyvä numero esittää, kuinka monta prosenttia skaalattuna eri versiot ovat suhteutettuna lähtökohtaiseen, pienimpään kuvaan. [48.] Sopivan kokoisen kuvan tarjoamisen tuloksena on varmasti terävä kuvanlaatu näytön ominaisuuksista riippumatta, olettaen että kuvatiedoston sisältö on tarkka. Kuvassa 10 on esimerkki adaptiivisesti skaalatun kuvan eri versioista.



Kuva 10. Kuvake kolmessa eri adaptiivisessa koossa: yksin-, kaksin- ja nelinkertaisena.

Adaptiivista skaalausta käytetään myös sovelluksen käynnistysruudun ja käynnistyskuvan kuvakkeissa. Käyttäjän asetuksista riippuen mobiililaitteen kotinäkymässä tai tietokoneen aloitusvalikossa käynnistysruutuja voi olla neljää eri kokoa, joten erikokoisten kuvakkeiden tarjoaminen on siitäkin syystä oleellista. Ruutujen ja sovelluksen käynnistyessä näytettävän käynnistyskuvan kuvakkeisiin Microsoft suosittelee tarjoamaan kaksin- ja nelinkertaisesti suurennetut versiot optimaalisen tarkkuuden mahdollistamiseksi kullekin päätelaitteen näytölle. Leveälle käynnistysruudulle tulee myös antaa oma kaksi kertaa tavallista leveämpi versionsa. Sovelluksen logon ympärillä tulee olla tarpeeksi tyhjää tilaa käynnistysruudun taustaan suhteutettuna. Tilaa tulee olla pienellä ruudulla vähintään puolet kuvakkeen leveydestä tai korkeudesta jokaiselta sivulta tai suurella ruudulla vähintään kuvakkeen leveyden tai korkeuden verran. Käynnistysruutu on siis aina joko kaksi tai kolme kertaa kuvaketta leveämpi ja korkeampi. Leveässä ruudussa vaakatilaa on vielä reilusti enemmän, noin kuusi kertaa kuvakkeen leveyden verran yhteensä. [48.] Kuvassa 11 on esitetty käynnistysruutujen koot ja suositellut sovelluskuvakkeiden mitat taustan suhteen.



Kuva 11. Käynnistysruutujen koot ja suositellut kuvakkeiden mitat suhteessa ruutujen kokoon.

Yksinkertaisiin kuvakkeisiin voidaan UWP-sovelluksissa käyttää Windows 10:n mukana tulevaa Segoe MDL2 Assets -fonttia ja SymbolIcon-elementtiä. Käyttöjärjestelmä itse hyödyntää fonttia käyttöliittymässään, joten sovelluksissakin käytetyt Segoe MDL2 Assets -kuvakkeet ovat graafisesi yhteensopivia. Oppimissovelluksessa fontin kuvakkeita käytettiin muun muassa navigaatiossa ja käyttäjän kehitystä merkitsevien pisteiden ja tasojen yhteydessä.

UWP-sovelluksessa tyylit voi määrittää elementtikohtaisesti, mutta toistuvasti esiintyvien tyylien kanssa on kätevää määrittää viitattavia resursseja. XAML-elementille määritettävä ResourceDictionary kokoaa uudelleenkäytettäviä sisältöresursseja. Jokaisella resurssilla on x:key-määre, jonka avulla siihen viitataan. Resursseja voi käyttää missä tahansa määritellyn kehyselementin sisällä. Resurssi voi olla teksti, tyyli, väri, brush-väri tai template-malli. [49.] DataTemplate-mallia voi esimerkiksi käyttää listojen täyttämiseen dynaamisesti listalle osoitettujen tietojen pohjalta. Esimerkkikoodissa 1 on havainnollistettu brush-värin resurssimäärittelyä ja viittausta tekstielementissä.

```
<Page x:Class="AppName.NewPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
>
    <Page.Resources>
        <SolidColorBrush x:Key="ColorRed" Color="#ff0000"/>
    </Page.Resources>
    <TextBlock Text="I am red." Style="{StaticResource ColorRed}"/>
</Page>
```

Esimerkkikoodi 1. Brush-väriresurssin määrittely ja sen asettaminen tekstielementtiin.

Oppimissovelluksessa käytettiin resurssimäärittelyjä yhtenäisen käyttöliittymän tyylin toteuttamiseen. Sovelluksen ylimmälle tasolle luotiin universaaleja tyylejä, kuten erilaisia otsikoita, leipätekstin tyylejä ja värejä. Näin saatiin toteutettua CSS-tyylitiedoston kaltainen yleinen viitelista, josta useimmat eri rakenne-elementit pystyivät viittaamaan tyylejä. Yhtenäisten tyyliarvojen kautta tyylimääreiden sovelluksenlaajuinen muuttaminen ja eri arvojen testaus on helppoa. Kun määre on annettu vain kerran ja elementit viittaavat yhteen lähteeseen, pystytään helposti tyylittelemään käyttöliittymä vaikka kokonaan uudestaan uudelle brändille. Sovelluksen käyttöliittymä pysyy yhtenäisenä ja jaetut tyylit ennaltaehkäisevät huolimattomuusvirheitä.

Tyyliresursseja voi määritellä elementteihin joko staattisiksi StaticResource-resursseiksi tai ThemeResource-teemaresursseiksi. Värien määritteleminen teemoihin on esteettömyyttä ajatellen huomattavasti parempi tapa verrattuna staattisiin resursseihin. Staattiset resurssit tulevat aina teemamääritysten edelle. Jakamalla värit kolmeen eri sisäänrakennettuun Light-, Dark- ja High Contrast -teemaan käyttöliittymän värit ja kontrastit eivät mene tarkoituksettomasti ristiin. Käyttäjän vaihtaessa Windows-väriteemansa High Contrast -korkeakontrastiteemaksi luettavuus ei kärsi. Heikkonäköisille korkeakontrastiteeman saatavuus ja toimivuus on erityisen tärkeää siltä vaaditun 7:1-kontrastin vuoksi.

