Saavutettavuusdirektiivin käytäntöjen luominen ja automatisointi osaksi verkkosivujen kehitysprosessia

Linnea Myllynen

Haaga-Helia ammattikorkeakoulu

Tradenomi

Opinnäytetyö

2024

Tiivistelmä

Tekijä(t)
Linnea Myllynen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Saavutettavuusdirektiivin käytäntöjen luominen ja automatisointi osaksi verkkosivujen kehitysprosessia

Sivu- ja liitesivumäärä
39

Tämän toiminnallisen opinnäytetyön tavoitteena oli kartoittaa verkkosivuston saavutettavuustaso Web Content Accessibility Guidelines 2.1 (WCAG) -kriteeristön mukaisesti, korjata havaitut kriittisimmät ongelmakohdat ja luoda jatkuva saavutettavuustestauskäytäntö tulevaisuutta varten. Työ kohdistui Suomessa toimivan vähittäiskauppayrityksen brändisivustoon, joka ei kuitenkaan suoraan myy tuotteita kuluttajille. Työn rajauksena oli keskittyä apuvälineistä vain ruudunlukijan käyttöön, jotta tulokset palvelisivat mahdollisimman suurta käyttäjäryhmää.

Työn teoriaosuudessa käsitellään verkkosivustojen saavutettavuuden merkitystä ja etuja. Osiossa tarkastellaan voimassa olevaa ja tulevaa saavutettavuuslainsäädäntöä sekä saavutettavuuden mukanaan tuomia hyötyjä. Lisäksi esitellään WCAG-kriteeristö sekä Suomessa voimassa oleva digipalvelulaki, joka perustuu WCAG 2.1 -kriteereihin. Teoriaosuudessa käsitellään myös käytännön keinoja, joilla verkkosivustoista tehdään teknisesti saavutettavia.

Empiirisessä osiossa sivustolle tehtiin laaja auditointi käyttäen eri työvälineitä ja teknologioita. Sivuston saavutettavuutta testattiin mm. navigoimalla sivustolla manuaalisesti ruudunlukijalla ja käyttämällä eri apuvälineitä kuten skannauspalveluita. Löydetyt ongelmakohdat priorisoitiin niiden merkittävyyden mukaan, esimerkiksi sivuston toiminnallisuuksiin vaikuttavat ongelmakohdat olivat prioriteetiltaan korkeimmalla, samoin kuin selkeät, eri skannauspalveluiden löytämät virheet.

Työn tuloksena sivustolta korjattiin kriittisimmät ongelmakohdat, joihin luokiteltiin esteet toiminnallisuuksissa, kuten avautumattomat linkit, sekä skannauspalveluiden löytämät kriittiset ja vakavat virheet. Teknisten saavutettavuusongelmien juurisyynä vaikutti olevan enemmän tietämättömyys kuin osaamattomuus. Tulevaisuutta varten sivustolle luotiin automaattinen digitaalisen saavutettavuuden testauskäytäntö ja dokumentaatio kehitystiimille manuaalista saavutettavuustestausta varten. Testausprosessin avulla työssä tehty pohja saavutettavuudelle säilyy jatkossakin hyvällä tasolla. Tuloksissa esitellään myös jatkotutkimusaiheita, joiden avulla sivuston saavutettavuutta voidaan edelleen parantaa.

Asiasanat
Saavutettavuus, WCAG, verkkosivustot, testaus, inklusiivisuus

Sisällys

1 Johdanto ... 1
1.1 Työn tavoite ja rajaus ... 2
1.2 Terminologia ... 3
2 Saavutettavuuden teoreettinen viitekehys ... 4
2.1 Saavutettavuuden hyödyt ... 4
2.2 WCAG 2.1. ja digipalvelulaki ... 5
3 Miten digitaalinen palvelu tehdään saavutettavaksi? ... 8
3.1 Semanttinen HTML ... 8
3.2 Ohjelmallinen palaute ruudunlukijakäyttäjälle ... 9
4 Brändisivuston nykytilan analyysi ... 11
4.1 Nykytilan arvioinnin menetelmät ... 11
4.2 Navigointi ruudunlukijan avulla ... 13
4.3 Linkit ... 14
4.4 Kuvat ... 15
4.5 Otsikkotasot ... 16
4.6 Kontrasti ... 16
4.7 Zoomaus 200 % ... 17
4.8 Palaute käyttäjälle ... 17
4.9 Kriittisimmät ongelmakohdat ... 17
5 Toimenpidesuunnitelma ja korjaustoimenpiteet ... 19
5.1 Korjausstrategia ... 19
5.2 Käytännön toteutus ... 20
5.3 Kriittisimpien ongelmakohtien korjaustoimenpiteet ... 20
5.3.1 Ikonien puutteelisten vaihtoehtotekstien korjaus ... 21
5.3.2 Päävalikon navigoinnin korjaus ruudunlukijalle ... 21
5.3.3 Focustrap-toiminnallisuuden toteuttaminen sivuston modaaleihin ... 22
5.3.4 Palautteet ruudunlukijalle toiminnallisuuksista ... 23
5.3.5 Zoomaus 200 % ... 24
5.3.6 Käytettävyyteen liittyvät korjaukset ... 24
6 Testauskäytäntöjen kehittäminen ... 26
6.1 Manuaalisen testauksen menetelmät ... 26
6.2 Testien automatisointi ... 27
6.2.1 Saavutettavuustestien toteutus CI-putkeen ... 28
6.2.2 Testien toteutus ... 29
6.2.3 Saavutettavuustestien tulokset ... 31

7 Tulokset ja jatkotutkimusaiheet 32
8 Pohdinta 34
Lähteet 36

1 Johdanto

Yhä suurempi osa nykypäivän asioinnista tapahtuu digitaalisesti. Kommunikaatio, vapaa-ajanvietto, ostokset ja tiedonhaku hoidetaan yhä useammin verkossa ja erilaisten sovellusten kautta. Julkiset palvelut, pankit ja monet muut toimijat ohjaavat käyttäjiään ensisijaisesti digitaalisten palveluiden pariin. Median käytön digitalisoitumisen myötä perinteiset mediatkin keskittyvät kehittämään ensisijaisesti digitaalista käyttökokemusta printtilehtien sijasta. Samalla monilla toimijoilla vähennetään paikan päällä tapahtuvaa asiointia.

Digitaalisten palveluiden käyttö voi olla haastavaa monille ihmisryhmille, joilla on fyysisiä tai kognitiivisia rajoitteita. Esimerkiksi ikääntyneet henkilöt, joilla on heikentynyt näkö tai motorisia vaikeuksia, saattavat kokea verkkosivustojen käytön vaikeaksi, samoin kuin henkilöt, joilla on oppimisvaikeuksia tai synnynnäinen pysyvä vamma, kuten näkö- tai kuulovamma. (Etelä-Suomen Aluehallintovirasto.) Näiden käyttäjien riippuvuus apuvälineistä, kuten ruudunlukijoista tai pistenäytöistä, tekee heidän verkkokokemuksestaan erityisen herkän sille, miten sivustot on suunniteltu tukemaan näitä teknologioita. Jos verkkosivuston käyttöliittymä on epälooginen tai huonosti suunniteltu, se voi merkittävästi hankaloittaa heidän käyttökokemustaan, vaikka apuvälineet olisivat teknisesti yhteensopivia. Toisaalta digitalisaatio tarjoaa myös suuria mahdollisuuksia edistää inklusiivisuutta ja tukea rajoitteellisia ihmisiä erilaisten apuvälineiden avulla. Hyvin suunnitellut verkkopalvelut voivat lisätä itsenäisyyttä ja saavutettavuutta kaikille käyttäjäryhmille, erityisesti silloin, kun ne on optimoitu toimimaan saumattomasti apuvälineiden kanssa.

Eficoden yhteistyössä näkövammaisten liiton ja Annanpuran kanssa tehdyn tutkimuksen mukaan Suomessa on noin 55 000 näkövammaista, joista valtaosa on iäkkäitä ja heikkonäköisiä, näistä kaikista vain noin 22 % on täysin sokeita. Ruudunlukijoita käyttävät avustavana teknologiana myös henkilöt, joilla näkökyky on jossain määrin tallella, mutta kokevat hyödylliseksi ruudunlukijan. (Kallionpää & Kiiskilä 2021, 7.) Lisäksi on meille kaikille tilapäisiä haasteita, kuten kirkas sää, joka haittaa näytön lukua tai meluisa ympäristö. Palveluiden käyttö mobiililaitteilla on yleistynyt, mikä tarkoittaa, että ympäristön muutokset vaikuttavat käyttökokemukseen aiempaa voimakkaammin, koska laitteet kulkevat helposti mukana kaikkialle. Näiden ryhmien osallisuuden varmistamiseksi on tärkeää, että digitaaliset palvelut suunnitellaan saavutettaviksi ja helppokäyttöisiksi.

Verkko- ja mobiilisivustot tulisi kehittää jo lainsäädännön puolesta niin, että sivustot huomioivat ihmisten erilaisuuden. Suomessa digipalvelulaki velvoittaa viranomaisen asemassa toimivia organisaatioita, julkisoikeudellisia laitoksia ja osaa järjestöistä kehittämään verkkopalveluistaan saavutettavia (Etelä-Suomen Aluehallintovirasto). On kuitenkin tärkeää, että myös muut organisaatiot ja yritykset ottavat huomioon saavutettavuuden kehittäessään sivustojaan. Jos emme huomioi

saavutettavuutta jo kehitysvaiheessa, suljemme tietoisesti pois ihmisryhmiä, jotka ovat muutenkin rajoitteidensa vuoksi heikommassa asemassa.

Tämä tutkimus keskittyy työnantajani brändisivustoon, jota kutsun tässä tutkimuksessa kohdeyritykseksi. Kohdeyritys on toiminut Suomessa vähittäiskaupan alalla jo vuosikymmeniä, ja sen eri yksiköt ovat pitkään kehittäneet digitaalisia palveluja kuluttajille. Saavutettavuuden eteen on eri instansseissa tehty merkittävästi työtä, ja aihe on ollut esillä jo aikaisemmin. Kuitenkin, kun sivustot kehittyvät, saavutettavuudesta on pidettävä jatkuvasti huolta. Tämä työ liittyy yrityksen ICT-toimintoihin, ja tarkemmin ottaen yhteen sen brändisivustoista, jonka kehittäminen on pääasiallinen työtehtäväni kohdeyrityksessä. Työni alulle panijoita olivat tuleva EU-direktiivi, josta kerron myöhemmin lisää, vastuullisuus sekä sivustolle suunniteilla olevat tulevat muutokset.

1.1 Työn tavoite ja rajaus

Tavoitteenani on kartoittaa brändisivuston nykyinen saavutettavuustaso nojaten Web Content Accessibility Guidelines (WCAG) version 2.1. kriteereihin vaatimustasolla A ja AA, korjata kriittisimmät ongelmakohdat, luoda jatkuva testauskäytäntö tulevaisuuden varalle ja automatisoida nämä testit. Työn tuloksena odotetaan syntyvän kriittisimpien ongelmakohtien korjatut toteutukset sekä jatkuva testauskäytäntö, johon sisältyvät automatisoitu saavutettavuustestaus ja saavutettavuustestauksen dokumentaatio kehittäjäkollegoille. Molemmat osat ovat keskeisessä roolissa sivuston pysyvän saavutettavuuden varmistamisessa. Pelkkä nykyisten ongelmien korjaaminen ei riitä, sillä ilman jatkuvaa testausta saavutettavuus heikkenee vähitellen, kun sivustoa päivitetään ja kehitetään – etenkin nyt, kun sivusto on uudistumisvaiheessa.

Tässä tutkimuksessa selvitetään kohdesivuston saavutettavuutta apuvälinelaitteistoista vain ruudunlukijalla aiheen rajaamisen vuoksi. Verkkosivustojen saavutettavuus on laaja kokonaisuus, joka kattaa monenlaisia rajoitteita ja erilaisia saavutettavuusratkaisuja. Aikataulullisista syistä rajasin tutkimuksen koskemaan vain ruudunlukijan käyttöä, koska suurin osa apuvälineistä hyödyntää näppäimistönavigaatiota samalla tavalla kuin ruudunlukija, eli korjaukset, jotka toimivat ruudunlukijalla, toimivat usein myös muiden apuvälineiden kanssa. Lisäksi ruudunlukija on helposti saatavilla eikä sen käyttö vaadi erillisiä laitehankintoja tai lisenssejä. Ruudunlukijoista on käytössä ainoastaan Applen VoiceOver, koska sivuston käyttäjäanalytiikan perusteella suurin osa sivuston käyttäjistä vierailee sivulla mobiililaitteella ja VoiceOver on selkeästi suosituin mobiilissa käytettävistä ruudunlukijoista. WebAIMin vuonna 2024 julkaiseman tutkimuksen mukaan 1539 vastaajasta 70,6 % käytti Applen laitetta mobiiliselailuun, ja sama 70,6 % käytti VoiceOveria mobiilissa. Safari oli suosituin selain mobiilissa ruudunlukijalla selailuun, 58,2 % vastaajien käyttäessä sitä. (WebAIM 2024.) Näillä valinnoilla saavutettavuusparannukset todennäköisesti palvelevat tässä vaiheessa suurinta osaa ruudunlukijan käyttäjistä.

1.2 Terminologia

Attribuutti: Ominaisuus tai tieto, joka liitetään elementtiin tai objektiin ohjelmoinnissa, kuten HTML-elementtien määrittelyt.

Digitalisaatio: Prosessi, jossa tiedot ja palvelut siirretään sähköisiin muotoihin ja toimintoja automatisoidaan teknologian avulla.

Elementti: HTML-kielen perusyksikkö, joka määrittää verkkosivun rakenteen.

Karuselli: Verkkosivujen komponentti, jossa sisältöä vaihdetaan automaattisesti tai käyttäjän klikkausten avulla, usein käytetty kuvagallerioissa.

Komponentti: Itsenäinen osa käyttöliittymää tai koodia, jota voidaan uudelleenkäyttää sovelluksen tai verkkosivuston eri osissa (esim. painike, lomakekenttä).

Modaali: Ponnahdusikkuna tai -laatikko, joka avautuu pääsisällön päälle ja estää taustan käytön ennen sulkemista.

