OULUN AMMATTIKORKEAKOULU

Sami Jouppila

OHJELMISTOSUUNNITTELIJAN PÄIVÄKIRJA

OHJELMISTOSUUNNITTELIJAN PÄIVÄKIRJA

Sami Jouppila
Opinnäytetyö
Kevät 2022
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Sami Jouppila
Opinnäytetyön nimi: Ohjelmistosuunnittelijan päiväkirja
Työn ohjaaja: Lasse Haverinen
Työn valmistumislukukausi ja -vuosi: Kevät 2022 Sivumäärä: 36

Opinnäytetyössä kuvattiin päiväkirjamuotoisesti ohjelmistosuunnittelijan arkea potilastietojärjestelmän kehitystyössä. Työn toimeksiantaja oli Esko Systems Oy. Opinnäytetyöhön sisältyi kuvaus lähtötilanteesta, päiväkirjamuotoinen raportointi työtehtävistä ja pohdintaosuus ilmi tulleista asioista.

Lähtötilanne-luvussa kerrottiin ensin yrityksestä. Yrityksestä kerrottiin perustiedot, organisaation tiedot sekä tietoa yrityksen tuottamasta Esko-potilastietojärjestelmästä. Luvussa kerrottiin myös tietoa kirjoittajan osaamisesta, työtehtävästä ja keskeisestä ominaisuudesta rakenteisesta annostuksesta, mihin opinnäytetyön tehtävät keskittyvät.

Raportointiosuudessa on päiväkirjamerkinnät. Niissä on kuvattu viikkotasolla suunnittelua, työtehtävien toteuttamisen kuvausta ja pohdintaa. Työtehtävät-osioissa on kuvattu tehtävien kohdalla niihin liittyvää teknistä toteutusta ja sitä prosessia, jolla tehtävät saatiin toteutettua. Pohdinta-osioihin on mietitty kyseisenä viikkona opittuja asioita ja ongelmakohtia, joita kyseisellä viikolla havaittiin.

Pohdintaosuudessa mietitään opinnäytetyön sujuvuutta ja asioita, mitä työn aikana tuli ilmi. Pohdinnassa huomioidaan, että työn suunnittelu on ollut ongelmallista.

Asiasanat: ohjelmistosuunnittelu, potilastietojärjestelmä, päiväkirjamuotoinen opinnäytetyö, eResepti

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

Author: Sami Jouppila
Title of thesis: Diary of a software designer
Supervisor: Lasse Haverinen
Term and year when the thesis was submitted: Spring 2022
Number of pages: 36

The thesis is a diary format description of ordinary software design work for a patient information system. Thesis includes a description of the initial situation, reportage of diary entries and conclusion.

Initial situation chapter describes the target company, its organization and its product. Chapter also includes basic information about the author's work tasks and the key feature being developed during the thesis.

Reportage chapter includes diary entries on work tasks' planning, implementation and analysis on a weekly basis.

Conclusion chapter includes personal thoughts and reflection on the thesis. It is noted in the conclusion that effective planning has been problematic.

Keywords: software design, patient information system, thesis in diary format, electronic prescription

SISÄLLYS

1 JOHDANTO

Tämä päiväkirjamuotoisen opinnäytetyön raportti kertoo työstäni ohjelmistosuunnittelijana kahdeksan viikon ajalta. Opinnäytetyössä kerron työssä vastaantulleista ongelmista ja pohdin niiden ratkaisuja viikoittain. Työn seuranta sijoittuu ajalle 27.1.2022 – 23.3.2022.

Opinnäytetyön toimeksiantaja on Esko Systems Oy, jonka toiminta keskittyy Esko-potilastietojärjestelmän kehittämiseen ja ylläpitoon. Potilasjärjestelmää toimitetaan useammalle sairaanhoitopiirille, mutta kehitystä tehdään pääosin Oulussa yrityksen työtiloissa Kastellin tutkimuskeskuksessa. Työtä suoritetaan osana Eskon Lääkehoidon kehitystiimiä ja työ keskittyy Lääkehoito-osion kehittämiseen. Koronapandemiasta johtuen opinnäytetyötä tehdään pääosin etätyönä.

Opinnäytetyön tarkoitus on kuvata työtehtäviä seuranta-ajalta ja antaa lukijalle mielikuva siitä, millaista on suunnittelutyö potilastietojärjestelmän parissa. Henkilökohtaisena tavoitteena työssä on myös onnistua suunnittelupalavereissa sovituista työtehtävistä pääosin itsenäisesti ja tuoda omaa opinnoissa saatua osaamista esille työtehtäviä tehtäessä ja suunniteltaessa.

Opinnäytetyön raportissa kuvataan luvussa 2 työtehtäviin liittyviä oleellisia teknisiä ja lääketieteellisiä käsitteitä. Luvussa 3 kerrotaan perustietoa toimeksiantajayrityksestä, yrityksen tuotteesta ja omista työtehtävistä. Luku 4 sisältää päiväkirjamuotoisen raportoinnin sekä viikkotasoista pohdintaa. Luvussa 5 on pohdintaa koko seuranta-ajalta ilmenneistä asioista.

2 KÄSITTEISTÖ

NET	Avoimen lähdekoodin ohjelmistokehys, jolla tehdään tyypillisesti C#-kielellä ohjelmistoja.
Asynkroninen	Asynkronisessa suorituksessa koodin suoritus ei välttämättä etene rivi riviltä, vaan koodissa voi olla rivejä, joiden käsittely kestää kauemman aikaa ja niiden loppuunsaattamista ei odotella ennen kuin koodissa suoritetaan seuraavaa komentoa.
Axios	Node-kirjasto, joka helpottaa HTTP-pyyntöjen käsittelyä
Back end	Verkkosivujen kehityksessä käytetty termi, jolla viitataan palvelinpuoliseen kehitykseen.
Backlog	Scrum-kehityksessä käytettävä lista tulevista ominaisuuksista, joita sovellukselle halutaan kehittää.
Bootstrap	Suosittu sovelluskehys, jolla voidaan luoda responsiivisia www-sivuja. Toiminta perustuu HTML-luokkien käyttöön, joille on CSS-määrityksiä.
C#	Microsoftin .NET-alustalle kehitetty ohjelmistokieli.
CSS	Cascading Style Sheet. Verkkosivuille kehitetty tyylisivu, jolla määritellään, miten verkkosivulla olevat elementit näytetään sivulla.
DOM	Ohjelmointirajapinta HTML-dokumenteille, jonka avulla voidaan käsitellä dokumentin elementtejä
eResepti	Sähköinen lääkemääräys, jonka lääkäri laatii ja allekirjoittaa sähköisesti. Tallennetaan Kelan ylläpitämään Reseptikeskukseen. Lääkäri voi kirjata eReseptin käyttäen potilastietojärjestelmää.
Front end	Verkkosivujen kehityksessä käytetty termi, jolla viitataan selainpuoliseen kehitykseen.
Full stack	Full stack -osaamisella web-kehityksessä tarkoitetaan, että kehittäjällä on laaja-alainen osaaminen asiakas- ja palvelinympäristön sovelluskehityksestä sekä tietokantaratkaisuista.

