Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Nina Nevalainen & Petra Silavuori

Käyttäjäystävällisen web-sovelluksen suunnittelu ja toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

22.4.2020

Tekijä Otsikko Sivumäärä Aika	Nina Nevalainen & Petra Silavuori Käyttäjäystävällisen web-sovelluksen suunnittelu ja toteutus 56 sivua + 1 liite 22.4.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Hyvinvointi- ja terveysteknologia
Ohjaajat	Ohjaava opettaja: Yliopettaja Mikael Soini Työpaikkaohjaaja: Business Manager Toni Nylund

Tämän insinöörityön aiheena oli tutkia, miten käytettävyydeltään hyvä ja saavutettava web-sovellus rakennetaan ketterän kehityksen metodein sekä moderneja teknologioita käyttäen. Käytettävyyden, saavutettavuuden sekä ketterän kehityksen teorian perusteella suunniteltiin ja toteutettiin tilaajayritykselle web-sovellus, jonka avulla yrityksen työntekijät voivat tehdä sairauspoissaoloilmoituksensa helposti ja nopeasti.

Teorian kartoittamisen jälkeen tehtiin web-sovelluksen käyttöliittymäsuunnitelma. Koska suunnittelutyössä haluttiin varmistua suunnitellun sovelluksen käytettävyyden tasosta, rakennettiin käyttöliittymäsuunnitelman perusteella prototyyppi. Tällä prototyypillä suoritettiin insinöörityön ensimmäinen käytettävyystestaus. Ensimmäisen testauksen jälkeen voitiin todeta käyttäjäkokemuksen olevan jo miellyttävällä tasolla, mutta myös muutama kehityskohta tuli ilmi. Nämä kehityskohdat korjattiin sovelluksen toteutusvaiheen alussa.

Sovelluksen toteutukseen valittiin React ja Node.js -nimiset Javascript-pohjaiset teknologiat, ja niillä rakennettiin sovelluksen käyttöliittymä sekä taustajärjestelmät. Sovelluskehityksen tultua valmiiksi suoritettiin tämän työn toinen käytettävyystestaus. Sen tulosten perusteella web-sovelluksen käytettävyys oli erittäin korkealla tasolla, ja käyttäjät käyttivät sovellusta tehokkaasti ja mielellään. Valmis sovellus testattiin myös validaatiotyökalulla, ja sen tulosten perusteella sekä kasattuun teoriatietoon peilaten voidaan todeta sovelluksen olevan saavutettava, suorituskykyinen sekä hyvien ohjelmistokehityksen toimintatapojen mukainen.

Työn lopputuloksena syntyi tavoitteiden ja asiakasvaatimusten mukainen käyttäjäystävällinen, moderneja teknologioita käyttävä web-sovellus. Insinöörityössä toteutettua sovellusta on mahdollista jatkokehittää tässä työssä kasatun aineiston perusteella.

Avainsanat	Käytettävyys, saavutettavuus, moderni ohjelmistokehitys

Author Title Number of Pages Date	Nina Nevalainen and Petra Silavuori Design and Development of User-Friendly Web Application 56 pages + 1 appendix 22 April 2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Health Technology
Instructors	Mikael Soini, Principal Lecturer Toni Nylund, Business Manager

The topic of this thesis was to study how a modern and user-friendly web application is designed and developed using agile methodologies while following good usability and accessibility standards. Using the theory of usability and accessibility a web application was built for the client company. This web application enables the employees to submit a sick leave notice with ease and speed.

After studying related theories, a preliminary plan for the user interface of the web application was made. In order to ensure that the quality of the usability of the application during the design phase was at a proper level, a prototype was built based on the preliminary plan. This prototype was used for the first usability test of the study. After the first test it became clear that the usability of the application was already at a pleasing level, but few points of improvement were raised. These points were fixed during the early phases of the application build phase.

To build the user interface and the back end of the application, Javascript-based technologies React and Node.js were chosen. After the build phase, the second usability test was performed. Based on the results of this test, the usability of the application was very high, and the users found the usage of the application to be effective and pleasing. The final product was also tested with a validation tool and based on the results and the surrounding theory, the application is very accessible, performs well and has been built applying good software development standards.

The final product of this work was a highly user-friendly web application that was built using modern technologies. The application can be further developed based on the material and information gathered in the thesis.

Keywords	Usability, accessibility, modern software development

Sisällys

Lyhenteet

API	*Application Programming Interface (ohjelmointirajapinta)*. Määritelmä, jolla tehdään pyyntöjä eri ohjelmien välillä tiedon hakemiseksi sekä lähettämiseksi.
CSS	*Cascading Style Sheets*. Erityisesti WWW-dokumenteille kehitetty tyylioh-jeiden laji.
GUI	*Graphical User Interface (graafinen käyttöliittymä).* Tarkoittaa käyttäjän ra-janpintaa sovellukseen, jonka kautta hän voi käyttää sovellusta ja lähettää siihen toimintapyyntöjä.
HTML	*Hypertext Markup Language (hypertekstin merkintäkieli)*. Kuvauskieli, jolla verkkosisällön rakenne ja merkitys määritellään.
HTTP	*Hypertext Transfer Protocol*. Selaimien ja WWW-palvelimien käyttämä pro-tokolla tiedonsiirrossa.
PX	*Pixel (pikseli)*. Kuvan pienin yksittäinen osa, jolla usein kerrotaan kuvaruu-tujen koko, ja joka on myös yleinen yksikkö ohjelmistokehityksessä.
REST	*Representational State Transfer*. Ohjelmointirajapinnoissa käytetty arkkiteh-tuurimalli, joka perustuu HTTP-protokollaan.
SASS	*Syntantically Awesome Style Sheets*. CSS:n lisäosa, joka toimii sen syntak-sin laajentamisessa ja avustaa sen tulkitsemisessa selaimeen.
SUS	*System Usability Scale*. Käyttäjäkokemuksen arviointimenetelmä, jonka avulla voidaan määrittää tuotteen käytettävyyden kokonaisarvo.
WCAG	*Web Content Accessibility Guidelines*. Kokoelma saavutettavuussuosituk-sia web-sivustoille.

1 Johdanto

Tämän insinöörityön tavoite on luoda tilaajalle moderni, käytettävyydeltään hyvä sairauspoissaolokirjausjärjestelmä sekä raportoida kirjallisesti, mitä modernin ja käyttäjäystävällisen web-sovelluksen tuottaminen vaatii sekä mitä teknologioita tähän kyseiseen projektiin valitaan.

Insinöörityössä rakennettavan sovelluksen tilaajana toimii ALTEN Finland. ALTEN on ranskalainen monikansallinen teknologia- ja insinöörikonsultointiyritys, joka on perustettu 1988. ALTENilla on toimistoja 25 maassa, ja ALTEN Finland on Suomen yksikkö. Vuonna 2018 ALTENilla oli maailmanlaajuisesti 33 700 työntekijää, joista Suomessa on noin 600. ALTEN Finlandilla on Suomessa 11 toimistoa.

Tilaajan nykyinen tulostettava sairauspoissaoloilmoituslomake on aikaa vievä tapa ilmoittaa poissaolosta sitä valvovalle taholle. Digitalisoinnin toivotaan nopeuttavan ilmoittamisprosessia. Samalla se nostaa työntekijän työhyvinvointia vähentämällä turhaa ja ylimääräistä työkuormaa tilanteessa, jossa työntekijä on muutenkin jo normaalia rasittuneempi. Lomakkeiden säilytys ja ilmoitusten hyväksyntäprosessi saadaan myös keskitettyä yhteen paikkaan, kun siirrytään digitaaliseen ilmoitusjärjestelmään.

Sairauspoissaolokirjausjärjestelmän suunnitteluprosessissa halutaan selvittää, mikä määrittelee käyttäjäystävällisen sovelluksen sekä mitä sellaisen suunnittelussa tulee ottaa huomioon. Isoon osaan nousevat käytettävyys sekä saavutettavuus, jotka ovat tärkeitä teemoja web-sovelluksen käyttökokemuksesta puhuttaessa. Käyttäjäkeskeisellä suunnittelulla pyritään tuottamaan sovellus, joka on helppokäyttöinen ja käyttäjäryhmälleen sopiva tuote.

Sovellusta suunnitellessa tullaan ottamaan käyttäjät heti alusta asti huomioon. Sovelluksen käyttöliittymäsuunnittelussa painotetaan vahvasti käytettävyyden ja saavutettavuuden hyviä käytäntöjä. Käyttöliittymä testataan jo prototyyppivaiheessa käytettävyystestissä, jossa tuotteen loppukäyttäjät pääsevät osallistumaan suunnitteluprosessiin.

Sovellus rakennetaan käytettävyystestauksen tulosten perusteella ja työn lopussa toistetaan käytettävyystestaus valmiilla sovelluksella. Tavoitteena on, että valmis sovellus täyttää tilaajan toiveet sekä on loppukäyttäjien mielestä erittäin helppo ja miellyttävä käyttää. Itse sovelluskehitys halutaan tehdä ketterän kehityksen toimintamalleilla.

Kaikki edellä mainittu tullaan toteuttamaan ketterän kehityksen periaatteita noudattaen. Tämä helpottaa sekä edesauttaa sovelluksen jakamista pieniin osiin. Nämä osat ovat hallittavia kokonaisuuksia sovelluksesta, jotka on mahdollista toteuttaa lyhyiden, noin 1-2 viikon, kehitysiteraatioiden aikana.

2 Taustatietoja sekä asiakasvaatimukset

ALTEN Finlandin työntekijät ilmoittavat tällä hetkellä maksimissaan kolme päivää kestävät sairauspoissaolonsa paperisella lomakkeella, joka toimitetaan työntekijän esimiehelle. Tällaisesta poissaolosta tehtävästä ilmoituksesta käytetään tilaajalla nimitystä ”omailmoitus sairauspoissaolosta”, eikä tällaiseen poissaoloon vaadita liitteeksi lääkärintodistusta. Digitalisoimalla tämä prosessi kaikki tarvittava tieto löytyy sähköisesti, eikä siirreltäviä papereita tarvitse enää itse toimittaa henkilökohtaisesti tai lähettää muita reittejä pitkin. Sähköistetyn järjestelmän tarpeellisuus korostuu varsinkin tilanteissa, joissa työntekijän on vaikea toimittaa henkilökohtaisesti paperista lomaketta eteenpäin esimiehelle.

Työntekijöiden sairauspoissaolojen seurantaa vaaditaan työterveyshuoltolaissa, sillä työnantajan tulee ilmoittaa työterveyshuoltoon työntekijän pitkittyneestä sairauspoissaolosta. Myös tämän järjestäminen onnistuu helposti sähköisellä järjestelmällä, sillä osa toiminnoista voidaan automatisoida. Järjestelmä voisi esimerkiksi antaa hälytyksen esimiehelle, mikäli työntekijän sairauspoissaolot voidaan laskea pitkittyneiksi.

2.1 Sairauspoissaolojen seuranta

Työterveyslaissa määrätään, että yksi työnantajan velvollisuuksista on työntekijän sairauspoissaoloista ilmoittaminen. Työnantajan tulee ilmoittaa työntekijän sairauspoissaolosta työterveyshuoltoon viimeistään silloin, kun poissaolo on jatkunut kuukauden ajan.

(Työterveyshuoltolaki 1383/2001, § 10a). Työterveyslaitoksen mukaan työnantajan tulee tehdä ilmoitus myös, mikäli työntekijän poissaolo on jatkunut lyhyemmissä jaksoissa yhteensä 30 päivän ajan yhden vuoden sisällä.

Tällainen pitkittynyt sairauspoissaolo, tai toistuvat lyhyet sairauspoissaolot, voivat olla merkkejä työntekijän työkyvyn alentumisesta tai muista työhön liittyvistä ongelmista. Työntekijöiden sairauspoissaolojen seuraaminen on tärkeää työnantajan näkökulmasta, sillä näin voidaan seurata työntekijän työkykyä ja pystytään reagoimaan nopeasti, mikäli työntekijä tarvitsee tukea työkyvyn palautumiseen. (Työterveyshuolto.)

2.2 Asiakasvaatimukset

Tilattu sovellus on korvaava versio vanhalle paperiselle lomakkeelle, joka tulostettiin ja palautettiin fyysisenä kopiona niitä valvovalle taholle. Tämä web-sovellus tulee olemaan internetsivusto, joka on suunniteltu ja rakennettu toimimaan niin suurilla kuin mobiilikokoisillakin ruuduilla. Sovelluksesta pyritään rakentamaan virtaviivainen: tarkoituksena on minimoida käyttäjän sovelluksessa kuluttama aika, jotta käyttäjämukavuus ei kärsi turhista ominaisuuksista eikä epäolennaisesta sisällöstä.

Lyhytkestoisen sairauspoissaolon kirjaamiseen tehty sivu tulee perustumaan tilaajan tällä hetkellä käyttämään lomakkeeseen, ja sisältää samat täytettävät kohdat nettilomakkeen muodossa. Lomakkeen täytön lisäksi sovelluksessa on tärkeä sitoa lomakkeen data aina tiettyyn käyttäjätiliin, joten sivustolla tulee olla mahdollisuus rekisteröitymiseen sekä sisäänkirjautumiseen käyttäjätunnuksilla. Näiden vaatimusten perusteella tehtiin alustava suunnitelma sovelluksen sisällöstä.

Tämän suunnitelman mukaan sovellus pitää sisällään kolme pääsivua, jotka sisältävät käyttäjän useimmiten käyttämät ominaisuudet ja jotka myös ovat merkittävimmät sovelluksen toimivuudelle. Nämä kolme sivua ovat seuraavat:

- sisäänkirjautumissivu
- käyttäjän etusivu
- sivu sairauspoissaoloilmoituksen tekemiseen.

Näiden lisäksi sivustolla on sivuja, jotka ovat tarpeellisia olla olemassa sovelluksessa, mutta joiden käyttäminen ei välttämättä ole koskaan ajankohtaista kaikille yksittäisille käyttäjille. Tämän vuoksi ne pidetään sivuston hierarkiassa alhaisemmassa asemassa. Näitä sekundäärisiä sivuja on kolme, ja ne ovat seuraavat:

- uuden käyttäjän rekisteröintisivu
- salasanan palautussivu
- omien tietojen muutossivu.

Uuden käyttäjän tiedot, kuten myös itse sairauspoissaoloilmoituksen tiedot, tulevat siirtymään sovelluksesta REST (Representational State Transfer) API (Application Programming Interface) -rajapintoja käyttäen tietokantaan, johon tiedot tallennetaan. Kyseisen rajapinnan kautta tallennettuja tietoja on mahdollista käsitellä myöhemmin tarpeen tullen, kuten käyttäjän kirjautuessa sisään, tai sairauspoissaoloilmoituksia tarkastellessa.

Tässä insinöörityössä on tarkoituksena luoda peruskäyttäjän toiminnallisuudet. Peruskäyttäjälle sivusto on yksinkertainen, sillä hänelle näkyvät käyttäjän etusivulla ainoastaan hänen omat tietonsa sekä historia hänen lähettämistään ilmoituksista, joiden tila (hyväksytty/hylätty) päivittyy automaattisesti tilan muuttuessa. Peruskäyttäjälle tärkeimmät toiminnallisuudet ovat käyttäjätietojen hallinta sekä sairauspoissaolon omailmoituksen tekeminen. Peruskäyttäjän lisäksi sovelluksessa tulee myöhemmin, tämän insinöörityön ulkopuolella, olemaan hallintakäyttäjiä, jotka omien tietojensa lisäksi pääsevät näkemään sovellukseen lähetetyt sairauspoissaoloilmoitukset, joita he voivat hallita sekä hyväksyä tai hylätä. Tilaajan on mahdollisesti tarkoitus jatkokehittää sovellusta myöhemmin lisäten myös muita toiminnallisuuksia sekä muita lomakkeita sovelluksen kautta käytettäväksi.

2.3 Teknologioiden valitseminen sovelluksen rakentamiseen

Sovelluksen rakentamiseen tarvittiin sopivat teknologiat niin käyttöliittymän (front-end) tuottamiseen kuin myös taustajärjestelmien (back-end) tekoon. Teknologia sanana tässä kontekstissa kattaa niin ohjelmointikielen kuin myös siihen mahdollisesti liittyvät kirjastot

ja ohjelmistokehykset sekä muut taustateknologiat. Koska nykyään markkinoilla on laajalti avoimeen lähdekoodin perustuvia teknologioita, on ohjelmistokehittäjän tärkeää valita projektiinsa sopivat teknologiat ei ainoastaan sen perusteella, mikä vaikuttaa kiinnostavimmalta ja uusimmalta, myös muita kriteerejä ajatellen. Kaikki tässä projektissa käytetyt teknologiat valittiin neljää kriteeriä käyttäen, jotka määrittelivät niiden sopivuuden projektiin:

- luotettavuus
- laaja käyttäjäkunta
- teknologian ylläpitäjien tuki ja aktiivisuus
- yhteensopivuus.

Luotettavuudella tässä tapauksessa tarkoitetaan mahdollisen teknologian niin sanottua bugittomuutta, eli miten virheetön teknologia on käytössä. Tämä on erityisen tärkeää ammattitason sovelluksissa, joissa halutaan minimoida mahdolliset haitalliset tekijät, jotka hankaloittavat ohjelmistokehittäjän mahdollisuutta tehdä sovelluksesta virheetön sen käyttäjille. Mitä luotettavampi teknologia on käyttää, sitä nopeammin ja varmemmin ohjelmistokehittäjä pystyy luomaan sen avulla haluamansa toiminnallisuudet.

Laaja käyttäjäkunta on suoraan liitoksissa niin luotettavuuteen kuin myös kehittäjien tukeen, sillä mitä suurempi käyttäjäkunta teknologialla on, sen tehokkaammin tulevat mahdolliset virheet esille. Teknologian ylläpitäjät korjaavat nämä virheet ja näin ollen parantavat teknologian luotettavuutta. Markkinoilla on useita mielenkiintoisia uusia ohjelmointikieliä, kirjastoja sekä muita teknologioita, mutta jos suurta käyttäjäkuntaa ei ole, ei tätä teknologiaa tulisi käyttää ammattiprojektissa, sillä se ei ole vielä saavuttanut tarpeellista luotettavuuden tasoa.

Teknologian ylläpitäjien tuesta ja aktiivisuudesta mainittiin jo aiemmissa luvuissa, mutta tämä on kohta, jonka tärkeyttä ei voi liioitella. Ilman teknologian aktiivista ylläpitäjien sekä kehittäjien (usein yksi ja sama asia) tukea, on kyseisen teknologian käyttö turhan riskialtista, sillä ei voida luottaa, että tämä teknologia olisi aktiivisessa ja luotettavassa tilassa vielä vuosienkin päästä. Vaikka paljon puhutaan siitä, miten nopeasti ala kehittyy, ja uusia teknologioita ilmestyy markkinoille nopeaan tahtiin, on aina olemassa myös niitä teknologioita, joita voidaan pitää luotettavina ja joiden tuki on melkeinpä taattu. Esimerkkinä

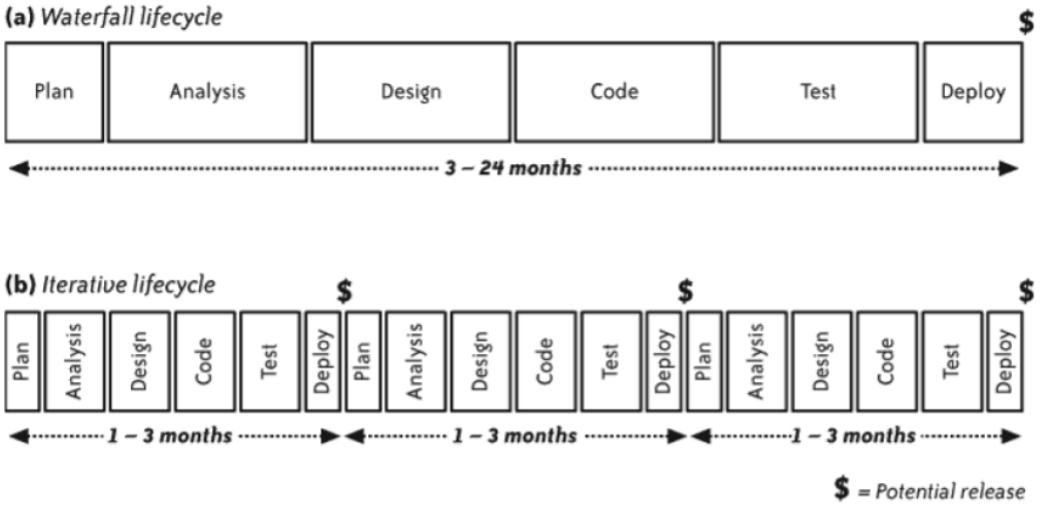
tämänkaltaisesta teknologiasta voidaan pitää React-nimistä avoimen lähdekoodin Javascript-kirjastoa. Reactin takana on paljon tukea sekä aktiivisuutta, sillä sen on luonut ja sitä ylläpitää Facebook Inc. On tosin sanottava, että teknologian ei tarvitse olla varakkaan monikansallisen yrityksen luomus ollakseen käytettävä sekä luotettava, vaikka usein siitä onkin hyötyä.

