SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

TERVEYS- JA HOITOSUUNNITELMAN TIETOSISÄLLÖN AVUSTETTU YHDISTELY ERI VERSIOIDEN VÄLILLÄ

TEKIJÄ: Tero Ikäheimo

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Tero Ikäheimo	
Työn nimi Terveys- ja hoitosuunnitelman tietosisällön avustettu yhdistely eri versioiden välillä	
Päiväys 31.5.2021	Sivumäärä/Liitteet 51
Toimeksiantaja/Yhteistyökumppani(t) Mediconsult Oy	

Tiivistelmä

Opinnäytetyön tavoitteena oli toteuttaa avustavat toiminnot kahden terveys- ja hoitosuunnitelma dokumentin tietosisällön yhdistämiseen. Toteutusympäristönä toimi asiakasyrityksen olemassa oleva verkkopohjainen käyttöliittymätoteutus.

Toteutusosuudessa tuotettiin avustavat toiminnot kahden eroavan tekstin yhdistämiseen ja tiedon kopiontiin Kanta-dokumentista käyttäjän luonnokseen. Näiden lisäksi tuotettiin käyttöliittymän tarvitsemat ikonit ja tietosisällön rivikohtainen eroavaisuuksien tunnistuksen logiikka. Kaikki ominaisuudet toteutettiin käyttäen Angular-sovelluskehystä ja TypeScript ohjelmointikieltä.

Tuloksena saatiin yleiskäyttöinen Angular-komponentti kahden eroavan tekstin yhdistelyyn raahaa ja pudota toiminnallisuudella. Kompleksisen tiedon kopiointi ominaisuus terveys- ja hoitosuunnitelma näkymän vertailutilaan. Tekstin yhdistely ikoni ja tiedon kopiointi-ikoni. Tietosisällön eroavaisuuksien tunnistukseen tuotettiin lisätoiminnallisuus, mikä mahdollisti taulukoiden rivikohtaisten erojen tunnistuksen.

Avainsanat

Angular, TypeScript, tiedonhallintapalvelu, web-ohjelmointi

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Tero Ikäheimo	
Title of Thesis Aided merging of information content on differing versions of the health and care plan	
Date 31 May 2021	Pages/Appendices 51
Client Organization /Partners Mediconsult Oy	

Abstract

The aim of this thesis was to implement features for merging the information content of two health and care plan documents. The thesis was commissioned by Mediconsult Oy and the implementation was done in the commissioner's existing web-based user interface.

A feature for merging two differing texts was produced and the copying of complex data from the Kanta-document to the user's draft was implemented. In addition to these, the icons needed for the user interface were created and an additional row-based logic for detecting discrepancies was implemented. All these features were implemented using the Angular web framework and the TypeScript programming language.

As a result of this thesis, a general-purpose Angular component was produced to merge two differing texts using the drag and drop functionality. A feature for copying complex data from a document to another was implemented in the user interface. Icons for merging texts and copying data were produced and implemented. An additional functionality was made for detecting differences on a row basis.

Keywords

Angular, TypeScript, Information Management Service, Web Development

SISÄLTÖ

1 JOHDANTO

1.1 Toimeksiantaja

Mediconsult Oy on vuonna 1975 perustettu kotimainen tietojärjestelmiä kehittävä yritys. Pääasiallisesti se keskittyy potilastiedon, toiminnanohjauksen, omahoidon ja sähköisen asioinnin ratkaisujen kehittämiseen. Yritys on yksi isoimmista tietojärjestelmätoimittajista sosiaali- ja terveydenhuollon alalla. (Mediconsult Oy, 2021)

1.2 Lyhenteet ja määritelmät

ANGULAR

Googlen kehittämä ja ylläpitämä web-sovelluskehys (Angular v2 ja suuremmat versiot)

MODUULI

Moduuli sanalla viitataan ominaisuus tai pienoisohjelma Angular-moduuleihin. Ominaisuus moduulit sisältävät yleensä yhden isomman toiminto kokonaisuuden, joka koostuu useasta eri komponentista. Pienoisohjelma moduulit sen sijaan kokoavat sisälleen useampia yleiskäyttöisiä pienoisohjelmia. Näitä ohjelmia voidaan käyttää missä tahansa muussa moduulissa, kun ne on siihen tuotu (engl. import).

KOMPONENTTI

Komponentti sanalla viitataan Angular-komponentteihin. Angular ympäristössä komponentti koostuu ohjelman logiikasta, HTML-mallista ja tyylitiedostosta.

PIENOISOHJELMA

Käännös englannin kielen sanasta ”widget”. Tässä opinnäytetyössä sillä viitataan uudelleen käytettävään ohjelman osaan.

SÄÄNNÖLLINEN LAUSEKE

Merkkijono, joka määrittelee haku kaavan, mitä useimmiten käytetään tekstin osien tai niiden sijaintitietojen hakuun.

THS

Lyhenne sanoista: terveys- ja hoitosuunnitelma.

TYPESCRIPT

Microsoftin kehittämä ja ylläpitämä ohjelmointikieli.

LODASH

Funktionaalista ohjelmointi paradigmaa käyttävä JavaScript apuohjelmakirjasto

SCSS

On CSS (engl. Cascading Style Sheet) skriptauskielen laajennus, joka tuo tyylien määrittelyyn uusia ominaisuuksia, kuten muuttujat.

GIT

Versionhallintatyökalu koodille.

SVG

Skaalattava vektorigrafiikka kuvatiedostostandardi.

CRUD

Lyhenne englannin kielen sanoista “Create Read Update Delete”, jota usein käytettään määrittelemään tiedon käsittelyyn tarvittavat toiminnot.

CDK

Lyhenne englannin kielen sanoista ”Component Development Kit”, tällä viitataan koodikirjastoon mikä tarjoaa joukon pohjatoiminnallisuuksia, joita voi käyttää sovelluksen kehittämisessä.

PRETTIER

Koodin muotosääntöjen kokoelma, jolla koodi muotoillaan ennalta määriteltyyn ulkoasuun.

ESLINT

Työkalu, jota käytetään analysoimaan lähdekoodia ja ilmoittamaan mahdollisista virheistä, bugeista, tyylivirheistä tai epämääräisistä rakenteista.

FONT AWESOME

Ikonikirjasto.

ANGULAR MATERIAL

Työkalu, jota käytetään analysoimaan lähdekoodia ja ilmoittamaan mahdollisista virheistä, bugeista, tyylivirheistä tai epämääräisistä rakenteista.

STACK OVERFLOW

Ohjelmointiin keskittynyt foorumi, mistä löytyy laajasti tietoa yleisistä ohjelmointi ongelmista ja niiden ratkaisuista.

1.3 Lähtötilanne

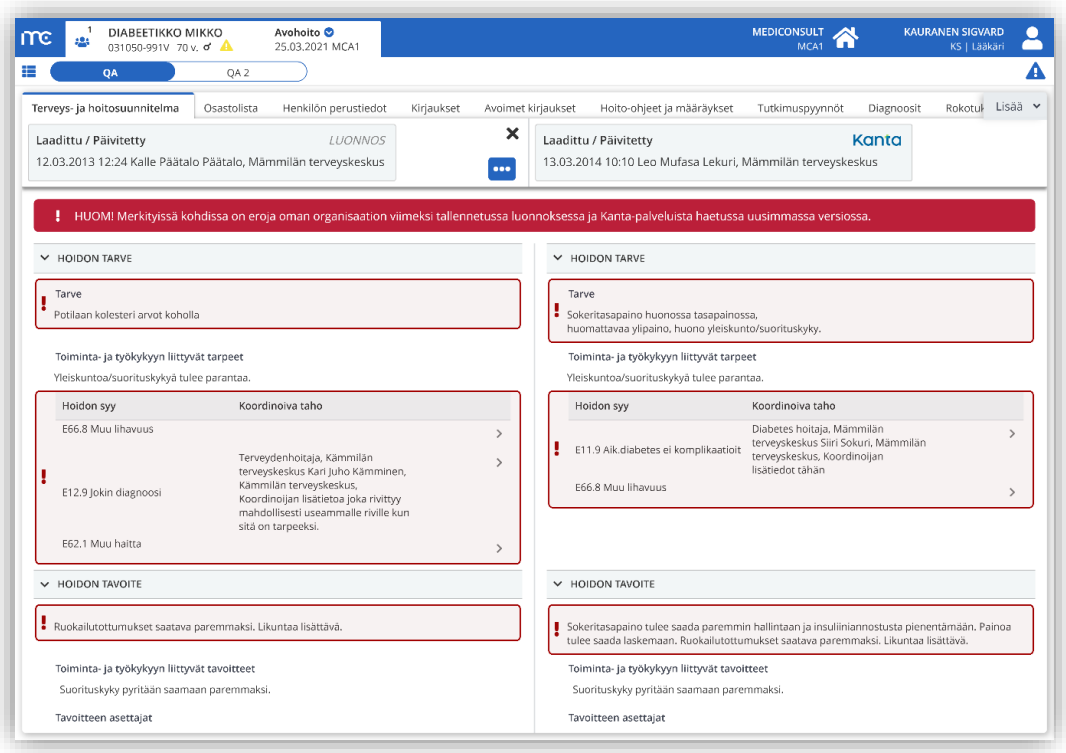
Terveys- ja hoitosuunnitelma on kansalliseen käyttöön tarkoitettu dokumentti, jolla pyritään tukemaan pitkäaikaissairaiden tavoitteellista hoitoa. Dokumentin laadintaan osallistuu potilas itse ja terveydenalan ammattilaiset, jotka yhteistyössä laativat hoitosuunnitelman. Tätä dokumenttia ylläpidetään kansallisen terveysarkiston tiedonhallintapalvelussa, jossa se on terveydenalan ammattilaisten muokattavissa useiden eri palveluntarjoajien järjestelmien kautta. Dokumentti on myös potilaan saatavilla kyseisestä järjestelmästä. (Komulainen;Vuokko;& Mäkelä, 2011)

Dokumentin samanaikainen muokkaaminen useassa eri paikassa oli mahdollista, joten käyttöliittymän tuli tukea kahden eri dokumenttiversion tietosisällön yhdistämistä. Dokumenttien tietosisällön yhdistelyyn ei saa käyttää automaatiota, koska tämä voisi mahdollisesti vaarantaa potilasterveyden.

Opinnäytetyön toteutusosuutta aloittaessa, oli terveys- ja hoitosuunnitelman käyttöliittymä toteutus ja taustajärjestelmien toiminnot pääosin tehty (CRUD). Eroavien osioiden korostuselementit olivat myös toiminnassa.

Vertailutila -komponentti oli toteutettu siihen pisteeseen, että Kanta-palvelusta saatiin viimeisin versio tarkasteltavaksi käyttäjän luonnoksen vierelle. Vertailutilasta päästään pois hylkäämällä- tai tallentamalla luonnos uutena dokumenttina Kanta-palveluihin. Luonnoksen muokkaaminen ennen tallentamista on mahdollista.

Käyttäjä ei pystynyt kopiomaan kompleksista tietoa uudemmasta dokumentista, eikä tekstien yhdistelyyn ollut avustavaa toiminnallisuutta. Tietorakenteet täytyi luonnokselle luoda alusta asti uudestaan ennen kopioidun tiedon liittämistä. Tiedon siirto oli mahdollista vain käyttäen perinteisiä kopioi sekä liitä toimintoja.



Kuva 1. Terveys- ja hoitosuunnitelman yhdistelytila ennen avustavia toimintoja.

1.4 Opinnäytetyön tavoitteet ja aiheen rajaus

Opinnäytetyö rajattiin avustavien toimintojen käyttöliittymän suunnitteluun ja sen toteutukseen liittyviin vaiheisiin.

Tavoitteena oli toteuttaa terveys- ja hoitosuunnitelman käyttöliittymän vertailutilaan tiedon yhdistelyä helpottavia toimintoja. Toteutuksen jälkeen käyttäjän ei tarvitsisi luoda dokumentissa olevia rakenteita uusiksi, kun tietoa halutaan siirtää dokumentilta toiselle. Tiedon yhdistelyn ja kopioinnin tulisi olla helpompaa, kuin käyttäen perinteisiä kopioi ja liitä toimintoja.

Aikataulutavoitteena oli saada toteutus osa valmiiksi noin kuukaudessa. Tekstin yhdistelyyn käytettävä komponentti oli saatava valmiiksi tässä ajassa. Ajan riittäessä, oli tarkoitus toteuttaa toiminnallisuus tiedon kopioimiseksi Kanta-dokumentista käyttäjän luonnokseen.

Toimintojen toteutusta rajaavat vaatimukset olivat seuraavat:

- Kaikkien tietoja muokkaavien toimintojen tulee vaatia käyttäjän interaktiota, koska automaattista tiedon muokkaamista ei saa tehdä.
- Käyttöliittymän suunnittelussa pyritään samankaltaisuuteen aikaisempien toteutusten kanssa, jotta mahdollisimman moni toiminto tuntuisi käyttäjälle ennalta tutulta.
- Toimintojen toteutukseen tulisi käyttää käyttöliittymän kehitysympäristöstä löytyviä kirjastoja, jotta siihen ei tarvitsisi tuoda enempää riippuvuuksia.

2 KÄYTETYT TYÖKALUT JA TEKNOLOGIAT

2.1 Työkalut

2.1.1 Visual Studio Code

Kyseessä on avoimeen lähdekoodiin perustuva edistynyt lähdekoodi editori. (Microsoft, 2021)

Kyseinen työkalu on valikoitunut yrityksen pääasialliseksi käyttöliittymän kehitystyökaluksi hyvän TypeScript ja Angular tukensa ansiosta. Koodin formaattisäännöistä pidetään kiinni Prettier ja ESLint lisäosilla, jotka ovat editoriin saatavilla sen sisäänrakennetusta kauppapaikasta.

2.1.2 Git

Git on versionhallintajärjestelmä, jonka pääperiaate on kehittää koodia useissa paikallisissa haaroissa. Haarat pysyvät täysin itsenäisinä toisistaan, kunnes ne yhdistetään takaisin päähaaraan. (Scott Chacon, 2021)

Yrityksellä on käytössään oma Git infrastruktuuri, jota käytetään hyväksi tämän opinnäytetyön toteutuksessa.
Näille ominaisuuksille luotiin oma git-haara nimeämiskäytännöllä ”feature/<Jira tehtävän koodi>”. Lopulta kun koodi on katselmoitu ja toiminnot testattu hyväksytysti, viedään kyseinen haara pääkehityshaaraan. Sieltä ominaisuudet jatkavat matkaansa seuraavaan julkaisukandidaattiin.

2.1.3 Inkscape

Inkscape on ilmainen avoimenlähdekoodin vektorigrafiikka editori (Inkscape, 2021). Kyseistä editoria käytettiin vertailutilan avustavien toimintojen ikonien toteuttamiseen. Tämän lisäksi sillä generoitiin ikoneista optimoidut SVG-koodit.

2.2 Teknologiat

2.2.1 Angular

Angular on TypeScriptin päälle rakennettu sovelluskehys, johon on saatavilla hyvä valikoima valmiita kirjastoja verkkosovelluksen kehittämiseen. Se on suunniteltu erittäin skaalautuvaksi, helpoksi päivittää ja sillä on taustallaan laaja kehittäjien yhteisö (Angular, 2021)

Yrityksen verkkopohjainen käyttöliittymä rakentuu kyseisen sovelluskehyksen varaan, joten se toimi suunniteltavien toimintojen pohjana tässä opinnäytetyössä.

2.2.2 Angular Material

Kyseessä on Angular-komponenttikirjasto, joka tarjoaa myös valmiita toimintokehyksiä (CDK) omien komponenttien kehittämiseen (Angular Material, 2021).

Raahaa ja pudota toiminnallisuuksien kehittämiseen käytettiin tämän kirjaston toimintokehystä.

2.2.3 PrimeNG

PrimeNG on Angular spesifinen versio Primefaces komponenttikirjastosta, joka perustuu avoimeen lähdekoodiin. (Primefaces, 2021)

Tämä komponenttikirjasto on laajasti käytössä yrityksen verkkokäyttöliittymässä ja kyseisen kirjaston dialogikomponenttia käytettiin tekstin yhdistely komponentin toteutuksen pohjana.

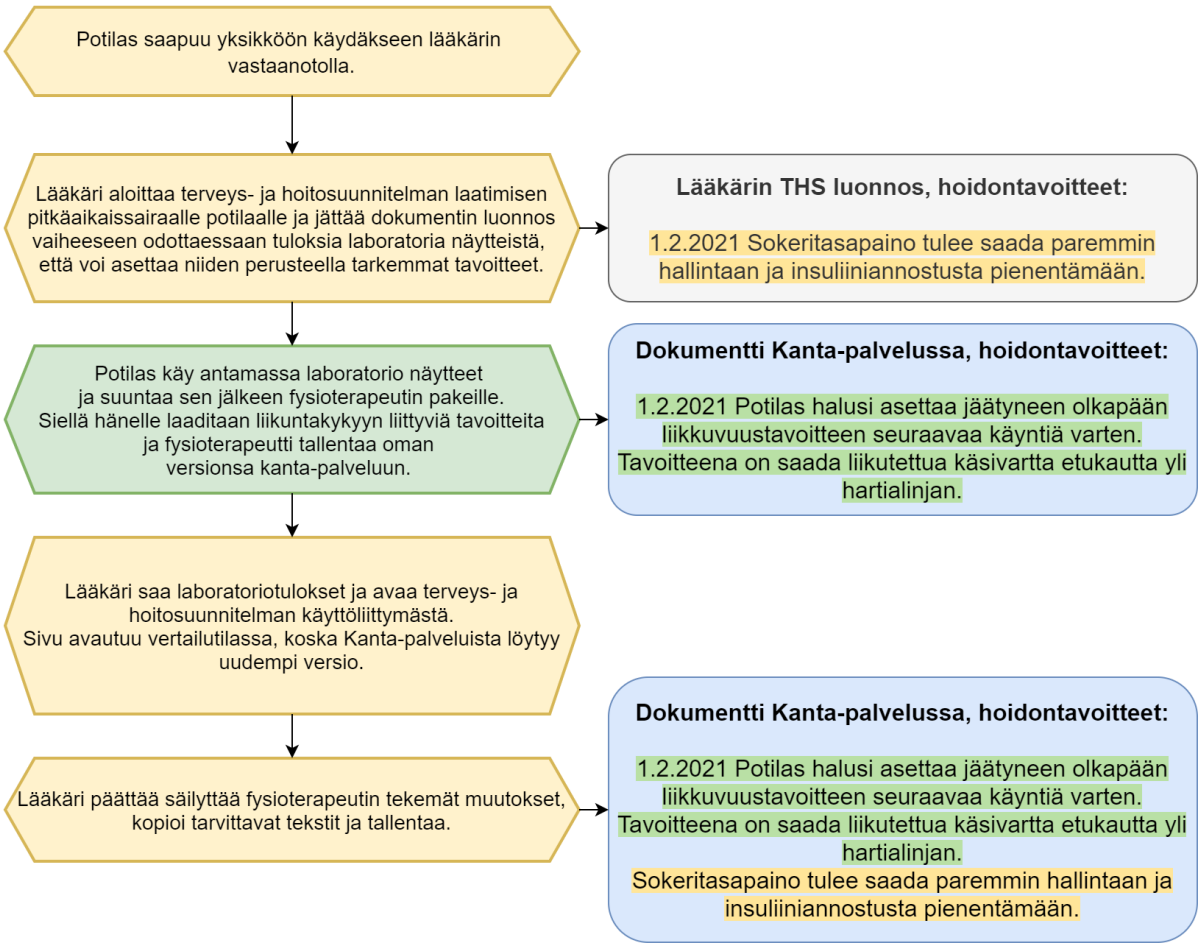
3 TOTEUTUS

3.1 Tekstisisällön yhdistely

Suurin osa terveys- ja hoitosuunnitelman tietosisällöstä on tekstisisältöä. Tästä johtuen kahden eroavan tekstin yhdistelyn helpottaminen oli priorisoituna tässä opinnäytetyössä.

Vertailutilaan päädytään luultavimmin tilanteissa, joissa muutoksia tehdään yksikön sisällä ja ammattilaiset täydentävät dokumenttiin potilaan kanssa sovittuja tavoitteita.

Seuraavassa kuviossa käydään läpi yksi mahdollinen käyttötapaus.



Kuva 2. Mahdollinen käyttötapaus.