HTML	HyperText Markup Language. Merkintäkieli, jolla internetsivut on rakennettu.
JavaScript	Usein web-ohjelmoinnissa käytettävä ohjelmointikieli, jota voidaan suorittaa myös selaimessa.
JSON	Tekstimuotoinen tietotyyppi, johon voidaan säilyttää JavaScriptin olio-syntaksin muotoista tietoa.
jQuery	Laajasti käytetty ja nopea JavaScript-kirjasto, joka mahdollistaa monipuolisen DOM-käsittelyn
Kontrolleri	Controller on MVC-mallissa palvelimessa vastaanottava luokka, johon voi tehdä rajapintakyselyitä.
Material UI	Reactille suunniteltu laajasti käytetty UI-komponenttikirjasto
Lääkerekisteri	Kelan ylläpitämä ammattilaisille tarkoitettu palvelu, joka sisältää yhtenäiset ja ajantasaiset tiedot lääkkeistä ja lääkeaineista lääkkeen määräämistä ja toimittamista varten.
Pyrähdys	Scrum-projektinhallinnassa määritelty ajanjakso, jolle tehtäviä suunnitellaan etukäteen. Useammin käytetään termiä sprintti, mutta Eskossa pyritään käyttämään suomenkielisiä termejä.
React	Facebookin kehittämä JavaScript-kehys, jolla voidaan rakentaa verkkosivuja.
Redux	Tilanhallintajärjestelmä JavaScriptiin pohjautuville sovelluksille, jonka tarkoitus on helpottaa laajojen ohjelmistojen tilanhallintaa.
REST	Arkkitehtuurityyli ohjelmointirajapintojen toteuttamiseen. Käytetään usein HTTP-protokollaa käytettävissä rajapinnoissa.
Scrum	Ketterässä ohjelmointikehityksessä käytettävä projektinhallinnan viitekehys, jossa ohjelmistokehitystä suunnitellaan ja toteutetaan jonkin ajanjakson pituisina pyrähdyksinä.
SQL	Structured Query Language. Standardisoitu kyselykieli, jolla voi tehdä relaatiotietokantaan hakuja, muutoksia ja lisäyksiä.
TFS	Team Foundation Server. Microsoftin tuottama alusta, jota voidaan käyttää versionhallintaan, projektinhallintaan sekä testiympäristön ylläpitämiseen.

3 LÄHTÖTILANNE

Tässä luvussa kuvataan työn lähtötilannetta. Aluksi kuvataan toimeksiantajan Esko Systems Oy:n perustietoja ja yrityksen organisaatiota sekä lääkehoito-tiimiä omana yksikkönään. Luvussa annetaan myös lyhyt kuvaus Esko-potilastietojärjestelmästä sekä alustava selvennys rakenteisen annostuksen uudistuksesta, sillä tämä on laaja uudistus joka liittyy pitkälti tarkastusjaksolla tehtäviin työtehtäviin. Luvussa kuvataan myös tekijän omia työtehtäviä. Kuvaus koskee osittain sekä aiemmin tehtyjä työtehtäviä että tarkastelujaksolla tehtäviä työtehtäviä, sillä tarkastelujakson tehtävät pohjautuvat aiempiin tehtäviin.

3.1 Yrityksen perustiedot

Esko Systems on asiakkaidensa omistama, voittoa tavoittelematon vuonna 2019 perustettu inhouse-yhtiö. Yhtiön tuotetta, Esko-potilastietojärjestelmää, on käytetty erikoissairaanhoidossa vuodesta 1996 alkaen. Jatkossa järjestelmä laajenee kattamaan perusterveydenhuollon ja potilashallinnon toiminnallisuudet. Yritys kehittää Esko-potilastietojärjestelmää asiakasomistajien käyttöä varten ja hyödyntäen heiltä saamaansa palautetta. Yrityksessä on töissä monialaisia osaajia mm. terveydenhoidon ja tietotekniikan aloilta. (1.)

3.2 Yrityksen organisaatio

Yrityksen organisaatiokaaviossa työntekijät jaetaan työrooleittain karkeasti kolmeen kategoriaan. Palvelupuolen tuoteasiantuntijat ovat lääketieteellisen alan asiantuntijoita, joilla on yleisesti ottaen käytännön kokemusta Esko-potilasjärjestelmän käytöstä lääkärin tai sairaanhoitajan roolissa. Tuotekehityksen suunnittelijat ovat tietotekniikan asiantuntijoita, joilla on lähtökohtaisesti Esko-potilasjärjestelmän kehitykseen soveltuvaa full stack -osaamista. Uudessa kolmannessa teknologiakategoriassa on myös tietotekniikan osaajia. Erona tuotekehityksen ja teknologian työssä on se, että tuotekehityksen suunnittelijat ovat osana jotakin tiimiä, jossa kehitetään tiettyä Eskon osasovellusta. Teknologia keskittyy taas laaja-alaisempiin uudistuksiin, jotka saattavat koskea useampaa osasovellusta tai koko Esko-potilastietojärjestelmää. (2.)

Tuotekehitystiimijaossa työntekijät jaetaan eri tiimeihin, joilla on omat kehitysalueet ja vastuut (3). Lähtökohtaisesti kaikissa tiimeissä on palvelupuolen tuoteasiantuntijoita ja tuotekehityksen suunnittelijoita. Tiimeissä sovelletaan Scrum-mallia, jossa sekä tuoteasiantuntijat sekä suunnittelijat osallistuvat työtehtävien määrittelyihin ja suunnitteluihin. Suunnittelijoiden vastuulla on tietoteknisten tehtävien toteutus ja tuoteasiantuntijat toimivat tiimissä ominaisuuksien määrittelijöinä sekä testaajina.

3.3 Esko-potilastietojärjestelmä

Esko-potilastietojärjestelmä on web-selaimella tai mobiililaitteella käytettävä sovellus. Esko on modulaarinen potilastietojärjestelmä, joka muodostuu yli 60:stä toisiinsa integroidusta tietojärjestelmästä ja tuotteesta. Kehityksen keskiössä ovat potilasturvallisuudesta, helppokäyttöisyydestä sekä toimintavarmuudesta huolehtiminen. (4.)

3.4 Lääkehoito

Lääkehoito on yksi Esko-potilastietojärjestelmän osasovelluksista. Lääkehoidossa voidaan mm. tarkastella potilaan lääkityslistaa, tehdä lääkemääräyksiä ja kirjoittaa sähköisiä reseptejä.

Lääkehoitoa ylläpitää ja kehittää lääkehoito-tiimi. Tiimi on tuotekehitystiimi, jossa on lääketieteellisen taustan omaavia tuoteasiantuntijoita ja tietoteknisen taustan omaavia suunnittelijoita. Tiimissä sovelletaan Scrum-mallia kahden viikon pituisilla pyrähdyksillä. Lääkehoidon kehitystä suunniteltaessa tulee ottaa huomioon myös nykyisten asiakkaiden tarpeet, sillä sovellus on aktiivisessa käytössä useissa eri sairaanhoitopiireissä.

Tämän opinnäytetyön aikaiset työt keskittyvät lääkehoidon osasovelluksen kehittämiseen. Mahdollisuuksien mukaan työt keskittyvät rakenteisen annostuksen kehittämiseen, sillä oma aiempi työ muutaman kuukauden takaa on keskittynyt tähän uudistukseen. Työtehtäviä voi tulla kuitenkin tarpeen mukaan muihin lääkehoidon kehitysprojekteihin seurantajakson aikana.

3.5 Rakenteinen annostus

Rakenteisen annostuksen uudistus on osa kansallisen lääkityslistan ensimmäisen vaiheen käyttöönottoa (5). Rakenteisessa annostuksessa reseptejä voidaan kirjata rakenteisessa muodossa, mikä tarkoittaa, että reseptin annostus voidaan kirjata vapaamuotoisen tekstin sijaan rakenteisena käyttäen määriteltyjä tietosisältöjä (6). Aikataulun mukaan uusi HL7-määrittelypaketti otetaan käyttöön 1.5.2022 mennessä. Tämä määrittelypaketti voidaan kuitenkin ottaa käyttöön kokonaan rakenteisen annostuksen kanssa tai vain uusilla tietosisällöillä. Eskoon ei ole ehditty tehdä rakenteisen annostuksen uudistusta tuohon mennessä, joten toistaiseksi käyttöön otetaan vain vaihtoehtoinen eteenpäin yhteensopiva toteutus. Tällöin rakenteisen annostuksen käyttöönotolle on aikaa 31.12.2024 asti. (7.)

