Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Jaakko Mäkilaurila

Saavutettavan verkkopalvelun kehittäminen Sun vuoro -hankkeessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

11.10.2020

Tekijä Otsikko Sivumäärä Aika	Jaakko Mäkilaurila Saavutettavan verkkopalvelun kehittäminen Sun vuoro -hankkeessa 35 sivua 11.10.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tietotekniikka
Ammatillinen pääaine	ohjelmistotekniikka
Ohjaajat	lehtori Ilpo Kuivanen

Saavutettavuuden huomioiminen on etenkin lähivuosina noussut olennaiseksi osaksi laadukkaasti toteutetun verkkosivuston kehitystä. Digitalisaation myötä monet julkiset palvelut ovat siirtyneet verkkoon, mikä on herätellyt lainsäätäjää tarpeeseen varmistaa se, että myös vammaiset ja erilaisista rajoitteista kärsivät ihmiset pystyvät käyttämään julkisia palveluja yhdenvertaisesti.

Opinnäytetyön teoriaosuudessa tutkittiin saavutettavuuden käsitettä ja saavutettavuuslain asettamia vaatimuksia. Lain vaatimukset käytännössä peilaavat WCAG 2.1 -ohjeistuksen saavutettavuuskriteerejä. Lisäksi työssä selvitettiin, miten saavutettavuuden tason voi arvioida ja mitä työkaluja sitä varten on olemassa.

Opinnäytetyön käytännön osuudessa muutin Sun vuoro -nimisen verkkopalvelun saavutettavaksi. Sun vuoro on osa ESR-rahoituksen saanutta hanketta, jonka tarkoitus on yhdistää avoinna olevia palkattomia tehtäviä ja toimintoja ihmisiin, jotka ovat niistä kiinnostuneita. Liityin hankkeeseen kehitystyön loppuvaiheilla, jonka vuoksi tein sovellukseen myös jonkin verran refaktorointia saadakseni kaikki komponentit saavutettaviksi.

Tutkiessani aihetta huomasin, että virallisen WCAG 2.1 -standardin lukeminen on paikoittain haastavaa ja asiat on selitetty korkealla ja abstraktilla tasolla. Tämän työn tarkoituksena onkin myös toimia helpommin ymmärrettävänä johdantona kehittäjille, jotka eivät ole ennestään tutustuneet saavutettavuuteen.

Avainsanat	Saavutettavuus, WCAG 2.1, HTML, CSS, JavaScript

Author Title Number of Pages Date	Jaakko Mäkilaurila Development of Accessible Web Service 35 pages 11 October 2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Development
Instructors	Ilpo Kuivanen, Senior Lecturer

In the past few years accessibility concerns have become an integral part of quality web application development. The trend of transforming services from physical to digital has caused many public services to shift to the web. This has awakened lawmakers to ensure that the disabled and people suffering from various constraints are also able to use public services on an equal basis.

The present study explores the notion of accessibility and the requirements enforced by the current accessibility legislation. In practice the requirements mirror the criteria found in the WCAG 2.1 guideline. Additionally, the study discusses how accessibility is evaluated and what tools exist for this purpose.

In the study, a web service called Sun vuoro was transformed to follow the official accessibility requirements. Sun vuoro is a project funded by ESF. It aims to connect open unpaid positions with interested people. The present study became part of the project towards the end of development work, and therefore also some refactoring was needed in order to make all the components accessible.

The study indicates that the WCAG 2.1 standard is quite hard to read and things are explained on a very high and abstract level. Therefore, the thesis also aims to act as an easier to understand introduction for developers who are not previously familiar with the notion of accessibility.

Keywords	Accessibility, WCAG 2.1, HTML, CSS, JavaScript

Sisällys

Lyhenteet

CLI	Command-line Interface. Tapa järjestää kommunikointi ihmisen ja tietokoneen välillä. Ihminen syöttää tekstimuotoisia komentoja, jotka tietokone tulkitsee.
CSS	Cascading Stylesheets. Verkkosivuilla käytettävä tyyliohjeiden laji. Sillä määritellään WWW-sivun ulkonäkö, mukaan lukien asettelu, värit ja fontti.
ESR	Europan sosiaalirahasto. Yksi Euroopan unionin rakennerahastoista. ESR myöntää vuosittain 10 miljardia euroa tukea erilaisiin hankkeisiin, jotka tähtäävät erityisesti vaikeasti työllistyvien ihmisten työnäkymien parantamiseen.
GDPR	General Data Protection Act. Euroopan unionin yleinen tietosuoja-asetus on EU:n pyrkimys yhtenäistää tietosuojalainsäädäntö kaikkien jäsenmaiden kesken.
HTML	Hypertext Markup Language. Standardoitu internetin kuvauskieli. Sillä voidaan kuvata hyperlinkkejä sisältävää tekstiä ja merkitä sen rakenne.
JWT	JSON Web Token. Eräs standardi käyttöoikeuksien hallintaan internetissä olevien sovellusten välillä. Dataformaattina toimii JSON.
NodeJS	Palvelinpuolella toimiva JavaScript-alusta.
REST	Representational State Transfer. Eräs yleinen HTTP-protokollaa hyödyntävä arkkitehtuurimalli, jolla toteutetaan ohjelmistorajapintoja.
WCAG	Web Content Accessibility Guidelines. Kattava ohjeistus verkkosisällön eri saavutettavuustasojen vaatimuksista ja niiden toteuttamisesta.
WWW	World Wide Web. Internetissä toimiva dokumenttien ja muiden resurssien jakamisjärjestelmä.

1 Johdanto

Metropolia Ammattikorkeakoulussa alkoi syksyllä 2019 hanke, jossa pyrittiin tuottamaan verkkopalvelu Hyvinvoinnin tiloihin vapaaehtoistoimintaan, työkuntoutukseen tai työkokeiluun haluaville henkilöille. Hyvinvoinnin tiloilla tarkoitetaan hyvää tekevien tilojen verkostoa, jossa kehitetään tyhjeneville kiinteistöille uutta hyvinvointiin liittyvää käyttöä. Tiloissa pyritään rakentamaan kulttuuria, hyvinvointia, yhteisöllisyyttä, työhön integrointia ja osallistumisen mahdollisuuksia. Esimerkkinä voisi mainita Helsingin Vanhassakaupungissa sijaitsevan vanhan Annalan huvilan, jossa on mahdollisuus osallistua puutarhanhoitoon, kahvilan toimintaan tai rakennuksen kunnostus- ja siivoustalkoisiin. Huvilassa järjestetään myös konsertteja, teatteriesityksiä ja tapahtumia, joissa voi olla mukana auttamassa. Vuoden 2019 hankkeessa toteutettiin verkkopalvelu Llinder (Lapinlahden Lähteen Tinder). (1; 2.)

Talvella 2020 aloitettiin uusi hanke, jossa jatkettiin alkuperäisessä hankkeessa tuotettua verkkopalvelua. Hankkeen nimeksi tuli nyt *Sun vuoro*, joka korostaa ajatusta siitä, että on käyttäjän vuoro ryhtyä toimeen ja löytää itselleen mielekästä tekemistä. Projektin rahoitus tulee Euroopan sosiaalirahastolta (ESR).

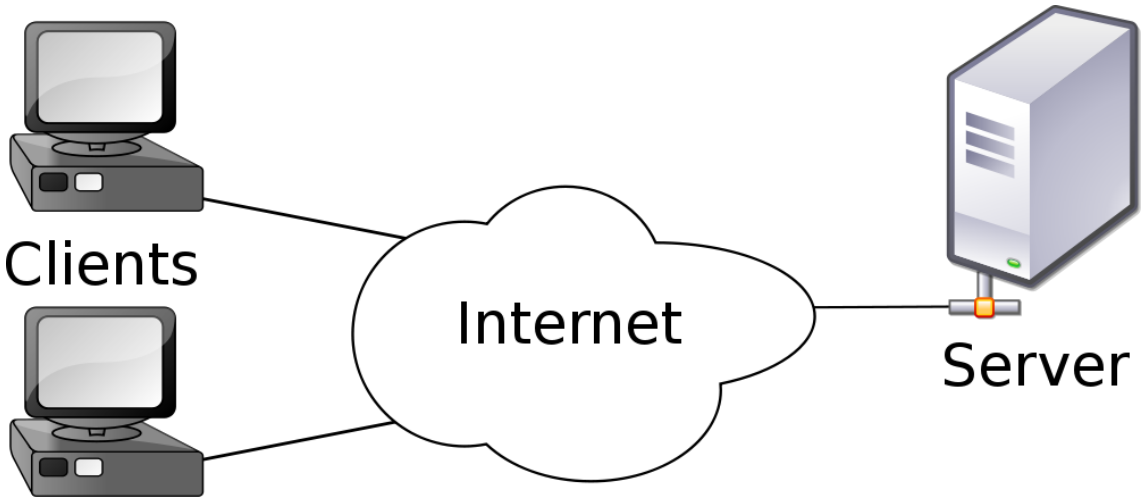
Hankkeen tavoite oli kehittää sovellus, joka mahdollisimman tehokkaasti yhdistää avoinna olevia palkattomia tehtäviä ja toimintoja ihmisiin, jotka ovat niistä kiinnostuneita. Joillakin käyttäjillä saattaa olla erilaisia fyysisiä tai henkisiä rajoitteita. Sovelluksen tulisi ottaa huomioon hakijan omat mieltymykset, vaatimukset ja rajoitteet sekä tarjota paikkoja yksilöllisesti näiden perusteella.

Sun vuoro -palvelulla oli julkisrahoitteisena projektina lakisääteiset saavutettavuusvaatimukset (accessibility requirements). Minut otettiin hankkeen loppuvaiheilla mukaan huolehtimaan näiden vaatimusten toteutumisesta. Insinöörityö perustuu loppukeväällä 2020 päättyneeseen projektiin, johon siis liityin kevään aikana. Insinöörityöraportissani tarkastelen verkkosovelluksen saavutettavuuden teoriapohjaa ja lain määräämiä saavutettavuusvaatimuksia sekä kuvaan näiden vaatimusten toteuttamista Sun vuoro -verkkopalvelussa. Saavutettavuusvaatimuksia ohjaa laki digitaalisten palvelujen tarjoamisesta (306/2019), joka perustuu julkisen sektorin elinten

verkkosivustojen ja mobiilisovellusten saavutettavuudesta annettuun Euroopan parlamentin ja neuvoston direktiiviin (2016/2102). Lain 1§:ssä direktiivistä käytetään lyhennelmämuotoa *saavutettavuusdirektiivi*.

2 Sovelluksen rakenne

Sun vuoro on verkkopohjaisilla teknologioilla toteutettu verkkopalvelu. Verkkopalvelu on WWW:ssä (World Wide Web) näkyvissä oleva sivu, joka tarjoaa palvelua (3). Sovellus on jaettu tyypilliseen tapaan backendiin ja frontendiin, jotka kommunikoivat keskenään verkon yli asiakasohjelma/palvelin-arkkitehtuurimallin (Client/Server Architecture) mukaisesti, kuten kuvasta 1 näkee (4). Kommunikaatioon käytetään REST-rajapintaa (Representational State Transfer). Backend-teknologiana toimii palvelinpuolen JavaScript-ympäristö NodeJS Express-kehyksellä. Frontend eli käyttäjärajapinta on toteutettu Facebookin suositulla React-kirjastolla.