3.1.1 Käyttäjälähtöinen suunnittelu

Käyttäjälähtöinen suunnittelu on iteratiivinen prosessi, missä suunnittelijat keskittyvät käyttäjiin ja heidän tarpeisiinsa suunniteltavien käyttöliittymien suhteen. Tavoitteena on osallistaa käyttöliittymän suunnittelu prosessiin asiantuntijoita mahdollisimman monialaisesti (esim. ohjelmoija, laitesuunnittelija, etnografi, jne.) loppukäyttäjien lisäksi. (Interaction Design Foundation, 2021)

Tekstin yhdistely komponentin toteutuksessa käytettiin mahdollisimman paljon käyttäjälähtöisen suunnittelun periaatteita. Täysin puhdasoppisesti ei periaatteita pystytty soveltamaan, koska loppupäänkäyttäjiä ei ollut saatavilla testauksen ja suunnittelun tueksi. Yrityksen sisältä kuitenkin löytyi henkilöitä, joilla oli kokemusta terveydenhuollon ammattilaisena toimimisesta. Näiden henkilöiden lisäksi käyttötestaus ryhmään otettiin mukaan terveys- ja hoitosuunnitelman käyttöliittymän suunnittelusta vastaava UX-suunnittelija sekä tuotepäällikkö.

Käyttäjälähtöinen suunnittelu vaati suunnitteluprosessin askelien läpikäymistä sekä vastauksien etsimistä tiettyihin kysymyksiin.

Kuten (Interaction Design Foundation, 2021) mainitaan. Askeleet käyttäjälähtöiseen suunnitteluun ovat:

Kuva 3. Käyttäjälähtöisen suunnitteluprosessin askeleet.

3.1.1.1 Ymmärrä käyttökonteksti

Ketkä käyttävät?

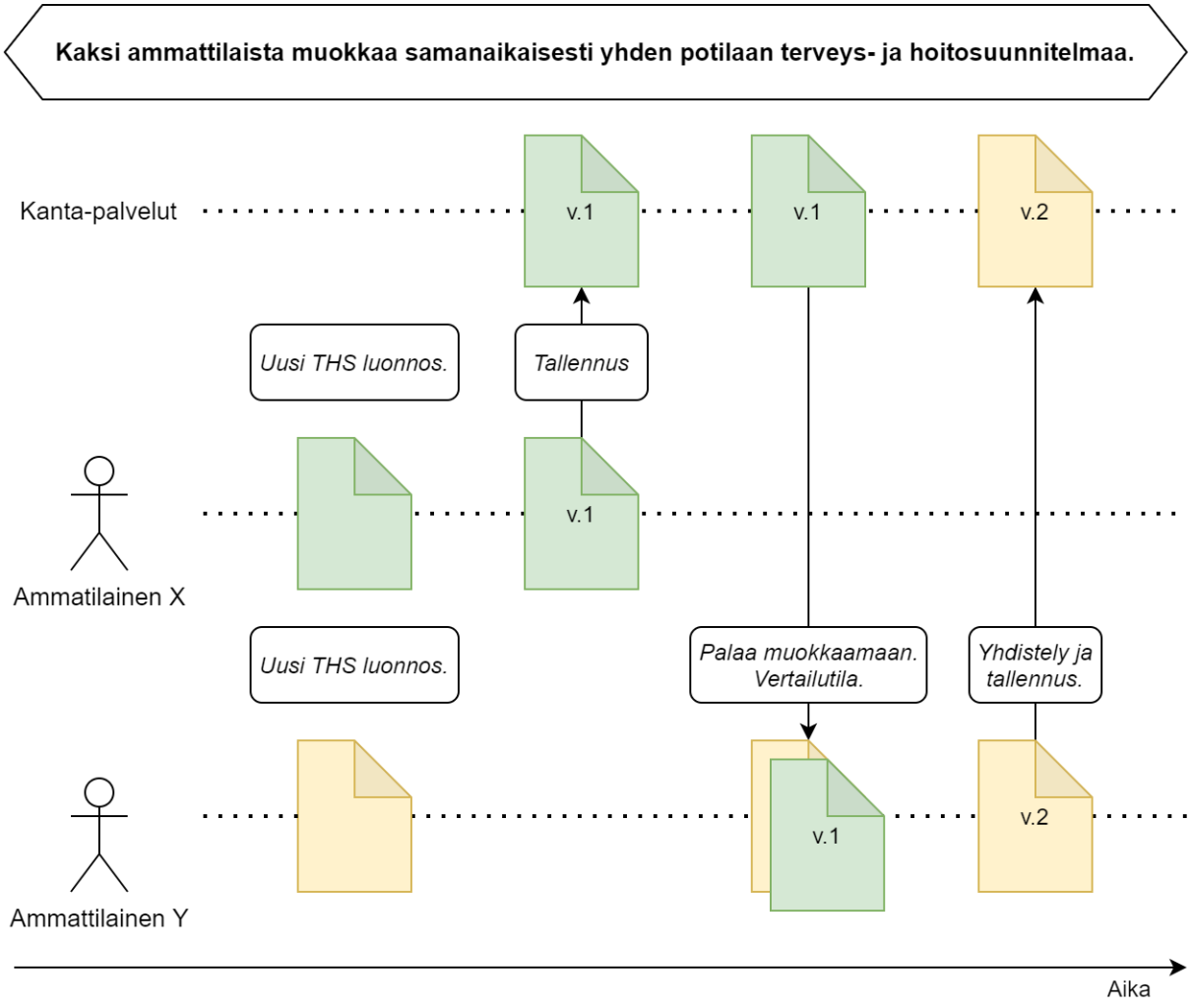
Käyttäjinä ovat lääkärit, sairaanhoitajat, terapeutit, fysioterapeutit jne. Oikeastaan kaikki ammattihenkilöt, jotka osallistuvat pitkäaikaisesti potilaan hoitoon.

Mitä varten he käyttävät tätä ominaisuutta?

He käyttävät tätä ominaisuutta kahden toisistaan poikkeavan tekstin yhdistelyyn. Tavoitteena on päivittää kyseinen tekstisisältö viimeisimmillä tiedoilla ja mahdollisesti poistaa vanhaksi jäänyttä tietoa.

Minkälaisessa tilanteessa he käyttävät sitä?

Pääsääntöisesti ammattilainen ja asiakas laativat yhteistyössä dokumentin sisältöä. Tämä voi tapahtua lääkärin vastaanotolla tai potilaan kotona (kotihoito). Päätelaitteena on tietokone tai tabletti.



Kuva 4. Kaksi ammattilaista muokkaa samanaikaisesti yhden potilaan terveys- ja hoitosuunnitelmaa.

3.1.1.2 Määritä vaatimukset

Mitä tarvitaan, että tuote tekee sen mitä sen on tarkoitus tehdä?

Tarvittiin työkalu, jolla voitiin yhdistää kahden eroavan tekstin sisältö helpommin kuin kopioimalla ja liittämällä. Tekstien erot tulisi pystyä tunnistamaan nopeasti, jotta käyttäjälle on selkeää mitkä osat ovat muuttuneet. Muokkaukset on pystyttävä perumaan, mikäli käyttäjä ei niitä halua ottaa käyttöön. Työkalu pitää pystyä avaamaan vertailutilassa yhdisteltävien tekstien kohdalta.

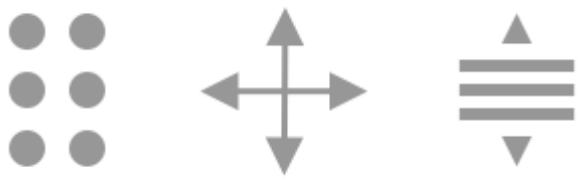
3.1.1.3 Luo ratkaisu tietojen pohjalta

Näiden tietojen pohjalta päädyttiin siihen lopputulokseen, että tekstit täytyy jakaa pienempiin kokonaisuuksiin. Näiden täytyi olla helposti muokattavissa tai liikuteltavissa tekstien välillä sekä niiden sisällä (lauseet). Lause palasten liikutteluun valitsin raahaa ja pudota toiminnallisuuden. Sen avulla palasten järjestely sekä kopiointi on mahdollista tehdä yhdellä liikkeellä. Ennen ensimmäisen ratkaisun toteutusta, täytyi tutustua raahaa ja pudota käyttöliittymien suunnitteluun sekä tehdä käyttöliittymästä ensimmäinen prototyyppi.

3.1.1.4 Käyttöliittymän ensimmäisen vedoksen suunnittelu

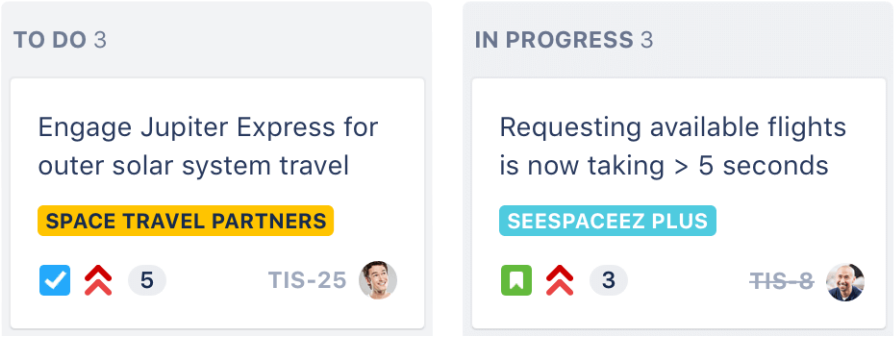
Kuten (Laubheimer, 2020) mainitaan, niin tärkeitä ominaisuuksia raahaa ja pudota käyttöliittymässä ovat seuraavat:

- Kaikki interaktiot ja raahattavat asiat ovat heti näkyvillä.
- Esineen valinta tapahtuu tutulla eleellä kuten pitämällä nappi painettuna esineen päällä.
- Raahaaminen tapahtuu suoraana edeltäneen eleen jatkumona, eli pidetään nappi painettuja ja liikutetaan osoitinta ruudulla.
- Pudotus vaiheessa käyttäjä lopettaa edellisissä vaiheissa käynnissä olleen eleen, jolloin esineen valinta raukeaa. Tämän jälkeen toiminto suoritetaan tai jätetään suorittamatta.
- Joissain tapauksissa on tärkeätä ilmaista käyttäjälle mitkä esineet ovat raahattavissa. Esimerkiksi ikonin avulla.

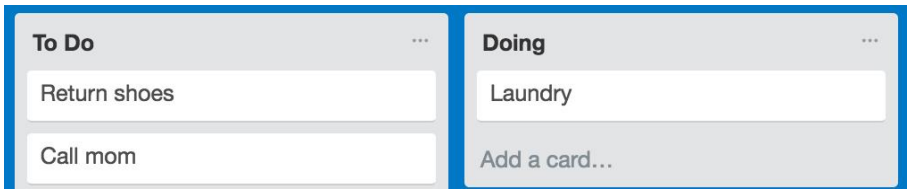


Kuva 5. Ikoneita raahattavuuden esittämiseksi. (Interaction Design Foundation, 2021)

Raahattavuuden ilmaisuun ei kuitenkaan ole muodostunut vakiintunutta käytäntöä (Interaction Design Foundation, 2021) ja monissa nykyaikaisissa käyttöliittymistä raahattavuutta ei kuvata ollenkaan ikonilla.



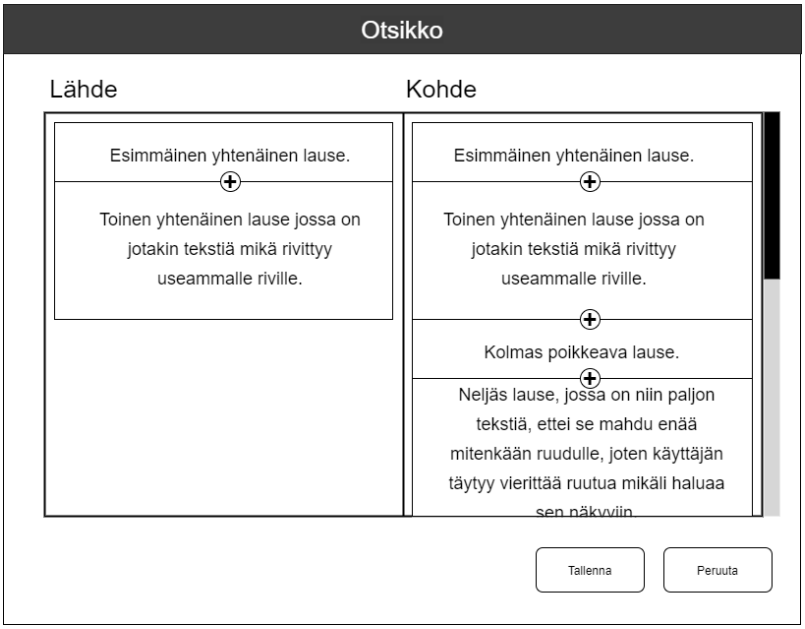
Kuva 6. Esimerkki Jiran käyttöliittymästä, missä ei ole raahattavuuden ikonia. (Atlassian, 2021)



Kuva 7. Esimerkki Trellon käyttöliittymästä, missä ei ole raahattavuuden ikonia. (Trello, 2021)

Käyttöliittymän ensimmäisestä vedoksesta jätettiin raahattavuuden ikonit pois lause-esineistä, jotta ruudulla olisi mahdollisimman vähän huomiota vaativia elementtejä.

Plusikoni lause-esineiden välillä antaa mahdollisuuden liittää nopeasti kaksi palasta yhdeksi kokonaisuudeksi. Tämä koettiin tarpeelliseksi, koska käyttäjät saattavat tuottaa tekstiä, jota ei voi pilkkoa lauseiksi virheellisen muodon takia.



Kuva 8. Dialogin ensimmäinen hahmotelma.

3.1.2 Lauseiden erottaminen tekstistä

Tavoitteena oli tuottaa algoritmi, joka pystyisi suhteellisen isolla luotettavuudella pilkkomaan tekstin lauseiksi. Alusta asti oli selvää, että tähän tullaan käyttämään säännöllistä lauseketta (Engl. regular expression).

Säännöllinen lauseke koostuu yhdestä tai useammasta säännöstä, joiden avulla tekstistä löydetään tiettyjä kohtia tai tekstin osia. Tavoitteena oli löytää tekstistä kohdat, jossa lause päättyy ja uusi alkaa.

Lähtökohdaksi valikoitui Stack Overflow foorumeilta löytynyt lauseke:

Kuva 9. Säännöllinen lauseke foorumeilta, joka tunnistaa kohdat, josta uusi lause alkaa. (Stack overflow, 2015)

3.1.2.1 Lauseke ei osannut tunnistaa lyhenteitä

Ensimmäisissä testeissä huomattiin, että lauseke ei osaa tunnistaa lyhenteitä vaan esittää uuden lauseen alkavaksi heti sen jälkeen.

```
Tämä on ensimmäinen lause.↓Heti perään on kirjattu toinen lause, jonka jälkeen
on rivinvaihto.↓
Rivinvaihdon jälkeinen kolmas lause sisältää lyhenteen tms.↓ääkkös sanalla heti
perään ja rivinvaihdon.↓
1.12.2020 Päivämäärällä alkava lause.↓
```

Testitekstin tulokset, missä osuma kohta on korvattu ↓ merkillä.

Tarvittiin keino suodattaa tuloksista pois kaavan ”<piste><väli><pienikirjain>”. Tämä onnistui lisäämällä seuraavanlainen sääntö:

Kuva 10. Sääntö lyhenteiden aiheuttamien osumien pois jättämiseksi.

Kyseessä on negatiivinen eteenpäin katsova sääntö, joka jättää osumista pois kaikki kuviota vastaavat kohdat.

Osuma lyhenteen jälkeen oli poissa. Tavoite saavutettu.

```
Tämä on ensimmäinen lause.↓Heti perään on kirjattu toinen lause, jonka jälkeen
on rivinvaihto.↓
Rivinvaihdon jälkeinen kolmas lause sisältää lyhenteen tms. ääkkös sanalla heti
perään ja rivinvaihdon.↓
1.12.2020 Päivämäärällä alkava lause.↓
```

Testitekstin tulokset, missä osuma kohta on korvattu ↓ merkillä.

3.1.2.2 Sulkujen sisällä oleva teksti oli mukana lauseiden tunnistuksessa

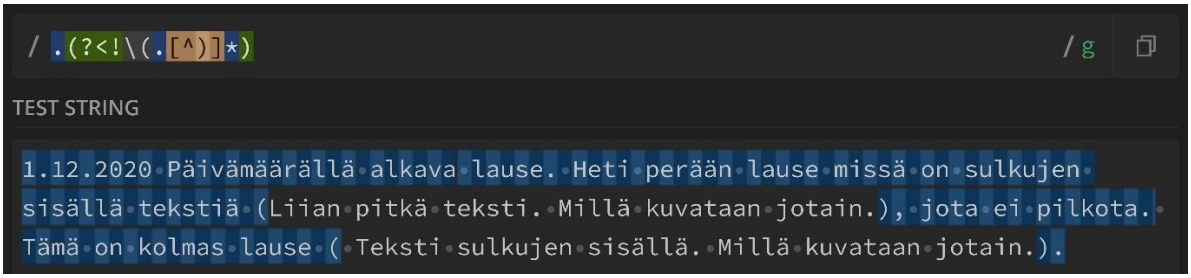
Sulkujen sisällä oleva lause tuotti väärän osuman. Tarvittiin sääntö mikä jätti sulkujen sisällä olevat osumat pois.

```
Tämä on ensimmäinen lause.↓Heti perään on kirjattu toinen lause, jonka jälkeen
on rivinvaihto.↓
Rivinvaihdon jälkeinen kolmas lause sisältää lyhenteen tms. ääkkös sanalla heti
perään ja rivinvaihdon.↓
1.12.2020 Päivämäärällä alkava lause.↓Heti perään lause missä on sulkujen si-
sällä tekstiä (Liian pitkä teksti.↓Millä kuvataan jotain.), jota ei pilkota.↓
```

Testitekstin tulokset, missä osuma kohta on korvattu ↓ merkillä.

Koska pois jätettävä osuma oli sulkujen sisällä, sitä täytyi tekstissä edeltää avaava kaarisulku ”(”. Tätä logiikkaa käytettiin säännön muodostamisen pohjana. Ratkaisuksi saatiin seuraava sääntö:

Kuva 11. Sulkujen sisältä löytyvät osumat pois suodattava lause.



Kuva 12. Sääntö testattuna lausekkeessa minkä ensimmäinen sääntö täsmää kaikkiin merkkeihin (ei rivinvaihto). Seuraavalla jätetään pois sulkujen sisällä olevat osumat.

Testitekstissä, väärä osuma sulkujen sisällä jäi pois.

```
Tämä on ensimmäinen lause.↓Heti perään on kirjattu toinen lause, jonka jälkeen
on rivinvaihto.↓
Rivinvaihdon jälkeinen kolmas lause sisältää lyhenteen tms. ääkkös sanalla heti
perään ja rivinvaihdon.↓
1.12.2020 Päivämäärällä alkava lause.↓Heti perään lause missä on sulkujen si-
sällä tekstiä (Liian pitkä teksti. Millä kuvataan jotain.), jota ei pilkota.↓
```

Testitekstin tulokset, missä osuma kohta on korvattu ↓ merkillä.

Kuva 13. Ennen uudelleen kirjoitusta.

3.1.2.3 Lausekkeen viimeistely

Säännöllisten lausekkeiden toiminnan tullessa tutuksi, tunnistin pohjana olleessa lausekkeessa turhia osioita ja päätin kirjoittaa sen kokonaan uusiksi. Samalla lisäsin vaihtoehtoisia lauseen päättäviä merkkejä, kuten kysymysmerkki. Lisäsin myös säännön, joka jättää lyhyet päivämäärät sekä piste-erottimella annetut luvut pois osumista.

Kuva 14. Lauseke uudelleen kirjoituksen jälkeen.