Ruudunlukija: Apuvälineohjelma, joka muuntaa verkkosivujen tekstin puheeksi tai pistekirjoitukseksi, auttaen erityisesti näkörajoitteisia käyttäjiä.

Saavutettavuus: Verkkosivustojen ja sovellusten suunnittelu siten, että ne ovat käytettävissä kaikille käyttäjille, myös niille, joilla on fyysisiä tai kognitiivisia rajoitteita.

Semanttinen HTML: HTML-koodauskäytäntö, jossa käytetään merkityksellisiä elementtejä (esim. article, header), jotka kuvaavat sivun rakenteen tarkoitusta.

Skripti / komentosarja: Ohjelmointikoodi, joka suorittaa tietyn toiminnon verkkosivulla tai sovelluksessa (esim. lomaketietojen käsittely tai animaatio).

Tyylikirjasto / tyyliluokka: Ohjelmallinen tyylisääntö, jota voidaan käyttää verkkosivuston useissa elementeissä yhtenäisen ulkoasun saavuttamiseksi.

Versionhallinta: Työkalu, jolla seurataan koodimuutoksia ja hallitaan versioita, esimerkiksi Git. Mahdollistaa yhteistyön ja historian hallinnan.

WCAG: (Web Content Accessibility Guidelines, eli verkkosisällön saavutettavuusohjeet) tarjoaa suosituksia verkkosivujen saavutettavuudelle sekä testattavia kriteereitä saavutettavuuden arvioimiseksi.

2 Saavutettavuuden teoreettinen viitekehys

Kuntaliitto määrittelee julkaisussaan Kuntien saavutettavuusopas, 2017 saavutettavuuden seuraavanlaisesti:

> ”Saavutettavuus ja esteettömyys merkitsee ympäristön, kohteen, tuotteiden, viestinnän tai palvelun helppoa lähestyttävyyttä kaikille, myös liikkumis- ja toimimisesteisille henkilöille.” (Tamminen, Alinikula, Hagerlund & Lindroth 2017)

Saavutettavuudella viitataan nykyään usein digitaalisiin palveluihin ja viestintään verkossa, kun taas esteettömyys yhdistetään useammin fyysisiin tilanteisiin, kuten porrasrampit pyörätuoleille.

2.1 Saavutettavuuden hyödyt

Kun sivusto rakennetaan saavutettavaksi, se palvelee suurempaa käyttäjäkuntaa. Osalle käyttäjistä saavutettava sivusto on välttämättömyys, mutta saavutettavuus palvelee jokaista käyttäjää. (Etelä-Suomen Aluehallintovirasto.) Monet saavutettavuuden aspektit, kuten selkeä sivustolla liikkuminen, eli navigointi, intuitiivinen käyttöliittymä ja ymmärrettävät ohjeet, parantavat käyttökokemusta kaikille käyttäjille, ei vain niille, joilla on toimintarajoitteita. Sivustojen omistajilla ei yleensä ole tarkoituksena suunnata sivustoja vain ihmisille, joilla ei ole minkäänlaisia rajoitteita. Valitettavasti inklusiivisuus voi kuitenkin helposti unohtua tai siitä halutaan säästää, erityisesti silloin, kun omistaja itse kuuluu siihen ryhmään, joille ei ole hankaluuksia käyttää palvelua muutoinkaan. Kohdeyrityksen tapauksessa sen palveluiden digitaalinen saavutettavuus nähdään yhtenä osana vastuullisuutta, jota halutaan kehittää.

Saavutettavuuden huomioiminen auttaa myös välttämään juridisia toimia. EU:n saavutettavuusdirektiivi koskee tällä hetkellä vain tuotteita tai palveluita, jotka on todettu tärkeimmäksi rajoitteellisille henkilöille (EU Directive 2019/882). Verkossa toimivien palveluiden osalta näihin kuuluu esimerkiksi pankkipalvelut, nettikaupat ja matkustuspalveluita tarjoavat sivustot (European Commission). Suomessa digipalvelulain 1 luvun 3 § velvoittaa tämän lisäksi tiettyjen tahojen tarjoamia verkkopalveluita olemaan saavutettavia. Tähän kuuluvat viranomaispalvelut, julkisoikeudelliset laitokset, osa järjestöistä, jotka saavat avustusta esimerkiksi STEA:lta (Sosiaali- ja terveysjärjestöjen avustuskeskus), jolloin ne katsotaan julkisoikeudellisiksi laitoksiksi. Lisäksi myös osa yksityisestä sektorista kuten finanssialan toimijat, sekä toimijat, joiden taustalla vaikuttaa viranomainen merkittävästi, esimerkiksi Posti kuuluvat digipalvelulain alle. (Laki digitaalisten palveluiden tarjoamisesta, 306/2019.)

Kohdesivustoa ei tällä hetkellä koske lainsäädännölliset vaatimukset saavutettavuudesta sen ollessa yksityisen sektorin kuluttajille suunnattu brändisivusto, jossa ei suoraan myydä kuluttajille. Vuonna 2025 voimaan astuva uusi EU-direktiivi, European Accessibility Act, koskee myös Suomea

ja vaikuttaa kaikkiin Euroopassa toimiviin vähittäiskaupan yrityksiin, joissa on yli 10 työntekijää (European Commission). Lain tavoitteena on yhdenmukaistaa EU-maiden saavutettavuuslainsäädäntöä ja parantaa digitaalista käyttökokemusta jopa 87 miljoonalle eurooppalaiselle, joilla on jokin vamma tai rajoite. Tuleva laki ei aseta tarkkoja teknisiä vaatimuksia saavutettavuuden toteuttamiseksi, mutta se määrittelee tuote- ja palveluominaisuudet, jotka on oltava saavutettavia lain voimaanastumiseen mennessä. Tämä tuleva direktiivi on toiminut myös kohdeyrityksessä sisäisenä ajurina palveluiden saavutettavuuden kehittämisessä.

Lainsäädännön lisäksi saavutettavuuden parantaminen lisää asiakastyytyväisyyttä ja vahvistaa kuluttajien mielikuvaa brändistä. Se myös parantaa hakukonenäkyvyyttä, kun esimerkiksi sisältöhierarkia on selkeä ja kuville on asetettu tekstivastineet. Lisäksi saavutettavuus tuo pitkällä aikavälillä kustannussäästöjä, sillä hyvin suunniteltu ja saavutettava palvelu vaatii vähemmän tukea ja huoltotoimenpiteitä. (Lindholm, 21.4.2020.)

2.2 WCAG 2.1. ja digipalvelulaki

Web Content Accessibility Guidelines (WCAG / Verkkosisällön saavutettavuusohjeet) määrittelee suositukset, joiden avulla verkkosivuista voidaan tehdä paremmin saavutettavia henkilöille, joilla on erilaisia vammoja tai rajoitteita (W3C). Kutsun jatkossa ohjeistusta lyhenteellä WCAG.

WCAG:sta on julkaistu useita versioita, ja tässä keskitymme versioon 2.1, joka julkaistiin vuonna 2018. Tämä versio täydentää aiempaa versiota 2.0, joka julkaistiin vuonna 2008, ja sen tarkoituksena on tuoda lisäohjeistusta uusien saavutettavuustarpeiden huomioimiseksi. WCAG version 2.1 noudattaminen takaa myös, että verkkosivu täyttää WCAG 2.0 sekä aiempien versioiden vaatimukset.

WCAG 2.0 tarjoaa ohjeistusta eri tasoilla, joita ovat periaatteet, ohjeet, onnistumiskriteerit sekä riittävät ja neuvoa antavat tekniikat. Periaatteita on neljä, ja ne muodostavat verkkosaavutettavuuden perustan: havaittavuus, hallittavuus, ymmärrettävyys ja toimintavarmuus. Näiden alapuolella ovat ohjeet, jotka antavat yleisiä tavoitteita sisällöntuottajille saavutettavuuden huomioimiseksi. Verkkosivujen saavutettavuutta arvioidaan onnistumiskriteerein, joita on kolme eri tasoa: A, AA ja AAA. Nämä sisältävät konkreettisia, testattavia ehtoja, joiden avulla pystytään arvioimaan sivuston saavutettavuustasoa. Taulukossa 1 on kuvattu WCAG 2.1 eri tasoja selityksineen ja muutamin esimerkkitapauksin, millaisia onnistumiskriteerejä tason alle kuuluu. On kuitenkin tärkeää muistaa, että vaikka kaikki onnistumiskriteerit täyttyisivät, sivusto ei välttämättä ole aidosti saavutettava kaikille käyttäjille. Saavutettavuus on laaja kokonaisuus, ja käyttäjillä voi olla hyvin monenlaisia rajoitteita sekä tapoja käyttää sivustoa.

Taulukko 1. WCAG 2.1 -tasot A, AA ja AAA: selitykset ja esimerkkionnistumiskriteerit

Taso	Selitys	Esimerkit
A	Perustason saavutettavuus, joka varmistaa, että kriittiset esteet on poistettu.	– Tekstivastineet kuville – Täydellinen näppäimistösaavutettavuus – Tekstitykset ennakkoon nauhoitetulle äänisisällölle
AA	Taso AA laajentaa tasoa A käsittelemällä laajempaa valikoimaa saavutettavuusongelmia ja tekee sisällöstä käyttökelpoisempaa eri rajoitteista kärsiville henkilöille.	– Värikontrasti tekstin ja taustan välillä (vähintään 4,5:1) – Zoomaus 200 % – Selkeät nimeämiset ja ohjeistukset, kun käyttäjältä vaaditaan syötettä (Esim. lomakkeet) – Sisältö saavutettavaa ja ymmärrettävää
AAA	Taso AAA sisältää kaikki tason A ja AA vaatimukset sekä lisää vaativampia standardeja, mutta niitä ei yleensä vaadita useimmilta verkkosivustoilta. Tyypillisesti käytetään verkkosivustoilla, jotka palvelevat hyvin monimuotoista yleisöä.	– Viittomakielen tulkkaus videomediasisällöille – Tiukempi värikontrasti 7:1 – Ei aikarajoja käyttäjän toiminnallisuuksiin (Esim. sisäänkirjautuminen)

Digipalvelulaki velvoittaa lain alle sovellettavia sivustoja täyttämään WCAG 2.1. A- ja AA-tason ohjeistuksen onnistumiskriteerit, joitain poikkeuksia lukuun ottamatta, kuten videomuotoisten livelähetysten tekstitykset. (Laki Digitaalisten palveluiden tarjoamisesta 15.3.2019/306; Aluehallintovirasto). Jokaisella digipalvelulain alla sovellettavalla sivustolla on myös oltava saavutettavuusseloste, jossa kuvaillaan sivuston saavutettavuutta ja täyttääkö sivusto digipalvelulain vaatimukset. Vaatimuksista voi digipalvelulain 3.luvun 8 § mukaan poiketa, jos palveluntarjoaja ennakkoon tehdyn saavutettavuusarvionnin perusteella voi osoittaa, että vaatimusten toteuttaminen aiheuttaa toiminnalle kohtuuttoman rasitteen. Esimerkiksi vuonna 2021 Etelä-Suomen Aluehallintovirasto antoi Verohallinnolle huomautuksen puutteellisesta saavutettavuudesta Palkka.fi -sivustolla (Aluehallintovirasto 9.9.2021). Sivuston käyttäjä oli antanut aluehallintovirastolle kantelun sivuston saavutettavuusongelmista. Sanktiot tapauksesta olivat kuitenkin suhteellisen pienet, eli huomautus ja vaatimus tehdä suunnitelma saavutettavuuden takaamiseksi ja ryhtyä tarvittaviin toimiin. Verohallinto vetosi kohtuuttomaan rasitteeseen, koska sivusto oltiin uudistamassa lähiaikoina ja saavutettavuusongelmat olisi korjattu samalla. Tämä ei ainakaan suoraan täyttänyt kohtuuttoman rasitteen kriteeristöä. Kohdeyrityksen osalta voidaan olettaa, että poikkeukset, jotka perustuvat

kohtuuttomaan rasitteeseen, eivät ole mahdollisia, sillä yritys on Suomen mittakaavassa merkittävä vähittäiskaupan toimija ja sillä on käytössään riittävät resurssit.

Huomautuksen lisäksi digipalvelulakia valvovat viranomaiset voivat määrätä uhkasakon, tai ottaa digitaalisen palvelun valvonnan alle.

3 Miten digitaalinen palvelu tehdään saavutettavaksi?

Kun sivustosta halutaan kaikille saavutettava, jo suunnitteluvaiheessa on syytä pohtia tarkkaan, kenelle palvelu on tarkoitettu. Käyttäjäkunta voi olla hyvin monimuotoinen, ja on otettava huomioon esimerkiksi käyttäjien ikä sekä mahdolliset toimintarajoitteet ja erityistarpeet, kuten apuvälineiden käyttö. Samalla on tärkeää miettiä, millaisilla laitteilla palvelua käytetään. Käyttäjät voivat käyttää palvelua niin tietokoneilla, mobiililaitteilla kuin tableteillakin, ja lisäksi jotkut käyttäjät saattavat hyödyntää ruudunlukuohjelmia tai muita apuvälineitä. Siksi palvelun on oltava toimiva ja saavutettava eri laitteilla ja selaimilla. Myös se, mitä sivustolla tehdään, vaikuttaa suunnitteluun. Käyttäjien tehtävät voivat vaihdella yksinkertaisesta tiedonhausta monimutkaisempiin toimintoihin, kuten lomakkeiden täyttämiseen tai vuorovaikutukseen muiden käyttäjien kanssa. Palvelun toiminnallisuuksien tulee siis olla selkeitä, helposti saavutettavia ja loogisia, jotta ne palvelevat kaikkia käyttäjiä tasavertaisesti. Kun sivuston saavutettavuus on suunniteltu, saavutettava sivusto tehdään tietyillä tekniikoilla, joita luettelen seuraavissa kappaleissa.