Esko-potilastietojärjestelmässä rakenteisen annostuksen käyttöönoton yhteydessä toteutetaan myös laajempi teknologinen uudistus, jossa koko reseptilomakkeen front end uudistetaan React-pohjaiseksi.

3.6 Omat työtehtävät

Olen toiminut 5.4.2021 alkaen suunnittelijana tiimissä, jonka vastuulla on Eskon lääkehoito-osio. Tiimissä sovelletaan Scrum-mallia kahden viikon pyrähdyksillä ja käytetään TFS:ää projektinhallintatyökaluna. Aiemmissa työtehtävissä Eskon parissa on tullut käytettyä mm. seuraavia teknologioita: C# .Net-ympäristössä, JavaScript, HTML/CSS ja SQL. Eskoa kehitetään Windows-kehitysympäristössä, joten myös siihen on tullut osaamista.

Vuoden 2021 loppupuolella aloitin rakenteisen annostuksen projektin parissa. Rakenteisen annostuksen yhteydessä lääkehoitoon tulee myös Reactin käyttöönotto front endin puolella pelkän HTML-pohjaisen JavaScript/jQuery-toteutuksen sijaan. React otetaan käyttöön vaiheittain muiden uudistuksien yhteydessä, joten ensimmäisenä uudistetaan vain reseptilomake.

Toistaiseksi rakenteisen annostuksen parissa on ollut kaksi suunnittelijaa Eskon lääkehoidosta. Minun työtehtäviini on kuulunut reseptilomakkeen uudistaminen React-pohjaiseksi ja toinen kehittäjä on ollut uudistamassa reseptilomaketta eteenpäin yhteensopivaksi tulevien HL7-määrittelyjen kanssa. Tähän mennessä React-reseptilomakkeessa on päästy alkuun siten, että

uudistetulla lomakkeella näkyy osa tarvittavista tietokentistä. Lomake ei ole vielä toiminnallinen eikä sillä voi luoda eReseptejä.

4 RAPORTOINTI

Tässä luvussa kuvataan seurantajakson tärkeimpiä työtehtäviä viikoittain. Joka toisen viikon alussa kuvaan kahden viikon pyrähdyksen suunnittelupalaverissa sovitut työtehtävät. Työtehtävien kuvauksien jälkeen kuvataan lyhyesti omaa pohdintaa viikon työtehtävistä.

4.1 Viikko 1

4.1.1 Suunnittelu

Viikko aloitettiin kahden viikon pyrähdyksen suunnittelupalaverilla. Minulle asetetut tämän pyrähdyksen työtehtävät kohdistuivat reseptilomakkeen alaikäisen puolesta asioinnin osuudelle. Yksi tehtävistä oli melko suoraviivainen UI-parannus, jolla yhtenäistettiin reseptilomakkeen näkymää muiden potilastietojärjestelmien osioiden kanssa uudistamalla ilmoituksia ja korjattiin tekstivalintaa, jotta lomakkeen käyttö olisi käyttäjälle selkeämpää. Kaksi muuta tehtävää liittyivät reseptilomakkeen hoitotapahtuman valintaan. Suunnittelupalaverissa huomattiin, että hoitotapahtumiin liittyviä tehtäviä piti määritellä tarkemmin, ja siihen varattiin aikaa seuraavalle päivälle.

Ensimmäinen hoitotapahtumiin liittyvä tehtävä oli käyttäjän valitseman hoitotapahtuman säilyminen lomakkeella, kun kirjoitettu resepti avataan uudelleen ennen lähettämistä. Aiemmassa toteutuksessa reseptilomakkeen hoitotapahtuman oletusvalinta on hoitosuhteen varmistuksessa valittu hoitotapahtuma. Kun resepti tallennetaan, tämä tieto tallennetaan näennäisesti oikein reseptiin, mutta resepti on mahdollista myös avata ennen reseptin lähettämistä Reseptikeskukseen, jolloin aiemmin valittu valinta ei säily uudelleen avatulla lomakkeella.

Toinen hoitotapahtumiin liittyvä tehtävä on reseptilomakkeelle tehtävä muutos, jolla sallitaan jatkossa reseptin korjauksessa hoitotapahtuman muutos. Aiempi hoitotapahtumiin liittyvä tehtävä kytkeytyy myös tähän siten, että jos reseptiä korjattaessa reseptiin oli liittynyt aiemmin olemassa oleva hoitotapahtuma, tämän tulee olla reseptilomakkeella hoitotapahtuman oletusvalinta. Toteutuksessa tulee ottaa huomioon, että myös ulkopuolisten organisaatioiden reseptejä on mahdollista korjata. Lomakkeen toimintaa ulkopuolisten organisaatioiden reseptien korjauksessa

ei määritelty vielä tarkasti, sillä tätä täytyi testata tarkemmin ja pyytää Kelalta ulkopuolisen organisaation tekemä resepti testaustarkoitukseen.

4.1.2 Työtehtävät

Otin UI-parannukset työn alle ensimmäisinä ja sain ne toteutettua melko nopeasti. Historiallisista syistä reseptilomakkeeseen oli mahdollista valita valinta "Ei palvelutapahtumaa", joka tuli vain nimetä selvennyksen vuoksi "Ei hoitotapahtumaa" -valinnaksi, sillä valikossa valitaan hoitotapahtumaa eikä palvelutapahtumaa. Lomakkeelle oli myös lisättävä huomautus silloin, kun valitaan ”Ei hoitotapahtumaa”, sillä hoitotapahtuma tulisi yleensä valita, mutta hoitotapahtuman valitsematta jättäminen voidaan sallia poikkeustilanteissa.

Lomakkeella näkyviä huomautuksia tuli myös yhtenäistää tässä yhteydessä. Uusi huomautus tehtiin käyttäen Bootstrapin alert-luokkaa (8), jota on myös käytetty joissakin muissa lääkehoidon osioissa. Samalla päivitettiin edellisessä pyrähdyksessä toteutettua palvelutapahtumaan liittyvää alaikäisen päätöskyvyn huomautusta siten, että myös se näyttää ilmoituksen Bootstrapin alert-luokkaa käyttäen. Ilmoitusten päivityksen tarkoitus on yhtenäistää eri osioiden ulkoasua. Toinen hyöty ilmoituksien päivityksillä on se, että Bootstrapin alert-luokka on ulkonäöltään ja toiminnallisuudeltaan samankaltainen Material UI:n Alert-komponentin kanssa (9). Uudistuksessa varaudutaan siten myös tulevaa reseptilomakkeen React-uudistusta varten.

Seuraavana työtehtävänä selvitin tallennetun reseptin hoitotapahtuman säilymisen lomakkeella, kun resepti avataan uudelleen ennen lähettämistä. Lomakkeen toiminnan selvittäminen oli melko työlästä ja haastavaa, sillä nykyisessä toteutuksessa lomakkeen toiminta nojaa vahvasti palvelinpuoliseen istuntoon sekä siellä sijaitsevan datan käsittelyyn ajan myötä laajenneessa kontrollerissa. Visual Studion virheenetsintätyökalua käyttäen sain selville, että ongelma johtuu sekaannuksesta palvelutapahtumien ja hoitotapahtumien välillä.

