Low-code -menetelmä saavutettavien sovellusten toteuttamisessa – Tapaus Power Apps

Mariia Pyylampi

Haaga-Helia ammattikorkeakoulu
Tradenomi, Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
2024

Tiivistelmä

Tekijä(t)
Mariia Pyylampi

Tutkinto
Tradenomi, tietojenkäsittely

Raportin/Opinnäytetyön nimi
Low-code -menetelmä saavutettavien sovellusten toteuttamisessa – Tapaus Power Apps

Sivu- ja liitesivumäärä
35 + 24

Low-code menetelmä on yksi kehityssuuntaus kohti ketterämpää ja kustannustehokkaampaa sovelluskehitystä. Menetelmä on noussut vastaamaan vallitsevaan kehittäjäpulaan ja digitaalisen siirtymän myötä kasvaneeseen tarpeeseen räätälöidyille sovelluksille. Samaan aikaan lainsäädäntö ohjaa organisaatioita sekä yrityksiä noudattamaan saavutettavuusvaatimuksia ja tekemään digitaalisista palveluista saavutettavia erilaisille käyttäjille.

Tämän opinnäytetyön tavoitteena oli tarkastella low-code menetelmää sekä luoda käsitys, miten hyvin saavutettavuusvaatimukset on huomioitu low-code menetelmässä. Opinnäytetyössä tarkasteltiin Microsoftin Power Apps -sovelluskehitysympäristöä tapausesimerkkinä low-code sovelluskehitysympäristöstä. Tutkimuksessa tarkasteltiin Power Apps -sovelluskehitysympäristön valmiita käyttöliittymäkomponentteja sekä arvioitiin, miten hyvin ne täyttivät Web Content Accessibility Guidelines (WCAG) -ohjeistuksen onnistumiskriteerit. Opinnäytetyön aineisto koostui Power Apps -sovelluskehitysympäristöstä tehdyistä havainnoista, jotka kerättiin tarkastelemalla ympäristön käyttöliittymäkomponenttien saavutettavuusominaisuuksia sekä testaamalla käyttöliittymäkomponentteja manuaalisesti avustavaa teknologiaa hyödyntäen.

Tutkimuksen avulla muodostui kattava kuva Power Apps -sovelluskehitysympäristöstä ja sen ominaisuuksista sekä saavutettavuusnäkökulman huomioimisesta sovelluskehitysympäristössä. Tulokset osoittivat, että Power Apps -sovelluskehitysympäristössä saavutettavuus oli huomioitu käyttöliittymäkomponenttien ominaisuuksissa pääsääntöisesti hyvin, ja että Power Apps -sovelluskehitysympäristön käyttöliittymäkomponentteja hyödyntämällä on mahdollista toteuttaa yksinkertaisia sovelluksia saavutettavasti.

Tutkimus tarjosi uutta tietoa Power Apps -sovelluskehitysympäristöstä, mutta sen tulokset eivät ole yleistettävissä low-code menetelmään tai muihin low-code sovelluskehitysympäristöihin, sillä tutkimus keskittyi vain yhteen low-code sovelluskehitysympäristöön ja sen ominaisuuksiin. Tutkimus osoitti sen sijaan tarpeen low-code sovelluskehitysympäristöjen saavutettavuutta koskevalle jatkotutkimukselle.

Asiasanat
Low-code, sovelluskehitysmenetelmä, saavutettavuus, WCAG-ohjeistus, saavutettavuusvaatimukset

Sisällys

1 Johdanto

Digitaalinen siirtymä on lisännyt sovellusten kysyntää ja räätälöidyille sovelluksille on jatkuvasti kasvava tarve (Woo 2022, 961). Samanaikaisesti kehittäjäpula rajoittaa sovelluskehitysprojektien saatavilla olevia resursseja. Low-code -menetelmä on noussut vastaamaan tähän haasteeseen. Kyseessä on kehityssuuntaus, jonka tavoitteena on vähentää perinteisen ohjelmakoodin kirjoittamista sekä lisätä koodin uudelleen käytettävyyttä, ja sitä kautta lisätä sovelluskehityksen tuottavuutta. (Garcia 2021.) Isot teknologiajätit ovat tuoneet markkinoille low-code tuotteitaan, joiden myötä low-code on kasvanut merkittäväksi osaksi sovelluskehityskenttää ja –markkinoita. (Matvitskyy ym. 2023.)

Saavutettavuusvaatimuksista säädetään digitaalisten palvelujen tarjoamisesta annetussa laissa (2019/306), jonka yhtenä tavoitteena on digitaalisten palveluiden saavutettavuuden parantaminen. Digipalvelulain tarkoituksena on turvata digitaaliset palvelut yhdenvertaisesti kaikille käyttäjille (laki digitaalisten palvelujen tarjoamisesta 15.3.2019/306, 1 §). Digipalvelulaki velvoittaa palveluntarjoajia, huomioimaan saavutettavuusvaatimukset digitaalisissa palveluissa (laki digitaalisten palvelujen tarjoamisesta, 7 §).

Vaikka low-code -menetelmän suosio kasvaa jatkuvasti, on aiheesta saatavilla niukasti tutkimustietoa. Aiemmissa tutkimuksissa (luku 2) esimerkiksi Matvitskyy ja muut (2023) sekä Bock ja Frank (2021) ovat pyrkineet löytämään low-code menetelmälle selkeää määritelmää etsimällä eri palveluntarjoajien low-code -sovelluskehitysympäristöistä yhteisiä ominaisuuksia. Aiemmissa tutkimuksissa esimerkiksi Pinho, Aguiar ja Amaral (2023) ovat tarkastelleet low-code -sovelluskehitysalustojen käytettävyyttä sekä yleisimpiä käyttötapauksia. Toistaiseksi tutkimustietoa ei ole kuitenkaan saatavilla low-code -sovelluskehitysympäristöjen saavutettavuudesta eikä low-code -sovelluskehitysympäristöjen käyttöliittymäkomponenttien saavutettavuusominaisuuksista. Myöskään tutkimustietoa siitä, mitä vaikutuksia käyttöliittymäkomponenttien saavutettavuusominaisuuksilla on sovellusten saavutettavuuteen, ei toistaiseksi ole saatavilla. Tämän opinnäytetyön tarkoituksena on vastata tähän tutkimusaukkoon.

1.1 Opinnäytetyön tarkoitus ja tavoitteet

Teknologioiden ominaisuudet ovat tärkeässä roolissa, kun tehdään teknologiavalintoja sovellusprojektiin. Low-code-sovelluskehitysympäristöjen ominaisuudet sekä valmiudet saavutettavien käyttöliittymäkomponenttien toteuttamiseen, ovat teknologiavalintojen kannalta kiinnostavia. Digipalvelulain vaatimukset koskevat laajalti erilaisia palveluntarjoajia, ja sovellusprojekteja tilaavat asiakkaat ovat tietoisempia saavutettavuuden merkityksestä. Tämä lisää asiakkaiden suunnalta

saavutettavuusvaatimuksia sovellusprojekteille, joka vaikuttaa sekä teknologiavalintoihin että projektitiimiltä edellytettäviin taitoihin.

Tämän opinnäytetyön tavoitteena on tarkastella low-code -menetelmää sekä sitä, miten hyvin low-code-sovelluskehitysympäristöissä on otettu huomioon saavutettavuuden periaatteet. Tavoitteena on arvioida low-code sovelluskehityksen soveltuvuutta menetelmänä saavutettavien sovellusten kehitykseen.

Opinnäytetyössä pyritään vastamaan kahteen tutkimuskysymykseen:

1. Soveltuuko low-code -menetelmä teknologiana saavutettavien palveluiden toteuttamiseen?
2. Onko Power Apps -sovelluskehitysalustalla mahdollista toteuttaa saavutettavia sovelluksia?

Opinnäytetyössä tarkastellaan esimerkkinä low-code -sovelluskehitysympäristöstä Microsoftin Power Apps -sovelluskehitysympäristöä, joka on yksi tunnetuimmista low-code sovelluskehitysympäristöistä. Tutkimuksessa tarkastellaan Power Apps -sovelluskehitysympäristön ominaisuuksia, jotka tukevat saavutettavien käyttöliittymäkomponenttien toteuttamista. Tämän lisäksi kartoitetaan, millaisia työkaluja Power Apps tarjoaa saavutettavuuden testaamiseen sekä tarkastellaan Power Apps -sovelluskehitysympäristön saavutettavuusohjeistuksien sekä dokumentaation kattavuutta.

Tavoitteena on luoda käsitys siitä, miten Power Apps -sovelluskehitysympäristössä on huomioitu saavutettavuusvaatimusten mukaisten käyttöliittymäkomponenttien toteuttaminen, ja millaisia mahdollisuuksia Power Apps -sovelluskehitysympäristö tarjoaa saavutettavien sovellusten toteuttamiseen. Tavoitteena on arvioida Power Apps sovelluskehitysympäristön soveltuvuutta saavutettavuusvaatimukset huomioiviin sovelluskehitysprojekteihin.

Tutkimuksen tulokset tarjoavat tietoa Power Apps -sovelluskehitysympäristön saavutettavuusominaisuuksista, sekä tarjoavat tietoa saavutettavuuden huomioimisesta low-code sovelluskehitysympäristöissä. Tutkimuksen tuloksia voidaan myös hyödyntää päätöksenteossa, kun valitaan toteutusteknologioita sovelluskehitysprojektiin. Lisäksi opinnäytetyön tavoitteena on edistää opinnäytetyöntekijän ammatillista kehitystä, lisätä opinnäytetyöntekijän tietoa low-code sovelluskehityksestä sekä Power Apps -sovelluskehitysympäristöstä, ja syventää opinnäytetyöntekijän ymmärrystä saavutettavuudesta sekä saavutettavuuden testaamisesta ja arvioinnista.

1.2 Opinnäytetyön eteneminen

Opinnäytetyön tietoperustan ensimmäisessä osiossa käsitellään low-code -menetelmää. Aluksi tarkastellaan low-code menetelmän historiaa ja avataan keskeisimmät low-code menetelmään liittyvät käsitteet. Tämän jälkeen tarkastellaan low-code -menetelmälle tyypillisiä piirteitä ja ominaisuuksia

sekä tunnistettuja hyötyjä ja haasteita. Lisäksi avataan low-code -menetelmän tyypillisimmät käyttötapaukset ja luodaan katsaus low-code -menetelmän tulevaisuudennäkymiin.

Tietoperustan toisessa osiossa tarkastellaan saavutettavuutta. Aluksi avataan, mitä saavutettavuus tarkoittaa, miksi se on tärkeää, ja mitä saavutettavien digitaalisten palveluiden suunnittelussa tulee ottaa huomioon. Tämän jälkeen tarkastellaan saavutettavuutta koskevaa lainsäädäntöä sekä syvennytään WCAG-saavutettavuusohjeistukseen, johon lainsäädäntö pitkälti pohjautuu. Lopuksi tietoperustassa perehdytään saavutettavuuden automaattiseen sekä manuaaliseen testaukseen.

Empiirinen osuus koostuu opinnäytetyön tutkimusmenetelmän esittelystä sekä tutkimusaineiston analyysistä. Opinnäytetyön tutkimusstrategiana käytettiin tapaustutkimusta ja tutkimuksen aineisto koottiin havainnoimalla ja testaamalla tutkimuksen kohteena olevaa Microsoftin Power Apps -sovelluskehitysympäristöä. Empiirisessä osuudessa esitellään aluksi tutkimuksen kohteena oleva Power Apps -sovelluskehitysympäristö sekä käydään läpi tutkimukseen liittyvät rajaukset. Tämän jälkeen avataan tarkemmin, miten tutkimusaineisto on koottu.

Opinnäytetyön lopussa käydään läpi, miten kerätty aineisto valmisteltiin analysointia varten, ja miten aineiston analyysi toteutettiin. Tämän jälkeen esitellään keskeiset aineistosta tehdyt löydökset ja esitellään tärkeimmät havainnot. Lopuksi esitetään johtopäätökset sekä tarkastellaan, miten tutkimus onnistui tavoitteiden saavuttamisessa. Tämän jälkeen avataan tutkimuksen tekoon liittyvät haasteet, otetaan kantaa tutkimuksen luotettavuuteen sekä tulosten hyödynnettävyyteen ja esitetään ehdotuksia jatkotutkimusaiheiksi. Lopuksi arvioidaan, opinnäytetyötä kokonaisuutena sekä sen onnistumista.

2 Low-code -menetelmä

Low-code -menetelmä tarkoittaa lähestymistapaa, jossa ohjelmistokehityksessä ei kirjoiteta tai kirjoitetaan vain hyvin vähän koodia tekstillisillä ohjelmistokielillä, kuten Java, Python tai C. Low-code -menetelmässä on erilaisia lähestymistapoja ohjelmointiin, kuten ohjelmointia visuaalisten elementtien tai luonnollisen kielen avulla, joiden myötä tarve opetella perinteisiä ohjelmistokieliä vähenee. Tällaiset lähestymistavat vaativat vähemmän teknistä osaamista, mikä tekee ohjelmoinnista helpommin lähestyttävää. Low-code -menetelmän käyttäjät vaihtelevat ammattimaisista ohjelmistokehittäjistä käyttäjiin, joilla ei ole ohjelmointikoulutusta tai teknistä osaamista. (Hirzel 2023, 76–77.)

Low-code on terminä melko uusi, vaikka menetelmänä se on ollut olemassa pitkään. Bock ja Frank (2021, 60–61) nostivat artikkelissaan esille, että monet low-code -alustojentarjoajista ovat olleet markkinoilla pitkään, mutta ovat tarjonneet palveluitaan muilla termeillä kuin low-code. Heidän mukaansa vähäkoodisia kehitysalustoja ja työkaluja on ollut tarjolla muun muassa nimillä Rapid Application Development (nopea sovelluskehitys), Platform as a Service (alusta palveluna) tai Software as a Service (ohjelmisto palveluna). Nykyään vähäkoodisia kehitysalustoja tarjotaan pääosin low-code -termillä ja monet palveluntarjoajista ovat asemoineet itseään markkinoilla uudelleen kasvavan low-code trendin myötä. (Bock & Frank 2021, 60–61).

Hirzelin (2023, 76) mukaan usein low-code -menetelmästä puhuttaessa esille nousee myös kaksi muuta menetelmää, jotka liittyvät läheisesti low-code menetelmään- Näitä ovat no-code ja end-user programming. No-code -menetelmä eroaa low-code -menetelmästä siten, että no-code -menetelmässä ei kirjoiteta lainkaan koodia. End-user programming eli loppukäyttäjäohjelmointi -käsitteellä puolestaan viitataan siihen, että ohjelmistoja kehittävät henkilöt, joka ovat myös valmiin ohjelmiston tulevia käyttäjiä. End-user programming -termillä viitataan ohjelmoinnin suorittavaan tekijään, kun taas low-code ja no-code viittaavat käytettyyn ohjelmointitekniikkaan. End-user programming on tunnetumpi akateemisessa kirjallisuudessa kuin low-code tai no-code. (Hirzel 2023, 76–77.)

Seuraavissa luvuissa käsitellään tarkemmin low-code -menetelmään liittyviä keskeisiä tekniikoita ja tyypillisiä ominaisuuksia, ja tarkastellaan hyötyjä ja haasteita, joita low-code menetelmän käyttöön liittyy. Lisäksi tarkastellaan low-code -menetelmään liittyviä uhkia ja niiden vaikutuksia. Seuraavissa luvuissa käsitellään myös erilaisia low-code -menetelmän käyttötapauksia, ja lopuksi tarkastellaan low-code -menetelmän tulevaisuudennäkymiä ja kehityssuuntia.

2.1 Low-code -tekniikat ja ominaisuudet

Low-code -alustat perustuvat erilaisiin ohjelmointitekniikoihin. Visuaaliseen ohjelmointiin perustuvilla alustoilla käyttäjä toteuttaa ohjelmistoa raahaamalla ja muokkaamalla visuaalisia komponentteja, ja liittämällä niitä toisiinsa. Visuaalisiin ohjelmointikieliin perustuvat alustat sisältävät usein pohjan, jossa käyttäjä voi sekä lukea että kirjoittaa ohjelmaa. Ohjelmointitekniikassa, joka perustuu käyttäjän suorittamaan esitykseen, käyttäjä suorittaa jonkin tehtävän low-code -alustalla ja alusta nauhoittaa suorituksen. Tämän jälkeen alusta tekee ohjelman, joka suorittaa saman tehtävän automaattisesti. Luonnolliseen kieleen perustuvissa low-code -alustoissa käyttäjä syöttää tekstiä luonnollisella kielellä ja se käännetään alustassa ohjelmaksi. Luonnolliseen kieleen perustuvassa tekniikassa virheiden määrä on usein suuri, sillä luonnollinen kieli on moniselitteinen. (Hirzel 2023, 79–81.)

Erilaisista ohjelmointitekniikoista huolimatta, low-code -alustoilla on paljon yhteisiä ominaisuuksia. Bock ja Frank (2021, 62) listasivat artikkelissaan tyypillisiä ominaisuuksia, joita on useilla low-code -alustoilla. Heidän mukaansa useat alustat tarjoavat mahdollisuuden lisätä ulkoisen tietolähteen käytettävän datan hakua varten ja komponentin, jonka avulla sovelluksessa käytettävän datan voi mallintaa. Heidän mukaansa toinen yleinen piirre on ”raahaa ja pudota” -ominaisuus, jolla käyttöliittymäkomponentteja otetaan käyttöön ennalta määritellystä valikosta. Elementit voivat olla peruselementtejä, kuten tekstikentät ja painikkeet, tai interaktiivisia kuten kaaviot tai galleriat. Heidän mukaansa myös toiminnallisten määrityksien lisääminen ennalta määritellyillä matemaattisilla funktioilla tai kirjoittamalla erilaisia päätöksentekosääntöjä, integroiminen ulkoisiin palveluihin API-rajapintojen kautta sekä käyttöoikeuksien hallintatyökalut, ovat ominaisuuksia, joita usein low-code alustoilta löytyy. (Bock ja Frank 2021, 62–63.) Pinho, Aguiar ja Amaral (2023) puolestaan nostivat artikkelissaan esille, että low-code -alustoille yhteisiä piirteitä ovat alustojen korkea abstraktion taso ja taustalla olevat mallit, pilvipohjaiset palvelut, nopea sovelluskehitys sekä elinkaarenhallinnan työkalut. Pinhon, Aguiarin ja Amaralin (2023) mukaan on myös tyypillistä, että low-code -menetelmän käyttäjällä ei ole aikaisempaa ohjelmointitaustaa.