Lausekkeen ensimmäisellä säännöllä suodatetaan pois lyhennetyistä päivämääristä syntyvät osumat (esim. 2.4. tai 12.12) ja piste-erottimella syötetyt numeroarvot (esim. 1.99).

```
(?<!\d\.\d\.?)
```

```
(?<! )   - Määrittelee negatiivisen taaksepäin katsovan säännön.
\d       - Täsmää numeroihin 0-9
\.       - Täsmää pistemerkkiin.
\.?      - Täsmää pistemerkkiin nolla tai yhden kerran.
```

Toisella säännöllä suodatetaan osumista pois ne, jotka sijoittuvat sulkujen sisälle tekstissä.

```
(?<!\(.[^)]*)
```

```
(?<! )   - Määrittelee negatiivisen taaksepäin katsovan säännön.
\(       - Täsmää aloitus kaarisulkeiseen.
.        - Täsmää kaikkiin muihin paitsi rivinvaihto merkkeihin.
           Tässä säännössä huolehtii siitä, että avaava kaarisulkeinen voi vielä
           olla täsmättävissä myöhemmin.
[^)]*    - Täsmää nolla kertaa tai loputtomiin mihin tahansa merkkiin,
           jota ei ole listan sisällä, eli sulkeva kaarisulkeinen.
```

Kolmas sääntö varmistaa sen, että osumaa edeltää piste, kysymysmerkki tai huutomerkki ja sulkeutuva kaarisulkeinen.

```
(?<=\.|\?|!|\.[\)”"'’])
```

```
(?< )      - Määrittelee positiivisen taaksepäin katsovan säännön.
|          - Tai ehdon merkki.
\.         - Täsmää pisteeseen.
\?         - Täsmää kysymysmerkkiin.
!          - Täsmää huutomerkkiin.
\)         - Täsmää sulkeutuvaan kaarisulkeiseen.
\.[\)”"'’] – Täsmää pisteeseen, jota seuraa jokin hakasulkeiden sisällä oleva
             merkki.
```

Neljäs sääntö suodattaa osumista pois lyhenteiden aiheuttamat osumat (esimerkiksi: yms. tai jne.)

```
(?!\s[a-zööå])
```

```
(?! )     - Määrittelee negatiivisen eteenpäin katsovan säännön.
\s        - Täsmää kaikkiin välilyönti- ja rivinvaihtomerkkeihin.
[a-zööå] - Lista kirjaimista, joihin tämä sääntö täsmää. Kaikki englannin
           aakkoset(a-z) ja suomen ääkköset(ööå).
```

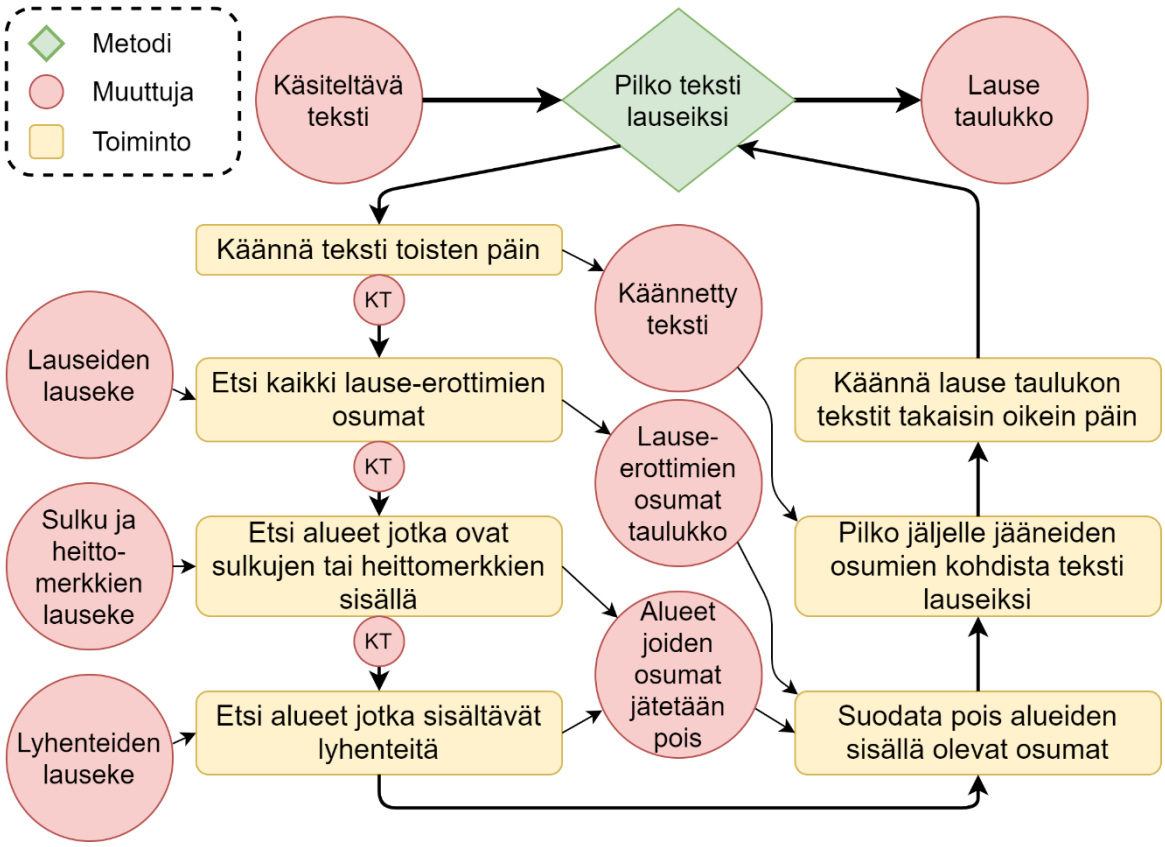
Viidennellä säännöllä rajataan referenssipisteet, josta aikaisemmat säännöt katsovat niille määritettyihin suuntiin. Tässä tapauksessa referenssipisteinä käytetään välilyönti- ja rivinvaihtomerkkejä.

```
\s
```

```
\s        - Täsmää kaikkiin välilyönti- ja rivinvaihtomerkkeihin.
```

3.1.2.4 IE11 ei tue taaksepäin katsovia sääntöjä

Projektin loppuvaiheilla selvisi, että Internet Explorer 11 selain ei tue JavaScriptin taaksepäin katsovia sääntöjä. Tämä tarkoitti sitä, että koko lauseke oli kirjoitettava uusiksi. Suurin osa säännöistä oli taaksepäin katsovia, joten oli mahdollista käyttää osaa vanhoista säännöistä kääntämällä käsiteltävä tieto toisten päin. Lauseiksi jakamisen logiikka oli kuitenkin pilkottava eri osa-alueisiin, koska yhden täydellisen lausekkeen kirjoittaminen ilman taaksepäin katsovia sääntö oli todella hankalaa, ellei mahdotonta.



Kuva 15. Muutettu lauseiksi pilkkomisen logiikka.

3.1.2.5 IE11 yhteensopiva säännöllinen lauseke

Useiden sääntöjen pääasiallinen sisältö saatiin pidettyä samana. Sääntöjen suunnat vaihdettiin eteenpäin katsoviksi ja orientaation omaavat merkit on vaihdettu niiden vastapariin. Tämä oli tarpeellista, koska merkkijonoa käsitellään toistenpäin käännettynä. Huomaa myös, että referenssi pisteiden rajaus tehdään nyt ensimmäisenä.

Kuva 16. IE11 yhteensopiva lauseiden tunnistus lauseke.

Esimerkki käännetystä lauseesta:

```
Tämä on ensimmäinen lause. Heti perään on kirjattu toinen lause, jonka jälkeen
on rivinvaihto.
Rivinvaihdon jälkeinen kolmas lause sisältää lyhenteen tms. ääkkös sanalla heti
perään ja rivinvaihdon.
1.12.2020 Päivämäärällä alkava lause. Heti perään lause missä on sulkujen si-
sällä tekstiä (Liian pitkä teksti. Millä kuvataan jotain.), jota ei pilkota.
```

```
.atoklip ie atoj ,).niatoj naatavuk älliM .itsket äktip naiiL( äitsket älläsis
nejuklus no ässim esual näärep iteH .esual avakla älläräämäviäP 0202.21.1

.nodhiavnivir aj näärep iteh allanas sökkää .smt neetnehyl äätläsis esual samlok
nenieklä j nodhiavniviR

.othiavnivir no neekläj aknoj ,esual neniot uttajrik no näärep iteH .esual ne-
niämmisne no ämäT
```

Kuten edeltävästä esimerkistä käy ilmi, täytyi sääntöjen suhteen ajattelua hieman muuttaa. Sulku avataan sulkevalla sulkumerkillä ja lopetetaan avaavalla sulkumerkillä. Vastaava tilanne on kaikilla orientaation omaavilla merkeillä.

Heitto- ja sulkumerkkien sisällä olevien tekstien tunnistukseen toteutettiin erillinen lauseke. Vaikka sääntö on pitkä, siinä toistetaan samaa yksinkertaista kaavaa. Määritetään aloitusmerkki, sen jälkeen saa tulla mitä tahansa merkkejä, kunnes vastaan tulee aloitusmerkin vastinpari.

Lauseiden tunnistus lausekkeen löytämistä osumista suodatetaan pois osumat, jotka sijaitsevat heitto- ja sulkumerkkien lausekkeen löytämillä alueilla.
Esimerkkilauseessa tämä alue olisi :

```
).niatoj naatavuk älliM .itsket äktip naiiL(
```

Kuva 17. Sulku- ja heittomerkkien lauseke. Löytää tekstit, jotka ovat näiden merkkien sisällä.

```
 \).*\(     - Täsmää sulkeutuvaan kaari sulkeiseen, jota seuraa x-määrä mitä
              tahansa merkkejä, kunnes avaava kaarisulkumerkki tulee vastaan.
 |          - Tai ehto.
```

Lyhenteiden tunnistukseen lisättiin tuki useammalle eri lyhennystavalle. Tavoitteena oli tällä lauseella karsia pois pisteitä käyttävät lyhenteet osumista, joita lauseiden tunnistus lauseke on mahdollisesti löytänyt. Lyhenteet täytyi löytää takaperin käännetystä tekstistä, joten säännöt hakevat takaperin olevia lyhenteiden kuvioita.

Kuva 18. Lyhenteiden lauseke. Löytää käännetystä tekstistä alueet, joilla on lyhenteitä.

Kaikki säännöt on sijoitettu ryhmittävän syntaksin sisään. Ryhmän sisällä olevan säännön täytyy täyttyä, että osuma löytyy.

```
(?:<säännöt>)
```

```
(?: )      - Ryhmittää useita sääntöjä yhteen ilman, että luo sieppaavan ryhmän
```

Lausekkeen ensimmäinen sääntö löytää lyhenteet: ’jKr.’,’eKr.’, …

```
\.[a-zöää][A-ZÖÄÅ][a-zöää]
```

```
 \.         – Täsmää piste merkkiin.
[a-zöää]    - Täsmää pieneen kirjaimeen.
[A-ZÖÄÅ]    - Täsmää isoon kirjaimeen.
```

Toinen sääntö etsii lyhenteet: ’esim.’, ’tms.’, …
Tämän säännön määrittelyssä tuli olla erityisen tarkkana. Lauseen lopussa olevat sanat, jotka ovat kahdesta neljään merkkiä pitkiä, olivat vaarassa tulla säännön tunnistamiksi. Tämän takia lyhenteen jälkeen odotetaan välilyöntiä ja sen jälkeinen merkki ei saa olla isokirjain tai rivinvaihtomerkki.

```
[^A-ZÖÄÅ\r\n]\s\.[a-zöää]{2,4}\s{1}
```

```
\[          - Avaava kaarisulkeinen.
\s          - Täsmää kaikkiin välilyönti- ja rivinvaihtomerkkeihin.
\.          - Täsmää pistemerkkiin.
\r          - Täsmää paluumerkkiin (rivinvaihto).
\n          - Täsmää rivinvaihtomerkkiin.
[^A-ZÖÄÅ\r\n] Täsmää, jos hakasulkujen sisällä olevaa merkkiä ei löydy kohdasta.
[a-zöää]{2,4} Täsmää pieniin kirjaimiin, joita on 2-4kpl
\s{1}       - Välilyönti- tai rivinvaihtomerkki, joita on tasan yksi.
```

Kolmas sääntö löytää lyhenteet : ’N.N.’, ’T.N.T.’, …

```
(:?\.[A-ZÖÄÅ]){2,}
```

```
(?: )       - Ryhmittää useita sääntöjä yhteen ilman, että luo sieppaavan ryhmän
\.[A-ZÖÄÅ]  - Täsmää pistemerkkiin, jota seuraa isokirjain.
{2,}        - Edeltäneen kuvion täytyy toistua kahdesti tai useammin.
```

Viimeinen sääntö löytää esimerkiksi lyhenteen ’o.s.’ ja vastaavat pidemmät ketjutukset.

```
(:?\.[a-zöää]){2,}
```

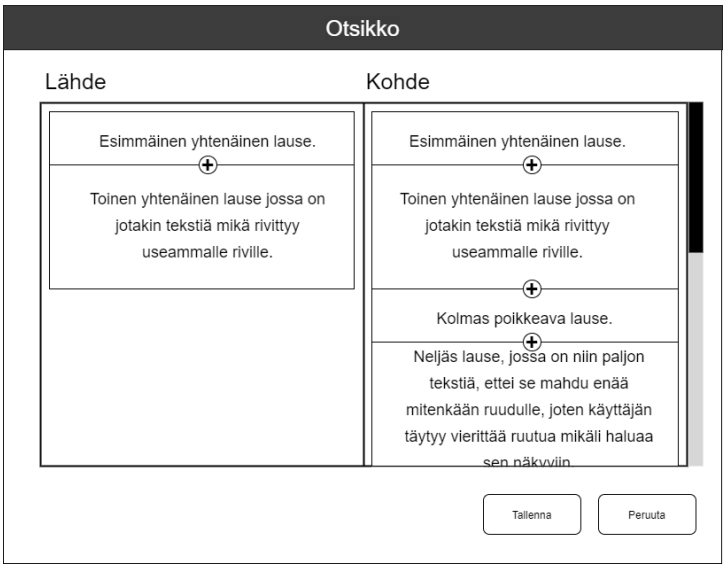
```
(?: )       - Ryhmittää useita sääntöjä yhteen ilman, että luo sieppaavan ryhmän
\.[A-ZÖÄÅ]  - Täsmää pistemerkkiin, jota seuraa isokirjain.
{2,}        - Edeltäneen kuvion täytyy toistua kahdesti tai useammin.
```

3.1.3 Tekstin yhdistely dialogi -komponentin käyttöliittymän kehitysvaiheet

Ensimmäisessä prototyypissä kaksi raahattavien elementtien listaa oli sijoitettu rinnakkain dialogi elementin sisälle. Listat oli sijoitettu rullattavaan ikkunaan, koska lausekomponentteja oli mahdollista olla enemmän kuin ruudulle mahtuu.

Plusikoni lause-esineiden välillä antaa mahdollisuuden liittää nopeasti kaksi palasta yhdeksi kokonaisuudeksi. Tämä nähtiin tarpeelliseksi väärän lausejaon tilanteissa. Samalla tämä ominaisuus mahdollistaisi isompien kokonaisuuksien muodostamisen. Näitä isompia kokonaisuuksia olisi sitten helppo järjestää uusiksi muuhun tekstiin nähden.

Poisto ja muokkaus toimintojen suunnittelu jätettiin myöhemmälle. Oli parempi odottaa, että raahaa ja pudota toimintojen rajoitteet hahmottuvat paremmin.

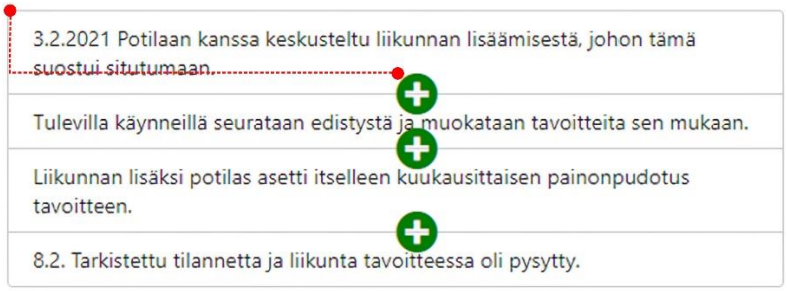


Kuva 19. Dialogin ensimmäinen hahmotelma.

Komponentille luotiin oma Angular kehitysympäristö, jonka sisällön pystyi julkaisemaan verkkoon käyttötestauspalavereita varten. Ympäristöön tuotiin kaikki projektin kannalta oleelliset riippuvuudet, kuten PrimeNG, Angular Materials CDK ja Font Awesome-ikonit.

3.1.3.1 Lausekomponentit

Ensimmäisen prototyypin käytännön toteutus aloitettiin lausekomponenteista. Komponentin juurielementissä käytettiin relatiivista asettelua, jotta plusikoni voidaan asetella sen aseman suhteen. Ikonin juurielementissä käytettiin absoluuttista asettelua, joka poistaa elementin perinteisestä dokumentti asettelusta (engl. document flow). Ikoni-komponentti ei siis vie tilaa elementtien asettelussa.

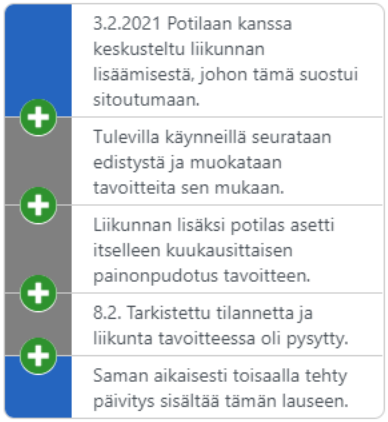


Kuva 20. Ensimmäinen versio lausekomponentista. Korostettuna ikonielementin referenssi piste.

Nopeasti kävi selväksi, että plus ikoni ei voi sijaita komponenttien keskellä. Siinä se veisi vertikaalisesti turhan paljon tilaa ja siksi se siirrettiin komponentin vasempaan reunaan.

Kuva 21. Toinen versio lausekomponentista.

Samalla kokeiltiin toimisiko värillinen alue myös indikaattorina. Harmaa väri kertoo, että teksti löytyy vastaavassa muodossa käyttäjän luonnoksesta. Sininnen taas eroavasta tekstistä.

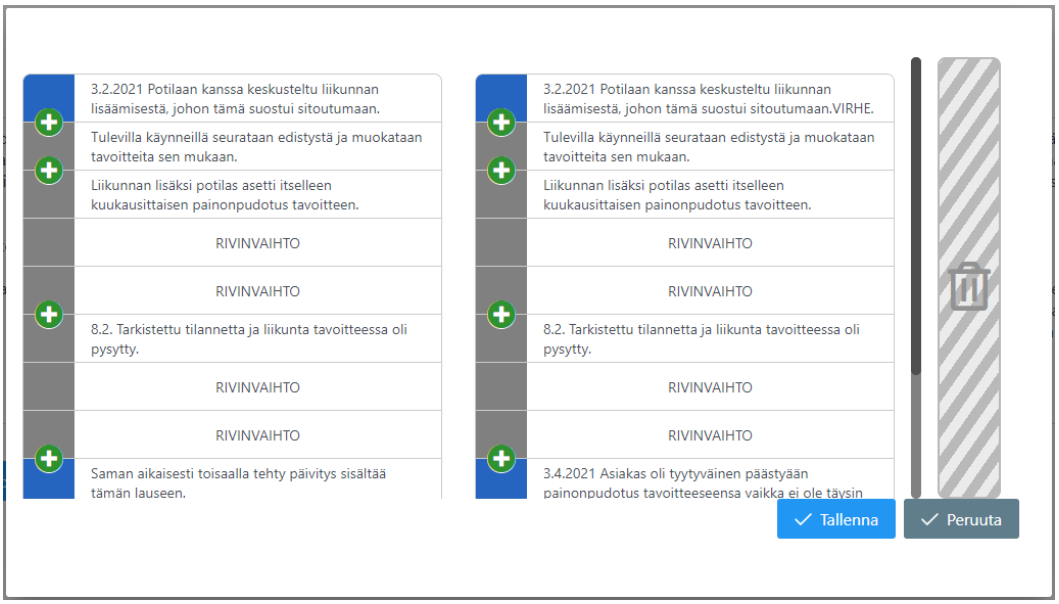


Kuva 22. Toimintoalue käytössä eroavaisuuksien indikaattorina.

Seuraavana toteutettiin dialogin pohjarakenteet ja poisto toiminnallisuus. Lauseen poistaminen tapahtui raahaamalla se kohde puolelta (oikealla) roskakorialueelle ja pudottamalla esine sinne. Alue korostettiin punaisella, kun raahattava esine oli sen yllä.

Kuva 23. Poistotoiminnon ensimmäinen versio.

Lauseiden välisille rivinvaihto merkeille lisättiin omat elementtinsä, jotta voitiin kokeilla, koetaanko niiden nopea muokkaaminen tarpeelliseksi. Ulkoasun hiomiseen ei tässä vaiheessa käytetty aikaa.



Kuva 24. Nopea testi rivinvaihtojen visualisoinnista käyttötestipalaveria varten.