3.1 Semanttinen HTML

HTML-kieli (lyhenne sanoista Hypertext Markup Language), on merkintäkieli, jolla standardoidusta syntaksista, eli kieliopista muodostetaan digialustoille sisältöä. HTML:n eri elementit muodostavat rakenteen sivustolle, jota käyttäjät pystyvät tutkimaan, joko ruudun näytöltä tai esimerkiksi ruudunlukijalla. Semanttinen HTML tarkoittaa sitä, että HTML:n elementtejä käytetään siihen tarkoitukseen, mihin ne on tarkoitettu (MDN). Sivusto, joka on kehitetty oikeaoppisesti semanttisen HTML:n mukaan, on jo suurelta osin saavutettava.

Semanttiset elementit, kuten header, nav, main, article, ja footer, kuvaavat sivun rakenteen loogisesti. Ruudunlukijat ja muut apuvälineet voivat tunnistaa nämä elementit ja tarjota käyttäjille selkeän käsityksen sivun sisällön hierarkiasta ja navigoinnista. Esimerkiksi ruudunlukija voi hyppiä suoraan navigointiosioon (nav) tai pääsisältöön (main), mikä parantaa käyttökokemusta. Kun HTML-elementit on nimetty tarkoituksenmukaisesti, käyttäjät voivat helpommin ymmärtää sivun rakenteen. Esimerkiksi otsikkotasot h1–h6 auttavat jäsentämään sisällön hierarkkisesti. Näkevä käyttäjä pystyy yleensä hahmottamaan sivun rakenteen nopealla silmäyksellä erikokoisten otsikkotasojen myötä, jos pääotsikon alle on asetettu fonttikooltaan pienempiä alaotsikoita, ja niiden alle itse asiateksti. Tämä sama tulisi toteuttaa ohjelmallisesti käyttämällä oikeita otsikkotasoja, jota ruudunlukija osaa tulkita. Tämän voi tuottaa oikeaoppisesti käyttämällä semanttisia merkintöjä ja rakennetta otsikkotasossa, tai väärin välittämättä otsikkotasoista ja esimerkkinä tyylittelemällä otsikot vain fonttikooltaan isommaksi. Ruudunlukija ei huomioi suurempaa kirjaisinkokoa tai muita tyylimuotoiluja, eli tässä tapauksessa ruudunlukijan käyttäjille sivusto olisi sekava. Käyttäjä, jolla näkökykyä ei ole tai se on merkittävästi huonontunut ja käyttää ruudunlukijaa sivun ymmärtämiseen,

oikealla tavalla semanttisesti merkitty sisältö on rakenteen kannalta erittäin tärkeä. Eficoden tekemän kyselyn perusteella jopa 47 % käyttäjistä tutustuu sivustoon käymällä sen otsikot ensin läpi Riihiaho 30.6.2022).

Ongelmia sivustoilla tuottaa usein se, että nykypäivänä verkkosivustoja tehdään erilaisilla kielikirjastoilla, jotka tarjoavat valmiita elementtejä kuten ohjelmistokieli Javascriptiin perustuva kirjasto React, joka käyttää JSX-syntaksia. JSX-syntaksi muistuttaa läheisesti HTML-kieltä, mutta sen avulla kirjoitetaan Javascript-koodia. Nämä kirjastot kääntävät sivuston elementit HTML-kielelle vasta selaimessa, eikä kehittäjä välttämättä muista sivuston semanttisen HTML-rakenteen tärkeyttä. Otsikkotasoja saatetaan käyttää väärin, sivuston linkit voivat olla syntaksin vastaisesti tehty, kuvista puuttuu tekstivastine (alt-teksti), tai käytetään väärissä kohdin geneerisiä elementtejä, joita ruudunlukijat eivät lue.

3.2 Ohjelmallinen palaute ruudunlukijakäyttäjälle

Ensisijaisesti on tärkeää merkitä sivuston kirjoitettu kieli ruudunlukijoita varten. Jos kieltä ei ole merkattu ja sivuston sisältö olisi esimerkiksi suomeksi, ruudunlukija lukee suomenkielisen tekstin englannin kielen korostuksella, jonka vuoksi lopputuloksesta voi tulla mahdotonta ymmärtää.

Semanttinen HTML ei itsessään anna jokaisessa tapauksessa riittävästi selkeää palautetta ruudunlukijan käyttäjälle. Esimerkiksi painikkeiden ja lomakkeiden toiminnallisuus ei välity pelkän HTML:n kautta, jos kehittäjä ei ole lisännyt elementteihin kuvaavia attribuutteja eli ARIA-attribuutteja. Käyttäjä voi esimerkiksi navigoida painikkeeseen, jonka pitäisi lisätä tuote ostoskoriin. Jos elementti on kirjoitettu vain

```
<button onClick=”add(product.id)”> Lisää ostoskoriin </button>
```

ja sitä kuvaamassa on kuva tuotteesta, jolle ei ole asetettu tekstivastinetta, ruudunlukija osaa lukea tekstin ”Lisää ostoskoriin”, mutta sivusto ei kerro ruudunlukijalle mikä tuote on kyseessä. Tämä tekee tuotteiden valitsemisen ilman näkökykyä haastavaa ja jopa mahdottoman.

Ruudunlukijalle voi antaa kuvaavia ominaisuuksia, jotka eivät näy visuaalisesti käyttäjille. Näitä ominaisuuksia eli attribuutteja on hyvä antaa varsinkin toiminnallisuuksissa, kuten lomakkeissa, latauskohdissa, käyttäjän tehdessä valintoja sivulla ja muissa vastaavissa tilanteissa, jossa sivustolla tapahtuu jotain ohjelmallista. Näitä tietoja kutsutaan ARIA-attribuuteiksi ja -rooleiksi (Accessible Rich Internet Applications). Semanttinen HTML sisältää jo itsessään roolituksen ja erilaisia kuvailuja elementeille, eikä ARIA-attribuutteja pitäisi antaa elementeille, jos HTML jo itsessään sisältää tiedot. Väärin käytetty ARIA-elementti tekee sivustosta vaikeammin hahmotettavan, kuin ARIA-attribuuttien käyttämättä jättäminen. WebAimin vuonna 2024 tekemän tutkimuksen mukaan, jossa

analysoitiin miljoonan verkkosivuston saavutettavuutta, todettiin, että sivustoilla, joissa käytetään ARIA-elementtejä, esiintyi keskimäärin 41 % enemmän saavutettavuusvirheitä verrattuna sivustoihin, joissa ARIA-elementtejä ei käytetty lainkaan (WebAIM 2024).

ARIA-attribuutteja annetaan elementeille, joilla ei ole semanttisessa HTML:ssä tarkoitusta, kuten div-elementeille. Div-elementti on HTML-kielen yleinen jakoelementti, joka itsessään ei vaikuta sisältöön. Sitä käytetään kuitenkin esimerkiksi ryhmittelemään muita elementtejä, jolloin se auttaa luomaan visuaalisesti selkeämpiä ja strukturoituja käyttöliittymiä käyttäjille. (MDN.) Div-elementillä voi rakentaa esimerkiksi painikkeen, jonka sisällä on kuva ja tekstiä, ja joka painattaessa lisää tuotteen ostoskoriin. Silloin div-elementille annetaan ARIA-rooli ”button”, sekä aria-label ”lisää *tuotteen nimi* ostoskoriin”, joka antaa selkeän tekstivastineen ruudunlukijoille painikkeen tarkoituksesta. Saman painikkeen olisi kuitenkin voinut rakentaa käyttämällä HTML-elementtiä button, joka sisältää sisäänrakennettuna jo roolin. Myös painikkeet, jotka näkyvät käyttäjälle esimerkiksi ikonina, merkitään aria-attribuutilla ja annetaan toimintoa kuvaava tekstivastine, jotta ruudunlukija osaa lukea käyttäjälle mitä painike tekee.

4 Brändisivuston nykytilan analyysi

Tässä luvussa tarkastellaan sivuston tämänhetkistä tilaa sekä teknisestä näkökulmasta. Nykytilan arvioinnissa käytetyt menetelmät kattavat tekniset analyysit, joihin kuuluu esimerkiksi sivuston käytettävyyden ja saavutettavuuden tarkastelu. Vaikka kognitiivinen näkökulma, kuten käyttäjien kokemukseen liittyvät tekijät, on tässä työssä rajattu pois tarkemmasta analyysistä, sen osalta voidaan todeta, että sivuston ymmärrettävyys ja loogisuus vaikuttavat olevan kohtuullisella tasolla. Luvussa keskitytään kuitenkin pääosin teknisten tulosten analysointiin ja niiden vaikutukseen sivuston toimivuuden kannalta.

4.1 Nykytilan arvioinnin menetelmät

Taulukossa 2 on esitetty työssä käytetyt teknisen saavutettavuustestauksen työkalut:

Taulukko 2. Teknisen saavutettavuustestauksen työkalut

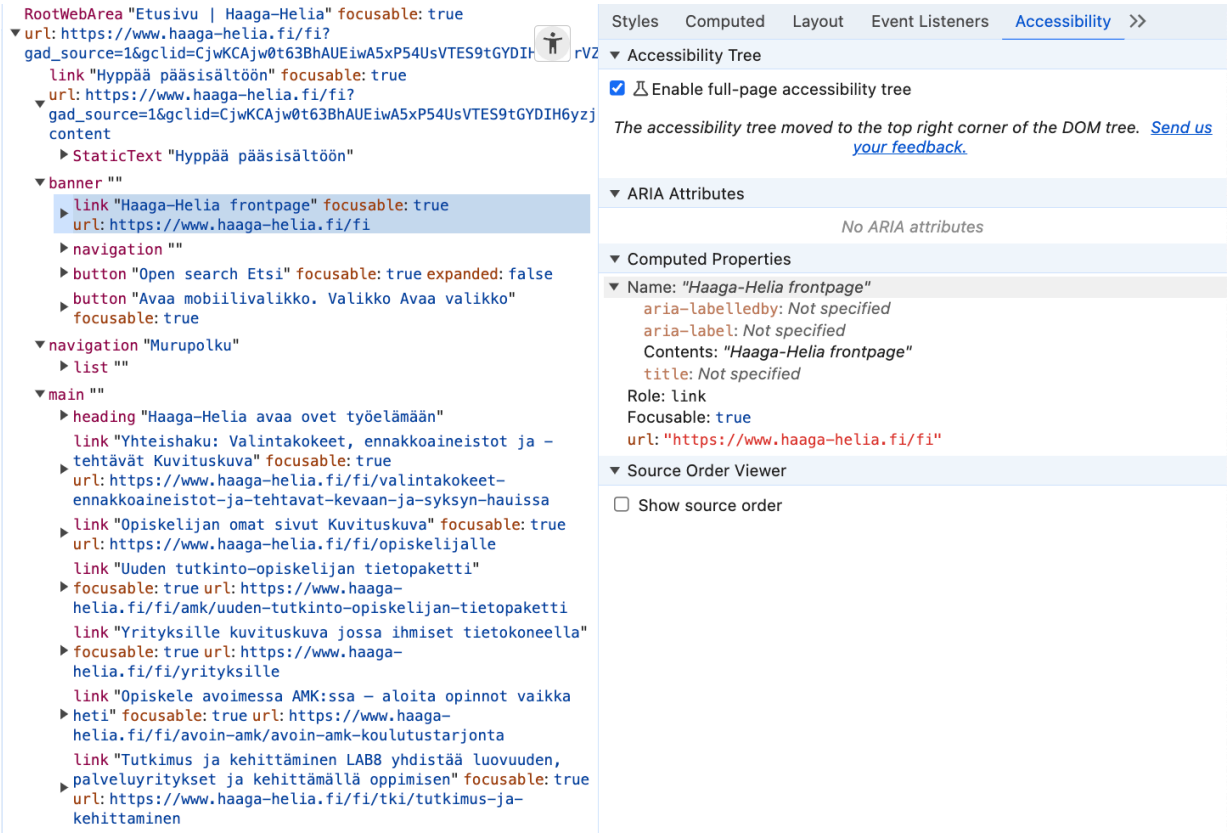
Laite	Selaimet	Apuvälineet
MacBook Pro -tietokone	– Edge – Chrome – Firefox – Safari	– Chrome Devtools – Wave Web Accessibility tool – Color Contrast Check – VoiceOver-ruudunlukuohjelma – Pelkän näppäimistön käyttö – Zoomaus 200 %
iOS -puhelin	– Safari – Chrome – Firefox	– VoiceOver -ruudunlukuohjelma – Zoomaus 200 %

Ensimmäiseksi analysoin sivuston nykytilaa käyttämällä Wave Web Accessibility Tool -työkalua. Työkaluun syötetään sivun URL-osoite, jonka perusteella se arvioi sivun saavutettavuuden ja merkitsee löydetyt tulokset erivärisillä ikoneilla. Jos sivulta löytyy punaisella merkattuja ikoneja, se tarkoittaa, että sivulla on saavutettavuuteen liittyviä virheitä. On kuitenkin tärkeää huomata, että virheiden puuttuminen ei automaattisesti tarkoita, että sivusto olisi täysin saavutettava. Wave-työkalu tunnistaa lähinnä suurimmat ongelmat, joten sivuston saavutettavuuden kokonaisvaltainen arviointi vaatii syvempää tarkastelua.

Color Contrast Check -työkalulla voidaan testata tekstin ja taustan välinen kontrasti syöttämällä niiden värikoodit. Tätä työkalua on käytetty tässä työssä silloin, kun sivustolla on havaittu silmäilemällä elementtejä, joissa kontrastiongelma saattaa ilmetä. Esimerkkejä tästä ovat värillinen teksti värillisellä taustalla, mikä voi heikentää tekstin luettavuutta. Liian matala kontrasti tekstin ja taustan

välillä on ongelmallinen saavutettavuuden kannalta, mutta väritysten ollessa design -päätöksiä, löydöksiä ei ole keretty korjata tämän työn aikarajojen puitteissa.

Google Chrome -selaimen DevTools-työkalut tarjoavat mahdollisuuden ottaa käyttöön "Full Accessibility Tree" -ominaisuuden, jonka avulla voi tarkastella, mitkä sivuston elementit ovat saavutettavissa ja miten ne luetaan ruudunlukijalla. Tämä työkalu on erityisen hyödyllinen, kun halutaan tutkia yksittäisten elementtien saavutettavuutta ja niiden toimivuutta käyttäjän apuvälineiden näkökulmasta. Kuvassa 1 on näyttökuva Chromen DevTools-työkalujen "Full Accessibility Tree" -ominaisuudesta. Kuvassa vasemmalla näkyy Full Accessibility Tree, kun taas oikealla on DevToolsin Accessibility-välilehti, josta voidaan tarkastella elementtien saavutettavuusominaisuuksia, kuten roolia ja nimeä.