Kela vaatii, että reseptille on mahdollista tallentaa palvelutapahtuma reseptiä kirjoittaessa. Esko-potilastietojärjestelmän käyttäjät ovat tottuneet valitsemaan erinäisissä osioissa hoitotapahtumia eivätkä palvelutapahtumia, joten monissa osioissa näytetään käyttäjille hoitotapahtumia palvelutapahtumien sijaan mahdollisuuksien mukaan. Reseptilomakkeella käyttäjä valitsee hoitotapahtuman, jota vastaa yksiselitteinen palvelutapahtuma. Ongelmia on aiheuttanut sen sijaan

se, että lomakkeella valitaan hoitotapahtuma, mutta tallennetaan ja lähetetään vain palvelutapahtuma. Jokaista hoitotapahtumaa vastaa yksiselitteinen palvelutapahtuma, mutta palvelutapahtumaan saattaa liittyä useampi hoitotapahtuma. Reseptin uudelleen avaamisen yhteydessä ei siten ole mahdollista asettaa lomakkeelle aiemmin valittua hoitotapahtumaa, sillä nykyisessä toteutuksessa se tieto on hävinnyt kokonaan.

Selvityksen jälkeen järjestimme lisäpalaverin reseptin kehittämiseen osallistuvien sovelluskehittäjien kanssa ja esitin kyseisessä palaverissa aiemman selvityksen. Palaverissa mietittiin yhdessä vaihtoehtoja korjauksen teknisestä toteutuksesta. Korjaukseksi valittiin hoitotapahtuman lisääminen piilotietona lomakkeelle ja sen tallentaminen istuntoon tallennuksen yhteydessä. Tämä ratkaisu mahdollistaa reseptilomakkeen nykyisen toiminnan säilymisen mahdollisimman hyvin ja on myös yhteensopiva tulevan lomakkeen React-uudistuksen kanssa, jossa kyseinen tieto voidaan säilyttää React-sovelluksen tilassa vielä paremmin. Tässä ratkaisussa tuli kuitenkin huomioida se, että hoitotapahtuma on tärkeä vain sovelluksen sisäisessä logiikassa ja sitä ei tule lähettää Reseptikeskukseen, sillä Esko-potilastietojärjestelmän ulkopuolella vain palvelutapahtumalla on merkitystä. Toinen reseptiuudistusta tekevä sovelluskehittäjä otti työkseen varmistaa, että tämä ratkaisu ei aiheuta ongelmia reseptin lähettämisessä Reseptikeskukseen.

Korjaustavan valinnan jälkeen sain toteutettua hoitotapahtuman tallentamisen piilotietona ja sen lisäämisen palvelimen istuntotietoon. Tämän tehtävän toteutuksesta jäi tällä viikolla toteuttamatta vielä logiikan lisäys, jonka perusteella lomakkeelle valitaan oletuksena oikea hoitotapahtuma.

4.1.3 Pohdinta

Työtehtävät keskittyivät tällä viikolla potilastietojärjestelmän osa-alueelle, josta minulla ei ollut sillä hetkellä paljon tietoa. Alaikäisen puolesta asiointi -ominaisuutta oli tehnyt pääosin toinen suunnittelija, joten työ vaati oma-aloitteista koodin tutkimista sekä kysymyksiä nykytoiminnasta ja toivotusta lisäominaisuudesta.

Tämän viikon työtehtävät olivat siltä osin hyödyllisiä jatkon kannalta, että niiden parissa tutustuin tarkemmin reseptilomakkeen nykyiseen back end-toteutukseen. Olin aiemmin tehnyt tulevaa reseptilomakkeen React-uudistusta, mutta toteutus siihen tähän mennessä ei ole vaatinut back end

-uudistusta eikä sen nykyinen toiminta ole sen takia tuttua. Tiedossa kuitenkin on, että tuleva uudistus tulee vaatimaan muutoksia sinne, joten tämän viikon tehtävistä saadusta osaamisesta tulee olemaan apua tulevaisuudessa.

4.2 Viikko 2

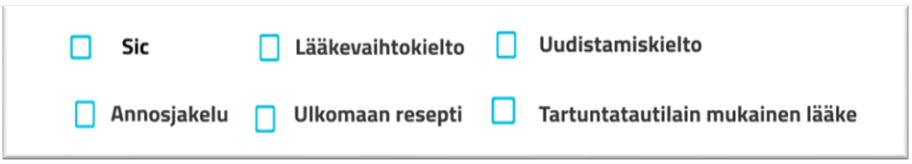
4.2.1 Työtehtävät

Edellisen viikon työtehtävistä jäi toteuttamatta logiikka sille, että lomakkeelle tulee valituksi tallennettu hoitotapahtuma, jos sellainen oli olemassa. Tälle korjaukselle löytyi looginen paikka eräässä C#:ssä tehdyssä ReseptiModel-tiedostossa, jossa oli jo mahdollista saada tieto aiemmasta oletuksesta eli hoitosuhteen varmistuksessa valitusta tapahtumasta sekä tallennetusta reseptin hoitotapahtumasta. Modelin ulkoinen toiminta ei muuttunut muutoksen jälkeen, joten korjaus oli siltä osin helppo toteuttaa.

Toteutuksen jälkeen muutos oli myös vietävä oikeaan testiympäristöön. Alaikäisen puolesta asioinnin ominaisuudella oli omat vaatimukset testiympäristölle sen takia, että sen toiminta nojaa toisiin Esko-osasovelluksiin, joissa on myös alaikäisen puolesta asiointiin liittyviä uudistuksia. Viennissä oli siis huomioitava, että ympäristöön tulee kaikki lääkehoidon osalta tarvittavat alaikäisen puolesta asioinnin ominaisuudet. Testiympäristöön viennin jälkeen oletushoitotapahtuman tallentaminen toimi myös testiympäristössä toivotulla tavalla, joten tämä tehtävä oli sillä suoritettu.

Edellisellä viikolla pohdittiin reseptin korjauksessa hoitotapahtuman valinnan muuttamista ja sen toteutusta. Tätä mietittiin enemmän tällä viikolla palaverissa ja siihen oli myös pyydetty Kelalta tarkennusta. Kelalta tullut vastaus viittasi Kelan dokumenttiin “Sähköinen lääkemääräys vaatimusmäärittely” kohtaan 3.9: "Käyttäjä voi korjata lääkemääräyksen annostelua ja muita ohjeita koskevia tietoja, mutta ei potilaan henkilötunnusta tai lääkemääräyksen laatijan tietoa tai muita tunnistetietoja (mm. määräyspäivä, paikka tai reseptin lajia)". Vastaus tarkensi myös, että palvelutapahtuma kuuluu noihin tunnistetietoihin. Tätä tehtävää ei siis voi selvityksen perusteella tehdä. (10.)

Peruuntuneen tehtävän tilalle valitsimme sopivan tehtävän backlogista, jonka ehti tehdä vielä tässä pyrähdyksessä. Tehtävä valittiin eReseptilomakkeen React-uudistuksen backlogista. Tehtävässä kuvattiin kuuden valintaruudun lisäämistä lomakkeelle. Tehtävän suunnittelussa käytin apuna Figmassa tehtyä hahmotelmaa, jonka tuoteasiantuntijat ovat tehneet reseptilomakkeen uudistusta suunniteltaessa. Figma on graafisen suunnittelun työkalu, jota erinäiset tiimit käyttävät yrityksen sisällä UI-suunnitelmien tekemiseen. Kuvassa 1 on Figmassa hahmoteltu suunnitelma valintaruuduista. (11.)



KUVA 1. Kuvakaappaus Figman lääkehoidon näkymästä.

Ainoa toiminnallisuus näillä valintaruuduilla on niiden tilan säilyttäminen Redux-tilanhallintajärjestelmässä, sillä tuleva React-reseptilomake on vasta UI-toteutuksen vaiheessa eikä sillä pysty vielä luomaan reseptiä edes testiympäristössä.