Kuva 1. Asiakasohjelma/palvelin-arkkitehtuurimallia esittävä kaavio.

2.1 Backend

Sovelluksen backend tarjoaa palveluita asiakassovelluksen käytettäväksi. Näitä ovat käyttäjän autentikointi, autorisointii sekä vapaaehtoistyön tehtävien, työtä tarjoavien

organisaatioiden, tehtäväsijaintien ja työn ajankohtien lisääminen, palauttaminen, muokkaaminen ja poistaminen.

Käyttäjän auktorisointiin eli valtuuttamiseen käytetään JWT:tä (JSON Web Token) ja salasanan kryptaamiseen bcrypt-kirjastoa. Tällöin salasana tallentuu turvallisesti kryptattuna tietokantaan. Kun käyttäjä kirjautuu sisään verkkosivulla, palvelin palauttaa JWT:n, jonka asiakasohjelma säilöö. Palvelin tarkastaa jokaisen sille tulevan auktorisointia vaativan pyynnön. Jos asiakasohjelma lähettää pyynnön mukana oikean JWT:n, tapahtuu auktorisointi ja palvelin palauttaa pyydetyn datan. Auktorisointia vaativat kaikki Sun vuoro -palvelun hallintapaneelin toiminnot. Ilman auktorisointia voi ainoastaan nähdä työtehtävät ja täyttää niihin liittyvän hakulomakkeen.

Backendissä on myös mekanismi sähköpostiviestien lähettämiseen. Tehtävästä kiinnostuneet työnhakijat lähettävät asiakasohjelmasta lomakkeen. Lomake lähetetään REST-rajapinnan yli palvelimelle, joka vuorostaan lähettää lomakkeen sähköpostiviestillä kullekin organisaatiolle. Viestin lähettämiseen käytetään Nodemailer-moduulia, joka on eräs suosittu keino lähettää sähköpostiviestejä ohjelmallisesti verkkosovelluksesta (5). Sun vuoro toimii hakijan henkilötietojen välittäjänä eikä säilytä niitä mitenkään. Näin noudatamme GDPR:n (General Data Protection Act) asettamia tietosuojaan liittyviä vaatimuksia.

Lisäksi järjestelmä kerää anonyymiä tilastodataa. Datankeruu on toteutettu REST-rajapinnan kautta. Kerättävä data sisältää muun muassa tehtävämodaalin avaamiset ja tehtävän hakunapin painallukset. Kerättävää dataa käytetään tulevaisuudessa arvioimaan, mitkä tehtävät kiinnostavat hakijoita, ja apuna sivun jatkokehityksessä.

2.2 Frontend

Frontend on toteutettu Create React App -komentorivityökalulla (CLI), joka alustaa uuden React-projektin (esimerkkikoodi 1). Työkalu lataa ja konfiguroi kaikki tarvittavat kirjastot sekä varmistaa niiden yhteensopivuuden keskenään. Mukana tulee suositeltava kansiorakenne, Babel- ja Webpack -työkalujen toimiva konfiguraatio sekä valmiit skriptit muun muassa kehityspalvelimen käynnistämiseen ja projektin kokoamiseen. Lisäksi projektia on helppo laajentaa suosituilla kirjastoilla, kuten TypeScript tai Sass. (6.)

```
npx create-react-app sun-vuoro-client
cd sun-vuoro-client
npm start
```

Esimerkkikoodi 1. Uuden React-projektin luominen Create React App -työkalulla ja sen käynnistäminen.

Tilanhallintaan käytettiin Reactin omaa lokaalia tilaa. Tila sijaitsi muutamassa ylätason komponentissa (containers), ja se välitettiin propseina niiden lapsille. Sovellus oli rakenteeltaan tarpeeksi yksinkertainen, että muita tilanhallintaratkaisuja kuten Reduxia, ei ollut koettu tarpeelliseksi. Itse olisin tässäkin tapauksessa käyttänyt Reduxia, koska se olisi vähentänyt koodin duplikaatiota ja parantanut sovelluksen ylläpidettävyyttä. Reduxin käytössä sovelluksen tila olisi siirretty yhteen tilapuuhun (Single Source of Truth), jossa se olisi ollut helppo hahmottaa kokonaisudessaan (7). Päävastuuni projektissa oli kuitenkin saavutettavuuden aikaansaaminen, eikä näin laajamittaiselle refaktoroinnille yksinkertaisesti ollut aikaa.

Sovellus oli toteutettu Reactin luokkasyntaksilla, joka oli ainoa tapa kirjoittaa tilan omaavia komponentteja ennen versiota 16.8. Helmikuussa 2019 julkaistussa päivityksessä julkaistiin hooksit (8). Hooksit ovat tapa käyttää Reactin ominaisuuksia ilman luokkia. Refaktoroin useimmat komponentit, joihin tein muutoksia käyttämään luokan sijaan funktionaalista komponenttirakennetta ja hookseja tilanhallintaan sekä sivuvaikutuksiin.

Koodin muotoilukonventiot ovat sovelluksen yleisen ylläpidettävyyden ja helppolukuisuuden kannalta erittäin tärkeitä. Projektissa ei ollut kiinnitetty asiaan tarpeeksi paljon huomiota, tuloksena eri kehittäjät kirjoittivat koodia hieman eri tyylillä. Päätin ottaa asiasta kopin, ja lisäsin projektiin ESlint-nimisen staattisen koodianalyysityökalun. ESlint voidaan konfiguroida kiinnittämään kehittäjän huomio tiettyihin JavaScript-koodin muotoiluseikkoihin, kuten sisennykseen, tekstirivien pituuteen, puolipisteen käyttöön, välitettyjen propsien destrukturointiin ja moniin muihin asioihin, joiden tarkastaminen ei vaadi koodin pyörittämistä JavaScript-kääntäjässä. ESlint kehitettiin etenkin mahdollistamaan omien sääntöjen luominen kehitystiimin tarpeisiin. (9.)

3 Saavutettavuuden teoriapohja

3.1 Saavutettavuus verkossa

Suomi on ratifioinut vammaisten henkilöiden oikeuksia koskevan Yhdistyneiden kansakuntien (YK) yleissopimuksen ja sen mukaisesti sitoutunut siihen, että vammaisilla ihmisillä on yhdenvertainen pääsy muiden kanssa tieto- ja viestintätekniikkoihin ja tietojärjestelmiin. (10.)

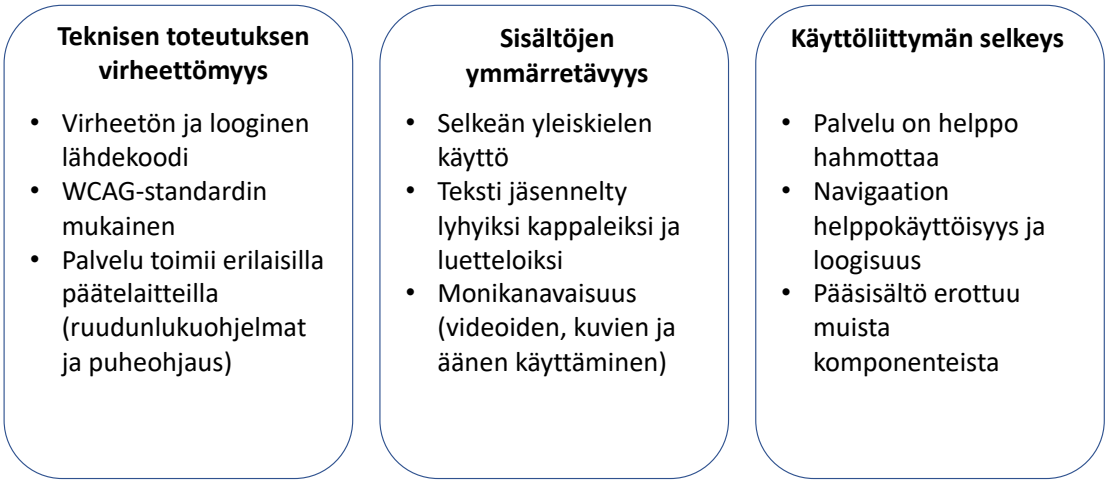
Digitaalisten palvelujen saavutettavuus liittyy myös yleiseen yhdenvertaisuuteen, jota Suomen perustuslain 2 luvun 6 § suojaa. Kaikilla täytyy olla oikeus päästä tietoyhteiskunnan jäseniksi.

3.1.1 Saavutettavuuden määritelmä

Digitaalisten palvelujen tarjoamisesta annetun lain 2 §:n 4-kohdassa määritellään saavutettavuus. Se tarkoittaa sitä, että palvelujen suunnittelun, kehittämisen, ylläpidon ja päivittämisen yhteydessä on noudatettava sellaisia periaatteita ja tekniikoita, joiden avulla käyttäjät, aivan erityisesti vammaiset henkilöt, pääsevät palveluista osallisiksi.

Saavutettavuus on verrattavissa reaalimaailman käsitteeseen *esteettömyys.* Esteettömyyttä käytetään usein puhuttaessa fyysisistä rakennuksista, joiden suunnittelussa on otettu huomioon esimerkiksi helppokulkuisuus ja erilaiset näkörajoitteet. Samalla tavalla kuin rullatuoleilla liikkuvat pääsevät rakennuksiin erilaisten ramppien avulla, saavutettavuus mahdollistaa pääsyn verkon palvelujen ääreen. Saavutettavia digipalveluja ovat verkkosivut, joiden suunnittelussa on otettu huomioon esimerkiksi sisällön selkeys, ymmärrettävyys ja looginen jaottelu. Termi siis tarkoittaa käytettävyyttä, jossa huomioidaan myös erilaisista rajoitteista kärsivät ihmiset. Käytettävyys viittaa siihen, miten vaivattomasti ja miellyttävästi käyttäjä saavuttaa tavoitteensa palvelua käyttäessään. (11; 12, s. 6.)

Kuten kuvassa 2 on nähtävissä, yleisperiaatteet saavutettavan verkkosivuston suunnittelussa ja kehityksessä ovat teknisen toteutuksen virheettömyys, käyttöliittymän selkeys ja sisältöjen ymmärrettävyys (11).



Kuva 2. Saavutettavuuden yleisperiaatteet.

Teknisen toteutuksen laatu on eduksi muutenkin kuin ainoastaan saavutettavuuden näkökulmasta. Se, että sivusto toimii erilaisilla avustavilla teknologioilla, takaa sen, että se toimii yleisesti käytössä olevien selainten ja päätelaitteiden lisäksi myös harvinaisemmilla ja vähemmän tuetuilla laitteilla sekä käyttäjäagenteilla. Laatu tarkoittaa sitä, että verkkopalvelu on toteutettu validilla, semanttisella HTML-koodilla (HyperText Markup Language) ja WCAG-ohjeistusta (Web Content Accessibility Guidelines) on noudatettu. (11.)