Tyypillisimpien ominaisuuksien lisäksi Bock ja Frank (2021, 63–64) listasivat ominaisuuksia, joita low-code alustoilla usein on, mutta jotka eivät ole yhtä yleisiä. Heidän mukaansa näitä ovat esimerkiksi työnkulujen suunnittelutyökalu, tapahtumien hallintaan liittyvät skriptit, perinteisten ohjelmointikielien käyttö sekä erilaiset tekoälyavustajat ja automaatiotyökalut.

2.2 Low-code -menetelmän hyödyt ja haasteet

Pinhon, Aguiarin ja Amaralin (2023) mukaan keskeisimmät low-code -menetelmän hyödyt ovat vähäisempi teknisen osaamisen tarve verrattuna perinteiseen ohjelmointiin, sekä se, että se on

helpommin opeteltavissa kuin perinteinen ohjelmointi. Heidän mukaansa muita keskeisiä hyötyjä ovat matalampi resurssien ja ajan tarve, nopeampi ja tehokkaampi kehitys verrattuna perinteiseen ohjelmistokehitykseen, sekä low-code menetelmien korkea laatu ja vähäinen ylläpidon tarve.

Bockin ja Frankin (2021, 64) mukaan tuottavuus syntyy rutiinitehtävien vähentämisestä sekä low-code alustojen valmiiden integraatioiden hyödyntämisestä. Heidän mukaansa korkea tuottavuus ei kuitenkaan toteudu kaikilla low-code -alustoilla. Bock ja Frank (2021, 64) nostivat artikkelissaan esille, että klassisessa ohjelmistokehityksessä käytetään useita työkaluja kuten koodieditoria, mallinnustyökaluja, käyttöliittymätyökaluja ja kolmansien osapuolien kirjastoja sekä koodinkääntämiseen ja julkaisuun liittyviä työkaluja. Heidän mukaansa yksi tärkeimmistä hyödyistä on, että low-code -alustat kokoavat nämä työkalut samaan paikkaan, mikä vähentää huomattavasti ylläpidon sekä integraatioiden tarvetta.

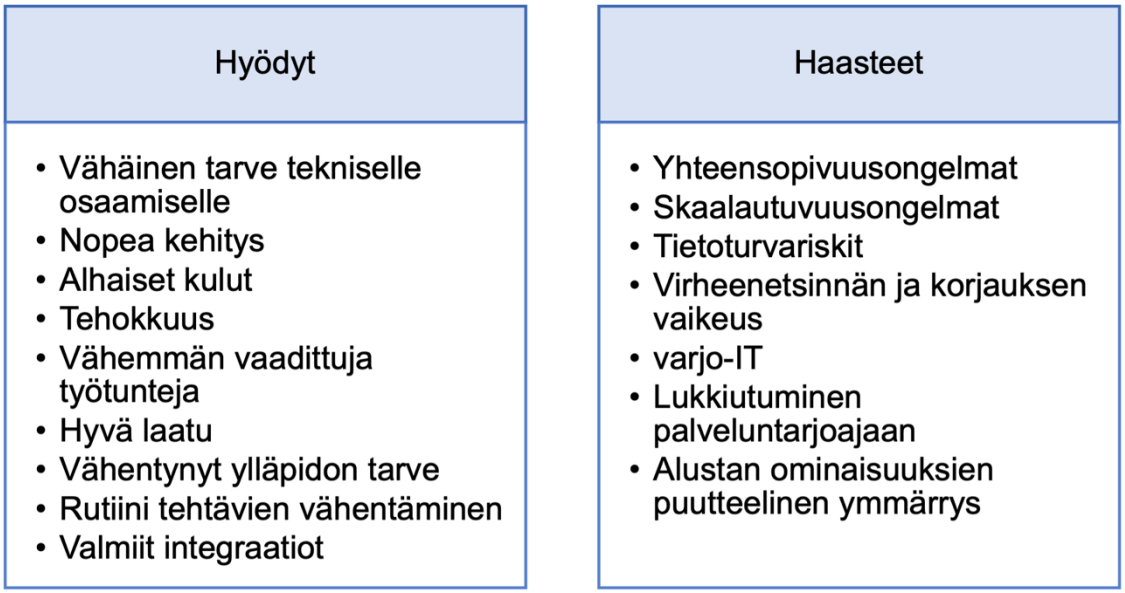
Low-code -menetelmässä on hyötyjen lisäksi tunnistettu haasteita ja uhkia. Pinhon, Aguiarin ja Amaralin (2023) mukaan yleisimpiä haasteita ovat yhteensopivuus- ja skaalautuvuusongelmat sekä lukkiutuminen palveluntarjoajan palveluun. Heidän mukaansa usein päädytään kirjoittamaan enemmän koodia kuin alun perin oli tarkoituksena, mikä voi heidän mukaansa johtua esimerkiksi alustan ominaisuuksien puutteellisesta ymmärryksestä tai siitä, että alustan ominaisuudet muuttuvat usein, jonka vuoksi joudutaan tekemään muutoksia. Lisäksi tunnistettuja haasteita olivat virheiden etsinnän hankaluus korkean abstraktion vuoksi ja siitä johtuva ajan menetys ja tuottavuuden lasku sekä käyttötarkoitukseen sopivan alustan valinta laajan palveluntarjonnan vuoksi. (Pinho, Aguiar & Amaral 2023.)

Low-code -menetelmässä on myös tunnistettu tietoturvaan liittyviä riskejä. Hughes (2022) huomautti artikkelissaan, että low-code -alustat ovat itsessään myös ohjelmistoja, jonka lähdekoodiin alustan käyttäjillä ei ole näkyvyyttä. Hänen mukaansa low-code -alustojen käyttäjät eivät voi tietää, millaisia tietoturvaan liittyviä riskejä alustan ohjelmistossa on tai miten hyvin alustan tietoturvallisuutta on testattu, ennen kuin se on julkaistu markkinoille.

Hughesin (2022) nosti artikkelissaan esille, että niin kutsuttu shadow IT, eli varjo-IT, on myös yksi low-code -menetelmään liitetyistä ongelmista. Varjo-IT tarkoittaa, että yritysten sisällä luodaan epävirallisia työkaluja jonkin sisäisen prosessin sujuvoittamiseen. Hughesin (2022) mukaan usein epävirallisia työkalujen kehityksessä käytetään low-code -menetelmää ja työkaluja kehittävät sellaiset henkilöt, joilla ei ole teknistä taustaa tai ohjelmointiosaamista, joka voi johtaa työkalujen tietoturvahaavoittuvuuksiin. Hänen mukaansa epävirallisissa työkaluissa käsitellään usein organisaation sisäistä dataa tai jopa asiakkaiden dataa, jonka joutuminen ulkopuolisten käsiin on tietoturvariski. Lisäksi ohjelmistot ja sovellukset, jotka ovat toteutettu low-code alustalla SaaS-palveluna (Software

as a Service) saattavat ovat alttiita palveluntarjoajan käyttökatkoksille, ja palveluntarjoajaan kohdistuville uhille ja hyökkäyksille. (Hughes 2022.)

Kuten kuvasta 1 voidaan havaita, low-code -menetelmän tunnistetut hyödyt liittyvät pitkälti vähentyneisiin kuluihin ja ajankäyttöön sekä lisääntyneeseen tehokkuuteen. Low-code menetelmästä tunnistetut haasteet sen sijaan liittyvät low-code alustan teknisiin ominaisuuksiin, kuten tietoturvaan tai skaalautuvuuteen.



Kuva 1. Low-code menetelmän hyödyt ja haasteet (mukaillen Bock & Frank 2021, 64; Pinho, Aguiar & Amaral 2023; Hughes 2022)

2.3 Käyttötapaukset

Matvitskyy ja muut (2023) esittivät tutkimuksessaan low-code -menetelmän keskeisiksi käyttötapauksiksi sisäisten liiketoiminnallisten ohjelmistojen kehittämisen sekä sisäisten liiketoiminnallisten prosessien automatisoinnin. Heidän mukaansa low-code on yksi tärkeä tekijä sille, miksi automatisointi liiketoiminnan prosesseissa on tällä hetkellä suosittua. Monelle liiketoiminnalliselle tarpeelle löytyy jokin low-code -palveluntarjoaja, joka tarjoaa alustan liiketoiminnallisten ohjelmistojen kehittämiseen. (Matvitskyy ym. 2023.) Myös Garcian (2021) mukaan yksi suosituimpia low-code -käyttötapauksia on yritysten sisäisten työkalujen toteuttaminen. Hänen mukaansa low-code tekee sisäisten työkalujen toteuttamisesta helpompaa, joka säästää yritysten aikaa resursseja.

Matvitskyy ja muiden (2023) mukaan yksi suosittu käyttötapaus low-code menetelmälle on vanhojen liiketoiminnallisten ohjelmistojen modernisointi. Garcia (2021) nosti blogissaan esille, että vanhat järjestelmät ovat hitaita ja monimutkaisia, ja että low-code menetelmän avulla vanhoja järjestelmiä ei välttämättä tarvitse purkaa kokonaan ja rakentaa uudelleen. Hänen mukaansa low-code -

työkalujen avulla voidaan korvata vain joitain järjestelmän osia tai luoda järjestelmän päälle intuitiivisempia käyttöliittymiä. Garcian (2021) mukaan low-code -menetelmää käytetäänkin usein käyttöliittyminen suunniteluun ja toteutukseen. Low-code -menetelmä tekee käyttöliittymän kehittämisestä helpommin lähestyttävää myös suunnittelijoille, joilla ei välttämättä ole teknistä osaamista tai sellaisille kehittäjille, jotka yleensä keskittyvät vain back-end puolen ohjelmointiin. (Garcia 2021).

Prototyyppien kehittäminen on myös keskeinen käyttötapaus low-code -menetelmille. Garcian (2021) mukaan ajatuksena on, että low-code alustaa hyödyntäen saadaan prototyyppi mahdollisimman nopeasti valmiiksi, ilman, että joudutaan käyttämään aikaa ja resursseja mahdollisen taustalla toimivan infrastruktuurin luomiseksi. Hänen mukaansa low-code menetelmän avulla prototyyppi voidaan saada mahdollisimman aikaisessa vaiheessa testattavaksi asiakkaille ja käyttäjille, jolloin saadun palautteen pohjalta voidaan varmistaa, että kehitys on menossa oikeaan suuntaan ja ollaan tekemässä ominaisuuksia, mitä asiakkaat ja käyttäjät todellisuudessa tarvitsevat.

2.4 Low-code -menetelmän tulevaisuus

Low-code on kasvamassa merkittäväksi osaksi ohjelmistokehitysmarkkinoita. Matvitskyy ja kumppanit (2023) ennustavat low-code markkinan kasvavan yli 20 prosenttia vuosittain 2021–2026 välillä. Heidän mukaansa low-code menetelmän lisääntyneeseen suosioon on vaikuttanut yritysten aktiivisuus low-code alustojen käyttöönotossa, lisääntynyt automatisointi liiketoiminnallisissa prosesseissa sekä sovellusten kehittäminen eri tiimien luomista komponenteista. Woon (2022, 960) mukaan low-code ja no-code -menetelmien suosiosta kertoo myös se, että suuret ohjelmistojätit, kuten Microsoft, Salesforce, Oracle ja Alibaba tarjoavat low-code ja no-code -alustoja. Lisäksi monet start-up-yritykset tarjoavat low-code ja no-code -alustoja (Woo 2022, 960).

Digitaalinen siirtymä ja kasvava tarve räätälöidyille sovelluksille, on kääntänyt yritysten katseet low-code -menetelmään (Woo 2022, 961). Kasvava paine digitalisessa siirtymässä pakottaa IT-yrityksiä siirtymään perinteisestä ohjelmistokehityksestä kohti korkeamman abstraktion ohjelmistokehitystä sekä omaksumaan low-code -menetelmät osaksi toimintamalliaan. Low-code -menetelmän avulla voidaan vastata resurssipulaan ja lyhentää aikaa, jota uusien palveluiden ja tuotteiden saaminen markkinoille vie. (Matvitskyy ym. 2023.)

Low-code -menetelmästä voi olla myös apua kasvavaan kehittäjäpulaan. Project Management Instituten (2021, osa 1) mukaan termi citizen developer, eli kansalaiskehittäjä, tarkoittaa käyttäjiä, joilla ei ole vahvaa teknistä taustaa. Kansalaiskehittäjät ovat usein yrityksen työntekijöitä, joilla on vahva substanssiosaaminen, ja joilla on tarve ratkaista jokin liiketoiminnallinen ongelma tai vähentää manuaalista työtä, joita toistuvat tehtävät vaativat. (Project Management Institute 2021, osa 1.)

Niiranen (6.4.2021) kirjoitti blogissaan, että low-code menetelmä on demokratisoinut ohjelmistokehitystä, joka on madaltanut kynnystä kansalaiskehittäjänä osallistua sovellusten kehittämiseen. Hänen mukaansa low-code -menetelmän yleistyessä yhä useampi ohjelmistokehittäjä tulee tekemään entistä vähemmän perinteistä ohjelmointia, ja yhä useampi työskentelee täysipäiväisesti low-code -menetelmän parissa. Hän ottaa kantaa siihen, että erottelu perinteisen ohjelmistokehittäjän ja kansalaiskehittäjän välillä ei ole enää tulevaisuudessa mielekästä, sillä kompleksisten prosessien ratkaiseminen low-code -menetelmää hyödyntäen vaatii kehittyneiden teknologioiden vahvaa osaamista ja ammattitaitoa, huolimatta siitä kirjoitetaanko koodia perinteisessä mielessä vai ei.

3 Saavutettavuus

Saavutettavuus tarkoittaa käsitteenä esteettömyyttä digitaalisessa ympäristössä. Esteettömyys tarkoittaa puolestaan sitä, että fyysisessä ympäristössä on helppo liikkua esimerkiksi pyörätuolilla. Saavutettavuus on vakiintunut termi esteettömyydelle verkkosivuilla ja digitaalisissa palveluissa. Saavutettavuus siis tarkoittaa sitä, että verkkosivut ja digitaaliset palvelut ovat suunniteltu ja toteutettu niin, että ovat helppoja käyttää erilaisille ihmisille. (Aluehallintovirasto s.a. a.)

Aluehallintoviraston mukaan (s.a. b.) Suomessa on arvioitu olevan yli miljoona ihmistä, joilla voi olla haasteita digitaalisten palveluiden käytössä. Kehitysvammaliitto ry:n (2023a) mukaan haasteet digitaalisten palveluiden käytössä johtuvat monista erilaisista asioista. Käyttäjillä voi olla esimerkiksi erilaisia toimintarajoitteita tai vammoja, kuten näkökykyyn ja kuuloon liittyviä rajoitteita, fyysisiä ja motorisia rajoitteita, kognitiivisia vaikeuksia tai neurologisia sairauksia. Toimintarajoitteet voivat olla synnynnäisiä tai ne voivat johtua esimerkiksi tapaturmasta tai korkean liittyä käyttäjän korkeaan ikään. (Kehitysvammaliitto ry 2023a.)

Kalbagin (2017, luku 2) mukaan toimintarajoitteiden lisäksi myös ympäristö voi aiheuttaa hetkellisiä vaikeuksia tai haasteita verkkosivujen käytössä. Hänen mukaansa käytössä olevan päätelaitteen koko ja resoluutio vaikuttavat siihen, miten käyttäjä näkee verkkosivun. Hänen mukaansa myös selain voi vaikuttaa siihen, miltä verkkosivu näyttää sillä verkkosivu saattaa näyttää erilaiselta eri selaimissa. Vanhentuneella selaimella tai huonolla verkkoyhteydellä verkkosivu voi toimia huonosti tai ei välttämättä toimi lainkaan. Ympäristö voi myös vaikuttaa saavutettavuuteen. Esimerkiksi kirkas valo lumisessa ympäristössä voi vaikeuttaa verkkosivun käyttöä, jos verkkosivun kontrasti on liian pieni. Meluisa ympäristö puolestaan voi vaikeuttaa verkkosivulla toistettavan äänen kuulemista, tai se voi vaikeuttaa keskittymistä monimutkaiseen sisältöön. Verkkosivun kielellä on myös merkitystä saavutettavuuden kannalta. Vieras tai monimutkainen kieli voi myös vaikeuttaa verkkosivujen käyttöä, jos ei puhu kieltä äidinkielenään. (Kalbag 2017, luku 2).

Seuraavissa luvuissa käsitellään saavutettavuutta suunnittelun näkökulmasta. Sen lisäksi luvussa jäsennetään, mitä on otettava huomioon, kun suunnitellaan saavutettavia verkkosivuja ja digitaalisia palveluita erilaisille ihmisille. Tämän jälkeen tarkastellaan saavutettavuuteen liittyvää lainsäädäntöä sekä syvennytään verkkosisällön saavutettavuus ohjeen sisältöön ja rakenteeseen. Luvun lopussa tarkastellaan saavutettavuuden testausta ja arviointia.