Yksi asia, mistä huomautettiin aloittaessa, on se, että vaikka tehtävään oli kuvattu kuusi valintaruutua, sinne on tulossa alaikäisen puolesta asioinnin yhteydessä vielä seitsemäs. Tehtävän olisi voinut toteuttaa esimerkiksi lisäämällä kaksi riviä ja laittamalla jokaiseen riviin kolme valintaruutua vierekkäin. Koska tässä tapauksessa tiedettiin, että rivien määrä ja sisältö tulee vaihtumaan, parempi ratkaisu mielestäni oli tehdä dynaamisesti valintaruututaulukko käyttäen Material UI:n Grid-komponenttia. Tällä toteutuksella uuden valintaruudun tekeminen tulevaisuudessa on helppoa, sillä taulukkoon lisätään vain uuden valintaruudun tiedot ja Grid tuottaa sen perusteella sopivan määrän rivejä ja valintaruutuja. Tässä toteutuksessa valintaruutu on myös toteutettu uudelleenkäytettävänä React-komponenttina, joka mahdollistaa sen käyttämisen muualla tai sen vaihtamiseen johonkin toiseen.

Kuvassa 2 on kuvakaappaus kehitysympäristössä olevasta toteutuksesta suunnitelmaan vertailua varten.

KUVA 2. Kuvakaappaus valintaruuduista kehitysympäristössä.

4.2.2 Pohdinta

Tällä viikolla peruuntunut hoitotapahtuman valinnan muutos reseptin korjauksessa on esimerkki siitä, miten haastavaa potilastietojärjestelmän kehitys voi olla. Tehtävä oli pieni osa-alue isosta kokonaisuudesta, mutta kokonaisuuden suunnitteluun oli silti osallistunut useampi Eskon kehitystiimi ja näiden sisällä useampi Esko-asiantuntija.

Tehtävän määrittelyssä tulee myös usein se ongelma, että tehtävän on saattanut kirjata lääketieteellisen taustan omaava Esko-asiantuntija, jolla ei ole taustaa ohjelmoinnista. Tehtävä oli aiemmin suunniteltu yhden valintaruudun perusteella, jonka sijaintia muutetaan React-uudistuksen yhteydessä. Keskustelujen perusteella sain tarkemman kuvauksen asiasta ja yhteysymmärryksessä muutimme tehtävän kuvauksen siten, että tämän tehtävän yhteydessä voi samalla vaivalla React-tyylisesti tehdä yhden komponentin valintaruudulle ja lisätä kaikki kuusi kerralla lomakkeelle.

4.3 Viikko 3

4.3.1 Suunnittelu

Tämän pyrähdyksen aikana siirryin takaisin eReseptilomakkeen React-uudistukseen pariin. Tähän mennessä olin rakentanut nykyistä React-uudistusta eReseptilomakkeelle itsenäisesti, mutta tässä pyrähdyksessä tähän tuli mukaan toinen Lääkehoito-tiimin kehittäjä. Työnjako sovittiin aluksi siten, että minä aloitin laajemman teknisen uudistuksen back endin kontrollerin kanssa ja toinen kehittäjä tutustui nykyisen toteutukseen ja jatkoi reseptilomakkeen kenttien parissa.

Ainoa ohjelmistokehittämiseen liittyvä tehtävä tälle pyrähdykselle oli sovittaa React-uudistettu reseptilomake yhteensopivaksi nykyisen back endin kanssa. Uudistuksen yhteydessä haluttiin, että

back endin sessioon tallennettu tieto siirretään React-uudistuksen sisälle hallittavaksi. Tätä varten oli tarkoitus tehdä uusi kontrolleri, joka on yhteensopiva React-uudistetun reseptilomakkeen kanssa. Tämän toteutuksen tekeminen suoraan oli kuitenkin todettu liian monimutkaiseksi, sillä nykyisessä kontrollerissa on liian paljon sessioon kytkeytyneitä osioita. Nykyisen suunnitelman mukaan tehdään sittenkin niin, että kontrollerin rajapinnat päivitetään JSON-muotoisiksi ja yhteensopiviksi React-uudistuksen kanssa. Uuteen kontrolleriin siirrytään siten askeleittain ja tämä mahdollistaa myös pienempien kokonaisuuksien siirtämisen React-puolelle sen sijaan, että koko uusi kontrolleri ominaisuuksineen luodaan tyhjästä.

Toisena tehtävänä oli aiempaan tehtävään liittyvä dokumentin päivittäminen. Kyseessä on tiimin sisäinen tekninen dokumentti, jota ylläpidetään säännöllisesti. Tälläisia dokumentteja on hyvä pitää ajan tasalla, sillä joku toinen kehittäjä saattaa tarvita tietoa siitä, millaisia muutoksia järjestelmään on tehty.

4.3.2 Työtehtävät

Toteutusta aloittaessa olimme suunnitteluissa tulleet yhteisymmärrykseen siitä, miten React-koodin tulisi ottaa yhteys back endin kontrolleriin. Yksi asia oli vielä jäänyt avoimeksi eli se, miten lääkehaussa valitun lääkkeen tieto saadaan Reactin tiedoksi. Nykyisessä toteutuksessa käyttäjä valitsee lääkehausta halutun lääkkeen, jonka perusteella lomake palautuu back endistä ja aukeaa näytölle. Lääkehaku on jQuery-toteutusta ja sitä ei haluta vielä uusia tässä yhteydessä. Reseptilomakkeen React-osuuden on siis saatava tieto valitusta lääkkeestä, vaikka sitä ei ole saatavilla sillä hetkellä, kun lomake käynnistyy React-puolella. Toteutusta vaikeuttaa se, että reseptilomake tulee voida avata erityyppisillä lääkerakenteilla eikä välttämättä vain lääkerekisteristä löytyvillä valmisteilla.

Ratkaisuksi ehdotin, että lääkehaku ottaa toistaiseksi yhteyden back endiin kuten ennen, mutta back end ei tee mitään toiminnallisuutta, vaan palauttaa vain takaisin valitut lääkehaun tiedot piilotietona React-lomakkeen lisäksi. React-reseptilomake voi siten etsiä kirjatut tiedot DOM-hakua käyttäen ja näiden perustella avata uuden yhteyden kontrolleriin. Kaikki toiminnalliset yhteydet back endin kontrollerin kanssa tulevat siten React-lomakkeen kautta. Muilla kehittäjillä ei ollut tähän esittää mitään vaihtoehtoista ratkaisua, joten etenin tällä suunnitelmalla.

Reseptiä tehtäessä käyttäjä ensin valitsee lääkehausta valmisteen, jolle resepti tehdään. Käyttäjä voi valita lääkerekisteristä löytyvän aineen, geneerisen valmisteen, ex tempore -valmisteen tai muun valmisteen. Asiaa tutkittaessa todettiin, että ex tempore -valmistetta käyttäessä toteutus poikkeaa muista sen verran, että asiaa täytyy tutkia enemmän ja tehdä siihen erillinen ratkaisu. Ex tempore -valmisteen tapausta ei huomioida toistaiseksi ja sen toteutus tehdään myöhemmässä vaiheessa.

Tämän viikon aikana ehdin tehdä tähän suurimmalta osalta back end-puolisen toteutuksen. Lääkehaku palauttaa nykytoteutuksessa back endistä reseptilomakkeen HTML-muotoisena. Lomakkeessa on React-koodin hallinnoima div-elementti, viittaus React-koodin käynnistävään JavaScript-tiedostoon sekä tarvittavat piilotiedot, joilla React-koodissa voidaan hakea lomakkeen alustukseen tarvittava tieto JSON-muodossa. Aiemmassa toteutuksessa palvelimen sessio alustettiin jo tässä vaiheessa, mutta alustus toteutetaan jatkossa vasta sitten, kun React-puolelta otetaan yhteys palvelimeen.

Toteutuksesta puuttuu vielä täysin front endin eli React-puolen toteutus. React-koodissa on luettava piilossa oleva tieto lääkehaun valmisteesta. Sen jälkeen React-puolelle on toteutettava rajapinta, jolla voidaan hakea lomakkeen alustukseen tarvittava data palvelimelta.