Ymmärrettävyys koskee itse informaatiosisältöä. Palvelussa tulisi käyttää helppolukuista selkokieltä, jossa tärkein asia löytyy alusta. Olisi syytä käyttää kuvaavia väliotsikoita ja jäsennellä teksti kappaleiksi. Linkkitekstien pitäisi aina kertoa, mihin ne vievät. Sisältöjä tulisi tarjota eri muodoissa: videoina, kuvina ja äänenä. (11.)

Käyttöliittymän selkeys taas koskee sivuston visuaalista rakennetta. Verkkopalvelun pitäisi olla helppokäyttöinen. Navigaation tulee olla selkeä, eikä liian monitasoinen. Sivulta pitää pystyä löytämään vaivattomasti haluttu alasivu ja sisältö. (11.)

3.1.2 Saavutettavuuden merkitys käytännössä

Arviolta noin miljoona suomalaista eli noin 20 % suomalaisista ei pysty vaivattomasti käyttämään verkkopalveluja, joiden saavutettavuuteen ei ole kiinnitetty huomiota (13). Digitalisaation myötä yhä suurempi osa viranomaisten ja muiden palveluntarjoajien palveluista on siirtynyt tai siirtymässä verkkoon (12, s. 7). Yleishyödyllisten ja sosiaalisten palvelujen, näin ollen myös verkkopalvelujen on oltava kaikkien suomalaisten saatavilla. Saavutettavuus on huomioitava digitaalisten palvelujen suunnittelussa, kehittämisessä ja päivittämisessä.

Esimerkiksi seuraavia asioita ei aina huomioida riittävästi verkkopalvelun suunnittelussa:

- huono näköaisti ja sokeus
- huono kuulo ja kuurous
- fyysiset rajoitteet, kuten motoriset sairaudet tai halvaantuminen
- luki- ja oppimisvaikeudet
- mielenterveysongelmat
- muistihäiriöt
- huono suomen kielen taito
- vaikeudet oppia käyttämään digitaalisia palveluja
- autismi
- väliaikaiset rajoitteet. (12, s. 7; 13.)

Saavutettavuus palvelee todellisuudessa meitä kaikkia, eikä pelkästään erityisryhmiä. Monilla käyttäjillä voi olla hidas internetyhteys, tilapäisesti heikentynyt näkökyky tai he saattavat esimerkiksi käyttää palvelua meluisassa tilassa (12, s. 6). Saavutettavuus on keskeinen osa Suunnittele kaikille -periaatetta (Design For All), jonka tarkoituksena on huomioida erilaiset käyttäjät jo palvelun suunnitteluvaiheessa (11).

Lisäetuna saavutettavuuden huomioimisessa on se, että tutkimusten mukaan saavutettavien verkkosivujen hakukonenäkyvyys on selvästi parempi ja myös ylläpitokustannukset ovat matalampia verrattuna sellaisiin sivustoihin, joiden kohdalla saavutettavuuteen ei ole kiinnitetty huomiota (14). Tämä johtuu siitä, että saavutettavuusvaatimukset edellyttävät sivuston loogista rakennetta, hyvää otsikointia ja elementtien nimeämistä. Hakukoneiden suorittama indeksointi sijoittaa internetsivuja

korkeammalle hakutuloksissa nimenomaan selkeän merkkauskielen sekä hyvän otsikoinnin ja nimeämisen perusteella. Samat periaatteet sivuston suunnittelussa myös helpottavat ylläpidettävyyttä ja siten pienentävät ylläpitokustannuksia. Saavutettava verkkopalvelu on siis myös eduksi liiketoiminnalle.

3.2 Saavutettavuuden normitausta

3.2.1 Lainsäädännölliset puitteet

Digitaalisten palvelujen tarjoamisesta annetun lain määräämät saavutettavuusvaatimukset koskevat kaikkia viranomaisia, julkisoikeudellisia laitoksia ja eräitä yksityisen sektorin organisaatioita, kuten pankkeja. Laki määrittelee myös muiden organisaatioiden tuottaman verkkopalvelun saavutettavuuslainsäädännön piiriin, jos tämä organisaatio saa palvelun kehittämiseen kohdennettua avustusta. Ennen 23.9.2018 julkaistujen verkkosivujen tuli täyttää saavutettavuusvaatimukset 23.9.2019 mennessä. Jälkeen 23.9.2018 julkaistujen verkkosivujen tulee täyttää vaatimukset 23.9.2020 mennessä. (15; 16.)

Sun vuoro -hanke kuuluu saavutettavuuslainsäädännön piiriin. Se täyttää kriteerit kahdella tavalla. Ensinnäkin sovellusta toteuttaa Metropolia Ammattikorkeakoulu, joka lasketaan viranomaiseksi lain tarkoittamassa mielessä. Toisekseen hanke saa rahoituksensa Euroopan unionilta, jolloin se lasketaan verkkopalveluksi, jonka toteutukseen saadaan erityisavustusta. *Sun vuoro* tulee olemaan saavutettavuusvaatimusten piirissä heti julkaisuvaiheessa. (15; 16.)

Saavutettavuusvaatimukset määritellään Euroopan unionin yhdenmukaistettujen standardien mukaisesti (Laki digitaalisten palvelujen tarjoamisesta 7§). Nämä standardit noudattavat Web Content Accessibility Guidelines (WCAG) 2.1-ohjeistuksen AA-tasoa (12, s. 8).

Saavutettavuusvaatimukset koskevat kaikkea lain piirissä olevien organisaatioiden verkkosisältöä, kuten tekstejä, kuvia, kaavioita, videoita, ääniklippejä, upotettuja chat-palveluja, navigointia, kalentereita, lomakkeita, linkkejä, toimisto-ohjelmilla tuotettuja

tiedostoja jne. Organisaatio vastaa myös siitä, että heidän tuottamansa sisältö, joka julkaistaan toisen organisaation palvelussa, täyttää saavutettavuusvaatimukset. (17.)

Saavutettavuuslaki koskee niin ikään verkkosivuja kuin mobiilisovelluksia, jotka ovat tarjolla verkossa. Laki koskee myös organisaatioiden intra- ja ekstranettiä tietyin rajoituksin. Uusien järjestelmien tulee noudattaa saavutettavuusvaatimuksia. Vanhaa intra- ja ekstranettiä ei tarvitse muuttaa saavutettavaksi, ellei sitä uudisteta täysin. Organisaation sosiaalisen median sisällön tulee myös olla saavutettavaa. Käyttäjien itsensä lisäämä sisältö, kuten kommentit, eivät kuitenkaan kuulu lain piiriin. (17.)

Verkkosivulla näytettävät suorat video- tai äänilähetykset muodostavat poikkeuksen saavutettavuusvaatimuksiin. Jos ne säilyvät sivulla yli 14 vuorokautta, ne pitää kuitenkin muuttaa saavutettaviksi tekstittämällä tai kuvailutulkkaamalla. Tilapäinen verkossa oleva opetussisältö ei myöskään kuulu lain soveltamisen piiriin. Kuitenkin pysyvä opetusmateriaali pitää muuttaa saavutettavaksi. Poikkeuksen muodostavat myös verkossa olevat testit, joiden tarkoitus on kokeilla käyttäjän kuullun ymmärtämistä tai visuaalisia taitoja. Tehtäviin tulee kuitenkin liittää tekstimuotoinen kuvaus. Karttojen ei tarvitse täyttää saavutettavuuskriteerejä. Navigointiin käytettyjen karttojen rinnalle tulee kuitenkin lisätä tekstimuotoinen opastus, joka täyttää saavutettavuusvaatimukset. Datan visualisointiin käytetyt kartat eivät kuulu poikkeuksen piiriin, vaan kaikki data tulee myös esittää tekstuaalisessa muodossa. (17.)

3.2.2 WCAG-ohjeistus

WCAG-ohjeistus on World Wide Web -konsortion julkaisema ohjeistus, jonka tavoite on varmistaa minimitaso saavutettavuudelle verkossa. WCAG 1.0 julkaistiin jo vuonna 1999. Versio 2.0 julkaistiin vuonna 2008. Lopulta se päivitettiin versioon 2.1 vuonna 2018. (18.)

Ohjeistus on hierarkkinen, ja se koostuu neljästä *pääperiaatteesta*, jotka on jaettu *yläkriteereihin*. Yläkriteerit on vielä jaettu varsinaisiin toteuttamiskelpoisiin onnistumiskriteereihin eli *kriteereihin*. Ohjeistus ei ole sidottu mihinkään tiettyyn teknologiaan, mikä tekee siitä yleispätevän, joskin hieman abstraktin. WCAG on myös

jaettu kolmeen vaatimustasoon (A-, AA- ja AAA -tasot), joista jokainen laajentaa saavutettavuutta yhä suuremmalle käyttäjäryhmälle. (18.)

Minimitason muodostavat neljä pääperiaatetta ovat:

- havaittavuus
- hallittavuus
- ymmärrettävyys
- toimintavarmuus. (19.)

Havaittavuus

Havaittavuus koskee sivun informaatiota ja käyttöliittymäkomponentteja. Havaitseminen käsitetään yleensä näkemiseksi. Se on kuitenkin todellisuudessa laajempi käsite, joka kattaa myös muun muassa kuulo- ja tuntoaistin. Käyttäjän tulee siis pystyä havaitsemaan sivun sisältöä riippumatta rajoitteistaan, kuten kuulo- tai näkövammasta. (19; 20.)

Osalla käyttäjistä on heikentynyt näköaisti. Tällöin suunnittelussa tulisi huomioida tekstin koko. Fonttikoon pitäisi olla tarpeeksi suuri ja mielellään suurennettavissa. Vähintäänkin erilaisten ulkopuolisien työkalujen pitäisi voida suurentaa sivuston tekstiä. Tekstin ja taustan välisen kontrastin pitäisi olla tarpeeksi suuri. Informaation välittämisessä ei pitäisi käyttää yksinomaan värejä, koska osalla käyttäjistä on värisokeus. (19; 20.)

Jotkut käyttäjät eivät pysty näkemään ollenkaan, vaan ovat riippuvaisia kuulo- tai tuntoaististaan. HTML:n rakennetta hyödyntäen sivusto voidaan muuntaa toiseen esitystapaan. Muita esitystapoja ovat esimerkiksi sivun lukeminen ruudunlukuohjelmalla tai esittäminen pistenäytöllä. Tällöin korostuu oikeanmuotoisen semanttisen HTML:n kirjoittaminen, josta välittyy sivuston rakenne ja suhteet. Kehittäjän pitäisi käyttää oikeita elementtejä oikeissa paikoissa, hyödyntää otsikointihierarkian esittämiseen heading-elementtejä ja kirjoittaa tekstivastineet informaatiota sisältäville kuville. (19; 20.)

Sivustolla oleva äänisisältö pitää esittää vaihtoehtoisella esitystavalla, yleensä tekstinä. Ylipäänsä kaikki informaatiosisältö tulee esittää tekstinä. Toisin kuin muut sisältötyypit, teksti voidaan muuntaa mihin tahansa esitysmuotoon. Se voidaan esittää niin

visuaalisesti, auditiivisesti kuin käsin kosketeltavassa muodossa. Sitä voidaan myös esimerkiksi suurentaa. (19; 20.)