Kuva 1. Näyttökuva Google Chrome-selaimen Full Accessibility Tree ominaisuudesta, jolla pystyy tarkastelemaan sivuston elementtien näkyvyyttä eri apuväleillä, kuten ruudunlukijalla.

Tässä työssä teknisen saavutettavuuden testaaminen on rajoitettua, koska käytössä on vain Applen MacBook ja iPhone, joissa molemmissa on Applen oma ruudunlukija, VoiceOver. Eri ruudunlukijat voivat toimia eri tavoin, joten ongelmia, joita VoiceOver ei havaitse, saattaa ilmetä toisilla ruudunlukijoilla. Nämä ovat kuitenkin yleensä yksittäisiä ongelmia, eivätkä ne merkittävästi vaikuta

kokonaiskuvaan, sillä sivuston saavutettavuus on pääosin samanlainen ruudunlukijasta riippumatta. Ruudunlukijan käytöstä kerron enemmän seuraavassa kappaleessa.

Taulukossa 3 on lueteltu kognitiivisen saavutettavuustestauksen menetelmiä. Vaikka tässä työssä kognitiivinen saavutettavuus rajataan tarkastelun ulkopuolelle, on tämä näkökulma saavutettavuuden kannalta merkittävä. Kun sivusto on helppokäyttöinen, visuaalisesti selkeä, sisältää selkeää kieltä sekä on jäsennelty asianmukaisesti otsikoiden ja alaotsikoiden avulla, voivat myös henkilöt, joilla on oppimis-, hahmottamis-, muisti- tai vaikka kielitaidon haasteita, käyttää sivustoa sujuvammin. (Kehitysvammatuki & Kehitysvammaliitto.)

Taulukko 3. Kognitiivisen saavutettavuustestauksen menelmät

Menetelmä	**Tarkastelun kohde**
Ulkoasun ymmärrettävyys	Tarkastellaan, kuinka selkeä ja käyttäjäystävällinen sivuston visuaalinen ilme on, esimerkiksi visuaalisten elementtien sijoittelu ja värien kontrastit.
Informaation määrä	Tarkastellaan sisällön määrää ja sen jäsentelyä. Arvioidaan, onko sisältö esimerkiksi otsikoitu ja jaoteltu selkeästi, eikä tietoa ole liikaa yhdessä yhtenäisessä tekstilohkossa.
Sivustolla liikkuminen	Arvioidaan, onko sivustolla navigointi selkeää ja onko tiedot löydettävissä esimerkiksi ruudunlukijaa käytettäessä. Tarkastellaan myös sivukarttojen ja valikoiden käytettävyyttä.
Palaute käyttäjälle	Tarkastellaan, antavatko toiminnallisuudet selkeän palautteen käyttäjälle. Esimerkiksi ilmoitus onnistuneesta ostoskoriin lisäämisestä, ilmoitukset ja painikkeiden palaute.
Kieli ja terminologia	Arvioidaan, onko sivustolla käytetty kieli selkeää ja terminologia ymmärrettävää kaikille käyttäjäryhmille, mukaan lukien mahdolliset aloittelevat tai erityistarpeiset käyttäjät.

4.2 Navigointi ruudunlukijan avulla

Navigointi tapahtuu yleensä näppäimistön avulla. Esimerkiksi eteenpäin siirrytään tabulaattorilla (Tab-näppäin), ja MacOS-käyttöjärjestelmässä valintoja tehdään näppäinyhdistelmällä **Control + Option + välilyönti**. Kehittäjän on hyvä tiedostaa, että eri ruudunlukuohjelmat on kehitetty vasten tiettyjä selaimia. Applen VoiceOver toimii parhaiten Safari-selaimen kanssa, kun taas Windows-

käyttöjärjestelmällä käytetään yleensä yhdistelmää Google Chrome tai Microsoft Edge + NVDA tai JAWS. Mobiililaitteilla navigointi tapahtuu erilaisin liikkein, kuten pyyhkäisemällä ja napauttamalla. Eficoden tuottaman tutkimuksen mukaan Suomessa 71 % mobiililaitteella käyttävistä käyttää Applen iOS-käyttöjärjestelmää ja näistä käyttäjistä 78 % käyttää VoiceOveria ruudunlukijana (Kallionpää & Kiiskilä 2021, 3).

Kohdesivustolla, johon auditointi tehdään, on toiminnallisuuteen vaikuttavia ongelmia. Sivustolla on kirjautumisominaisuus, jonka avulla käyttäjät voivat tallentaa suosikkisisältöä omalle tililleen. Näppäimistön käytössä on havaittu ongelma, jossa kirjautumispainike ohitetaan kokonaan, ja fokus siirtyy suoraan muuhun sisältöön. Tämä johtuu painikkeen toteutuksesta div-elementtinä. Ruudunlukija ei huomioi tiettyjä geneerisiä elementtejä, jos niille ei ole asetettu roolia.

Yläpalkin valikko sisältää neljä eri osaa, joista klikkaamalla käyttäjä navigoituu toiselle sivulla tai hiirellä päällä hoveroimalla avautuu alavalikko, joka mahdollistaa lisäsisältöön navigoinnin. Tämä alavalikko ei kuitenkaan ole saavutettavissa näppäimistöllä, koska näppäimistö tunnistaa ainoastaan päävalikon linkkinä, mutta ei avaa alavalikkoa. Tämä toistuu myös mobiililaitteilla. Katso korjausstrategia luvusta 5.3.1. Myös haku-sivulla käyttäjä pystyy hakemaan sisältöä, mutta ei rajaamaan tuloksia, koska eri alasvetovalikkojen sisältöön ei pääse käsiksi näppäimistöllä.

Sivustolla on myös useampia pienempiä ongelmia, kuten eri sisältökarusellien ohittaminen Tab-näppäimellä. Sisältö itsessään on selkeää ja ruudunlukija tunnistaa otsikon ja muun sisällön, mutta käyttäjä joutuu näppäilemään koko karusellin sisällön alusta loppuun päästäkseen eteenpäin sivulla. Katso toteutettu korjaus luvusta 5.3.4.

Muuten navigointi on toimivaa. On kuitenkin muistettava, etten ole tottunut käyttämään ruudunlukuohjelmia, eli navigoinnin vaikeudessa saattaa olla myös käyttäjävirheitä. Yllä mainitut ongelmakohdat ovat kuitenkin todellisia myös oikeille käyttäjille.

4.3 Linkit

Auditoinnin perusteella sivustolta löytyi linkkien kanssa muutama selkeä ongelmakohta. Oikeaoppisesti käytettynä HTML:n a -elementti osaa kertoa ruudunlukijoille, mihin linkki johtaa. A-elementin sisään ei semanttisen HTML:n mukaan tulisi sijoittaa toista interaktiivista elementtiä, kuten button -elementtiä. Tämä voi sekoittaa erilaisia apuvälineitä tulkinnassa, koska niiden on haastava ymmärtää, mitä elementti tekee. Linkki on tarkoitettu navigoimaan käyttäjän toiseen osoitteeseen, kun taas painike suorittaa toiminnon sivun sisällä. Tämän vuoksi näiden elementtien tarkoitukset on pidettävä erillään, jotta käytettävyys ja saavutettavuus eivät kärsi.

Allaolevassa kuvassa (kuva 2) on sivustolta löydetty linkki, joka vei käyttäjän toiseen osoitteeseen ja samalla kirjasi analytiikkatiedon käyttäjän klikkauksesta. Tämä oli toteutettu virheellisesti siten, että painike (button) oli asetettu linkin (a) sisään. Tämä johti siihen, että ruudunlukija ohitti tällä tavalla rakennetut linkit kokonaan, ja siirtyi seuraavaan elementtiin sivulla.

```
1   return (
2       <a href={buttonProps.href} target={buttonProps.target || '_self'}>
3         <Button
4           className={`${buttonProps.className || ''}`}
5           inverse={buttonProps.inverse}
6           onClick={(event) => {
7             buttonProps.analyticsEvent && sendEvent(buttonProps.analyticsEvent);
8             buttonProps.onClick && buttonProps.onClick(event);
9           }}
10        >
11          {buttonProps.text}
12        </Button>
13      </a>
14    );
```

Kuva 2. Esimerkki virheellisestä toteutuksesta, jossa <a> elementin sisällä käytetään interaktiivista <button> elementtiä, mikä aiheuttaa saavutettavuus- ja käytettävyysongelmia.

4.4 Kuvat

Saavutettavuuden kannalta on tärkeää, että jokaisella verkkosivustolla käytetyllä kuvalla on määriteltynä tekstivastine, jonka apuvälineet, kuten ruudunlukijat, voivat lukea. Jos sivustolla on esimerkiksi kuva, jossa on kaksi naista, kuvaan tulisi lisätä HTML-koodiin alt-attribuutti, joka kuvailee kuvan sisällön. Esimerkiksi:

<img src="naiset.jpg" alt="Kaksi naista seisoo">

Tämä varmistaa, että kuvan sisältö välittyy kaikille käyttäjille, myös niille, jotka eivät pysty näkemään kuvaa.

Auditoinnin tuloksena havaittiin, että suurimmassa osassa sivuston kuvista on käytetty alt-attribuuttia, ja alt-tekstit eli tekstivastineet kuville ovat kuvailevia ja informatiivisia. Sivustolla käytetyissä ikoneissa on havaittu puutteellisia tai epäkuvaavia alt-tekstejä, mikä rikkoo WCAG 2.1 (A) -kriteerin 4.1.2 ("Name, Role, Value") vaatimuksia. Tämä muodostaa saavutettavuusongelman, sillä

apuvälineiden käyttäjät eivät saa riittävää tietoa ikonien tarkoituksesta. Kriteerin mukaan jokaisella käyttöliittymäkomponentilla, kuten ikoneilla, tulee olla selkeä nimi, rooli ja arvo, jotta niiden toiminta on ymmärrettävissä myös apuvälineiden kautta.

Jotkin työkalut, kuten Wave, antavat palautetta liian pitkästä tekstivastineesta. Osa ruudunlukijoista saattaa katkaista tekstivastineen, jos merkkimäärä ylittää tietyn rajan. Tästä on kuitenkin saatavilla ristiriitaista tietoa. Esimerkiksi Googlen valmiita komponentteja ja tyylejä tarjoava Material Design -tyylikirjasto suosittelee tekstivastineen rajaamista alle 140 merkkiin (Material Design), kun taas University of Maine System ehdottaa rajaa 125 merkille (University of Maine System). Osa lähteistä puolestaan katsoo, että tämä tieto on vanhentunutta (Graham 2022). Luotettavista lähteistä ei löytynyt selkeää konsensusta merkkirajoituksesta, mutta tämä tieto toistui useissa eri lähteissä. Tämän perusteella suosittelen, että kuvien tekstivastine pidetään yhdestä kahteen lauseeseen, jotta vältetään mahdolliset ongelmat ruudunlukijoiden käytössä.

4.5 Otsikkotasot

Auditoinnin perusteella sivuston otsikkotasot ovat pääosin hyvin jäsenneltyjä. Kuitenkin joiltakin sivuilta puuttuu pääotsikko (h1), mikä on saavutettavuuden kannalta merkittävä ongelma. Pääotsikon puuttuminen vaikeuttaa sivun sisällön ymmärtämistä ja navigointia erityisesti apuvälineiden käyttäjille, kuten ruudunlukijoille, jotka käyttävät otsikoita hahmottaakseen sivun rakenteen ja sisällön hierarkian.

4.6 Kontrasti

Brändisivuston kehityksessä hyödynnetään kohdeyrityksen omaa tyylikirjastoa, joka on suunniteltu täyttämään WCAG 2.1 A ja AA -tason onnistumiskriteerit. Tyylikirjasto ratkaisee noin 40 % komponenttien saavutettavuusongelmista, jos se on kokonaisuudessaan käytössä, mutta muutoin sivuston saavutettavuus on kehitystiimin vastuulla. Tyylikirjastossa on mukana kohdeyrityksen brändiin liittyviä komponentteja ja värejä, ja se sisältää eri palveluille määritellyt teemat. Käyttöön otettavat värit on valittu siten, että ne noudattavat WCAG 2.1 -kriteerien mukaisia kontrastivaatimuksia. Tekstin ja tekstinä esitettyjen kuvien visuaalisessa esityksessä kontrastisuhteen tulisi olla vähintään 4,5:1, lukuun ottamatta puhtaasti kuvitteellisia elementtejä ja logoja. Suurikokoisille kuville vaaditaan vähintään 3:1 kontrastisuhde. (W3C.) Tyylikirjasto on pääasiassa käytössä sivustolla, lukuun ottamatta muutamia poikkeuksia, jolloin sivuston värikontrastit täyttävät pääosin kriteerit.

Huomioitava ongelma on artikkelien lopussa olevassa tietolaatikossa, jossa vaaleanliilalla pohjalla olevat linkit ovat väriltään vaaleanvihreitä, mikä heikentää saavutettavuutta. Kontrasti tässä on 4.43:1. Heikot värikontrastit voivat vaikeuttaa käyttäjäkokemusta erityisesti värisokeille ja

heikentyneen näön omaaville. Vahvat kontrastit parantavat luettavuutta ja tekevät sisällöstä saavutettavampaa kaikille, erityisesti kirkkaassa valaistuksessa, kuten auringonpaisteessa.

4.7 Zoomaus 200 %

Sivuston zoomaus 200 % vastaa WCAG 2.1 AA -tason onnistumiskriteeriä 1.4.4. Kaikkien sivuston elementtien tulee olla käytettävissä ja näkyvissä, kun sivua zoomataan 200 %. Brändisivuston auditoinnin perusteella on kuitenkin havaittu muutamia ongelmakohtia. Esimerkiksi listauksissa, joissa sisältö esitetään ruudukossa kuvien ja tekstin kera, kuvat häviävät näkyvistä, kun sivua zoomataan mobiililaitteella niin suureksi. Selaimessa tätä ongelmaa ei ilmene.