3.1 Saavutettavuus suunnittelussa

Saavutettavien verkkosivujen suunnittelussa ja toteutuksessa huomioidaan kolme osa-aluetta. Tekninen saavutettavuus, helppokäyttöisyys sekä ymmärrettävyys. Tekninen saavutettavuus

tarkoittaa, että palvelu toimii eri päätelaitteilla sekä avustavilla teknologioilla. Helppokäyttöisyys tarkoittaa sitä, että palvelu on selkeä sekä intuitiivinen ja vaivaton käyttää. Ymmärrettävyys tarkoittaa puolestaan ymmärrettävää kielen käyttöä, jäsenneltyä sisältöä sekä sisällön tarjoamista tekstin lisäksi videoina, kuvina tai äänenä. (Aluehallintovirasto s.a. a.)

Teknisesti saavutettava verkkosivu on toteutettu käyttäen HTML-standardia, ja verkkosisällölle tarkoitettua saavutettavuusohjeistusta (WCAG-ohjeistus) on noudatettu (Aluehallintovirasto s.a. a). Toimintarajoitteiset käyttävät verkkosivujen selaamiseen erilaisia apuvälineitä tai avustavia teknologioita; esimerkiksi sokea käyttäjä tarvitsee ruudunlukijaa, jotta voi havaita näytöllä olevan visuaalisen sisällön, ja neliraajahalvaantunut käyttäjä käyttää näppäimistöselausta, jotta voi selata verkkosivuja ilman hiirtä. Avustavan teknologian käytön ymmärtäminen auttaa suunnittelemaan sekä toteuttamaan verkkosivuja saavutettavammin. Avustava teknologia hyödyntää HMTL-standardia. (Kehitysvammaliitto ry 2024a.)

Saavutettavuuden suunnittelussa oleellista on miettiä erilaisten käyttäjien toiminnallisia tarpeita ja käyttötilanteita (Kehitysvammaliitto ry 2023a). Kun sivustolla navigointi on selkeää ja sivuston hierarkia on yksinkertainen, käyttäjän on helpompi löytää haluttu sisältö sekä toiminnallisuus. Helppokäyttöisen verkkosivun toiminta on ennakoitavaa. Se tarkoittaa, että toimintojen suorittamisen ei tulisi aiheuttaa odottamattomia muutoksia, kuten sivupohjan rakenteen suuria muutoksia. Käyttäjälle annettavat ohjeet sekä palaute toiminnon onnistumisesta tai virheestä on myös keskeinen osa sivuston helppokäyttöisyyttä. Lomakkeella käyttäjälle annettu ohjeistus ja esimerkit siitä, mitä lomakkeen kenttään täytetään ja missä muodossa kenttä tulisi täyttää sekä käyttäjälle annettu selkeä ilmoitus onnistuneesta toiminnosta tai virheestä ja ohje virheen korjaamiseksi, auttavat lomakkeen täytössä ja tekevät siitä helpompaa. (Kehitysvammaliitto ry 2024b.)

Ymmärrettävä verkkosivu on kognitiivisesti saavutettava ja se auttaa henkilöitä, joilla on kognitiivisia haasteita esimerkiksi oppimisessa, muistamisessa ja hahmottamisessa. Ymmärrettävällä verkkosivulla on selkeä ja yksinkertainen rakenne, toiminnot löytyvät tutuista paikoista ja painikkeet sekä kuvakkeet ovat tuttuja. Tärkeä osa verkkosivun kognitiivista saavutettavuutta on, että sivulla oleva tieto on saatavilla eri muodoissa, kuten tekstinä, äänenä tai videoina, jolloin käyttäjällä on mahdollisuus valita itselleen sopivin muoto. Selkeä ja helposti ymmärrettävä teksti tekee verkkosivusta kognitiivisesti saavutettavamman. Vaikeiden sanojen selittäminen auki, lyhenteiden välttäminen sekä selkokielen käyttö helpottaa tekstin ymmärtämistä. Myös tekstin sijoittelu väljästi, väliotsikoiden sekä asioiden listaaminen, helpottaa tekstin silmäilyä ja lukemista. (Selkeästi meille -hanke 2023.)

Aluehallintovirasto (s.a. a.) nostaa esille, että saavutettavuuden suunnittelussa on tärkeää huomioida ihmisten erilaisuus ja moninaisuus. Suunnittele kaikille -periaate kiinnittääkin huomiota

erilaisiin käyttäjiin ja tilanteisiin alusta asti jo suunnitteluvaiheessa, jolloin kaikki erilaiset ihmiset voivat hyödyntää samaa ja yhtä palvelua. Saavutettavuuden huomioiminen suunnitteluvaiheessa on asiakaslähtöisyyttä ja se parantaa yhdenvertaisuutta digitaalisessa yhteiskunnassa. (Aluehallintovirasto s.a. a.)

3.2 Saavutettavuus ja lainsäädäntö

Digipalvelulaki (2023/104) on tullut Suomessa voimaan vuonna 2019, ja se perustuu Euroopan unionin vuonna 2016 voimaantulleeseen esteettömyysdirektiiviin (2019/882) (laki digitaalisten palvelujen tarjoamisesta). Digipalvelulaissa säädetään saavutettavuuden vaatimusten täyttämisestä, ja lain tavoitteena on, parantaa digitaalisten palveluiden saavutettavuutta sekä parantaa digitaalisten palveluiden yhdenvertaista käyttöä (laki digitaalisten palvelujen tarjoamisesta, 1 § ja 7 §).

Digipalvelulakiin alkuvuodesta 2023 voimaan tulleiden muutosten myötä, lain piiriin tuli uusia palveluita ja toimijoita (laki digitaalisten palvelujen tarjoamisesta annetun lain muuttamisesta 19.1.2023/104). Jatkossa digipalvelulaki koskee myös osaa palveluntarjoajista, jotka tarjoavat kuluttajille tarkoitettuja palveluita, kuten verkkokauppoja sähkökirjoja, osaa henkilöliikenteen palveluita, audiovisuaalista sisältöä tarjoavia palveluita, kuten suoratoistopalveluita ja viestintäpalveluita (laki digitaalisten palvelujen tarjoamisesta annetun lain muuttamisesta, 2 §). Muutosten siirtymäaika päättyy 28.6.2025, jonka jälkeen palveluiden, joita laki koskee, tulee olla saavutettavuusvaatimusten mukaisia (Aluehallintovirasto s.a. c).

Digipalvelulaki sisältää kolme keskeistä vaatimusta (laki digitaalisten palvelujen tarjoamisesta):

1. Digitaalisten palvelun pitää täyttää eurooppalaisen standardin EN 301 549 tekniset vaatimukset (EN 301 549:2021.)
2. Digitaalisesta palvelusta pitää löytyä saavutettavuusseloste (laki digitaalisten palvelujen tarjoamisesta, 9 §.)
3. Käyttäjillä pitää olla mahdollisuus antaa palautetta palvelun saavutettavuudesta (laki digitaalisten palvelujen tarjoamisesta, 9 §.)

Digipalvelulain sisältämät tekniset vaatimukset on lueteltu eurooppalaisessa EN 301 549 standardissa (Aluehallintovirasto s.a. d; EN 301 549:2021). EN-standardi sisältää suosituksia julkisen sektorin tieto- ja viestintätekniikan hankinnoille, ja siinä määritellään vähimmäistaso hankintojen saavutettavuudelle (EN 301 549:2021, 9–10). EN-standardissa viitataan WCAG-ohjeistuksen (Web Content Accessibility Guidelines) onnistumiskriteereihin (549:2021, 45). Seuraavassa luvussa tarkastellaan tarkemmin WCAG-ohjeistuksen sisältöä.

Saavutettavuusseloste on yksi digipalvelulain vaatimuksista. Saavutettavuusselosteen tulee sisältää tietoa verkkopalvelun saavutettavuudentasosta sekä selvityksen palvelun osista, jotka

poikkeavat saavutettavuusvaatimuksista. Saavutettavuusselosteesta on myös löydyttävä palvelutarjoajan yhteystiedot, jotta käyttäjä voi antaa palautetta palvelun saavutettavuudesta ja ohjeet, miten käyttäjä voi ottaa yhteyttä valvovaan viranomaiseen selvityspyyntöä tai kantelua varten. (laki digitaalisten palvelujen tarjoamisesta, 9 §).

Digipalvelulain lisäksi digitaaliseen saavutettavuuteen ohjaavat myös muut lait. YK:n yleissopimus vammaisten henkilöiden oikeuksista (2016/27) korostaa vammaisten ihmisen tasa-arvoisuutta ja yhdenvertaisuutta, ja kieltää syrjinnän vammaisuuden perusteella. Sopimuksen tarkoituksena on taata yhdenvertaiset oikeudet kaikille. (yleissopimus vammaisten henkilöiden oikeuksista 2016/27.) Yhdenvertaisuuslaki (2014/1325) kieltää syrjinnän, joka perustuu esimerkiksi vammaan tai toimintarajoitteeseen (yhdenvertaisuuslaki 30.1.2014/1325, 8 §). Laki koskee julkista ja yksityistä sektoria ja se velvoittaa palveluntarjoajia tarjoamaan palvelun kaikille käyttäjille (yhdenvertaisuuslaki, 2 § ja 15 §).

3.3 Verkkosisällön saavutettavuusohje WCAG

Web Content Accessibility Guidelines (WCAG) on verkkosisällön saavutettavuus ohje, jonka on tuottanut World Wide Web -konsortio (W3C). WCAG-ohjeistus sisältää ohjeistuksia siitä, miten verkkosisältöjä tehdään saavutettavaksi henkilöille, joilla on toimintarajoitteita. Ensimmäinen versio ohjeistuksesta julkaistiin vuonna 1999, jonka jälkeen ohjeistusta on päivitetty vuosina 2008 ja 2018. Uusin versio ohjeistuksesta, WCAG 2.2, on julkaistu loppuvuodesta 2023. (Aluehallintovirasto s.a. e; World Wide Web -konsortio 7.3.2024.)

WCAG-ohjeistuksen tavoitteena on varmistaa, että mahdollisimman moni voisi käyttää verkkopalvelua. Ohjeistuksessa annetaan etenkin ohjeita siihen, miten verkkosisällön käyttöä voi parantaa avustavalle teknologialle. (World Wide Web -konsortio 2023.) WCAG-ohjeistus sisältää ohjeita etenkin tekniseen saavutettavuuteen ja se on tarkoitettu etenkin verkkosisältöjen suunnittelijoille ja kehittäjille sekä kehittäjille, jotka tekevät työkaluja saavutettavuuden tarkastamiseen. (World Wide Web -konsortio 7.3.2024).

WCAG-ohjeistus ei takaa verkkopalvelun täydellistä saavutettavuutta, mutta sitä noudattamalla voidaan tehdä verkkosisällöstä käytettävämpää ja saavutettavampaa monille käyttäjille. WCAG-ohjeistuksessa korostetaan, että vaikka kaikki tekniset onnistumiskriteerit täyttyisivät, verkkosisältö ei välttämättä ole saavutettavaa kaikille käyttäjille. WCAG-ohjeistuksessa mainitaan, onnistumiskriteerien noudattaminen ei varmista saavutettavuutta käyttäjille, joilla on kognitiivisia, kielellisiä tai oppimiseen liittyviä haasteita. (World Wide Web -konsortio 2023.)

WCAG-ohjeistus koostuu neljästä osa-alueesta (kuva 2). Nämä ovat periaatteet, jotka muodostavat perustan ohjeistukselle. Periaatteet sisältävät 13 yleisluontoista ohjetta, jotka kertovat millaisia

asioita verkkosivujen saavutettavuudessa on otettava huomioon. Jokaisella ohjeella on onnistumiskriteerit, joihin tulee vastata, jotta saavutettavuusvaatimukset täyttyvät. Lisäksi ohjeistuksessa on jokaiselle onnistumiskriteerille ohjeet ja tekniikat, jotka auttavan kriteerien vaatimusten täyttämisessä. (World Wide Web -konsortio 2023.)

Kuva 2. WCAG-ohjeistuksen osa-alueet (World Wide Web -konsortio 2023.)

WCAG-ohjeistuksella on neljä periaatetta, jotka sijaitsevat ylimpänä kuvassa 2. Periaatteisiin kuuluvat havaittavuus, hallittavuus, ymmärrettävyys, ja toimintavarmuus. Havaittavuudella tarkoitetaan, että verkkosivuilla oleva tieto sekä käyttöliittymäkomponentit on esitettävät siten, että käyttäjä voi havaita ne. Hallittavuudella tarkoitetaan, että käyttöliittymäkomponentit ja navigointi on toteutettava siten, että käyttäjän on mahdollista hallita niitä. Verkkosivun navigoinnin on toimittava myös esimerkiksi ruudunlukijalla. Ymmärrettävyys tarkoittaa, että tiedon sekä käyttöliittymän toiminnan on oltava ymmärrettävää. Toimintavarmuudella tarkoitetaan, että verkkosivu toimii luotettavasti erilaisilla laitteilla ja ohjelmilla. (World Wide Web -konsortio 2.5.2024.)

Neljän periaatteen alle on luokiteltu 13 yleisluontoista ohjetta (kuvassa 2 toiseksi ylimpänä). Ohjeet tarjoavat kehyksen ja yleiset tavoitteet, jotka auttavat ymmärtämään WCAG-ohjeistusta. Ohjeet eivät ole testattavia. Tähän tarkoitukseen jokaiselle ohjeelle on luotu omat onnistumiskriteerit. Onnistumiskriteerit on luotu niin, että ne ovat testattavissa, jolloin niitä voi hyödyntää esimerkiksi verkkosivun saavutettavuuden tason arvioinnissa. Onnistumiskriteerit on puolestaan jaettu kolmeen eri tasoon: A, AA ja AAA (kuvassa 2 toiseksi alimpana). Näistä AAA-taso on korkein. AAA-tason kriteerien noudattaminen tekee verkkopalveluista saavutettavia mahdollisimman monelle. (World Wide Web -konsortio 2023.)

Näiden lisäksi ohjeistus tarjoaa erilaisia ohjeita ja tekniikoita siihen, miten onnistumiskriteerit voidaan toteuttaa käytännössä (kuvassa 2 alimpana). Ohjeet ja tekniikat on jaettu kahteen kategoriaan. Toinen kategoria sisältää ohjeita, joita noudattamalla voidaan vastata onnistumiskriteereihin.

Toinen kategoria sisältää sellaisia ohjeita, joita ei vaadita onnistumiskriteerien täyttämiseksi. Sen sijaan ne tarjoavat vinkkejä vieläkin paremman saavutettavuuden tason saavuttamiselle. Näiden lisäksi ohjeissa on listattuna myös tunnettuja virheitä, jotka johtavat epäonnistumiseen onnistumiskriteerien täyttämisessä. (World Wide Web -konsortio 2023.) WCAG-ohjeistus on erittäin laaja dokumentaatio, joka tulostettuna olisi yli tuhat sivua (Aluehallintovirasto s.a. e).

Tässä opinnäytetyössä WCAG-ohjeistusta hyödynnetään aineiston hankinnassa. Onnistumiskriteereitä käytetään apuna, kun havainnoidaan ja testataan tutkimuksen kohteena olevaa Power Apps -sovelluskehitysympäristöä ja sen ominaisuuksia saavutettavuusnäkökulmasta. Onnistumiskriteerit ovat teknisesti testattavissa, joten niiden avulla voidaan mitata, onko jonkin elementin toteuttaminen saavutettavasti mahdollista sovelluskehitysympäristössä vai ei. Aluehallintoviraston (s.a. e) mukaan onnistumiskriteerien A- ja AA-tason kriteerit ovat pitkälti teknisiä, ja eivät ota kantaa sisällön ymmärrettävyyteen. Sen sijaan AAA-tason onnistumiskriteereissä on sisältöön ja kognitiiviseen ymmärrettävyyteen liittyviä kriteereitä (Aluehallintovirasto s.a. e). Tässä opinnäytetyössä keskitytään sovelluskehitysympäristön teknisiin ominaisuuksiin, joten opinnäytetyössä ei otetaan huomioon sisältöön liittyviä kriteereitä.

3.4 Saavutettavuuden testaus ja arviointi

Saavutettavien verkkosivujen suunnittelun sekä toteutuksen lisäksi saavutettavuus tulee myös testata. Kalbagin (2017, luku 6) mukaan saavutettavuuden testaus on osa kehitysprojektia, mutta se on myös jatkuva prosessi ja osa ylläpitoa, jonka avulla varmistetaan, että palvelu pysyy saavutettavana myös silloin, kun tehdään muutoksia. Tässä luvussa käsitellään saavutettavuuden testausta ja asioita, joita testauksessa on hyvä ottaa huomioon. Aluksi keskitytään automaattiseen ja manuaaliseen testaukseen ja lopuksi käsitellään käyttäjätestausta. Luvussa tarkastellaan myös erilaisia työkaluja sekä avustavaa teknologiaa, joita testauksessa voidaan hyödyntää.

Automaattisessa testauksessa voidaan hyödyntää saavutettavuuden testaukseen tarkoitettuja ohjelmia, ohjelmakirjastoja tai selaimeen asennetavia lisäosia. Automaattiset testaustyökalut skannaavat verkkosivun ja kertovat, mitkä käyttöliittymäelementit eivät noudata saavutettavuusvaatimuksia. Automaattisilla testaustyökaluilla on mahdollista löytää ongelmia, jotka liittyvät esimerkiksi kontrasteihin, sivun rakenteeseen, puuttuviin otsikoihin, HTML-standardiin tai kuvien ja videoiden vaihtoehtoisiin teksteihin. Useat työkalut antavat myös ohjeita löydettyjen virheiden korjaamiseksi. (Georgakas 2023, luku 6.)