Yhteensopivuudella tarkoitetaan sitä, miten helppoa käytetyt teknologiat, esimerkiksi käyttöliittymässä ja taustajärjestelmissä, on saada keskustelemaan sekä toimimaan saumattomasti yhdessä. Toki melkein mitkä tahansa teknologiat on mahdollista saada toimimaan yhdessä, mutta usein järkevintä on valita käyttöön ne, joiden saaminen toimimaan yhdessä on ohjelmistokehittäjälle mahdollisimman vaivatonta. Tässä projektissa valittiin käyttöön, niin käyttöliittymään kuin myös taustajärjestelmiin, Javascript-pohjaiset teknologiat, sillä niiden yhteensopivuus on korkealla tasolla.

Näiden tietojen pohjalta päätettiin, että sovelluksen käyttöliittymän rakentaminen tehdään Reactilla, sekä sivuston tyylittelyyn käytetään CSS:ää (Cascading Style Sheets). Taustajärjestelmien toteutukseen käytetään Node.js-nimistä Javascript-kirjastoa, ja tietokannaksi valittiin MongoDB. Tietokanta on tässä työssä vain testauskäyttöä varten, sillä tilaaja yhdistää sovelluksen omaan tietokantaansa myöhemmin. Tietokannan toiminnallisuuden mallintamiseen ja testaamiseen MongoDB on hyvä valinta, sillä sinne tallentaminen ei vaadi vastaanotettavan datan vahvaa etukäteismäärittelyä.

2.4 Ketterä kehitys ohjelmistotuotannossa

Ketterä kehitys on kattotermi erinäisille menetelmille, joita käytetään ohjelmistokehityksessä projektin läpiviemiseen mahdollisimman ketterästi. Usein tämä tapahtuu niin sanotuissa nopearytmisissä sprinteissä, joiden aikana, ja projektin edetessä, projektin tarpeet saattavat muuttua sekä tarkentua (Shore & Warden 2008, 9-12). Tässä toimintatavassa on tarkoitus maksimoida arvon luonti asiakkaalle mahdollisimman pienillä resursseilla sekä mukautuvasti, jotta asiakas näkee mahdollisia tuloksia ja vastinetta sijoitukselleen mahdollisimman usein. Tämän tyyppinen ohjelmistokehitys on vaihtoehto perinteiselle vesiputousmallille (kuva 1).



Kuva 1. Esimerkki vesiputousmallista, eli lineaarisesta kehityksestä, sekä ketterästä, eli iteratiivisesta kehityksestä (Shore & Warden 2008, 16).

Vesiputousmalli tarkoittaa lineaarista kehitystä, jossa edellisen osan tuotoksen saaminen valmiiksi on edellytyksenä prosessin seuraavalle vaiheelle, ja tuloksien näkyminen on usein pidemmän aikavälin tulos. Ketterässä kehityksessä eri osien kehityksen kesto on lyhyempi, ja kaikkia osia kehitetään jatkuvassa syklissä.

Ketterä kehitys voi tarkoittaa montaa eri metodia, mutta lopulta se tarkoittaa ainoastaan ketterän toimintavan filosofian seuraamista, joka perustuu vuonna 2001 laadittuun “Agile Manifesto” -nimiseen julkaisuun (Shore & Warden 2008, 10), jonka kirjoittajia olivat useat aikansa ketterän kehityksen puolestapuhujat. Suomennettu versio julkaisusta on seuraavanlainen:

> Me etsimme parempia keinoja ohjelmistojen kehittämiseen tekemällä sitä itse ja auttamalla siinä muita.
> Tässä työssämme olemme päätyneet arvostamaan:
>
> **Yksilöitä ja vuorovaikutusta** enemmän kuin prosesseja ja työkaluja.
> **Toimivaa sovellusta** enemmän kuin kokonaisvaltaista dokumentaatiota.
> **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluita.
> **Muutokseen reagoimista** enemmän kuin suunnitelman noudattamista.
>
> Vaikka oikeallakin puolella on arvoa, me arvostamme vasemmalla olevia asioita enemmän.

Ketterä, eli iteratiivinen ohjelmistokehitys, on saanut paljon suosiota, mutta se ei välttämättä tee ketterästä ohjelmistokehityksestä lineaarista (vesiputousmalli) ohjelmistokehitystä parempaa. Se, miten ohjelmistoprojektia on parasta ja tehokkainta edistää ja suunnitella, on aina sitä työstävän ryhmän päätettävissä. Ei ole olemassa objektiivisesti parasta metodia, ainoastaan useita eri vaihtoehtoja, sillä projektit ja asiakkaat ovat aina yksilöllisiä omine tarpeineen.

Eräs syy ketterän kehityksen suosiolle tulee asiakkaiden toiveesta tämäntyyppiseen ohjelmistokehitykseen. Ketterässä kehityksessä monesti niin sanottu sprintti, joka on usein noin 1-2 viikon pituinen, päättyy asiakkaalle esitettävään demonstraation. Demonstraatiossa esitellään, mitä meneillään ollessa kehitysiteraatiossa on tuotettu, ja tämä on tehokas tapa ylläpitää asiakkaan tyytyväisyyttä.

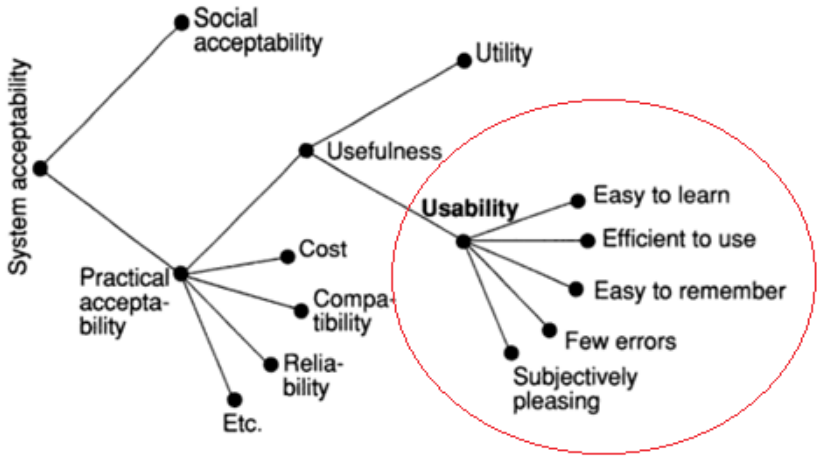
Toisena tärkeänä syynä voidaan pitää ketterän ohjelmistokehityksen kautta tulevaa tarvetta pilkkoa projekti pienempiin osioihin, jotta olisi mahdollista tuottaa edellä mainittu demonstraatio asiakkaalle. Tämä pieniin osioihin pilkkominen on usein ohjelmistokehittäjälle kevyempi tapa edistää projektia, joka omalta osaltaan edesauttaa työntekijän hyvinvointia. (Shore & Warden 2008, 275-276.)

3 Käytettävyys ja saavutettavuus

Tässä insinöörityössä rakennettavan sovelluksen suunnittelussa haluttiin vahvasti painottaa hyvää käyttäjäkokemusta. Käyttäjäkokemuksessa isossa osassa ovat käytettävyys sekä saavutettavuus, varsinkin kun kyseessä on web-sovellus. Koska sovellusta käyttävä henkilö tulee pääsääntöisesti käyttämään sovellusta yksin ilman ohjausta, tulisi sovelluksen käytön olla helppoa ja miellyttävää. Sovelluksella on myös tarkoitus tehostaa sairauspoissaoloilmoituksen tekoa, joten ilmoituksen tekemisen tulisi onnistua mahdollisimman nopeasti. Mitä korkeampi käytettävyys sekä saavutettavuus käytettävällä sovelluksella on, sitä paremmin näihin tavoitteisiin päästään.

3.1 Käytettävyys

Käytettävyydelle on useita määritelmiä. Nielsenin (1993) mukaan tuotteen hyväksyttävyys koostuu monesta osasta, joista yksi on käytettävyys. Käytettävyys on laadun attribuutti, jolla arvioidaan tuotteen tai palvelun helppokäyttöisyyttä. Käytettävyys voidaan jakaa vielä viiteen ala-attribuuttiin: opittavuuteen, tehokkuuteen, muistettavuuteen, virheettömyyteen sekä subjektiiviseen miellyttävyyteen (kuva 2).

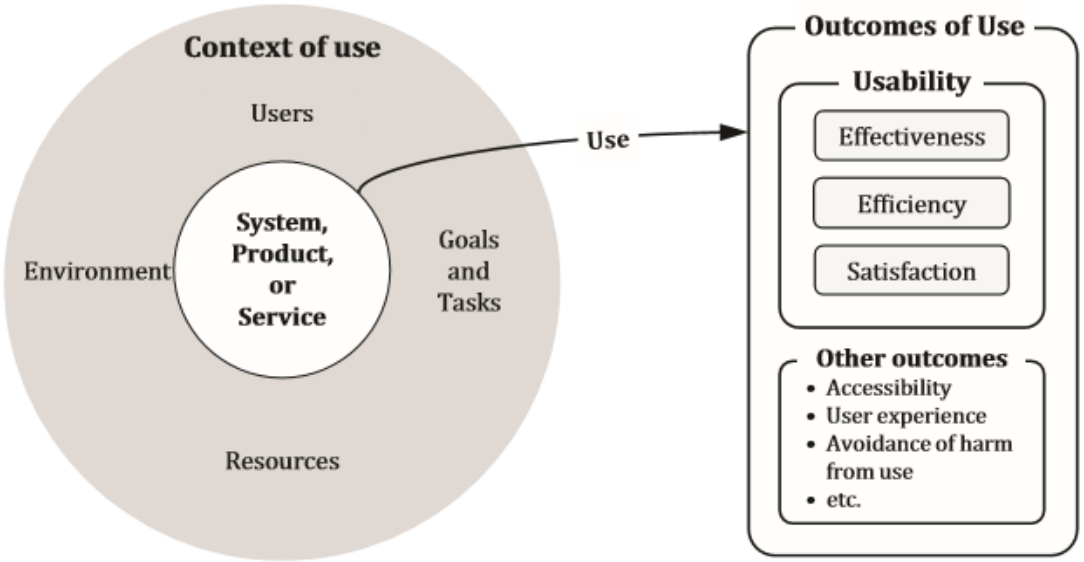


Kuva 2. Järjestelmän hyväksyttävyyden osa-alueet, joista käytettävyys on yhtenä osana. (Nielsen 1993, 25)

Opittavuudessa tarkastellaan sitä, kuinka helppo ensikertalaisen on käyttää tuotetta, kun taas tehokkuudessa selvitetään, kuinka helppoa kokeneen käyttäjän on käyttää tuotetta. Muistettavuudessa tarkastellaan sitä, kuinka helppoa satunnaisesti tuotetta käyttävän käyttäjän on käyttää tuotetta tauon jälkeen. Virheettömyydellä pyritään siihen, että käyttäjä tekee mahdollisimman vähän virheitä tuotetta käyttäessään. Miellyttävyydessä tarkastellaan sitä, kuinka mukavaa käyttäjän on käyttää tuotetta. (Nielsen 1993, 25-35.)

Käytettävyyteen ottaa kantaa myös ISO 9241-11 standardi. Kyseinen standardi on vapaamuotoisesti suomennettuna nimeltään ”Ihmisen ja järjestelmän vuorovaikutuksen ergonomia. Osa 11: Käytettävyys: määrittely ja konsepti”. Standardi määrittelee, että kun järjestelmien, tuotteiden ja palveluiden suunnittelu ja määritys tehdään käytettävyyttä ajatellen, mikä mahdollistaa loppukäyttäjän tavoitteiden saavuttaminen tehokkaasti, tu-

loksellisesti ja tyydyttävästi. Standardin mukaan käytettävyys on riippuvainen käyttökontekstista (kuva 3). Käytettävyyttä arvioidessa tuotteen käytön tavoitteet tulee kuvata, ja näiden perusteella tarkastellaan, kuinka käytettävä tuote on.



Kuva 3. Käytettävyys syntyy järjestelmän, tuotteen tai palvelun käytöstä sen käyttökontekstissa. (ISO 9241-11:2018, 12)

Kuten kuvassa 3 mallinnetaan, ISO 9241-11 -standardin mukaan järjestelmän, tuotteen tai palvelun käyttökonteksti koostuu käyttäjistä, tavoitteista ja tehtävistä, resursseista sekä ympäristöstä. Käytettävyys koostuu tehokkuudesta, tuloksellisuudesta sekä tyytyväisyydestä, ja se on järjestelmän, tuotteen tai palvelun käytön seurausta. Käytännössä käytettävyys saavutetaan, kun tuote sopii juuri kyseisen tuotteen käyttäjälle, sekä siihen käyttöympäristöön että tehtäviin, joihin se on tarkoitettu. ISO 9241-11 -standardissa nostetaan esiin käytettävyyden mittareina Nielsenin tapaan tehokkuus ja tyytyväisyys, mutta lisäksi mainitaan myös tuloksellisuus. Tuloksellisuudella tarkoitetaan tarkkuutta ja täydellisyyttä, joilla käyttäjä saavuttaa asetetut tavoitteet. Standardin mukaan käytettävyys on olennaista, kun kehitetään, hankitaan, läpikäydään, vertaillaan tai markkinoidaan järjestelmää, tuotetta tai palvelua. (ISO 9241-11:2018, 5, 11-12, 15-18.)

Tuotekehityksessä käytettävyyden tärkeys nousee esiin, kun halutaan suunnitella uutta tai kehittää jo olemassa olevaa tuotetta. Hyvä käytettävyys parantaa käyttäjä-/asiakaskokemusta sekä tuo konkreettista rahansäästöä tuotetta valmistavalle yritykselle. Kun tuotetta on helppo käyttää, saadaan sen käyttökoulutuskuluja sekä käyttövirheitä vähennettyä. Käytettävyyden on myös todettu vaikuttavan paljon yrityksen brändin arvoon sekä markkinaosuuteen. (Bias & Mayhew. 2005, 17-18.) Myös ISO 9241-11 -standardissa mainitaan, että käytettävyyden ollessa heikko eivät käyttäjät välttämättä pysty tai halua käyttää tuotetta. Käytettävyyden ollessa riittävällä tasolla tuote tarjoaa tarkoitetut hyödyt käyttäjälle, ja käytettävyyden taas ollessa korkealla tasolla on tuotteella kilpailuetu sen markkinoilla.

Web-sovelluksia suunnitellessa käytettävyys on erittäin tärkeää ottaa huomioon, sillä web-sovellusten käyttö on yleensä vapaaehtoista, ja kilpailu muiden sovellusten kanssa on kovaa. Tämän takia web-sovellus on tehtävä mahdollisimman helppokäyttöiseksi sekä houkuttelevaksi. Muuten käyttäjät siirtyvät kilpailevan sovelluksen käyttäjiksi. Myös niissä web-sovelluksissa, joita käyttäjät käyttävät työnsä puolesta, tulee käytettävyys ottaa huomioon, vaikkei pelkoa toisen sovelluksen pariin siirtymisestä olekaan. Työhön ja työntekoon liittyviä sovelluksia käytettäessä on tärkeää, että niillä tehtävät toiminnot tulee tehdyksi virheettä ja nopeasti. (Sinkkonen ym. 2009, 17-18.)

Nielsenin (1994) määrittelemät käytettävyyden heuristiikat ovat erittäin hyvä lähtökohta, kun web-sovelluksen käytettävyyttä lähdetään suunnittelemaan ja arvioimaan. Kyseinen 10 säännön lista on vapaasti suomennettuna seuraava:

1. **Järjestelmän tilan näkyvyys**. Järjestelmän tulee aina sopivalla palautteella pitää käyttäjä tietoisena siitä, mitä tapahtuu.
2. **Järjestelmän ja todellisen maailman yhteensopivuus**. Järjestelmän tulee käyttää käyttäjälle tuttuja sanoja ja konsepteja, järjestelmäpainotteisten termien sijaan.
3. **Käyttäjän kontrolli ja vapaus**. Käyttäjän valitessa väärän toiminnon tulee hänelle tarjota selkeästi merkitty poistumiskeino.
4. **Yhdenmukaisuus ja standardit**. Käyttäjän ei tule joutua miettimään, tarkoittavatko eri sanat, tilanteet tai toiminnot samaa asiaa. Seuraa yleisiä käytäntöjä.
5. **Virheiden estäminen**. Eliminoi virheisiin johtavat olosuhteet, tai varmista käyttäjältä toiminnan tarkoituksellisuus ennen kuin toiminto suoritetaan.

6. **Tunnistaminen mieluummin kuin muistaminen**. Käyttäjän muistin kuormitusta tulee välttää käyttämällä näkyviä kohteita. Käyttäjän ei tule joutua muistamaan informaatiota siirryttäessä kohdasta toiseen. Järjestelmän käyttämisen ohjeistus tulee olla näkyvillä tai helposti saatavilla.
7. **Käytön joustavuus ja tehokkuus**. Pikavalinoilla voidaan nopeuttaa asiantuntijan vuorovaikutusta järjestelmän kanssa. Mahdollista usein käytettyjen toimintojen räätälöinti.
8. **Esteettinen ja minimalistinen suunnittelu**. Dialogin ei tule sisältää tarpeetonta informaatiota. Kaikki ylimääräinen informaatio vie näkyvyyttä tarpeelliselta informaatiolta.
9. **Virhetilanteiden tunnistus, diagnosointi ja niistä palautuminen**. Virheviestit tulee tehdä selkokielisiksi ja tarkoiksi, ja niiden tulee tarjota ratkaisuehdotus.
10. **Opastus ja dokumentaatio**. Vaikka järjestelmää on hyvä pystyä käyttämään ilman dokumentaatiota, voi olla tarpeellista tarjota apua. Tällaisen informaation tulee olla helposti löydettävissä ja keskittyä tukemaan käyttäjää tämän tehtävissä.

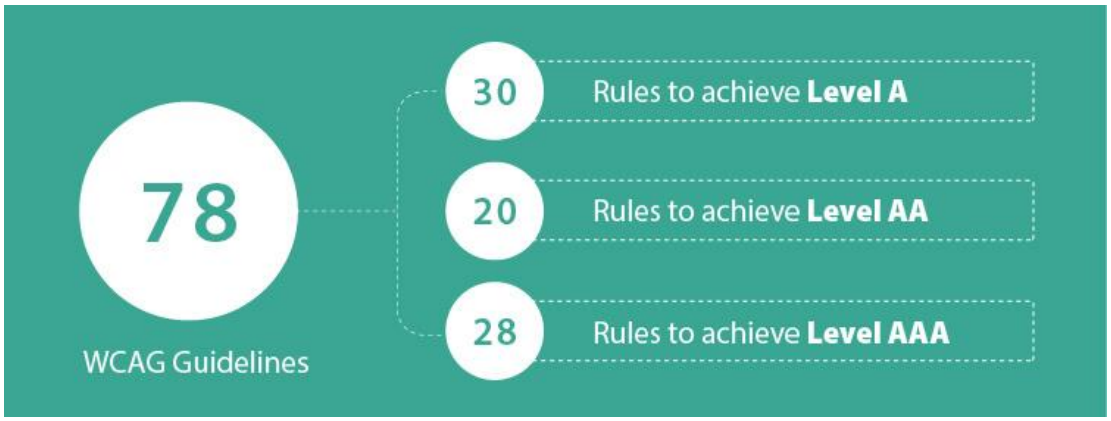
Käytettävyyttä voidaan arvioida eri käytettävyystutkimuksen menetelmin. Käytettävyystutkimuksen arviointimenetelmät jaetaan usein kahteen ryhmään riippuen siitä, osallistuuko käyttäjä arviointiin vai ei. (Ovaska ym. 2005, 5-6.) Riihiahon (2000) mukaan ilman käyttäjää tehtävää arviointia kutsutaan asiantuntija-arvioinniksi, ja käyttäjän kanssa tehtävää arviointia kutsutaan käyttäjätestiksi. Asiantuntija-arvioihin lukeutuu muun muassa heuristinen arviointi, kognitiivinen läpikäynti sekä standardikatselmus. Käyttäjätesteihin kuuluu muun muassa käytettävyystestaus, paritestaus sekä ryhmäläpikäynti. Tässä insinöörityössä käytettävyyden arviointimenetelmäksi valittiin käytettävyystestaus, joka esitellään tarkemmin luvussa 5.