3.1.3.2 Käyttötestipalaverit

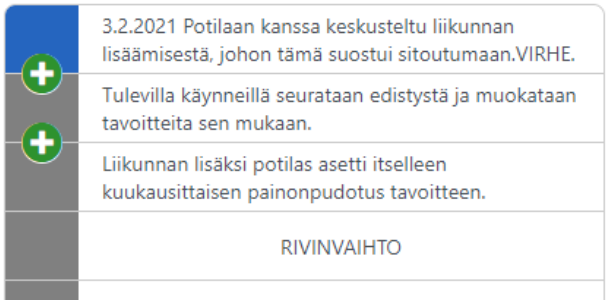
Käyttäjälähtöisen suunnittelun periaatteiden mukaisesti käyttöliittymän suunnitteluun pyritään osallistamaan monipuolisesti eri alojen ammattilaisia. Tiimissämme oli mukana terveys- ja hoitosuunnitelman tuoteomistaja, UX-pääsuunnittelija sekä kaksi testaajaa, joista toisella oli kokemusta terveydenalan ammattilaisena toimimisesta.

Palavereita järjestettiin viikon välein kolmen viikon ajan. Jokaisessa palaverissa testaajat saivat viimeisimmän version dialogista testattavakseen. Dialogin käyttöön ei annettu ohjeita, jotta käyttöliittymän intuitiivisuutta voitiin paremmin havainnoida. Testaajia pyydettiin yhdistelemään valmiita testitekstejä ja kommentoimaan toimintojen puutteita tai epäselvyyksiä.

Palaverit etenivät opinnäytetyöntekijän johdolla, joka kirjasi samalla tärkeimpiä pointteja keskustelusta, käyttöliittymän puutteista sekä parannusehdotuksista. Palavereissa osallistujia rohkaistiin keskustelemaan ja ideoimaan positiivisessa hengessä. Kaikki ideat olivat tervetulleita ja niitä käytiin läpi palaverien aikana.

3.1.3.3 Ensimmäisen käyttötestipalaverin tulokset

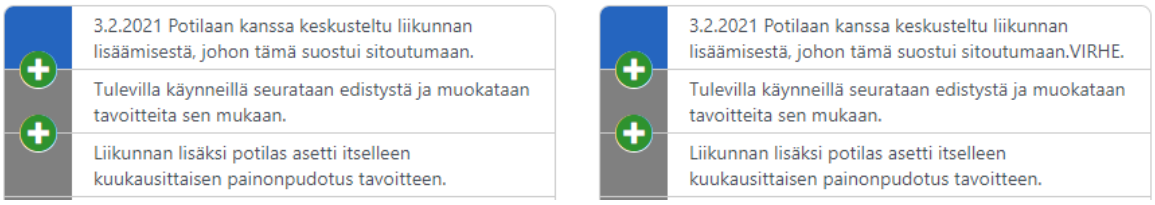
Palaverin alkupuolella kävi ilmi, että testaajilla oli vaikeuksia hahmottaa rivien välissä olevien plusikonien käyttötarkoitus ja ylipäänsä se mihin toimintoa tarvitaan. Suurin osa oli sitä mieltä, että toiminto voidaan poistaa, joten se siirrettiin sivuun.



Kuva 25. Kuvassa lausekomponentit, joissa turhaksi koettu plusnappi.

Hämmennystä testaajien keskuudessa aiheutti se, kun osaa lausekomponenteista pystyi raahaamaan ja osaa ei. Tämä oli kuitenkin tarkoituksellista, koska kyseessä oli teksti, joka jo löytyi kummaltakin listalta.

Tätä ei kuitenkaan käytännössä vielä indikoitu mitenkään käyttöliittymässä. Harmaata ja sinistä väriä käytettiin vain ilmaisemaan lauseiden eroavaisuuksia. Lausekomponenttiin tarvittiin siis selkeä visuaalinen indikaattori sille, mitkä niistä ovat raahattavissa.



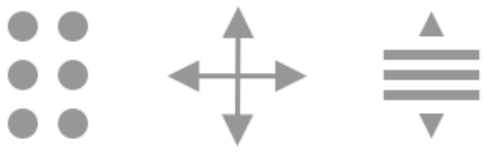
Kuva 26. Lausekomponentit, joista puuttuu indikaattori raahattavuudelle.

Lausekomponentin muokkaustilaan pääsemiseksi piti sen sisältämää tekstiä tupla klikata ja tämä tuntui joistain epäintuitiiviselta. Keskustelussa päädyttiin siihen, että perinteisen kynäikonin sijoittaminen lausekomponenttiin olisi yhdenmukainen tapa aikaisempien toteutuksien kanssa. Samalla se antaisi käyttäjille tuttuuden tunteen, koska he ovat vastaa toimintoa käyttäneet muuallakin.

Roskakorialueesta testaajat olivat montaa mieltä. Osan mielestä kyseessä oli hyvä ja selkeä ominaisuus. Toisten mielestä, oli vaikea hahmottaa mitä sinne voi ylipäänsä laittaa. Lopulta roskakorialue päätettiin korvata poistopainikkeella, joka sijoitettaisiin vierekkäin editointi painikkeen kanssa. Tämä poistaisi epäselvyyden siitä mitä voi poistaa ja se tarjoaisi tutun käyttöliittymä kokemuksen.

Kuva 27. Roskakorialue.

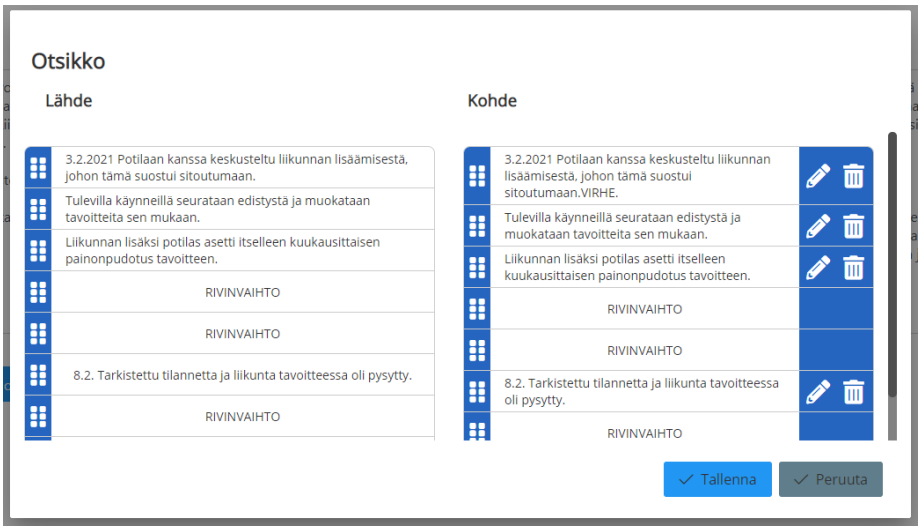
Kokouksessa todettiin, että lauseiden raahattavuuden ilmaisuun voisi löytyä ratkaisu käyttämällä raahattavuuden ikonia. Alla olevasta kuvasta vasemmanpuoleisin tartuntapintaikoni miellytti suurinta osaa testaajista. Jokin vastaava ikoni tuli siis ottaa käyttöön ennen seuraavaa käyttötestipalaveria.



Kuva 28. Esimerkki yleisistä raahattavuuden ikoneista. (Nielsen Norman Group, 23)

3.1.3.4 Käyttötestipalaverin palautteen mukaiset muokkaukset

Plusikoni ja roskakorialue poistettiin käytöstä. Lausekomponetteihin lisättiin Font Awesomen raahausikoni vasempaan reunaan. Kohdelistan lausekomponentteihin lisättiin kynäikoni muokkauksen aloittamista varten sekä roskakori-ikoni rivin poistamiseen. Dialogiin lisättiin myös otsikko ja aliotsikkoelementit.

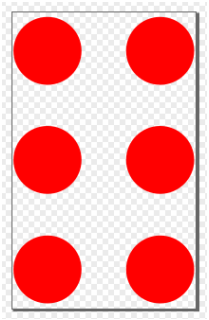


Kuva 29. Dialogi käyttötestipalaverin korjausten jälkeen.

3.1.3.5 Raahausikoni

Font Awesomin raahausikoni ei ollut tarpeeksi lähellä valittua ikonin muotoa, joten siitä täytyi tehdä oma versio. Ikonin grafiikat piirrettiin Inkscape-ohjelmalla, josta sen sai vietyä tiedostoon optimoituna SVG-koodina.

Kuva 30. Font Awesomen raahausikoni. (Font Awesome, 2021)



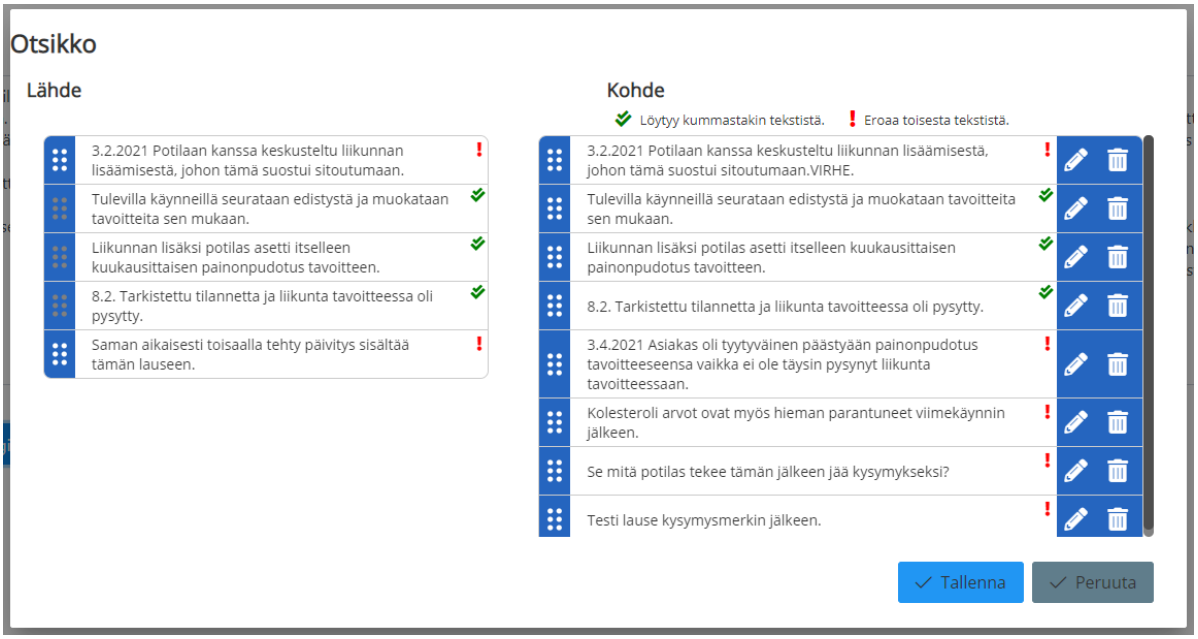
Kuva 31. Opinnäytetyön tekijän toteuttama raahausikoni.

3.1.3.6 Ei raahattavien elementtien ulkoasun selkeyttäminen

Ensimmäisen käyttötestipalaverin jälkeen vaikutti siltä, että yksinkertaisimmat ratkaisut ovat parhaita. Tämän johdosta erilliset rivinvaihto elementit poistettiin käytöstä. Kaikki lauseiden välissä olevat rivinvaihto- ja välilyöntimerkit sisällytetään edelliseen löydettyyn lauseeseen.

Testiryhmän jäsenet eivät ymmärtäneet miksi osa lausekomponenteista ei ollut raahattavissa, kun osa oli. Tämä johtui toiminnallisuudesta, missä kummastakin tekstistä löytyviä lauseita ei voi kopioida listalta toiselle. Lausekomponenttien vasemmanreunan käytöstä indikaattorina jouduttiin luopumaan, jotta nämä kaksi tilaa saatiin paremmin esitettyä.

Uudessa ratkaisussa käytettiin hyväksi raahausikonia ja sen värin muuttamista harmaaksi. Tietokoneella käytettäessä kursori muuttuu myös ei sallitun toiminnon kursoriksi. Sen lisäksi kaikkiin teksteihin lisättiin ikoni, joka symboloi niiden tilaa. Kohdelistan yläreunaan laitettiin esille selite näiden kahden ikonin tarkoituksesta.



Kuva 32. Ei raahattavien elementtien korostus ja tekstin eroavaisuuksien ikonit lisättynä.

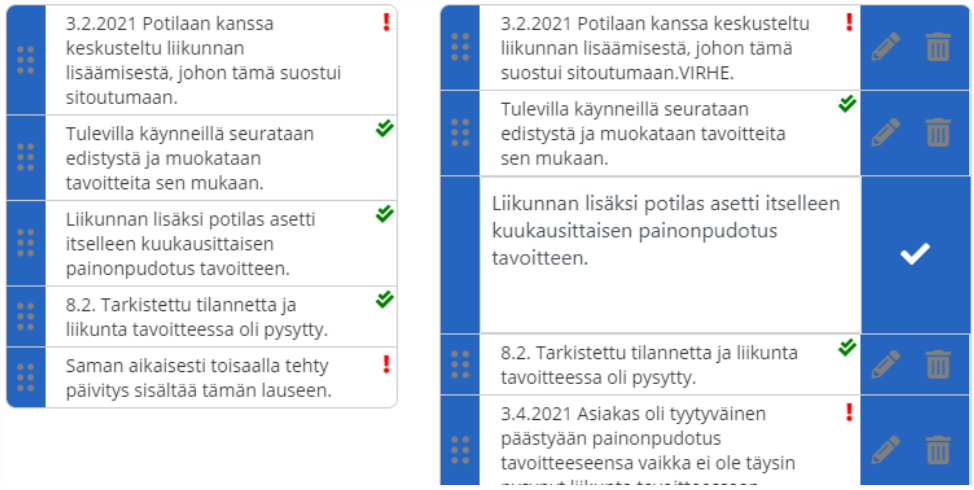
3.1.3.7 Ulkoasun hiomista ennen seuraavaa käyttötestipalaveria

Kohde puolelle sijoittui paljon tilaa vievää toiminnallisuutta ja tämän takia sen osuutta käyttöliittymän pinta-alasta lisättiin. Käyttäjien silmissä muutos myös korostaa elementtien hierarkiaa. Heille on siis selkeämpää, kumpi niistä on tärkeämpi ja se luontaisesti houkuttaa enemmän huomiota (Nielsen Norman Group, 2021).

Listojen vieritettävälle ikkunalle lisättiin varjostus, jotta se selkeämmin erottuu omaksi alueekseen. Dialogin ulkoasua tuotiin myös lähemmäksi yrityksen yleistä dialogin rakennetta.

Kuva 33. Ulkoasu ennen toista käyttötestipalaveria.

Uutena toiminnallisuutena lisättiin kumoa ja palauta napit. Tämä edellytti muutosta lausekomponenttien muokkaustoimintoon, joka piti rajata yhden tekstin muokkaamiseen kerrallaan. Kaikki muut toiminnot ovat myös poissa käytöstä sen aikaa, kun muutoksia tehdään.



Kuva 34. Lausekomponentti muokkaustilassa ja muut toiminnot poistettu käytöstä.

3.1.3.8 Toisen käyttötestipalaverin tulokset

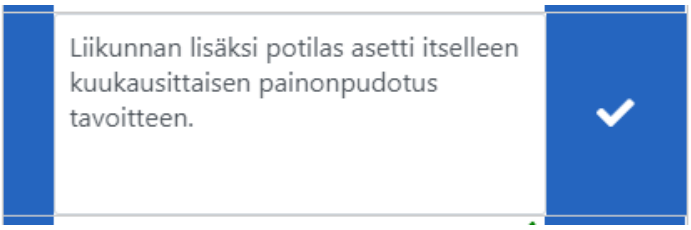
Dialogin värityksessä päätettiin kokeilla erillaista painotusta, koska ikonien taustojen värit veivät käyttäjältä turhaan huomiota. Tästä johtuen ikonit väritetään jatkossa yrityksen sinisellä päävärillä ja taustat valkoisella. Suuren kontrastieron värit korostavat myös visuaalista hierarkiaa, eikä niitä kannata käyttää paikoissa johon käyttäjän huomiota ei haluta viedä (Nielsen Norman Group, 2021).

Kumoa ja palauta nappien sijainti häiritsi suurinta osaa testaajista, joten ne päätettiin nostaa kohdelistan aliotsikon tasolle. Tämä muutos myös mahdollistaa matalamman dialogin alapalkin.

Osan mielestä raahausikonin selitteen lisääminen legendaan selkeyttäisi sen merkitystä varsinkin käyttäjille, jotka eivät ole vastaavaa ikonia koskaan nähneet. Kaikki olivat sitä mieltä, ettei siitä ollut haittaakaan, joten selite päätettiin lisätä.

Kuva 35. Dialogin tila toisen käyttötestipalaverin aikana.

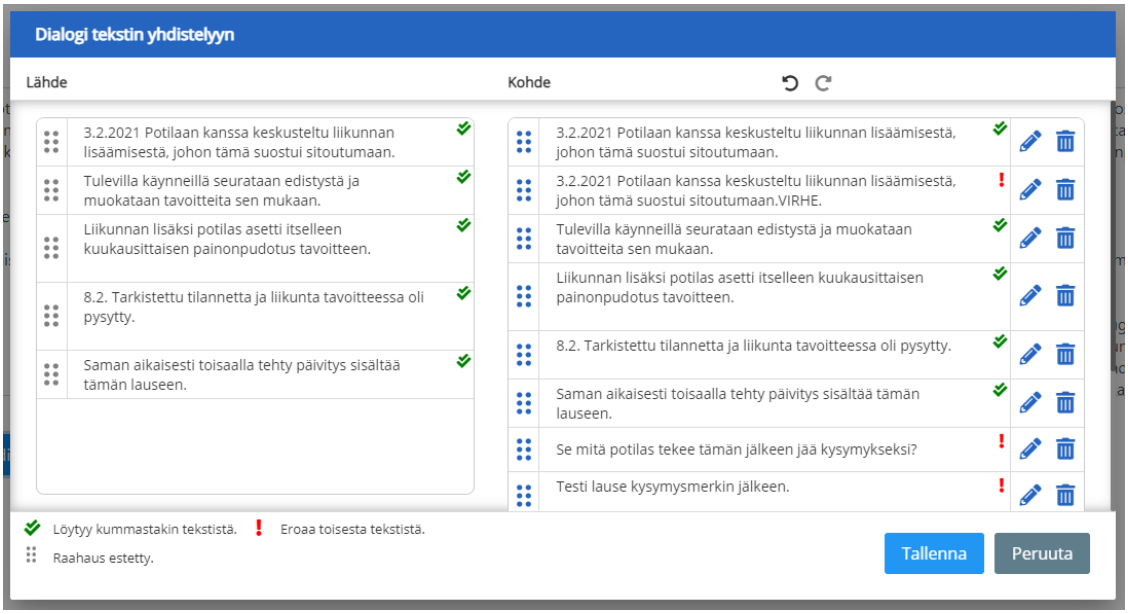
Yksi testaajista oli sitä mieltä, että muokkaustilainen lausekomponentti ei ole tarpeeksi korostetun näköinen. Hän toivoi muokattavan komponentin reunoille värikorostusta. Samaa korostus efektiä käytettii yrityksen taulukkojen riveillä, kun ne olivat valittuna. Yhdenmukaisuuden takia lisäys päätettiin toteuttaa.



Kuva 36. Lausekomponentti muokkaustilassa.