UWP-sovellukselle ohjelmoitavan käyttöliittymän esteettömyyttä ajatellen on huomioitava useita teknisiä seikkoja. Lähtökohtaisina tavoitteina esteettömyyden varmistamiseksi ovat näppäimistön ja ruudunlukuohjelman käytön tuki, suurennus- ja korkeakontrastiasetusten sisällyttäminen sovellukseen ja mahdollisten vaihtoehtoisten tai täydentävien käyttöliittymän osien tarjoaminen. UWP-arkkitehtuurin interaktiiviset XAML-elementit, kuten napit ja valitsimet, ovat automaattisesti esteettömiä ja tukevat näppäimistöä ja ruudunlukuohjelmia. Ne myös tukevat erilaisia syöttömenetelmiä, kuten kosketusnäyttöjä, äänikomentoja tai hiirtä. Valmiita elementtejä suositellaan käytettävän aina kun mahdollista itse tehtyjen elementtien sijaan. Jos ohjelmistokehittäjä haluaa rakentaa itse räätälöityjä interaktiivisia elementtejä, suositellaan niiden pohjautuvan johonkin jo valmiiksi olemassa olevaan elementtiin. [50.]

GridView- ja ListView-listaelementtien selaaminen kohdistuksen avulla voi esteettömyyttä ajatellen kaivata muutosta. Lähtökohtaisesti kohdistuksen siirtäminen listan sisällä näppäimistöllä, kaukosäätimellä, peliohjaimella tai äänikomennoilla on mahdollista

vain nuoli- tai suuntanäppäimillä, muttei tavanomaisella kohdistuksen siirrolla. Laaditussa sovelluksessa haluttiin listojen olevan helposti selattavia kohdistuksen avulla niiden toimiessa myös linkkeinä. Listat eivät myöskään olleet pitkiä.

Listojen sisään automaattisesti luoduille ListViewItem-sisältöelementeille annettiin määre TabNavigation arvolla Local hyödyntämällä XAML-kielen Style-tyylittelyelementtiä itse listan sisällä. Tyylisääntö lisättiin sovelluksen ylätasoon niin, että se vaikuttaa kaikkiin listoihin ja korvaa kohdistuksen oletusarvoisen käyttäytymisen. Lopputuloksena sovelluksen sisällä olevat listat, kuten ryhmä- ja kurssilistat, olivat saavutettavissa tavallisen kohdistuksen siirron avulla. Myös ruudunlukijaohjelman käyttö helpottui huomattavasti, sillä se hyötyy merkittävästi kohdistuksen käytöstä sovelluksen sisältöä selatessa. Esimerkkikoodissa 2 havainnollistetaan tyylielementin avulla tehtyä ratkaisua listojen kohdistuksen toiminnan muutokseen.

```
<Application
    x:Class="AppName.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
>
    <Application.Resources>
        <Style TargetType="ListView">
            <Setter Property="TabNavigation" Value="Local"/>
        </Style>
    </Application.Resources>
</Application>
```

Esimerkkikoodi 2. ListView-listaelementtien kohdistuksen toiminnan muutos. Muutoksen jälkeen listan sisältämät elementit ovat osana tavallista kohdistusjärjestystä.

Joissain tapauksissa AutomationProperties.Name-nimimääritteen lisääminen elementtiin on tarpeen ruudunlukuohjelmia varten. Jos elementillä ei ole kyseistä määritettä tai välitöntä tekstisisältöä, ruudunlukuohjelma ei pysty lukemaan elementtiä ymmärrettävästi. Napeille, linkeille, tekstikentille ja muille vuorovaikuttaville elementeille on syytä antaa nimimäärite tarkentamaan vuorovaikutuksen merkitystä. Esimerkkikoodissa 3 esitetään AutomationProperties.Name-määritteen lisäys nappielementille.

```
<Button Click="SendButtonClicked" AutomationProperties.Name="Send">
    <SymbolIcon Symbol="Send"/>
</Button>
```

Esimerkkikoodi 3. Määrite AutomationProperties.Name on lisätty XAML-nappielementille, joka sisältää vain toimintaa havainnollistavan symbolin.