3.2 Saavutettavuus

Verkkopalveluissa saavutettavuus tarkoittaa niiden käytön esteettömyyttä, eli sitä, onko suunnittelussa otettu huomioon käyttäjien erityistarpeet ja muut mahdollisesti verkkosivuston käyttöön vaikuttavat tekijät. Kun suunnitellaan ja toteutetaan palvelua, jonka halutaan olevan saavutettava, tulee ottaa huomioon helppokäyttöisyys, tekninen toteutus sekä sisältöjen selkeys ja ymmärrettävyys. Teknisessä toteutuksessa tulee huomioida

lähdekoodin virheettömyys ja loogisuus, jotta esimerkiksi puheohjaus sekä ruudunlukuohjelmat toimivat verkkopalvelussa hyvin.

EU:n saavutettavuusdirektiivi ((EU) 2016/2102) määrittelee tavoitteet, joihin EU-maiden on pyrittävä. Jokainen maa saa kuitenkin itse päättää lait, joilla nämä tavoitteet toteutetaan. Suomessa on laki nimeltä “Laki digitaalisten palveluiden toteuttamisesta 306/2019”, joka velvoittaa niin julkista sektoria kuin myös yksityistä ja kolmatta sektoria osaltaan noudattamaan tätä lakia ja siinä määriteltyjä saavutettavuusvaatimuksia. Tämä laki perustuu WCAG-standardiin (Web Content Accessibility Guidelines), jossa saavutettavuus arvioidaan kolmella eri saavutettavuusluokalla: A, AA ja AAA (kuva 4).



Kuva 4. Jokainen saavutettavuusluokka vaatii tiettyjen sääntöjen noudattamisen, ja korkeimman AAA-luokan internetsivuston rakentaminen vaatii kaiken kaikkiaan 78 säännön noudattamista. (Outsystems)

Luokista A on vaatimustasoltaan matalin luokan AAA ollessa korkein. Kuitenkin vain A ja AA saavutettavuusluokat ovat luokkia, joiden kriteerejä laki määrää seuraamaan. Mikäli verkkosovellus täyttää kaikki A-saavutusluokan kriteerit, mutta ei kaikkia AA-saavutusluokan kriteerejä, tulee sivuston saavutettavuusluokaksi A. (The World Wide Web Consortium (W3C).)

Koska WCAG:n mukaan määritelmiä luokille on erittäin paljon, ei niiden tarkka listaaminen ole järkevää. Jotta lukija kykenisi luomaan selkeän kokonaiskuvan saavutettavuusluokista ja jonka perusteella on mahdollista saada kokonaisvaltainen kuva kunkin luokan vaatimuksista, voidaan jokaista saavutettavuusluokkaa arvioida neljän eri perusperiaatteen näkökulmasta. Nämä ovat WCAG:n mukaan

- havaittavuus
- hallittavuus
- ymmärrettävyys
- lujatekoisuus.

Näiden neljän periaatteen kautta jokaisella saavutettavuusluokalla on eri määrä vaatimuksia, ja niissä kriteerejä, joista jokainen on täytettävä, jotta saavutettavuusluokka toteutuu. Kaikissa vaatimuksissa ei ole kriteerejä jokaiselle saavutettavuusluokalle. Tästä on esimerkkinä Havaittavuusperiaatteen alla olevassa Tekstivastineet-vaatimuksessa (WCAG kohta 1.1), jonka alla oleva onnistumiskriteeri ”Ei-tekstuaalinen sisältö” (WCAG kohta 1.1.1) määrittelee ainoastaan kriteerit, jotka täyttämällä voi saada saavutettavuusluokan A. Käytännössä tämä kuitenkin tarkoittaa, että mikäli verkkopalvelu haluaa saavutettavuusluokan AA, on tämäkin edellä mainittu luokan A kriteeri täytettävä, vaikka kriteerejä ei erikseen ollut saavutettavuusluokalle AA.

Toisena hyvänä esimerkkinä saavutettavuusluokkien erosta on vaatimus, jossa on selkeästi kuvattu eri vaatimusluokkien kriteerierot. Hallittavuusperiaatteen alla kohdassa Sairauskohtaukset (WCAG kohta 2.3) löytyy kohdat 2.3.1 ja 2.3.2, joissa käsitellään verkkopalvelussa tapahtuvia ruudun välähdyksiä. Kohdassa 2.3.1 annetaan kriteerit saavutettavuusluokalle A, jotka ovat tiivistetysti seuraavat:

> Mikään sivun oleva sisältö ei välky tiheämmin kuin 3 kertaa sekunnissa. Poikkeukset:
>
> - välkkyvä sisältö on kooltaan pieni
> - välähdysten kontrastisuhde on pieni
>
> Välähdysten sisältämä määrä punaista on rajattu. Kriteerin täyttävä esimerkki: Sivuun upotetussa elokuvassa kirkas salaman välke on editoitu niin, että sen välkyntänopeus on enintään kolme kertaa sekunnissa...

Kuten kriteereistä huomaa, on siinä annettu poikkeuksia, jotka on mahdollista sisällyttää verkkopalveluun, ja siitä huolimatta saavuttaa saavutettavuusluokka A. Saavutettavuusluokka AAA samasta ruudun välähdyksistä kohdassa 2.3.2 sanoo seuraavaa:

> Verkkosivut eivät sisällä mitään, joka milloinkaan välähtäisi useammin kuin kolme kertaa sekunnissa.

Näitä kahta kriteeriä vertaamalla on helppo huomata saavutettavuusluokkien A ja AAA erot. Siinä missä saavutettavuusluokka A on huomattavasti joustavampi kriteerien suhteen, ja niissä on monesti tulkinnanvaraa, on luokan AAA kriteerien tulkinta erittäin yksiselitteistä. Tämä sama ero kriteerien tulkinnanvaraisuuden ja yksiselitteisyyden välillä saavutettavuusluokkia verratessa on läsnä melkein jokaisessa WCAG:n periaatteessa ja vaatimuksessa.

Jotta verkkopalvelun on mahdollista saavuttaa AAA-luokka, on sille asetettu erittäin tarkat ja yksiselitteiset vaatimukset, joiden noudattaminen vaatii usein suunnittelijalta sekä kehittäjältä selkeää kokonaiskuvaa saavutettavuudesta, jotta kriteerien täyttäminen ei tuntuisi ylitsepääsemättömältä. Mahdollisimman korkean saavutettavuusluokan kriteerien läpäisy johtaa verkkopalveluun, joka on käyttäjäkokemukseltaan helppo ja suoraviivainen. Tämä omalta osaltaan nostaa käyttäjämukavuutta, sekä vähentää mahdollisen tuen tarvetta verkkopalvelun käytössä, jolloin asiakaspalvelun tarve vähenee ja siihen varatut resurssit on mahdollista ottaa käyttöön muualle.

AAA-luokka on kuitenkin useimmille internetpalveluille erittäin vaikea saavuttaa, ellei jopa mahdotonta, sillä ne kaikki vaatimukset, jotka AAA-luokka pitää sisällään, häiritsevät sivuston suunnittelua sekä yleistä tyyliä suurissa määrin. Jopa WCAG:ssä itsessään tuodaan tämä esiin:

> Note 2: It is not recommended that Level AAA conformance be required as a general policy for entire sites because it is not possible to satisfy all Level AAA Success Criteria for some content.

Onkin hyvä muistaa, että sivusto on kokonaisuus, joka rakentuu niin saavutettavuudesta, käytettävyydestä kuin myös sen tyylistä, ja sisällöstä. Tämän vuoksi pyrkiminen AAA-luokkaan ei useinkaan ole se järkevin toimintamalli, sillä siinä usein kärsivät tyyli ja sisältö, jotka ovat isossa osassa nykyajan internetkokemusta. Tästä syystä AAA-luokan internetsivustot ovat harvassa, sillä usein tämän luokan saavuttaminen vaatii liikaa uhrauksia sivuston muilta tärkeiltä osa-alueilta, eikä sivuston käyttäjä usein tule edes huomaaman eroa AA-luokan sekä AAA-luokan sivustojen eroa.

4 Sovelluksen käyttöliittymän suunnitteluprosessi

Web-sovelluksessa käyttäjän kannalta keskeisimmässä osassa on käyttöliittymä. Käyttäjä käyttää sovellusta käyttöliittymän kautta, sillä vaikka web-sovellusten taustajärjestelmät usein tekevät hyvin ison osan sovelluksen toiminnallisuudesta nimenomaan kaiken ”taustalla”, käyttäjälle harvoin välittyy tietoa taustajärjestelmien sisällöstä. Tämän takia sovelluksen käyttöliittymän suunnittelu on erittäin tärkeää, ja se tulee tehdä huolella heti alusta alkaen.

4.1 Web-sovelluksen käyttöliittymän suunnittelusta

Suunnitellessa internetsivustoa, kuten web-sovellusta, ja etenkin sen visuaalista osiota, on erityisen tärkeä ylläpitää selkeää, yksiselitteistä sekä yhtenäistä ilmettä. Tämä auttaa etenkin uusia käyttäjiä hahmottamaan sivuston kokonaisuuden ilman, että heidän täytyy kuluttaa aikaa sivuston ymmärtämiseen. Sivuston on siis parasta vastata yleisen käyttäjän käsitystä siitä, miten internetsivuston tulisi toimia. (Sinkkonen ym. 2006, 109-110.)

Yleisimpinä ja tärkeimpinä internetsivuston suunnitteluun liittyvinä konsepteina voidaan pitää seuraavia (Sinkkonen ym. 2006, 110):

- tiedon esittämistapa: selkeyden priorisointi, jolloin lukija voi keskittyä tiedon sisältöön eikä sen esittämistapaan
- tiedon määrä: riittävä tasapaino jotta sivulla on mahdollisimman vähän mutta kuitenkin tarpeellinen määrä ymmärrettävyyden kannalta
- järjestys: looginen, jossa selkeä aloitus
- hierarkiat: sisällön painotus, jolla luodaan tärkeysjärjestys lukijalle
- rytmitys: käyttäjän katseen ohjaus sekä sivun sisäinen navigointi
- estetiikka: riittävästi niin sanottua “white space” -tilaa, jotta sivun sisältö saa hengittää ja on helpommin luettavaa, luo tasapainoa
- asioiden näkyvyys: varmistetaan visuaalisin vihjein, että käyttäjä huomaa tarpeellisen.

Nämä konseptit, sekä tämän insinöörityön aiemmin kohdassa 3.1 esitellyt käytettävyyden heuristiikat, ovat hyvä ohjenuora käyttöliittymäsuunnittelussa, joita seuraamalla ja pohtimalla luodaan edellytykset käytettävään sekä saavutettavaan sivustoon. Orjallisesti näitä, kuten luovalla alalla usein on, ei tarvitse seurata, mikäli suunnittelijalla on kyky

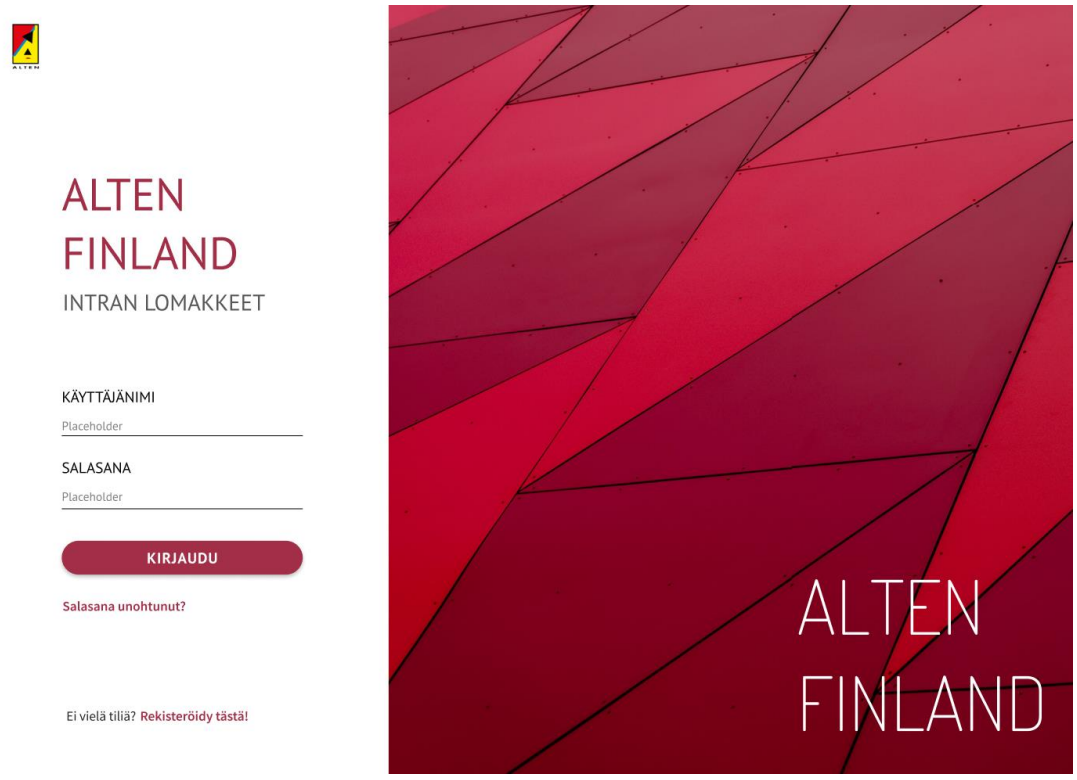
itsenäiseen laadun tuottamiseen. Visuaalisuuden ei kuitenkaan koskaan pidä mennä käytettävyyden ja saavutettavuuden edelle, sillä tämä heikentää käyttäjän halua palata sivustolle uudelleen. Koska tässä insinöörityössä kyseessä on yrityksen sisäinen sairauspoissaolonilmoitusjärjestelmä, ei käyttäjän palaamista tarvitse kyseenalaistaa, mutta tämänkaltaisenkin sivuston kohdalla hyvän käyttäjäkokemuksen takaaminen on tärkeää.

4.2 Kirjautumissivun suunnittelu

Kirjautumissivusto on usein käyttäjän ensikosketus sovellukseen, kuten tämänkin sovelluksen kohdalla. Sovelluksen kirjautumissivu on suunniteltu Nielsenin (1999, 2) eliminointiperiaatteen mukaisesti, mikä käytännössä tarkoittaa sivun elementtien läpikäyntiä testaten, onko elementti tarpeellinen, ja tuoko se minkäänlaista lisäarvoa käyttäjälle. Mikäli ei, voi sen harkintaa käyttäen poistaa. Tämänkaltainen lähestymistapa internetsivustoa suunnitellessa edesauttaa sivuston pitämistä kevyenä, niin visuaalisesti kuin myös latausaikoja arvioidessa.

Käyttäjät yleisesti ottaen kokevat sivuston toimivan viiveettömästi, mikäli vasteaika on maksimissaan 0,1 sekuntia (Nielsen 1999, 2). Harva sivusto tähän kykenee, etenkin sivustojen latautuessa, sillä tähän vaikuttaa liian moni tekijä, varsinkin käyttäjän internetyhteyden nopeus. Sivuston suunnittelua tehdessä hyvä periaate on pitää mielessä sivuston keveys, eli turhien elementtien eliminointi, jolla on iso vaikutus sivuston latausaikoihin. Myös elementtien yksinkertaisuus, eli niiden riippumattomuus suuresta määrästä koodia, tai riippumattomuus sivuston ulkopuolisiin resursseihin, kuten esimerkiksi kuviin, vaikuttaa koko sivuston latausnopeuteen ja sen kykyyn päästä mahdollisimman luettavaan ja ehyeen tilaan mahdollisimman lyhyessä ajassa.

Ensivaikutelma on tärkeä osa mitä tahansa sovellusta, sillä se luo käyttäjälle hyvin nopeasti vaikutelman siitä, minkälainen käyttökokemus hänellä tulee olemaan. On helpompi luoda hyvä ensivaikutelma kuin muuttaa huonoa myöhemmin. Värimaailmaksi tässä sovelluksessa on valittu tilaajayrityksen omia brändivärejä (kuva 5), sillä tämä luo käyttäjälle tutun ja turvallisen tunteen varsinkin tämänkaltaisen sovelluksen kohdalla.



Kuva 5. Sovelluksen kirjautumissivu on suunniteltu mahdollisimman kevyeksi ja käytettäväksi, jotta käyttäjän käyttökokemus on alusta lähtien mahdollisimman vaivaton ja nopea.

Tässä sovelluksessa käyttäjä tulee antamaan henkilökohtaisia tietoja, joten on tärkeää vakuuttaa käyttäjä tapahtumien luotettavuudesta. Värimaailma on käyttäjälle ennalta tuttu, sillä sovellusta käyttävät tilaajayrityksen omat työntekijät.

Visuaalisella hierarkialla käyttäjä ohjataan ensisijaisesti kirjautumaan sovellukseen sisälle, sillä tämä tulee olemaan kirjautumissivun ensisijainen toiminto. Otsikko sivulla kertoo, mikä yritys on kyseessä, ja alla oleva alaotsikko kertoo, mitä yrityksen sovellusta käyttäjä tulee käyttämään. Tämän tärkeän tiedon alla on käyttäjälle nostettu esille sivun tärkein toiminnallisuus, joka on käyttäjänimen sekä salasanan kirjoittaminen niille annettuihin kenttiin. On hyvä muistaa, että sivuston suunnittelijalle sivuston käyttötarkoitus on usein erittäin selkeä, mutta käyttäjälle (etenkin uudelle) on usein hyvä kertoa mahdollisimman selkeästi, missä hän on, mitä hän tulee tekemään ja miten. Sivu on siis hyvä suunnitella uudet käyttäjät mielessä ja pohtia, miten käyttäjä, joka ei tiedä, mihin on tulossa ja mitä on tekemässä, tulee selviytymään suunnitellun sivuston käytöstä. (Nielsen 1999, 4.)

Sivu on haluttu pitää yksinkertaisena, mutta se ei tarkoita sitä, että tärkeitä ominaisuuksia voi jättää pois. Edellä mainittujen otsikoiden ja kirjautumisosioiden alle on laitettu salasanan palautuslinkki, joka on erotettu muusta tekstistä erillisen kirjaisinkoon sekä värinsä kautta. Itse teksti on myös kirjoitettu kysymysmuotoon, joka jo itsessään kiinnittää käyttäjän huomion, sillä kysymys on suunnattu hänelle. Samaa periaatetta seuraa näiden osioiden alle suunniteltu osio, jossa käyttäjää kehotetaan luomaan uusi käyttäjätili, mikäli hänellä sitä ei ole. On myös hyvä huomioida, että yksinkertaisuuteen ja käytettävyyteen keskittyminen ei tarkoita visuaalisen ilmeen unohtamista, sillä etenkin, jos kyseessä on julkiseen käyttöön tuleva sovellus, on tärkeä luoda positiivinen sekä ammattimainen kuva sivustosta sen käyttäjille. Tämä usein tapahtuu modernilla ja yleisiä vallitsevia trendejä seuraavassa käyttöliittymässä.

Kaikki edellä mainittu on tehty siksi, että käyttäjän tulee kyetä havaitsemaan sivun kaikki käyttötarkoitukset nopeasti ja vaivattomasti. Jos käyttäjä ei tähän kykene, johtuu se yleensä huonosta käyttöliittymäsuunnittelusta, jossa väärät asiat vievät käyttäjän huomiokyvyn. Tämä johtaa siihen, että sivun käyttö ei hahmotu käyttäjälle oikein. Koska havaitseminen ei ole pelkkää aistimista, ei ole riittävää, että sivun käyttöliittymästä löytyy kaikki tarpeellinen. Tämän kaiken täytyy myös olla tunnistettavassa muodossa, sillä käyttäjillä on usein ennakkokäsityksiä siitä, mitä sivulla tulisi olla ja missä muodossa. Tähän kaikkeen voi vaikuttaa ärsykkeiden voimakkuudella, muodoilla ja modaliteetilla eli aistipiirillä. (Sinkkonen ym. 2006, 69-72.)

Sovelluksen mobiiliversio seuraa esimerkin (kuva 6) kaltaista yksinkertaistettua mallia, jossa on kuitenkin käytössä samat periaatteet, joita suuren ruudun versiossa on seurattu.