Hallittavuus

Hallittavuus koskee sivun käyttöliittymäkomponentteja. Käyttäjän tulee pystyä havaitsemisen lisäksi myöskin hallitsemaan komponentteja. Tämä tarkoittaa esimerkiksi navigointia sivulla ja linkkien painamista. Perinteisesti navigoinnin ajatellaan tapahtuvan hiirellä tai kosketusnäytöllä. Hiiri tai muut osoitinlaitteet eivät kuitenkaan ole ainoa tapa navigoida sivua. Joillakin käyttäjillä voi olla alentunut motoriikka, jolloin hiiren kaltaisten pistemäisten osoitinlaitteiden käyttäminen ei mitenkään onnistu. Osa ihmisistä taas on tottunut käyttämään tiettyjä näppäimistökomentoja yleisten toimenpiteiden suorittamiseksi verkkosivuilla. Navigoinnin tulee siis onnistua näppäimistöllä ja muilla syötelaitteilla, jotka eivät nojaa jonkin tietyn kohdan osoittamiseen näytöllä. Hyvin toteutettu näppäimistöhallittavuus mahdollistaa sivun navigoinnin jopa imupuhalluskytkimellä. (19; 20.)

Erittäin tärkeä kokonaisuus hallittavuudessa on siis näppäimistön käyttö, jonka huolellinen toteuttaminen myös yleensä mahdollistaa muiden syötelaitteiden hyvän toimivuuden sivustolla.

Tyypillisiä tapoja navigoida sivua näppäimistön avulla ovat esimerkiksi:

- Elementtien kohdistaminen järjestyksessä sarkaimella ja vaihtopainikkeen sekä sarkaimen yhdistelmällä.
- Lomakkeiden lähettäminen palautusnapin painalluksella.
- Sivun vierittäminen ylös ja alas nuolinäppäimillä. (20.)

Jos sivustot eivät reagoi, tai reagoivat odotusten vastaisesti tyypillisiin näppäimistökomentoihin, laskee se huomattavasti sivuston käytettävyyttä. Onneksi natiivit HTML-elementit tukevat näppäimistön käyttöä hyvin ja niissä olisikin sen vuoksi syytä pysytellä. Kuitenkin jos räätälöityjä elementtejä joutuu käyttämään, niissä pitäisi hyödyntää niin kutsuttuja ARIA-attribuutteja. Ne kertovat avustavalle teknologialle, mikä elementti on kyseessä ja mikä sen tila on. Pitää kuitenkin muistaa, että ARIA-attribuutti

ei itsessään tee mitään, se vain tarjoaa informaatiota. Luvattu toiminnallisuus pitää toteuttaa itse JavaScriptilla. (19; 20.)

Navigoitavuuden pitäisi muutenkin olla hyvin toteutettu. Kohdistujärjestyksen pitäisi olla ennakoitavissa, sivuilla pitäisi olla otsikot ja käyttäjälle pitäisi tarjota useita tapoja navigoida eri alasivuille. WCAG 2.1-kriteeri ”Useampi tapa” tarkoittaa yksinkertaisesti sitä, että sisällön ei pitäisi olla liian vaikeasti löydettävissä. Jos yksi tapa löytää sisältö ei jostain syystä toimisi tai se olisi toteutettu epäsaavutettavasti, tulisi kuitenkin olla tarjolla vähintään toinen keino päästä samaan sisältöön käsiksi. Sivukartta ja hakukenttä ovat tapoja toteuttaa vaihtoehtoinen navigointi perinteisen ylävalikon lisäksi. (19; 20.)

Ymmärrettävyys

Ymmärrettävyys tarkoittaa informaation ja käyttöliittymän selkeyttä ja loogisuutta käyttäjän näkökulmasta. Sivustolla vierailee lopultakin hyvin monimuotoisia ihmisiä, joista osalla voi olla kielellisiä tai ymmärrykseen liittyviä haasteita. Tämä yläkriteeri on pitkälti sisällöntuottajien ja palvelumuotoilijoiden vastuulla, koska se asettaa ehtoja itse informaatiosisältöön. Sivustolla olevan sisällön pitäisi olla melko yksinkertaista ja epätavallisia sanoja sekä lyhenteitä tulisi välttää. Olisi hyvä suosia lyhyitä lauseita monimutkaisien rakenteiden sijaan. (19; 20.)

On kuitenkin joitain konkreettisia seikkoja, jotka kehittäjänkin pitäisi huomioida. Käyttöliittymän tulee olla johdonmukaisesti rakennettu. Lomakekenttien tulee sisältää johdonmukaiset nimet, käyttöohjeet ja syötevirheiden tunnistuksen. Sivuston kielen pitäisi olla ohjelmallisesti tunnistettavissa, jotta erilaiset ruudunlukuohjelmat osaisivat tulkata sivuston informaatiota oikein. Tämä koskee itse kielen lisäksi myös esimerkiksi tekstin lukusuuntaa. Lisäksi sivuston sisällön muutoksien tulisi tapahtua ainoastaan käyttäjän pyynnöstä. Esimerkiksi uusia hakutuloksia ei saisi tuoda näkyviin ennen kuin käyttäjä painaa hakunappia, ellei siitä kerrota jotenkin etukäteen. (19; 20.)

Toimintavarmuus

Toimintavarmuus edellyttää, että sisällön tulee olla helposti tulkittavissa eri asiakasohjelmilla, kuten avustavilla teknologioilla. Eli sivun tulee olla koneellisesti ymmärrettävissä. (19.)

Yleisesti tiedostetaan, että eri selaimet saattavat tulkita verkkosivuja hieman eri tavoin. Selainpuolella ongelman aiheuttavat yleensä JavaScript- ja CSS (Cascading Stylesheets) -standardien epäjohdonmukainen toteutus eri selaimissa tai kehittäjän itsensä kirjoittama epävalidi koodi. Avustavien teknologioiden osalta ongelman aiheuttaa useammin heikosti standardeja noudattava HTML, sillä nämä teknologiat ovat erittäin riippuvaisia sen tasosta. Esimerkkejä huonosti toteutetusta merkkauskielen käytöstä ovat tuplatunnisteet ja epävalidit HTML-tagit tai -attribuutit. (20.)

3.2.3 Saavutettavuusseloste

Saavutettavuuslainsäädännön piirissä olevien organisaatioiden on laadittava saavutettavuusseloste verkkosivuistaan ja mobiilisovelluksistaan. Selosteen tulee olla julkaistu saavutettavuudelle asetettuihin siirtymäaikoihin mennessä, saavutettavan sisällön ohella. Selosteen pakollinen sisältö on määritelty EU-komission säädöksessä. (21.)

Saavutettavuusselosteen pakolliset tiedot ovat:

- Arvio sovelluksen saavutettavuuden tilasta. Arviointiin käytetään ABC-luokitusta, jossa A kuvaa täysin saavutettavaa palvelua, B palvelua, joka täyttää kriittiset saavutettavuusvaatimukset ja C palvelua, josta löytyy vielä melko paljon kriittisiä puutteita.
- Miltä osin verkkosivu ei vastaa saavutettavuusvaatimuksia.
- Selosteen päivämäärä (milloin laadittu ja päivitetty) ja minkä tahon arvioon seloste perustuu.
- Miten käyttäjä voi antaa palautetta saavutettavuudesta ja kuka tai mikä taho organisaatiossa vastaa palautteen käsittelystä.
- Miten käyttäjä voi jättää aluehallintovirastoon selvityspyynnön sivuston saavutettavuudesta. (21.)

4 Saavutettavuusvaatimusten toteuttaminen

Laki velvoittaa noudattamaan WCAG 2.1 -ohjeistuksen A- ja AA-tason kriteerejä. Niitä on yhteensä 49 kappaletta (19). On tärkeää huomioida, että nämä kriteerit eivät vielä tee sivustosta täydellisen saavutettavaa. AAA-tason kriteerit edistävät käytettävyyttä ja ymmärrettävyyttä, mutta niidenkään täyttyminen ei välttämättä takaa saavutettavuutta. Kriteereistä osa on tulkinnanvaraisia, joten ne voitaisiin periaatteessa toteuttaa siten, että ne eivät kuitenkaan täyttäisi kyseisen kriteerin taustalla olevia tavoitetta täydellisesti. Tulkinnassa pitäisi pyrkiä siihen, että tavoitteet toteutuisivat mahdollisimman hyvin.

Sun vuoro -verkkosovelluksen kehitystyö oli jo hyvässä vauhdissa, kun projektissa herättiin saavutettavuuden vaatimuksiin. Saavutettavuutta ei siis ollut otettu alusta pitäen lainkaan huomioon. Minut otettiin tässä vaiheessa tiimiin. Tehtäväni oli perehtyä saavutettavuuteen ja pyrkiä toteuttamaan saavutettavuusvaatimukset Sun vuoro -sovelluksessa.

Tässä luvussa käydään läpi saavutettavuuden toteuttaminen Sun vuoro -verkkopalvelussa. Ensiksi esitellään apuna käytetyt työkalut. Seuraavissa luvuissa käydään läpi käytännön toteutus. Toteutuksen esittely on jaoteltu WCAG 2.1 -standardin mukaisten otsikoiden alle. Toteutus siis esitellään samassa järjestyksessä kuin pääperiaatteet, yläkriteerit ja kriteerit on lueteltu virallisessa standardissa. Tämä helpottaa kokonaisuuden hahmottamista ja toimii myös lukijalle pedagogisena työkaluna

saavutettavuusvaatimusten opetteluun. Tässä kuitenkin pääosin sivuutetaan ne yläkriteerit ja kriteerit, jotka eivät vaatineet toimenpiteitä Sun vuoro -verkkopalvelussa. Toimenpiteitä eivät vaatineet ne kriteerit, jotka olivat jo vaatimusten mukaisia tai eivät yksinkertaisesti koskeneet sivun toiminnallisuutta.

4.1 Työkalut

Kun olin perehtynyt saavutettavuuden teoriaan, aloin seuraavaksi tutkia, mitä työkaluja sen toteuttamiseksi ja valvomiseksi oli olemassa. Työkaluja on paljon, joista suuri osa on visuaalisia työkaluja, jotka osoittavat sivulla, mitkä elementit eivät ole saavutettavuusvaatimusten mukaisia. Eräs poikkeus tähän on Axe-projekti, joka toimii koodin tasolla ja ajetaan samalla tavalla kuin automaattiset testit. Päädyin siis saavutettavuustestiajuriin Axe ja visuaaliseen saavutettavuustyökalupalkkiin nimeltään Tota11y. Käytin lisäksi macOS:n sisäänrakennettua ruudunlukuohjelmaa VoiceOver.

4.1.1 Axe

Saavutettavuuden automaattisessa testaamisessa yksi ylitse muiden on kirjasto nimeltään Axe. Axe on siis verkkosivujen saavutettavuuden testaamiseen tarkoitettu ajuri. Sen suurimpiin etuihin kuuluu nopeus, tietoturva ja vaivaton integrointi sovelluksen automaattisiin yksikkötesteihin. (22.)