6 Windows 10 -sovelluksen testaus ja julkaisu

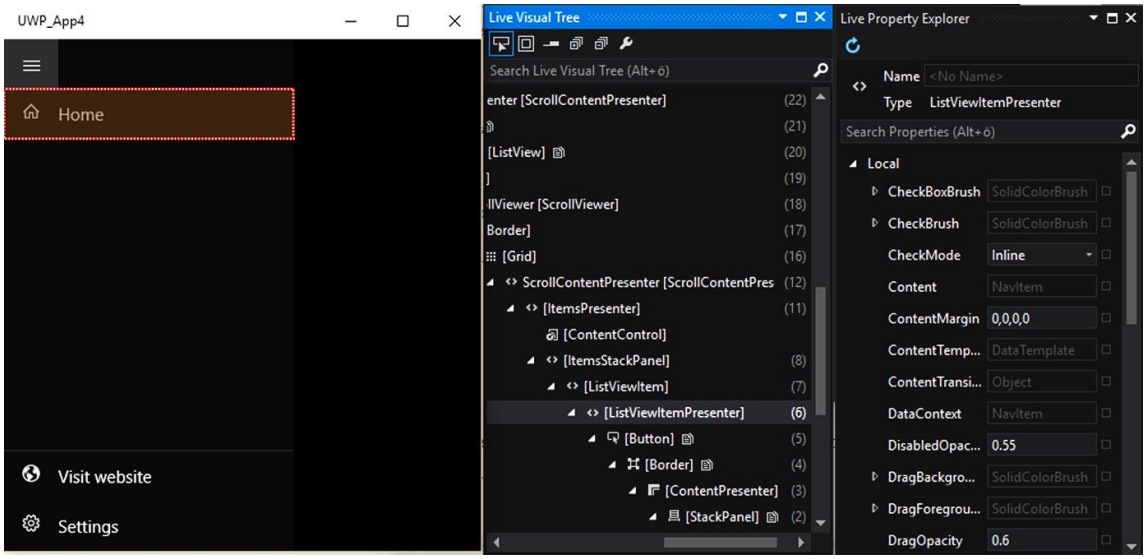
6.1 Käyttöliittymän testaus

Käyttöliittymän testaus on käytettävyyden testaamista ja toiminnallisuuden varmistamista ennen sovelluksen tai palvelun julkaisua. Tähän on useita keinoja. Testit voidaan jakaa automatisoituun, manuaaliseen ja käyttäjätestaukseen. Testausta on tärkeää tehdä läpi koko kehitysprosessin eikä pelkästään juuri ennen julkaisua. Näin voidaan välttää perusteellisia virheitä toiminnallisuudessa jo aikaisessa vaiheessa ja varmistaa käyttöliittymän toimivuus myös lopputuotteessa.

Visual Studio 2015 tarkistaa automaattisesti sovelluksen koodin toimivuuden haluttaessa jo tiedoston tallennuksen yhteydessä. Lisäksi Visual Studio suorittaa testejä, kun sovellus käännetään ja käynnistetään testausta tai julkaisua varten. Tämä toimii ohjelmiston IntelliSense-ominaisuuden avulla. Sovelluksen voi käynnistää testausta varten kehityskäytössä olevalle tietokoneelle, yhdistetylle testilaitteelle tai laite-emulaattorille. Emulaattorilla voi mallintaa eri mobiililaitteita ja käyttöjärjestelmäversioita ja kokeilla sovelluksen toimivuutta ilman konkreettista laitteistoa. Emulaattori ei kuitenkaan koskaan korvaa oikealla laitteella testausta. [51.]

Vaikka Visual Studio 2015 sisältää Design-näkymän, joka näyttää käyttöliittymän elementit ja asettelun kääntämättä koodia, siihen ei kannata täysin luottaa. Design-näkymässä eivät näy ulkoisesta tiedosta generoidut elementit, kuten tietokannan täyttämät listat. On tärkeää testata sovellusta aika ajoin ja kokeilla toimivuutta manuaalisesti. Automaattiset testauksen menetelmät auttavat pitämään sovelluksen eheänä ja toimivana, mutta manuaalista testausta tarvitaan käytettävyyden ja käyttöliittymän toimivuuden varmistamiseen.

Live Visual Tree ja Live Property Explorer ovat Visual Studio 2015 -ohjelmiston osia, joiden avulla voi reaaliajassa tarkastella ja muuttaa XAML-elementtejä ja niiden määreitä sovelluksen ollessa käynnissä, esimerkiksi testauksen aikana. Tämä onnistuu kohdistamalla elementtiä testilaitteella tai tietokoneella. Nämä aputyökalut auttavat löytämään optimaaliset asetukset kullekin elementille ja varmistamaan, että ne ovat oikein muotoutuneet esimerkiksi esteettömyyttä ajatellen. Tällä tavalla voi ratkaista yksittäisiä ongelmakohtia ja toteuttaa käyttöliittymän hienosäätöä. Kuvassa 12 on havainnollistettu Live Visual Tree- ja Live Property Explorer -näkymien toimintaa.



Kuva 12. Live Visual Tree- ja Live Property Explorer -näkymät Visual Studio 2015 -ohjelmassa.

Käyttäjätutkimus on osa käyttäjäkeskeistä suunnittelua. Käyttäjäkeskeinen suunnittelu perustuu käyttäjien arvioimiseen ja seuraamiseen ja heidän tarpeisiinsa vastaamiseen. Tarkoituksena käyttäjätutkimuksessa on perehtyä sovelluksen tai palvelun tosiasialliseen käyttöön oikeissa tilanteissa tai mahdollisimman lähellä niitä. Käyttäjätutkimuksen tärkeitä työkaluja ovat muun muassa fiktiiviseen tai oikeaan kerättyyn dataan perustuvat tyypillisen käyttäjän persoonat ja reaaliaikainen käyttäjätestaus. [52, s. 46, 49—50.]

Minkä tahansa sovelluksen tai palvelun käyttöliittymää voi testata käyttäjätestauksen avulla. Käyttäjätestauksessa simulaatio, prototyyppi tai valmis sovellus annetaan loppukäyttäjää vastaavan henkilön kokeiltavaksi ja seurataan sen käyttöä. Usein testaajalle annetaan yksi tai useampia tavoitteita, kuten käyttäjäryhmän luominen, jonka tämä toteuttaa testitilanteessa. Käyttäjätestauksessa ei ole tarkoitus testata käyttäjää, vaan käyttöliittymää. Käyttäjä voidaan mieltää avustajaksi. Käyttäjätestaus antaa realistisen

kuvan käyttöliittymään osoitetuista oletuksista, käyttäjän tarpeista ja käyttöliittymän senhetkisistä haasteista. Kuten kaikkea testausta, käyttäjätestausta tulisi harjoittaa kaikissa projektin vaiheissa konseptoinnista lähtien. [52, s. 47—48, 158.]