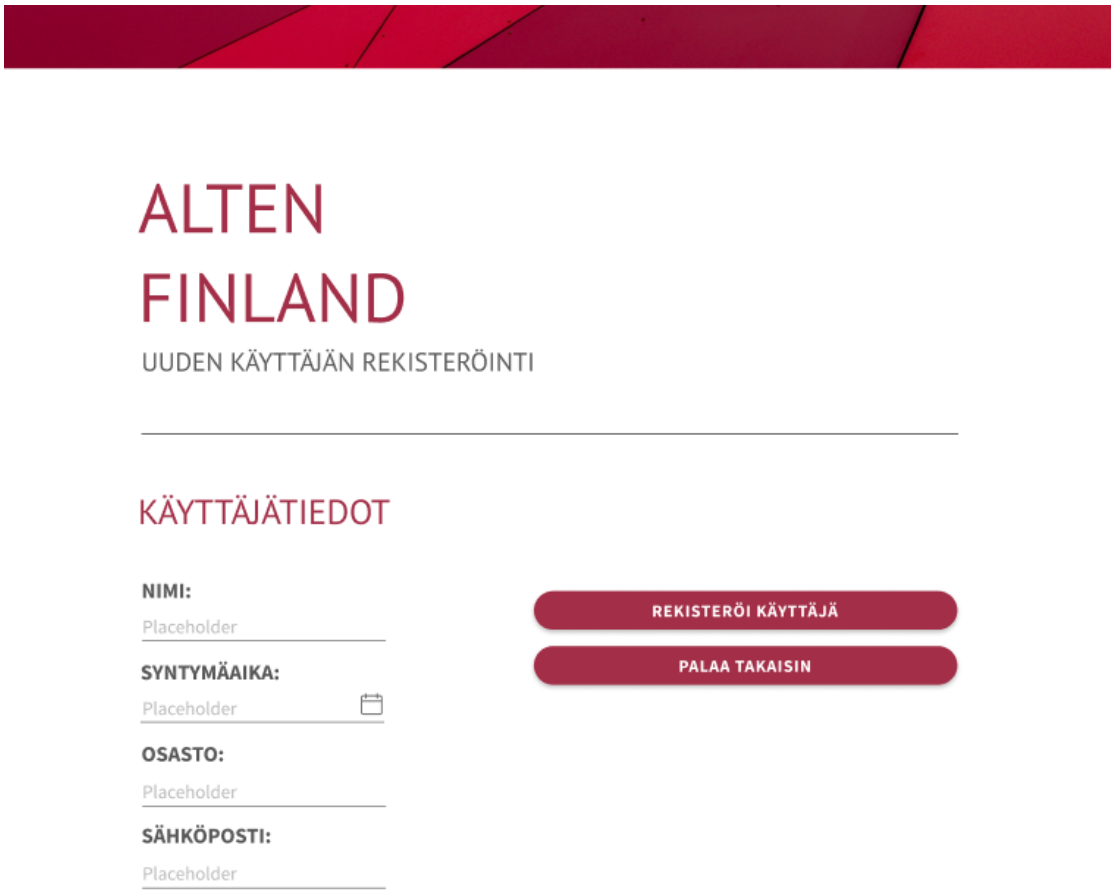
Kuva 6. Kirjautumissivuston mobiiliversio on riisuttu versio leveän näytön versiosta, mobiililaitteiden ruutujen kokorajoitteet sekä käyttörajoitteet ottaen huomioon. Suuri osa visuaalisista elementeistä, jotka ovat suurella ruudulla läsnä, on poistettu pienen ruudun versiossa, sillä ne häiritsisivät käyttökokemusta.

Yleisilmeeltään ja yhtenäisyydeltään mobiiliversio ei juurikaan eroa suurempien ruutujen versiosta, ja tämä on tarkoituksenmukaista. Sovelluksessa on haluttu säilyttää tuttu ja yhtenäinen ulkoasu ruudun koosta huolimatta, ja suurimpana muutoksena mobiiliversiota verratessa suuren ruudun versioon on sovelluksen niin sanottu riisutumpi estetiikka. Tähän riisuttuun ulkoasuun on päädytty niin käytännöllisistä syistä kuin myös käytettävyyden vuoksi, sillä pienelle ruudulle ei haluta mitään ylimääräistä heikentämään käyttökokemusta tai saavutettavuutta.

Tämä sama riisuttu ulkoasu on käytössä jokaisen sivun mobiiliversiossa, minkä vuoksi käyttöliittymän esimerkkikuvina on enimmäkseen käytetty suuren ruudun versioita, joissa variaatiota eri sovelluksen sivujen välillä on enemmän.

4.3 Uuden käyttäjän rekisteröinnin suunnittelu

Uuden käyttäjän rekisteröintiin tehty sivu (kuva 7) on kaikessa yksinkertaisuudessaan ainoastaan lomake kahden painikkeen kanssa. Myös käyttäjän ENTER-painike on toiminnallistettu lomakkeen lähettämiseen, sillä tätä ominaisuutta käyttäjät usein odottavat internetsivustojen lomakkeilta, ja painavat tätä painiketta olettaen siltä sen yleistä toimintoa.



Kuva 7. Uuden käyttäjän rekisteröintisivusto on mahdollisimman yksinkertainen ja riisuttu kaikesta ylimääräisestä, jotta käyttäjän ei tarvitse kuluttaa aikaa sivuun tutustumiseen.

Tällä yksinkertaisuudella on pyritty minimoimaan käyttäjän mahdollisuudet virheisiin ja vääriin valintoihin, mikä on saavutettu jo aiemmin mainitulla Nielsenin (1999, 2) ehdottamalla eliminointiperiaatteella. Eliminaatioperiaatteen mukaan sivusta poistetaan kaikki ylimääräinen, joka ei ole välttämätöntä käyttäjälle ja sivun toimivuudelle.

Uuden käyttäjän rekisteröintisivulla tulee selkeämmin myös esille hyvin suunniteltujen tiedonsyöttökenttien tärkeys sekä niiden ominaisuudet. Sivulla, jossa on syöttökenttiä, tulee olla selkeät ohjeet käyttäjältä odotetuista toiminnoista. Nyt tämä selkeys tulee visuaalista hierarkiaa hyväksikäyttävien syöttökenttien otsikoiden (label) kautta, jotka ovat normaalia tekstiä paksumpia sekä tummempia. Tämä syöttökenttien otsikoiden tyyli yhdistettynä itse syöttökenttien ohjaaviin teksteihin (placeholder text), jotka kertovat esimerkin kautta, mitä käyttäjältä odotetaan, luo kyseiselle käyttäjälle selkeät vaatimukset, jotka hänen on täytettävä sivustolla vieraillessaan.

Tämä kaikki voidaan saavuttaa ilman suurta määrää selitetekstiä, joka usein vie tilaa ja huomiota tärkeämmiltä asioilta. Vaikka selitteksteille on aikansa ja paikkansa, on hyvä pyrkiä suunnittelemaan sivuston elementit ja toiminnot tavalla, jotta ne ovat mahdollisimman itsestään selviä, jolloin suuria määriä selitetekstejä ei ole tarve luoda käyttäjän avuksi.

4.4 Käyttäjän etusivun suunnittelu

Käyttäjän etusivu, jota usein kutsutaan nimellä ”dashboard”, on tämän sovelluksen sisältörikkain sivu. Tällä sivulla käyttäjä näkee omat henkilökohtaiset tietonsa, omien sairauspoissaoloilmoituksiensa historian ja tilan, sekä myös painikkeet, joiden kautta käyttäjä pääsee luomaan uuden sairauspoissaoloilmoituksen sekä muokkaamaan omia käyttäjätietojaan (kuva 8).

Kuva 8. Käyttäjän etusivua voidaan pitää koko sovelluksen tärkeimpänä sivuna, sillä tällä sivulla tulee käyttäjä aina käymään mahdollisesti useammin kuin kerran jokaisella toimintakerrallaan.

Tältä sivulta käyttäjän tulee olla mahdollista päästä tekemään kaikki halutut toiminnot mahdollisimman vähällä vaivalla, minkä vuoksi oli tärkeää pitää sivun sisältö tiiviinä, mutta kuitenkin antamalla tarpeeksi tilaa, jotta toiminnot eivät huku toisiinsa. Sivulla on myös painike tapaturmailmoitukselle, joka on vailla toiminnallisuutta sovelluksessa, mutta joka on sijoitettu sivulle tulevaa jatkokehitystä varten.

Vaikka käyttäjätietojen tarkastelu tulee olemaan harvemmin tarpeellista, oli niiden sisällytys tälle sivulle pakollista. Mahdolliset epäkohdat näissä tärkeissä tiedoissa täytyy nousta esille mahdollisimman nopeasti ja selkeästi, jonka vuoksi käyttäjän etusivu oli ainoa looginen sijainti tälle tiedolle. Tämän lisäksi sivulle oli sisällytettävä painike, jonka kautta käyttäjä näitä tietoja pääsee yksinkertaisesti muokkaamaan.

Etusivun kaksi tärkeintä ominaisuutta ovat painike, joka johtaa sivulle, jolla käyttäjän on mahdollista tehdä sairauspoissaolon omailmoitus, sekä sivulla oleva sairauspoissaoloilmoitusten historiakomponentti. ”Omailmoitus sairauspoissaolosta” -painike on painikelistassa korkeimmalla juuri sen tärkeyden vuoksi. Käyttäjän ei pidä joutua etsimään tätä tärkeää ominaisuutta, vaan sen kuuluu sijaita paikassa, josta se on helppo ja nopea huomata.

Käyttäjän poissaoloilmoitusten historiakomponenttia voidaan pitää yhtä tärkeänä kuin niiden tekemiseen johtavaa painiketta. Käyttäjän suurin syy tulla sivulle sairauspoissaoloilmoituksen teon ohella, on aiempien ilmoitusten tilan seuranta, sillä nämä ilmoitukset tarvitsevat hyväksynnän niitä valvovalta taholta. Tämä ilmoitushistoriakomponentti sisältää ilmoituksen päivämäärän ilmoituksen tyypin sekä ilmoituksen tilan, joka voi olla “Hyväksytty”, “Odottaa hyväksyntää” tai “Hylätty” (kuva 9).

ILMOITUSHISTORIA

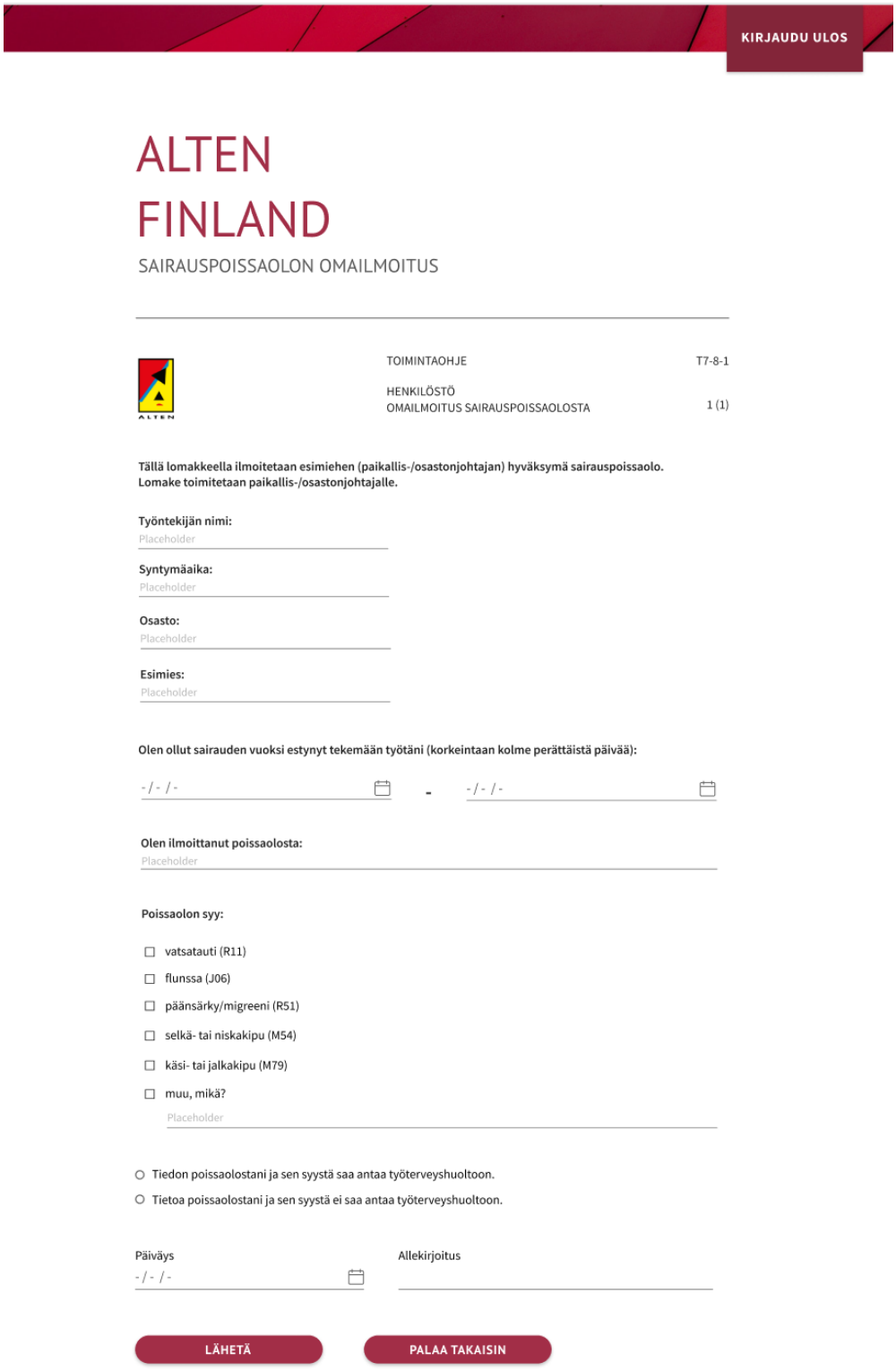
PVM	ILMOITUKSEN TYYPPI	TILA
28 / 01 / 2020	Omailmoitus sairauspoissaolosta	Odottaa hyväksyntää
30 / 11 / 2019	Omailmoitus sairauspoissaolosta	Hyväksytty
15 / 09 / 2019	Omailmoitus sairauspoissaolosta	Hyväksytty
08 / 08 / 2019	Omailmoitus sairauspoissaolosta	Hyväksytty
07 / 06 / 2019	Omailmoitus sairauspoissaolosta	Hyväksytty

Kuva 9. Ilmoitushistoriakomponentissa ilmoituksen tila kertoo käyttäjälle hänen ilmoitustensa mahdolliset tilamuutokset.

Ilmoitushistoria-komponentissa on käytetty sovelluksen väriteemaa sillä poikkeuksella, että joka toinen ilmoitus on harmaalla pohjalla. Tämä edesauttaa rivien lukemista, ja nostaa niiden mahdolliset eroavaisuudet tehokkaammin esille. Tilamuutoksissa on myös käytetty hyväksi värejä. Ilmoituksen tilan ollessa jotain muuta kuin “Hyväksytty”, on sen väri muuta tekstiä tummempi, jotta se on helpompi havainnoida, ja mahdolliset muutokset niin sanotusti hyppäävät käyttäjän silmille.

4.5 Uuden sairauspoissaoloilmoitussivun suunnittelu

Uuden sairauspoissaoloilmoitussivun lomakkeen suunnittelussa haastavaa oli suuri tiedon määrä, jota uuden ilmoituksen tekoon vaaditaan. Lomakkeelta karsittiin kaikki ylimääräinen, mutta siitäkin huolimatta on tämä sivu koko sovelluksen pisin (kuva 10).



KIRJAUDU ULOS

ALTEN
FINLAND

SAIRAUSPOISSAOLON OMAILMOITUS

TOIMINTAOHJE T7-8-1

HENKILÖSTÖ
OMAILMOITUS SAIRAUSPOISSAOLOSTA 1 (1)

Tällä lomakkeella ilmoitetaan esimiehen (paikallis-/osastonjohtajan) hyväksymä sairauspoissaolo. Lomake toimitetaan paikallis-/osastonjohtajalle.

Työntekijän nimi:
Placeholder

Syntymäaika:
Placeholder

Osasto:
Placeholder

Esimies:
Placeholder

Olen ollut sairauden vuoksi estynyt tekemään työtäni (korkeintaan kolme perättäistä päivää):

-/-/- - -/-/-

Olen ilmoittanut poissaolosta:
Placeholder

Poissaolon syy:

- ☐ vatsatauti (R11)
- ☐ flunssa (J06)
- ☐ päänsärky/migreeni (R51)
- ☐ selkä- tai niskakipu (M54)
- ☐ käsi- tai jalkakipu (M79)
- ☐ muu, mikä?

Placeholder

- ○ Tiedon poissaolostani ja sen syystä saa antaa työterveyshuoltoon.
- ○ Tietoa poissaolostani ja sen syystä ei saa antaa työterveyshuoltoon.

Päiväys -/-/- Allekirjoitus

LÄHETÄ PALAA TAKAISIN

Kuva 10. Uuden sairauspoissaoloilmoituksen sivun pituus on haastava käyttäjälle, mutta kysyttävän tiedon määrää ei ole mahdollista karsia liikaa, jotta kaikki tarvittava saadaan talteen.

Pitkien sivujen ongelma käytettävyyden kannalta on niiden luomat haasteet pienillä ruuduilla. Pitkä sivu johtaa tarpeeseen rullata sivua paljon alaspäin, jotta käyttäjä näkee kaiken sivulla olevan tiedon. Tämän vuoksi suurilla ruuduilla osa kysyttävistä tiedoista ja niihin liittyvät syöttökentät on sijoitettu vierekkäin, sillä suuren ruudun tilaa on tässä tapauksessa hyvä käyttää tehokkaasti.

Pienillä ruuduilla, kuten mobiililaitteilla, ovat kaikki tiedot allekkain ruudun kapeuden vuoksi. Suurin osa puhelimista on leveimmillään vain 480 pikseliä leveitä (StatCounter), jonka vuoksi informaatio on välttämätöntä sijoittaa allekkain. Tämä allekkain sijoittaminen edesauttaa ruudulla olevan tekstin sekä muun sisällön luettavuutta. Tällöin käyttäjän ei tarvitse rullata ruutua horisontaalisesti, sillä yleisesti ottaen tämä vaikuttaa negatiivisesti käyttäjäkokemukseen.

Sivun luettavuutta on pyritty, suuresta tekstin määrästä huolimatta, ylläpitämään vahvan visuaalisen hierarkian avulla. Tämän avulla on eritelty eri tiedonsyöttöosiot toisistaan, oli ruudun koko mikä hyvänsä. Luettavuutta on vahvistettu myös jättämällä tiedonsyöttöosioiden väliin riittävästi tilaa, ja vaikka tämä johtaakin sivun pituuden kasvuun, on se käyttäjäkokemuksen kannalta tärkeä ratkaisu.

Tärkeä osa sivun suunnittelua oli myös ylläpitää kysyttävän tiedon semanttista järjestystä, sillä tämä luo käyttäjälle paremman kokonaiskuvan sivusta ja siinä kysyttävästä tiedosta. Semanttisella järjestyksellä tarkoitetaan sitä, miten jokin asia tulisi niin sanotusti luonnollisesti mieleen, ja tämän järjestyksen noudattaminen on tärkeää mahdollisimman tehokkaan lomakkeen täytön kannalta. Sivun loogisuus vähentää käyttäjän hämmennystä ja ylläpitää käyttäjän tiedon tarpeen luonnollista rytmiä (Sinkkonen ym. 2006, 162-166), joka omalta osaltaan on erittäin tärkeä osa-alue sivuston kokonaiskäytettävyyttä arvioidessa.

4.6 Suunnittelun yhteenveto

Suunnitellussa ja toteutuksessa on pyritty hyvien suunnittelu-, käytettävyys- ja saavutettavuuskäytäntöjen mukaisesti selkeyteen ja yksinkertaisuuteen. Käytännössä tämä tarkoittaa monen asian yhdistelmää. Esimerkkinä on kaksi sivuston painiketta, ”lähetä pyyntö” sekä ”palaa takaisin” (kuva 11).

Kuva 11. Jotta käyttäjä ei ajaudu tilanteeseen, jossa hän on epävarma ja jossa hän voi käyttää sovellusta virheellisesti (vastoin suunnittelijan tarkoitusta), on esimerkkipainikkeissa selkokielellä ilmaistu, mitä tätä painiketta painaessa tapahtuu.

Sen sijaan, että painikkeiden tekstit olisivat vain yhden sanan mittaiset (”lähetä” ja ”palaa”), toiminnallisuutta paremmin kuvaavilla painiketeksteillä vähennetään käyttäjän epävarmuutta sekä virhetoimintoja. Useimmissa sovelluksen sivuissa on sama periaate, eli käyttäjävirheiden mahdollisuudet on minimoitu, ja sivujen toiminnot luotu mahdollisimman selkeästi värimaailman sekä typografian kautta. Nämä yhdistettynä selkeään tekstiin, minimoi käyttäjän mahdollisuudet tehdä vääriä valintoja sivustoa käyttäessä ja navigoidessa.