4.3.3 Pohdinta

Toisen kehittäjän React-lomakkeen kehitykseen siirtymisen yhteydessä oli pari palaveria, joissa tuli käytyä nykyisen toteutuksen toimintaa läpi. Tämä oli hyvä tilaisuus kerrata itsellekin millä tavalla sitä on rakennettu tähän mennessä ja miettiä myös, millä tavalla siihen kannattaa rakentaa uutta.

En ole henkilökohtaisesti täysin tyytyväinen kaikkiin ohjelmallisen toteutuksen rakenteen teknisiin ratkaisuihin, mitä tällä viikolla toteutettiin. Useammassa ohjelmistosuunnittelijoiden palaverissa olemme miettineet, miten voimme järkevästi toteutettavalla tavalla siirtää logiikkaa back endistä React-koodin puolelle, sillä palvelinpuolisesta sessiosta halutaan päästä eroon. Tätä on kuitenkin hyvin vaikea tehdä kerralla. Sen takia tavoitteena on saada lomakkeesta ensin toimiva ja vasta sen jälkeen pala palalta eritellä toimintoja uudelle sessiottomalle back endin kontrollerille sekä React-koodin tilaan.

4.4 Viikko 4

4.4.1 Työtehtävät

Työtehtävänä oli jatkaa yhteensopivuuden rakentamista back endin kanssa. Olin viime viikolla toteuttanut useimmat back endin tarvittavat muutokset, tosin virheenkäsittelyn osalta niissä oli toteutus vielä kesken. Otin kuitenkin ensimmäiseksi työn alle React-puolen uudistuksen.

Lomakkeen avausta muutettiin siten, että sen sijaan että lomake avautuisi suoraan näkyviin, React-komponentti näkyy aluksi ainoastaan Material UI Progress-komponenttina, jolla kuvatan käyttäjälle että lataus on edelleen käynnissä (12). React-koodissa etsitään DOM-haulla id:n perusteella komponentti, jonka sisällä on lääkehaussa muodostettu data. Koska lääkehakua käyttäen voidaan luoda reseptilääke käyttäen lääkerekisterin valmistetta, geneeristä valmistetta tai muuta valmistetta käyttäen, näille tarvitaan oma komponentti eri id:llä. Back end kuitenkin aina palauttaa tasan yhden näistä, joten React-koodissa voidaan päätellä löydetyn id:n perusteella, millä tavalla lomake on avattu.

Datan lukemisen lisäksi tuli toteuttaa reseptilomakkeen avaamiseen tarvittavan tiedon hakeminen back endistä. Tein tätä varten moduulin, johon tein axios-kirjastoa käyttäen toteutuksen back endistä datan hakemista varten jokaiselle avaustavalle erikseen (13). Moduulin rajapintoina ovat joka avaustavalle erilliset asynkroniset metodit, joille annetaan parametreinä tarvittava lääkkeeseen liittyvä tieto. Rajapintojen palautuksena tulee kontrollerista haettu data. Back endissä oli jo aiemmassa toteutuksessa yhtenäistetty lomakkeella oleva tieto C# ReseptiModel-mallissa, joten onnistuneessa avauskerrassa tieto tulee samassa muodossa riippumatta siitä, millä lääkehaun tavalla reseptilomake on avattu.

Lisäsin tämän jälkeen reseptilomakkeen alustuksen yhteyteen lomaketiedon haut käyttäen aiemmin luodun moduulin rajapintoja. Lomakkeen onnistuneen avauksen jälkeen lomake voidaan nyt näyttää lataavan Progressin sijaan. Alustuksessa saadaan myös back endistä tarvittava data, joka voidaan tallentaa React-sovelluksen tilaan. Lomakkeen React-toteutuksessa käytetään jo Redux-kirjastoa tilanhallintaan, joten back endistä tuleva tieto tulisi myös yhtenäistää nykyisen Reduxissa sijaitsevan tilan kanssa. Tämä ei kuulu kuitenkaan vielä tämän tehtävän toteutukseen ja sitä tulee suunnitella enemmän tulevassa tehtävässä.

Virheenkäsittelyn osalta mainittakoon, että aiemmassa mallissa back end palautti aina HTML-muotoista koodia, jolla renderöitiin lomakkeen näkymä. Joissakin tapauksissa palautettu näkymä ei ollut lomake, vaan virheviestin näyttävä näkymä. Näiden toteutus muutettiin siten, että palautus on nyt sopivalla REST virhekoodilla varustettu vastaus, jossa on myös JSON-muotoinen virheen selite. Virheenkäsittelyn viimeistelyyn tulisi myös React-puolelle luoda komponentti, jolla näytetään tämä virheviesti. Tämä jätetään myös myöhemmin toteutettavaksi.

4.4.2 Pohdinta

Reseptilomakkeen Redux store -tilan rakenne toteutettiin siten, että kenttiä lisättiin tarpeen mukaan lomakkeelle. Toisin sanoen sen rakenne ja tietotyypit ovat sellaisia, että ne sopivat lomakkeen käytettäväksi ja hallittavaksi, mutta eivät sovellu kovin hyvin siihen, että ne alustetaan back endistä tulevalla tiedolla eivätkä myöskään siihen, että tallennettu tieto voitaisiin lähettää sellaisenaan back endille, kun reseptiä tallennetaan tai luodaan.

Kyseinen ratkaisu aiheuttaa nyt sen, että reseptilomakkeen puolella ei ole nyt varauduttu siihen, millä tavalla yhdistetään lomakkeen tieto back endin kanssa ja esimerkiksi alustetaan lomake back endistä tulevalla tiedolla. Tähän tulee miettiä sopiva ratkaisu reseptipuolen teknisten osaajien kanssa. Omasta mielestäni reseptilomakkeen alustus back endin tiedoilla kannattaisi hoitaa reducerin kautta, mutta muuten asiassa on vielä paljon mietittävää.

4.5 Viikko 5

4.5.1 Suunnittelu

Aiemmalla viikolla oli mainittu, että yhteensopivuudessa ei otettu toistaiseksi huomioon ex tempore -valmisteella tehtyä reseptiä. Tämän viikon ensimmäisessä ja isommassa tehtävässä oli tarkoitus selvittää tähän sopiva ratkaisu ja myös toteuttaa se.

Toisessa tehtävässä reseptilomakkeelle lisättiin uusia tietokenttiä: Annostus voimaan, hoitolaji ja käyttötarkoitus. Annostus voimaan ja hoitolaji voitiin toteuttaa Figman suunnitelman mukaisesti valintanappeina. Käyttötarkoitus on vain tekstikenttä, johon käyttäjä voi kirjata vapaamuotoisesti

käyttötarkoituksen. Näiden lomakkeiden tilat säilötään myös muiden reseptilomakkeen kenttien kaltaisesti Redux-kirjaston hallitsemaan tilaan. Toteutus on näiden osalta melko suoraviivainen, koska lomakkeella on jo samankaltaista toteutusta.

4.5.2 Työtehtävät

Ex tempore -valmisteen reseptin luomisesta on hyvä ymmärtää, millä tavalla reseptin kirjoittaminen eroaa muihin verrattuna. Pääosin ex tempore -valmisteella reseptin luominen eroaa kahdella tavalla. Lääkerekisterin valmisteella, geneerisellä valmisteella ja muulla valmisteella avattuna kontrollerissa tarvitaan vain yhteen valmisteeseen liittyvää tietoa. Ex tempore -valmiste taas apteekissa apteekkarin valmistama yhdistelmä rajattomasta määrästä eri lääkevalmisteesta. Ratkaisua täytyy siis täydentää sillä tavalla, että lääkevalmisteen tietona voi tulla ex tempore -valmisteen tapauksessa lista lääkevalmisteita.