Microsoft Narrator -ruudunlukuohjelmaa voi käyttää työkaluna esteettömyyden testaamiseen. Testaukseen kannattaa käyttää myös muita ruudunlukuohjelmia ohjelmien erojen vuoksi. JAWS on erinomainen vaihtoehto sen yleisyyden vuoksi. Ohjelman suosion laskusta huolimatta vuonna 2015 yli 30 % ruudunlukuohjelman käyttäjistä suosi JAWSia, enemmän kuin mitään muuta ruudunlukuohjelmaa [53]. Ruudunlukuohjelmat toimivat moitteettomasti UWP-sovelluksen testauksen aikana. Ruudunlukuohjelman avulla on helppo tarkistaa, luetaanko elementit oikein, onko elementtien kohdistuksen järjestys looginen ja kuinka käytettävä sovellus on yleisesti ruudunlukuohjelmaa käyttäville. Narratoria, JAWSia tai muita ruudunlukuohjelmia voi käyttää myös osana käyttäjätestausta antamalla testattava sovellus kokeiltavaksi ruudunlukuohjelman avulla.

Ennen UWP-sovelluksen julkaisua Windows Store -palveluun sovellus on tarkistettava käyttäen Microsoftin Windows App Certification Kit -sertifiointitestiä. Jakopalvelu käyttää samaa testipakettia sertifioidessaan palveluun lähetettyjä sovelluksia, joten testin avulla voidaan varmistaa sovelluksen läpipääsy julkaisuprosessista. Testin voi ladata ja suorittaa itse tai sen voi tehdä automaattisesti UWP-sovelluksen paketoinnin yhteydessä Visual Studio 2015:lla. [54.] Sertifiointitesti ajaa useita sovelluksen toiminnallisuuteen ja eheyteen liittyviä testejä. Se varmistaa muun muassa, että sovelluksen tarvittavat metatiedot on sisällytetty oikein, sovellus on laskennallisesti kyllin optimoitu ja sovellus on turvallinen käyttää. Testit eivät ota kantaa esteettömyyteen tai käyttöliittymän ominaisuuksiin. Sertifiointiestin ajaminen vaatii testattavan sovelluksen ja tietokoneen lisäksi vain testausympäristöksi asetetun Windows 10 -käyttöjärjestelmän [54].

UWP-sovelluksen prosessointitehon käyttöä voi tutkia myös itse Visual Studion Debug-valikon alta löytyvän diagnostiikkatyökalun avulla. Diagnoosin avulla voi löytää ongelmakohtia sovelluksen optimointiin liittyvissä ongelmissa. Ongelmakohtia korjaamalla voidaan esimerkiksi parantaa sovelluksen nopeutta ja pienentää virrankulutusta mobiililaitteilla.

6.2 Windows Store -julkaisu

Kun UWP-sovellus on valmis ja testattu, se voidaan lähettää Windows Store -jakelupalveluun julkaistavaksi. Tätä varten sovellus täytyy paketoida käyttäen Visual Studio 2015 -ohjelmaa. Paketoinnin aikana ohjelma opastaa tarvittaessa rekisteröitymään Microsoft-kehittäjäksi ja varaamaan sovellukselle nimen tai valitsemaan jo aiemmin omalla tilillä rekisteröidyn käyttämättömän nimen. Prosessin ohella sovelluksen voi myös validoida Windows App Certification Kit -testillä ja testata paketoitua sovellusta paketoinnin tekevällä tai siihen liitetyllä laitteella. Paketoinnin tuloksena on appxupload-tiedosto, joka ladataan palveluun. [55.]

Microsoft Account -tilin rekisteröiminen kehittäjäksi maksaa 14 euroa yksityishenkilölle tai 75 euroa yrittäjälle tai yrityksen työntekijälle. Rekisteröidyttyä tilin nimissä voi myydä sovelluksia ja palveluita Windows Storen kautta. Yritystilin lisäominaisuuksiin kuuluvat yrityksen rekisteröidyn nimen käyttäminen ja kattavammat analytiikka- ja sovellusominaisuudet.

Julkaisun Windows Storeen voi tehdä Windows Dev Center -sivustolla oman Microsoft kehittäjätilin kautta. Julkaisun yhteydessä voi määrittää asetuksia sovelluksen hintaan, saatavuuteen, julkaisuajankohtaan, kategoriaan, laitevaatimuksiin, ikärajoituksiin, tietoihin ja käyttöehtoihin liittyen. Asetuksia voi muokata myös myöhemmin ja julkaistaessa ei ole pakko vielä tarjota valmista sovellusta. Sovelluksen tiedot voi alustaa ja nimen varata etukäteen. Tietoja ja sovellusversiota voi myös päivittää julkaisun jälkeen. [56.]

Kun sovellus on toimitettu julkaistavaksi, se siirtyy Windows Storen tarkistusjonoon. Jonossa edettyään sovellus päätyy sertifiointitesteihin, joiden avulla Microsoft varmistaa sovelluksen turvallisuuden, toimivuuden ja sisällön säännönmukaisuuden. Sisällön tarkistaa oikea henkilö, vain toimivuuden kannalta prosessi on automatisoitu käyttäen Windows Certification Kit -sertifiointitestiä. [57.] Tarkistusta varten täytyy varmistaa, että sovelluksen kaikki osat ovat toimivia ja että testaajalle annetaan riittävät oikeudet ja ohjeet kaikkien sovelluksen ominaisuuksien testaamiseen. Koko tarkistusprosessi on yleensä valmis muutamassa tunnissa, mutta se saattaa ajoittain kestää kauemmin. [58.]