4.8 Palaute käyttäjälle

Auditoinnin perusteella sivuston ruudunlukijakäyttäjille antama palaute on puutteellista ja epäselvää. Kun käyttäjä on kirjautunut sisään, sivusto esittää erilaisia ilmoituksia onnistuneista tai epäonnistuneista toiminnoista visuaalisina ilmoituksina, mutta näitä ilmoituksia ei lueta ruudunlukijalla. Tämän seurauksena käyttäjä ei tiedä, mitä hänen tekemänsä toiminnot ovat aiheuttaneet tai ovatko ne onnistuneet. Lisäksi joidenkin painikkeiden tiedot ovat geneerisiä, eivätkä ne kerro, mihin sisältöön käyttäjän tekemä toiminto liittyy. Ruudunlukijalle suunnatut painikkeiden nimet eivät päivity samalla tavoin kuin visuaalisesti. Positiivisena seikkana mainittakoon, että sivuston latausprosessin edistymistä kuvaava palkki antaa kuitenkin käyttäjälle palautetta.

4.9 Kriittisimmät ongelmakohdat

Taulukossa 4 on auditoinnin perusteella löydetyistä kriittisimmistä ongelmakohdista, jotka tullaan korjaamaan opinnäytetyön rajausten sisällä. Nämä korjaukset parantavat selkeästi ruudunlukijakäyttäjien käyttökokemusta rajoittavia ongelmia. Muut auditoinnin perusteella löydetyt ongelmat sivustolta tullaan korjaamaan vähitellen.

Taulukko 4. Kohdesivustolta löydetyt kriittisimmät ongelmakohdat kuvauksineen.

Ongelma	**Kuvaus**	**Korjaus**
Ikonien puutteelliset vaihtoehtotekstit	Erityisesti toiminnallisissa ikoneissa on havaittu puuttuvia tai epäselviä kuvien tekstivastineita, mikä estää apuvälineitä käyttäviltä käyttäjiltä ikonien tarkoituksen ymmärtämisen. Mobiilinäytöllä valikko avautuu käyttäjälle klikkaamalla ikonista, joka kuvastaa valikkoelementtiä. Tähän ei ole lisätty kuvaavaa tekstivaihtoehtoa, ja ruudunlukija ei osaa kertoa käyttäjälle painikkeesta mikä sen tarkoitus on.	Luku 5.3.1.
Päävalikon navigointi ruudunlukijalla	Suuri osa navigaation linkeistä ei ole saavutettavissa virheellisen HTML-koodin käytön vuoksi tai koska on oletettu, että käyttäjä pystyy käyttämään hiirtä tai kosketuslevyä, koska valikon linkkien alavalikot avautuvat kohdistamalla hiiri valikon päälinkin päälle.	Luku 5.3.2
Modaalit, joiden sisältö ei ole ruudunlukijan saavutettavissa	Modaalit avautuvat, mutta niiden sisältö jää ruudunlukijoilta tavoittamatta, mikä estää sivuston käytön apuvälineillä.	Luku 5.3.3.
Toiminnalliset painikkeet puutteellisin palauttein	Sivustolla voi lisätä sisällön suosikkeihin omalle käyttäjätilille. Ruudunlukija ei anna tarpeellista palautetta siitä, että sisällön lisääminen suosikit-listaan onnistui. Lisäksi painikkeet eivät kerro mistä sisällöstä on kyse painikkeen geneerisen nimen vuoksi (”Lisää suosikkeihin”).	Luku 5.3.4
Zoomaus 200 %	Mobiililaitteella zoomatessa näyttöä 200 %, listauksissa ei näy kuvat. Lisäksi sisäänkirjautumisen mahdollistava painike ei aukaise sisäänkirjautumismodaalia.	Luku 5.3.5.
Muut käytettävyysongelmat, jotka vaikuttavat ruudunlukijan toimintaan	Karusellikomponentit, joissa on runsaasti sisältöä, tekevät navigoinnista ruudunlukijalla turhauttavaa, koska jokainen elementti on käytävä läpi erikseen.	Luku 5.3.6.

5 Toimenpidesuunnitelma ja korjaustoimenpiteet

Tässä luvussa käsittelen lähestymistapaani teknisten ongelmien korjaamiseen työelämässä, erityisesti saavutettavuushaasteiden osalta. Esittelen, mitä olennaista tietämystä saavutettavuusongelmien ratkaiseminen edellyttää, ja millä menetelmillä näitä ongelmia voidaan korjata. Lisäksi kerron työni rajauksen puitteissa havaitsemani kriittisimmät saavutettavuuspuutteet ja kuvaan tarkemmin, millaiset korjaustoimenpiteet näihin löydöksiin sovelletaan.

5.1 Korjausstrategia

Ohjelmistokehittäjät kohtaavat työelämässä jatkuvasti haasteita, joihin ei välttämättä ole valmiita ratkaisuja. Oma lähestymistapani tällaisiin tilanteisiin alkaa ongelman syvällisestä ymmärtämisestä ja sen nykyisen ilmenemismuodon hahmottamisesta. Tämän jälkeen etsin sivustolta kohdan, jossa ongelma esiintyy, ja paikannan vastaavan osuuden koodista voidakseni aloittaa korjaustyön.

Saavutettavuusongelmien ratkaisemisessa on olennaista ymmärtää, miten todelliset käyttäjät erilaisine rajoitteineen vuorovaikuttavat sivuston kanssa. Koska tässä tutkimuksessa keskityn erityisesti saavutettavuuteen ruudunlukijan avulla, on ollut tärkeää perehtyä syvällisesti ruudunlukijan käyttöön. Tämä on vaatinut sekä käytännön harjoittelua että teknisten ohjeiden ja dokumentaation huolellista lukemista. Jos ruudunlukijan käyttöön ei perehdy kunnolla, voi helposti tehdä turhaa työtä korjaamalla asioita, jotka itse asiassa toimivat oikein, mutta joita ei osaa navigoida ruudunlukijalla. Lisäksi puutteellinen osaaminen voi johtaa tahattomiin saavutettavuusongelmiin: yrittäessä korjata jotain voi vahingossa luoda uusia haasteita, jotka hämmentävät todellisia käyttäjiä. Huolellinen taustatyö auttaa tunnistamaan ongelmat käyttäjän näkökulmasta ja löytämään ratkaisuja, jotka parantavat sivuston saavutettavuutta kaikille käyttäjäryhmille. Tunnistan kuitenkin, että aikataulullisten rajoitteiden vuoksi en ole pystynyt syvällisesti perehtymään siihen, miten oikeat käyttäjät navigoivat ruudunlukijalla. Saavutettavuuden käytettävyyden testaaminen antaa aidon tuloksen vain, kun se tehdään todellisten käyttäjien kanssa. Tästä huolimatta uskon, että tekemäni korjaukset poistavat ainakin suurimmat saavutettavuuteen liittyvät ongelmat ja parantavat merkittävästi sivuston käytettävyyttä.

Löydetyt saavutettavuusongelmat priorisoidaan niiden vakavuuden perusteella. Tässä tutkimuksessa käytettävä Axe-saavutettavuustyökalu luokittelee löydökset eri vakavuusasteisiin (Deque Systems), joita kuvaan seuraavassa taulukossa (taulukko 5).

Taulukko 5. Axe-saavutettavuustyökalun löydöksien tasot ja kuvaus

Taso	Kuvaus
Critical (taso 3)	Tämän tason ongelmat tarkoittavat, että jokin sivuston toiminnallisuus ei toimi tai sisältö on ruudunlukijan käyttäjille tavoittamattomissa. Tämä voi pahimmillaan johtaa oikeustoimiin, jos sivusto kuuluu digipalvelulain piiriin.
Serious (taso 2)	Vaikka toiminnallisuus tai sisältö on ruudunlukijan käyttäjille osittain tavoittamattomissa, se ei ole sivuston perustoimintojen kannalta ratkaisevan tärkeä. Tämänkin tason ongelmat voivat kuitenkin aiheuttaa lakisääteisiä seuraamuksia.
Moderate (taso 1)	Sivuston perustoiminnallisuudet ovat käytettävissä ruudunlukijalla, mutta ongelmat voivat heikentää käyttökokemusta merkittävästi. Vaikka kyseessä ei ole kriittinen este, nämä virheet ärsyttävät käyttäjiä ja olisi suositeltavaa korjata ne.
Minor (taso 0)	Nämä ongelmat esiintyvät harvoin ja vain tietyissä tilanteissa. Jos ne voidaan korjata vähäisellä vaivalla ja riskillä, korjaaminen on suositeltavaa, mutta ei välttämätöntä.

Ensisijaisesti korjataan skannauksen tasolle 3 (critical) luokitellut löydökset, ja sen jälkeen edetään asteittain vähemmän vakaviin ongelmiin. Lisätietoja Axe-työkalusta löytyy luvusta 5 – Testauskäytäntöjen kehittäminen.

5.2 Käytännön toteutus

Chrome Devtools, VoiceOver-ruudunlukija ja Axe-devTools-skannaus ovat työkaluja, joita käytetään korjauksen toimivuuden testaamiseen saavutettavuuskriteeristön mukaisesti. Ruudunlukijalla navigointi on erityisen tärkeää, sillä skannaustyökalut eivät aina havaitse kaikkia mahdollisia ongelmia automaattisesti. Käytännössä korjauksen jälkeen siirrytään sivulle, aktivoidaan ruudunlukija ja aletaan navigoida. Jos navigointi on sujuvaa ja esimerkiksi tietyn toiminnon suorittaminen onnistuu pelkästään kuuntelemalla ruudunlukijaa, voidaan olettaa, että ongelma on saatu korjattua.

5.3 Kriittisimpien ongelmakohtien korjaustoimenpiteet

Suurin osa edellisessä luvussa luetelluista kriittisimmistä ongelmakohdista ovat olleet yksinkertaisia korjata. Nämä ongelmat johtuivat pääasiassa geneeristen elementtien, kuten div-elementtien, käytöstä toiminnallisuuksissa. Korjaukset eivät parhaimmillaan vaatineet muuta kuin oikean roolin määrittämisen kyseisille elementeille.

5.3.1 Ikonien puutteelisten vaihtoehtotekstien korjaus

Tämä painike oli kapeammilla näytöillä käytetty navigaatiomenun avaava painike, joka kuvattiin ikonina. Korjaus tapahtui antamalla elementille aria-label attribuutti, jonka arvo muuttui dynaamisesti siihen nähden, oliko navigaatiopalkki auki vai suljettu. Kun navigaatiopalkki on suljettuna, arvo on "Avaa valikko", ja kun se on avoinna, arvo muuttuu muotoon "Sulje valikko". Aria-label-attribuutin arvossa on hyödynnetty kielikäännöstä, joka mahdollistaa saman tekstin kääntämisen eri kielille ilman muutoksia itse komponenttiin. Koodiesimerkki korjatusta toteutuksesta on nähtävissä alla kuvassa 3.

```
1   <Button
2     className="no-borders"
3     onClick={() => {
4       changeMobileMenuState();
5     }}
6     as="div"
7     basic
8     data-testid="mobile-menu-button"
9     aria-label={t(
10      COMMON_NS,
11      isMobileMenuOpen
12        ? 'action_buttons.close_menu'
13        : 'action_buttons.open_menu'
14    )}
15  >
16    <MenuIcon isOpen={isMobileMenuOpen} />
17  </Button>
```

Kuva 3. Korjattu toteutus käyttää ikonina esitetyssä painikkeessa aria-label -attribuuttia, jonka arvo päivittyy sen mukaan, onko valikko auki vai suljettu.

5.3.2 Päävalikon navigoinnin korjaus ruudunlukijalle

Navigaation linkkeihin lisättiin HTML-painike, jossa käytettiin "sr-only" -tyyliluokkaa. Tämä luokka tekee elementin näkyväksi ainoastaan ruudunlukijoille, mutta piilottaa sen visuaalisesti. Ratkaisun avulla käyttäjä voi siirtyä näppäimistöllä ensimmäisen päälinkin jälkeen painikkeeseen, joka avaa alavalikon ilman tarvetta käyttää hiiren kohdistinta linkin päällä. Painikkeeseen on lisätty useita ARIA-attribuutteja. Aria-haspopup="true" ilmoittaa ruudunlukijalle, että painikkeen painaminen avaa valikon, dialogin, alavalikon tai muun vastaavan sisällön, mikä auttaa käyttäjää ymmärtämään, että elementin käyttö laajentaa sisältöä. Aria-expanded puolestaan kertoo, onko

kyseinen elementti jo avattuna vai suljettuna. Lisäksi aria-controls="submenu-{name}" yhdistää painikkeen siihen elementtiin, joka avautuu vastaavalla ID-arvolla (identiefier, eli tunniste), auttaen ruudunlukijaa yhdistämään interaktiot oikein. Aria-label on painikkeen nimi, jonka ruudunlukija lukee. Tässä tapauksessa “Avaa alavalikko {päävalikon nimi}”. Aria-labelin arvossa on käytetty kielikäännöstä. (Kuva 4.)

```
1  {hasSubLinks && (
2    <>
3      <button
4        className="submenu-toggle sr-only"
5        aria-expanded={secondLevelNaviOpen}
6        aria-haspopup="true"
7        tabIndex={0}
8        aria-label={t(COMMON_NS, 'action_buttons.open_submenu', { name })}
9        aria-controls={`submenu ${name}`}
10       onClick={() => setSecondLevelNaviOpen(!secondLevelNaviOpen)}
11     ></button>
12     <SecondLevelNavi
13       isOpen={secondLevelNaviOpen}
14       subLinks={subLinks}
15       currentPath={currentPath}
16       name={name}
17     />
18   </>
19 )}
```

Kuva 4. Korjattu navigaation alavalikkojen avaaminen toteutettiin lisäämällä HTML:ään button-elementti, johon sovellettiin "sr-only" -tyyliluokkaa. Tämä elementti on näkyvissä vain ruudunlukijoille, mikä mahdollistaa alavalikkojen avaamisen näppäimistöllä.