Toinen eroavaisuus on luonteeltaan tekninen. Muissa tapauksissa kuin ex tempore -lääkkeellä lomakkeen avaamisessa kaikki lääkevalmisteen tiedot lähetettiin back endiin kerralla, jonka avulla back endin sessioon alustettiin lomakkeen tiedot ja palautettiin takaisin lomakkeen renderöivä HTML-koodi. Ex tempore -lääkkeen tapauksessa tämä tehtiin kaksivaiheisesti. Ex tempore -lääkkeellä lähetettäessä haettiin ensin back endiltä pohja reseptilomakkeen renderöintiä varten, mutta lähetyksessä ei annettu lääkevalmisteiden tietoa. Kun reseptilomakkeen pohja on saatu back endiltä, ex tempore -toteutuksessa lähetettiin erillisellä pyynnöllä lääkevalmisteiden tiedot ja alustettiin back endin puolella vasta sitten sessioon lääkkeen tiedot. Back end palautti tässä tapauksessa vain rajatummalla tavalla lääkekentän tiedot eikä koko reseptilomaketta, sillä pohja palautui jo aiemmin.

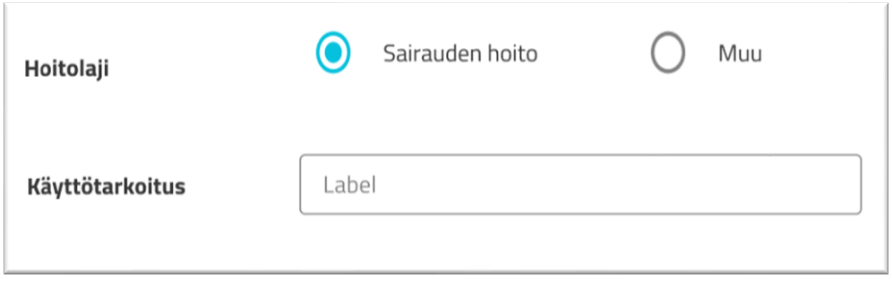
Ensimmäisenä piti ratkaista, miten lääkevalmisteen tiedot saadaan reseptilomakkeen React-toteutukseen. Minulla oli tähän ehdottaa kaksi vaihtoehtoista ratkaisua oman selvityksen perusteella. Ensimmäisessä vaihtoehdossa ex tempore -valmisteella avausta voitaisiin muuttaa siten, että lääkevalmisteen tiedot lähtevät jo heti ensimmäisessä pyynnössä back endille. Tällöin back endille voidaan tehdä samankaltainen ratkaisu, jossa lääkevalmisteen tiedot palautetaan DOM:n mukana piilotietona takaisin listamuodossa ja luetaan uudestaan React-puolella. Tämän jälkeen React-puolella voidaan osittain käyttää muilla avaustavoilla olemassa olevaa toisen avausvaiheen rajapintaa lomakkeen alustamiseen back endin sessiopuolella.

Toisessa vaihtoehdossa huomioin, että ex tempore -valmisteen aikaisemman toteutuksen käyttöä olisi mahdollista osittain jatkaa tietyillä muutoksilla. Kun ex tempore -lääkkeen tietoja täydennetään lääkehakukenttään, siinä luodaan jo HTML:n ul-komponenttimuotoinen lista lääkevalmisteista kaikkine tietoineen. Ex tempore -lääkkeen aiemmassa toteutuksessa luettiin tämä lista ja muut tarvittavat tiedot, kun tietoja lähetettiin back endiin. Koska tämä toteutus on osa lääkehaun JavaScript-koodia, tätä listan lukua ja lähetyksen muodostamista olisi mahdollista käyttää uudestaan myös React-puolella. Back endin rajapintojen toteutusta ei tarvitsisi muuttaa juuri lainkaan, vaan riittäisi, että back endin sessiota alustaessa palautettaisiin tieto JSON-muodossa.

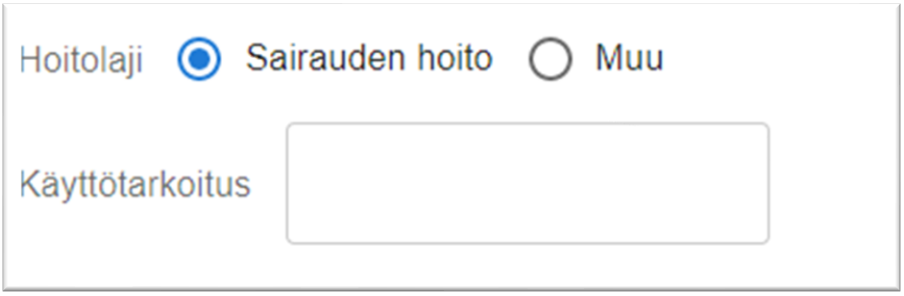
Esitin nämä vaihtoehdot reseptipuolen kehittäjille ja alustava päätelmä oli, että voidaan toteuttaa toisen vaihtoehdon mukaisesti. Ex tempore -toteutusta ei aloitettu vielä tällä viikolla, sillä yksityiskohtia pitää vielä tutkia ja harkita vaihtoehtoja.

Ex tempore -tapauksen selvittämisen lisäksi tein tällä viikolla toisen tehtävän, jossa reseptilomakkeelle lisättiin tietokenttiä. Kuten suunnittelussa mainittiin, näiden lisääminen oli melko suoraviivaista ja lomakkeella oli jo aiempaa samankaltaista toteutusta. Toteutukseen riitti tehdä ”Annostus voimaan”- ja ”Hoitolaji”-valinnoille valintanapit sekä vapaamuotoinen tekstikenttä ’Käyttötarkoitus”-kentälle. Käyttöliittymän puolesta näihin käytettiin Material UI:n ratkaisuja RadioGroup ja TextField. Tietokenttien tila tallennettiin muiden lomakkeen tietokenttien kaltaisesti Reduxin tilanhallintaan.

Havainnollistamisen vuoksi kuvassa 3 on Figmassa tehty suunnitelma ja kuvassa 4 kehitysympäristössä tehty toteutus. Kuvissa ei näy Annostus voimaan -valintaa, sillä se sijaitsee eri kohdassa reseptilomaketta. Sen osalta suunnitelma ja toteutus ovat samankaltaisia hoitolaji-valinnan kanssa.



KUVA 3. Kuvakaappaus Figman lääkehoidon näkymästä.



KUVA 4. Kuvakaappaus tietokentistä kehitysympäristössä.

4.5.3 Pohdinta

Selvitystä aloittaessa ei ollut selvää ymmärrystä sille, miksi ex tempore -tapauksessa reseptilomakkeen avaaminen on toteutettu teknisesti hyvin eri tavalla. Alkuperäinen toteutus on tehty sen verran kauan sitten, että kehittäjillä ei ollut tästä varmaa tietoa, kun tätä asiaa mietittiin.

Toiminnallisuuden perusteella yksi mahdollisuus on se, että tämä liittyy lääkevalmisteen vaihtoon. Ex tempore -lääkettä luotaessa lääkehaussa ainesosat lisätään yksitellen ja niitä kerääntyy listana lääkehakuun. Tätä listaa käytetään ex tempore -valmisteella avattaessa toisessa vaiheessa, sillä lähetetty FormData muodostetaan sen tietojen perusteella. Kaksivaiheista avaamisprosessia käytetään myös sillä lailla toteutuksessa hyväksi, että jos reseptilomake on jo avattu ja annostusta muutetaan, muut lomakkeen kentät säilytetään sellaisenaan ja vain ex tempore -annostuksen kentät renderöidään näkymälle uudelleen.