Sovelluksen voi halutessaan julkaista beetatestaukseen. Tällöin lupaa julkiseen julkaisuun ei anneta, vaan Windows Store luo korkeintaan 250 koodia, joiden avulla valitut henkilöt voivat ladata sovelluksen ja testata sitä. Koodin voi aktivoida beetatestaukseen

tarkoitetulla sivulla erityisen linkin takaa. Testikäyttäjä tarvitsee Windows 10 -varustetun laitteen ja Microsoft Account -tilin testausta varten. Jakelupalvelu ei tarjoa erillistä palautekanavaa beetatestaukseen, joten sovelluksen sisälle on hyvä lisätä tähän tarkoitettu ominaisuus tai vähintään yhteystiedot. Sovellus ja sen sisäiset maksut kannattaa myös asettaa ilmaiseksi, jotta testaajien ei tarvitse käyttää omaa rahaansa testaukseen. Sovellus täytyy olla tarkistettu ja muutoin valmis julkaistavaksi, jotta beetatestaus on mahdollista. [59.]

7 Yhteenveto

Insinöörityön lopputuloksena oli toimiva, Windows 10 -laitteisiin mukautuva ja esteetön oppimissovellus. Koska sovellus on toistaiseksi julkaisematta, dataa käyttöönotosta ei ole saatavilla. Käyttäjätyytyväisyyttä voidaan mitata vasta julkaisun jälkeen. Mahdolliset päivitykset, jälkityö ja raportointi jäävät myös julkaisun jälkeiseen lähitulevaisuuteen.

Projektin aikana opittiin paljon, sillä sovellus oli ensimmäinen Bitvillen tuottama UWP-sovellus. Yritys on aiemmin tuottanut vastaavanlaisen Windows 8.1 -mobiilisovelluksen samaa oppimispalvelua varten, mutta UWP-arkkitehtuuri ja adaptiiviset sovellukset olivat täysin uutta. Tekniikka on vielä alle vuoden vanha. Projekti oli erinomainen tilaisuus oppia käyttöliittymäsuunnittelua, sovelluskehitystä, sovellusprojektin etenemisen seurantaa, sovelluksen julkaisua ja usealle laitetyypillä kohdentamista. Asiakas, Bitville ja itse tekijät olivat tyytyväisiä lopputulokseen ja projektin etenemiseen pitkästä esteettömyyden varmistusprosessista huolimatta. Projektin päätteeksi, kun sovellus on julkaistu, projektin tiimi pitää vielä retrospektiivin, jossa käsitellään lisää opittua ja analysoidaan projektia.

Arvioituja syitä projektin onnistumiseen on monia. Versionhallinnan käyttö, työtehtävien avoin jakaminen, läheinen työskentely, huolellinen suunnittelu, selkeä suunnitelma ja säännölliset katsaukset projektin etenemiseen auttoivat projektin työstämisessä. GitHubin avulla pystyi helposti merkitsemään puutteita ja korjattavia virheitä niin, ettei useampi henkilö tehnyt samaa asiaa yhtä aikaa. Projektin koodin haarauttaminen väliaikaisesti useaan lomittaiseen versioon oli elintärkeää työnkulkua ajatellen. Versioiden ylläpitäminen helpotti ominaisuuksien erillistä kehitystä ja vaikutti varmasti lopullisen sovelluksen yhtenäisyyteen. Kun kehittäjät istuivat lähekkäin avoimessa toimistoympäristössä, kyn-

nys keskinäiseen kommunikointiin oli matala. Kommunikoinnin helppous auttoi ratkaisemaan pieniä ongelmia nopeasti, jakamaan mielipiteitä eri ratkaisumahdollisuuksista ja antamaan neuvoja. Ketterä kehitys ilman vesiputousmaista suunnitelmaa mahdollisti käyttöliittymän visuaaliset muutokset myös kehityksen aikana ilman suurta vaivaa. Ajoittaiset viidestätoista minuutista tuntiin kestäneet aivoriihet helpottivat sovelluksen käyttöliittymän, käytettävyyden ja koodin rakenteellisten haasteiden ratkaisemisessa kehitystyön aikana.

Projektissa olisi voinut suoriutua paremmin aikatauluttamalla työn osuudet huolellisemmin. Bitvillen sisäisesti sovittu asiakkaan vaatimuksia tiukempi aikataulu venyi hieman odotetusta. Aikataulun muuttuminen ja työmäärän aliarvioiminen oli kuitenkin odotettavissa uuden teknologian käyttöönoton yhteydessä. Myös sisäinen kommunikaatio olisi voinut olla vielä huolellisempaa ja tiuhempaa. Yhdessä tapauksessa kaksi kehittäjää päätyi tekemään samaa, onneksi pientä, asiaa yhtä aikaa. Pääosin kommunikaatio toimi kuitenkin moitteettomasti sekä sisäisesti että asiakkaan kanssa.

Esteettömyyden toteuttaminen osoittautui haasteelliseksi projektitiimin kokemattomuuden vuoksi. Elementeille yhteisten värien asettaminen teemoittain ja listojen optimoiminen ruudunlukuohjelmia varten tulivat esille vasta myöhäisessä kehitysvaiheessa, kun esteettömyyttä tarkasti kolmas osapuoli. Korkeakontrastiteeman sisällyttäminen myöhäisessä vaiheessa söi useamman työpäivän, mikä alusta asti suunniteltuna olisi varmasti vaatinut vähemmän. Näistä ongelmista olisi voitu välttyä yhä perusteellisemmalla opiskelulla, koulutuksella ja laadunvarmistuksella projektin alkuvaiheessa.

Responsiivinen suunnittelu, ja mukautuvat käyttöliittymät yleisesti, on verkkoselaimelle toteutettavan sisällön kannalta oletusarvo. Mukautuvuus on natiivisovelluksissa uutta. Muutosta johtaa Universal Windows Platform -sovellusarkkitehtuuri ja laiteyhteensopivat Windows 10 -sovellukset. Tulevaisuudessa mukautuvien käyttöliittymien muodot kehittyvät todennäköisesti entisestään laitteiden ja sovellusten monimuotoistuessa. Virtuaalitodellisuuden maailmanlaajuisen liikevaihdon arvioidaan kaksinkertaistuvat kahdessa vuodessa vuoteen 2015 verrattuna [60]. Tämä ja muut uudet teknologiat avaavat laiteyhteensopivien natiivisovellusten tavoin uusia uria ja haasteita käyttöliittymäsuunnittelun ja -tekniikan alalla. Jää kuitenkin nähtäväksi, kuinka pitkälle Windows 10 Mobile -alusta etenee hyvin pienellä markkinaosuudella.