Yksinkertaistettuna käyttäjän vuorovaikutus käyttöliittymän kanssa perustuu siihen, että käyttäjä osaa tulkita sivuston suunnittelijan niin sanottua merkkikieltä. Sovelluksen ollessa GUI (Graphical User Interface) -järjestelmä, käyttäjät ovat tottuneet ja oppineet tietynlaiseen symboliikkaan, jossa visuaaliset vihjeet ja merkit tarkoittavat tietyntyyppistä toimintamahdollisuutta, ja näiden perusteella käyttäjä toimii totuttuun tapaansa. Jos tätä käyttäjälle tuttua symboliikkaa lähdetään muuttamaan ja uudistamaan, on riskinä aina käyttäjäkokemuksen heikentäminen. Tämä usein luo suuren haasteen suunnittelijalle, jolta odotetaan modernia, mutta hyvin käytettävää käyttöliittymää. Tämän vuoksi on tärkeä ymmärtää käyttöliittymäsuunnittelun perusperiaatteet, jotka ovat vuosien saatossa

muuttuneet huomattavan vähän, ja ovat hyvä lähtökohta sivuston suunnittelussa (Sinkkonen ym. 2006, 109-114).

Tämän web-sovelluksen suunnittelussa pyrkimyksenä oli rakentaa selkeä vuorovaikutus käyttäjän ja käyttöliittymän välille. Tämä tapahtuu usein käyttämällä niin sanottuja toimintatiloja sivuston osioissa, joiden kanssa käyttäjän on mahdollista vuorovaikuttaa (kuva 12), ja näillä edistetään sivuston käyttöä.

Kuva 12. Verrattuna kuvaan 11: tässä ylempi painike on valittuna esimerkiksi hiiren kohdistimella. On helppo havaita, miten yksinkertaisilla värien muutoksilla käyttäjälle on mahdollista luoda selkeä visuaalinen vastike, jolloin käyttäjälle ei jää epäselväksi, onko toiminto tapahtumassa vai ei.

Verratessa kuvia 11 ja 12 on helppo huomata niiden ero, sillä kuvassa 12 käyttäjä on valinnut “lähetä pyyntö” -painikkeen, ja tämän takia painikkeen värit ovat vaihtuneet. Teksti on muuttunut edeltäneen tilan taustaväriksi, sekä taustaväri on muuttunut edeltäneen tilan tekstin väriksi, jonka lisäksi painikkeen ympärille on luotu kapea rajaus, jotta se erottuu selkeämmin sivuston valkoisesta taustasta. Usein yllä olevan kaltaiset yksinkertaiset muutokset ovat parhaita, sillä niissä pysytään sivuston perusväreissä eikä luoda suuria muutoksia, joilla on riski hämmentää käyttäjää, sekä vähentää käytettävyyttä sekä käyttömukavuutta.

Saavutettavuuden kannalta on tärkeää, että tämänkaltaiset toimintatilojen muutokset, jotka tapahtuvat käyttäjän valinnan kautta, tapahtuvat käyttäjän valintatavasta riippumatta. Käytännössä tällä tarkoitetaan sitä, että sivuston vastikkeen on toimittava samalla tavoin ja näyttää samalta, valittiin sivuston elementti hiiren kursorilla, näppäimistön tabulaattori (TAB) -painiketta käyttäen tai esimerkiksi käyttäen mobiililaitetta, jossa ei yleensä ole hiirtä tai näppäimistöä käytettävissä, ja jossa toiminnot tapahtuvat kosketusnäytön kautta.

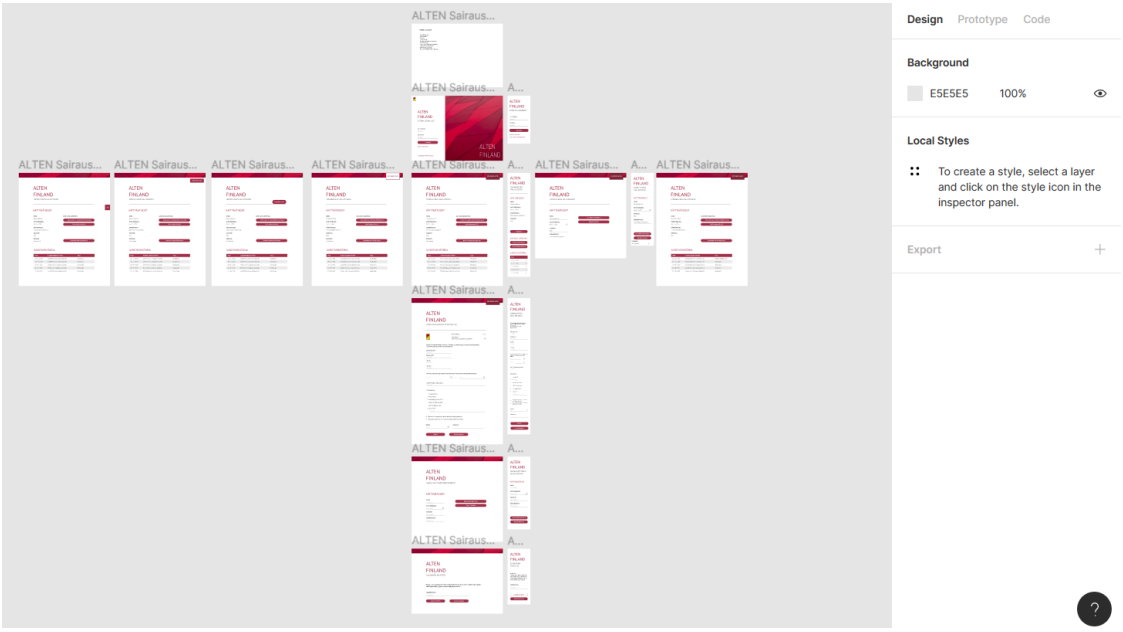
5 Sovelluksen käyttöliittymän prototyyppi ja sen käytettävyystestaus

Käyttöliittymän suunnittelun jälkeen haluttiin varmistua siitä, että suunnitelman mukainen web-sovellus olisi käytettävyydeltään hyvä. Käytettävyystestaus on hyvä tapa varmistua tuotteen käytettävyydestä, ja sen avulla saadaan usein esiin kehitysehdotuksia, joiden perusteella tuotetta tai sen suunnitelmaa voidaan kehittää vielä paremmaksi. Käytettävyystestausta varten tarvittava prototyyppi luotiin käyttöliittymän suunnitelman perusteella.

5.1 Käyttöliittymän prototyyppi

Tämän sovelluksen käyttöliittymän prototyyppi tehtiin luomalla web-sovelluksen kaikille osioille (sivuille ja niiden elementeille) niin sanottu ”layout”, ulkoasusuunnitelma. Se on usein 1:1-suhteen suunnitelma, joka sisältää esimerkin sivusta ja sen sisältämästä sisällöstä kertoen tarkalleen, mitä värejä sekä minkä kokoista fonttia käytetään, mikä typografia on sivuston eri elementeille valittu sekä muut tarvittavat yksityiskohdat, joiden perusteella ohjelmistokehittäjä voi lopullisen käyttöliittymän luoda. Usein, kuten tämän insinöörityön sovelluksen prototyypin kohdalla, luodaan sivuston eri aktiivisuustiloille selkeät suunnitelmat, kuten myös sille, miten sivusto toimii eri toimintojen kohdalla.

Prototyypin tekoon valittiin Figma-niminen suunnitteluohjelma, jolla luotiin koko sovelluksen käyttöliittymän graafinen suunnitelma (kuva 13). Yksi tärkein syy Figman valintaan oli sen mahdollistama yhteistyö monen suunnittelijan kesken, sillä Figman prototyypit on mahdollista jakaa pilvipalveluissa reaaliaikaisesti. Tällöin useamman henkilön on mahdollista työstää samaa prototyyppiä yhtä aikaa.



Kuva 13. Figma -suunnitteluohjelmassa luodaan sovelluksesta graafinen suunnitelma, joka toimii lopullisen ohjelman esimerkkinä. Tämän perusteella ohjelmistokehittäjä luo itse ohjelmiston koodin, käyttäen prototyypistä löytyvää tietoa apunaan.

Toinen tärkeä syy Figman valintaan oli sen interaktiivinen prototyypin käyttö, mikä käytännössä tarkoittaa mahdollisuutta jakaa prototyyppi julkisesti linkin kautta, jolloin sitä on mahdollista näyttää ja testauttaa vaivattomasti. Interaktiivisessa prototyypissä leiskojen elementteihin on mahdollista luoda interaktiivisuutta, jolloin suunnittelija voi esimerkiksi luoda kirjautumissivun, jossa kirjautumispainiketta painamalla prototyypin testaaja voi edetä seuraavalle sivulle.

Tästä prototyypistä sekä etenkin sen interaktiivisuudesta on suuri hyöty käyttöliittymätestauksessa. Käytettävyystestaajat pääsevät navigoimaan prototyyppiä, joka imitoi normaalin sivuston toimintoja, ja näin saadaan oleellista palautetta sovelluksen toiminnallisuudesta jo suunnitteluvaiheessa. Palautteen perusteella käyttöliittymäsuunnitelmaa voidaan muokata yhä käyttäjäystävällisemmäksi lopulliseen prototyyppiin, jonka perusteella itse sovellus rakennetaan.

5.2 Käytettävyystestaus

Hyysalon (2009, 12) mukaan ”käyttäjiä ja käyttöä koskeva tiedonkeruu on yksi tuotekehityksen avaintaidoista”. Jos tuotteen käytön suunnitteluun panostetaan, voidaan välttää toiminnallisia epäonnistumisia. Yksi hyväksi havaittu keino tuotteen käytettävyyden tarkasteluun on käytettävyystestaus. Sinkkosen ym. (2009, 302) mukaan käytettävyystestauksella pyritään selvittämään se, kuinka tuotetta ensimmäistä kertaa tai harvoin käyttävä selviää tuotteen käytöstä.

Käytettävyystestauksella tarkoitetaan tuotteen käytettävyyden selvittämistä käyttäjien avulla. Testauksen avulla voidaan selvittää esimerkiksi, miten käyttäjät hahmottavat tuotteen toiminnan, miten käyttäjät suoriutuvat tehtävistään käyttäessään tuotetta ja ymmärtävätkö käyttäjät tuotteen käyttämisen niin kuin tuotteen suunnittelijat tarkoittavat. Käytettävyystestauksia tekemällä pyritään saamaan esiin tuotteen muutostarpeita. (Hyysalo 2009, 164.) Sinkkosen ym. (2009, 297) mukaan kehittämisprosessit muuttuvat sitä kalliimmiksi, mitä myöhemmin virheet havaitaan ja saadaan korjattua.

Käytettävyystestaus on yleistä tuotekehityksessä, ja testejä voidaan suorittaa niin paperiprototyypeillä kuin toimivilla tuotteillakin. Testejä on helppo toteuttaa ja niissä voidaan tarkkailla suoraan käyttäjien toimintaa. (Hyysalo 2009, 164.) Esimerkiksi alansa ammattilaisten tekemät käyttöliittymät on hyvä testata aloittelijoilla, sillä ammattilainen sokeutuu omalle työlleen helposti eikä näin ollen näe, mikä käyttöliittymän käytössä on aloittelijalle epäselvää. (Sinkkonen ym. 2009, 297.)

Sinkkosen ym. (2009) mukaan käytettävyystestausta on kahdentyyppistä:

- testit, jotka ovat osa kehitystyötä
- testit, joissa mitataan, onko tuote käytettävyydeltään ja käyttäjäkokemukseltaan levitykseen hyväksyttäviä.

Käytettävyystestaus tehdään todellisilla käyttäjillä, jotka tekevät oikeita tehtäviä testattavalla tuotteella mahdollisimman oikean kaltaisessa ympäristössä. Käytettävyystestauksella pyritään havaitsemaan tuotteen käytön ongelmakohdat sekä tarkkaillaan sitä, kuinka hyvin se toimii käytännössä. (Sinkkonen ym. 2009, 299.)

Testattavaksi voidaan valita koko tuote, sen prototyyppi tai vaikka tuotteen keskeiset toiminnot. Käytettävyystestausta tulisi tehdä koko tuotekehityksen ajan, sillä monen pienen testin on havaittu olevan hyödyllisempää kuin yhden ison testin. Tällöin mahdollisuus siihen, että testikäyttäjät kiinnittävät huomiota epäolennaisiin asioihin, pienenee. (Sinkkonen ym. 2009, 299-300.)

Itse testaustilanne on yleensä rakenteeltaan seuraavanlainen (Sinkkonen ym. 2009, 306):

- testaustilanteen selvittäminen testikäyttäjälle
- alkukysely tai -haastattelu
- testitehtävien tekeminen
- loppukysely tai -haastattelu.

Testikäyttäjille tulee kertoa, että testin kohteena on tuote, ei käyttäjä itse, ja että hän voi kysyä testin aikana vapaasti kysymyksiä. Alkukyselyn avulla selvitetään käyttäjien taustaa, ja varsinkin osaamista testin kohdealueilta. Loppuhaastattelussa pyritään selvittämään käyttäjän tunnelmat testin jälkeen, ja varsinkin valmista tuotetta testatessa pyritään saada selville yksityiskohtia testikäyttäjän käyttökokemuksesta. (Sinkkonen ym. 2009, 306-307.)

Sinkkonen ym. (2009, 303-305) mukaan käytettävyystestausta suunnitellessa tulee käydä läpi seuraavat vaiheet ja kirjata ne ylös testaussuunnitelmaan:

- testauksen tavoitteiden selvittäminen
- käyttäjäryhmien selvittäminen
- testattavien toimintojen valinta
- testitarinan- ja tehtävien laatiminen
- testauspaikan valinta
- testausmenetelmän valinta
- mahdollinen interaktiivinen tilanne
- pilottitestin järjestäminen.

Käytettävyystestausta suunnitellessa määritellään ensin tavoitteet, joihin testillä pyritään, sekä tuotteen kohderyhmä ja käyttäjäprofiili. Kun päätetään testikäyttäjien määrää, tulee ottaa huomioon, että jokaisesta käyttäjäryhmästä tulee löytää testikäyttäjiä. Käyttäjäryhmien ollessa keskenään melkein samanlaisia, suositellaan testi tehtäväksi ainakin viidelle testikäyttäjälle. (Sinkkonen ym. 2009, 303.)

Tämän jälkeen määritellään tärkeimmät asiat, jotka käyttäjien tulisi tuotteella (tai sen osalla) pystyä tekemään. Näiden määritysten perusteella tehdään testitarina, jossa kuvattuja tehtäviä testikäyttäjät tekevät käytettävyystestissä. Tehtävät tulisi kuvata tarinassa niin, että ne tuottaisivat toimintaa, mutta eivät johdattelisi ennalta oletettuun lopputulokseen. (Hyysalo 2009, 165, 173.) Hyvä testitarina on lyhyt, kuvaa käyttäjien arkitai työelämää ja kuljettaa käyttäjän testitehtävien läpi. Tarina kirjoitetaan kerronnalliseksi: ”Olet iltavuorossa kassalla ja sinulle tulee asiakas, joka...”. Mikäli testitilanteeseen tarvitaan interaktiivinen tilanne, esimerkiksi asiakaspuhelu, tulee suunnitella, kuinka tämä suoritetaan. (Sinkkonen ym. 2009, 304-305.)

Testauspaikkaa valitessa tulisi pyrkiä simuloimaan mahdollisimman tarkasti tuotteen oikeaa käyttöympäristöä. Tämä auttaa käyttäjiä asettumaan tuotetta oikeasti käyttävän henkilön rooliin ja siten takaa paremman ennusteen tuotteen suorituskyvystä oikeissa käyttöolosuhteissa. (Ruben & Chisnell 2003, 87.)

Käytettävyystestausta suunnitellessa tulee päättää, mitä eri tutkimusmenetelmiä kyseisessä testissä käytetään tiedon keräämiseksi. Käyttäjien kanssa tehtävien arvioiden tiedonkeruumenetelmiä ovat muun muassa ääneen ajattelu, paritestaus, mittaaminen, havainnointi, kuvanauhahaastattelu sekä jälkikäteen haastattelu. Jälkikäteen haastattelu on yleinen tapa päättää käytettävyystesti. (Hyysalo 2009, 175-176.)

Yhden käyttäjän käytettävyystestauksen pituus on tyypillisesti 1-2 tuntia, mutta se voi myös kestää vain muutamasta minuutista kokonaiseen päivään. (Sinkkonen ym. 2009, 299.) Hyysalo (2009, 173) suosittelee, että käytettävyystestauksen yksittäinen tehtävä olisi kestoltaan aina noin 2-20 minuuttia, sillä sen pituiset tehtävät ovat selkeämpiä analysoida.

Kun kaikki osa-alueet on käyty läpi ja kirjattu testaussuunnitelmaan, suoritetaan pilottitestaus. Pilottitestauksessa käytettävyystestaus suoritetaan pilottitestaajalla. Pilottitestaajan tulisi osaamistasoltaan muistuttaa oikeiden testikäyttäjien profiilia. Pilottitestin avulla tarkistetaan esimerkiksi tekniikan toimiminen, testaukseen kuluva aika sekä tehtävien ja haastattelukysymysten selkeys. Pilottitestissä huomatut puutteet ja ongelmat korjataan lopullista testausta varten suunnitelmaan. (Sinkkonen ym. 2009, 305.)

5.3 Sovelluksen prototyypin käytettävyystestauksen suunnittelu ja toteutus

Kun prototyyppi oli valmis, aloitettiin suunnittelemaan käytettävyystestausta. Tämän käytettävyystestauksen tarkoituksena oli selvittää, kuinka helppokäyttöinen sovelluksesta tulisi, mikäli se rakennettaisiin prototyypin perusteella. Tavoitteena on rakentaa mahdollisimman korkean käytettävyyden ja saavutettavuuden omaava sovellus, joten testaus rakennettiin tämän tavoitteen ympärille.

Sovelluksen käyttäjäryhmänä ovat kaikki tilaajan yrityksessä työskentelevät henkilöt, jotka voivat tulla kaikenlaisista taustoista ja joiden tietotekniset taidot voivat olla matalasta korkeaan tasoon. Tämän sovelluksen käyttäjäryhmään kuuluu siis käytännössä kaikki työikäiset henkilöt, ja tähän käytettävyystestaukseen käyttäjäryhmäksi valittiin tilaajan työntekijät Helsingin toimipisteestä. Testikäyttäjiä testaukseen saatiin seitsemän kappaletta. Ennen varsinaisen käytettävyystestauksen alkua suoritettiin pilottitesti käyttäjäryhmään kuuluvalla testikäyttäjällä, jotta varmistuttiin, että tehtävät olivat helposti ymmärrettäviä ja että testaus pystyttiin viemään läpi suunnitelman mukaisesti.

Testauspaikaksi valittiin tilaajan toimistotila, jossa oli mahdollisuus saada käyttöön hiljainen huone. Koska usein sairastunut henkilö tekee sairauspoissaoloilmoituksen kotoaan käsin, pyrittiin testaustila muokkaamaan mahdollisimman kodinomaiseksi ympäristöksi: testikäyttäjä sai suorittaa testin istuen sohvalla kannettavan tietokoneen sijaitessa pienellä työtasolla sohvan ääressä.

Käytettävyystestauksessa haluttiin tarkastella kaikkia suunnitellun sovelluksen osia, joten testitarina rakennettiin sen mukaisesti. Testitarina ja -tehtävät olivat seuraavanlaiset:

“Olet sairastunut, ja sinun tulee tehdä omailmoitus sairauspoissaolosta sähköisesti esimiehellesi yrityksen omassa järjestelmässä. Et ole käyttänyt järjestelmää aiemmin, joten sinun tulee rekisteröityä järjestelmään.

Rekisteröitymisen jälkeen täytä kyseinen lomake omilla tiedoillasi. Lomaketta täyttäessä huomaat, että olet täyttänyt syntymäaikasi väärin. Mene muuttamaan syntymäaikasi käyttäjätietoihisi, jonka jälkeen palaa täyttämään sairauspoissaoloilmoitus. Tallenna ilmoitus, ja tarkista että ilmoitus ilmestyi ilmoitushistoriaan. Tämän jälkeen kirjaudu ulos järjestelmästä.”

Tiivistettynä:

- *Rekisteröidy.*
- *Ala täyttää lomaketta.*
- *Siirry muuttamaan syntymäaika käyttäjätietoihin.*
- *Täytä lomake ja tallenna.*
- *Tarkista, että ilmoitus on ilmoitushistoriassa.*
- *Kirjaudu ulos.*”

Testaustilanteessa oli paikalla kaksi testausta ohjaavaa henkilöä sekä yksi testikäyttäjä. Testaus suoritettiin kannettavalla tietokoneella, jossa oli valmiiksi avattuna sovelluksen prototyyppi. Pöydällä tietokoneen viereen asetettiin testitarina ja -tehtävät paperille tulostettuna, jotta testikäyttäjä pystyi rauhassa tarkistamaan seuraavan tehtäväkohdan, mikäli sille tuli tarve. Testikäyttäjiä ohjeistettiin myös kysymään apua, mikäli jonkin tehtävän ymmärtämisessä oli haasteita. Ohjaavat henkilöt pyrkivät selittämään tehtävät niin, että ne tulevat ymmärretyiksi, mutta kuitenkin johdattelematta testikäyttäjää toimimaan jollain tietyllä tavalla.