5.3.3 Focustrap-toiminnallisuuden toteuttaminen sivuston modaaleihin

Sivuston modaalit avautuivat, mutta ruudunlukija ei pystynyt selaamaan niitä, vaan fokus siirtyi modaalin taustalla oleviin elementteihin. Tämän ongelman ratkaisemiseksi modaalille toteutettiin focustrap-toiminnallisuus. Sen avulla ruudunlukijan fokus siirtyy modaalin avautuessa automaattisesti modaalin ensimmäiseen elementtiin ja pysyy modaalin sisällä, kunnes käyttäjä sulkee sen.

Tämä toiminnallisuus luotiin käyttämällä ref-viitettä, joka osoittaa modaalielementtiin. Viitteen avulla varmistetaan, että käyttäjän fokus pysyy modaalin sisällä eikä siirry muualle sivustolle. Jos fokus yrittää siirtyä modaalin ulkopuolelle, se estetään ja palautetaan takaisin modaalin sisään. Koodi ajetaan aina, kun komponentti päivittyy, eli esimerkiksi kun modaali avataan tai suljetaan.

Tämä onnistuu määrittelemällä focustrap -funktio Reactin useEffect -hookin sisällä. Kuvassa 5 näkuu toteutettu focustrap -toiminnallisuus.

```
1   useEffect(() => {
2     // Trap focus within the modal for screen reader users
3     const trapFocus = (event: FocusEvent) => {
4       if (modalRef.current && !modalRef.current.contains(event.target)) {
5         event.stopPropagation();
6         modalRef.current.focus();
7       }
8     };
9
10    document.addEventListener('focus', trapFocus, true);
11    return () => {
12      document.removeEventListener('focus', trapFocus, true);
13    };
14  }, []);
```

Kuva 5. Focustrap-toiminnallisuus, joka keskittää ruudunlukijan fokuksen avautuneeseen modaaliin ja estää navigoinnin modaalin taustalla oleviin elementteihin.

5.3.4 Palautteet ruudunlukijalle toiminnallisuuksista

Sivulla nouseviin ilmoituskomponentteihin lisättiin aria-elementit role=”alert” ja aria-live=”assertive”. Nämä kertovat ruudunlukijalle, että komponentti on ilmoitus, johon ruudunlukijan tulee reagoida välittömästi, eli lukea sen sisältämän sisällön ääneen.

Painikkeille, joilla käyttäjä voi lisätä sisältöä suosikiksi, on annettu kuvaavampi tekstivastine ruudunlukijalle. Nyt ruudunlukija lukee painikkeen sisällön muodossa "Lisää {sisällön nimi} suosikiksi" sen sijaan, että lukisi vain "Lisää suosikiksi", mikä saattaisi hämmentää ruudunlukijaa käyttäviä henkilöitä. Kuvassa 6 näkyy koodi, jolla tämä toiminnallisuus on toteutettu.

```
1   const saveButtonLabel = isFavorite
2     ? t(COMMON_NS, 'action_buttons.remove_from_favorites', {
3         name: contentItem.name,
4       })
5     : t(COMMON_NS, 'action_buttons.save_to_favorites', {
6         name: contentItem.name,
7       });
8
9   return (
10      <Button
11        icon={!isFavorite ? heartIcon : heartFilled}
12        id="favorite-button-logged-in"
13        dataTestId="favorite-button-logged-in"
14        className={compact ? 'bookmark-btn-compact' : 'bookmark-btn'}
15        inverse={!compact}
16        onClick={handleClick}
17        ariaLabel={saveButtonLabel}
18      >
19    )
```

Kuva 6. Korjattu toteutus painikkeesta, joka kertoo käyttäjälle mihin sisältöön painike liittyy. Käytössä kielikäännökset.

5.3.5 Zoomaus 200 %

Sivustolle oli määritelty eri leveyspisteet eri laitteille, kuten mobiililaitteille, tableteille ja kannettaville tietokoneille. Mobiililaitteen minimileveydeksi oli asetettu 320px. Kun käyttäjä suurensi sivua 200 % Safarin mobiiliselaimella, koko käyttäjälle verkkosivustolla näkyvä osa, eli näkymän leveys ”kapeni”. Tämä johti iOS-laitteilla siihen, että sivun leveys pieneni pikselimäärissä alle 320px. Ongelma vaikutti sivun kuviin ja ylänavigaation "kirjaudu käyttäjätilille" -painikkeeseen, jotka oli määritelty näkyviksi vain mobiililaitteille asetettujen leveyksien sisällä. Ongelma ratkaistiin laskemalla minimileveys 150px, jolloin sivusto toimi oikein myös zoomattuna. Alkuperäinen 320px määrittely oli todennäköisesti jäänyt vanhasta, jo poistuneesta koodista.

5.3.6 Käytettävyyteen liittyvät korjaukset

Sivustolla tuodaan esille sisältöä eri karusellikomponenteilla. Nämä ovat elementtejä, joissa näytetään sisältöä niin, että peräkkäiset osiot tulevat esiin yksi kerrallaan, luoden interaktiivisen ja dynaamisen katselukokemuksen. Kun karusellikomponentissa on runsaasti sisältöä, ruudunlukijaa käyttävälle voi olla turhauttavaa selata jokainen osio läpi vain päästäkseen seuraavaan tärkeään elementtiin. Tämän ongelman ratkaisemiseksi sivustolle on lisätty painike, joka on näkyvissä vain ruudunlukijan käyttäjille. Tämän painikkeen avulla käyttäjä voi ohittaa koko karusellin ja jatkaa

selaamista nopeammin, ilman että hänen tarvitsee käydä läpi kaikkia karusellin sisältöjä yksi kerrallaan.

```
1  <a
2    ref={carouselRef}
3    href="#"
4    onClick={(e) => {
5      e.preventDefault();
6      handleSkipToNextContent({ ref: carouselRef });
7    }}
8    className="sr-only">
9      Ohita karuselli
10 </a>
```

Kuva 7. Ruudunlukijan käyttäjille näkyvä linkki mahdollistaa karusellin ohittamisen sivulla.

Kuvassa 7 oleva linkki on varustettu ref-viitteellä, joka tallentaa karusellin sijainnin ja ohjaa ruudunlukijan seuraavaan löydettävään interaktiiviseen elementtiin, jolloin käyttäjä voi jatkaa navigointia sujuvasti ilman tarpeetonta sisällön selaamista. Linkistä painamalla toteutuu toiminnallisuus ”handleSkipToNextContent” – joka etsii seuraavan interaktiivisen elementin sivulla ja siirtää kohdistuksen siihen (kuva 8).

```
1  const getFocusableElements = (container = document) => {
2    return Array.from(
3      container.querySelectorAll(
4        'a[href], button, textarea, input, select, [tabindex]:not([tabindex="-1"])'
5      )
6    ).filter((el) => !el.hasAttribute('disabled'));
7  };
8
9  const getNextFocusableElement = (currentElement: HTMLElement) => {
10   const focusableElements = getFocusableElements();
11   const currentIndex = focusableElements.indexOf(currentElement);
12   return focusableElements[currentIndex + 1] || null;
13 };
14
15 const handleSkipToNextContent = ({
16   ref,
17 }: {
18   ref: React.MutableRefObject<any>;
19 }) => {
20   const nextFocusableElement = getNextFocusableElement(ref.current);
21   if (nextFocusableElement) {
22     (nextFocusableElement as HTMLElement).focus();
23   }
24 };
```

Kuva 8. Funktio, joka tunnistaa seuraavan interaktiivisen elementin ja siirtää ruudunlukijan fokuksen siihen.

6 Testauskäytäntöjen kehittäminen

WCAG 2.1 -saavutettavuuskriteerien noudattamiseksi kehittäjien on varmistettava, että he eivät luo uusia esteitä sivuston kehitystyössä. Erilaisten elementtien ja toiminnallisuuksien toteutustapojen moninaisuus voi helposti johtaa esteiden syntymiseen, joten niiden huolellinen välttäminen on tärkeää. Saavutettavuuden testaaminen on helppoa ja yksittäisten elementtien kohdalla todella nopeaa. Sen sijaan saavutettavuusesteiden korjaaminen jälkikäteen voi olla huomattavasti työläämpää ja vaatia laajaa refaktorointia, eli koodin rakenteen parantamista, ilman että toiminnallisuus muuttuu. Siksi on paljon tehokkaampaa tunnistaa ja ratkaista mahdolliset ongelmat jo kehitysvaiheessa. Ottamalla saavutettavuustestauksen osaksi kehitysprosessia, sivuston saavutettavuus säilyy jatkuvasti hallinnassa ja mahdolliset ongelmat voidaan havaita ja korjata varhaisessa vaiheessa.

6.1 Manuaalisen testauksen menetelmät

Uusia elementtejä kehitettäessä kehittäjien tulisi aktiivisesti testata niitä ruudunlukijalla manuaalisesti varmistaakseen, että elementit ovat saavutettavia ja toimivat sujuvasti myös näppäimistönavigoinnilla, aivan kuten hiirellä. Elementtien testaamiseen ja niiden läpi navigoimiseen tulisi käyttää ruudunlukijaa, joka on yhteensopiva käyttöjärjestelmän ja selaimen kanssa (katso alla oleva taulukko yhteensopivista selain + ruudunlukijayhdistelmistä). Ruudunlukijat antavat yleensä käyttäjille ohjeita siitä, miten elementtien kanssa vuorovaikutetaan. Ruudunlukijaa kannattaa aina käyttää siihen tarkoitetulla ruudunlukija + selain -yhdistelmällä. Näin navigointikokemus on käyttäjälle paras mahdollinen. Esimerkiksi Safari ja Chrome -selainten toimintatapa eroaa toisistaan, joten Chromelle suunniteltu ruudunlukija ei välttämättä ymmärrä kaikkia Safari-selaimen ominaisuuksia. Taulukossa 6 on kuvattu toisilleen yhteensopivia ruudunlukija + selain -yhdistelmiä:

Taulukko 6. Yhteensopivat ruudunlukijan ja selaimen yhdistelmät

Ruudunlukija	**Selain**
VoiceOver	Safari
Windows Narrator	Microsoft Edge
JAWS & Google Talkback (Android)	Google Chrome
NVDA	Firefox

Elementtien saavutettavuutta voidaan testata ruudunlukijan lisäksi erilaisilla palveluilla ja selaimen lisäosilla, jotka analysoivat verkkosivun antamalla sen osoitteen. Esimerkkeinä tällaisista työkaluista ovat esimerkiksi Wave Web Evaluation Tool, josta on saatavilla myös Google Chrome -selainlaajennus, sekä vaihtoehtoisesti mahdollisuus syöttää suoraan verkkosivun osoite Waven sivustolla. Näitä työkaluja ei kuitenkaan tulisi käyttää yksinomaan, sillä ne pystyvät tunnistamaan ilmeisiä saavutettavuusongelmia, kuten puuttuvia kuvan vaihtoehtotekstejä, mutta eivät löydä ongelmia, joita ruudunlukija ei huomaa. Esimerkiksi jos painike on toteutettu div-elementtinä ilman roolia, ruudunlukija ei tunnista sen tarkoitusta eikä lue sitä, eikä myöskään automaattiset skannaustyökalut havaitse tätä.

6.2 Testien automatisointi

Ohjelmistokehityksessä käytetään nykyään usein niin kutsuttua jatkuvan käyttöönoton ja integroinnin (Continuous Integration & Continuous Deployment, CI/CD) mallia. Tässä mallissa uusien muutosten julkaiseminen tuotantoympäristöön on tehty mahdollisimman nopeaksi ja helpoksi automatisoimalla monia työvaiheita, jotka kehittäjät ovat perinteisesti tehneet manuaalisesti. Näitä työvaiheita ovat esimerkiksi automaattiset testit, ohjelmiston julkaisu sekä "build"-prosessi, jossa sovellus käännetään ja valmistellaan julkaistavaksi. Nämä vaiheet ja niiden järjestys määritellään yleensä tiedostossa, jota eri versionhallintaohjelmistot osaavat lukea. Tätä tiedostoa ja koko työvaihetta kun automatisoidut vaiheet käynnistyvät, kutsutaan usein CI-putkeksi,

On olemassa erilaisia testityyppejä, kuten yksikkötestit, joissa arvioidaan yksittäisten toimintojen tai komponenttien toimivuutta, sekä end-to-end (E2E) -testit, jotka simuloivat julkaistun ohjelmiston toimintaa ohjelmallisesti. E2E-testeissä koodikanta käynnistetään sisäisessä ympäristössä, jossa testikirjasto arvioi sivuston toimivuutta automaattisesti navigoiden sivustolla, kuten oikea käyttäjä. Tässä opinnäytetyössä toteutin brändisivuston CI-putkeen saavutettavuustestauksen tukemiseksi automaattiset E2E-testit, jotka kykenevät löytämään määritetyistä sivuista mahdollisia saavutettavuusongelmia. Nämä testit suoritetaan aina, kun uusia muutoksia tuodaan prosessin aikana eteenpäin, eli kun kehittäjä on tehnyt koodiin muutoksen ja lähettää sen versionhallintaan katselmoitavaksi. Jos testit epäonnistuvat, CI-putki pysähtyy eikä siinä olevia muutoksia tulisi julkaista.

Koodikannassa on jo valmiiksi toteutettu end-to-end (E2E) testejä Microsoftin kehittämällä avoimen lähdekoodin Playwright -kirjastolla, joka on laajalti käytössä moderneissa web-sivustoissa. Saavutettavuustestien avuksi otettiin käyttöön Playwrightin lisäosa axe-core.

6.2.1 Saavutettavuustestien toteutus CI-putkeen

Ensimmäiseksi määrittelin räätälöidyt komennot package.json-tiedostossa, joita käytetään CI-putkessa. Tämä ei ole pakollista, mutta räätälöityjen komentojen luominen selkeyttää koodin lukemista, sillä komennot voivat sisältää erilaisia optimointeja, miten testi ajetaan ja mitä tietoja se käyttää avukseen.

```
1  "test:accessibility": "playwright test --config=playwright/playwright.config.ts --grep=accessibility --max-failures=0",
2  "test:axe:summarize": "node playwright/tests/utils/summarize-axe-data.mjs",
```

Kuva 9: Räätälöydyt komennot saavutettavuustestien ajamiseen yksinkertaistavat ajettavia komentoja ja helpottavat koodin lukemista.