Esteettömän käyttöliittymän toteutus ei yleisesti ottaen ole vaikeaa. Esteettömyyden huomioiminen ja suunnitteleminen on erittäin perusteltua kaikissa tapauksissa. Julkisilla tahoilla on syystä tiukkoja vaatimuksia esteettömyydelle. Yrityksille ja organisaatioille esteettömyys kaikessa on erinomainen mahdollisuus, jos ei eettinen velvollisuus. Esteettömyys ei vaadi erityisammattilaista, ellei sovelluksen tai palvelun käyttöliittymää suunnata erityisesti vammaisille tai aisteiltaan rajoittuneille ihmisille. Kenen tahansa suunnittelijan, ohjelmoijan ja käyttöliittymien kanssa toimivan olisi hyvä tietää ainakin perusteet esteettömyydestä, sen merkityksestä, tarpeesta ja tekniikasta. Esteettömyyttä ei voi jättää täysin sitä tarvitsevien ja heidän käyttämien laitteiden ja palvelujen armoille teknisistäkin syistä. Esteettömyydelle on annettava mahdollisuus. Muutoin sitä ei ole ollenkaan, ja tämä voi estää saavutettavuuden merkittävälle osalle potentiaalisia käyttäjiä.

Universal Windows Platform on toimiva, helposti opittava ja hyvin dokumentoitu sovellusarkkitehtuuri. Sen käyttöönotto ei vaadi suurta vaivaa tai panosta, eikä se eroa merkittävästi aikaisemmille käyttöjärjestelmäversioille suunnatusta Windows Presentation Foundationista (WPF). Alustana Windows 10 on yhtenäinen, mutta sen heikkous on kehno mobiilialustojen kattavuus. Universaali laiteyhteensopivuus voi kuulostaa hienolta, mutta ei ole vielä selvää, kuinka merkittäväksi ilmiöksi ajatus lopulta kasvaa.

Sovellusten jakopalvelu Windows Storen tarjonta on vielä pientä, mutta sitä ei voi samalla tavalla verrata Googlen tai Amazonin vastaaviin palveluihin, sillä ne kohdistavat sovellustarjonnan ainoastaan mobiililaitteille. Applen vastaava App Store tarjoaa sovelluksia myös tietokoneille, mutta sovellukset eivät ole samalla lailla adaptiivisia eri päätelaitteille. Windows 10 tai siihen päivittyvät käyttöjärjestelmät ovat myös huomattavasti yleisempiä jopa 75 %:n markkinaosuudella tietokoneille suunnatuista käyttöjärjestelmistä [42]. Windows Storen tarjonnan voidaan odottaa vielä paranevan, kun yhä useampi siirtyy Windows 10 -sovellusten kehitykseen. Tulevaisuudessa on erittäin todennäköistä, että tähän mennessä totutusti eri verkkosivuilta ladatut ohjelmistot siirtyvät myös Microsoftin jakopalveluun.

Lähteet

1 Introduction to Web Accessibility. 2016. Verkkodokumentti. WebAIM. <http://webaim.org/intro/>. Päivitetty 15.3.2016. Luettu 27.3.2016.

2 Henry, Shawn Lawton. 2012. Social Factors in Developing a Web Accessibility Business Case for Your Organization. Verkkodokumentti. W3C Web Accessibility Initiative. <https://www.w3.org/WAI/bcase/soc>. Julkaistu elokuussa 2005. Päivitetty 7.10.2012. Luettu 27.3.2016.

3 Leiniö, Timo. 2012. Mitä on responsiivinen design? Verkkodokumentti. Sofokus. <https://www.sofokus.com/blogi/mita-on-responsiivinen-design/>. Päivitetty 19.7.2012. Luettu 27.3.2016.

4 Mill, Chris ynnä muut. 2016. Responsive design versus adaptive design. Verkkodokumentti. Mozilla Dev Center. <https://developer.mozilla.org/en-US/docs/Archive/Apps/Design/UI_layout_basics/Responsive_design_versus_adaptive_design>. Päivitetty 29.2.2016. Luettu 27.3.2016.

5 Frost, Brad. 2013. The Many Faces of 'Adaptive Design'. Verkkodokumentti. <http://bradfrost.com/blog/post/the-many-faces-of-adaptive-design/>. Päivitetty 19.12.2013. Luettu 27.3.2016.

6 Marcotte, Ethan. 2011. Responsive Web Design. New York: A Book Apart.

7 Graham, Geoff. 2015. Responsive and Adaptive Design. Verkkodokumentti. CSS-Tricks. <https://css-tricks.com/the-difference-between-responsive-and-adaptive-design/>. Päivitetty 11.11.2015. Luettu 27.3.2016.

8 Hale, David. 2016. Guide to Universal Windows Platform (UWP) apps. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/get-started/universal-application-platform-guide>. Päivitetty 28.2.2016. Luettu 27.3.2016.

9 Harris, Matthew. 2015. Responsive or Adaptive Design - Which is Best for Mobile Viewing of Your Website? Verkkodokumentti. The Medium Well. <http://mediumwell.com/responsive-adaptive-mobile/> Päivitetty 21.5.2015. Luettu 27.3.2016.

10 Benefits of a Responsive Website! 2013. Verkkodokumentti. Isadora Design. <http://isadoradesign.com/web-design-journal/benefits-of-responsive-web-design>. Päivitetty 11.10.2013. Luettu 27.3.2016.