Käytettävyystestauksen aikana ohjaavat henkilöt havainnoivat testikäyttäjän toimintoja ja kirjaavat ylös muistiinpanoja testauksen kulusta. Testikäyttäjää myös pyydettiin ajattelemaan ääneen, eli kertomaan mitä hän testaustilanteessa teki ja ajatteli. Tämän lisäksi testikäyttäjä täytti testauksen lopuksi kyselylomakkeen (liite 1), jossa kyseltiin hänen kokemuksiaan sovelluksen käytöstä käytettävyystestauksen aikana. Kyselylomake koostui monivalintakysymyksistä sekä neljästä avoimesta kysymyksestä. Monivalintakysymyk-

sissä käytettiin SUS (System Usability Scale) -menetelmää. Sen mukaan laaditaan kymmenen väittämää, joihin käyttäjä vastaa valitsemalla asteikolla 1-5 olevan numeron oman mielipiteensä mukaisesti. Numero 1 tarkoittaa, että käyttäjä on täysin eri mieltä väittämän kanssa, ja numero 5 tarkoittaa, että käyttäjä on täysin samaa mieltä väittämän kanssa. Tämän jälkeen lasketaan käytettävyyden kokonaisarvo ennalta määrätyn laskentakaavan mukaisesti. Testaustapana SUS-menetelmä toimii niin isoille kuin pienille testausryhmille. (Usability.gov.)

5.4 Sovelluksen käytettävyystestauksen tulokset ja analysointi

Käytettävyystestaukseen osallistui seitsemän käyttäjäryhmään kuuluvaa testikäyttäjää. Testikäyttäjistä kaikki työskentelevät tietokoneella joka päivä, joten heillä on perusosaaminen tietokoneohjelmistojen käytöstä. Jokainen yksittäinen käytettävyystestaus saatiin vietyä ongelmitta läpi, eivätkä testikäyttäjät tarvinneet suuresti apua tehtävistä suoriutumiseen tai sovelluksen käytön ymmärtämiseen. Sovelluksen prototyyppi toimi testauksissa suunnitelman mukaisesti.

Kaikki testikäyttäjät löysivät helposti rekisteröitymissivun, osasivat täyttää vaaditut tiedot sekä siirtyä sisäänkirjautumiseen nopeasti. Myös siirtyminen sairauspoissaoloilmoituksen täyttämiseen tapahtui sujuvasti jokaiselta käyttäjältä. Yksi testikäyttäjä jumiutui hetkeksi tehtävässä, jossa tuli palata lomakkeen täytöstä takaisinpäin ja siirtyä muuttamaan käyttäjätunnuksen syntymäajan arvo, mutta pienen hetken kuluttua tämäkin testikäyttäjä löysi lomakesivun ”Palaa takaisin” -painikkeen. Syntymäajan muutos, lomakkeeseen paluu ja sen tietojen täyttäminen sekä lähettäminen onnistuivat jokaiselta käyttäjältä nopeasti ja sujuvasti. Kahden viimeisen tehtävän kohdalla ilmeni kuitenkin kolmella testikäyttäjistä ongelmia: ilmoitushistoriasta uuden ilmoituksen löytymisen tarkistaminen vei 5-10 sekuntia aikaa ja vaati testikäyttäjiä rullaamaan sivua alaspäin, ja ”Kirjaudu ulos” -painikkeen löytäminen vei myös useamman (5-10 s) sekunnin aikaa.

Testaustilanteen jälkeen testikäyttäjät täyttivät kyselylomakkeen, jossa he vastasivat SUS-menetelmän mukaiseen kymmeneen väittämään sekä neljään avoimeen kysymykseen. Testikäyttäjien vastaukset kymmeneen väittämään (taulukko 1) olivat hyvin samankaltaisia, lähes kaikissa kysymyksissä suurin osa testikäyttäjistä valitsi saman vaihtoehdon.

Taulukko 1. Kyselylomakkeen 10 väittämää ja testikäyttäjien vastausjakaumat.

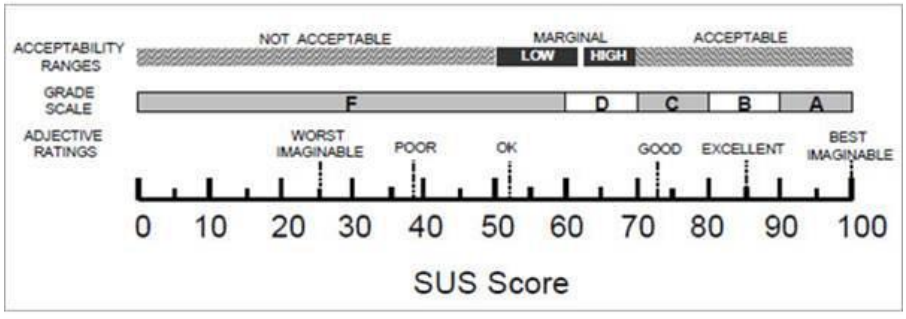
Kysymykset	**1** Täysin eri mieltä	**2**	**3**	**4**	**5** Täysin samaa mieltä
1. Luulen, että käyttäisin tätä sovellusta mielelläni	0 (0.0%)	0 (0.0%)	0 (0.0%)	2 (28.6%)	5 (71.4%)
2. Mielestäni sovellus oli tarpeettoman monimutkainen	5 (71.4%)	2 (28.6%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
3. Mielestäni sovellusta oli helppo käyttää	0 (0.0%)	0 (0.0%)	0 (0.0%)	4 (57.1%)	3 (42.9%)
4. Luulen, että tarvitsen teknisen henkilön tukea, jotta osaisin käyttää tätä sovellusta	7 (100%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
5. Mielestäni sovelluksen eri osat toimivat hyvin yhteen	0 (0.0%)	1 (14.3%)	0 (0.0%)	2 (28.6%)	4 (57.1%)
6. Mielestäni sovelluksessa on liian paljon eri lailla toimivia asioita	5 (71.4%)	1 (14.3%)	1 (14.3%)	0 (0.0%)	0 (0.0%)
7. Luulen, että useimmat ihmiset oppivat sovelluksen käytön erittäin nopeasti	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (14.3%)	6 (85.7.%)
8. Mielestäni sovelluksen käyttö oli hyvin hankalaa	6 (85.7%)	1 (14.3%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
9. Tunsin itseni hyvin varmaksi, kun käytin sovellusta	0 (0.0%)	0 (0.0%)	0 (0.0%)	4 (57.1%)	3 (42.9%)
10. Minun piti opetella paljon asioita, ennen kuin sovelluksen käyttö alkoi sujua	5 (71.4%)	2 (28.6%)	0 (0.0%)	0 (0.0%)	0 (0.0%)

Viisi testikäyttäjää oli täysin samaa mieltä väittämän “Luulen, että käyttäisin tätä sovellusta mielelläni” kanssa, valiten asteikolta arvon 5. Kaksi muuta valitsivat arvon 4, joka tarkoittaa, että he ovat lähes samaa mieltä väittämän kanssa. Seuraavaan väittämään, “mielestäni sovellus oli tarpeettoman monimutkainen”, vastausten jakautuma oli samanlainen: viisi testikäyttäjää valitsi arvon 1 (täysin eri mieltä), ja kaksi muuta testikäyttäjää valitsi numeron 2 (lähes eri mieltä). Kolme testikäyttäjää oli täysin samaa mieltä väittämän “mielestäni sovellusta oli helppo käyttää” kanssa, kun taas neljä muuta olivat lähes samaa mieltä. Neljännessä väittämässä kaikki testikäyttäjät olivat samaa mieltä, eivätkä he kokeneet, että he tarvitsisivat teknisen henkilön tukea sovelluksen käyttämiseen.

Viidennessä ja kuudennessa väittämässä vastaukset erosivat eniten. Viidennessä väittämässä (“mielestäni sovelluksen eri osat toimivat hyvin yhteen”) neljä testikäyttäjää oli väittämän kanssa täysin samaa mieltä, kaksi testikäyttäjää oli lähes samaa mieltä ja yksi testikäyttäjä oli lähes eri mieltä. Kuudes väittämä “mielestäni sovelluksessa on liian paljon eri lailla toimivia asioita” keräsi viisi “täysin eri mieltä” -vastausta yhden “lähes eri mieltä” -vastauksen sekä yhden “ei samaa eikä eri mieltä” -vastauksen (numero 3). Seitsemännessä väittämässä (“luulen, että useimmat ihmiset oppivat sovelluksen käytön erittäin nopeasti”) tulokset jakautuivat taas tasaisemmin, kuusi testikäyttäjää valitsi arvon 5 (täysin samaa mieltä), ja yksi testikäyttäjä valitsi arvon 4.

Testikäyttäjien mielestä sovelluksen käyttö ei ollut hankalaa, sillä kuusi heistä oli täysin eri mieltä väittämän numero kahdeksan kanssa, ja seitsemäskin testikäyttäjä oli vain lähes eri mieltä (arvo 2). Kolme käyttäjää tunsi itsensä hyvin varmaksi käyttäessään sovellusta, ja he valitsivat arvon 5 yhdeksänteen kysymykseen, neljän muun käyttäjän valitessa arvon 4. Viisi testikäyttäjää oli täysin eri mieltä kymmenennen väittämän kanssa (“minun piti opetella paljon asioita, ennen kuin sovelluksen käyttö alkoi sujua”), ja loput kaksi olivat lähes eri mieltä väittämän kanssa.

SUS-menetelmän mukaisesti näistä tuloksista saadaan laskettua käytettävyyden kokonaisarvosana. Arvosana lasketaan ottamalla parittomien kysymysten vastausarvot ja vähentämällä niistä yksi, esimerkiksi 2-1=1, jonka jälkeen parillisten kysymysten vastausarvot miinustetaan vuorotellen numerosta viisi, esimerkiksi 5-4=1. Nämä uudet arvot lasketaan yhteen, ja kerrotaan 2,5:lla, jonka jälkeen kokonaisarvoksi tulee jotain välillä 0-100. (Usability.gov) Tämän arvon perusteella testatulle tuotteelle annetaan arvosana, joka kuvaa tuotteen käytettävyyttä (kuva 14).



Kuva 14. SUS-arvosana-asteikko käytettävyyden arviointiin. (Userlytics)

Kuvan 14 mukaisesti arvosanan ”hyvä” saa noin 73 pisteellä, ja arvosanan ”erinomainen” 85 pisteellä. Alle 68 pisteen arvoa pidetään ”reilusti keskiarvoa huonompana”, tasan 68 pisteen arvoa ”keskiarvoisena”, ja 100 pistettä ”parhaimpana mahdollisena” käyttäjäkokemuksena. Testikäyttäjien tulokset olivat seuraavat: 92,5, 80, 100, 85, 100, 90 ja 92,5. Näiden tulosten keskiarvo on 92,4, jolloin käytettävyyden kokonaisarvosanaksi tulee ”erinomainen”.

SUS-menetelmän mukaisten väittämien lisäksi käyttäjät vastasivat vapaasti neljään avoimeen kysymykseen. Vastausten perusteella testikäyttäjät pitivät sovellusta prototyypin perusteella selkeänä, ”modernin näköisenä” sekä loogiselta toiminnallisuudeltaan. Kaikki testikäyttäjät kokivat tietävänsä, mitä sovelluksen eri sivuilla pystyi tekemään, ja yksi testikäyttäjä korosti erityisesti, että ”vaihtoehdot joka tilanteessa oli niin selkeät, että virheklikkailun mahdollisuutta ei juuri ollut”. Muutosehdotuksina avoimissa kysymyksissä tuli jo havainnointivaiheessa huomatut asiat: “kirjaudu ulos” -painikkeen näkyvyys koettiin hieman heikkona, ja ilmoitushistoria-osio oli hieman liian alhaalla suhteessa muuhun sisältöön.

Vaikka käytettävyystestauksessa ilmenikin muutama kehityskohta, oli sovelluksen prototyyppi jo hyvällä tasolla käytettävyyden suhteen. Testikäyttäjät pitivät sovellusta helposti ymmärrettävänä ja selkeänä, ja onnistuivat hyvin testitehtävien teossa. SUS -arvosana-asteikolla käytettävyyden kokonaisarvosana oli korkealla (”erinomainen”), ja testikäyttäjät kertoivat käyttökokemuksen olleen miellyttävä. Kehityskohdiksi nostetaan testauksen perusteella seuraavat kohdat:

- ”Kirjaudu ulos”-painike tulee olla helpommin erottuva.
- Ilmoitushistoria-osio tulee tehdä helpommin nähtäväksi sovelluksen etusivulla.

Nämä kohdat tarkistetaan ja käyttöliittymäsuunnitelmaa muokataan niin, että lopullisessa sovelluksessa käyttäjien on vielä helpompaa käyttää sovellusta.

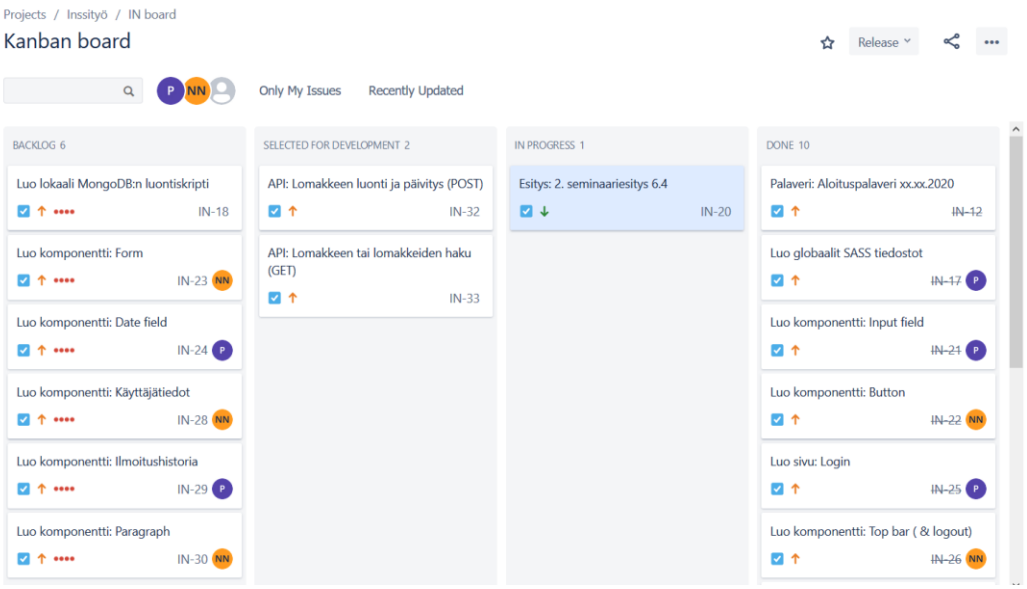
6 Sovelluksen toteuttaminen

Sovelluksen toteuttaminen on yhdistelmä ohjelmointia, kehityssyklien hallintaa sekä testausta ja laadunvarmistusta. Jokaisen näistä osa-alueista on toimittava, jotta lopputuloksena on ehyt kokonaisuus ja onnistunut projekti.

6.1 Sovelluksen toteuttaminen ketterästi

Sovelluksen kehitys- ja suunnitteluvaiheet toteutettiin työnhallintatyökalulla nimeltä Jira. Jiran Kanban-taulua käyttämällä insinöörityön hallinta sekä sen edistyksen tilastoiminen oli selkeää. Käytetyssä taulussa työt jaettiin insinöörityön tekijöiden välillä, jolloin vastuu jakautui tasaisesti ja vältyttiin epäselvyyksiltä. Tämä oli erittäin tärkeää kahden hengen opinnäytetyössä, jossa työn määrä on suurempi ja sen hallinta haastavampaa.

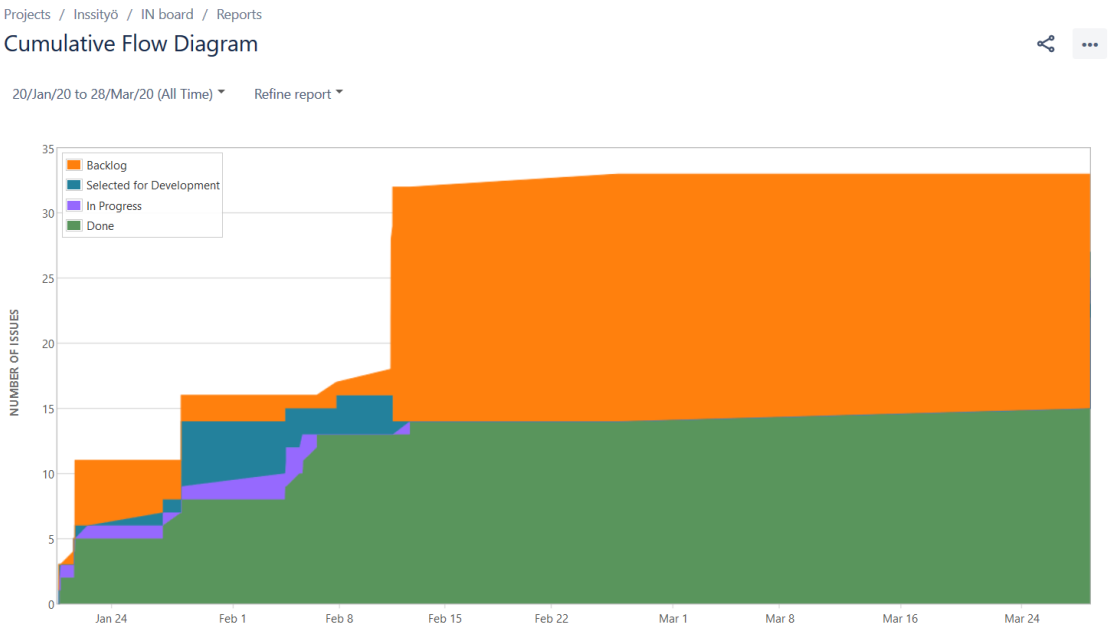
Kanban-taulussa sen käyttäjä valitsee itse niin sanotusta Backlog-osiosta (kasautuneiden töiden jono) itselleen työn, jonka hän siirtää Selected for Development -osioon ja ajallaan In Progress-osioon. Valitun työn tullessa valmiiksi siirretään se Done-osioon, jolloin kaikki Kanban-taulua käyttävät näkevät edistyksen (kuva 15).



Kuva 15. Esimerkki Kanban-taulun perusnäkymästä.

Kanban-taulussa on mahdollista nähdä selkeästi, missä vaiheessa mikäkin työ on ja kuka työstää mitäkin työtä, mikä on suureksi avuksi, ellei jopa välttämätöntä, monen työntekijän projekteissa. Tämä selkeä töiden organisointi edesauttaa projektin pysymistä hallinnassa sekä aikataulussa.

Jokainen työ (kulkee myös nimellä tiketti/ticket) sisältää tarkemman kuvauksen (description), mitä kyseinen työ sisältää, sekä kuvauksen siitä, mitä vaaditaan, jotta työn voidaan sanoa olevan valmis (acceptance criteria), se voidaan siirtää Done-osioon. Kaikki tämä edistyminen on myös mahdollista nähdä projektin kehitysdiagrammina (kuva 16).



Kuva 16. Jira sisältää myös ominaisuuden tarkastella graafina projektin kehitystä, joka omalta osaltaan on suureksi avuksi kokonaiskuvan hahmottamisessa ketterässä projektissa, jossa edistys usein tapahtuu sykleissä.

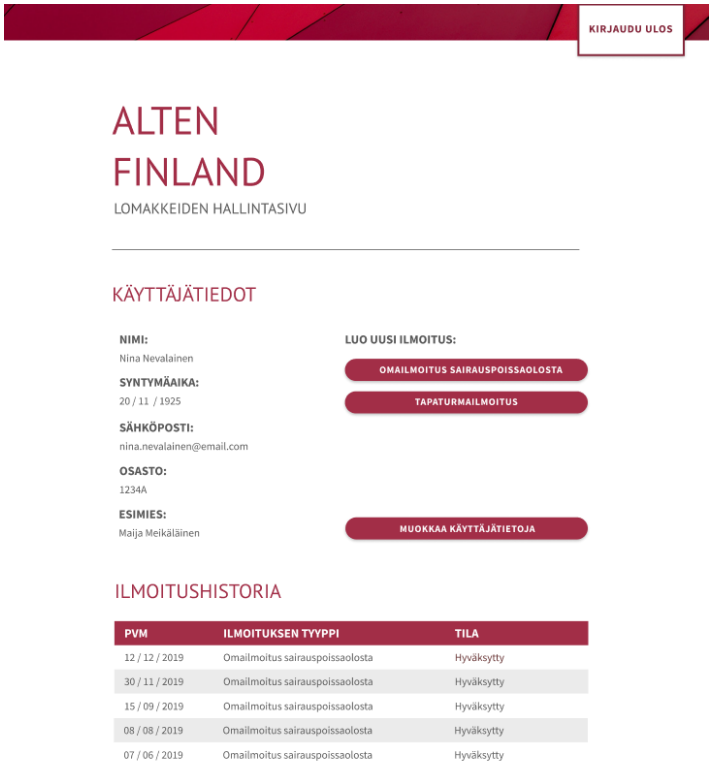
Diagrammi antaa selkeän kuvan projektin etenemisestä pitemmällä aikavälillä. Tästä diagrammista saa myös helposti jälkikäteen selville, mitkä ovat olleet ongelmallisia hetkiä projektin kehityksessä, mikä edesauttaa tulevien kehityssyklien onnistumista.