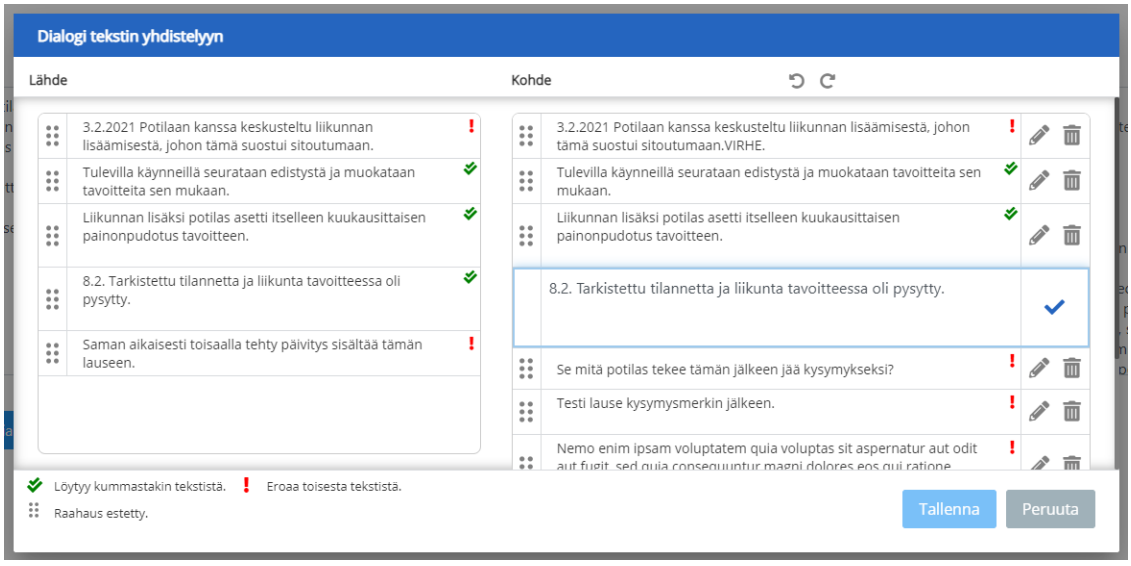
3.1.3.9 Toisen käyttötestipalaverin palautteen mukaiset muokkaukset

Palaverin korjausten lisäksi listat asetettiin ottamaan minimissään sisältöosan korkeuden verran tilaa. Kumoa ja palauta napit indikoivat toiminnon käytettävyydestä vaihtamalla ikonin väriä ja niille lisättiin vihjetekstit.

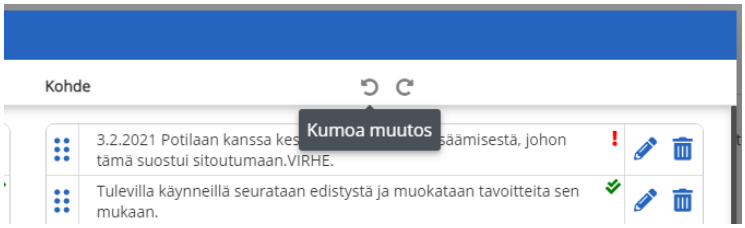


Kuva 37. Dialogi toisen käyttötestipalaverin mukaisten korjausten jälkeen.

Muokkaustilassa olevan lausekomponentin korostusta parannettiin reunojen hehkuefektillä.



Kuva 38. Paranneltu muokkaustilassa olevan lausekomponentin korostus.

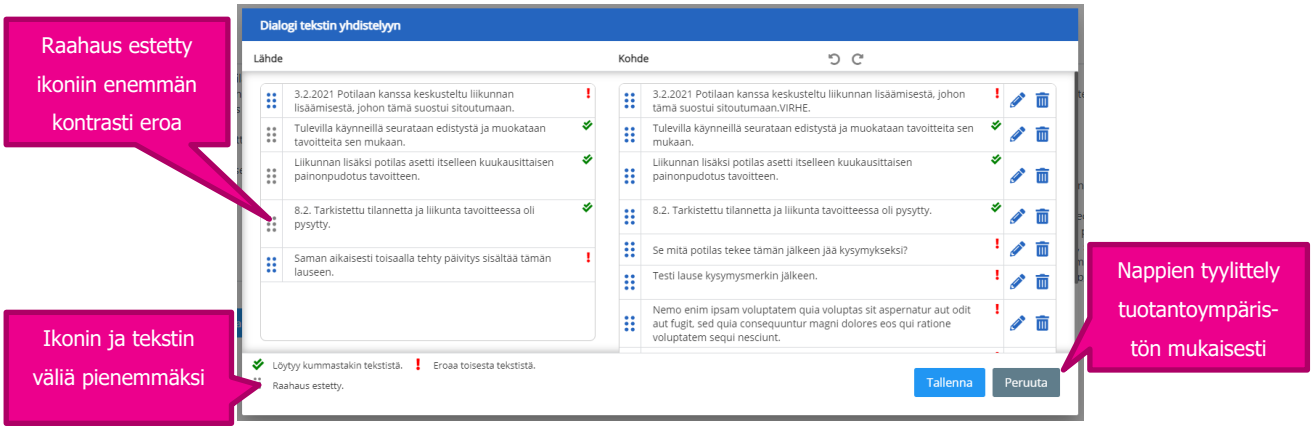


Kuva 39. Kumoa ja palauta nappien vihjetekstit.

3.1.3.10 Viimeinen käyttötestipalaveri

Palautetta tuli tässä palaverissa niukasti ja parannusehdotukset olivat pääasiassa pientä kosmeettista hiontaa.

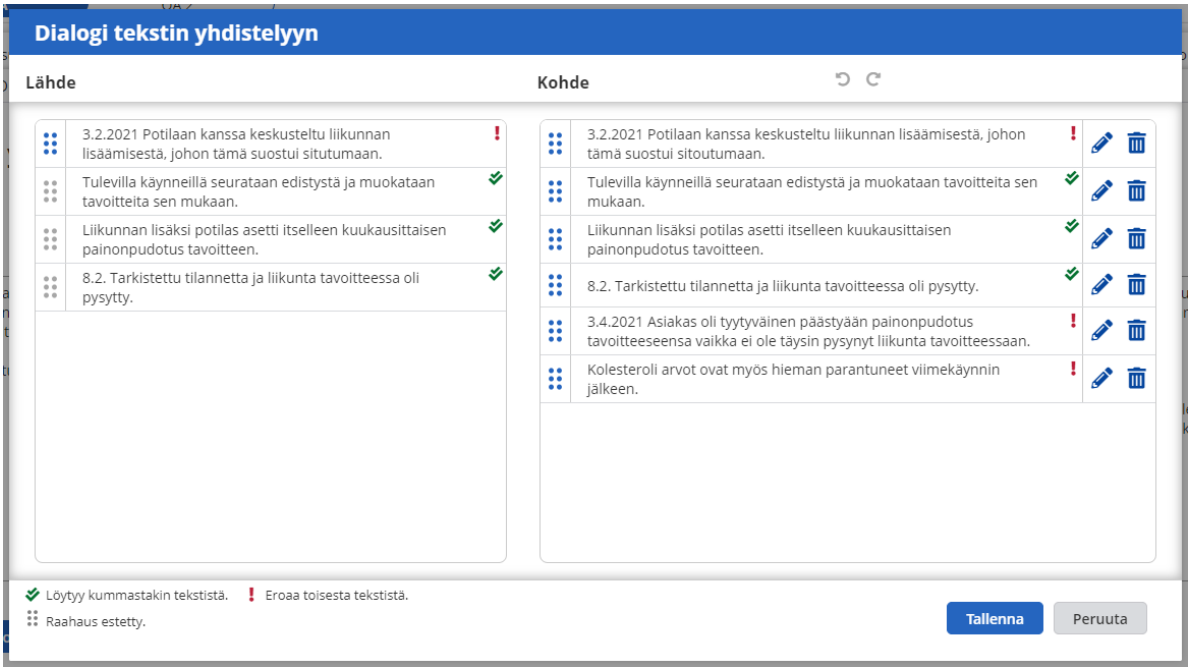
Testaajat kokivat, että ”raahaus estetty” tilaisen ikonin kontrasti ero ei ollut riittävän selkeä. Se tulisi korjataan lopulliseen versioon. Legendan teksteissä ikonin ja selitteen välinen matka koettiin liian suureksi. Tätä väliä tuli pienentää. Tallenna ja peruuta nappien tyylittely oli yhtenäistettävä tuotantoympäristön kanssa.



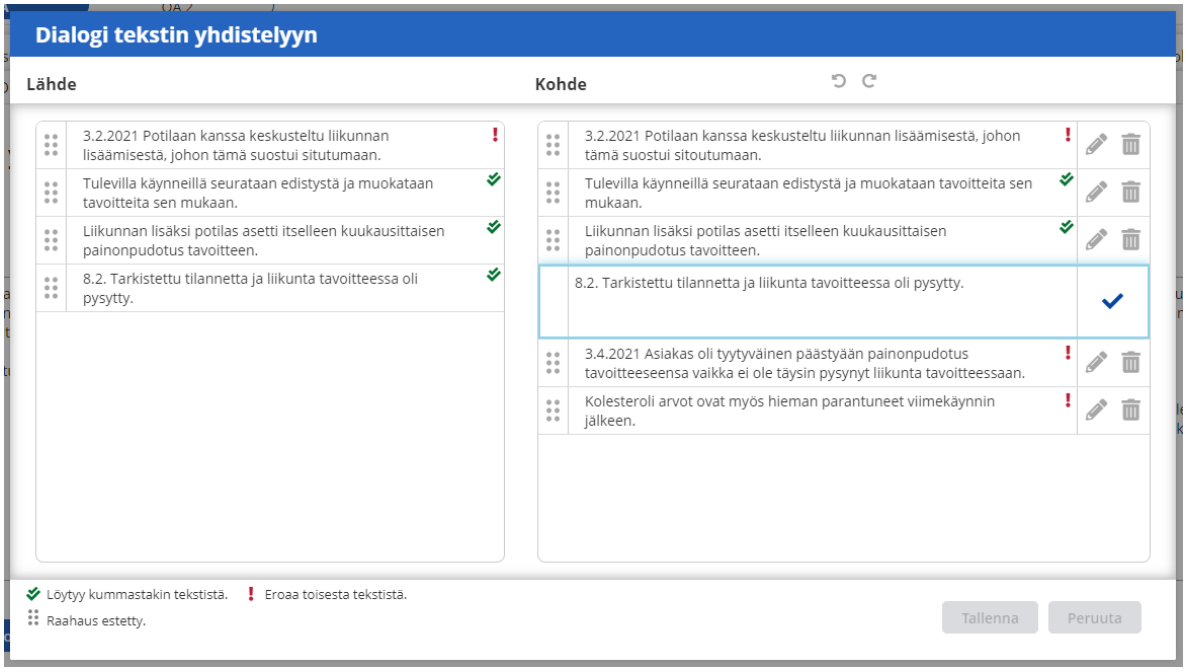
Kuva 40. Dialogin tilanne viimeisessä käyttötestipalaverissa.

3.1.3.11 Dialogin valmiina

Korjausten jälkeen komponentti vietiin yrityksen varsinaiseen kehitysympäristöön. Se lisättiin osaksi pienohjelmakirjastoa, missä muutkin yrityksen yleiskäyttöiset komponentit ovat. Lopulta dialogia esiteltiin firman laajuisessa demo tilaisuudessa ja vastaanotto oli todella positiivinen.



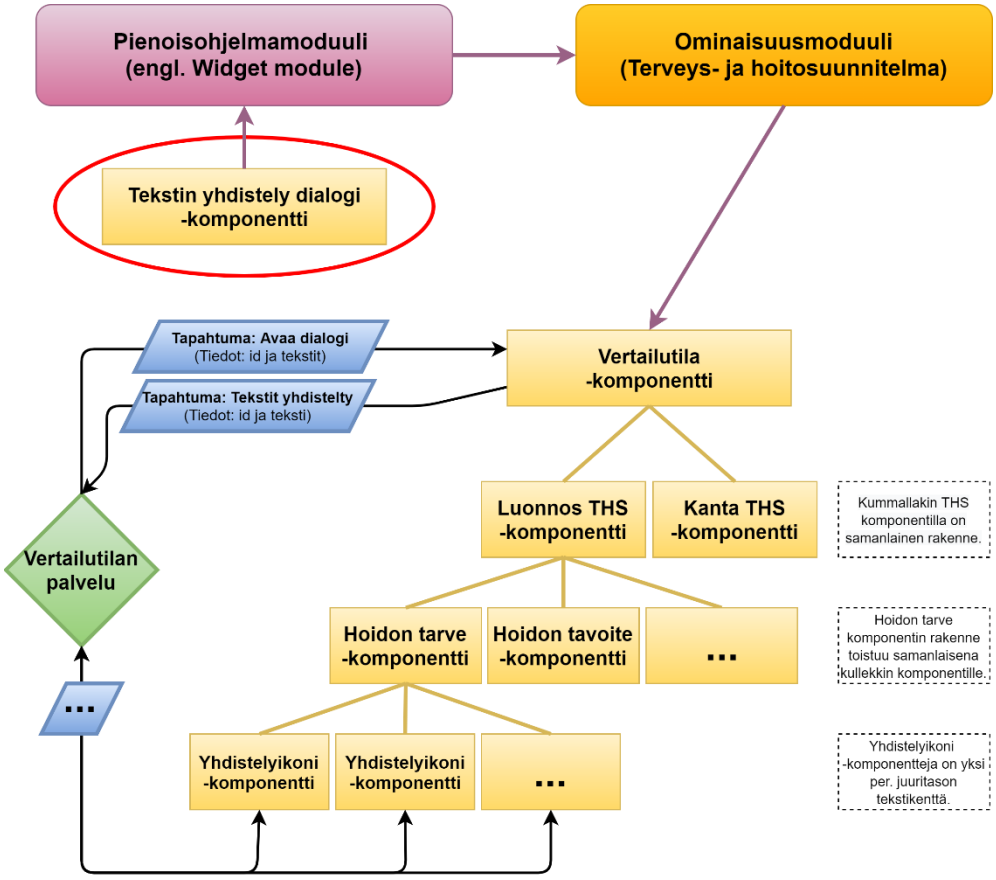
Kuva 41. Dialogi valmiina tuotantoympäristössä.



Kuva 42. Dialogi valmiina tuotantoympäristössä. Lausekomponentti editointitilassa.

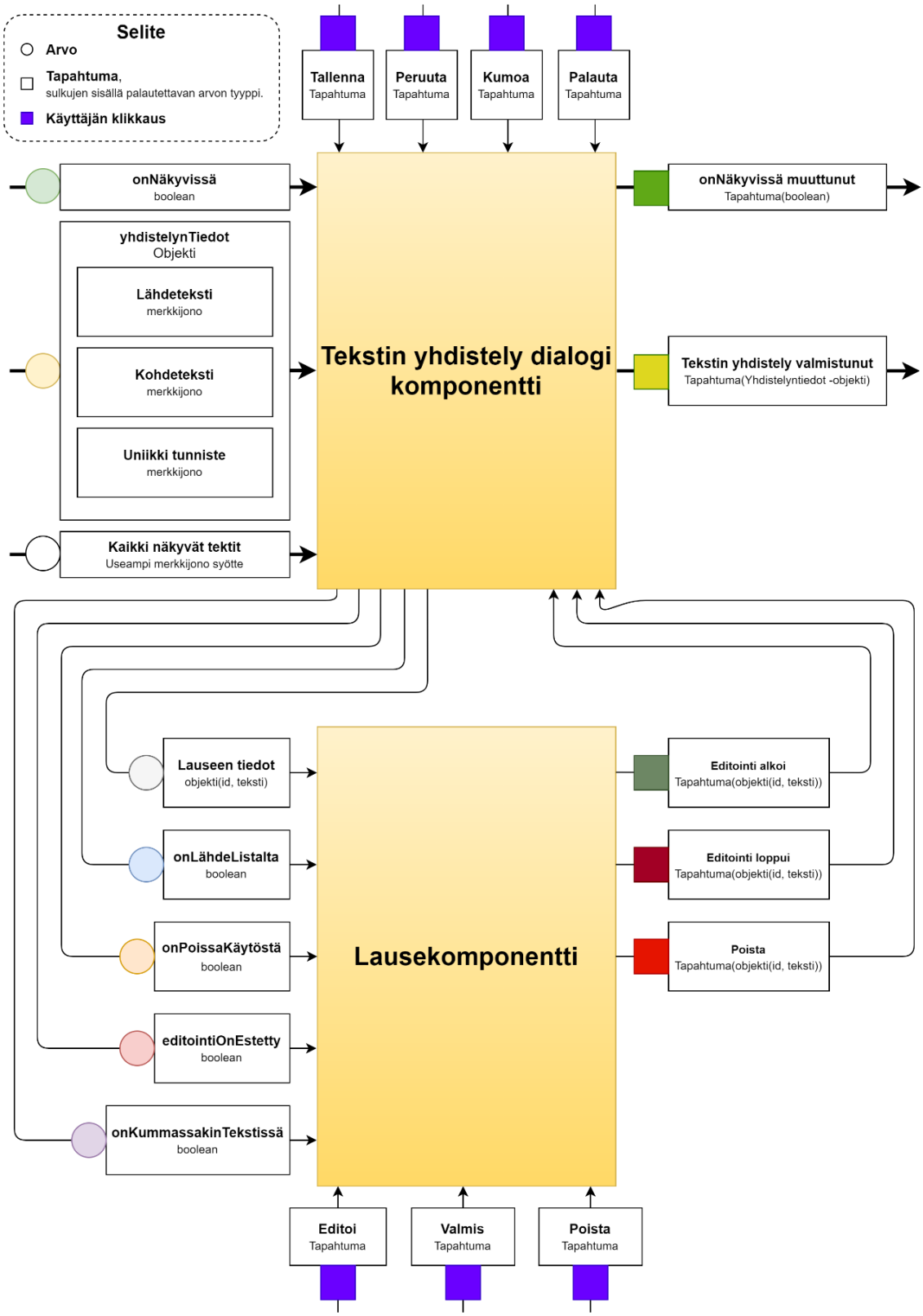
3.1.4 Tekstin yhdistely dialogi -komponentin tekninen toteutus

Projektin alussa kyseinen komponentti päätettiin toteuttaa yleiskäyttöisenä pienoisohjelmana (engl. widget). Sitä oli siis mahdollista käyttää terveys- ja hoitosuunnitelma moduulin lisäksi muissa käyttöliittymän moduuleissa. Dialogin otettiin käyttöön vertailutila -komponentissa, jossa sille annettiin tarvittavat syötteet ja tekstit sidottiin käännösjärjestelmän muuttujiin.



Kuva 43. Projektin Angular rakenne. Yhdistely dialogi korostettuna.

Dialogin ”onNäkyvissä” muuttuja on sidottu kahteen suuntaan. Isäntäkomponentin muuttujan tila päivittyy dialogille automaattisesti ja dialogin sisällä tehdyt muutokset päivitetään isäntäkomponentin muuttujaan tapahtuman kautta. Yhdistelyn tiedot dialogi saa vertailutilan palvelulta, kun käyttäjä klikkaa jotakin ikoneista. Mukana on uniikkitunniste, joka kulkee ”tekstin yhdistely valmis” -tapahtuman mukana takaisin vertailutilan palvelulle. Kaikki tekstin yhdistely ikonit kuuntelevat valmistumistapahtumia vain omalla tunnisteellaan.

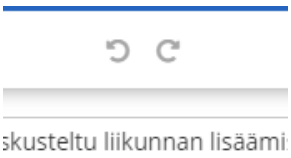


Kuva 44. Tekstin yhdistely dialogin tiedonkulun kaavio.

3.1.4.1 Kumoa ja palauta toiminto

Käyttäjällä on mahdollisuus kumota ja palauttaa tekemiään lauseiden järjestyksen tai tekstisisältöjen muutoksia. Toiminto tallentaa kopion listan tilasta aina, kun siihen tehdään muutoksia. Kopioita tallennetaan enintään kymmenen kappaletta, jonka jälkeen uuden lisäys pudottaa vanhimman pois.

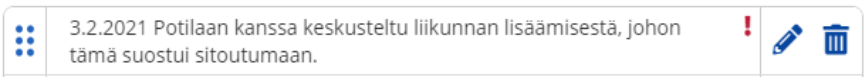
Palautustoiminnolla on myös kymmenen muutoksen verran historiaa. Se kumminkin tyhjennetään käyttäjän tehdessä uusia muutoksia.



Kuva 45. Kohdelistan kumoa ja palauta toiminto. Ikonit inaktiivisina, kun ei ole historiaa.