11 Make sure your site's ready for mobile-friendly Google search results. 2015. Verkkodokumentti. Google. <https://support.google.com/adsense/answer/6196932?hl=en>. Luettu 27.3.2016.

12 Mander, Jason; Mc Grath, Felim. 2016. GWI Device Summary Q1 2016. Verkkodokumentti. GlobalWebIndex. <http://www.globalwebindex.net/hubfs/Reports/GWI_Device_-_Q1_2016_-_Summary.pdf>. Luettu 27.3.2016.

13 Älli, Sami; Kara, Henna. 2009. Saavutettavuus verkkopalveluissa. Verkkodokumentti. Papunet. <http://papunet.net/sites/papunet.net/files/saavutettavuus/tiedostot/saavutettavuus_opas_2009.pdf>. Luettu 27.3.2016.

14 Visual impairment and blindness. 2014. Verkkodokumentti. World Health Organization. <http://www.who.int/mediacentre/factsheets/fs282/en/>. Luettu 27.3.2016.

15 Saarelma, Osmo. 2015. Värisokeus ja poikkeava värinäkö. Verkkodokumentti. Terveyskirjasto. <http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00347>. Päivitetty 21.9.2015. Luettu 27.3.2016.

16 Types of Colour Blindness. Verkkodokumentti. Colour Blind Awareness. <http://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/>. Luettu 27.3.2016.

17 WHO global estimates on prevalence of hearing loss. 2012. Verkkodokumentti. World Health Organization. <http://www.who.int/pbd/deafness/WHO_GE_HL.pdf>. Luettu 27.3.2016.

18 European Accessibility Act, improving the accessibility of products and services in the single market. Verkkodokumentti. European Commission. <http://ec.europa.eu/social/BlobServlet?docId=14869&langId=en>. Luettu 27.3.2016.

19 Accessible Technology in Computing - Examining Awareness, Use, and Future Potential. 2004. Verkkodokumentti. Forrester Research Inc., Microsoft. <http://download.microsoft.com/download/0/1/f/01f506eb-2d1e-42a6-bc7b-1f33d25fd40f/ResearchReport-Phase2.doc>. Luettu 27.3.2016.

20 Flash Eurobarometer 345, Accessibility. 2012. Verkkodokumentti. European Commission. <http://ec.europa.eu/public_opinion/flash/fl_345_sum_en.pdf>. Luettu 27.3.2016.

21 Perustuslaki. 731/11.6.1999.

22 Virkamäki, Markku. 2015. Eduskunta hyväksyi YK-sopimuksen, mutta ei sitä vielä ratifioinut. Verkkodokumentti. Kehitysvammaisten palvelusäätiö. <http://www.kvps.fi/blogit/toiminnanjohtajan-blogi/eduskunta-hyvaksyi-yk-sopimuksen-mutta-ei-viela-ratifioinut>. Päivitetty 5.3.2015. Luettu 27.3.2016.

23 Section 508 of the Rehabilitation Act. Verkkodokumentti. SSB BART Group. <http://www.ssbbartgroup.com/reference/laws-and-standards/section-508-of-the-rehabilitation-act/>. Luettu 27.3.2016.

24 About the Telecommunications Act Section 255 Guidelines. Verkkodokumentti. United States Access Board. <https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-telecommunications-act-guidelines>. Luettu 27.3.2016.

25 Equality Act 2010: guidance. 2015. Verkkodokumentti. Government Equalities Office. <https://www.gov.uk/guidance/equality-act-2010-guidance>. Julkaistu 27.2.2013. Päivitetty 16.6.2015. Luettu 27.3.2016.

26 European Union Accessibility Requirements. Verkkodokumentti. SSB BART Group. <http://www.ssbbartgroup.com/reference/laws-and-standards/european-union-accessibility-requirements/>. Luettu 27.3.2016.

27 Valtiovarainministeriö. Verkkopalvelujen laatukriteeristö. 2012. Verkkodokumentti. Suomi.fi. <http://www.suomi.fi/suomifi/tyohuone/laatua_verkkoon/laatukriteeristo/uusi_kriteeristo/Verkkopalvelujen_laatukriteerist_4a_2012.pdf>. Luettu 27.3.2016.

28 Henry, Shawn Lawton; Arch, Andrew. 2012. Financial Factors in Developing a Web Accessibility Business Case for Your Organization. Verkkodokumentti. W3C Web Accessibility Initiative. <https://www.w3.org/WAI/bcase/fin>. Julkaistu 7.9.2005. Päivitetty 7.10.2012. Luettu 27.3.2016.

29 Henry, Shawn Lawton. 2012. Web Content Accessibility Guidelines (WCAG) Overview. Verkkodokumentti. W3C Web Accessibility Initiative. <https://www.w3.org/WAI/intro/wcag>. Julkaistu heinäkuussa 2005. Päivitetty 2.10.2012. Luettu 27.3.2016.

30 Myths of Web Accessibility. Verkkodokumentti. Pennsylvania State University. <http://accessibility.psu.edu/accommodations/myths/>. Luettu 27.3.2016.

31 Which Type of Red-Green Color Blindness is It? 2007. Verkkodokumentti. Colblindor. <http://www.color-blindness.com/2007/04/23/which-type-of-red-green-color-blindness-is-it/>. Päivitetty 23.4.2007. Luettu 27.3.2016.

32 Hale, David ym. 2016. Accessible text requirements. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/accessibility/accessible-text-requirements>. Päivitetty 18.3.2016. Luettu 27.3.2016.

33 Eduskunnan esteettömyystyöryhmä. 2006. Eduskunnan esteettömyys- ja saavutettavuusselvitys. Verkkodokumentti. Invalidiliitto. <http://www.invalidiliitto.fi/files/attachments/eduskuknnan_raportti.pdf>. Luettu 27.3.2016.