Axe-projekti on varsin laaja. On olemassa kirjastoja, jotka yhdistävät axe-core-ydintoiminnallisuuden johonkin tunnettuun yksikkötestikehykseen. Sun vuoro -projektissa oli käytössä Jest-kehys testien ajamiseen. Käytin Jest-axe-kirjastoa näiden yhteen liimaamiseen. Saavutettavuutta testaava testitapaus kirjoitettiin kuten esimerkkikoodissa 2. Ensiksi tapahtuu testattavan komponentin alustus, ja expect-lauseessa testataan, että saavutettavuusrikkeitä ei löydy. Testitapaus epäonnistuu, jos rikkeitä löytyy. Saavutettavuustestit ajetaan kätevästi yhtä aikaa tavallisten yksikkötestien kanssa.

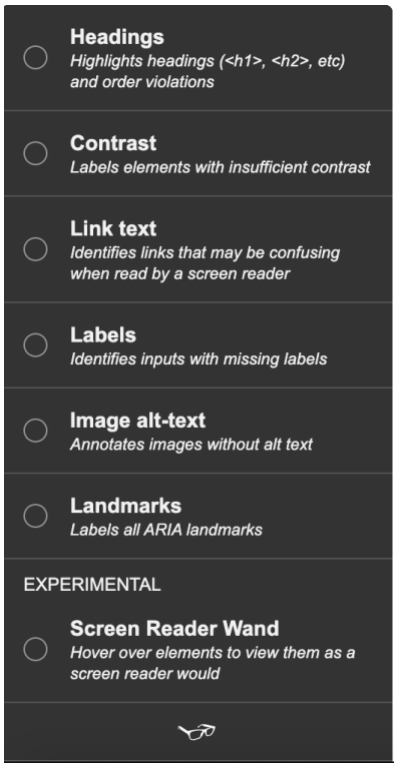
```
it('Main should not have basic accessibility issues', async () => {
  const { container } = render(<Main />);
  const results = await axe(container);
  expect(results).toHaveNoViolations();
});
```

Esimerkkikoodi 2. Axen saavutettavuusrikkeitä testaava testitapaus.

4.1.2 Tota11y

Tota11y auttaa kehittäjää visualisoimaan kehitettävän sivun saavutettavuuspuutteet. Se on käytännössä sivun alareunassa oleva työkalupalkki. Palkissa on erilaisia liitännäisiä (plugin), kuten kuvassa 3 näkee. Ne annotoivat ja korostavat sivulta tiettyjä elementtejä ja saavutettavuuspuutteita. Työkalupalkissa on korostus otsikoinnille, kahden väriyhdistelmän konrastisuhteelle, hämmentäville ja epämääräisille linkkiteksteille, syötekenttien nimilapuille, kuvien alt-teksteille ja sivulla oleville ARIA-rajapyykeille (landmark). (23.)

Tota11yn ajatuksena on helpottaa saavutettavuuden toteuttamista sivulla. Se on suunniteltu siten, että kehittäjällä ei tarvitse edes olla aiempaa kokemusta saavutettavuuden toteuttamisesta. Työkalu antaa saavutettavuusongelmien visualisoinnin lisäksi valistuneita arvauksia siitä, miten ne tulisi korjata. Havaittuaan esimerkiksi liian matalan etu- ja taka-alan kontrastisuhteen, antaa se suoraan ehdotuksen paremmasta väriyhdistelmästä. (23.)



Kuva 3. Tota11y-saavutettavuustyökalupalkki.

4.1.3 VoiceOver

VoiceOver on Mac OS X:ään integroitu ruudunlukuohjelma. Ruudunlukuohjelma on näkövammaisten käyttämä avustava ohjelma. Käytännössä se lukee ääneen näytöllä olevan tekstin ja tarjoaa näppäimistö- ja elekomentoja tietokoneen ohjaamiseen. (24; 25.)

Ruudunlukuohjelma on hyvä väline saavutettavuuden testaamiseen, sillä se on eräs loppukäyttäjien oikeasti hyödyntämä työkalu. Se yhdistettynä muihin välineisiin muodostaa jämäkän perustan saavutettavuuden tilan kokonaisvaltaiseen testaamiseen verkossa.

4.2 Havaittavuus

Saavutettavuuden ensimmäinen pääperiaate havaittavuus tarkoittaa sitä, että käyttäjä pystyy havaitsemaan käyttöliittymäkomponentit jollakin keinolla (19). Havaittavuuden

osalta käsittelen ainoastaan tekstivastineet, mukautettavuuden ja erottuvuuden, koska vain ne vaativat toimenpiteitäni.

4.2.1 Tekstivastineet

Kaikki sivun sisältö tulisi esittää elektronisena tekstinä. Kaikki muut esitysmuodot pitäisi siis muuntaa tekstiksi tai tarjota vaihtoehtoinen tekstimuotoinen esitystapa. (26.)

Tekstivastineet-yläkriteerin alla on yksi kriteeri: *Ei-tekstuaalinen sisältö*. Se edellyttää, että kuvista, videoista ja äänipätkistä tarjotaan myös tekstimuotoinen esitys. Sun vuorossa ei vielä kehitysvaiheessa ollut videoita tai äänipätkiä. Siten tämän kriteerin toteutus suuntautui kuviin. Sovelluksessa oli kahdenlaisia kuvia: kehittäjien lisäämiä kuvia ja käyttäjän (palvelua käyttävän organisaation) lisäämiä kuvia.

Kehittäjien lisäämiä kuvia oli suhteellisen vähän ja ne olivat pääosin koristeellisia. Esimerkiksi *Löydä tehtäväsi* -sivulla olevat tehtävän laadulliset painotukset (pulmanratkaisu, fyysisyys, fysiikka) -komponentti oli kuvitettu. Kuvat sinänsä eivät kuitenkaan tuoneet lisäinformaatiota, joka hyödyttäisi näkörajoitteista käyttäjää, sillä komponentti sisälsi valmiiksi kuvaavan otsikon ja selitystekstin. Tässä tapauksessa päädyin antamaan kuvaelementeille tyhjät alt-attribuutit. Tyhjä alt-attribuutti tarkoittaa sitä, että kyseessä on koristeellinen kuva, jonka yli voidaan hypätä näytönlukijalla. Alt-attribuutin jättäminen kokonaan pois taas aiheuttaisi sen, että jotkin ruudunlukulaitteet lukisivat sen sijaan automaattisesti kuvan nimen. (27.)

Käyttäjäorganisaatiot pystyvät lisäämään kuvan luomaansa tehtävään. Nämä kuvat päätin myös tulkita koristeellisina elementteinä, eikä informaatiota sisältävinä kuvina. Kirjoitin kuitenkin frontendiin jatkokehitystä varten logiikan lukemaan palvelimelta tuleva alt-teksti ja laittamaan se kuvan alt-attribuuttiin. Muutoin alt-tekstiksi tulisi kyseisen tehtävän otsikko. Tämä ominaisuus on helppo aktivoida, kunhan tarpeelliset muutokset tehdään ensiksi palvelinpuolelle.

4.2.2 Mukautettava

Sivuston sisältöä pitäisi voida esittää eri muodoissa, kuten yksinkertaisemmalla asettelulla. Esimerkiksi Mozilla Firefox- ja Applen Safari-selaimissa on Reader View -painike osoiterivillä, jonka painalluksesta sivulta poistetaan lukemiskokemuksen helpottamiseksi kaikki ylimääräinen. Jotta tämä on mahdollista, pitää sisällön informaation, suhteiden ja oikean lukemisjärjestyksen olla koneellisesti selvitettävissä. (19.)

Tämä yläkriteeri oli alun perin jokseenkin kunnossa. Muutamassa kohdassa lähdekoodia löysin kuitenkin komponentteja, jotka eivät olleet oikeassa lukemisjärjestyksessä siitä huolimatta, että ne olivat vierekkäin sivua visuaalisesti tarkasteltaessa. Näissä tapauksissa oli yleensä käytetty CSS:ää komponenttien siirtämiseen toistensa viereen, vaikka komponenttien sijainti HTML-puussa ei ollut vierekkäinen. Siirsin nämä komponentit toistensa viereen lähdekoodissa ja muutin CSS:sää, jotta aikaisempi visuaalinen ilme saatiin pidettyä.

4.2.3 Erottuva

Sisällön pitäisi olla helposti taustasta erottuvaa (19). Sun vuoro ei sisältänyt komponentteja, joissa värien käyttö olisi ollut ainoa keino välittää informaatiota tai erottaa visuaalinen elementti toisesta. Myöskään tekstiä esittäviä kuvia ei ollut. Sivustolla ei ollut sisältöä, joka kohdistettaessa (hiirellä tai näppäimistöllä) toisi näkyviin lisäsisältöä. Näihin asioihin liittyvät hyväksymiskriteerit eivät siis olleet relevantteja tässä projektissa.

Luettelen seuraavaksi erottuvuuteen liittyviä saavutettavuuspuutteita. Palvelu sisälsi komponentteja, joiden taustan ja sisällön värien kontrastit eivät olleet saavutettavuuskriteerien vaatimusten mukaisia. Eräs muu erottuvuuteen kuuluva toteuttamaton hyväksymiskriteeri oli tekstin koon kasvattaminen jopa 200 prosenttiin asti. Sivusto ei myöskään ollut täysin responsiivisuuskriteerin mukainen, mikä siis edellyttää, että sivusto toimisi 320 CSS-pikselin levyisellä ja 256 CSS-pikselin korkuisella näytöllä ilman toiminnallisuuden menettämistä. Myös tekstin välityksessä oli hieman korjailtavaa.

Komponenttien taustojen ja sisältöjen värikontrastin havaitsemiseksi ja korjaamiseksi käytin työkalua Tota11y. Se osoitti visuaalisesti kunkin kriteeriä rikkovan komponentin kontrastisuhteen ja antoi parannusehdotuksen. Yhtenäistin samalla sivun värejä käyttäen aina samoja värikoodeja samanlaisiin elementteihin ja hyödyntämällä uudelleenkäytettäviä CSS-luokkia. Muutin myös sivuston värimaailman räikeän oranssista rahallisempiin lilan, harmaan ja valkoisen sävyihin. Rauhallisemmat sävyt ovat tärkeitä etenkin autismikirjon ihmisille.

Lisäsin sivustolle kolme tekstin kokovaihtoehtoa: 100 prosenttia, 150 prosenttia ja 200 prosenttia eli kaksinkertaisen koon. Vaihtoehdot löytyvät sivuston alapalkista ja valittu vaihtoehto kirjoitetaan verkkoselaimen local storageen, jotta käyttäjän ei tarvitsisi joka käyntikerralla valita haluamaansa tekstinkokoa uudelleen. Tekstin koon vaihtaminen tapahtuu muuttamalla sivuston HTML-juurielementin fonttikokoa, joka periytyy kaikille muille elementeille.