Suomen Kuntaliitto ry:n (2017) mukaan manuaalinen testaus voidaan usein toteuttaa vasta sitten, kun kehitystyö on pitkällä, ja suuri osa palvelusta valmiina. Kuntaliiton (2017) mukaan automaattisen testauksen avulla saavutettavuuteen liittyvät ongelmat voidaan havaita jo ennen kuin

manuaaliset testaukset aloitetaan. Automaattisten testaustyökalujen avulla laajat useita sivuja sisältävät verkkopalvelut on mahdollista testata nopeammin kuin manuaalisella testauksella, ja näin säästetään aikaa ja resursseja huolellisen manuaalisen testauksen tekemiselle. (Kuntaliitto 2017.) Automaattiset testaustyökalut toimivat hyvänä tukena manuaaliselle testaukselle, mutta automaattisten testaustyökalujen ongelmana on kuitenkin se, että ne eivät anna täydellistä kuvaa verkkosivun saavutettavuudesta. Automaattiset työkalut eivät esimerkiksi pysty arvioimaan, onko sisältö saavutettavaa, onko sivuilla näppäimistöansaa, tai ovatko kuvien vaihtoehtoiset tekstit loogisia. Automaattiset työkalut pystyvät tällä hetkellä löytämään noin 20–30 prosenttia WCAG onnistumiskriteereihin liittyvistä saavutettavuusongelmista. (Georgakas 2023, luku 6.)

Kehitysvammaliitto ry:n (2023b) mukaan manuaalisessa testauksessa on tärkeää varmistaa, että verkkosivulla mahdollista liikkua ilman hiirtä. Kehitysvammaliitto ry (2023b) kehottaa tarkistamaan, että sivuston kaikki toiminnot toimivat pelkillä näppäinkomennoilla, että sarkainjärjestys looginen, ja että elementtien kohdistukset selkeästi erotettavissa, jotta käyttäjä voi havaita, missä kohtaa sivustoa hän on. Georgakasin (2023, luku 6) mukaan näppäimistöselauksen avulla voidaan testata, onko verkkosivuilla niin kutsuttuja näppäimistöansoja, jolla tarkoitetaan, että kohdistus jää jumiin johonkin elementtiin ja käyttäjä ei pääse sivustolla eteenpäin.

Manuaalitestauksessa voidaan käyttää apuna myös avustavaa teknologiaa, kuten ruudunlukuohjelmaa, jonka avulla voidaan tarkistaa, miltä verkkosivu kuulostaa luettuna (Kehitysvammaliitto ry 2023b). Näkövammaisten liitto ry:n (2023) mukaan ruudunlukijaa ohjataan myös näppäimistön avulla, mutta navigointi eroaa näppäimistöllä navigoinnista siten, että se pysähtyy elementteihin, jotka näppäimistöllä liikuttaessa ohitetaan, kuten tekstisisällöt. Ruudunlukijan avulla voidaan varmistaa, että kaikki tarvittava informaatio välittyy näkövammaiselle käyttäjälle, ja varmistaa, että sivun rakenne looginen sekä sisältö ymmärrettävissä myös ilman visuaalisesti esitettävää informaatiota. (Näkövammaisten liitto ry 2023.)

Testaus eri päätelaitteilla ja selainversioilla on osa manuaalista testausta, sillä käyttöjärjestelmä ja selaimen eri versiot saattavat vaikuttaa verkkosivun toimivuuteen. Päätelaitteen koko voi puolestaan vaikuttaa siihen, miltä sivu näyttää. Avustavat teknologiat ovat usein päätelaitekohtaisia, ja eri valmistajien laitteissa voi olla eri ruudunlukijat. Sivustot saattavat toimia eri ruudunlukijoilla eri tavoin. Manuaalisessa testauksessa huomioidaan eri päätelaitteet ja selaimet sekä niiden eri yhdistelmät. (Kehitysvammaliitto ry 2023b; Kuntaliitto 2017.)

Kehitysvammaliitto ry (2023b) kehottaa testaamaan sivustoa ilman värejä sekä käännetyillä väreillä, jolloin voidaan varmistua siitä, että visuaaliset värit eivät ole ainoa keino korostaa asian merkitystä sivulla ja, että merkitys on selvää myös ilman värejä tai käännetyillä väreillä.

Kehitysvammaliitto ry (2023b) kehottaa myös testaamaan sivuston tai tekstin koon suurentamista selaimessa, jolloin voidaan varmistua, että rakenne pysyy loogisena ja sisältö luettavana.

Georgakasin (2023, luku 6) mukaan manuaaliseen testaukseen sisältyy sisällön tarkistus. Sisällön testauksessa tarkistetaan, että ovatko linkit kuvaavia ja tietävätkö käyttäjät, mihin sisältöön linkin kautta siirrytään, ja että, onko kaikilla kuvilla kuvaavat vaihtoehtoiset tekstit. Sisällön loogisuuden sekä rakenteen manuaalisessa testauksessa tarkistetaan, että ovatko otsikot kuvaavia, ja onko kaikilla sivuilla yksilölliset otsikot. Sisällön testauksessa varmistetaan myös, että onko käytetty kieli ymmärrettävää ja välittyykö informaatio käyttäjälle. Videoiden ja niiden tekstitysten toimivuus tarkastetaan, samoin kuin ääniraitojen toimivuus ja äänenlaatu. (Georgakas 2023, luku 6.)

Georgakasin (2023, luku 6) mukaan on myös tärkeää tarkistaa, että sivulla HTML-elementtejä ja WAI-ARIAA on käytetty tarkoituksen mukaisesti. Hän korostaa, että natiiveja HTML-elementtejä tulisi käyttää aina, jos mahdollista, sillä ne tarjoavat elementistä semanttista tietoa ruudunlukijalle. WAI-ARIAA voi hänen mukaansa käyttää apuna tilanteissa, joissa natiivia HTML-elementtiä ei voida käyttää tai elementistä tarvitsee välittää käyttäjälle enemmän tietoa kuin natiivi elementti voi välittää.

Kehitysvammaliitto ry:n (2023c) mukaan tärkeänä osana testausta on saavutettavuus- ja käytettävyystestaus oikeilla käyttäjillä, sillä monet ongelmat huomataan vasta käyttäjätestauksessa. Käyttäjätestauksessa testataan verkkosivuston toiminnallisuuksia, ja tarkastellaan, että saako käyttäjä suoritettua jonkin ennalta määritellyn tehtävän sivulla, ja onko tehtävän suorittaminen helppoa. Käyttäjätestauksen avulla saadaan lisäksi tietoa palvelun käytettävyydestä, kuten siitä onko palvelun käyttäminen tehokasta ja miellyttävää. (Kehitysvammaliitto ry 2023c.)

Myös Kalbag (2017, luku 6) korostaa käyttäjätestauksen tärkeyttä. Hänen mukaansa monet ongelmat jäävät usein huomaamatta, mikäli sivusto testataan vain projektin sisäisesti. Kalbagin (2017, luku 6) mukaan tämä johtuu siitä, että verkkosivusto, sen sisältö ja toiminnallisuudet ovat usein itsestään selviä henkilöille, jotka työskentelevät verkkosivuston kehitysprojektissa. Hän myös huomauttaa, että projektissa työskentelevät henkilöt eivät todennäköisesti ole verkkosivun oikeaa kohderyhmää. Kalbag (2017, luku 6) painottaakin, että käyttäjätestauksessa on hyvä huomioida, että testaajat vastaavat mahdollisimman hyvin verkkosivun kohderyhmää myös silloin kun testaajina toimivat käyttäjät, joilla on toimintarajoitteita. Hänen mukaansa käyttäjätestauksessa on hyvä olla mukana myös testaajia, jotka käyttävät avustavaa teknologiaa päivittäin sillä he ovat perusteellisempia, ja löytävät ongelmat saavutettavuudessa helpommin verrattuna käyttäjiin, jotka eivät käytä avustavaa teknologiaa päivittäin.

Seuraavassa luvussa esitellään opinnäytetyöhön valittu tutkimusstrategia ja avataan sen valintaan johtaneita näkökulmia. Luvussa esitellään myös tarkemmin tutkimuksen kohteena oleva Microsoftin Power Apps -sovelluskehitysympäristö sekä kerrotaan, mitkä tekijät vaikuttivat siihen, että kyseisen sovelluskehitysympäristö valittiin tutkimuksen kohteeksi.

4 Tutkimuksen toteutus

Tässä opinnäytetyössä tutkimusstrategiana on käytetty tapaustutkimusta. Erikssonin ja Koistisen (2014, 4) mukaan tapaustutkimuksessa tarkastellaan yhtä tai useampaa tapausta ja keskeisimpänä tavoitteena on määritellä ja analysoida tapaus. Tapaustutkimuksessa on tärkeää valita, rajata ja perustella, miksi tapaus on valittu tutkimuksen kohteeksi (Eriksson & Koistinen 2014, 4). Tapaustutkimuksen tavoitteena on pyrkiä luomaan tarkka ja havainnollinen kuvaus tutkimuskohteesta. Tapaus on esimerkki tutkittavasta ilmiöstä, ja se on aina liitoksissa kontekstiinsa eli aikaan ja toimintaympäristöön. Tapaustutkimuksessa tapaus on rajattu esimerkki, joka edustaa laajempaa ilmiötä, ja jonka kautta voidaan oppia ilmiöstä jotain uutta. (Vuori 2021.)

Opinnäytetyön tutkimuskysymyksenä oli, soveltuuko low-code -menetelmä saavutettavien sovellusten toteuttamiseen. Opinnäytetyön laajuuden vuoksi päädyttiin valitsemaan kohteesi yksi esimerkkitapaus, joka edustaisi low-code -menetelmää, ja jonka saavutettavuusominaisuuksista pyrittäisiin luomaan tarkka kuvaus. Tutkimuksen kohteeksi valittiin Microsoftin Power Apps -sovelluskehitysympäristö. Tavoitteena oli luoda tapausesimerkistä kattava kuvaus tutkimuksen kohteesta sekä sen ominaisuuksista. Valintaan vaikutti Microsoftin laaja tunnettuus palveluntarjoajana sekä yksityishenkilöiden että yritysten ja organisaatioiden keskuudessa. Matvitskyyn ja kumppanien (2023) mukaan Microsoft on yksi johtavista low-code -alustojen tarjoajista. Heidän mukaansa Power Appsin laajaa käyttäjäkuntaa selittää muun muassa se, että Microsoft Office 365 and Dynamics 365 yrityskäyttäjät saavat osittaisen käyttöoikeudet sovelluskehitysympäristöön. Matvitskyyn ja kumppanien (2023) mukaan Power Appsin suosiota selittää myös Power myös se, että se on helposti integroitavissa sekä Microsoftin muihin palveluihin, että laajasti myös kolmannen osapuolen palveluntarjoajien rajapintoihin sekä tietolähteisiin.

Valintaan vaikuttivat myös Power Apps -alustan ominaisuudet ja niiden yhteneväisyys ominaisuuksien kanssa, joita on käytetty määrittelemään low-code -menetelmää (ks. luku 2.1 Low-code). Microsoftin (22.22024) dokumentaatiosta selviää, että Power Apps -sovelluskehitysympäristöllä voi luoda sovelluksia ilman, että tarvitsee kirjoittaa lainkaan perinteistä ohjelmointikieltä. Power Appsin käyttöliittymäkomponentteja voi raahata osaksi sovellusta. Käyttöliittymäkomponentteja muokataan visuaalisessa editorissa ja muokkaus samankaltaista kuin tekstin koon, värin tai asettelun muokkaus on Microsoft Word -tekstinkäsittelyohjelmassa. (Microsoft 22.2.2024.) Toimintojen toteuttamiseksi Power Appsissa käytetään Power Fx -kaavoja, jotka ovat samankaltaisia, kuin Microsoftin laskentaohjelman, Excelin, kaavat. Ohjausobjektien ominaisuuksiin, kuten, Visible, näkyvyys, voidaan vaikuttaa kaavoilla. Power Fx -kaavoilla voidaan kertoa esimerkiksi painikkeille toiminto, joka suoritetaan, kun käyttäjä klikkaa painiketta. (Microsoft 22.3.2024.) Power Apps -

sovelluskehitysympäristön tunnettuuden sekä ominaisuusien takia Power Appsin todettiin olevan perusteltu esimerkkitapaus edustamaan low-code ympäristöjä.

Tässä opinnäytetyössä aineiston hankinnassa käytettiin strukturoitua havainnointia (Saaranen-Kauppinen & Puusniekka 2006). Seuraavissa luvuissa esitellään tarkemmin käytetty aineistonhankintamenetelmä sekä kuvataan, millaisia rajauksia aineiston hankinnassa käytettiin, ja miten empiirinen aineisto kerättiin tutkimuksen kohteena olevasta Power Apps -sovelluskehitysympäristöstä

4.1 Aineiston hankinta

Tässä opinnäytetyössä aineistonhankinnassa käytettiin menetelmänä strukturoitua havainnointia. Saaranen-Kauppisen ja Puusniekan (2006) mukaan havainnointi on tarkkailua, jossa kerätään tietoa havainnoitavasta kohteesta. Strukturoitu havainnointi on systemaattista ja standardoitua. Strukturoidussa havainnoinnissa tutkimusongelma jäsennellään ennen varsinaista havainnointia, ja sen perusteella tehdään päätös, että mitä tutkittavasta kohteesta havainnoidaan. (Saaranen-Kauppinen & Puusniekka 2006.)

Ennen havainnoinnin toteuttamista, tutustuttiin Power Apps -sovelluskehitysympäristön ja sen dokumentaatioon sekä ohjeisiin liittyen saavutettavuuteen. Power Appsiin tutustumalla saatiin selville, että Power Appsin käyttöliittymäkomponenteilla, eli ohjausobjekteilla, oli erilaisia saavutettavuuteen liittyviä ominaisuuksia, ja että Power Appsissa oli sisäänrakennettuna saavutettavuustarkastaja, joka kertoi käyttäjälle ohjausobjektien saavutettavuusvirheistä ja antoi ohjeita saavutettavuuden parantamiseen. Tämän pohjalta tehtiin päätös, että tutkimuksen kohteesta havainnoitaisiin ohjausobjekteja ja niiden ominaisuuksia sekä lisäksi päätettiin havainnoida Power Appsin saavutettavuustarkastajaa.

Ohjausobjektien ominaisuuksia pyrittiin havainnoimaan jäsennellysti vain saavutettavuuden näkökulmasta. Tästä syystä havainnoinnin tueksi otettiin WCAG-ohjeistuksen onnistumiskriteerit, jotka ovat teknisestä näkökulmasta testattavissa (ks. luku 3.3). Ohjausobjektien ominaisuuksia tarkasteltiin suhteessa onnistumiskriteereihin; miten hyvin ominaisuuksilla pystyttiin vastaamaan kriteerin vaatimuksiin. Havainnoinnin tavoitteena oli tarkastella Power Apps -ohjausobjektien ominaisuuksia ja valmiuksia teknisestä näkökulmasta. Tämän takia havainnoinnissa otettiin huomioon vain sellaiset onnistumiskriteerit, jotka oli mahdollista testata teknisestä näkökulmasta ja havainnoinnista rajattiin pois sellaiset onnistumiskriteerit, jotka liittyivät verkkosivun sisältöön. Tämän lisäksi onnistumiskriteereistä jätettiin pois kriteerit, joiden onnistuminen oli riippuvainen sisällön lisäksi designvalinnoista.

Jokaisen onnistumiskriteerin kohdalla havainnoitiin ohjausobjekteja, jotka liittyivät läheisesti kyseiseen WCAG-ohjeeseen. Opinnäytetyön laajuuden vuoksi kaikkia ohjausobjekteja ei havainnoitu

kaikkien onnistumiskriteerien kohdalla. Jokaisen onnistumiskriteerin kohdalla otetiin huomioon sellaiset ohjausobjektit, jotka läheisimmin liittyivät kyseiseen kriteeriin ja sen vaatimuksiin. Joidenkin kriteerien kohdalla testattiin useampi ohjausobjekti, ja joidenkin kohdalla vain yksi.

Tapaustutkimukselle on tyypillistä käyttää rinnakkain sekä laadullista, että määrällistä aineistoa (Eriksson & Koistinen 2014, 4). Vilkan (2021, luku 5) mukaan havainnointia varten on hyvä luoda asiaruko. Hänen mukaansa asiarunko voi olla haastattelulomakkeen omainen ja siihen määritellään, mitä ja miten havainnoidaan. Tässä opinnäytetyössä havainnoinnin tueksi luotiin testauslomake. Tavoitteena oli kerätä sekä laadullista, kuten millaisia saavutettavuusominaisuuksia ohjausobjekteilla on, sekä määrällistä aineistoa, kuten kuinka moneen WCAG-ohjeistuksen kriteeriin pysyttiin vastaamaan. Testauslomakkeen avulla pyrittiin tekemään havainnoinnista järjestelmällistä ja jäsenneltyä.

Seuraavassa luvussa avataan, miten havainnoinnin tukena käytetty testauslomake luotiin, ja millainen testauslomakkeen rakenne on sekä kerrotaan, millaisia asioita testauslomakkeen avulla havainnoitiin, ja perustellaan, miksi ne on valittu testauslomakkeeseen. Luvussa kerrotaan myös, miten testauslomaketta hyödynnettiin havainnoinnin tukena, ja miten havainnointi toteutettiin käytännössä.

4.2 Testaaminen

Power Apps -sovelluskehitysympäristöön tehdyn alkukartoituksen perusteella oli päätetty havainnoida ympäristön ohjausobjekteja ja niiden ominaisuuksia WCAG-onnistumiskriteereitä vasten sekä havainnoida ympäristön omaa saavutettavuustarkastajaa. Nämä näkökulmat pyrittiin ottamaan huomioon, kun aineiston hankinnan tueksi tarkoitettua testauslomaketta rakennettiin. Testauslomakkeessa oli alustavasti kolme kohtaa havainnoille: Ohjausobjektin ominaisuudet, saavutettavuustarkastaja sekä ohjausobjektin saavutettavuuden arviointi. Testauslomakkeen luontia varten päätettiin suorittaa alustava testaus, jotta voitaisiin arvioida testauslomakkeen toimivuutta. Testauksen jälkeen rakennettiin lopullinen testauslomake, jota käytettiin aineiston hankinnassa. Kuvassa 3 kuvattuna opinnäytetyön aineistonhankinnan eteneminen.