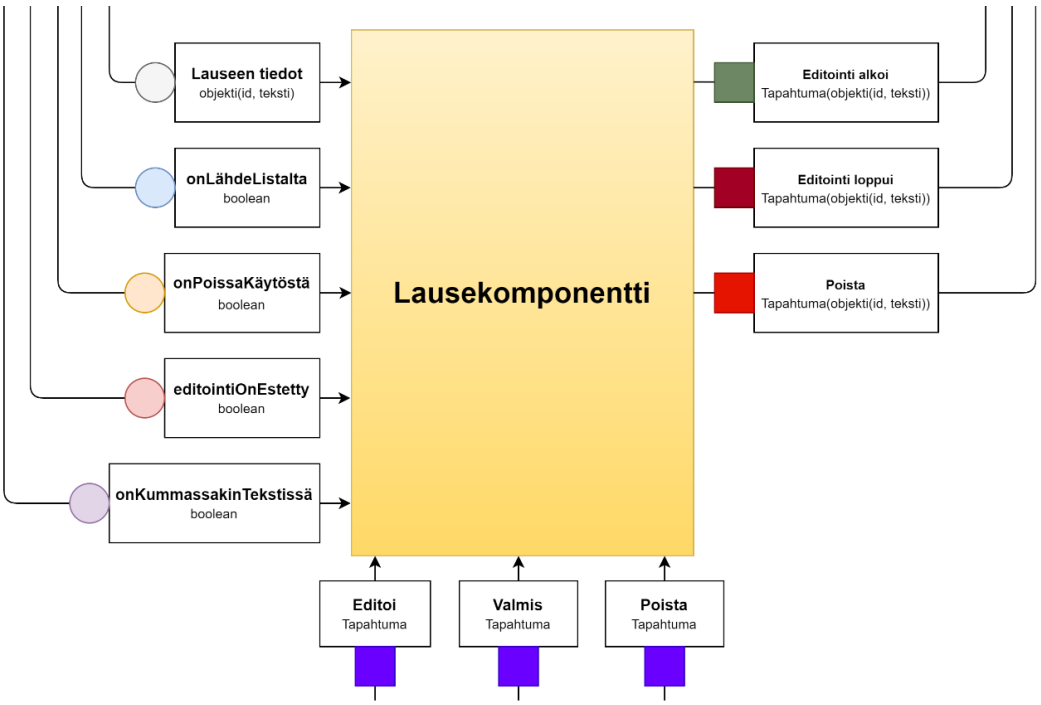
3.1.4.2 Lausekomponentti

Lausekomponentit esittävät dialogin raahattavia tekstin osia. Ne sisältävät kaikki tekstin muokkaukseen liittyvät toimintopainikkeet ja elementin tyylit.



Kuva 46. Esimerkki lausekomponentista käyttöliittymässä.

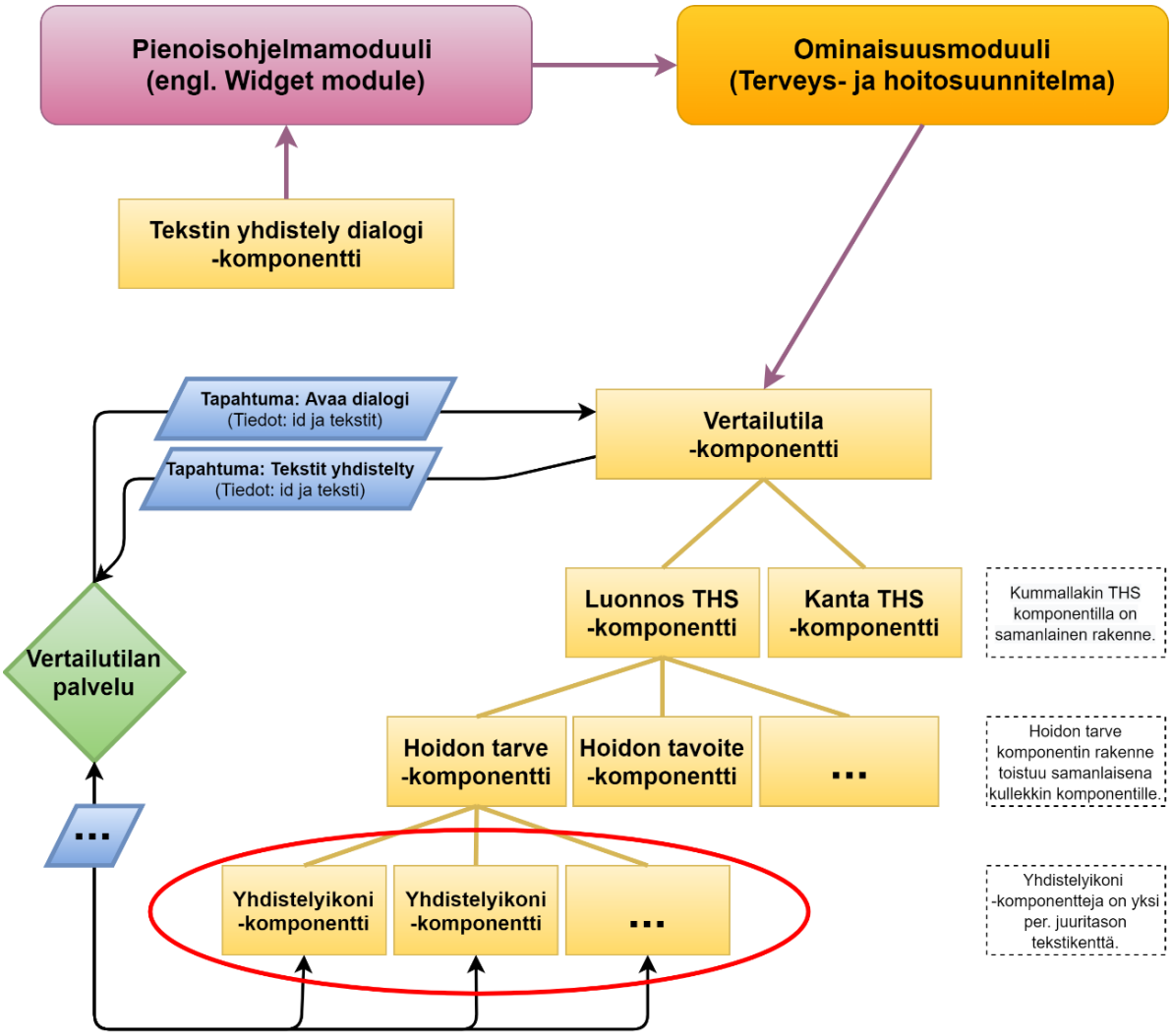
Jokainen komponentti saa tekstiarvonsa kanssa tunnisteen, jonka perusteella sen lähettämien tapahtumien toiminnot ohjataan oikeisiin paikkoihin. Editoinnin aloitus tapahtuman jälkeen dialogi muuttaa tarvittavat tyylit ja estää muiden toimintojen käytön, kunnes se saa editoinnin lopetus tapahtuman.



Kuva 47. Lausekomponentin tiedonkulun kaavio.

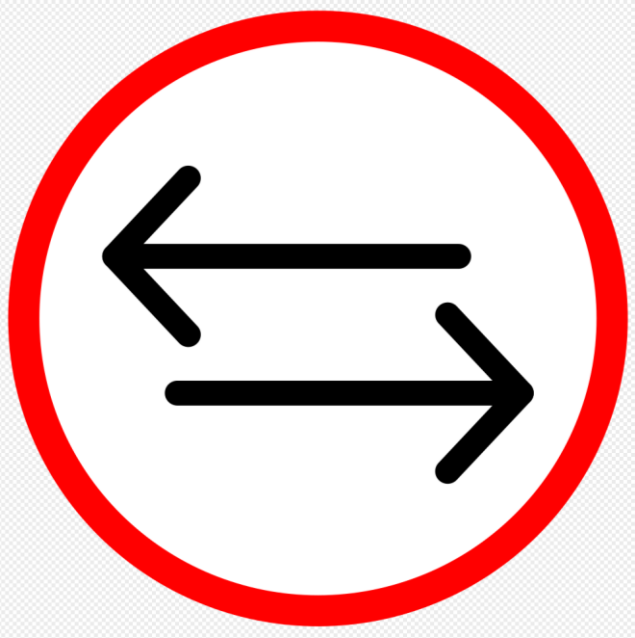
3.1.5 Yhdistelyikoni -komponentti

Terveys- ja hoitosuunnitelman vertailutila tarvitsi ikonin, josta avataan tekstin yhdistely dialogi. Ikoni sijoitetaan kahden dokumentin keskelle, niin että yhdisteltävät tekstit ovat sen kumminkin puolin. Itse ikonikomponentti sijoitetaan tekstikentän rinnalle HTML-mallissa ja jokaiselle sitä tarvitsevalle tekstikentälle tehdään oma instanssinsa.



Kuva 48. Projektin Angular rakenne, tekstin yhdistely ikoni komponentit korostettuna.

Itse ikonin toteutettiin Inkscape-ohjelmalla, jonka avulla vektori grafiikasta tallennettiin optimoitu SVG-tiedosto. Optimoinnissa grafiikan vaatimasta koodista karsittiin pois kaikki ylimääräinen. Tämä koodi vietiin suoraan Angular-komponentin HTML-mallin tiedostoon. Siellä tarvittavat parametrit sidottiin komponentin syötteenä ottamiin arvoihin, kuten koko. Ikonin lopulliset värit otettiin yrityksen väripaletin SCSS-muuttujista.

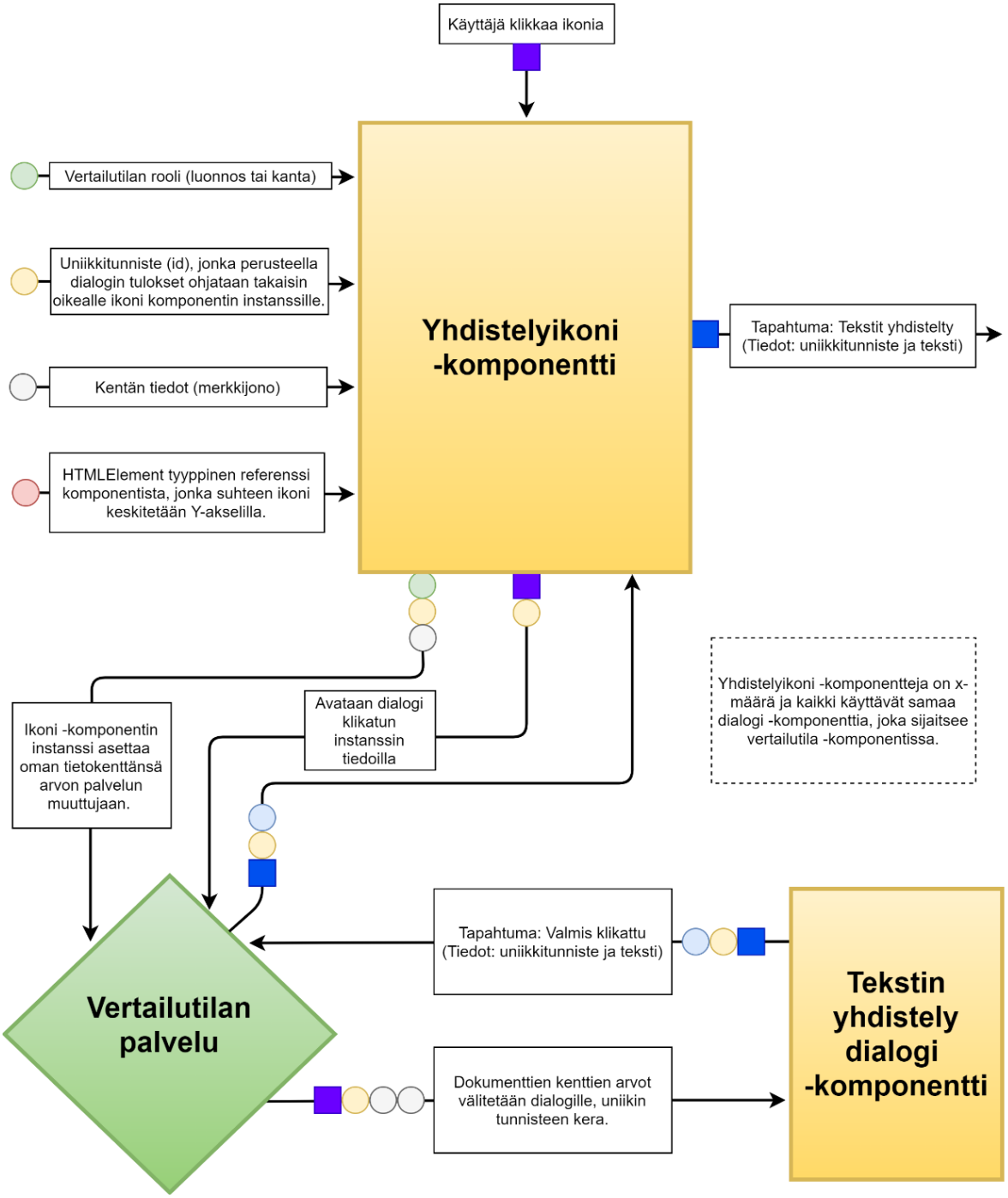


Kuva 49. Opinnäytetyöntekijän toteuttama yhdistelyikoni.

```
<svg width="100" height="100" version="1.1" viewBox="0 0 26.458 26.458" xmlns="http://www.w
3.org/2000/svg">
    <g transform="matrix(.96472 0 0 .96471 -1.945 .9723)" stroke-width="1.3713">
        <circle transform="scale(-1,1)" cx="-
15.729" cy="12.705" r="12.897" fill="#fff" stroke="#f00" stroke-linecap="round" stroke-
linejoin="round" stroke-width="1.3713"/>
    </g>
    <g fill="none" stroke="#000" stroke-linecap="round" stroke-width=".75117">
        <g transform="matrix(1.4089 0 0 1.4089 -10.903 -2.2334)">
            <path d="m12.907 13.229h10.321l-2.2065 2.3641" stroke-linejoin="round"/>
            <path d="m23.229 13.229-2.2065-2.3641"/>
        </g>
        <g transform="matrix(-1.4089 0 0 1.4089 37.362 -8.0543)">
            <path d="m12.907 13.229h10.321l-2.2065 2.3641" stroke-linejoin="round"/>
            <path d="m23.229 13.229-2.2065-2.3641"/>
        </g>
    </g>
</svg>
```

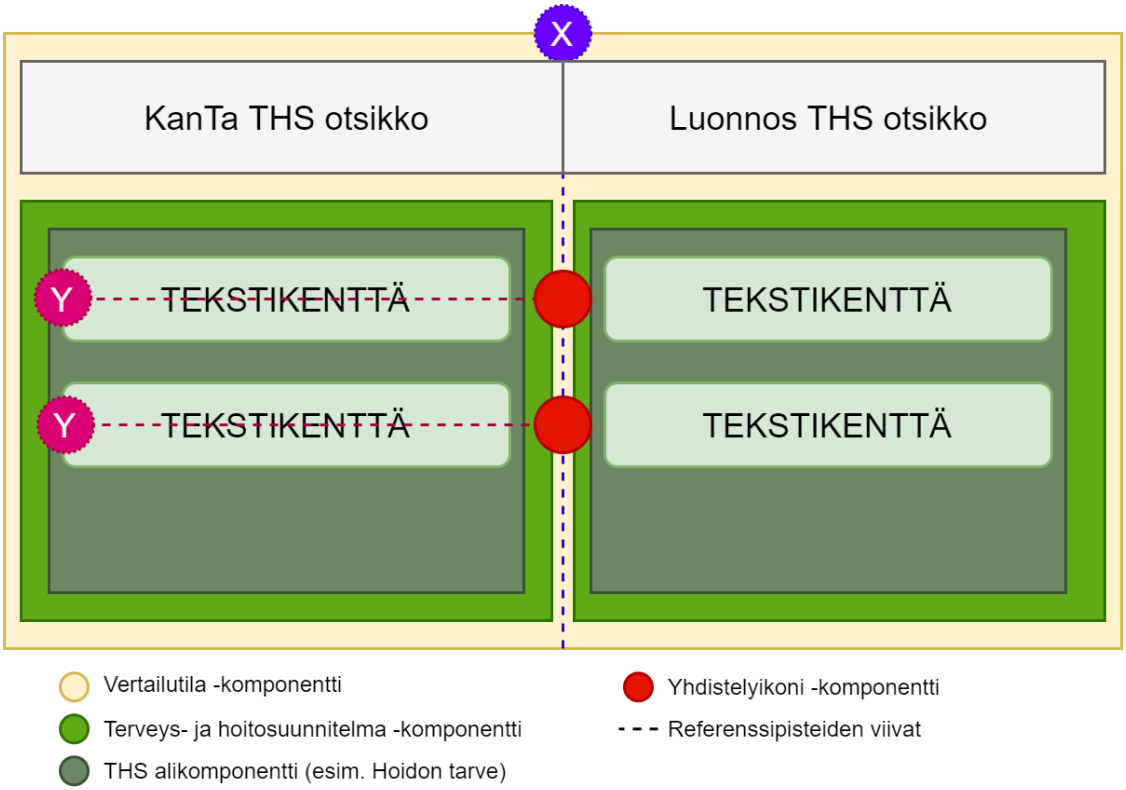
Koodi 1. Yhdistelyikonin optimoitu SVG-koodi.

Ikonikomponentin kautta välitetään kentän tiedot ja ikonin klikkaus tapahtuma vertailutilan palvelulle. Palvelua puolestaan käytetään katkaisemaan suora riippuvuus dialogista. Ikonikomponentin kautta saadaan myös tieto ja tulokset yhdistelyn valmistumisesta. Nämä tiedot käsitellään kussakin isäntäkomponentissa sen tarvitsemalla tavalla. Tämän ansiosta tekstin yhdistely toiminnallisuuden lisääminen ei muuta tiedon kulun reittiä alikomponentilta, moduulin juurikomponentille ja sieltä tietomalliin.



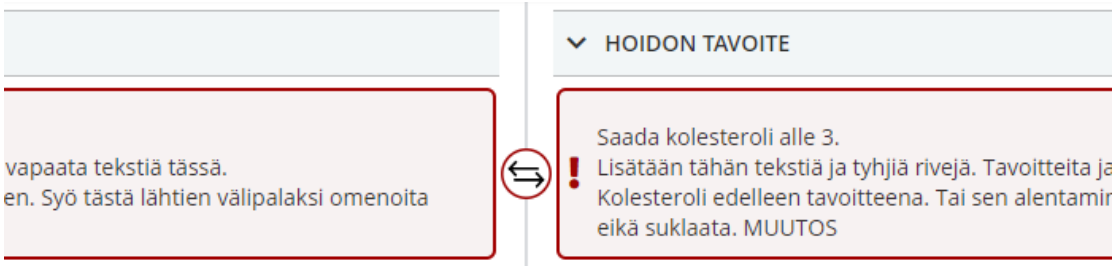
Kuva 50. Tekstin yhdistely ikoni komponentin toimintakaavio.

Vain Kanta-puolen tekstikentillä olevien yhdistelyikonien visuaalinen puoli näytetään käyttöliittymässä. Ikonin keskikohta x-akselilla otetaan vertailutila -komponentin keskikohdasta ja y-akselilla se lasketaan referenssinä saadusta tekstikentän koosta ja sijainnista.



Kuva 51. Yhdistelyikoni -komponentin referenssi pisteet.

Ikonien sijainnin päivitys on sidottu Angularin ”ngDoCheck” -elinkaarikoukkuun. Kyseinen koukku on tarkoitettu oman muutosten tunnistuslogiikan implementointiin ja sen huono puoli on se, että sitä kutsutaan todella usein (Angular, 2021). Tämän takia ikonien sijainnin uudelleen laskeminen tehdään vain silloin, kun referenssi komponentin koossa/sijainnissa tapahtuu muutoksia tai x-akselin referenssi piste siirtyy. Jokainen yhdistelyikonin instanssi pitää siis edellisen sijaintipäivityksen tiedot tallessa ja päättelee niiden avulla, tarvitseeko sijaintia laskea uudelleen.



Kuva 52. Ikoni käytössä terveys- ja hoitosuunnitelman vertailutilassa.

3.2 Kompleksisen tiedon yhdistely

Aikaa tämän opinnäytetyö aiheen käytännön toteuttamiseen oli hyvin rajattu määrä (n. 1kk). Tästä johtuen työnantajan edustajien kanssa päätettiin yhdessä tuumin jättää kompleksisen tiedon avustavat toiminnot mahdollisimman yksinkertaisiksi. Näimme riittäväksi ratkaisuksi kopiointi toiminnon, jota tarjotaan tietomallin taulukkotyyppiselle datalle mahdollisimman lähellä juuritasoa.

Suunnitellulla toiminnolla voitaisiin kopioida esimerkiksi alla olevan tietomallin serviceProviders tai services taulukon objekti dokumentista toiseen.

```
implementationOfCare: {
    textual: 'Ravitsemusterapeutin tapaaminen 2 kk välein.',
    serviceProviders: [
        // Tyhjäksi jätetty palveluntarjoajien taulukko.
    ],
    services: [
        {
          name: 'Fysioterapeuttinen kuntoutus',
          subservices: [
            {
            contents: ['Liikuntaohjelman suunnittelu', 'Liikuntaohjelman seuranta'],
              preconditions: [
                {
                  textuals: ['Pitää toteutua ennen kuin alipalvelu aloitetaan.'],
                  orCriteria: [
                      // Esimerkki sääntökokonaisuus
                    {
                      andCriteria: [
                        // Esimerkki sääntö
                        {
                            name: 'Alapaine',
                            low: {
                              value: 80,
                              unit: 'mmHg'
                            },
                            high: {
                              value: 120,
                              unit: 'mmHg'
                            }
                        }
                        // ... n-määrä JA sääntöjä.
                      ]
                    }
                    // ... n-määrä TAI sääntökokonaisuuksia.
                  ]
                }
              ]
            }
            // ... alipalveluita listattuna n-määrä.
          ]
        }
        // ... palveluita listattuna n-määrä.
    ]
  }
```

Koodi 2. Terveys- ja hoitosuunnitelman tietomallin rakenne esimerkki. Kuvattuna hoidon toteutus ja keinot.