Responsiivisuuden havaitsemiseksi käytin Chromen sisäänrakennettua device toobaria, jossa ikkunan kokoa voi säätää haluamaansa leveyteen ja korkeuteen. Yleinen syy responsiivisuusongelmiin ovat kovakoodatut leveys- tai korkeusarvot CSS-tyyleissä. Toinen yleinen syy on sisällön asetteleminen leveyssuunnassa. Usein tietokonenäytöt ovat suurempia leveydeltään kuin korkeudeltaan. Mobiilinäytöt taas ovat tyypillisesti suurempia korkeudeltaan. Responsiivisessa suunnittelussa pitää siis huomioida päätelaitteiden erot. Kovakoodattuja arvoja tulisi välttää mahdollisuuksien mukaan ja suosia pikseleiden sijaan REM-yksiköitä, jotka mukautuvat HTML-puun juurielementin fonttikokoon. Tällöin ulkoasu säilyy hyvänä myös fonttikokoa muutettaessa. Mobiilinäytöille voi luoda oman tyylin CSS-kielen mediakyselyillä. Mobiilissa sisältö kannattaa asetella korkeussuunnassa.

Yhtenäistin sivulla olevaa tekstin välitystä määrittelemällä tekstin riviväliksi 1,5 kertaa kirjasinkoko ja kappaleen jälkeisen tyhjän tilan kooksi kaksi kertaa fonttikoko.

4.3 Hallittavuus

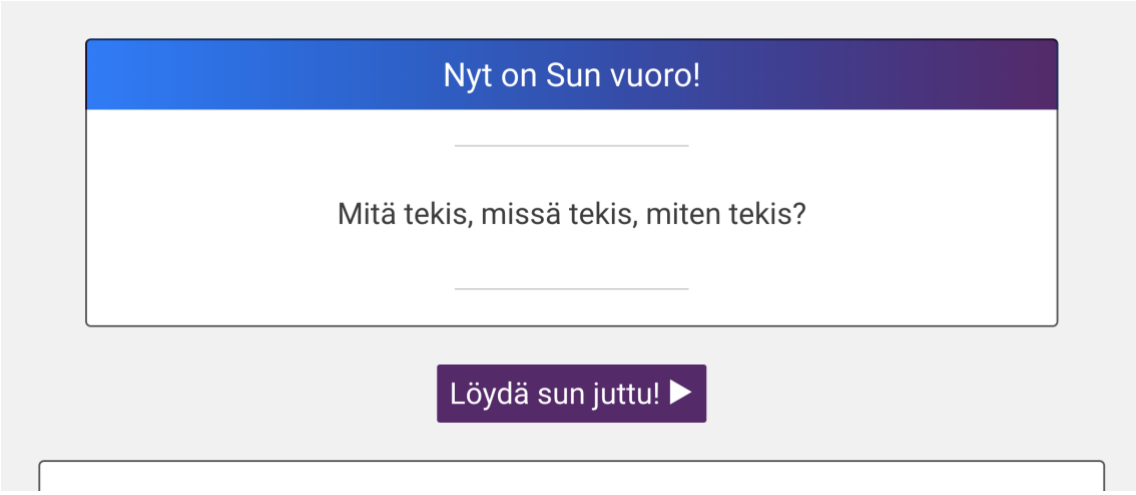
Hallittavuus merkitsee sitä, että käyttöliittymää voi kontrolloida eri tavoin. Hiiren ja muiden osoitinlaitteiden lisäksi sivuston pitäisi olla täysin läpikäytävissä esimerkiksi

näppäimistöllä. (19.) Käyn seuraavaksi läpi näppäimistökäytettävyyden ja navigoitavuuden yläkriteerit.

4.3.1 Käytettävissä näppäimistöltä

Sun vuorossa oli useitakin kohtia, joissa näppäimillä tapahtuvaan sivulla liikkumiseen ei ollut kiinnitetty tarpeeksi huomiota. Suoranaisia näppäimistöansoja ei kuitenkaan ollut. Yleisesti ottaen standardinmukaisen HTML:n kirjoittaminen takaa myös melkein automaattisesti näppäimistökäytettävyyden. HTML-elementit tukevat natiivisti yleisesti käytettäviä näppäimistökomentoja, kuten esimerkiksi eteen- ja taaksepäin liikkumista tabulaattorilla ja tabulaattorin sekä vaihtonäppäimen yhdistelmällä.

On olemassa tilanteita, jotka ovat sen verran monimutkaisia, että selain ei osaa tehdä kohdistusta elementtiin halutulla tavalla. Tämän kaltaisia tilanteita ovat esimerkiksi ponnahdusikkunat ja modaalit. Toinen verkkosivun näppäimistökäytettävyyden yleinen rikkoja on epästandardinmukaisesti, virheellisesti tai epäsemanttisesti toteutettu HTML. Esimerkiksi div-elementti, joka on JavaScriptilla muutettu tosiasiallisesti napiksi ei saa automaattisesti kohdistusta. Näin oli käynyt Sun vuoron etusivulla (kuva 4), jossa vierailijoita tehtävän hakuun kehottava nappi ei todellisuudessa ollut nappi. Sama ongelma toistui muuallakin sivulla. Näissä tapauksissa pitäisikin aina suosia HTML:n valmista button-elementtiä. Haitarivalikot oli myös toteutettu div-elementeillä, jolloin niihin ei pystynyt kohdistamaan näppäimistöllä. Sama lääke tehoaa tähänkin – natiivit HTML-elementit.



Kuva 4. Löydä sun juttu! -nappi oli alun perin div-elementti eikä HTML-nappi.

Ponnahdusikkunat eivät ole paras mahdollinen tapa toteuttaa sivulle dynaamista sisältöä. Kun tulin projektiin, niitä oli kuitenkin käytetty sivulla, eikä aika riittänyt niiden korvaamiseen suositeltavammalla toiminnallisuudella. Kyseessä ei kuitenkaan ollut perinteisellä JavaScriptin alert-funktiolla toteutetut ponnahdusikkunat vaan modaalit, jotka oli muutettu näyttämään ponnahdusikkunoilta. Ne eivät siis ponnahtaneet uuteen ikkunaan vaan pysyivät selainikkunan sisällä.

Näissä ponnahdusikkunoissa oli kuittausnappula, joka ei saanut automaattisesti näppäimistökohdistusta. Sen sijaan kohdistus jäi ikkunan alla olevalle sivulle. Korjasin ongelman siirtämällä kohdistuksen kuittausnappulaan, kun ponnahdusikkunan avaava tapahtuma laukesi. Kun esimerkiksi organisaation rekisteröintisivulta lähetetään lomake, laukeaa tapahtuma. Tapahtumaan on liitetty takaisinkutsufunktio. Tämä funktio muuttaa erään tilamuuttujan arvoa, joka vuorostaan siirtää kohdistuksen ponnahdusikkunan kuittausnappulaan seuraavan kerran, kun käyttöliittymä päivittyy. Siirsin vastaavalla tavalla myös kohdistuksen takaisin alla olevalle sivulle, kun kuittauspainiketta oli painettu ja ponnahdusikkuna oli suljettu.

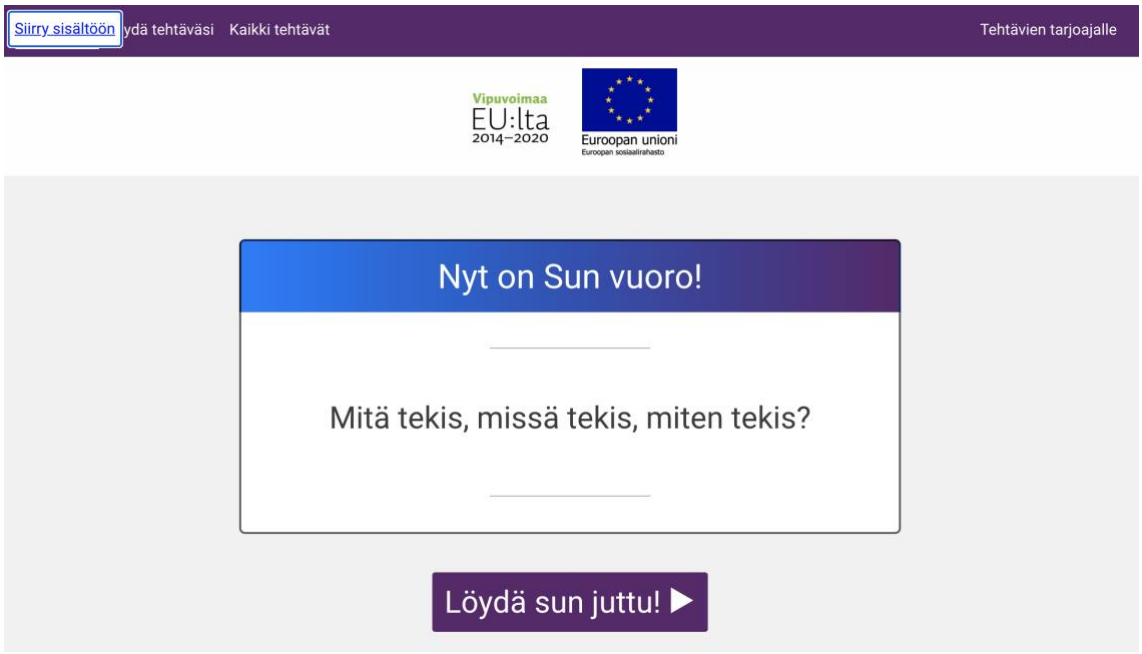
Sun vuoro -palvelusta löytyi myös modaaleja, joissa oli muun muassa lomakekenttiä. Jotta nämä modaalit saatiin saavutettavammiksi, siirsin kohdistuksen suositeltavalla tavalla aina ensimmäiseen syötekenttään ja niissä tapauksissa, joissa syötekenttiä ei ollut, siirsin kohdistuksen ensimmäiseen tekstikappaleeseen (28). Modaalit oli toteutettu

ulkoisella kirjastolla, jossa oli onneksi kiinnitetty huomiota saavutettavuuteen. Kirjasto hoiti kohdistuksen palauttamisen takaisin alla olevalle sivulle.

4.3.2 Navigoitava

Navigoitavuuden alle kuuluvien kriteerien tarkoitus on tehdä sivun selaamisesta ja sisällön etsimisestä mahdollisimman helppoa ja selkeää. Tein Sun vuoro -verkkopalveluun HTML-standardin mukaiset header-, main- ja footer -lohkot. Sen jälkeen toteutin mekanismin, jonka avulla jokaisella sivulla toistuvan otsikkolohkon pystyi ylittämään näppäimistöllä tai muilla sivua lineaarisesti lukevilla työkaluilla. Navigoitavuuden helpottamiseksi kaiken sisällön pitäisi olla otsikoitua. Lisäsin otsikot niihin kohtiin, joista se puuttui tai oli toteutettu epästandardinmukaisesti. Myös linkkien pitäisi selkeästi kertoa, mihin ne on tarkoitettu ja mihin ne vievät. Tämä oli pääasiassa toteutettu oikein Sun vuorossa. Neljäs navigoitavuuteen liittyvä seikka on kohdistusjärjestyksen loogisuus. Kun näppäimistöä käytetään navigointiin, tulee kohdistusjärjestyksen säilyttää tarkoitettu merkitys ja toiminnallisuus. Ei pidä siis olla tilanteita, joissa kohdistus hyppää jonkin tekstikappaleen otsikosta johonkin ihan muualle. Kohdistuksen tulisi siis siirtyä otsikosta tekstikappaleeseen. Lisäksi kohdistuksen tulisi olla näkyvä. Tähän kriteeriin liittyen oli muutama puute. Verkkopalvelun jokaiselle sivulle pitäisi olla enemmän kuin yksi tapa navigoida. Alun perin vain Löydä tehtäväsi -sivulle oli olemassa useampi kuin yksi reitti (etusivulla olevan painikkeen ja navigaatiopalkin kautta).