Kuva 3. Aineistonhankinnan eteneminen

Alustavassa testauksessa valittiin muutama WCAG-ohjeistuksen onnistumiskriteeri sekä muutama ohjausobjekti. Ohjausobjektit pyrittiin toteuttamaan siten, että ne täyttävät onnistumiskriteerit ja samalla tarkasteltiin ohjausobjektien saavutettavuuteen liittyviä ominaisuuksia. Ohjausobjektien saavutettavuuden arvioinnissa hyödynnettiin sekä automaattista että manuaalista testausta (ks. luku 3.3). Alustavan testauksen aikana huomattiin, että selainpohjaiset automaattiset kolmannen osapuolen tarkastustyökalut eivät toimineet Power Apps -sovelluskehitysympäristössä, joten saavutettavuuden automaattinen testaaminen toteutettiin ympäristön omalla saavutettavuustarkastajalla. Samalla tarkasteltiin saavutettavuustarkastajan antamia virheitä ja huomioita. Lopuksi ohjausobjektien saavutettavuus testattiin manuaalisesti käyttäen joko ruudunlukijaa tai näppäimistöselausta.

Alustavan testauksen pohjalta huomattiin, että osa onnistumiskriteerien vaatimuksista pystyttiin täyttämään joillain ohjausobjekteilla täysin ja toisilla ohjausobjekteilla vain osittain. Samankaltainen huomio nousi esille myös saavutettavuustarkastajasta. Saavutettavauustarkastaja löysi osan virheistä, mutta jätti myös joitain virheitä huomioimatta. Näiden huomioiden perusteella koettiin, että yksittäisen ohjausobjektin saavutettavuutta suhteessa WCAG-ohjeistuksen onnistumiskriteeriin ei voitu arvioida yksiselitteisesti joko onnistuneeksi tai epäonnistuneeksi. Tämän takia lomakkeeseen päätettiin luoda arviointiasteikko, jolla voitiin arvioida onnistumiskriteerin täyttymistä, sekä saada tarkempaa määrällistä arviota saavutettavuusvaatimusten täyttymisestä, ja vetää tuloksia yhteen. Arviointiasteikon tasot olivat:

3. Kriteeri täyttyi
4. Kriteeri täyttyi osittain
5. Kriteeri ei täyttynyt

Arviointiasteikkoa päätettiin hyödyntää kaikissa havainnoitavissa osa-alueissa: ohjausobjektin ominaisuuksien havainnoinnissa, tarkastustyökalun havainnoinnissa sekä ohjausobjektin saavutettavuuden arvioinnissa. Alustavan testauksen aikana huomattiin, että Power Appsin dokumentaatio oli tärkeä tiedonlähde ohjausobjektin ominaisuuksien selvittämisessä, joten dokumentaation kattavuudesta päätettiin myös kerätä havaintoja testauksen aikana. Dokumentaatiota ei kuitenkaan ollut mielekästä arvioida arviointiasteikolla.

Lopullinen testauslomake muodostui edellä mainittujen havaintojen pohjalta (Liite 1). Testauslomakkeen alussa esiteltiin WCAG-ohje, siihen liittyvä testattava onnistumiskriteeri (ks. luku 3.3) sekä lyhyet kuvaukset. Kuvaukset auttavat lukijaa ymmärtämään, mihin testattava WCAG-kriteeri liittyy, ja mitä kyseisen kriteerin onnistuminen vaatii. Testauslomakkeissa käytetyt kuvaukset on otettu Kehitysvammaliitto ry:n tekemästä WCAG-ohjeistuksen virallisesta suomennoksesta vuodelta 2019.

Testauslomake koostui neljästä kohdasta, joihin kerättiin havaintoja tutkimuksen kohteesta. Ensimmäinen kohta oli varattu testattavan ohjausobjektin ominaisuuksille. Tähän kohtaan kuvatiin ne ominaisuudet, joita hyödynnettiin onnistumiskriteerien vaatimusten täyttämisessä. Lisäksi kohtaan kirjattiin huomioita, joita nousi ohjausobjektin ominaisuuksiin liittyen. Lopuksi ohjausobjektin ominaisuudet arviointiin arviointiasteikon mukaan. Mikäli ohjausobjektista löytyi ominaisuus tai ominaisuuksia, joilla pystyttiin vastaamaan kriteerin vaatimuksiin, arvioitiin se asteikolla ylimmälle tasolle. Vastaavasti, mikäli ohjausobjektilla oli ominaisuuksia, joilla voitiin täyttää kriteerin vaatimukset osittain, se arvioitiin asteikoilla toiselle tasolle. Ohjausobjekti arviointiin asteikolla alimmalle tasolle, mikäli ohjausobjektilla ei ollut ominaisuuksia, joilla voitiin vastata kriteerin vaatimuksiin ja ohjausobjektia ei voitu toteuttaa saavutettavaksi.

Testauslomakkeen toinen kohta oli varattu Power Appsin omalle tarkastustyökalulle. Ohjausobjekti pyrittiin aluksi toteuttamaan siten, että se tarkoituksellisesti rikkoisi saavutettavuusvaatimuksia, jotta voitiin selvittää, antoiko tarkastustyökalu virheen saavutettavuusvirheen tai ohjeita ohjausobjektin saavutettavuuden parantamiseksi. Tarkastustyökalu arviointiin myös arviointiasteikolla. Mikäli tarkastustyökalu löysi ohjausobjektin saavutettavuudessa ongelmia, se arvioitiin ylimmälle tasolle, jos taas tarkastustyökalu löysi osan virheistä, se arviointiin toiselle tasolle, ja mikäli tarkastustyökalu ei löytänyt ilmiselvää virhettä se arviointiin alimmalle tasolle. Myös tarkastustyökalusta kirjattiin ylös esiin nousseita huomioita.

Kolmas kohta testauslomakkeessa oli varattu manuaalisille testeille. Manuaalista testausta varten ohjausobjekti pyrittiin toteuttamaan saavutettavuusvaatimusten mukaisesti. Tämän jälkeen se testattiin ruudunlukijalla tai muulla avustavalla teknologialla, kuten näppäimistöselauksella ja tulokset arvioitiin arviointiasteikolla. Mikäli ohjausobjekti pystyttiin toteuttamaan siten, että se oli saavutettava avustavalle teknologialle ja täytti onnistumiskriteerin vaatimukset, se arvoitiin ylimmälle tasolle. Mikäli ohjausobjektia ei ollut mahdollista toteuttaa avustavalle teknologialle saavutettavasti, se arvioitiin alimmalle tasolle, ja jos ohjausobjekti oli vain osittain saavutettava avustavalle teknologialle, se arviointiin toiselle tasolle. Manuaalisen testauksen aikana kirjattiin myös huomioita, joita testauksen aikana nousi esille.

Viimeisenä kohtana testauslomakkeessa oli varattu tilaa muille huomioille. Tähän kohtaan kirjattiin ylös esimerkiksi huomiot, joita Power Appsin omasta dokumentaatiosta löytyi liittyen testattavan ohjausobjektin ominaisuuksiin tai saavutettavuusohjeisiin. Power Appsin dokumentaatiossa oli myös nostettu esille tunnettuja ongelmia saavutettavuuteen liittyen. Nämä huomiot kirjattiin myös testauslomakkeelle. testauslomakkeen viimeistä kohtaa ei arvioitu arviointiasteikon mukaan.

Seuraavassa luvussa avataan, miten aineisto on analysoitu ja esitellään tutkimuksen tulokset. Luvussa tarkastellaan tuloksia yksityiskohtaisesti sekä käydään läpi tärkeimmät havainnot ja teemat,

jotka nousivat esille analyysin aikana. Tämän jälkeen, avataan johtopäätökset ja pyritään vertaamaan niitä tietopohjaan.

5 Tutkimuksen tulokset

Tässä luvussa avataan, miten kerätty aineisto on analysoitu sekä vedetään yhteen Power Apps sovelluskehitysympäristön testaamisessa esiin nousseet havainnot. Luvussa käydään läpi, mitkä WCAG onnistumiskriteereistä onnistuttiin täyttämään hyvin, ja mitkä onnistumiskriteeristä olivat sellaisia, joiden täyttämisessä oli haasteita. Tämän lisäksi kuvataan, millaisia ominaisuuksia Power Appsilla oli saavutettavuuden parantamiseen sekä tarkastellaan, mitkä ohjausobjekteista oli mahdollista toteuttaa vaatimusten mukaisesti, ja missä ohjausobjekteissa oli haasteita saavutettavuuden näkökulmasta.

Power Apps sovelluskehitysympäristön ohjausobjekteja ja niiden ominaisuuksia havainnoitiin yhteensä kahdeksaatoista WCAG-ohjeistuksen onnistumiskriteeriä vasten. Näistä onnistumiskriteereistä 13 oli tason A kriteereitä ja 5 tason AA kriteereitä. Testauksessa tarkasteltiin yhteensä seitsemäätoista eri ohjausobjektia, ja osaa näistä ohjausobjekteista tarkasteltiin useampaa onnistumiskriteeriä vasten. Kaiken kaikkiaan testauslomakkeen avulla testejä tehtiin yhteensä 32.

Tutkimusaineistoa analysoitiin sekä määrällisesti että laadullisesti. Määrällisessä analyysissä hyödynnettiin testauslomakkeen arviointiasteikon avulla kerättyjä arvioita sekä analysoitiin niitä Excel-taulukon avulla. Laadullisessa analyysissä hyödynnettiin testauslomakkeeseen kerättyjä kirjallisia havaintoja. Seuraavaksi käydään läpi tarkemmin, miten aineisto on analysoitu sekä esitetään tutkimuksen tulokset.

5.1 Onnistumiskriteerit ja ohjausobjektit

Tutkimusaineiston määrällisessä analyysissä hyödynnettiin testauslomakkeen arviointiasteikon avulla kerättyjä arvioita. Arviot vietiin Excel-taulukkoon, jossa aineistoa tarkasteltiin kahdesta eri näkökulmasta: onnistumiskriteerit ja ohjausobjektit. Ensimmäisessä näkökulmassa tavoitteena oli tarkastella onnistumiskriteerien vaatimusten täyttymistä kokonaisuutena yleiskuvan muodostamiseksi. Lisäksi tavoitteena oli tarkastella yksittäisten onnistumiskriteerien onnistumista yksityiskohtaisemman tiedon keräämiseksi. Toisessa näkökulmassa tavoitteena oli tarkastella ohjaisobjektien saavutettavuuden tasoa kokonaisuutena, jonka lisäksi tavoitteena oli tarkastella yksittäisten ohjausobjektien saavutettavuuden tasoa.

Onnistumiskriteerien tarkastelua varten jokaiselle onnistumiskriteerille laskettiin keskiarvo, niiden ohjausobjektien saamista arviointiasteikon arvoista, jotka testattiin kyseisen onnistumiskriteerin kohdalla. Keskiarvon laskennassa otettiin huomioon ohjausobjektien ominaisuuksien ja manuaalisten testauksen saamat arvot. Tämän jälkeen onnistumiskriteerit järjestettiin lasketun keskiarvon mukaan.

Keskiarvoja tarkasteltaessa voitiin havaita, että onnistumiskriteerien vaatimuksiin, jotka liittyivät saavutettavan nimen lisäämiseen (WCAG 1.1.1) sivun ja osien kielen asettamiseen (WCAG 3.1.1 ja 3.1.2), sekä käyttäjille annettavaan palautteeseen (4.1.3 ja 3.3.1) pystyttiin vastaamaan hyvin. Sen sijaan onnistumiskriteerien, jotka liittyivät toimintojen suorittamiseen pelkän näppäimistön avulla (WCAG 2.1.1) tai toimintojen suorittamiseen yhdellä klikkauksessa monipisteohjauksen (raahaus) sijaan (WCAG 2.5.1) vaatimusten täyttämisessä oli eniten ongelmia. Haasteita oli myös sellaisten onnistumiskriteerien kohdalla, jotka liittyivät käyttöliittymäelementin rooleihin (WCAG 4.1.2) sekä käyttöliittymäelementtien välisten suhteiden tai sivun rakenteiden ja informaation välittämiseen ohjelmallisesti (WCAG 1.3.1).

Kahdeksan onnistumiskriteerin keskiarvo oli 1, joka tarkoitti onnistumiskriteerin vaatimusten täyttymistä onnistuneesti. Yhdeksän onnistumiskriteerin keskiarvo oli 1–2, joka tarkoitti kriteerin osittaista täyttymistä. Yhden onnistumiskriteerin keskiarvo oli yli 2, joka tarkoitti suuria ongelmia vaatimusten täyttymisessä tai epäonnistumisesta vaatimusten täyttämisessä. Kaikkiaan 45 % onnistumiskriteereistä täyttyi, 50 % onnistumiskriteereistä täyttyi osittain ja 5 % onnistumiskriteereistä ei täyttynyt tai täyttyi huonosti.

Ohjausobjektien tarkastelua varten jokaisen ohjausobjektin ominaisuuksien sekä manuaalisen testauksen saamista arviointiasteikon arvoista muodostettiin keskiarvo. Sellaisille ohjausobjekteille, jotka oli testattu useamman onnistumiskriteerin kohdalla, laskettiin yhteinen keskiarvo. Tämän jälkeen ohjausobjektit järjestettiin lasketun keskiarvon mukaan.

Keskiarvoja tarkasteltaessa voitiin havaita, että yksinkertaisimmat ohjausobjektit (Teksti, Kuva, Kuvake ja Valintanappi) saivat parhaimman keskiarvon. Nämä ohjausobjektit oli mahdollista toteuttaa saavutettavuusvaatimusten mukaisesti. Monimutkaisempien ohjausobjektien keskiarvot olivat sen sijaan huonompia, ja näiden ohjausobjektien toteuttamisessa saavutettavasti havaittiin ongelmia. Monimutkaisempia ohjausobjekteja olivat muun muassa Tekstieditori, Valintalaatikko ja 3D-objekti. Myös joissain yksinkertaisemmissa ohjausobjekteissa oli haasteita saavutettavuusvaatimusten täyttämisessä, kuten Painike ja Säiliö, joka on verrattavissa HTML-elementtiin, joka ryhmittelee ohjausobjekteja.

Seitsemästätoista ohjausobjektista kahdeksalla ohjausobjektilla oli keskiarvo 1, kuudella ohjausobjektilla keskiarvo oli 1–2 ja kolmella ohjausobjektilla keskiarvo oli enemmän kuin 2. Ohjausobjekteista 47 % täytti onnistumiskriteerit täysin, 35 % täytti onnistumiskriteerit osittain ja 18 % ohjausobjektia täytti kriteerit huonosti tai ei täyttänyt kriteereitä lainkaan.

5.2 Ominaisuudet, saavutettavuustarkastaja ja dokumentaatio

Tutkimusaineiston laadullisessa analyysissä hyödynnettiin testauslomakkeeseen kerättyjä havaintoja ohjausobjektien ominaisuuksista, saavutettavuustarkastajasta sekä dokumentaatiosta. Tavoitteena oli luoda kattava kuva siitä, millaisia ominaisuuksia ohjausobjekteilla oli saavutettavuuden parantamiseen, ja siitä, miten hyvin saavutettavuustarkastaja löysi erilaisia saavutettavuusvirheitä. Lisäksi tavoitteena oli arvioida Power Appsin dokumentaation kattavuutta saavutettavuusohjeiden näkökulmasta. Ominaisuuksista, saavutettavuustarkastajasta sekä dokumentaatiosta kerätyt havainnot koottiin testauslomakkeista, jonka jälkeen tarkasteltiin, että löytyikö erilaisten ohjausobjektien kohdalla jotain havainnoista samankaltaisia huomioita tai yhteneväisiä teemoja.

Ominaisuuksien näkökulmasta tarkasteltuna keskeisimmät saavutettavuutta parantavat ominaisuudet olivat esimerkiksi ruudunlukijaa varten asetettava saavutettava nimi, näppäimistönavigointia varten asetettava kohdistuksen väri ja koko sekä kohdistuksen pysäytys. Edellä mainitut ominaisuudet löytyivät lähes kaikilta ohjausobjekteilta. Näiden lisäksi monimutkaisemmilla ohjausobjekteilla oli myös yksilöllisiä saavutettavuutta parantavia ominaisuuksia, esimerkiksi Video-ohjausobjektilla oli ominaisuus tekstityksen linkin lisäämiselle, Teksti-ohjausobjektilla oli ominaisuus sisällön muutoksien välittämiselle, Liukusäädin-ohjausobjektilla oli ominaisuus arvon välittämiselle ja Kartta-ohjausobjektilla oli mahdollisuus lisätä kartan kontrolloinnille painikkeet näppäimistönavigointia varten.

Havainnoista nousi esille huomioita liittyen ohjausobjektien roolien ja tilan välittämiseen. Usean ohjausobjektin kohdalla, olisi puutteita ominaisuuksista, jolla olisi välitetty tietoa ohjausobjektista ruudunlukijalle, kuten WAI-ARIAn roolit ja attribuutit. Tällaisia ominaisuuksia olisit tarvittu esimerkiksi tiedon välittämiseen sovelluksen rakenteesta, kuten navigointi, pääsisältö tai banneri sekä tiedon välittämiseen käyttöliittymäelementtien suhteesta, kuten syötekentän nimen liittäminen syötekenttään ohjelmallisesti. Jossain tapauksissa ohjausobjektista ei ollut mahdollista välittää oikeaa tietoa; Teksti-ohjausobjektia ohjeistettiin käyttämään linkkinä, mutta tässä tapauksessa ruudunlukija tulkitsi käyttöliittymäelementin kuitenkin painikkeeksi eikä linkiksi. Ohjausobjektit kuitenkin suurimmaksi osaksi välittivät oikeaa tietoa roolista ja ohjausobjektin tilasta, kuten tulkitsi kuvan painikkeeksi, jos sitä oli käytetty interaktiivisena elementtinä.