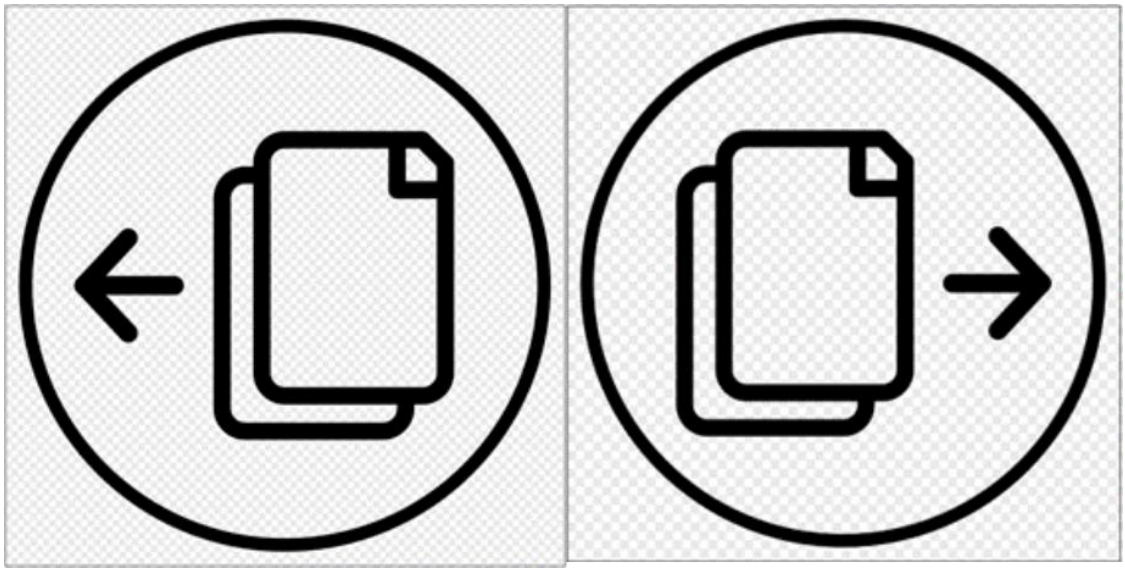
Pidemmälle jalostettujen avustavien toimintojen toteuttamista haittaa se, että jotkin tietomallin taulukot sallivat identtisten rivien olemassaolon(mm. services). Tämä tarkoittaa sitä, että sama palvelu voi esiintyä useampaan kertaan ja yhteen kuuluvien rivien luotettavaa tunnistusta ei voida tehdä. Esimerkiksi, on mahdotonta tietää mille riville kopioitava palvelun sääntö vietäisiin, kun listassa voi olla useampi samanlainen palvelu. Rivien järjestys voi myös vaihdella, niin positiota ei pystyttäisi käyttämään hyväksi tunnistuksessa.

3.2.1 Tiedon kopiointi-ikoni

Tiedon kopiointi toiminnolle tarvittiin vastaavanlainen visuaalinen toteutus, mitä tehtiin yhdistelyikonin kanssa. Sen tuli hyödyntää samaa asettelu menetelmää kuin yhdistelyikoni. Tämän toiminnon kuvaamiseen ei löytynyt valmista toteutusta Font Awesome -ikonikirjastosta, joten ikoni oli tuotettava itse.

Kopiotavalla tiedolla, oli selkeä suunta Kanta -dokumentista käyttäjän luonnokseen ja kopioinnille oli olemassa erittäin yleisesti käytetty ikoni. Tämän takia tuntui luonnolliselta yhdistää nuoli ja kopiointi-ikoni yhdeksi kokonaisuudeksi.

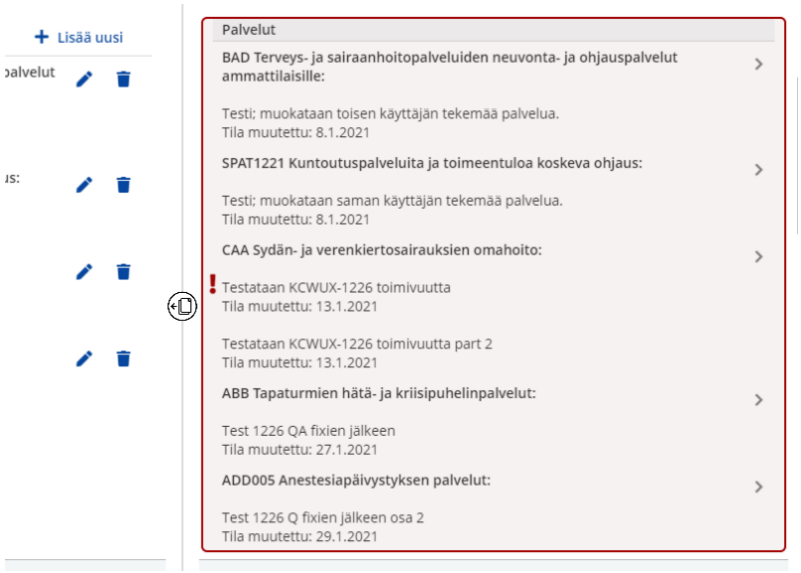
Tuotin kaksi ikonia kummassakin orientaatiossa, koska ei ollut vielä varmaan missä järjestyksessä terveys- ja hoitosuunnitelma dokumentit tullaan lopullisesti vertailutilassa esittämään. Kopiointi-ikonin takia pelkkä peilikuva ei riittänyt.



Kuva 53. Tiedon kopiointi-ikoni kummassakin orientaatiossa.

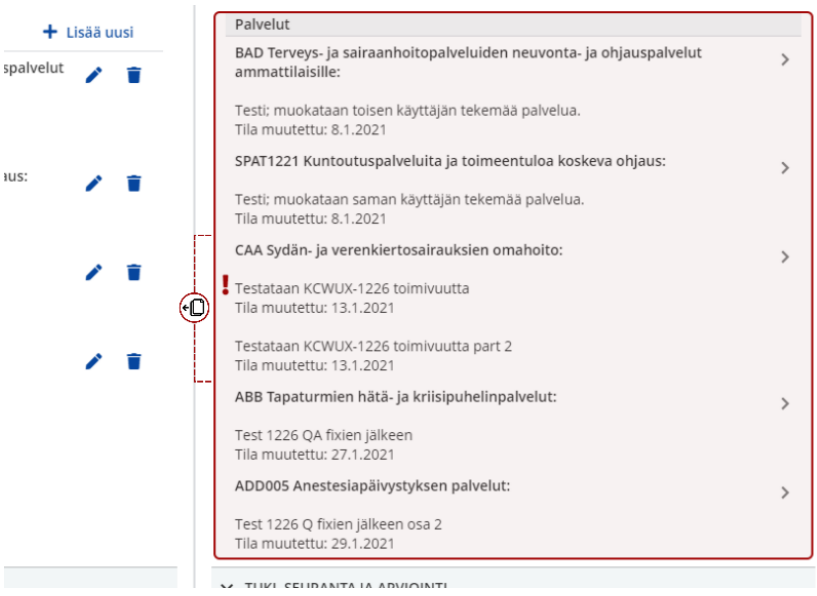
3.2.1.1 Tuotetun ikonin testaus kuviin muokkaamalla

Testaukset kohteeksi valittiin palvelut taulun, koska siinä ei ollut selkeästi rajattu missä rivin tiedot alkavat ja loppuvat. Kuten alla olevasta kuvasta näkyy, ei käyttäjälle ole kovin selkeätä mitä osaa tiedoista kyseisellä painikkeella kopioitaisiin. Eroavaisuuksien korostus suunnitellusti kehystää kokonaisuuden missä eroja on löydettävissä. Näin käyttäjä osaa kiinnittää siihen erityistä huomiota. Tämä yhdistettynä tiedon kopiointi-ikoniin saataa antaa käyttäjälle virheellisen kuvan siitä, että kyseisellä painikkeella kopiodaan kaikki tiedot dokumentilta toiselle. Oli siis tarpeen selkeyttää sitä mitä osaa kopiointi toiminnallisuus koskee.



Kuva 54. Tiedon kopiointi-ikoni muokattuna vertailutilan kuvan päälle.

Seuraavan kuvan hahmotelmassa käytettiin eroavaisuuksien korostamiseen käytettyä punaista väriä luomaan yhteys ikonin ja korostetun laatikon välille. Samanvärisellä katkoviivalla annettiin ikonille visuaalinen konteksti, jonka puitteissa se toimii. Näin käyttäjällä on mahdollisuus tunnistaa kopioitavat tiedot etukäteen.



Kuva 55. Tiedon kopiointi-ikoni ja katkoviiva muokattuna vertailutilan kuvan päälle.

3.2.2 Tiedon kopiointi komponentti

Tämän komponentin kehitys aloitettiin samalla tavalla kuin dialogikomponentin. Sille luotiin uusi Angular -projekti nopeaa kehitystyötä varten. Tästä kehitysympäristöstä ei ollut tärkeää saada tuotantoympäristön kaltaista riippuvuuksiltaan tai tyyleiltään. Tärkeämpää, oli päästä nopeasti luomaan grafiikan generoimiseen tarvittavat toiminnot.

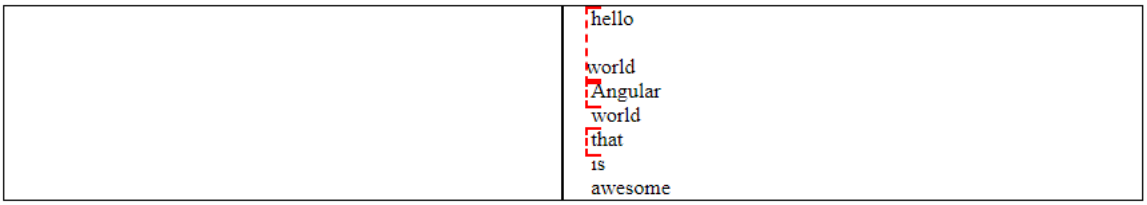
3.2.2.1 Prototyyppi

Sivun rakenteesta luotiin yksinkertaistettu versio, johon komponentti sijoitettiin. Toiselle puolelle iterointiin div -elementtejä, joihin sijoitettiin tekstiä rivinvaihtojen kanssa. Saman elementin sisään syötettiin ensimmäinen versio tiedon kopiointi-ikonista. Komponentti sai syötteenä oman rivinsä div -elementin referenssin (engl. template variable). Tämän perusteella laskettiin komponentin sijainti ja koko. Referenssin muuttuja julistetaan syntaksilla ”<risuaita><muuttujan nimi>”, esim. ”#row” ja se mihin se viittaa vaihtelee sijoitettavan elementin mukaan (Angular, 2021).

```
<div>
  <div>
    <div style="display: flex; width: 60%">
      <span style="flex: 1; border: 1px solid black"> </span>
      <span style="flex: 1; border: 1px solid black; padding-left: 1rem">
        <div
          #row
          *ngFor="let item of items; let i = index"
          style="white-space: pre-wrap"
        >
          {{ item.text }}
          <app-data-copy-icon
            *ngIf="showCopyDirective[item.index]"
            [parentRef]="row"
          ></app-data-copy-icon>
        </div>
      </span>
    </div>
  </div>
</div>
```

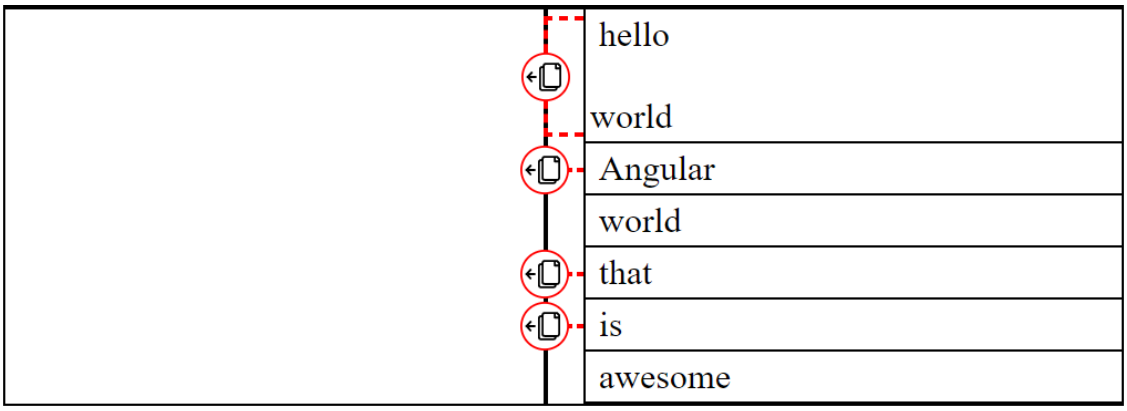
Koodi 3. Tiedon kopiointi-ikoni komponentin ensimmäinen versio koodissa.

Dynaamiset grafiikat komponentissa toteutettiin käyttäen hyväksi HTML-elementin reunan tyylejä. Reunat laitetaan näkyviin katkoviiva tyylillä, niin että yksi niistä jätetään piirtämättä. Elementin koko ja sijainti lasketaan referenssinä saadun elementin tietojen pohjalta.



Kuva 56. Tiedon kopiointi-ikoni komponentin ensimmäinen versio selaimessa.

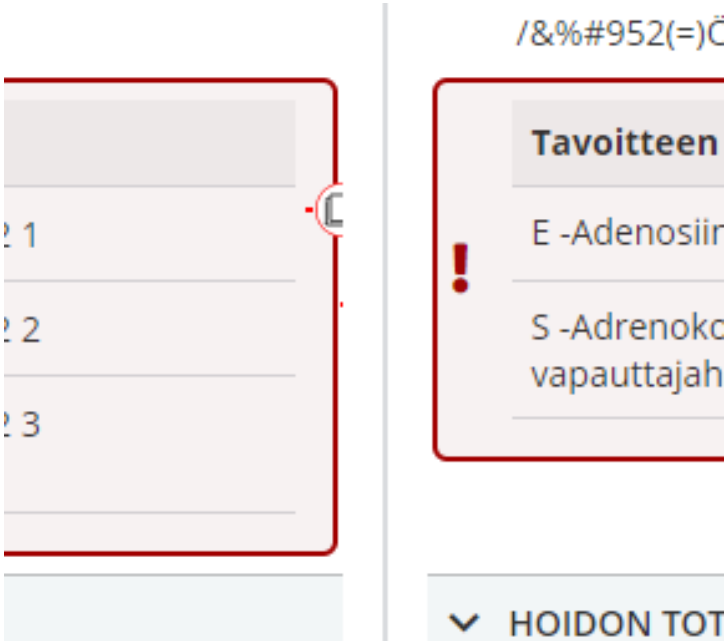
Jatkojalostuksessa todettiin tarpeelliseksi lisätä katkoviivalle vaihtoehtoinen tyyli, silloin kun rivin korkeus on lähellä ikonin kokoa. Alkuperäisessä tyylissä katkoviivan koko jätetään alle referenssi komponentin koon, että listassa vierekkäisten ikonien viivat eivät ole liian lähellä toisiaan.



Kuva 57. Tiedon kopiointi-ikonin prototyyppi jatkojalostuksen jälkeen.

3.2.2.2 Komponentin vienti varsinaiseen kehitysympäristöön

Varsinaisessa kehitysympäristössä ilmeni heti alkuun aikaa vieviä ongelmia, kun ikonit eivät asettunee dokumenttien keskelle päällimmäiseksi.



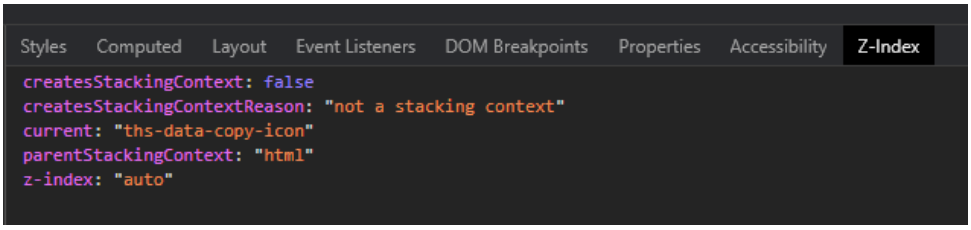
Kuva 58. Tiedon kopiointi-ikoni varsinaisessa kehitysympäristössä. Ikonia siirretty tyyleistä käsin esimerkkiä varten.

Tätä epäiltiin pitkään pinokonteksti ongelmaksi, sillä ikonikomponenteissa käytettiin relatiivista asettelua ja z-index arvojen muuttaminen ei vaikuttanut elementtien pinoutumiseen mitenkään. Relatiivisen asettelun ja z-index arvon käyttäminen luo oman pinokontekstinsa (Lombardi, 2021).

Ongelma olisi voinut johtua siitä, että ikonin isäntä konteksti on eri, kuin HTML-elementin, jonka taakse se piiloutuu. Tässä tapauksessa ongelmaa ei voisi korjata muuta kuin poistamalla kontekseja peittävältä elementiltä, niin monta että ne lopulta pohjaisivat yhteen ja samaan kontekstiin. Vasta tämän jälkeen z-index arvolla voitaisiin määritellä elementtien pinoutumisjärjestys.

Selaimeen löytyi lisäosa, jolla pystyi tarkistamaan kontekstit kullekin elementille. Se näytti, että ne pohjautuvat samaan isäntäkontekstiin, joten ongelma ei ollut pinokontekstissa.

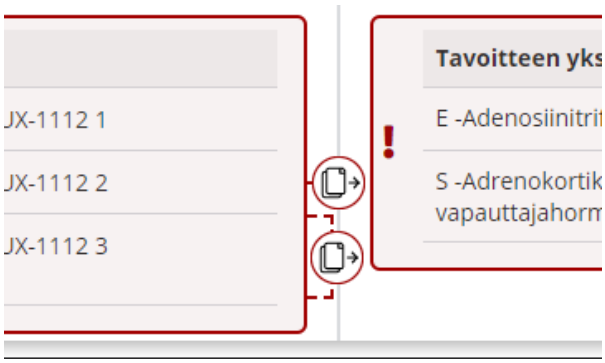
Kuva 59. Peittävän elementin kontekstitiedot.



Kuva 60. Kopiointi-ikonin kontekstitiedot sen jälkeen, kun z-index on otettu pois käytöstä.

Ongelman aiheuttajaksi paljastui THS-komponentin sisällön käärivän komponentin overflow -ominaisuus. Käärivää komponenttia oli käytetty mahdollistamaan THS-dokumentin sisällön ruudun vieritys. Se oli kuitenkin turha, koska vertailutilassa luotiin oma vieritysikkuna.

Korjauksena tyylit kumottiin vertailutilan tyylitiedostossa, käyttäen hyväksi ”::ng-deep” -valitsijaa (engl. selector), joka mahdollistaa lapsikomponenttien tyylien muutokset. Host-valitsijalla rajattiin tyylit vaikuttamaan vain vertailutila -komponentista alaspäin.



Kuva 61. Tiedon kopiointi-ikonit korjauksen jälkeen.

```
:host ::ng-deep {
  .component-content-wrapper-content {
    overflow: visible !important;
  }
  .component-content-wrapper {
    overflow-x: visible !important;
    overflow-y: visible !important;
  }
}
```

Koodi 4. Overflow ongelman korjaavat tyylit.