Otsikko- eli header-lohkon ohittamiseksi lisäsin sivulle linkin vasempaan yläreunaan, joka tulee esiin ainoastaan kohdistettaessa (kuva 5). Koska linkki on ensimmäinen elementti sivulla, saa se kohdistuksen ensimmäiseksi ja tulee näkyviin. Hiirtä käyttävät vierailijat, jotka eivät linkkiä tarvitse, eivät edes näe sitä. Kun linkki aktivoidaan, siirtyy kohdistus suoraan sivun pääsisältöön, jolloin käyttäjä säästyy vaivalta kulkea jokaisen navigaatiolinkin yli.



Kuva 5. Siirry sisältöön -linkki otsikkolohkon ylittämiseksi.

Sun vuorossa oli muutamia kohdistusjärjestykseen ja kohdistuksen näkyvyyteen liittyviä ongelmia. Löydä tehtäväsi -sivulla on kolme liukusäädintä, jotka säätävät arvoja tehtävää määritteleville taidoille. Ne ovat *fyysisyys*, *sosiaalisuus* ja *pulmanratkaisu*. Hakija voi liikuttaa säätimet haluamaansa asentoon. Tämän toteuttamiseen on käytetty kolmannen osapuolen kirjastoa. Kirjasto ei jostain syystä tukenut kohdistuksen näkyvyyttä. Eli kun säätimeen siirryttiin näppäimistöllä, sihen ei ilmestynyt minkäänlaista visuaalista indikaatiota kohdistuksesta. Korjasin tämän ylikirjoittamalla suoraan säätimen CSS-luokkia, jotka selvitin Chromen inspector-työkalulla.

Kriteeri, jonka mukaan jokaiselle sivulle pitäisi olla enemmän kuin yksi tapa navigoida, ei alun perin toteutunut. Sivulla olevassa navigaatiopalkissa oli reitit kaikkiin sivuston alasivuihin, eli etusivulle, Löydä tehtäväsi -sivulle, Kaikki tehtävät -sivulle ja hallintasivuille. Etusivulla sijaitsi lisäksi painike, joka vei suoraan Löydä tehtäväsi -sivulle. Kaikki tehtävät -sivulle vievää linkkiä taas ei ollut muualla kuin navigaatiopalkissa. Myöskään hallintasivuille eli sivuille, jotka näkyvät vain Sun vuoroa käyttäville organisaatioille, ei ollut useampaa reittiä. Lisäsin ensiksi Löydä tehtäväsi -sivulle linkin, joka johtaa Kaikki tehtävät -sivulle. Sen jälkeen lisäsin kirjautumissivulle pienen hallintavalikon, joka listasi Organisaation hallinta-, Tehtävien hallinta- ja Tilastot -sivut. Näille sivuille pääsee nyt sekä navigaatiopalkista että kirjautumissivulta.

4.4 Ymmärrettävyys

Sivun informaatiosisällön ja käyttöliittymän pitäisi olla käyttäjän ymmärrettävissä. Ei siis riitä pelkästään, että kieliasu on yksinkertainen ja selkeä. Käyttäjän pitää myös ymmärtää, miten sivun eri osiin pääsee käsiksi. (29.) Ymmärrettävyyden osalta käsittelen ainoastaan luettavuuden, ennakoitavuuden ja syötteen avustamisen, koska vain ne vaativat toimenpiteitäni.

4.4.1 Luettava

Tekstisisällön luettavuus koskee myös erilaisia ruudunlukulaitteita tai muita tekstiä puheeksi muuttavia ohjelmistoja. Totesin että palvelussa oli luettavuuteen liittyen kaksi toteutumatonta kriteeriä, jotka edellyttivät toimenpiteitäni. Sun vuoro -sivustolla ei ollut erikseen määritelty sivun kieltä. Tällöin oletusarvoinen kieli on englanti. Tämä johti siihen, että sivun suomenkielinen sisältö luettiin ruudunlukuohjelmalla ääneen englanninkielisellä aksentilla. Puute oli onneksi helppo korjata. Lisäsin sivuston HTML-juurielementtiin attribuutin *lang*, jolle annoin arvoksi suomenkielisen kielikoodin *fi* (esimerkkikoodi 3). Toinen kriteeri eli sivun osien kielen ohjelmallinen selvitettävyys ei ollut relevantti, koska sivulla ei esiintynyt lainkaan muuta kuin suomenkielistä sisältöä.

```
<!DOCTYPE html>
<html lang=”fi”>
      <head>…</head>
      <body>…</body>
</html>
```

Esimerkkikoodi 3. Kielikoodi *fi* kirjoitetaan HTML-elementin sisään.

4.4.2 Ennakoitava

Ennakoitavuus on olennaista erityisesti tietyille käyttäjäryhmille, joilla on vaikeuksia muodostaa yleiskuvaa sivustosta. Ruudunlukulaitteet lukevat sivua lineaarisesti ja ihmiset, joilla on kognitiivisia vaikeuksia saattavat hämmentyä sivuston elementtien ja sisällön muutoksista. Ennakoitavan sivuston pitää siis välttää kontekstin muuttamista elementtiin kohdistettaessa tai syötteen antamisen perusteella (kuten tekstikenttään kirjoittaminen tai sliderin arvon muuttaminen). Sivun sisällön ja komponenttien pitäisi muuttua ainoastaan tietyin ehdoin, esimerkiksi kun käyttäjä painaa siihen tarkoitettua nappia tai asiasta on muuten ilmoitettu. (30.)

Sun vuorossa ei ollut kohdistukseen liittyviä kontekstinmuutoksia. *Löydä tehtäväsi* -sivulla oli kuitenkin eräs syötteen arvon muutokseen reagoiva toiminnallisuus, joka olisi pitänyt korjata. Käyttäjän tullessa sivulle, tarjotaan hänelle erilaisia tehtävien suodattimia. Ne ovat kolme tehtävän laadullisen painotuksen slideria, tehtävän sijainti ja tehtävän ajankohdat. Näitä arvoja muuttamalla ei vielä ensimmäisellä kerralla tapahdu mitään. Suodatetut hakutulokset paljastuvat vasta, kun käyttäjä painaa hakunappia. Jos suodattimien arvoja muuttaa tämän jälkeen, muuttuvat hakutulokset kuitenkin välittömästi. Tämä rikkoo edellä mainittua hyväksymiskriteeriä. Kyseessä oli ainoa kriteeri, jota en harmikseni ehtinyt korjata ennen kuin projekti päättyi. Sen sijaan kirjoitin tästä puutteesta sivuston saavutettavuusselosteeseen, jossa on saavutettavuuspuutteille tarkoitettu osio. Tämä huomioidaan sivun jatkokehityksessä. Muut tämän yläkriteerin alla olevat kriteerit (johdonmukainen navigointi ja johdonmukainen merkitseminen) olivat hyvässä kunnossa.

4.4.3 Syötteen avustaminen

Kaikki tekevät virheitä. Erityisryhmillä ongelma kuitenkin korostuu, ja virheiden havaitseminen voi olla hankalampaa. Ruudunluku- tai suurennuslasiohjelman käyttö

rajaa näkökenttää, ja värisokeus saattaa vaikeuttaa virheviestien havaitsemista. Kognitiiviset ongelmat saattavat lisäksi vaikeuttaa jonkin syötekentän merkityksen ymmärtämistä. Sivuston suunnittelussa ja kehityksessä pitää siis varmistaa, että virheviestejä ei voi olla huomaamatta, ne tarjoavat korjausehdotuksia ja että syötekentät ovat selkeästi nimettyjä. Syötteen tarkastaminen on erityisen tärkeää, jos käyttäjän syöttämää dataa koskee oikeudellisia tai taloudellisia seuraamuksia. (31.)

Sivustolla on organisaation rekisteröintilomake, joka sisältää automaattisesti tarkastettaviksi soveltuvia kenttiä. Näitä ovat sähköpostiosoite, puhelinnumero, verkkosivut ja postinumero. Tähän olikin toteutettu validointi, joka myös tarjosi korjausehdotuksia. Tarkastusta ei ollut kuitenkaan tehty saavutettavaksi. Validointivirheet tulivat näkyviin kunkin syötekentän alapuolelle punaisella tekstillä. Tämä yksinään ei kuitenkaan vielä ole riittävää. Lisäsin lomakkeen yläosaan koostetekstin jokaisesta validointivirheestä. Kooste on nähtävissä kuvassa 6. Kun virheitä sisältävä lomake yritetään lähettää, siirtyy kohdistus automaattisesti yläosan virheluetteloon. Tällöin esimerkiksi ruudunlukuohjelma lukee virheet käyttäjälle välittömästi lähetysyrityksen jälkeen. Virheet ovat myös tarpeeksi suurella fonttikoolla, jotta käyttäjä pystyy erottamaan ne helposti muusta sisällöstä.

Kuva 6. Organisaation rekisteröintilomake ja validointivirheet.

Sivustolla oli myös esimerkiksi tehtävien lisäämiseen tarkoitettu lomake. Se sisältää kenttiä, joita ei substanssinsa vuoksi pysty helposti validoimaan. Lisääjä pystyy kuitenkin helposti muokkaamaan ja poistamaan lisäämiään tehtäviä. Syötevirheet pystytään siis helposti korjaamaan. Lopuksi lisäsin kaikkiin syötekenttiin selkeät nimilaput (kenttää kuvaava teksti).

4.5 Toimintavarmuus

Toimintavarmuuden varmistamisessa kiinnitetään huomiota siihen, että asiakasohjelmat tulkitsevat sivua oikein. HTML:n pitää olla virheetöntä, käyttöliittymäkomponenttien pitää olla nimettyjä ja sisältää niille kuuluvat rooliattribuutit. Lisäksi tilasta kertovien viestien tulisi välittyä ilman, että käyttäjän tarvitsee siirtää kohdistusta johonkin toisen elementtiin. Esimerkiksi haitarivalikon auki/suljettu -tilan pitäisi käydä ilmi jo valikon otsikkoelementistä. (32.)