Kuvassa 9 näkyvä komento “test:accessibility” ajaa testin Playwright kirjastolla, joka ottaa avukseen konfigurointitiedoston, valitsee kaikki playwright-testit, joiden nimessä on accessibility, ja asetus max-failures=0 sallii testien jatkamisen, vaikka jokin testi epäonnistuisi. Komento “test:axe:summarize” ajaa node.js skriptin, joka tiivistää testien löydökset selkeästi luettavaan muotoon.

```
1  accessibility-tests:
2    extends: .playwright-e2e-base
3    stage: test
4    needs:
5      - job: build-and-verify
6    dependencies: [build-and-verify]
7    script:
8      - yarn serve:bg --log app.log
9      - yarn test:accessibility
10   after_script:
11     - cd packages/xxx
12     - yarn stop:bg
13     - yarn test:axe:summarize
```

Kuva 10. CI-putkeen rakennettu osio saavutettavuustestien ajamiseen.

Yllä olevassa kuvassa (kuva 10) on CI-putken saavutettavuustestien määrittely, nimeltä accessibility-tests. Se perustuu playwright-e2e-base-määrittelyyn ja sijoittuu putken eri vaiheissa testivaiheeseen. Työ vaatii ensin build-and-verify-tehtävän suorittamista ja käyttää sen riippuvuuksia, eli valmiiksi käännettyä kooditiedostoa. Skriptissä käynnistetään taustapalvelin ja suoritetaan saavutettavuustestit ajamalla komento ”yarn test:accessibility”. Testien jälkeen siirrytään oikeaan

hakemistoon, sammutetaan taustapalvelin ja tiivistetään testitulokset räätälöidyllä komennolla ”yarn test:axe:summarize”.

6.2.2 Testien toteutus

Komento ”yarn test:accessibility” ajaa @playwright/axe-core-kirjaston avulla toteutetut E2E-testit. Testeille on määritelty ennalta sivustolta eri tyylisiä sivuja, joiden tulisi kattaa kaikki mahdolliset kohdesivuston sivupohjat. Nämä sivut on annettu listassa, joissa kerrotaan sivun nimi (name) ja osoite (url). Kun testi käynnistetään, ajetaan vuorotellen kaikilla sivuilla sama testi. Testi kokonaisuudessaan on nähtävillä kuvassa 11.

Ensimmäiseksi testi odottaa 9 sekuntia, jotta sivut ehtivät ladata sisältönsä ennen lopputestin suorittamista. Vaikka tämä ei ole optimaalinen ratkaisu, muuttuvan sisällön ja hakuprosessien vuoksi se on koettu tarpeelliseksi E2E-testeissä. Odotusajan jälkeen siirrytään ohjelmallisesti ensimmäisen sivun osoitteeseen ja tarkistetaan, onko sivu renderöitynyt oikein – eli onko koodi muuntunut visuaalisiksi elementeiksi. Tämä varmistetaan tarkistamalla, että jokaisella sivulla näkyvä alaosio, eli footer, on löydettävissä.

Tämän jälkeen sivu analysoidaan AxeBuilder-funktiolla, jolle annetaan ohjeeksi, ettei värikontrastivirheistä raportoida virhetilaa, koska värien määrittely on design-päätös, eikä niille ole vielä löydetty vaihtoehtoisia värejä. Analyysin tulokset tallennetaan muuttujaan accessibilityScanResults. Kun analyysi on suoritettu, tuloksia verrataan edellisiin tuloksiin, ja mikäli versiot eivät vastaa toisiaan, testi epäonnistuu.

```
1  Array.from(allPaths).forEach((path) => {
2    test.describe(path.name, () => {
3      test('should not have any automatically detectable accessibility issues, beyond those in the snapshot', async ({
4        page,
5      }, testinfo) => {
6        test.setTimeout(90000);
7
8        await page.goto(baseUrl + path.url);
9        const footerText = page.getByText('footer text');
10       const footerLogo = page.getByAltText('footer logo');
11       await page.waitForTimeout(10000);
12       await expect(footerText.or(footerLogo).first()).toBeVisible();
13
14       const accessibilityScanResults = await new AxeBuilder({
15         page,
16       })
17         .disableRules(['color-contrast'])
18         .analyze();
19
20       testinfo.snapshotSuffix = 'axe.json';
21
22       expect(
23         violationFingerprints(
24           accessibilityScanResults as AccessibilityScanResults
25         )
26       ).toMatchSnapshot(path.name);
27     });
28   });
29 });
```

Kuva 11. @playwright/axe-core-kirjaston avulla tehty testi, jolla testataan eri sivujen saavutettavuutta.

Funktio violationFingerprints (kuva 12) muuntaa testin tulokset helposti luettavaan muotoon ja tallentaa tiedot tiedostoon, jonka nimi on muodossa "sivun_nimi+axe.json", eli esimerkiksi etusivu-axe.json.

```
function violationFingerprints(
  accessibilityScanResults: AccessibilityScanResults
): string {
  const violationFingerprints: ViolationFingerprint[] =
    accessibilityScanResults.violations
      .map((violation) => ({
        rule: violation.id,
        impact: violation.impact,
        helpText: violation.help,
        description: violation.description,
        targetCount: violation.nodes.length,
        AffectedHtml: violation.nodes.map((node) => node.html)[0], // log only the first html element
        failureSummary: violation.nodes.map((node) => node.failureSummary)[0], // log only the first summary
      }))
      .sort((a, b) => a.rule.localeCompare(b.rule));

  return JSON.stringify(violationFingerprints, null, 2);
}
```

Kuva 12: violationFingerPrints -niminen funktio, joka muuntaa analyysin tulokset selkeään muotoon ja kertoo kehittäjille missä saavutettavuusvirhe sijaitsee koodissa.

Jokaisen sivun testin jälkeen tarkistetaan, onko testi epäonnistunut, eli testin tila (testInfo.status) on erilainen kuin odotettu tila (testInfo.expectedStatus). Jos näin on, kirjasto ottaa kuvakaappauksen koko sivusta ja tallentaa sen tiedostoon screenshot.png. Kuvakaappaus liitetään testin tuloksiin, jolloin se on saatavilla myöhemmin tarkasteltavaksi. Kuvakaappaus hyödyntää kehittäjiä siten, että he pääsevät katsomaan onko virhe todellinen saavutettavuusvirhe, vai esimerkiksi ongelma sivun sisällön latauksen kanssa. (Kuva 13.)

```
test.afterEach(async ({ page }, testInfo) => {
  if (testInfo.status !== testInfo.expectedStatus) {
    const screenshotPath = testInfo.outputPath('screenshot.png');
    testInfo.attachments.push({
      name: 'screenshot',
      path: screenshotPath,
      contentType: 'image/png',
    });
    await page.screenshot({
      path: screenshotPath,
      fullPage: true,
    });
  }
});
```

Kuva 13. Epäonnistuneen testin jälkeen tallennetaan näyttökuva löydetystä sivusta saavutettavuusongelmineen.

6.2.3 Saavutettavuustestien tulokset

Epäonnistuneiden testien tulokset tallennetaan tiedostoon, jonka nimi on epäonnistuneen testin sivun nimi + axe.json. Tiedostosta kehittäjä näkee ensinnäkin, mikä WCAG-standardin sääntö on rikottu ja sen vakavuuden (esimerkiksi "kohtalainen"). Lisäksi tiedot sisältävät aputekstin ja selityksen, joka kertoo, kuinka monta samanlaista ongelmaa sivulta löytyy, mihin HTML-koodin kohtaan ongelma liittyy ja miten se voidaan korjata.

```
1  [
2    {
3      "rule": "aria-command-name",
4      "impact": "serious",
5      "helpText": "ARIA commands must have an accessible name",
6      "description": "Ensure every ARIA button, link and menuitem has an accessible name",
7      "targetCount": 1,
8      "AffectedHtml": "<div role=\"button\" tabindex=\"0\">",
9      "failureSummary": "Fix any of the following:/n
10       Element does not have text that is visible to screen readers\n
11       aria-label attribute does not exist or is empty\n
12       aria-labelledby attribute does not exist,
13       references elements that do not exist or references elements that are empty\n
14       Element has no title attribute"
15   }
16 ]
```

Kuva 14. Esimerkki saavutettavuustestin tuloksesta, josta on löytynyt mahdollinen ongelmakohta saavutettavuuden kannalta.

Kuvassa 14 oleva tulos on oikea esimerkki kohdesivustolta epäonnistuneen testin tuloksesta. Kyseiseltä sivulta on löytynyt tason 2 vakava saavutettavuusongelma, jossa painikkeena toteutetulle div-elementille ei ole annettu nimeä, jonka ruudunlukija osaa lukea. Tämä on tosiasiassa jo aiemmin korjattu painike, joka oli visuaalisesti toteutettu ikonina, mutta sille ei ollut annettu tekstivastinetta.

7 Tulokset ja jatkotutkimusaiheet

Kun projektia aloitettiin, kohdesivustolla oli vakavia, toiminnallisuuksiin vaikuttavia puutteita, joiden vuoksi näppäimistönavigoinnilla sivuja selailevat käyttäjät jäivät pois mahdollisuudesta käyttää sivun kaikkia toiminnallisuuksia. Tämän projektin myötä kohdesivustolta on korjattu kriittisimmät ongelmakohdat digipalvelulain ja WCAG 2.1. A & AA-tasojen kriteerien mukaan. Konkreettiset korjaukset vaikuttavat ainakin huononäköisten ja sokeiden käyttäjien kokemukseen, jotka turvautuvat joko ruudunlukijaan apuvälineenä tai zoomaavat tekstiä.

Koska sivustolla oli paljon työtä ja projektiin asetettu aikaraja oli rajallinen, sivustolle jätettiin myöhempään korjaukseen saavutettavuusongelmia, jotka Axe-saavutettavuustyökalu luokitteli tasoille ”Moderate” ja ”Minor”. Näen nämä ongelmat kuitenkin mahdollisuutena tiimilleni – kehittäjät voivat hyödyntää projektin myötä syntynyttä dokumentaatiota ja oppeja näiden ongelmien ratkaisemisessa. Näin he voivat halutessaan vähitellen kehittää omaa osaamistaan ja varmistaa, että sivuston saavutettavuus säilyy hyvällä tasolla myös tulevaisuudessa. Suunnitelmana on korjata jäljelle jääneet saavutettavuusongelmat alkuvuodesta 2025.

Osaksi hyvän saavutettavuustason ylläpitoa edistää myös jatkuva testauskäytäntö ja CI-putkeen rakennettu saavutettavuustestauksen osio. Automaattinen saavutettavuustestaus muistuttaa kehittäjiä kiinnittämään huomiota sivuun myös saavutettavuuden osalta jokaisen muutoksen jälkeen, jonka he vievät versionhallintaan. Saavutettavuustestauksen selkeä ongelmakohta on se, että sivustolla on valtava määrä eri sivuja, ja niissä voi olla erilaista sisältöä. Näitä kaikkia sivuja ei ole järkevää eikä luultavasti myös mahdollista skannata automaattisessa CI-putkessa, vaan tällä hetkellä sivuja on valittu eri sisältöpohjilta kaikilta yksi. Tätä vaihetta voisi jatkokehittää esimerkiksi niin, että testit tekisivät skannaukset noin kymmenelle uusimmalle sisällölle. Näin työkalu testaisi eri sivuja laajasti, ja myös sellaiset ongelmakohdat voisivat löytyä, jotka eivät ilmene muissa sivuissa eri komponenttien vuoksi.

Työn tuloksena syntynyt dokumentaatio sivuston saavutettavuustestauksesta on tarkoitettu muille kehittäjille hyödynnettäväksi, kun he luovat sivustolle uusia ominaisuuksia. Dokumentaatio sisältää:

- Selitykset WCAG 2.1 -standardin tasoista ja muutaman esimerkkitapauksen eri tasojen onnistumiskriteereistä.
- Lyhyen ohjeistuksen ruudunlukijalla navigointiin
- Suositukset eri ruudunlukijoiden käytöstä eri selaimissa ja käyttöjärjestelmissä
- Listauksen saavutettavuuden testaustyökaluista, mukaan lukien sekä manuaaliset että automatisoidut työkalut
- Kokoelman hyödyllisiä linkkejä, jotka tarjoavat lisäohjeita ja tietoa saavutettavuudesta

Lisäksi dokumentaatiossa korostetaan saavutettavuuden jatkuvan ylläpitämisen tärkeyttä sekä työn merkitystä käyttäjien saavutettavuuden parantamisessa. Mainittakoon vielä, että jatkossa sivustolle tehdään puolivuosittain saavutettavuusarviointi koko tiimin yhteistyönä. Arvioinnissa sivusto käydään perusteellisesti läpi eri työkalujen avulla, ja löydetyt saavutettavuusongelmat lisätään tiimin työlistalle korjattaviksi.

Tällä hetkellä ei ole tiedossa kuinka paljon sivua käytetään apuvälineillä, eikä tiedon hankkiminen ole mahdollista, sekä teknisistä että eettisistä syistä. Tim Frick lainaa blogitekstissään ”How Many People with Disabilities Use My Website” Dennis Deaconia, Chicago Digital Accessibility & Inclusive Design -tapahtuman järjestäjää, joka selittää, ettei apuvälineiden, kuten ruudunlukijoiden, käyttöä voida teknisesti tunnistaa esimerkiksi analytiikkapalveluissa. Tämä johtuu siitä, että ruudunlukija saa verkkosivun tiedot selaimelta, ei suoraan verkkosivulta. Tästä syystä verkkosivu ei ole tietoinen siitä, että ruudunlukija on käytössä sivupyynnössä. (Frick 3.5.2024.) Mielestäni tämä on positiivinen asia. Jos yritykset tekisivät päätöksensä saavutettavuuden parantamisesta esimerkiksi ruudunlukijaa käyttävien käyttäjämäärien perusteella, monien sivustojen saavutettavuutta ei pidettäisi prioriteettilistalla korkealla. Ruudunlukijan käyttäjiä on kuitenkin huomattavasti vähemmän kuin perinteisiä selaajia. Moni saattaakin ajatella, että näin pienen käyttäjäryhmän vuoksi saavutettavuus ei ole sivustollamme tärkeää – ja juuri tällainen ajattelu johtaa inklusiivisuuden unohtumiseen. Olisi mielenkiintoista kuitenkin tietää, mitä ruudunlukijan käyttäjät kokevat sivuston saavutettavuuden – onko tehty työ oikeasti onnistunut. Tämän voisi toteuttaa esimerkiksi kyselytutkimuksella sivustolla, jossa ruudunlukijan käyttäjät voisivat ilmoittautua halutessaan testaamaan sivua ja raportoimaan löydöksistään. Oikea käyttäjädata on varsinkin saavutettavuuden kannalta todella oleellista, koska henkilö ilman rajoitteita ei pysty täysin eläytymään rajoitteellisen kokemukseen.