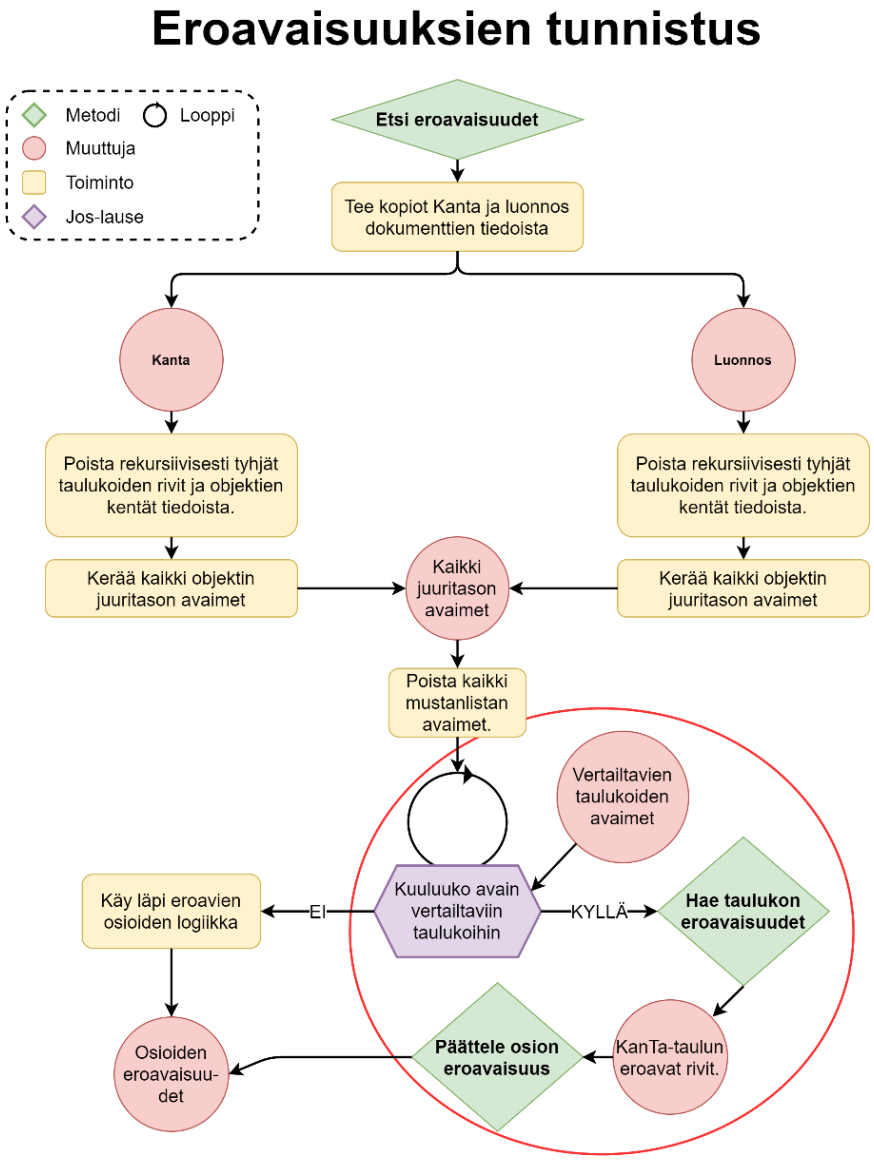
3.3 Tietojen eroavaisuuksien tunnistus

Kopiointi-ikoneja tuli näyttää vain riveille, joiden tietoja ei löydy käyttäjän terveys- ja hoitosuunnitelman luonnoksesta. Tämän takia kopiointi ominaisuutta käyttävien taulukoiden eroavaisuudet piti selvittää rivi kohtaisesti. Tekstien yhdistely dialogin ikonit voitiin näyttää vanhan tunnistuslogiikan tulosten avulla.

Vaatimukset lisättävälle tunnistukselle olivat:

1. Rivien järjestyksen muuttuminen ei saa vaikuttaa tunnistukseen.
2. Jokaiselle riville pitää olla tulos siitä, onko se eroava tiedoiltaan.
3. Mahdollisimman suorituskykyinen, käyttämättä kuitenkaan paljoa aikaa sen hiomiseen.

Aikaisemmassa toteutuksessa kumpienkin dokumenttien objektien avaimet kerättiin alkuun yhdeksi kokonaisuudeksi, jotta voitiin tunnistaa myös jommastakummasta objektista puuttuvat avaimet eroavaisuuksiksi. Tästä listasta vielä suodatettiin pois ”mustalla listalla” olevat avaimet. Näiden avainten takana on käyttöliittymän asettamaa tai ei vertailua tarvitsevaa tietoa. Vain jäljelle jäänei-den avainten takana olevia tietoja vertailtiin keskenään.

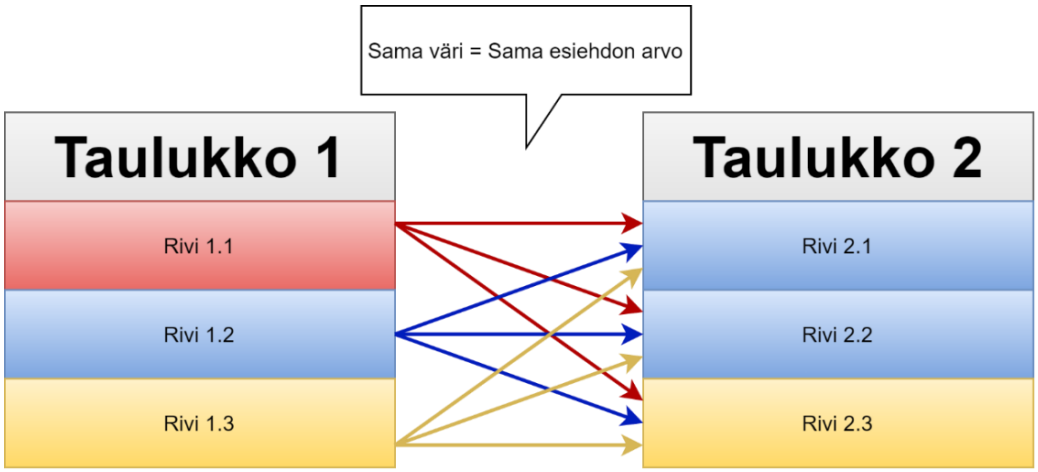


Kuva 62. Eroavaisuuksien tunnistus. Opinnäytetyöhön liittyvä osuus korostettuna.

3.3.1.1 Taulukoiden rivien eroavaisuuksien tunnistus

Tietoa kopioidaan vain Kanta-dokumentista luonnokseen päin, joten eroavaisuuksien tunnistus piti tehdä myös tästä näkökulmasta. Sen tuli löytää kaikki tiedot mitä ei ole luonnoksessa, tai tiedot, jotka eroavat luonnoksen tiedoista.

Ensimmäisenä oli ratkaistava, miten vältetään mahdolliset $n * n$, eli vaativuudeltaan $O(n^2)$ tilanteet tietojen vertailussa. Tämä oli erityisen tärkeää, koska rivien sisältämiä objekteja tullaan syvästi vertailemaan keskenään Lodash-kirjaston isEqual-metodilla.

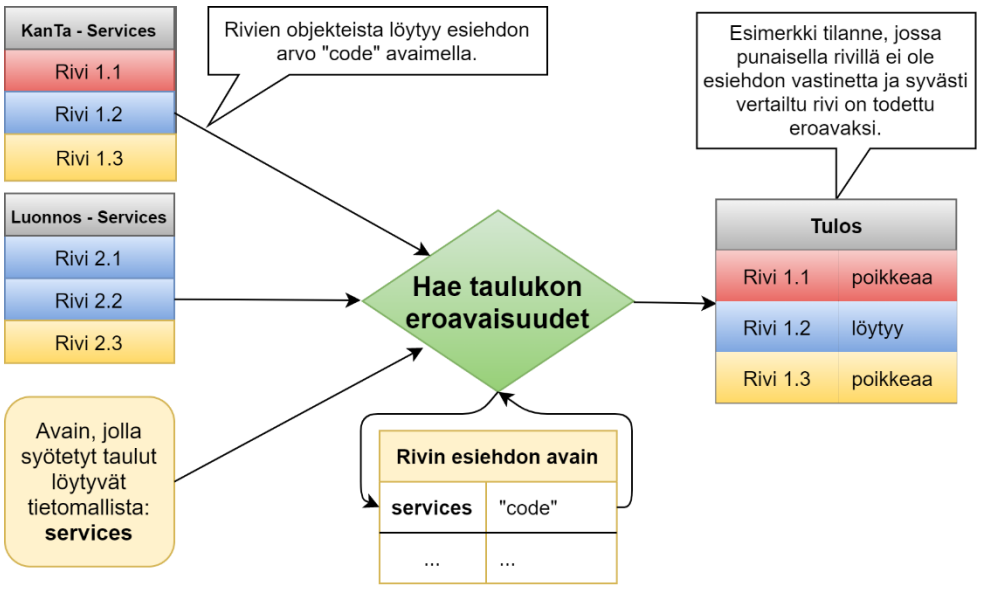


Kuva 63. Esimerkki $O(n^2)$ vaativuuden tilanteesta, kun samankokoisten taulujen rivejä pitää vertailla toisiinsa.

Ongelma ratkesi käyttämällä esiehtoa, vain sen täyttäviä rivejä vertaillaan keskenään. Esiehdoksi valitaan sellainen arvo, joka varmasti pysyy samana silloin muut tiedot voivat myös olla samoja.

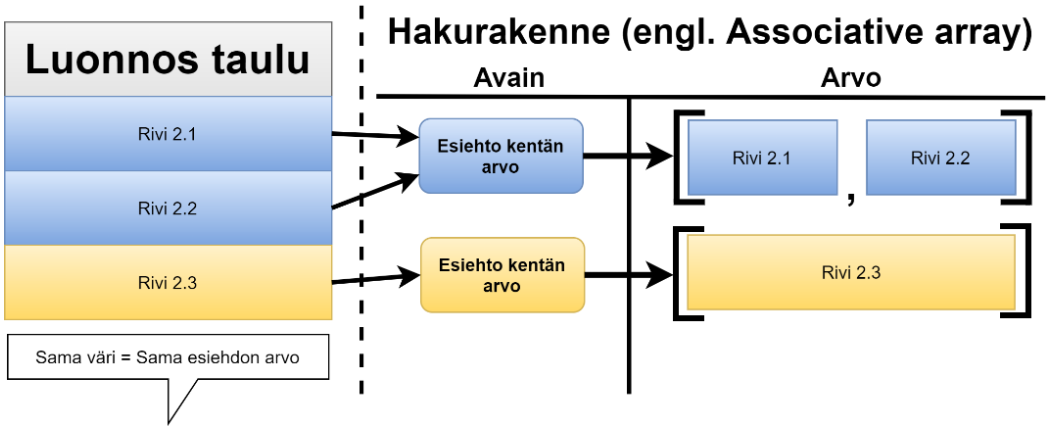
3.3.1.2 Rivien eroavaisuuksien tunnistus metodi

Metodi saa syötteekseen kummankin puolen taulukot sekä avaimen minkä perusteella haetaan esiehtokentän arvo.



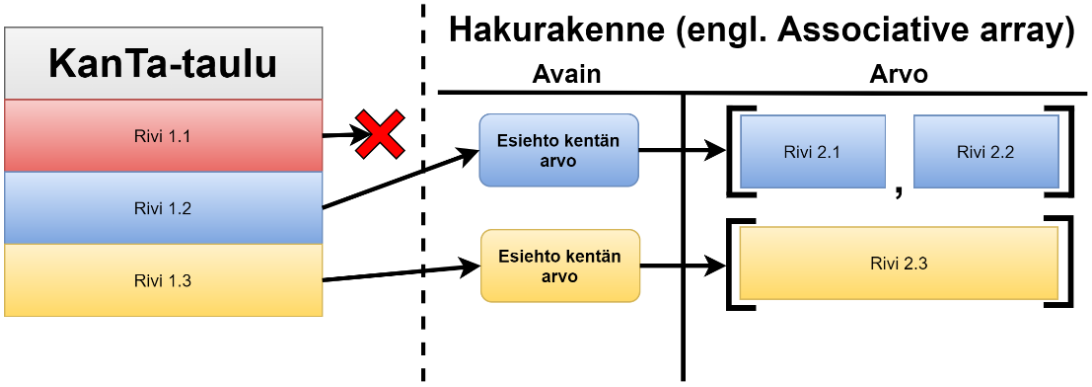
Kuva 64. Esimerkki eroavaisuuksien tunnistus metodin tuloksesta.

Aluksi luonnos taulun rivit viedään hakurakenteeseen (engl. associative array), niin että esiehtokentän arvo asetetaan avaimeksi. Avaimella löytyvä arvo on taulukko, jossa on kaikki esiehdon täyttävät rivit (objektit).



Kuva 65. Rivien eroavaisuuksien tunnistus. Tiedon lajittelu esiehdon mukaan.

Sen jälkeen käydään läpi kaikki Kanta-taulun rivit. Rivi merkataan eroavaksi, jos hakurakenteesta ei löydy esiehdon avainta tai yksikään syvästi vertailtu arvo ei ole täysin vastaava.



Kuva 66. Rivien eroavaisuuksien tunnistus. Tiedon vertaus lajiteltuihin tietoihin.

Kyseisessä esimerkissä, siis tarvitaan vain kolme syvää vertailua objektien välillä, yhdeksän sijaan (jokaista riviä olisi vertailtu toisiinsa). Näin vaativuus saadaan luokkaan $O(n)$.

3.3.1.3 Lopputulos

Metodi palauttaa Kanta-taulun kokoisen taulukon, jossa on boolean arvo jokaista riviä kohden. Arvo kertoo, löytyykö vastaava rivi jo luonnos taulusta. Tämän tiedon perusteella voidaan näyttää kopiointi-ikoni kaikille eroaville riveille.

Osion eroavaisuus päätellään rivieroavaisuuksien tuloksista. Näin säästytään yhdeltä syvältä vertailulta taulukoiden kesken. Päättelyn logiikka on seuraava. Yhdenkin eroavan rivin löytyminen tarkoittaa osioiden eroavaisuutta. Mikäli niitä ei ole, on vielä mahdollista, että luonnostaulussa on erimäärä rivejä. Tämä myös tarkoittaisi sitä, että osiot eroavat toisistaan.

4 YHTEENVETO

Projektin tarkoituksena oli suunnitella ja toteuttaa avustavia toimintoja kahden terveys- ja hoitosuunnitelma dokumentin tietosisällön yhdistelyyn. Toiminnot toteutettiin yrityksen olemassa olevaan verkkopohjaiseen käyttöliittymä toteutukseen. Aikaa tälle varattiin noin yksi kuukausi ja sen lisäksi yksi viikko yrityksen sisäisiin koodinkatselmointi yms. prosesseihin.

4.1 Pohdintaa

Suunnitteluun ja toteutukseen käytettiin lopulta viisi viikkoa ja kaksi päivää. Tämä johtui pääosin siitä, että aikataulua suunnitellessa ei osattu ottaa tarpeeksi huomioon palaverikuorman vaikutusta opinnäytetyöntekoon. Säännöllisen lausekkeen päivittäminen IE11 yhteensopivaksi vei myös kaksi päivää aikaa. Muuten projektin aikataulutus onnistui hyvin ja kullekin toiminnallisuudelle omistetut ajan jaksot auttoivat kokonaisuuden hallinnassa. Ensisijainen tavoite oli saada määritellyssä ajassa valmiiksi tekstien yhdistely dialogi ja loput toiminnallisuudet olisivat voineet jäädä myöhemmin toteutettavaksi. Kaksi ensimmäistä viikkoa panostettiin tekstin yhdistely dialogin suunnitteluun ja toteutukseen. Jälkimmäisillä viikoilla dialogiin tehtiin lähinnä käyttötestipalaverien tarvitsemia parannuksia ja loppu aika laitettiin kopiointi toiminnon suunnitteluun sekä toteutukseen. Viimeinen viikko ja loput päivät kuluivat lähinnä ulkoasun hiomisessa sekä bugien korjauksissa.

Kaiken kaikkiaan projekti oli todella monipuolinen ja sopivasti tekijänsä taitoja haastava. Ikonien luominen tuli täysin uutena asiana ja tietojen vertailun algoritmin kompleksisuuden minimointi onnistui hyvin. Säännöllisten lausekkeiden (engl. regular expression) käytöstä oli aikaisemmin vain pintapuolista tietoa ja tämän projektin aikana merkittävästi syvennettiin tekijän tietoa aiheesta.

Käyttäjälähtöisen suunnittelun toimintaperiaatteiden noudattamien merkittävästi paransi tuotetun dialogin ulkoasua ja käyttökokemusta. Vain noin kahden tunnin kokonaispanostuksella per. henkilö, saatiin aikaan merkittävää hyötyä, mikä kielii tämän metodin toimivuudesta.

4.2 Jatkokehittämismahdollisuudet

Kopiointi toiminnallisuutta olisi mahdollista viedä vielä pidemmälle, niin että syvällä tietorakenteessa olevia arvoja voisi vaivattomasti kopioida dokumentista toiseen tai sen sisällä. Tämän toteuttaisin hyödyntäen raahaa ja pudota toiminnallisuutta. Raahaamalla kopioitaviin tietoihin lisättäisiin jonkinlainen indikaattori siitä, että tämä toiminnallisuus on käytettävissä. Raahaustapahtuman alettua korostettaisiin alueet, joihin kyseisen tiedon pystyy kopiomaan.

Tekstin yhdistely dialogissa todella lähellä toisiaan olevista lauseista voi olla hankala erottaa eroavaa tekijää, esim. muuttunutta kirjainta tai ylimääräistä välilyöntiä lauseen keskellä. Lauseet kyllä indikoidaan selkeästi eroaviksi jo nyt, mutta käyttäjä voi ajautua kyseenalaistamaan dialogin toimintaa, mikäli ero ei ole helposti nähtävissä. Lauseiden toisiinsa liittyvyyden toteaminen luotettavasti, olisi luultavasti hankala toteuttaa ja jokseenkin epäluotettavaa. Tästä johtuen lisäisin yksinkertaisen ”korosta erot” toiminnallisuuden lausekomponentteihin, joka lisäisi punaisen taustavärin kaikkiin eroaviin merkkeihin toisen listan lausekomponenttien teksteissä.

LÄHDELUETTELO

Angular. (23. 4 2021). *Lifecycle-hooks*. Noudettu osoitteesta https://angular.io/guide/lifecycle-hooks

Angular. (23. 4 2021). *Template variables*. Noudettu osoitteesta https://angular.io/guide/template-reference-variables

Angular. (9. 4 2021). *What is angular?* Noudettu osoitteesta https://angular.io/guide/what-is-angular

Angular Material. (9. 4 2021). *Angular Material*. Noudettu osoitteesta www.material.angular.io/

Atlassian. (5. 4 2021). Noudettu osoitteesta www.atlassian.com/software/jira/features

Font Awesome. (16. 4 2021). *Grip vertical*. Noudettu osoitteesta https://fontawesome.com/icons/grip-vertical?style=solid

Inkscape. (29. 03 2021). *Inkscape about*. Noudettu osoitteesta https://inkscape.org/about/

Interaction Design Foundation. (21. 4 2021). *User Centered Design*. Noudettu osoitteesta https://www.interaction-design.org/literature/topics/user-centered-design

Komulainen, J.;Vuokko, R.;& Mäkelä, M. (7 2011). *Rakenteinen terveys- ja hoitosuunnitelma. Pdf-tiedosto.* Noudettu osoitteesta https://thl.fi/documents/10531/130066/Luokitukset_%20termist%C3%B6t%20ja%20tilasto-ohjeet%202011%207.pdf

Laubheimer, P. (23. 2 2020). *Drag–and–Drop: How to Design for Ease of Use*. Noudettu osoitteesta https://www.nngroup.com/articles/drag-drop/

Lombardi, P. (27. 3 2021). *The stacking context*. Noudettu osoitteesta https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context

Mediconsult Oy. (23. 3 2021). *Mediconsult kotisivut*. Noudettu osoitteesta www.mediconsult.fi

Microsoft. (23. 3 2021). *Visual Studio Code*. Noudettu osoitteesta https://code.visualstudio.com/docs

Nielsen Norman Group. (17. 1 2021). *Visual Hierarchy in UX: Definition*. Noudettu osoitteesta https://www.nngroup.com/articles/visual-hierarchy-ux-definition/

Nielsen Norman Group. (2020. 2 23). *Drag–and–Drop: How to Design for Ease of Use*. Noudettu osoitteesta https://www.nngroup.com/articles/drag-drop/

Primefaces. (9. 4 2021). *PrimeNG*. Noudettu osoitteesta https://www.primefaces.org/primeng/

Scott Chacon, J. L. (21. 3 2021). *Git*. Noudettu osoitteesta https://git-scm.com/about

Stack overflow. (2015). *Python - RegEx for splitting text into sentences (sentence-tokenizing) [duplicate]*. Noudettu osoitteesta https://stackoverflow.com/questions/25735644/python-regex-for-splitting-text-into-sentences-sentence-tokenizing

Trello. (5. 4 2021). *Trello 101*. Noudettu osoitteesta trello.com/guide/trello-101