Power Appsin saavutettavuustarkastajalle on määritelty dokumentaatiossa kymmenen kohtaa, jotka saavutettavuustarkastaja voi tunnistaa sovelluksesta. Saavutettavuustarkastajaa ei ollut mielekästä arvioida kaikkien onnistumiskriteerien kohdalla, sillä tiedossa oli etukäteen ne saavutettavuuteen liittyvät seikat, jotka saavutettavuustarkastaja voi löytää. Tämän takia saavutettavuustarkastajan saamia arvioita ei analysoitu määrällisesti.

Saavutettavuustarkastajan tunnistamat saavutettavuusongelmat oli kerrottu dokumentaatiossa. Niitä oli yhteensä kymmenen ja ne oli luokiteltu dokumentaatiossa kolmeen eri vakavuusasteeseen: virheet, varoitukset ja ohjeet. Testauksen aikana saavutettavuustarkastaja tunnisti näistä yhdeksän. Saavutettavuustarkastajalle määriteltyjä virheitä oli kaksi, saavutettavan nimen puuttuminen sekä kohdistuksen ilmaisimen puuttuminen, ja ne nousivat esille testauksen yhteydessä. Varoituksia oli neljä, ja ne kaikki nousivat esille testauksen yhteydessä. Ne olivat puuttuvat videon tekstitykset, puuttuvat audion ja videon kontrollointipainikkeet, HTML-elementtien käyttäminen muualla kuin Html teksti -ohjausobjektissa sekä automaattinen aloitus videossa tai kuvassa. Saavutettavuustarkastajalle määritellyistä ohjeista testauksessa nousi esille seuraavat: näkymän nimen asettaminen kuvaamaan näytöllä olevaa informaatiota, tilasta kertovat indikaation asettaminen päälle sekä näppäimistönavigoinnin järjestyksen tarkistaminen. Yksi määritellyistä ohjeista, vaihtoehtoisen syötteen antaminen Kynäsyöte -ohjausobjektille, ei noussut esille testauksessa, sillä Kynäsyöte-ohjausobjektia ei testattu.

Saavutettavuustarkastajan antamat virheet, varoitukset ja ohjeet eivät olleet kattavia. Saavutettavuustarkastaja ei esimerkiksi antanut virhettä liian matalasta värikontrastista, puuttuvista tai ohitetuista otsikkotasoista tai liian pienestä fonttikoosta (vrt. luku 3.4). Saavutettavuustarkastaja ei myöskään antanut virheitä tai varoituksia, mikäli Power Appsin dokumentaation ohjeistuksia saavutettavuuden parantamiseksi ei ole noudatettu, kuten ohje, että jokaisella sivulla tulisi olla vähintään yksi Teksti-ohjausobjekti otsikkona. Saavutettavuustarkastajan avulla voitiin kuitenkin löytää kriittisiä saavutettavuusongelmia, kuten puuttuvat saavutettavat nimet tai kohdistuksen ilmaisimen puuttuminen.

Power Appsin dokumentaatiosta löytyi paljon ohjeita saavutettavuuden parantamiseen. Dokumentaatiossa kerrottiin yleisesti saavutettavuudesta ja parhaista käytännöistä, jonka lisäksi dokumentaatiossa oli tarkemmat ohjeet esimerkiksi värien käytöstä, loogisen rakenteen luomisesta ruudunlukijoille ja näppäimistönavigoinnille sekä ohjeita dynaamisen sisällön käsittelemisestä saavutettavuuden näkökulmasta. Näiden lisäksi jokaisen ohjausobjektin ohjeissa oli oma kohta saavutettavuudelle, jossa kerrottiin, mitä ohjausobjektin saavutettavuudessa tulee ottaa huomioon, ja ohjeita saavutettavuusominaisuuksien käytöstä. Esimerkiksi Kuva-ohjausobjektissa ohjeistettiin käyttämään saavutettavaa nimeä vain siinä tapauksessa, jossa kuva ei ole vain koristeellinen elementti tai Video-ohjausobjektin automaattinen aloitus kehotettiin ottamaan pois päältä. Dokumentaation saavutettavuusohjeet mukailivat WCAG-saavutettavuusohjeiden periaatteita (ks. luku 3.3).

Dokumentaatiossa nostettiin myös esille tunnistetut rajoitukset koskien saavutettavuutta, jotka nousivat esille myös testauksessa. Tällaisia olivat esimerkiksi, että WAI-ARIA roolit eivät ole tuettuja tai, että näppäimistöllä ei ole mahdollista vierittää sisältöä hiiren sijaan. Power Appsin

dokumentaatiossa ohjeistetaan, että tunnistetut haasteet saavutettavuudessa on mahdollista kiertää toteuttamalla oma kustomoitu ohjausobjekti hyödyntäen HTML, CSS ja TypeScript -kieliä.

5.3 Tulokset

Analyysin perusteella voidaan todeta, että suuri osa havainnoinnin pohjana olleista WCAG-onnistumiskriteerien vaatimuksista täyttyi hyvin tai melko hyvin. Vain muutaman onnistumiskriteerin vaatimukset eivät täyttyneet lainkaan. Huolimatta siitä, että monien ohjausobjektien saavutettavuudessa oli selkeitä puutteita, voidaan kuitenkin todeta, Power Apps -sovelluskehitysympäristössä saavutettavuus on huomioitu ja monet yksinkertaisista ohjausobjekteista oli mahdollista toteuttaa saavutettavuusvaatimusten mukaisesti.

Ohjausobjekteissa sekä niiden ominaisuuksissa oli huomioitu yleisimmät saavutettavuuteen vaikuttavat asiat, kuten saavutettava nimi ja kohdistus sekä ohjausobjekteilla yksilöllisiä spesifejäkin ominaisuuksia. Tiedon ja tilan välittäminen ohjausobjekteista oli enimmäkseen hyvällä tasolla, mutta haasteita aiheutti, se että ohjausobjektien semanttinen tieto ei välttämättä ollut täsmällistä ja tätä korjaamaan ei ollut mahdollista käyttää WAI-ARIA rooleja ja attribuutteja (vrt. luku 3.4). Esimerkkinä saavutettavan nimen antaminen syötekentälle käyttäen WAI-ARIA attribuuttia aria-label tai käyttöliittymäelementin roolin muuttaminen vastaamaan sen tarkoitusta.

Aineiston perusteella voidaan todeta, saavutettavuustarkastajan helppokäyttöisyys sekä automaattinen kehityksenaikainen tarkastus voivat auttaa ongelmakohtien löytämisessä jo aikaisessa vaiheessa (ks. luku 3.4). Toisaalta puutteidensa vuoksi, saavutettavuustarkastaja saattaa kuitenkin antaa virheellisen kuvan siitä, että toteutettu sovellus olisi täysin saavutettavuusvaatimusten mukainen. Saavutettavuustarkastaja toimii hyvänä automaattisena tarkastajana ja apuna saavutettavuuden arvioinnissa ennen manuaalisen testauksen toteuttamista.

6 Pohdinta

Opinnäytetyössä tarkasteltiin low-code -sovelluskehityksen soveltuvuutta menetelmänä, minkä lisäksi tarkasteltiin Microsoftin Power Apps -sovelluskehitysympäristöä ja sen ominaisuuksia saavutettavien sovellusten toteuttamisessa. Tutkimuskysymyksinä oli, soveltuuko low-code -menetelmä teknologiana saavutettavien palveluiden toteuttamiseen, sekä onko Power Apps -sovelluskehitysalustalla mahdollista toteuttaa saavutettavia sovelluksia.

Tutkimus tarjosi kattavan kuvan Power Apps -sovelluskehitysympäristöstä ja sen ominaisuuksista sekä siitä, miten saavutettavuusnäkökulma on huomioitu kyseisessä sovelluskehitysympäristössä. Tulosten perusteella Power Apps -sovelluskehitysympäristössä on mahdollista toteuttaa yksinkertaisia saavutettavia sovelluksia suhteellisen pienellä vaivalla, kun noudatetaan Power Appsin dokumentaatiosta löytyviä ohjeita sekä saavutettavuustarkastajan huomioita. Power Apps voi tutkimuksen perusteella olla hyvä valinta yksinkertaisten sovellusten toteuttamiseen, ja silloin myös low-coden -menetelmään liitetyt hyödyt (nopea kehitys ja tehokkuus), voivat toteutua (ks. luku 2.2). Tulokset osoittivat, että monimutkaisempien toiminnallisuuksien toteuttaminen saavutettavasti voi sen sijaan olla haastavampaa tai vaatia kustomoitujen ohjausobjektien toteuttamista perinteisesti ohjelmoimalla, jolloin voidaan päätyä kirjoittamaan koodia enemmän kuin oli tarkoitus (ks. luku 2.2).

Power Appsin kattavat dokumentaatio yhdessä saavutettavuustarkastajan kanssa, voi yleisellä tasolla parantaa Power Apps -sovelluskehitysympäristössä luotujen sovellusten saavutettavuutta ja olla avuksi esimerkiksi kehittäjille, joilla ei ole kokemusta saavutettavien käyttöliittymien toteuttamisesta (ks. luku 2.3.). Toisaalta saavutettavuuteen perehtynyt kehittäjä on ehkä tottunut käyttämään WAI-ARIA rooleja tai niiden ominaisuuksia välittääkseen käyttäjälle tietoa käyttöliittymäelementin sisällöstä tai tilasta. Power Appsilla on omat saavutettavuusominaisuudet eivätkä niiden nimet vastaa täysin WAI-ARIA ominaisuuksia. Kehittäjä ei voi siis nojautua WAI-ARIAn semantiikkaan ja ohjausobjektien saavutettavuusominaisuuksien löytäminen sekä niiden toiminnan ymmärtäminen vaatii kehittäjältä perehtymistä (vrt. luku 2.2).

Vaikka tutkimus tarjosi hyödyllistä tietoa Power Appsista, on tärkeää todeta, että opinnäytetyön tavoitteet eivät täysin toteutuneet, sillä tutkimustulokset eivät ole suoraan yleistettävissä muiden palveluntarjoajien low-code -sovelluskehitysympäristöihin ja niiden ominaisuuksiin. Tutkimus oli tapaustutkimus yhden low-code -sovelluskehitysympäristön saavutettavuusominaisuuksista ja se antoi tutkimustietoa siitä, miten saavutettavuus on huomioitu tässä sovelluskehitysympäristössä. Tutkimus antoi kuitenkin käsityksen siitä, miten yksi merkittävistä low-code palveluntarjoajista on omaksunut saavutettavuuden periaatteet osaksi sovelluskehitysympäristöään.

Low-code -menetelmästä voidaan etsiä ratkaisua nopeampaan ja ketterämpään sovelluskehitykseen. Low-code -menetelmä madaltaa kynnystä erilaisille henkilöille ilman teknistä taustaa osallistua ohjelmistojen ja sovellusten kehittämiseen (ks. luku 2.4). Tämä voi lisätä monimuotoisuutta sovellusten kehittämisessä, sillä erilaisista taustoista tulevat kehittäjät näkevät asioita eri tavalla, ja tällä voi olla positiivinen vaikutus sovellusten saavutettavuuteen. Toisaalta low-code -alustojen ominaisuuksilla on merkitystä, kun tehdään teknologiavalintoja sovellusprojekteihin. Teknologian valinta luo reunaehdot sille, millaisia tuotteita alustoilla voi kehittää. Low-code -ympäristöjen ominaisuudet vaikuttavat siis myös siihen, onko kehitettävä sovellus mahdollista tehdä saavutettavaksi vai ei.

6.1 Haasteet

Valitun lähestymistavan haasteena oli WCAG-onnistumiskriteerien valinta sekä tutkittavien ohjausobjektien valinta. WCAG-onnistumiskriteereistä valittiin tutkimukseen vain sellaiset kriteerit, jotka olivat teknisesti testattavissa. Tämä jako ei kuitenkaan ollut kaikkien kriteerien kohdalla selkeä. Toinen haaste oli ohjausobjektien valinta tarkasteluun, sillä monia ohjausobjekteja olisi ollut mahdollista testata monia onnistumiskriteereitä vasten. Tämä vaikeutti myös tulosten vertailtavuutta. Valittu lähestymistapa heikentää tulosten luotettavuutta ja tutkimuksen toistettavuutta.

Vaihtoehtoinen lähestymistapa, jossa olisi tarkasteltu Power Appsin jokaista ohjausobjektia yksitellen saavutettavuuden näkökulmasta, olisi tarjonnut selkeämmän ja helpommin toistettavan tutkimusasetelman. Vaihtoehtoisessa lähestymistavassa eri ajankohtina tehty tarkastelu voisi tarjota myös tietoa siitä, miten Power Appsin ohjausobjektien saavutettavuutta on mahdollisesti parannettu.

Tutkimuksen luotettavuuteen vaikutti myös opinnäytetyöntekijän vähäinen kokemus ruudunlukijan käytöstä sekä näppäimistönavigoinnista. Ohjausobjektien saavutettavuuden arvioinnissa olennaisena osana oli manuaalinen testaus ruudunlukijalla sekä näppäimistönavigoinnilla. Opinnäytetyöntekijän vähäinen kokemus avustavien teknologioiden käytöstä saattoi vaikuttaa siihen, miten tarkasti ja systemaattisesti saavutettavuus testattiin sekä siihen, että löytyikö manuaalisessa testauksessa kaikki saavutettavuusvirheet vai johtuivatko virheet, siitä että ruudunlukijaa ei osattu käyttää oikein (ks. luku 3.4).

6.2 Tulosten hyödynnettävyys ja jatkokehitys

Tutkimustuloksista ei voida tehdä yleistyksiä muiden low-code -palveluntarjoajien pyrkimyksistä huomioida saavutettavuutta palveluidensa kehittämisessä. Tutkimustuloksia voidaan kuitenkin hyödyntää apuna teknologia valinnassa, jos vaihtoehtona toteutusteknologiaksi on Power Apps -sovelluskehitysympäristö. Tutkimus voi myös toimia hyvänä arvioinnin pohjana, kun arvioidaan low-code

-menetelmän soveltuvuutta tiettyyn sovelluskehitysprojektiin. Vaikka tulokset eivät olekaan yleistettävissä kaikkiin low-code ympäristöihin, ne tarjoavat näkökulmia, joita voidaan ottaa huomioon päätöksenteossa.

Yksi mielenkiintoinen jatkotutkimusaihe olisi vertailututkimus, jossa tarkasteltaisiin useamman low-code -ympäristön saavutettavuusominaisuuksia sekä niiden tarjoamia mahdollisuuksia tuottaa saavutettavia sovelluksia. Näin saataisiin yleistettävämpiä tuloksia low-code -ympäristöjen saavutettavuusominaisuuksista. Toinen mielenkiintoinen lähestymistapa olisi tarkastella low-code -ympäristöjen käytettävyyttä ja saavutettavuutta itsessään ja tarkastella, miten hyvin erilaiset käyttäjät on huomioitu sovelluskehitysympäristöjen toteuttamisessa, ja miten hyvin sovelluskehitysympäristöä on mahdollista käyttää erilaisten avustavien teknologioiden kanssa.

6.3 Opinnäytetyön arviointi

Opinnäytetyössä onnistuttiin osittain saavuttamaan tavoitteet eli tarkastelemaan low-code menetelmän soveltuvuutta saavutettavien sovellusten toteutuksessa. Tulokset eivät kuitenkaan ole laajasti yleistettävissä vaan tarjoavat lähtökohtaisesti tietoa vain Power Apps -sovelluskehitysympäristöstä.

Opinnäytetyöntekijän henkilökohtaiset tavoitteet täyttyivät hyvin. Opinnäytetyöntekijä oppi opinnäytetyöprojektin aikana paljon low-code -menetelmästä, ja sekä sen hyödyistä ja käyttötapauksista. Opinnäytetyöntekijä syvensi ymmärrystään saavutettavuudesta yleisesti sekä etenkin teknisestä näkökulmasta tutustuessaan WCAG-ohjeistukseen. Tämä kokemus avasi uusia näkökulmia saavutettavuuden merkitykseen ja haasteisiin sovelluskehityksessä, mikä on hyödyllistä hänen omassa työssään.

Opinnäytetyöprojektin edetessä opinnäytetyöntekijä sai arvokasta kokemusta avustavien teknologioiden käytöstä. Opinnäytetyöntekijällä oli ennen opinnäytetyötä vain hieman kokemusta ruudunlukija käytöstä, joten ruudunlukijan käytön harjoittelu vei jonkin verran aikaa tutkimuksen alussa. Navigointi näppäimistön avulla oli myös aluksi hidasta. Opinnäytetyöprojektin edetessä opinnäytetyöntekijä oppi sujuvasti käyttämään avustavia teknologioita, kuten ruudunlukijaa ja näppäimistönavigointi. Ruudunlukijan sujuva käyttö on hyödyllinen taito, jota opinnäytetyöntekijä voi jatkossa hyödyntää työssään.

Lähteet

Aluehallintovirasto s.a. a. Yleistä saavutettavuudesta. Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/ Luettu: 19.3.2024.

Aluehallintovirasto s.a. b. Kenelle saavutettavuus on tärkeää? Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa/ Luettu: 19.3.2024.

Aluehallintovirasto s.a. c. Muutokset digipalvelulakiin. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/muutokset-digipalvelulakiin/ Luettu: 31.3.2024.

Aluehallintovirasto s.a. d. Digipalvelulain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/ Luettu: 24.3.2024.

Aluehallintovirasto s.a. e. Tietoa WCAG-ohjeistuksesta. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/ Luettu: 7.1.2024.

Bock, A. C. & Frank, U. 2021. In Search of the Essence of Low-Code: An Exploratory Study of Seven Development Platforms. ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2021, s. 57–66.

EN 301 549:2021. Accessibility requirements for ICT products and services. Cen, Cenelec, Etsi. Brussels, Sophia Antipolis Cedex.