Tässä insinöörityöprojektissa kappaleessa 2.4 esitetty ja selitetty tapa jakaa ketterä kehitys niin sanottuihin sprintteihin, eli lyhyisiin kehityssykleihin, tapahtui enimmäkseen jakamalla työt opinnäytetyöseminaarien välille. Vaikka tätä metodia käyttäen kehityssyklit

olivat hieman normaalia pidempiä kuin mitä ketterässä kehityksessä monesti on, saatiin tätä kautta luonnolliset aikarajat työn edistymiselle.

6.2 Sovelluksen käyttöliittymän suunnitelman muokkaus

Käyttöliittymän toteutus (ohjelmointityö) tapahtui käyttöliittymätestauksen jälkeen, sillä palautteen saaminen käyttöliittymästä ennen ohjelmointia vähentää turhan työn määrää, ja täten edesauttaa ketterää kehitystä. Käyttöliittymätestauksessa selvisi, että sivustolla oleva uloskirjautumispainike, joka sijaitsee aina sivun oikeassa ylälaidassa, jäi ajoittain turhan huomaamattomaksi. Vaikka ongelma ei ollut suuri, oli se helposti korjattavissa muokkaamalla painikkeen värit erottumaan paremmin taustastaan. Tämä tapahtui muuttamalla alkuperäinen teeman punainen taustaväri valkoiseksi sekä rajaamalla painike tällä edellä mainitulla punaisella. Tämän lisäksi painikkeen tekstin väri vaihdettiin valkoisesta teeman punaiseksi (kuva 17).



Kuva 17. Käytettävyystestauksen perusteella tehtiin muutos kirjaudu ulos -nappulan näkyvyyteen sekä nostettiin ilmoitushistoria-komponenttia ylöspäin, lähemmäksi käyttäjätiedot-komponenttia.

Toinen käyttäjätestauksessa esille noussut palaute oli ilmoitushistorian sijaitseminen liian alhaalla ruudulla. Tämä korjattiin nostamalla ilmoitushistoria-komponenttia 32px (pikseliä) ruudulla ylemmäs, joka oli riittävä muutos tuomaan se esille selkeämmin ruudulla (kuva 17).

Edellä mainitut palautteen mukaan tehdyt muutokset vaikuttavat usein pieniltä, mutta ovat osana suurempaa käytettävyyskokonaisuutta, jonka tulisi aina olla ohjelmistokehityksessä ja suunnittelussa etusijalla. Tämän vuoksi mitättömiltäkin vaikuttavat palautteet on hyvä ottaa tarkempaan tarkasteluun. Palautteeseen perustuvat korjaukset, joiden avulla käyttäjäkokemusta voi parantaa, vaativat usein käytännössä varsin kevyitä muutoksia.

6.3 Sovelluksen käyttöliittymän teknologiat ja toteutus

Sovelluksen käyttöliittymän rakenne ja toiminnallisuus tehtiin Reactilla, ja tyylittelyyn käytettiin SASS:ia (Syntatically Awesome Style Sheets). Teknologiavalinnat käyttöliittymän kehitykseen tehtiin pitäen mielessä kyseisen teknologian tulevaisuudennäkymät, jotta sovellusta ei tultaisi kehittämään vanhenevilla tai muista syistä huonot tulevaisuudennäkymät omaavilla teknologioilla. Tämä pitäen mielessä valittiin käyttöliittymän toteutukseen pääteknologiaksi Javascript-ohjelmointikielen kirjasto React. React on tällä hetkellä suosituin internetkäyttöliittymien ohjelmointiteknologia, ja omaa vahvat tulevaisuudennäkymät, sillä se on Facebook Inc:in luoma ja ylläpitämä, sekä sen yhteisön tuki on aktiivista.

React soveltui hyvin tämän sovelluksen rakentamiseen, koska sillä on vaivattomampaa ohjelmoida dynaamisia internetsivuja kuin mitä esimerkiksi pelkällä Javascriptillä. Yksinkertaistettuna, dynaamisella sivustolla tarkoitetaan internetsivua, jossa sen sisältö on riippuvainen erilaisista muuttujista, joiden perusteella sivuston sisältö saattaa muuttua. Kovin kauas menneisyyteen ei tarvitse mennä, jotta päästään aikakaudelle, jolloin suurin osa internetissä olevista sivustoista oli dynaamisen sivuston vastakohtia, eli staattisia. Tällä tarkoitetaan sivua, jonka sisältö on niin sanotusti kovakoodattu, jolloin sen sisältö on aina sama, se omaa kykyä muuttua dynaamisesti eri muuttujien ohjaamana.

Reactissa yleinen tapa luoda sivusto on rakentaa se monesta uudelleenkäytettävästä komponentista (esimerkkikoodi 1), jotka on mahdollista helposti ottaa käyttöön useassa eri osiossa sivustoa, jolloin turhan työn määrä potentiaalisesti vähenee.

```
import React from 'react';

const Input = ({ type, placeholder, id, labelText, onChange }) =>
{
  return (
    <div className="input">
      <label htmlFor={id}>
        {labelText}
      </label>
      <input type={type} placeholder={placeholder}
      id={id} onChange={onChange} />
    </div>
  );
};

export default Input;
```

Esimerkkikoodi 1. Reactin ominaispiirteitä sen komponenteissa on niin sanottujen muuttujien sisällyttäminen koodin, jonka vuoksi komponentit ovatkin usein uudelleenkäytettäviä. Tässä esimerkkinä sovelluksen Input-komponentti.

Eräs suurimmista syistä Reactin komponenttien uudelleenkäytettävyyteen on niissä käytettyjen muuttujien olemassaolo, sillä nämä muuttujat on mahdollista korvata halutulla sisällöllä, kun komponentti tuodaan sivustolle. Tämän vuoksi Reactin komponentteja voikin usein pitää vain kehyksinä, jotka myöhemmin koodissa täytetään merkityksellisellä sisällöllä. Myös usean komponentin yhdistely on mahdollista yksittäisten komponenttien itsenäisyyden ja riippumattomuuden vuoksi.

Siinä missä React on käytössä sovelluksessa kehysten ja sisällön luontiin, on sovelluksessa käytössä myös muita teknologioita, jotka ovat yhtä tärkeitä kokonaisuuden luonnissa. Yksi näistä tärkeistä teknologioista on CSS. CSS:n avulla voidaan muokata sovelluksen ulkoasua, kuten taustavärejä, typografiaa sekä muuta visuaalista ilmettä. Sovelluksessa käytetään CSS:n niin sanottua jälkikäsittelijää (post processor) nimeltä SASS, jolla CSS:n kirjoittaminen sujuu vaivattomammin. Tämänkaltainen jälkikäsittelijä antaa kehittäjälle tehokkaamman syntaksin (esimerkkikoodi 2), jota hän käyttää koodinsa kirjoittamiseen, ja sen jälkeen jälkikäsittelijä muuntaa syntaksin muotoon, jonka selaimet ymmärtävät.

```
@import '../../globals/variables';
@import '../../globals/mixins';


input[type=text], input[type=password] {
  width: 100%;
  display: block;
  border: none;
  border-bottom: 1px solid $pure-black;
  padding: $size-one-third 0;
  margin-top: $size-one-third;
  font-size: $size-one;


  @include breakpoint-small {
    font-size: 0.9rem;
  }
}
```

Esimerkkikoodi 2. SASS antaa mahdollisuuden kirjoittaa CSS-koodia tehokkaammalla syntaksilla, jolloin toistuvan koodin määrä vähenee.

Niin sanottu normaali CSS ei sisällä mahdollisuutta kirjoittaa sisennettyä koodia, joka on erinomainen tapa tehostaa koodin määrä, sillä usein halutaan komponentin osion perivän koodia toiselta osiolta. SASS antaa myös mahdollisuuden ladata koodiin (import) eri tiedostoja, jotka sisältävät valmiiksi määriteltyjä funktioita tai muuttujia, joita voi käyttää SASS-tiedostossa ilman, että niitä on aina tarve kirjoittaa uudelleen.

6.4 Sovelluksen taustajärjestelmien teknologiat ja toteutus

Sovelluksen taustajärjestelmät tehtiin tuleva jatkokehitys mielessä. Tämä käytännössä tarkoittaa sitä, että tällä hetkellä taustajärjestelmät omaavat ne minimivaatimukset, joilla sovellus toimii tähän insinöörityöhön tarvittavassa tilassa. Esimerkiksi kattavan ja turvallisen tietokannan pystyttäminen ja ylläpitäminen vaati resursseja, joita ei ollut mahdollista saada opinnäytetyötä tehdessä. Tämän vuoksi tietokanta on tehty toimimaan testausmielessä, jotta sovelluksen toimivuus on ollut mahdollista varmistaa.

Kuten käyttöliittymänkin kohdalla, valittiin taustajärjestelmien teknologiat samoin perustein. Oli tärkeä varmistaa, että teknologioiksi ei valittu sellaisia, joiden tulevaisuudentuki ei olisi taattu. Tämä mielessä pitäen valittiin taustajärjestelmien pohjateknologiaksi Javascript-pohjainen Node.js sekä tietokantateknologiaksi MongoDB.

Node.js on avoimen lähdekoodin Javasript-ympäristö, jolla Javascript-koodia on mahdollista suorittaa palvelimella. Erona siihen, miten internetsivut ovat menneisyydessä toimineet, on edellä mainittu Node.js:n mahdollistama koodin suorittaminen palvelimella käyttäjän oman laitteen sijaan. Käytännössä verkkosivu rakennetaan sovellukseen liitetyllä palvelimella, jonka jälkeen tämä sivu lähetetään käyttäjälle.

Node.js on käytössä, kun tarkoituksena on luoda yhteys palvelimella, joka tapahtuu niin sanotun API-kutsun (ohjelmointirajapinta) myötä. Tämänkaltaisessa API-kutsussa koodissa yksinkertaistettuna kerrotaan, minkälainen kutsu palvelimelle tehdään ja mitä halutaan lähettää tai tuoda takaisin. Usein API-kutsut sisältävät suuren määrän muita yhdistettyjä kirjastoja, apufunktioita sekä muuta, jotka ovat olennainen osa kokonaisuuden toimimista (esimerkkikoodi 3).

```
const addUser = async (req, res) => {
  const { body } = req;
  try {
  const hashPassword = await bcrypt.hash(req.body.password, 10);
  const sentUser = await daoAddUser({
    name: body.name,
    birthdate: body.birthdate,
    email: body.email,
    password: hashPassword
  });
  return res.status(201).send(sentUser);
} catch (err) {
  return res.status(500).send({ message: err.toString()
  });
 };
};
```

Esimerkkikoodi 3. API-kutsuissa on tärkeä varmistaa liikkuvan tiedon turvallisuus, jonka vuoksi usein on tarve tuoda koodin avuksi ulkopuolisia kirjastoja tai muita järjestelmiä, jotka avustavat esimerkiksi salasanan suojaamisessa.

API-kutsut tarvitsevat paikan, jonne lähettää tietoa, sekä mistä sitä hakea. Mahdollisuuden tälle antaa tietokanta, joita on nykyään useita erilaisia eri teknologioilla suunniteltuina. Sovelluksen luonteen vuoksi valittiin insinöörityön osuuden kehitykseen MongoDB-tietokanta sen joustavuuden vuoksi. MongoDB mahdollistaa tiedon lähettämisen suhteellisen vapaassa muodossa, mutta jos kyseessä on suuri määrä tietoa, ei se välttämättä ole paras teknologinen ratkaisu.

MongoDB vaatii niin sanotun skeeman (esimerkkikoodi 4) luonnin, joka tarkoittaa tiedostoa, jolla ohjeistetaan API-kutsua siitä, minkälaista tietoa tietokannassa on sekä minkä tyyppistä tämän tiedon kuuluu olla.

```
const mongoose = require('mongoose');


const User = new mongoose.Schema({
  name: {
    required: [true, 'Name is required'],
    type: String
  },
  birthdate: {
    required: [true, 'Birthdate is required'],
    type: String
  },
  email: {
    required: [true, 'Email address is required'],
    type: String
  },
  password: {
    required: [true, 'Password is required'],
    type: String
  }
});


module.exports = mongoose.model('User', User);
```

Esimerkkikoodi 4. Tietokannalle täytyy melkein aina kertoa, minkälaisen tiedon kanssa se on tekemisissä, jotta sinne on mahdollista lähettää tietoa sekä hakea sitä. Tässä luodaan käyttäjän skeema.

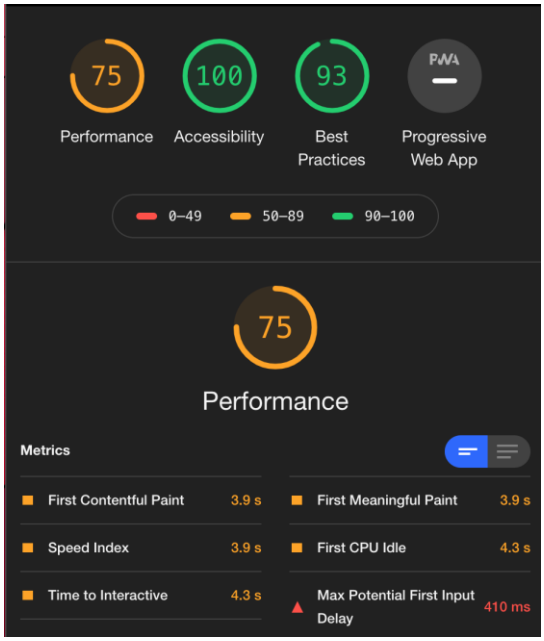
Katsoessa esimerkkikoodia 4 löytyy siitä yksinkertainen esimerkki siitä, minkälainen tämä edellä mainittu skeema voi olla. Tässä on määritelty, minkälaista tietoa tietokantaan tallennettavien käyttäjien tulee sisältää. On esimerkiksi mahdollista kertoa, onko jokin tieto pakollista ja täytyykö sen olla tekstiä vai numeroita. Tässä voidaan asettaa myös muita määritelmiä, jotka voivat olla hyvinkin tarkkoja. Määritellen selvästi esimerkiksi, minkä pituinen lähetetyn tiedon kuuluu olla.

6.5 Sovelluksen laadunvarmistus

Sovelluksen laadunvarmistus tehtiin Chrome-selaimen Lighthouse-validaatiotyökalua käyttäen. Pidemmällä tähtäimellä sovelluksen laadunvarmistus ja eheys on usein tärkeää tehdä myös itse ohjelman sisältä esimerkiksi automaatiotestauksella, mutta tässä

insinöörityössä Lighthouse oli sopiva ja tehokas työkalu web-sovelluksen tämänhetkisen tilan laadunvarmistukseen.

Lighthouse on selaimen sisällä ajettava lisäosa, jolla on mahdollista selvittää minkä tahansa sivuston toimivuutta, suorituskykyä sekä käytettävyyttä. Testi ajetaan antamalla muuttujia, joita testi seuraa ja joiden perusteella testi simuloi erityyppisiä tilanteita. Näihin saattavat vaikuttaa käyttäjän tietokoneen tai mobiililaitteen tehokkuus, muistin määrä tai muut tämän kaltaiset muuttujat. Testissä on mahdollista myös simuloida eri nopeuden omaavia internetyhteyksiä (maksimina toki testitietokoneen oma internetyhteysnopeus), selviää, onko sivusto käytettävä myös hitaimmilla yhteyksillä. Kaikesta testatusta Lighthouse antaa yhteenvedon (kuva 18), saa hyvän yleiskatsauksen sivustosta.



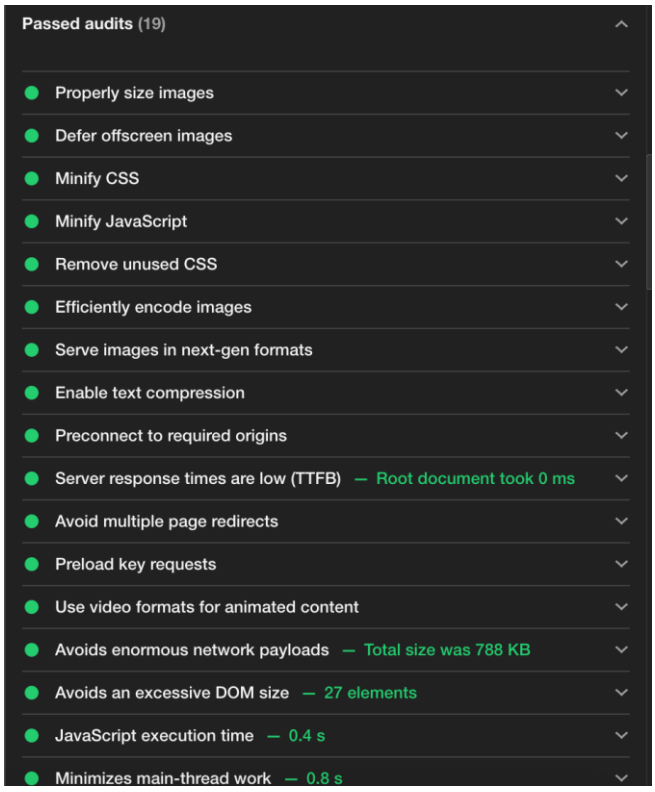
Kuva 18. Chrome-selaimen Lighthouse-validaatiotyökalu antaa erinomaisen yleiskuvan mistä tahansa sivustosta. Tässä insinöörityön sovelluksen yleispisteet.

Validaatiotestin yhteenveto pisteyttää sivuston kolmeen eri osa-alueeseen, jotka ovat suorituskyky, saavutettavuus sekä ohjelmistokehityksen ja suunnittelun parhaat toimintatavat. Jokaisesta osa-alueesta on mahdollista saada syvällisempää tietoa valitsemalla se osio lähempään tarkasteluun. Kuvassa 18 näkyy suorituskyvyn pistemäärä sekä osa sen arviointiin käytetyistä muuttujista. Koska kyseinen testi tehtiin simuloimalla hitaampaa internetyhteyttä, on saatu pistemäärä (75) hyvä. Tämä sama pistemäärä saatiin

myös simuloimalla hidasta internetyhteyttä mobiililaitteella. Tekemällä samat testit, mutta ilman hitaan internetyhteyden simulointia, oli pistemäärä usein vähintään 90, eli tältä osin sovellus on erittäin käytettävä.

Suurimpina hidastavina tekijöinä ovat sivuston tiedostojen ja muiden resurssien lataukset, jotka tapahtuvat käyttäjän tullessa sivulle ensimmäistä kertaa. Tämä sisältää niin itse koodin, tyylitiedostojen, typografian kuin myös kuvatiedostojen lataukset. Mikäli olisi tarvetta parempaan pistemäärään, varsinkin hitailla internetyhteyksillä, olisivat nämä asioita, joihin tulisi kiinnittää huomiota jatkokehityksessä. Esimerkiksi kuvatiedostot säilyttävät yleensä riittävästi laatunsa, vaikka ne pakattaisiinkin pienemmäksi latausnopeuden tehostamiseksi.