Sovelluksessa oli melko paljon pientä korjattavaa toimintavarmuuden ja nimenomaan yhteensopivuuden aikaansaamiseksi. Tämän kaltaisia pieniä virheitä on vaikea löytää manuaalisesti. Onneksi käytössäni oli saavutettavuusautomaatiotyökalu Axe. Sen avulla löysin kaikki tuplatunnisteet, väärin kirjoitetut sekä puuttuvat attribuutit ja virheellisesti toistensa sisään laitetut elementit. Löysin eräästä kolmannen osapuolen kirjastosta (React Multiselect Dropdown) useamman kerran toistuvan tunnisteen. Kirjaston jokaisella pudotusvalikolla oli sama ID-attribuutti. Jos pudotusvalikkoja oli useampi samalla sivulla, oli myös identtisiä tunnisteita useita. Ilmoitin puutteesta kirjaston tekijälle. Tekijä vastasi Githubiin jättämääni kommenttiin melko vikkelään ja korjasi ongelman.

Sivulta löytyi joitakin puutteita, joiden korjaamiseksi tarvitsi vain ottaa käyttöön semanttisessa mielessä oikeat HTML-elementit. HTML-standardin mukaiset elementit täyttävät kaikki saavutettavuuskriteerit myös toimintavarmuuden osalta. Tämän vuoksi niitä pitäisi suosia, kuten olen aiemmin maininnut. Rooli-attribuutteja ei tällöin tarvitse eksplisiittisesti kirjoittaa koodiin. Nimi- ja arvo -attribuutit pitää kuitenkin löytyä lähes kaikilta elementeiltä. Syötekentillä pitää olla joko siihen liitetty selite-elementti (label) tai otsikkoattribuutti (title). Vaikka kumpikin kelpaa toimintavarmuuden aikaansaamiseksi, tulisi kentän nimen olla myös näkyvissä käyttäjälle. Näin ollen kannattaa käyttää selitettä, kuten esimerkkikoodissa 4. Selitteen ja syötekentän yhdistää toisiinsa tunnisteattribuutti (id).

```
<label htmlFor="username-field">Käyttäjätunnus</label>
<input
  id="username-field"
  type="text"
  name="username"
/>
```

Esimerkkikoodi 4. Käyttäjätunnus-kenttä ja sen selite.

Lisäsin tilasta kertovia viestejä niihin kohtiin, joista semmoinen puuttui. Eräs puute sivulla olivat haitarivalikot, jotka eivät kerro tilaansa esimerkiksi ruudunlukulaitetta käyttävälle vierailijalle. Lisäsin haitarivalikkoihin attribuutin *aria-expanded*. Erilaiset asiakasohjelmat tunnistavat sen ja kertovat käyttäjälle, onko valikko auki- vai suljettu -tilassa.

5 Yhteenveto

Insinöörityön tavoitteena oli selvittää verkkosovelluksen lainsäädännölliset saavutettavuusvaatimukset ja toteuttaa ne Sun vuoro -verkkopalvelussa. Työn tuloksena syntyi erästä poikkeusta lukuun ottamatta WCAG 2.1 -ohjeistuksen A- ja AA-tasot täyttävä verkkosovellus ja saavutettavuusseloste. Saavutettavuuslaki velvoittaa julkisrahoitteiset verkkopalvelut täyttämään ohjeistuksen A- ja AA-tasot, tai jos jokin osa ei vielä täyty, siitä tulee olla maininta sovelluksen saavutettavuusselosteessa. Näin ollen Sun vuoro toteuttaa nyt lain vaatimukset. Samalla tein sovelluksen koodiin paljon laadullisia parannuksia. Otin käyttöön ESlint-nimisen työkalun koodin laadun tarkastelua varten. Muutin myös Reactin luokkasyntaksilla toteutettuja komponentteja nykyisin suositeltaviksi funktionaalisiksi komponenteiksi.

Saavutettavuuden toteuttamiseen lisähaastetta toi myöhäinen liittymiseni projektiin. Tiimin muilla jäsenillä ei ollut tietämystä tai kokemusta saavutettavuudesta, joten monet komponentit oli toteutettu ilman, että saavutettavuutta oli mitenkään otettu huomioon. Tämä vaati joissakin tapauksissa suuriakin muutoksia komponenttien arkkitehtuurissa ja tavassa, jolla tilaa säilytetään sekä välitetään alaspäin lapsikomponenteille. Asia kuvastaa ehkäpä suurempaa ongelmaa tämän hetkisessä verkkokehityksessä. Saavutettavuudesta ei ole puhuttu tarpeeksi, ja monet kehittäjät eivät ole välttämättä edes tietoisia koko käsitteestä. Saavutettavuuslain astuttua voimaan saavutettavuus ei kuitenkaan enää ole vain ”mukava olla” -ominaisuus kaikki käyttäjät huomioon ottavassa verkkopalvelussa, vaan suuri määrä lain piiriin kuuluvia sivustoja on jouduttu nopeasti päivittämään sen mukaiseksi. Koska toimenpiteet on jouduttu tekemään nopealla aikataululla, eikä niitä useinkaan ole otettu sivuston kehitysvaiheessa huomioon, on olemassa riski, että osa nimellisesti saavutettavista sivustoista omaa kaikesta huolimatta saavutettavuuspuutteita.

Saavutettavuus ja sen testaaminen saattaa tuntua aluksi hieman vaikeasti ymmärrettävältä kokonaisuudelta. Siksi tämän insinöörityöraportin tavoitteena oli myös tiivistää saavutettavuuslainsäädännön ja WCAG 2.1 -ohjeistuksen keskeiset vaatimukset lukijalle helpommin sisäistettävään muotoon.

Sun vuoro -verkkopalvelu on siirtynyt ylläpitovaiheeseen, jossa siihen tehdään yhä pienkehitystä toisten kehittäjien toimesta. Myös saavutettavuuden vahvistaminen ja siihen liittyvään asiakaspalautteeseen reagoiminen on jatkuvan kehityksen kohteena.

Lähteet

1 Mitä teemme. Verkkoaineisto. <https://hyvinvoinnintilat.fi/mitateemme>. Luettu 21.5.2020.

2 Annalan huvila. Verkkoaineisto. <https://hyvinvoinnintilat.fi/jasenet/annalan-huvila>. Luettu 21.5.2020.

3 JUHTA – Julkisen hallinnon tietohallinnon neuvottelukunta. 2013. Verkkoaineisto. JHS 183 Julkisen hallinnon palvelujen tietomalli ja ryhmittely verkkopalveluissa. <http://www.jhs-suositukset.fi/web/guest/jhs/recommendations/183>. 10.9.2015. Luettu 4.7.2020.

4 Client/Server Architecture. 2018. Verkkoaineisto. <https://www.techopedia.com/definition/438/clientserver-architecture>. 5.2.2018. Luettu 15.3.2020.

5 Nodemailer. Verkkoaineisto. <https://nodemailer.com/about>. Luettu 3.7.2020.

6 Chris on Code. Verkkoaineisto. Using Create React App to Make React Applications. <https://scotch.io/starters/react/using-create-react-app-to-make-react-applications>. Luettu 18.7.2020.

7 Three Principles. Verkkoaineisto. <https://redux.js.org/introduction/three-principles>. Luettu 24.7.2020.

8 16.8.0 (February 6, 2019). Verkkoaineisto. <https://github.com/facebook/react/blob/master/CHANGELOG.md#1680-february-6-2019>. 6.2.2019. Luettu 19.7.2020.

9 About. Verkkoaineisto. <https://eslint.org/docs/about>. Luettu 23.7.2020.

10 Yleissopimus vammaisten henkilöiden oikeuksista. 27/2016. Verkkoaineisto. Finlex. <https://www.finlex.fi/fi/sopimukset/sopsteksti/2016/20160027/20160027_2>. Luettu 15.3.2020.

11 Tietoa saavutettavuudesta. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/tietoa-saavutettavuudesta>. Luettu 16.3.2020.

12 Ilona Nurminen, Anu Vuokko, Janne Lahti. 2018. Saavutettavuusopas v.01. Verkkoaineisto. <https://www.preeriapingviini.com/wp-

content/uploads/2018/07/Preeriapingviini_Oy_Saavutettavuusopas_versio_01.pdf>. Luettu 16.3.2020.

13 Kenelle saavutettavuus on tärkeää. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/tietoa-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa>. Luettu 17.3.2020.

14 Accessibility. Verkkoaineisto. <https://www.w3.org/standards/webdesign/accessibility#case>. Luettu 18.3.2020.

15 Ketä laki velvoittaa. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/lait-ja-standardit/keita-laki-velvoittaa>. Luettu 15.3.2020.

16 Siirtymäajat. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/lait-ja-standardit/siirtymaajat/#julkiset>. Luettu 16.3.2020.

17 Mitä palveluja ja sisältöjä laki koskee? Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/lait-ja-standardit/mita-palveluja-ja-sisaltoja-laki-koskee>. Luettu 18.3.2020.

18 Tietoa WCAG-ohjeistuksesta. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/lait-ja-standardit/tietoa-wcag-kriteereista>. Luettu 19.3.2020.

19 WCAG 2.1: lain vaatimukset. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/lait-ja-standardit/wcag-2-1>. Luettu 19.3.2020.

20 Otso Lahti. 2019. Verkkoaineisto. Saavutettavuus – 4 tärkeintä perusperiaatetta eli POUR. <https://wunder.io/fi/artikkelit/saavutettavuus-4-tarkeinta-perusperiaatetta-eli-pour>. 21.11.2019. Luettu 21.5.2020.

21 Tietoa saavutettavuusselosteesta. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi/lait-ja-standardit/tietoa-saavutettavuusselosteesta>. Luettu 20.3.2020.

22 Axe-core. Verkkoaineisto. <https://github.com/dequelabs/axe-core>. Luettu 2.8.2020.

23 Tota11y. Verkkoaineisto. <https://khan.github.io/tota11y>. Luettu 27.5.2020.

24 Introducing VoiceOver. Verkkoaineisto. <https://www.apple.com/voiceover/info/guide/_1121.html>. Luettu 29.5.2020.

25 Ruudunlukuohjelmat. 2019. Verkkoaineisto. <https://papunet.net/saavutettavuus/ruudunlukuohjelmat>. Luettu 29.5.2020.

26 Understanding WCAG 2.1. Verkkoaineisto. <https://www.w3.org/WAI/WCAG21/Understanding/text-alternatives>. Luettu 31.3.2020.

27 Decorative Images. Verkkoaineisto. <https://www.w3.org/WAI/tutorials/images/decorative>. 27.6.2019. Luettu 1.6.2020.

28 Modal Dialog Example. Verkkoaineisto. <https://www.w3.org/TR/wai-aria-practices/examples/dialog-modal/dialog.html>. Luettu 7.6.2020.

29 Introduction to Understanding WCAG 2.1. Verkkoaineisto. <https://www.w3.org/WAI/WCAG21/Understanding/intro>. Luettu 22.6.2020.

30 Understanding Guideline 3.2: Predictable. Verkkoaineisto. <https://www.w3.org/WAI/WCAG21/Understanding/on-input>. Luettu 25.6.2020.

31 Understanding Guideline 3.3: Input Assistance. Verkkoaineisto. <https://www.w3.org/WAI/WCAG21/Understanding/input-assistance>. Luettu 26.6.2020.

32 Understanding Guideline 4.1: Compatible. Verkkoaineisto. <https://www.w3.org/WAI/WCAG21/Understanding/compatible>. Luettu 29.6.2020.