34 Accessibility testing. 2014. Verkkodokumentti. W3C. <https://www.w3.org/wiki/Accessibility_testing>. Julkaistu 30.6.2011. Päivitetty 14.3.2014. Luettu 27.3.2016.

35 Hale, David; Groce, Jason. 2016. What's a Universal Windows Platform (UWP) app? Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/get-started/whats-a-uwp>. Päivitetty 1.3.2016. Luettu 27.3.2016.

36 Hale, David. 2016. Guide to Universal Windows Platform (UWP) apps. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/get-started/universal-application-platform-guide>. Päivitetty 28.2.2016. Luettu 27.3.2016.

37 Jacobs, Mike. 2016. Introduction to UWP app design. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/Windows/uwp/layout/design-and-ui-intro>. Päivitetty 1.3.2016. Luettu 27.3.2016.

38 Hale, David; Groce, Jason. 2016. Get set up. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/get-started/get-set-up> Päivitetty 16.2.2016. Luettu 27.3.2016.

39 Welcome to Continuum for phone. Verkkodokumentti. Microsoft. <http://Windows.microsoft.com/en-us/Windows-10/getstarted-continuum-mobile>. Luettu 28.3.2016.

40 Ng, Simon. 2013. iOS Programming Tutorial: Creating a Universal App. Verkkodokumentti. AppCoda. http://www.appcoda.com/ios-univeral-app-tutorial/ Päivitetty 8.11.2013. Luettu 11.4.2016.

41 Broussard, Mitchel. 2015. App Store on iOS Begins Identifying Apps With Universal Support for Apple TV. Verkkodokumentti. MacRumors. http://www.macrumors.com/2015/12/11/ios-app-store-apple-tv-id/ Päivitetty 11.12.2015. Luettu 11.4.2016.

42 Desktop Operating System Market Share. 2016. Verkkodokumentti. NetMarketShare. <https://www.netmarketshare.com/operating-system-market-share.aspx> Luettu 28.3.2016.

43 Nixon, Jerry. 2015. Tiles, Notifications, and Action Center. Microsoft Ignite. Saatavissa: <https://channel9.msdn.com/Events/Ignite/2015/BRK2352>.

44 Smartphone OS Market Share, 2015 Q2. Verkkodokumentti. IDC. <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> Luettu 28.3.2016.

45 Smith, Josh. 2009. Patterns - WPF Apps With The Model-View-ViewModel Design Pattern. Verkkodokumentti. <https://msdn.microsoft.com/en-us/magazine/dd419663.aspx>. Luettu 28.3.2016.

46 Jacobs, Mike; Groce, Jason. 2016. Screen sizes and break points for responsive design. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/Windows/uwp/layout/screen-sizes-and-breakpoints-for-responsive-design>. Päivitetty 7.3.2016. Luettu 28.3.2016.

47 Jacobs, Mike. 2016. Layout Panels. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/Windows/uwp/layout/layout-panels>. Päivitetty 28.2.2016. Luettu 28.3.2016.

48 Hale, David. 2016. Guidelines for tile and icon assets. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/controls-and-patterns/tiles-and-notifications-app-assets>. Päivitetty 7.3.2016. Luettu 28.3.2016.

49 Jacobs, Mike; Groce, Jason. 2016. ResourceDictionary and XAML resource references. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/controls-and-patterns/resourcedictionary-and-xaml-resource-references>. Päivitetty 7.3.2016. Luettu 28.3.2016.

50 Hale, David ym. Accessibility checklist. 2016. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/accessibility/accessibility-checklist>. Päivitetty 18.3.2016. Luettu 28.3.2016.

51 Abad, Sandra Aldana; Groce, Jason. 2016. Deploying and debugging Universal Windows Platform (UWP) apps. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/debug-test-perf/deploying-and-debugging-uwp-apps>. Päivitetty 19.2.2016. Luettu 28.3.2016.

52 Garrett, Jesse James. 2011.The Elements of User Experience - User-Centered Design for the Web and Beyond. Second Edition. Berkeley: New Riders.

53 Screen Reader User Survey #6 Results. 2015. Verkkodokumentti. WebAIM. <http://webaim.org/projects/screenreadersurvey6>. Luettu 28.3.2016.

54 Abad, Sandra Aldana. 2016. Windows App Certification Kit. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/debug-test-perf/Windows-app-certification-kit>. Päivitetty 16.2.2016. Luettu 28.3.2016.

55 Halde, David; Groce, Jason. 2016. Packaging UWP Apps. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/packaging/packaging-uwp-apps>. Päivitetty 23.2.2016. Luettu 28.3.2016.

56 Frank, Jill. 2016. App submissions. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/publish/app-submissions>. Päivitetty 22.2.2016. Luettu 28.3.2016.

57 Frank, Jill. 2016. The app certification process. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/publish/the-app-certification-process>. Päivitetty 22.2.2016. Luettu 28.3.2016.

58 Frank, Jill. 2016. Notes for certification. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/publish/notes-for-certification>. Päivitetty 22.2.2016. Luettu 28.3.2016.

59 Frank, Jill. 2016. Beta testing and targeted distribution. Verkkodokumentti. Windows Dev Center. <https://msdn.microsoft.com/en-us/Windows/uwp/publish/beta-testing-and-targeted-distribution>. Päivitetty 29.2.2016. Luettu 28.3.2016.

60 Forecast revenue for virtual reality products* worldwide from 2014 to 2018 (in million U.S. dollars). 2016. Verkkodokumentti. Statista. <http://www.statista.com/statistics/426276/virtual-realiy-revenue-forecast-worldwide/> Luettu 28.3.2016.