Tässä työssä ei ole testattu muita ruudunlukijoita, kuin Applen VoiceOveria. Jatkotutkimusaiheena kokisin tärkeäksi, että heti ensi kädessä testattaisiin sivuston toimivuutta myös muilla ruudunlukijoilla ja korjattaisiin testauksesta löydetyt mahdolliset ongelmakohdat. Sivuston tulisi toimia kaikilla ruudunlukijoilla saumattomasti.

8 Pohdinta

Kun tiimilleni pidettiin viime keväänä saavutettavuuskoulutus, testasimme yhdessä kohdesivustoa ruudunlukijan avulla. Muistan, kuinka meitä hieman huvitti, kuinka huonosti sivu oli rakennettu saavutettavuuden näkökulmasta. Jos sivustoa tarkastellaan WCAG 2.1 -standardin ja digipalvelulain kriteerien valossa, se ei olisi täyttänyt kaikkia vaatimuksia. Vaikka osa elementeistä oli selvästi suunniteltu saavutettavuutta silmällä pitäen, nämä olivat yksittäisiä kohtia tai peräisin ulkoisista kirjastoista, eivätkä ratkaisut olleet laajasti käytössä sivustolla.

Mielestäni suurin ongelma sivuston kehityksessä saavutettavuuden kannalta oli tiedon puute. Saavutettavien verkkosivustojen rakentaminen ei teknisesti vaadi mitään erikoisteknologioita – lähes kaikki voidaan toteuttaa perinteisellä HTML:ä ja koodikielellä, jota muutenkin käytetään sivustolla. Tärkein oppi on ymmärrys siitä, miten apuvälineitä käyttävät henkilöt käyttävät sivustoja. Kun ajattelen, kuinka turhauttavaa ruudunlukijan käyttäjälle on navigoida karusellissa, jossa on 50 sisältöä ilman ohituslinkkiä, ymmärrän täysin, miksi käyttäjä poistuu sivustolta eikä koskaan enää palaa. Lisäksi semanttisen HTML:n merkitys saavutettavuuden kannalta on ollut todella tärkeä oppi. Olen itsekin tehnyt kohdesivustolle tietämättömyyttäni esteitä, kun olen käyttänyt esimerkiksi geneeristä div-elementtiä painikkeen tekoon, koska olen varmaan olettanut, että div-elementin tyylimuotoilu on helpompaa kuin button-elementin. Tämä ei pidä paikkaansa.

On hienoa, että EU-lainsäädäntö alkaa edistää sivustojen ja palveluiden tasa-arvoista saavutettavuutta kaikille. Olisi toki ihanteellista, jos tämä tapahtuisi ilman lainsäädännön pakkoa, mutta todellisuudessa monet toimijat eivät lähtisi toteuttamaan näitä muutoksia, ellei taustalla olisi pakottavia syitä. Nykyisen hallituksen ehdotukset ja päätökset leikata tuista, jotka kohdistuvat kaikkein heikoimpiin, kuten leikkaukset vammaisten selkokielisistä oppimateriaaleista, ovat esimerkki siitä, kuinka epätasa-arvoista elämä voi monille olla. Juuri nämä ryhmät, kuten ikääntyneet ja vammaiset, jotka usein käyttävät ruudunlukijoita, ovat erityisen haavoittuvassa asemassa. Vaikka kohdesivusto ei välttämättä ole juuri se sivusto, jonka nämä ryhmät tuntevat tarpeelliseksi, ainakin heillä on yhtäläinen mahdollisuus kuluttaa sivuston sisältöä, kuin henkilöt, joilla ei ole mitään rajoitteita.

Yrityksen näkökulmasta työni tulokset, eli brändisivuston saavutettavuuden parantaminen ja testauskäytäntöjen kehittäminen, ovat varmasti positiivinen saavutus. Sivustolla on nyt vahva perusta, jolle saavutettavuutta voidaan jatkossa edelleen parantaa. Suurimmat ongelmat on korjattu, eikä sivustolla ole enää ilmeisiä esteitä käytettävyydelle. Kehitystiimi on saanut koulutusta, jotta tulevat ominaisuudet voidaan suunnitella saavutettaviksi, ja saavutettavuuden testaamisen automatisointi varmistaa, että tämä näkökulma otetaan huomioon aina, kun koodia muutetaan ja viedään versionhallintaan. Tämä jatkuva käytäntö kehittää koko tiimin osaamista. Vaikka sivuston ei lainsäädännön puitteissa tarvitse täyttää saavutettavuusvaatimuksia, uskon, että saavutettavuuden

parantaminen on positiivinen yllätys apuvälineitä käyttäville. Samalla tämä myös vahvistaa brändin positiivista mielikuvaa.

Oma oppimispolkuni saavutettavuuden parissa on ollut melko jyrkkä nousu. Olin kyllä kuullut aiheesta aiemmin ja tiesin sen olevan tärkeä, mutta kuvittelin käytännön toteutuksen olevan monimutkaista pilkunviilausta. Mitä enemmän opin aiheesta, sitä enemmän kiinnostuin siitä ja halusin syventää osaamistani. Suurin syy kiinnostukselleni oli se, että koin aiheen todella merkityksellisiksi – minulla oli mahdollisuus tehdä konkreettisia tekoja inklusiivisuuden edistämiseksi IT-alalla, joka mielestäni on perinteisesti ollut jäykkä ja miesvaltainen, ja jossa raha usein ohjaa päätöksentekoa. Tulen ehdottomasti jatkamaan oman osaamiseni syventämistä saavutettavuuden parissa ja uskon, että tämä projekti tulee muovaamaan koko uraani.

Työnantajani tarjosi minulle saavutettavuuskoulutuksen jo aiemmin, ja aloitin opinnäytetyöni suurin piirtein samoihin aikoihin kuin tämän kurssin. Tarkoitukseni oli opiskella aiheesta ensin sen verran, että osaan lähteä ratkaisemaan ongelmia oikeaoppisesti. Huomasin kuitenkin, että oma oppimistapani ei ole videoiden katsominen tai dokumentaation lukeminen. Vaikka tunnistan, että tekninen dokumentaatio on usein ratkaiseva apu ongelmiin, en hyödy siitä samalla tavalla ilman käytännön kontekstia. Myös tämän projektin loppuvaiheessa joudun korjaamaan projektin alkuvaiheen vähemmän onnistuneita ratkaisuja, kun ymmärrykseni kasvoi. Tunnistan, että tämänkin projektin tuotoksena syntyneet kaikki ratkaisut eivät ole parhaita mahdollisia. Vaikka saavutettavuuden taso on parempi nyt mitä aiemmin, jouduin jossain kohdin käyttämään vähän rumia tapoja saavuttaakseni toimivan ratkaisun aikarajojen puitteissa. Rajoite Applen VoiceOver-ruudunlukijaan on myös oikea ongelma. Sivustoa ei ole tämän projektin puitteissa testattu lainkaan muilla ruudunlukijoilla, ja on tosiasia, että ruudunlukijat käyttäytyvät hiukan eri tavalla. Uskon kuitenkin, että jos ongelmia on muiden ruudunlukijoiden kohdalla, ongelmat eivät ole valtavia.

Tämä projekti on mielestäni ollut yksi koko työurani parhaita projekteja. Olen tyytyväinen työni tulokseen, sekä opinnäytetyön muodossa, että konkreettisiin parannuksiin sivustolla. Vaikka sivusto ei ole vielä saavutettavuuden kannalta täydellinen, olen oppinut valtavasti uutta, innostunut ja motivoitunut.

Lähteet

Deque Systems. Rule Impact Levels. Versio 2023.4.19. Luettavissa: https://docs.deque.com/dev-tools-mobile/2023.4.19/en/impact. Luettu: 24.9.2024.

Directive (EU) of the European parliament and of the council on the accessibility requirements for products and services 17.4.2019/882. Luettavissa: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019L0882. Luettu: 12.10.2024.

Etelä-Suomen Aluehallintovirasto. Digipalvelulain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/. Luettu: 19.8.2024.

Etelä-Suomen Aluehallintovirasto. Kenelle saavutettavuus on tärkeää? Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa/. Luettu: 31.8.2024.

Etelä-Suomen Aluehallintovirasto. Soveltamisala: Kuulummeko lain piiriin? Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/soveltamisala-kuulummeko-lain-piiriin/. Luettu: 19.8.2024.

Etelä-Suomen Aluehallintovirasto. WCAG 2.1: lain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/wcag-2-1/. Luettu: 18.9.2024.

European Commission. European Accessibility Act: Q & A. Luettavissa: https://ec.europa.eu/social/main.jsp?catId=1202&intPageId=5581&langId=en. Luettu: 7.10.2024.

European Commission. European Accessibility Act. Luettavissa: https://ec.europa.eu/social/main.jsp?catId=1202#:~:text=The%20European%20accessibility%20act%20covers,accessibility%20requirements%20across%20EU%20countries. Luettu: 12.10.2024.

Frick, T. 3.5.2024. How Many Peple With Disabilities Use My Website?. Mightybytes-ohjelmistoyrityksen blogi. Luettavissa: https://www.mightybytes.com/blog/how-many-users-with-disabilities-on-site/. Luettu: 26.10.2024.

Graham G. 2022. Just How Long Should Alt Text Be?. CSS-tricks. Luettavissa: https://css-tricks.com/just-how-long-should-alt-text-be/. Luettu: 12.10.2024.

Kallionpää, R & Kiiskilä, R. 2021. Suomalaisten ruudunlukijakäyttäjien tottumukset ja haasteet verkkopalveluiden käytössä – kyselytutkimuksen tulokset. Eficode. Helsinki. Luettavissa: https://cms.nkl.fi/sites/default/files/2021-

09/Suomalaisten%20ruudunlukijak%C3%A4ytt%C3%A4jien%20tottumukset%20ja%20haasteet%20verkkopalvelujen%20k%C3%A4yt%C3%B6ss%C3%A4.pdf. Luettu: 21.9.2024.

Kehitysvammatuki Ry & Kehitysvammaliitto. Miksi kognitiivinen saavutettavuus on tärkeää. Luettavissa: https://www.selkeastimeille.fi/kognitiivinen-saavutettavuus/miksi-kognitiivinen-saavutettavuus-on-tarkeaa/. Luettu: 19.8.2024.

Kehitysvammatuki Ry & Kehitysvammaliitto. Mitä on kognitiivinen saavutettavuus. Luettavissa: https://www.selkeastimeille.fi/kognitiivinen-saavutettavuus/mita-on-kognitiivinen-saavutettavuus/. Luettu: 28.10.2024.

Laki Digitaalisten palveluiden saavutettavuudesta 15.3.2019/306. Luettavissa: https://www.finlex.fi/fi/laki/alkup/2019/20190306#Lidm46111190770256. Luettu: 12.10.2024.

Lindholm, S. 21.4.2020. Saavutettavuus ja hakukoneoptiomointi – 6 helppoa vinkkiä. Karhu Helsingin blogi. Luettavissa: https://www.karhuhelsinki.fi/blogi/saavutettavuus-ja-hakukoneoptimointi-6-helppoa-vinkkia/. Luettu: 31.8.2024.

Material Design. Alt text. Luettavissa: https://m3.material.io/foundations/content-design/alt-text. Luettu: 16.10.2024.

MDN. Mozilla Corporation. : The Content Division Element. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div. Luettu 12.10.2024.

MDN. Mozilla Corporation. Semantics. Luettavissa: https://developer.mozilla.org/en-US/docs/Glossary/Semantics. Luettu: 12.10.2024.

Riihiaho S. 30.6.2022. Saavutettavat merkinnät. Eficoden blogi. Luettavissa: https://www.eficode.com/fi/blog/saavutettavat-merkinnat. Luettu: 21.9.2024.

Suomen tietotoimisto. 9.9.2021. Aluehallintovirasto antoi Verohallinnolle huomautuksen Palkka.fi-palvelun saavutettavuuspuutteista, myös Espoon ylläpitämän sivuston saavutettavuudessa merkittäviä ongelmia. STT. Luettavissa: https://www.sttinfo.fi/tiedote/69918164/aluehallintovirasto-antoi-verohallinnolle-huomautuksen-palkkafi-palvelun-saavutettavuuspuutteista-myos-espoon-yllapitaman-sivuston-saavutettavuudessa-merkittavia-ongelmia?publisherId=69818103. Luettu: 18.9.2021.

Tamminen, T., Alinikula, P., Hagerlund, T & Lindroth, M. Kuntien saavutettavuusopas, 2017. Luettavissa: https://www.kuntaliitto.fi/julkaisut/saavutettavuusopas/2-mita-on-saavutettavuus. Luettu: 26.8.2024.

University of Maine System. Alternative (Alt) Text – Accessibility Guide. Luettavissa: https://www.maine.edu/content-management/accessibility/images/alt-text/#:~:text=Alt%20text%20best%20practices&text=Many%20screen%20readers%20will%20cut,below%20100%20characters%20when%20possible. Luettu: 12.10.2024.

W3, Background of WCAG 2. Luettavissa: https://www.w3.org/TR/WCAG21/. Luettu: 5.9.2024.

W3. Contrast (Minimum) (Level AA). Luettavissa: https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html. Luettu: 21.9.2024.

WebAIM 2024. Screen Reader User Survey #10 Results. Luettavissa: https://webaim.org/projects/screenreadersurvey10/. Luettu: 26.10.2024.

WebAIM 2024. The WebAIM Million. Luettavissa: https://webaim.org/projects/million/. Luettu: 21.9.2024.