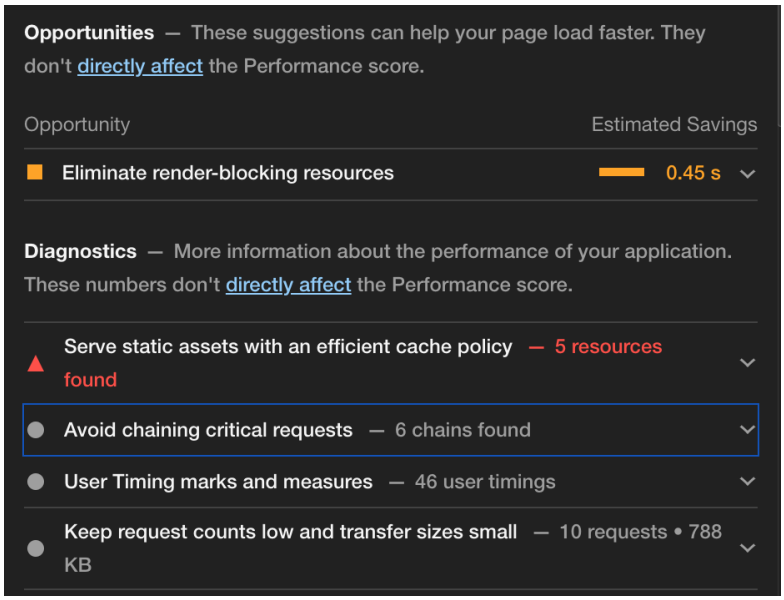
Tämän sivuston kohdalla suuriin toimiin ei kuitenkaan ole tarvetta, sillä esimerkiksi juuri kuvatiedostojen koko on Lighthouse-validaatiotestin mielestä hyväksyttävä (kuva 19), mikä selviää tutkimalla testiä tarkemmin.



Kuva 19. Tutkimalla Lighthouse-validaatiosovellusta ja sen tekemää testiä, on mahdollista löytää erittäinkin tarkkaa ja hyödyllistä tietoa siitä, mikä toimii, mikä ei ja mikä saattaa syötä liikaa resursseja.

Jokaisen osa-alueen kohdalla on “Passed audits”-osio, joka tarkoittaa läpimenneitä testejä. Tutkimalla lisää testejä, olivat ne läpimenneitä tai eivät, Lighthouse myös kertoo (mikäli mahdollista), kuinka paljon aikaa jokin osa sivustoa on käyttänyt lataukseen.

Tämä on erityisen tärkeää, sillä mikäli jokin osa sivustoa syö suuresti latausresursseja, johtuu tämä usein huonosta sovelluksen suunnittelusta ja/tai ohjelmoinnista. Tämän vuoksi testaaminen, tehdään se millä työkalulla hyvänsä, on aina erittäin tärkeä osa sovelluskehitystä. Modernit työkalut, kuten Lighthouse, kertovat usein myös erittäin selkeästi ja suoraan mahdollisia kehityskohteita (kuva 20), tutkimalla voidaan säästää suuriakin määriä resursseja.



Kuva 20. Lighthouse-validaatiosovellus antaa hyviä kehityskohteita sivuston tehostamiseen, mikäli kehittäjältä löytyy oikeanlaista osaamista, jonka avulla nämä kehityskohteet on mahdollista toteuttaa.

Osa Lighthouse-validaatiosovelluksen kehityskohteista vaatii sovelluskehittäjältä hyvää ymmärrystä internetsovellusten toiminnasta sekä siitä, miten esimerkiksi HTTP-pyynnöt (Hypertext Transfer Protocol) toimivat, tai miten API-kutsuja tehdään, sekä miten niistä saatua tietoa käsitellään. Tämä kaiken ymmärtäminen edesauttaa ymmärtämään, miten tätä kokonaisuutta on mahdollista tehostaa. Syvällisempi tieto ei aina tosin ole tarpeellista, sillä tekemällä pieniäkin muutoksia mahdollisiin ongelmakohtiin (kuten kuvatiedostojen kokoihin) on mahdollista tuoda suuriakin resurssisäästöjä, jotka parantavat käyttökokemusta.

7 Sovelluksen validointi

Kun sovellus saatiin valmiiksi, oli aika uusia aiemmin prototyypille tehty käytettävyystestaus. Käytettävyystestauksen uusinnalla voidaan validoida valmis sovellus ja varmistua sen toiminnasta ja sovelluskehityksen tavoitteisiin pääsystä. Uusitulla testauksella haluttiin varmistaa, että sovelluksen käytettävyys on edelleen hyvällä tasolla ja että ensimmäisen testauksen jälkeen tehdyt muutokset eivät vaikuttaneet sovelluksen käytettävyyteen negatiivisesti. Vaikka muutokset valmiin sovelluksen ja prototyypin välillä eivät olekaan suuria, voi niillä silti olla merkitystä käyttökokemukseen.

Toinen käytettävyystestaus oli tarkoitus suorittaa samaan tapaan kuin ensimmäinen käytettävyystestaus, niin testauspaikaltaan kuin testikäyttäjien osalta. Globaalin pandemiatilanteen takia tätä ei kuitenkaan pystytty järjestämään samalla tavalla, sillä ihmisten kokoontumisia rajoitettiin valtakunnallisella tasolla ja sosiaalisia kontakteja tuli välttää. Tämän insinöörityön aiheena olevan sovelluksen käyttäjäryhmä on kuitenkin onneksi laaja, joten testikäyttäjät kasattiin insinöörityötä tekevien henkilöiden lähipiiristä.

Tähän testaukseen saatiin viisi työikäistä testikäyttäjää, joilla oli perusosaaminen tietokoneohjelmistojen käytöstä. Tällä kertaa jokaista testaustilannetta ohjasi vain yksi ohjaaja, ja havainnoinnin helpottamiseksi testaustilanteet nauhoitettiin. Testikäyttäjiä ohjeistettiin samaan tapaan kuin ensimmäisessä käytettävyystestauksessa, ja heillä oli käytössään paperinen testitarina ja -tehtävät sekä valmiiksi sisäänkirjautumissivulle avattu sovellus.

Testaustilanteet sujuivat hyvin samaan tapaan kuin ensimmäisessä käytettävyystestauksessa: testikäyttäjät onnistuivat nopeasti siirtymään rekisteröinnistä lomakkeen täyttämiseen. Tällä kertaa kaikki testikäyttäjät onnistuivat myös palaamaan takaisinpäin lomakesivulta nopeasti, ja syntymäpäivän arvon muutos onnistui sujuvasti. Ensimmäiseen testaukseen verrattuna testikäyttäjillä kului nyt enemmän aikaa lomakesivun täyttämiseen, mutta tämä on ymmärrettävää, sillä prototyypissä itse lomakkeen tietoja ei täytetty, vaan sivulla vain käytiin. Lomakkeen täyttö oli nyt kuitenkin sujuvaa, ja testikäyttäjät löysivät lähetyspainikkeen helposti. Tällä kertaa kaikki testikäyttäjät suoriutuivat myös kahdesta

viimeisestä tehtävästä nopeasti: ilmoitushistoria löytyi käyttäjän etusivulta helposti lomakkeen lähetyksen jälkeen, ja myös uloskirjautuminen sujui nopeammin kuin ensimmäisessä testauksessa.

Testaustilanteen jälkeen testikäyttäjät täyttivät saman kyselylomakkeen, jota käytettiin ensimmäisessä käytettävyystestauksessa. SUS-metodin mukaisten kysymysten vastaukset löytyvät taulukosta 2.

Taulukko 2. Testikäyttäjien vastausjakaumat toisessa käytettävyystestauksessa.

Kysymykset	**1** Täysin eri mieltä	**2**	**3**	**4**	**5** Täysin samaa mieltä
1. Luulen, että käyttäisin tätä sovellusta mielelläni	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (20%)	4 (80%)
2. Mielestäni sovellus oli tarpeettoman monimutkainen	5 (100%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
3. Mielestäni sovellusta oli helppo käyttää	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (20%)	4 (80%)
4. Luulen, että tarvitsen teknisen henkilön tukea, jotta osaisin käyttää tätä sovellusta	5 (100%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
5. Mielestäni sovelluksen eri osat toimivat hyvin yhteen	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (20%)	4 (80%)
6. Mielestäni sovelluksessa on liian paljon eri lailla toimivia asioita	5 (100%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
7. Luulen, että useimmat ihmiset oppivat sovelluksen käytön erittäin nopeasti	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	5 (100%)
8. Mielestäni sovelluksen käyttö oli hyvin hankalaa	5 (100%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
9. Tunsin itseni hyvin varmaksi, kun käytin sovellusta	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (20%)	4 (80%)
10. Minun piti opetella paljon asioita, ennen kuin sovelluksen käyttö alkoi sujua	5 (100%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)

Tällä testauskerralla testikäyttäjien tulokset olivat seuraavat: 95, 100, 95, 100 ja 100. Näiden tulosten keskiarvo on 98, jolloin arvosana-asteikon mukaan käytettävyyden kokonaisarvosanaksi tulee edelleen ”erinomainen”, ja tulosten keskiarvo nousee 5,6 pisteellä ensimmäiseen testaukseen verratessa.

Avoimien kysymysten vastaukset olivat hyvin saman tapaisia kuin ensimmäisessä käytettävyystestauksessa. Sivuston käyttötarkoitus oli testikäyttäjille selkeä, ja sovellus itsessään oli vastausten perusteella selkeä ja loogisesti toimiva. Yksi testikäyttäjä haluaisi sovellukseen eri värimaailman, sillä nyt se on hänen mielestään ”liian punainen”. Yksi käyttäjä mainitsi myös syntymäpäivän valitsemisen olleen turhan hankalaa, sillä se valittiin kalenterinäkymästä. Käyttäjä olisi mieluummin halunnut kirjoittaa päivämäärän itse kyseiseen kohtaan.

Tämän käytettävyystestauksen perusteella rakennettu sovellus on käytettävyydeltään korkeaa luokkaa, ja testikäyttäjät käyttivät sitä mielellään. Vaikka testikäyttäjiä oli toisessa testauksessa kaksi vähemmän, on viisi testikäyttäjää kuitenkin vielä hyväksyttävä määrä käytettävyystestauksen suorittamiseen hyväksytysti. Testauksen perusteella voidaan todeta, että sovelluksen suunnittelussa ja toteutuksessa on päästy tavoitteisiin: sovellus on käytettävyydeltään hyvä, ja käyttäjät voivat luoda sillä nopeasti sairauspoissaoloilmoituksen. Lisäksi sovelluksen käytettävyyttä saatiin parannettua ensimmäisessä käytettävyystestauksessa esiin tulleiden kehityskohtien avulla.

8 Yhteenveto ja pohdinta

Tämän insinöörityön tarkoituksena oli suunnitella ja toteuttaa web-sovellus, joka olisi käytettävyydeltään korkea sekä saavutettavuusvaatimukset huomioiva. Toteutetun sairauspoissaoloilmoitussovelluksen on tarkoitus korvata tilaajan vanha, paperinen työntekijöiden lyhytkestoisten sairauspoissaolojen kirjausjärjestelmä. Toteutetun web-sovelluksen tuli olla mahdollisimman helppokäyttöinen sekä selkeä. Web-sovellus haluttiin kehittää ketterän kehityksen menetelmin, ja tämä tehtiin heti alusta alkaen pilkkomalla insinöörityön eri vaiheet omiksi sprinteikseen.

Työn aluksi tutustuttiin käytettävyyden ja saavutettavuuden teoriaan, jotta sovelluksen prototyyppiä kehittäessä osattiin huomioida nämä osa-alueet. Käytettävyys on isossa osassa web-sovelluksen käyttökokemusta, sillä huonosti käytettävä sovellus voi jopa estää käyttäjää suoriutumasta tehtävistä, joita hän sovelluksella haluaisi tehdä. Korkean käytettävyyden omaavalla sovelluksella tehtävien teko on tehokasta ja virheetöntä, ja käyttäjäkokemus on miellyttävä. Käyttäjäkokemukseen liittyy myös saavutettavuus, eli käytön esteettömyys. Web-sovelluksien suunnittelussa ja toteuttamisessa on tärkeää ottaa huomioon saavutettavuusvaatimukset, jotka varmistavat niin käyttöliittymän ulkoasun kuin myös sovelluksen lähdekoodin soveltuvan myös käyttäjille, joilla on erityistarpeita.

Suunnitteluvaiheessa haluttiin varmistua sitä, että suunniteltu web-sovellus on käytettävyydeltään korkealla tasolla, ja tätä varten suoritettiin käytettävyystestaus sovelluksen prototyypillä. Käytettävyystestauksen perusteella prototyypin käytettävyyden kokonaisarvosana SUS-menetelmän mukaan oli ”erinomainen”, ja tulosten avulla sovelluksen käyttöliittymän suunnitelmaa saatiin vielä paranneltua esimerkiksi ”kirjaudu ulos” -painikkeen näkyvyyttä vahvistamalla. Parannellun suunnitelman perusteella aloitettiin sovelluksen kehitystyö. Sovelluksen toteutukseen valittiin modernit, helposti yhteensopivat teknologiat.

Kun web-sovelluksen kehitystyö oli valmis, suoritettiin vielä toinen käytettävyystestaus, jotta varmistuttiin sovelluksen käytettävyyden tasosta. Toisen käytettävyystestauksen tulosten perusteella sovelluksen käytettävyys oli vieläkin korkeammalla tasolla kuin mitä

se oli ensimmäisessä testauksessa, ja käyttäjät kuvailivat sitä selkeäksi sekä miellyttäväksi käyttää. SUS-menetelmän mukainen käytettävyyden kokonaisarvosana oli erittäin korkea ”erinomainen”, tulosten keskiarvon ollessa 98/100. Näiden tulosten perusteella voidaan todeta, että sovelluksen suunnittelussa ja toteutuksessa saavutettiin insinöörityön alussa asetetut tavoitteet erinomaisesti. Myös sovelluksen laadunvarmistuksen testitulokset tukevat tätä näkökantaa, sillä sovellus sai validaatiotestissä suorituskyvystä, saavutettavuudesta sekä ”ohjelmistokehityksen ja suunnittelun parhaat toimintatavat” -kohdasta erittäin korkeat pisteet.

Web-sovelluksen toteutuksessa saavutettiin asiakasvaatimukset: sovelluksen avulla työntekijä pystyy rekisteröitymään ja kirjautumaan käyttäjätilillään sovellukseen, täyttämään ”omailmoituksen sairauspoissaolosta”, sekä tarkastelemaan tekemiensä sairauspoissaolojen ilmoitushistoria-näkymää. Web-sovellus on käytettävyydeltään korkealla tasolla, ja sen avulla sairauspoissaoloilmoituksen teko on tehokasta sekä miellyttävää. Insinöörityössä toteutetun web-sovelluksen jatkokehitys on tilaajalle helppoa, sillä sovelluksen toteutukseen valitut teknologiat ovat moderneja, ja niiden käyttö on yleistä web-sovelluskehityksessä. Jatkokehityksessä tulisi seuraavaksi keskittyä hallintakäyttäjien toiminnallisuuksien toteuttamiseen, jotta esimiehet pääsisivät sovelluksen kautta hyväksymään sairauspoissaoloilmoitukset. Suunnitteluvaiheessa käyttäjän etusivulle luotiin käyttöliittymäsuunnitelmaan esimerkki siitä, kuinka käyttäjä voisi tehdä myös muita yrityksen sisäisiä ilmoituksia sovelluksessa. Tämä olisi myös hyvä jatkokehityksen kohta tulevaisuudessa.

Käytettävyyden sekä saavutettavuuden huomioiminen web-sovellusta toteutettaessa on erittäin tärkeää, sillä hyvällä käytettävyydellä sekä käytön esteettömyydellä pidetään huoli siitä, että sovelluksen käyttäjäkunta on mahdollisimman laaja. Käyttäjäystävällisen web-sovelluksen luominen on monitahoinen projekti, jossa tarvitaan laajaa osaamista niin suunnittelusta, käyttäjätestauksesta kuin myös ohjelmoinnista. Insinöörityön aikana osaaminen ja ymmärrys kaikista näistä osa-alueista kehittyi erittäin paljon, ja tätä osaamista tullaan varmasti hyödyntämään jatkossa työelämässä.

Lähteet

Bias, R.; Mayhew, D. 2005. Cost-justifying usability: an update for an Internet age. Boston: Morgan Kaufman.

European committee for standardization. 2018. ISO 9241-11:2018. Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts.

Etelä-Suomen aluehallintovirasto. Saavutettavuusvaatimukset. Verkkoaineisto. <https://www.saavutettavuusvaatimukset.fi>. Luettu 20.2.2020.

Finlex. Työterveyshuoltolaki 1383/2001. Verkkoaineisto. <https://www.finlex.fi/fi/laki/ajantasa/2001/20011383>. Luettu 10.4.2020.

Hyysalo, Sampsa. 2009. Käyttäjä tuotekehityksessä. Tieto, tutkimus, menetelmät. E-kirja. Taideteollisen korkeakoulun julkaisu B 97.

Nielsen, Jakob. 1999. Designing Web Usability. E-kirja. Peachpit Press.

Nielsen, Jakob. 1993. Usability Engineering. Boston: Academic Press.

Nielsen, Jakob. 1994. 10 Usability Heuristics for User Interface Design. Verkkoaineisto. <https://www.nngroup.com/articles/ten-usability-heuristics/>. Luettu 21.2.2020.

Ovaska, S. & Aula, A. & Majaranta, P. 2005. Käytettävyystutkimuksen menetelmät. Tietojen-käsittelytieteiden laitos, Tampereen Yliopisto. Tampere: Juvenes Print.

Riihiaho Sirpa. 2000. Käytettävyystestauksen muunnelmia. Teoksessa Pantzar E. (toim.) Informaatio, tieto ja tietoyhteiskunta. Suomen Akatemian Tiedon tutkimusohjelman raportteja, 4/2000. Tampere: Tampereen yliopisto. s. 223–230.

Rubin, J. & Chisnell, D. 2008. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. E-kirja. John Wiley & Sons.

Shore, J. & Warden, S. 2008. The Art of Agile Development. E-kirja. O'Reilly Media.

Sinkkonen, I.; Kuoppala, H.; Parkkinen, J. & Vastamäki, R. 2006. Käytettävyyden psykologia. Helsinki: Edita.

Sinkkonen, I.; Nuutila, E. & Törmä, S. 2009. Helppokäyttöisen verkkopalvelun suunnittelu. Helsinki: Tietosanoma.

StatCounter. Mobile Screen Resolution Stats Worldwide. Verkkoaineisto. <https://gs.statcounter.com/screen-resolution-stats/mobile/worldwide>. Luettu 1.5.2020.

The World Wide Web Consortium (W3C). Web Content Accessibility Guidelines (WCAG) 2.1. Verkkoaineisto. <https://www.w3.org/Translations/WCAG21-fi/>. Luettu 24.3.2020.

Työterveyslaitos. Laki edellyttää reagointia sairauspoissaoloihin. Verkkoaineisto. <https://www.ttl.fi/tyontekija/tyoterveyshuolto/tyokyvyn-tuki/laki-edellyttaa-reagointia-sairauspoissaoloihin/>. Luettu 10.4.2020.

Usability.gov. System Usability Scale (SUS). Verkkoaineisto. <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>. Luettu 10.2.2020.

Userlytics. System Usability Score (SUS) and other User Testing Metrics. Verkko-aineisto. <https://www.userlytics.com/blog/system-usability-scale>. Luettu 10.2.2020.

Outsystems. Building Web Accessibility, Part 1: Barriers, Guidelines, and Standards. Verkkoaineisto. <https://www.outsystems.com/blog/posts/building-web-accessibility-barriers-guidelines-standards/>. Luettu 16.4.2020.

Käytettävyystestaus

Tämä kysymyslomake on ALTEN Finlandin uuden sairauspoissaoloilmoitusjärjestelmä omailmoituksen käytettävyyden arviointiin.

Vastaa jokaiseen kohtaan joko kirjoittamalla vastauksesi kysymyksen alle, tai monivalintakysymyksessä valitsemalle arvo väliltä 1 (täysin eri mieltä) - 5 (täysin samaa mieltä).

Kaikki vastaukset ovat luottamuksellisia.

Nimi: _____

Työnimike: _____

Monivalintakysymykset

Valitse arvo väliltä 1 (täysin eri mieltä) - 5 (täysin samaa mieltä).

1. Luulen, että käyttäisin tätä sovellusta mielelläni.
2. Mielestäni sovellus oli tarpeettoman monimutkainen.
3. Mielestäni sovellusta oli helppo käyttää.
4. Luulen, että tarvitsen teknisen henkilön tukea, jotta osaisin käyttää tätä sovellusta.
5. Mielestäni sovelluksen eri osat toimivat hyvin yhteen.
6. Mielestäni sovelluksessa on liian paljon eri lailla toimivia asioita.
7. Luulen, että useimmat ihmiset oppivat sovelluksen käytön erittäin nopeasti.
8. Mielestäni sovelluksen käyttö oli hyvin hankalaa.
9. Tunsin itseni hyvin varmaksi, kun käytin sovellusta.
10. Minun piti opetella paljon asioita, ennen kuin sovelluksen käyttö alkoi sujua.

Avoimet kysymykset

11. Tiesitkö heti sivustolle tullessasi mistä sovelluksessa oli kyse / mikä sen tarkoitus on?

12. Koitko aina tietäväsi, mitä sinun seuraavaksi pitäisi tehdä? Jos et, mikä oli epäselvää?

13. Mitä ajattelet sovelluksen ulkoasusta ja värimaailmasta?

14. Muuta palautetta sovelluksesta?

Kiitos!