Eriksson, P. & Koistinen, K. 2014. Monenlainen tapaustutkimus. Kuluttajatutkimuskeskus. Helsinki. E-kirja. Luettu: 9.4.2024.

Garcia, K. 2021. What is low code? Definition, use cases, and benefits. Luettavissa: https://retool.com/blog/what-is-low-code Luettu: 14.4.2024.

Georgakas, D. 2023. A11Y Unraveled: Become a Web Accessibility Ninja. Apress L. P. Berkeley, CA. E-kirja. Luettu: 23.3.2024.

Hirzel, M. 2023. Low-code programming models. Communications of the ACM, 66, 10, s.76–85.

Hughes, C. 2022. 4 security concerns for low-code and no-code development. CSO. Luettavissa: https://www.csoonline.com/article/572021/4-security-concerns-for-low-code-and-no-code-development-2.html Luettu: 13.4.2024.

Kalbag, L. 2017. Accessibility for Everyone. A Book Apart. New York, New York. E-kirja. Luettu: 25.3.2024.

Kehitysvammaliitto ry 2023a. Kuka hyötyy saavutettavuudesta? Luettavissa: https://papunet.net/saavutettavuus/miksi-saavutettava/kuka-hyotyy-saavutettavuudesta/ Luettu: 20.3.2024.

Kehitysvammaliitto ry 2023b. Saavutettavuuden testaaminen itse. Luettavissa: https://papunet.net/saavutettavuus/ohjeita-ja-oppaita/saavutettavuuden-arviointi/saavutettavuuden-testaaminen-itse/ Luettu: 24.3.2024.

Kehitysvammaliitto ry 2023c. Käyttäjätestaaminen. Luettavissa: https://papunet.net/saavutettavuus/ohjeita-ja-oppaita/saavutettavuuden-arviointi/kayttajatestaaminen/ Luettu: 24.3.2024.

Kehitysvammaliitto ry 2024a. Avustavat teknologiat. Luettavissa: https://papunet.net/saavutettavuus/miksi-saavutettava/avustavat-teknologiat/ Luettu: 24.3.2034.

Kehitysvammaliitto ry 2024b. Verkkosivujen helppokäyttöisyys. Luettavissa: https://papunet.net/saavutettavuus/ohjeita-ja-oppaita/verkkosivujen-helppokayttoisyys/ Luettu: 29.3.2024.

Kuntaliitto 2017. Kuntien saavutettavuusopas: Verkkopalveluiden saavutettavuuden testaaminen. Luettavissa: https://www.kuntaliitto.fi/julkaisut/saavutettavuusopas/7-verkkopalveluiden-saavutettavuuden-testaaminen Luettu: 23.3.2024.

Laki digitaalisten palvelujen tarjoamisesta annetun lain muuttamisesta 19.1.2023/104.

Laki digitaalisten palvelujen tarjoamisesta 15.3.2019/306.

Matvitskyy, O., Iijima, K., West, M., Davis, K., Jain, A. & Vincent, P. 2023. Magic Quadrant for Enterprise Low-Code Application Platforms. Gartner Research.

Microsoft 22.3.2024. Microsoft Power Fx overview. Luettavissa: https://learn.microsoft.com/en-us/power-platform/power-fx/overview Luettu: 18.4.2024.

Microsoft 22.2.2024. Understand Power Apps Studio. Luettavissa: https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/power-apps-studio Luettu: 19.4.2024.

Niiranen, J. 6.4.2021. Democratizing code. Jukka Niiranen. Luettavissa: https://jukkaniiranen.com/2021/04/democratizing-code/. Luettu: 7.2.2024.

Näkövammaisten liitto ry 2023. Saavutettavuuden testaaminen. Luettavissa: https://www.nakovammaistenliitto.fi/fi/verkkosivujen-saavutettavuus Luettu: 24.3.2024.

Project Management Institute 2021. Citizen Development: The Handbook for Creators and Changemakers. Project Management Institute. Pennsylvania. E-kirja. Luettu: 7.2.2024.

Pinho, D., Aguiar, A. & Amaral, V. 2023. What about the usability in low-code platforms? A systematic literature review. Journal of Computer Languages, 74, January 2023.

Saaranen-Kauppinen, A. & Puusniekka, A. 2006. Havainnointi. Teoksessa Saaranen-Kauppinen, A. & Puusniekka, A. (toim.). KvantiMOTV - Menetelmäopetuksen tietovaranto. Yhteiskuntatieteellinen tietoarkisto. Tampere. Verkkojulkaisu. Luettavissa: https://www.fsd.tuni.fi/menetelmaopetus/kvali/L6_4.html Luettu: 9.4.2024.

Selkeästi meille -hanke 2023. Mitä on kognitiivinen saavutettavuus? Luettavissa: https://www.selkeastimeille.fi/kognitiivinen-saavutettavuus/mita-on-kognitiivinen-saavutettavuus/ Luettu: 31.3.2024.

Vilkka, H. 2021. Tutki ja kehitä. 5. painos. PS-kustannus. Jyväskylä. E-kirja. Luettu: 12.4.2024.

Vuori, J. 2021. Tapaustutkimus. Teoksessa Vuori, J. (toim.). Laadullisen tutkimuksen verkkokäsikirja. Yhteiskuntatieteellinen tietoarkisto. Tampere. E-kirja. Luettavissa: https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvali/tutkimusasetelma/tapaustutkimus/ Luettu: 9.4.2024.

Woo, M. 2020. The Rise of No/Low Code Software Development—No Experience Needed? Engineering, 6, 9, s. 960–961.

World Wide Web -konsortio 2.5.2024. Introduction to Understanding WCAG. Luettavissa: https://www.w3.org/WAI/WCAG22/Understanding/intro#understanding-the-four-principles-of-accessibility. Luettu: 13.5.2024.

World Wide Web -konsortio 7.3.2024. WCAG 2 Overview. Luettavissa: https://www.w3.org/WAI/standards-guidelines/wcag/. Luettu: 20.3.2024.

World Wide Web -konsortio 2023. Web Content Accessibilty Guidelines (WCAG) 2.2. Luettavissa: https://www.w3.org/TR/WCAG22/ Luettu: 1.4.2024.

World Wide Web -konsortio 2019. Verkkosisällön saavutettavuusohjeet (WCAG) 2.1. Suomentanut Kehitysvammalitto ry. Luettavissa: https://www.w3.org/Translations/WCAG21-fi/ Luettu: 18.4.2024.

Yhdenvertaisuuslaki 30.1.2014/1325.

Yleissopimus vammaisten henkilöiden oikeuksista 2016/27.

Liitteet

Liite 1. Power Apps -sovelluskehitysympäristön testauslomake

WCAG Ohje: WCAG ohjeen järjestysnumero sekä kuvaus

Onnistumiskriteeri: WCAG-ohjeen onnistumiskriteerin järjestysnumero sekä kuvaus

Ohjausob-jekti	**Ohjausobjektin nimi (suomennettu nimi)**	**Arviointi***
Ominaisuudet	Löytyykö elementiltä ominaisuuksia, joiden avulla voidaan vastata kriteerin täyttymiseen?	1
Tarkastus	Power Apps -alustan oman saavutettavuustarkastajan antamat virheet ja huomiot.	1
Ruudunlukija tai muu työkalu	Ruudunlukijalla tai muulla avustavalla työkalulla tehdyn testin tulokset.	1
Huomioita	Huomioita testauksessa. Power Apps -kehitysalustan oma dokumentaatio aiheeseen liittyen.	-

* Kriteerin täyttymisen arviointi asteikolla, jossa 1=Täyttyi, 2=Täyttyi osittain, 3=Ei täyttynyt

Liite 2. Testausaineisto

WCAG Ohje: ”1.1 Tekstivastineet - Tarjoa tekstivastineet kaikelle ei-tekstuaaliselle sisällölle” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.1.1 Ei-tekstuaalinen sisältö - Kaikki käyttäjälle esitettävä ei-tekstuaalinen sisältö on varustettu saman tarpeen täyttävällä tekstivastineella” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Image (kuva)**	**Arviointi**
Ominaisuudet	AccessibleLable (saavutettava nimi). Ruudunlukijalle annettava tekstivastine.	1
Tarkastus	Jos kuvaa käytettiin interaktiivisena elementtinä, tarkastaja antoi virheen puuttuvasta tekstivastineesta. Kun kuva ei ollut interaktiivinen elementti, ei virhettä puuttuvasta tekstivastineesta.	2
Ruudunlukija tai muu työkalu	Ruudunlukija luki annetun tekstivastineen, muuten ohitti ohjausobjektin.	1
Huomioita	Dokumentaatiossa mainitaan, että AccessibleLabel on ruudunlukijaa varten. Mainita myös siitä, että jos kuva on vain koristeellinen elementti, tekstivastineen tulisi olla tyhjä.	

Ohjausobjekti	**Icon (kuvake)**	**Arviointi**
Ominaisuudet	AccessibleLable (saavutettava nimi). Ruudunlukijalle annettava tesktivastine.	1
Tarkastus	Jos kuvaketta käytettiin interaktiivisena elementtinä, tarkastaja antoi virheen puuttuvasta tekstivastineesta. Kun kuvake ei ollut interaktiivinen elementti, ei virhettä puuttuvasta tekstivastineesta.	2
Ruudunlukija tai muu työkalu	Ruudunlukija luki annetun tekstivastineen, muuten ohitti ohjausobjektin.	1
Huomioita	Dokumentaatiossa mainitaan, että AccessibleLabel on ruudunlukijaa varten. Mainita myös siitä, että jos kuvake on vain koristeellinen elementti, tekstivastineen tulisi olla tyhjä.	

Ohjausobjekti	**Video (video)**	**Arviointi**
Ominaisuudet	AccessibleLable (saavutettava nimi). Ruudunlukijalle annettava tekstivastine.	1
Tarkastus	Virhe, jos videolla ei ollut tekstivastinetta.	1
Ruudunlukija tai muu työkalu	Jos tekstivastine oli annettu, ruudunlukija luki sen.	1
Huomioita	Dokumentaatiossa mainitaan, että AccessibleLabel on ruudunlukijaa varten ja, että lyhyt videon kuvaus tulisi sijoittaa tähän. Jos kuvaus videosta on pitkä, ohjeistettiin käyttämään Label-ohjausobjektia ja sijoittamaan se videon läheisyyteen.	

Ohjausobjekti	**Text Input (tekstisyöte)**	**Arviointi**
Ominaisuudet	AccessibleLable (saavutettava nimi)	1
Tarkastus	Virhe, jos tekstisyötteellä ei ollut tekstivastinetta.	1
Ruudunlukija tai muu työkalu	Ruudunlukija luki annetun tekstivastineen.	1
Huomioita	Dokumentaatiossa ohjeistetaan, että saavutettava nimi on oltava ruudunlukijaa varten.	

WCAG Ohje: ”1.2 Aikasidonnainen media - Tarjoa vastine aikasidonnaiselle medialle” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.2.2 Tekstitys (tallennettu) - Kaikelle synkronoidussa mediassa olevalle tallennetulle audiosisällölle on tarjolla tekstitys, paitsi kun media on tekstin mediavastine ja selvästi merkitty sellaiseksi” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Video (video)**	**Arviointi**
Ominaisuudet	ClosedCaptionsUrl (tekstityksen osoite). Kenttä, johon voi lisätä tekstityksen osoitteen.	1
Tarkastus	Jos videolle ei asetettu tekstityksen osoitetta, antoi virheen puuttuvasta tekstityksestä.	1
Ruudunlukija tai muu työkalu	Tekstitys ei näkynyt videolla automaattisesti vaan se piti erikseen laittaa päälle videosoittimesta.	2
Huomioita	Dokumentaatiossa mainitaan, että videoille on lisättävä tekstitykset.	

WCAG Ohje: ”1.3 Mukautettava - Tuota sisältöä, joka voidaan esittää eri tavoin (esimerkiksi yksinkertaisemman asettelun avulla) ilman sisällön tai rakenteen menettämistä” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.3.1 Informaatio ja suhteet - Esitystavassa välittyvät informaatio, rakenne ja suhteet voidaan selvittää ohjelmallisesti tai ne ovat saatavilla tekstinä” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Container (säiliö)**	**Arviointi**
Ominaisuudet	Säiliölle ei ole mahdollista antaa tunnistetta tai roolia, joka kuvaisi sivun rakenteen eri alueita, kuten navigointi (navigation) tai pääsisältö (main).	3
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukijalla ei ollut mahdollista siirtyä sivun eri rakenteellisiin alueisiin käyttäen tunnisteita.	3
Huomioita	Dokumentaatiossa kerrotaan, että Power Apps ei tue HTML-elementtien kaltaisia alueita <header> tai <nav> eikä myöskään WAI-ARIA maamerkkejä, kuten banner tai navigation, joilla sivun rakennetta voisi esittää ohjelmallisesti.	

Ohjausobjekti	**Text label (teksti)**	**Arviointi**
Ominaisuudet	Role (rooli). Roolin avulla mahdollista määritellä otsikkotasot 1–4 (heading 1–4) ohjelmallisesti. Tekstiä ei ole mahdollista liittää ohjelmallisesti esimerkiksi käyttäjän syötteelle tarkoitettuun kenttään.	2
Tarkastus	Ei varoittanut, vaikka teksti puuttui sivulta (ks. Huomioita).	3
Ruudunlukija tai muu työkalu	Ruudunlukija luki otsikolle asetetun roolin (otsikkotason). Ruudunlukijan valikon avulla oli mahdollista navigoida otsikkoihin, joille oli annettu rooli.	1
Huomioita	Dokumentaatiossa mainitaan roolin olevan ruudunlukijalle, ja että jokaisella sivulla pitäisi olla ainakin yksi teksti.	

Ohjausobjekti	**Gallery (galleria)**	**Arviointi**
Ominaisuudet	AccessibleLable (saavutettava nimi), ItemAccessibleLable (saavutettava nimi kuvaamaan gallerian elementtiä).	1
Tarkastus	Virhe, mikäli gallerialla ei ollut asetettu saavutettavaa nimeä. Mutta ei virhettä, jos listan elementin saavutettava nimi puuttuu.	2
Ruudunlukija tai muu työkalu	Ruudunlukija saa galleriasta informaatiota rakenteesta ja suhteista, kuten gallerian elementtien lukumäärän sekä listan elementin järjestysnumeron.	1
Huomioita	Galleriassa ei mahdollista valita, että onko järjestetty vai järjestämätön lista.	

Ohjausobjekti	**Radio (valintanappi)**	**Arviointi**
Ominaisuudet	Valintanappi ryhmitellään automaattisesti ryhmäksi, joka sisältää valintanapit. Valintanappiryhmälle voi antaa saavutettavan nimen.	1
Tarkastus	Virhe, mikäli saavutettava nimi puuttui. Ei virhettä, jos listalla tyhjiä elementtejä (ks. huomioita).	2
Ruudunlukija tai muu työkalu	Ruudunlukija saa valintanappiryhmästä informaatiota rakenteesta ja suhteista, kuten valintanappiryhmän valintavaihtoehtojen lukumäärän, listan elementin järjestysnumeron sekä onko valintanappi valittuna.	1
Huomioita	Dokumentaatiossa ohjeistetaan varmistamaan ruudunlukijaa varten, että listalla ei ole tyhjiä valintavaihtoehtoja. Ohjeistetaan lisäämään otsikko valintanapin läheisyyteen.	

WCAG Ohje: ”1.3 Mukautettava - Tuota sisältöä, joka voidaan esittää eri tavoin (esimerkiksi yksinkertaisemman asettelun avulla) ilman sisällön tai rakenteen menettämistä” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.3.2 Merkitykseen vaikuttava järjestys - Kun sisällön esitysjärjestys vaikuttaa sisällön merkitykseen, oikea lukemisjärjestys voidaan selvittää ohjelmallisesti” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Image (Kuva)**	**Arviointi**
Ominaisuudet	TabIndexillä mahdollista muuttaa lukemisjärjestystä ohjelmallisesti.	1
Tarkastus	Jos TabIndex oli suurempi kuin 0, tarkastuksessa kehotetaan tarkastamaan elementtien järjestys näytöllä, sekä välttämään kohdistuksen muuttamista TabIndexillä.	1
Ruudunlukija tai muu työkalu	Jos TabIndex oli suurempi kuin 0, niin näppäimistönavigoinnilla siirryttiin ensimmäisenä tähän elementtiin.	1
Huomioita	Dokumentaatiossa myös kehotetaan välttämään elementtien ohjelmallisen järjestyksen määrittämistä suoraan elementille (TabIndexin) vaan tarkastelemaan rakennetta.	

Ohjausobjekti	**Container (säiliö)**	**Arviointi**
Ominaisuudet	Säiliöllä mahdollista muuttaa elementtien ohjelmallista järjestystä.	1
Tarkastus	Jos TabIndex oli suurempi kuin 0, tarkastuksessa kehotetaan tarkastamaan elementtien järjestys näytöllä, sekä välttämään kohdistuksen muuttamista TabIndexillä.	1
Ruudunlukija tai muu työkalu	Säiliön avulla voitiin vaikuttaa ruudunlukijan sekä näppäimistönavigoinnin kohdistusjärjestykseen.	1
Huomioita	Power Appsin omassa dokumentaatiossa vinkkejä, miten säiliön avulla, voidaan ohjelmallisesti vaihtaa kohdistuksen järjestystä ruudunlukijalle.	

WCAG Ohje: ”1.3 Mukautettava - Tuota sisältöä, joka voidaan esittää eri tavoin (esimerkiksi yksinkertaisemman asettelun avulla) ilman sisällön tai rakenteen menettämistä” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.3.4 Asento - Sisältöä ei ole rajoitettu vain tiettyyn näyttölaitteen asentoon kuten pysty- tai vaakasuuntaan, lukuun ottamatta tapausta, jossa tietty asento on olennainen” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Container (säiliö)**	**Arviointi**
Ominaisuudet	Säiliöiden avulla sisältö voidaan asetella automaattisesti täyttämään saatavilla olevan tilan ja näin tehdä responsiiviseksi. Ohjausobjektien sijaintia ei voi tällöin muuttaa manuaalisesti, joten se aiheuttaa rajoitteita elementtien sijoitteluun.	2
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Sisältö järjestyi uudelleen, kun simuloitiin erilaisten laitteiden kokoa tai asentoa.	1
Huomioita	Power Appsin omassa dokumentaatiossa mainitaan, että ohjausobjektien sijaintia ei voi muuttaa manuaalisesti, jos käytetään automaattista asettelua säiliössä.	

WCAG Ohje: ”1.4 Erottuva - Helpota käyttäjiä näkemään ja kuulemaan sisältö sekä erota etuala taustasta” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.4.2 Audion kontrollointi - Jos jokin ääni verkkosivulla soi automaattisesti kauemmin kuin kolme sekuntia, käytettävissä on joko mekanismi äänen keskeyttämiseen tai pysäyttämiseen tai mekanismi äänen voimakkuuden säätämiseksi koko järjestelmän äänenvoimakkuuden tasosta riippumatta” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Audio (audio)**	**Arviointi**
Ominaisuudet	Audiolla audion automaattisen aloittamisen voi ottaa pois päältä. Audiolle on mahdollista laittaa päälle painikkeen audion hallinnointia varten, kuten pysäytys ja äänenvoimakkuus. Ei ole mahdollista lisätä erillistä painiketta audion pysäyttämiseen, jonka voisi sijoittaa sivun yläosaan. Ei mahdollista myöskään ajastaa automaattista pysäytystä 3 sekunnin jälkeen.	2
Tarkastus	Tarkastus antaa varoituksen, mikäli automaattinen audion aloitus on päällä. Antaa varoituksen, jos hallinnointiin tarkoitetut painikkeet on otettu pois päältä.	1
Ruudunlukija tai muu työkalu	Ruudunlukijalla ja näppäimistönavigoinnilla audion pysäyttäminen vaikeaa, sillä audion kontrolloinnin painikkeet olivat ohjausobjektin sisällä.	2
Huomioita	Power Appsin omassa dokumentaatiossa mainitaan, että automaattinen aloitus pitäisi olla pois.	

WCAG Ohje: ”1.4 Erottuva - Helpota käyttäjiä näkemään ja kuulemaan sisältö sekä erota etuala taustasta” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”1.4.3 Kontrasti (minimi) - Tekstin ja tekstiä esittävien kuvien visuaalisen esitystavan kontrastisuhde on vähintään 4,5:1” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Text label (teksti)**	**Arviointi**
Ominaisuudet	Tekstille mahdollista muokata tekstin sekä taustan väriä.	1
Tarkastus	Tarkastus ei anna varoitusta kontrastista, vaikka teksti ja tausta olisivat samalla värillä.	3
Ruudunlukija tai muu työkalu	Kontrasti mahdollista tarkastaa erillisellä työkalulla vain manuaalisesti valitsemalla pipetillä värit halutuista elementeistä.	2
Huomioita	Mainitiin, että tekstin ja taustan välillä tulee olla riittävä kontrasti.	

WCAG Ohje: ”2.1 Käytettävissä näppäimistöltä - Toteuta kaikki toiminnallisuus siten, että se on käytettävissä näppäimistöltä” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”2.1.1 Näppäimistö - Kaikki sisällön toiminnallisuus on hallittavissa näppäimistörajapinnan välityksellä ilman vaatimusta yksittäisten näppäinpainallusten erityisestä ajoittamisesta, paitsi kun taustalla oleva toiminnallisuus vaatii syötettä, joka riippuu käyttäjän liikkeiden reitistä eikä vain päätepisteistä” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Image (kuva)**	**Arviointi**
Ominaisuudet	Mikäli kuvaa käytettiin interaktiivisena elementtinä, jota voi klikata, kuvalle voi lisätä TabIndexin, jotta näppäimistönavigoinnilla voidaan pysähytä siihen.	1
Tarkastus	Tarkastus antoi virheen, jos kuvaa käytettiin painikkeena. Ehdotti TabIndexin lisäämistä, jota kehotettiin yleisesti välttämään.	2
Ruudunlukija tai muu työkalu	Näppäimistöselauksella ei voitu pysähtyä kuvaan, jos sille ei lisätty TabIndexiä. Pysähtyminen sekä klikkaaminen onnistui, jos TabIndex asetettu.	1
Huomioita	Dokumentaatiossa, että TabIndexiä ei pitäisi käyttää, jotta näppäimistönavigointi pysähtyy elementtiin, vaan käyttää ensisijaisesti ohjausobjekteja, jotka on tarkoitettu interaktiivisiksi.	

Ohjausobjekti	**Screen (näyttö)**	**Arviointi**
Ominaisuudet	Näytöllä ei ollut ominaisuuksia, jotta sisällöstä voisi tehdä vieritettävän näppäimistölle.	3
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Näyttöä oli mahdollista vierittää hiirellä, mutta näppäimistöselauksella ei ollut mahdollista vierittää sivua ylös tai alas.	3
Huomioita	Dokumentaatiossa maininta, että näppäimistönavigoinnilla ei ole mahdollista selata vieritettäviä sivuja tai lomakkeita.	

Ohjausobjekti	**Date Picker (Kalenterivalitsin)**	**Arviointi**
Ominaisuudet	Mahdollista selata näppäimistöllä.	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Kalenterivalitsimessa pystyi selaamaan päiviä nuolinäppäimien avulla. Kuukausia ja vuosia oli mahdollista selata sivun ylös ja alas -painikkeilla.	1
Huomioita	Maininta siitä, miten kuukausia ja vuosia mahdollista selata näppäimistön avulla.	

Ohjausobjekti	**Rich Text Editor (Tekstieditori)**	**Arviointi**
Ominaisuudet	Tekstieditorilla ei ominaisuuksia, jolla voisi parantaa näppäimistön avulla toimintojen tekemistä.	3
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Näppäimistönavigoinnin avulla pystyi lisäämään tekstiä, mutta ei tekemään sille muutoksia, kuten lisäämään vahvennusta tai luetteloa.	2
Huomioita		

WCAG Ohje: ”2.1 Käytettävissä näppäimistöltä - Toteuta kaikki toiminnallisuus siten, että se on käytettävissä näppäimistöltä” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”2.1.2 Ei näppäimistöansaa - Jos kohdistus voidaan siirtää sivun komponenttiin näppäimistörajapinnan kautta, niin kohdistus voidaan siirtää myös pois kyseiseltä komponentilta pelkästään näppäimistörajapintaa käyttämällä. Mikäli tämä vaatii muuta kuin pelkkien nuoli- tai tab-näppäimien tai muiden standardinmukaisten poistumismenetelmien käyttämistä, käyttäjälle neuvotaan menetelmä kohdistuksen poissiirtämiseksi” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Combo Box (yhdistelmä laatikko)**	**Arviointi**
Ominaisuudet	Combo Box ohjausobjektilla ei ominaisuuksia, jolla näppäimistöansan saisi ratkaistua.	3
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Näppäimistö navigoinnilla ei pääse yhdistelmä laatikosta ulos, kun se on avattu.	3
Huomioita	Power Apps dokumentaatiossa mainitaan, että Combo Box ei ole saavutettava.	

WCAG Ohje: ”2.4 Navigoitava - Tarjoa käyttäjille tapoja navigoida, etsiä sisältöä ja määrittää sijaintinsa” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”2.4.1 Ohita lohkot - Tarjolla on mekanismi sellaisten sisällön lohkojen ohittamiseen, jotka toistuvat useilla verkkosivuilla” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Button (painike)**	**Arviointi**
Ominaisuudet	SetFocus (aseta kohdistus) funktion voi asettaa OnSelect (klikattaessa) ominaisuudelle. Funktiolla kohdistuksen voi asettaa interaktiivisiin elementteihin tai tekstiin.	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Painiketta klikatessa kohdistus siirtyy haluttuun paikkaan, jolloin käyttäjä voi ohittaa lohkoja näppäimistöllä navigoidessa.	1
Huomioita		

WCAG Ohje: ”2.4 Navigoitava - Tarjoa käyttäjille tapoja navigoida, etsiä sisältöä ja määrittää sijaintinsa” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”2.4.7 Näkyvä kohdistus - Kaikilla näppäimistöltä käytettävillä käyttöliittymillä on käyttötila, jossa näppäimistön kohdistuksen ilmaisin on näkyvissä” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Button (painike)**	**Arviointi**
Ominaisuudet	FocusedBorderColor (kohdistuksen ilmaisimen väri) ja FocusedBorderThickness (kohdistuksen ilmaisimen paksuus). Kohdistuksen ilmaisimen väriä ja paksuutta on mahdollista muokata.	1
Tarkastus	Tarkastus antoi virheen, jos kohdistimen ilmaisimen paksuus oli asetettu arvoon 0. Ei kuitenkaan antanut virhettä, jos kohdistimen ilmaisimen väri oli sama kuin elementillä, jolloin kohdistu ei erotu.	2
Ruudunlukija tai muu työkalu	Kohdistin oli helposti erotettavissa, mikäli paksuus ja väri oli asetettu.	1
Huomioita	Dokumentaatiossa mainitaan, että kohdistuksen ilmaisimen sekä taustan välillä tulee olla riittävä kontrasti.	

WCAG Ohje: ”2.5 Syötetavat - Tee toimintojen käyttämisestä käyttäjille helpompaa erilaisilla syötetavoilla näppäimistön lisäksi” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”2.5.1 Osoitineleet - Kaikkia toimintoja, joissa hyödynnetään monipiste- tai reittiin perustuvia ohjauseleitä, voidaan käyttää myös yhdellä osoittimella ja ilman reittiin perustuvaa elettä, paitsi jos kyseinen ohjaustapa on olennainen” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Slider (liukusäädin)**	**Arviointi**
Ominaisuudet	Liukusäädintä mahdollista klikata.	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Liukusäätimen arvon voi asettaa raahaamisen sijaan klikkaamalla. Liukusäädintä oli mahdollista liikuttaa myös nuolinäppäinten avulla.	1
Huomioita		

Ohjausobjekti	**Map (kartta)**	**Arviointi**
Ominaisuudet	Kartan kontrolloimiseen tarkoitetut painikkeet. Painikkeet mahdollista ottaa pois käytöstä tai ottaa käyttöön. Kartalla omat painikkeet, lähentämiseen ja loitontamiseen sekä kallistuksen muokkaamiseen.	1
Tarkastus	Ei virhettä, vaikka kontrollointiin tarkoitetut painikkeet pois päältä.	3
Ruudunlukija tai muu työkalu	Karttaa oli mahdollista loitontaa, lähentää ja kallistaa painikkeilla. Kartalla oli madollista liikkua tuplaklikkaamalla halutusta paikasta. Näppäimistönavigointi mahdollinen.	1
Huomioita		

Ohjausobjekti	**3D Object (3D objekti)**	**Arviointi**
Ominaisuudet	Ohjausobjektilla ei elementin kontrollointiin tarkoitettuja painikkeita.	3
Tarkastus	-	-
Ruudunlukija tai muu työkalu	3D objektia oli mahdollista liikutella nuolinäppäinten avulla. Hiirellä vaadittiin, että painiketta pidettiin pohjassa samalla kun liikutettiin.	2
Huomioita		

WCAG Ohje: ”2.5 Syötetavat - Tee toimintojen käyttämisestä käyttäjille helpompaa erilaisilla syötetavoilla näppäimistön lisäksi” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”2.5.3 Nimilappu nimessä - Tapauksissa, joissa käyttöliittymäkomponentin nimilapussa on tekstiä tai tekstiä esittävä kuva, komponentin nimi sisältää sen tekstin, joka on visuaalisesti näkyvissä” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Button (painike)**	**Arviointi**
Ominaisuudet	Painikkeella ei ole AccessibleLable ominaisuutta. Painikkeessa näkyvä visuaalinen nimi ainoa tapa kertoa, mitä painikkeen klikkaamisesta tapahtuu.	3
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija lukee painikkeen näkyvän nimen.	1
Huomioita		

WCAG Ohje: ”3.1 Luettava - Tee tekstisisällöstä luettavaa ja ymmärrettävää” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”3.1.1 Sivun kieli - Jokaisen verkkosivun oletusarvoinen luonnollinen kieli voidaan selvittää ohjelmallisesti” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Screen (näyttö)**	**Arviointi**
Ominaisuudet	Sovelluksen näytölle voi asettaa ContentLanguage (sisällön kieli).	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija tunnistaa sisällön kielen sekä osaa lukea sen oikealla kielellä.	1
Huomioita	Dokumentaatiossa mainitaan, että ohjausobjektien sisällön kielivalintaa voi muuttaa ruudunlukijaa varten.	

WCAG Ohje: ” 3.1 Luettava - Tee tekstisisällöstä luettavaa ja ymmärrettävää” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”3.1.2 Osien kieli - Sisällön jokaisen tekstikatkelman tai ilmaisun luonnollinen kieli voidaan selvittää ohjelmallisesti, paitsi seuraavien osalta: erisnimet, tekniset termit, määrittämättömän kielen sanat sekä sanat tai ilmaisut, jotka ovat muuttuneet läheisen tekstiympäristön kielen murteelliseksi osaksi” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Text label (teksti)**	**Arviointi**
Ominaisuudet	ContentLanguage (sisällön kieli).	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija tunnistaa sisällön kielen sekä osaa lukea sen oikealla kielellä.	1
Huomioita	Dokumentaatiossa mainitaan, että ohjausobjektien sisällön kielivalintaa voi muuttaa ruudunlukijaa varten.	

WCAG Ohje: ”3.3 Syötteen avustaminen - Auta käyttäjiä välttämään ja korjaamaan virheitä” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”3.3.1 Virheen tunnistaminen - Jos syötevirhe havaitaan automaattisesti, virheellinen kohta osoitetaan ja virhe kuvataan käyttäjälle tekstimuotoisena” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Text label (teksti)**	**Arviointi**
Ominaisuudet	Tekstillä on ominaisuus Live (live) ominaisuus, jolloin ruudunlukija lukee dynaamisesti päivittyvän tekstin. Teksti mahdollista päivittää tekstisyötteen validoinnin jälkeen.	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija lukee dynaamisesti päivittyvän virheviestin, jos validointi on mennyt virheeseen.	1
Huomioita	Dokumentaatiossa ohjeita dynaamisen sisällön käsittelyyn saavutettavasti.	

WCAG Ohje: ”4.1 Yhteensopiva - Maksimoi yhteensopivuus nykyisten ja tulevien käyttäjäagenttien, mukaan lukien avustavien teknologioiden, kanssa” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”4.1.2 Nimi, rooli, arvo - Kaikkien käyttöliittymäkomponenttien (mm. lomake-elementit, linkit ja skriptien luomat komponentit) nimi ja rooli voidaan selvittää ohjelmallisesti tilat, ominaisuudet ja arvot, jotka käyttäjä voi asettaa, voidaan myös asettaa ohjelmallisesti ja tieto näiden muutoksista on käyttäjäagenttien, mukaan lukien avustavien teknologioiden, saatavissa” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Icon (kuvake)**	**Arviointi**
Ominaisuudet	Jos kuvaketta käytettiin interaktiivisena painikkeena, rooli määräytyi automaattisesti painikkeeksi.	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija tunnistaa kuvakkeen painikkeeksi, jos se on klikattava.	1
Huomioita		

Ohjausobjekti	**Slider (liukusäädin)**	**Arviointi**
Ominaisuudet	Ominaisuus ShowValue (näytä arvo), jolla asetetaan liukusäätimen arvo näkyväksi myös ruudunlukijalle.	1
Tarkastus	Jos ominaisuus otettu pois päältä, tarkastaja antaa vinkin, että ilman tätä käyttäjät eivät voi tietää liukusäätimen arvoa.	1
Ruudunlukija tai muu työkalu	Show value kertoo ruudunlukijalle liukusäätimen tilan/arvon.	1
Huomioita	Dokumentaatiossa maininta, että ShowValue on oltava true.	

Ohjausobjekti	**Text label (teksti)**	**Arviointi**
Ominaisuudet	Tekstin ominaisuudelle OnSelect (klikattaessa) voidaan antaa funktio, joka avaa sivun, jolloin teksti toimii linkkinä. Voidaan antaa saavutettava nimi, mutta ei muuttaa roolia (role=link).	2
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija lukee saavutettavan nimen. Teksti näyttäytyy painikkeena, ei linkkinä.	2
Huomioita	Dokumentaatiossa mainita, että WAI-ARIA roolit ei ole tuettuja.	

WCAG Ohje: ”4.1 Yhteensopiva - Maksimoi yhteensopivuus nykyisten ja tulevien käyttäjäagenttien, mukaan lukien avustavien teknologioiden, kanssa” (World Wide Web -konsortio 2019).

Onnistumiskriteeri: ”4.1.3 Tilasta kertovat viestit - Sisällössä, joka on toteutettu käyttäen merkkauskieliä, tilasta kertovat viestit voidaan selvittää ohjelmallisesti sellaisen roolin tai ominaisuuksien avulla, jotka mahdollistavat viestin esittämisen käyttäjälle avustavan teknologian avulla ilman kohdistuksen siirtämistä” (World Wide Web -konsortio 2019).

Ohjausobjekti	**Text label (teksti)**	**Arviointi**
Ominaisuudet	Tekstillä on ominaisuus Live (live) ominaisuus, joka ilmaisee dynaamisesti päivittyvän tekstin.	1
Tarkastus	-	-
Ruudunlukija tai muu työkalu	Ruudunlukija lukee dynaamisesti päivittyvän virheviestin, jos validointi on mennyt virheeseen.	1
Huomioita	Dokumentaatiossa ohjeistetaan, että ohjausobjektin pitää olla asetettu näkyväksi ja Live ominaisuus tulee olla päällä koko ajan, jotta ruudunlukija tunnistaa sen.	