Tämä näyttää olevan ex tempore -lääkkeen ominaisuus, joka ei sellaisenaan säily uudistuksen jälkeen. Jos järjestelmässä vaaditaan, että tämä ominaisuus tulee säilyttää, sen voi toteuttaa eri tavalla myöhemmin. Ex tempore -ominaisuutta pitää miettiä asiantuntijoiden kanssa vielä jatkossa, kun toteutusta saadaan pidemmälle.

4.6 Viikko 6

4.6.1 Työtehtävät

Viime viikon selvityksen perusteella voitiin toteuttaa ex tempore -valmisteella avatun reseptilomakkeen yhteensopivuuden lisääminen React-reseptilomakkeelle. Suunnitelman mukaan pyrittiin käyttämään olemassaolevaa toteutusta mahdollisimman paljon.

Palvelinpuolella muutokset eivät olleet suuria. Muissa tapauksissa kuin ex tempore -lääkkeellä React-reseptilomaketta avattaessa back endistä lähetetään piilotietona DOM:n sisällä tarvittavat lääkkeen tiedot. Ex tempore -tapauksessa ne päätettiin lukea suoraan lääkehaun listasta. Tällöin ex tempore -tapauksessa riittää, että DOM:in sisälle laitetaan tieto siitä, että reseptilomake on avattu ex tempore -lääkkeellä.

Toinen palvelinpuolen muutos liittyi rajapintaan, jonka avulla reseptilomake aiemmin alusti palvelimen session ja palautti lomakkeen tiedot. Tämä oli verrattavissa siihen, miten muissa tapauksissa Reactin puolelta käynnistettiin session alustus uutta rajapintaa käyttäen ja haettiin lomakkeen tiedot. Kyseiselle rajapinnalle riitti, että rajapintaa muutettiin palauttamaan JSON-muotoista tietoa.

Lääkehaun JavaScript-toteutuksesta oli poistettava toiminnallisuus, jonka avulla ReseptiControllerin sessio alustettiin alun perin ex tempore -lääkkeen tapauksessa. Jatkossa kyseiseen rajapintaan otetaan yhteys Reactin puolelta.

Reactin puolella saatiin back endin muutoksien jälkeen tieto siitä, että lomake on avattu ex tempore -lääkkeellä. Lääkkeen tietojen lukemiseen voitiin käyttää lääkehaussa olemassa olevaa toteutusta, tosin sitä muokattiin sopivaksi, sillä Reactin puolelle ei haluttu ottaa mukaan suoraan jQuery-koodia. Palvelimen puolella sessiota alustettaessa lääkkeen tiedot otettiin vastaan samalla rajapinnalla, missä ne tuli antaa FormData-muodossa. FormData-kentän tietojen muoto ilmeni lääkehaun JavaScript-toteutuksesta. Axios-kirjastolla pystyi lähettämään tiedot myös kyseisessä muodossa. Lopputuloksena ex tempore -lääkkeellä reseptilomakkeen avaus saatiin toimivaan melko lailla samalla tavalla kuin muilla valmisteilla.

4.6.2 Pohdinta

Toteutuksessa tuli selvittää FormData-muotoisen tiedon käsittelyä ja lähettämistä, sillä päätettiin olla uudistamatta back endin vastaanottavaa rajapintaa tässä yhteydessä. Itselleni tämä ei ollut tuttu asia, sillä modernissa kehityksessä on käytetty enemmän REST-pohjaisia rajapintoja ja itsellä JSON-pohjaisia. Työtä aloittaessa ei ollut myöskään tiedossa, kuinka helposti tietojen lähettämiseen käytettävä node-moduuli axios toimii FormData-muotoisen datan lähettämisen kanssa.

Kokemus molemmista näistä oli, että teknisesti niitä oli helppo käyttää. Alkuperäisessä lääkehaussa tietojen muodostaminen ja lähettäminen oli tehty eri lailla serialisointia käyttäen, mutta FormData-olion luominen ja siihen tietojen lisääminen oli melko suoraviivaista ja ymmärrettävää. Samoin axios-moduuli on sen verran hyvin suunniteltu, että se taipui FormData-muotoisen tiedon lähettämiseen ilman suurempia ongelmia lisäämällä vain tieto siitä, millaista tietoa lähetetään.

4.7 Viikko 7

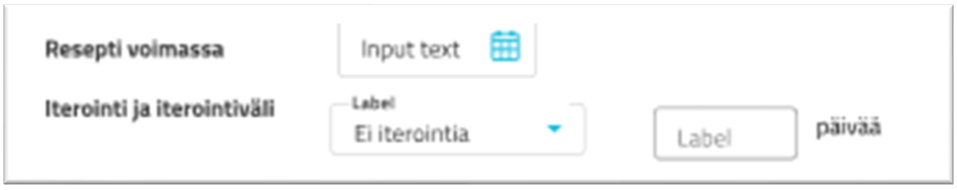
4.7.1 Suunnittelu

Ensimmäisessä tehtävässä lisätään jälleen tietokenttiä reseptilomakkeelle. Lomakkeelle lisätään kalenteritietokenttä, jossa määritellään aika, jonka resepti on voimassa. Lisäksi lomakkeelle lisätään tietokentät iteroinnille ja iterointivälille. Iterointi on valikko, josta voidaan valita, että lääkemääräys toimitetaan uudelleen kerran, kahdesti tai kolme kertaa. Iterointiväliin voidaan kirjoittaa päivinä iterointiväli, mikä kuvaa toimituksien välissä olevaa aikajakson pituutta.

Toinen tehtävä tälle pyrähdykselle on jatkoa back endin ja front endin yhteyden suunnittelulle. Tehtävä on varautumista tulevalle tekniselle uudistukselle, missä yhdistetään tietomallit siten, että myös front endin puolella käytetään back endissä määriteltyä tietomallia. Sitä ennen päätettiin kuitenkin parantaa yhteyden muodostamista ja eheyttää tietomallin rakennetta. Tehtävää ei ehditty suunnitella loppuun asti ennen suunnittelupalaveria, vaan viikon aikana oli tarkoitus määritellä asia tarkemmin.

4.7.2 Työtehtävät

Tietokenttien graafisen ilmeen osalta nojataan Figmassa tehtyyn hahmotelmaan. Kuvassa 5 on kuvakaappaus Figman suunnitelmasta tästä osiosta.



KUVA 5. Kuvakaappaus Figman lääkehoidon näkymästä.

Toteutuksessa voidaan soveltaa sopivia Material UI-komponentteja. Näistä löytyy myös olemassaolevaa toteutusta muualta React-uudistetussa reseptilomakkeessa, joita voidaan hyödyntää. Tietokenttien tila säilytetään muiden kanssa Reduxin tilanhallintajärjestelmässä. Kuvassa 6 on kuvakaappaus toteutetuista tietokentistä kehitysympäristössä.

KUVA 6. Kuvakaappaus reseptilomakkeen tietokentistä kehitysympäristössä.

Ainoa erikoisuus tähän tehtävään liittyen on, että reseptin voimassaolon loppumiseen laitetaan oletuksena päivämäärä kaksi vuotta eteenpäin. Oletusarvo voidaan tässä vaiheessa määritellä Reduxin tilaan, kun tilaa alustetaan. Tämä toteutus tulee myös ottaa huomioon myöhemmässä vaiheessa, kun Reduxin tilaa muutetaan siten, että se käyttää yhteensopivaa tietomallia back endin kanssa.

Toinen asia tähän liittyen on se, että iteroinnin valinnat ovat itse asiassa kansallisessa koodistopalvelussa määriteltyjä (14). Tehtävää tehtäessä huomattiin, että näiden toteuttamisessa ja muutaman muunkin valikon vaihtoehdot tulisi tulla back endin puolelta. Tätä ei kuitenkaan toteutettu vielä tässä vaiheessa.

Tämän viikon aikana mietittiin reseptin kehittäjien kanssa myös jatkoa back endin ja front endin yhteydelle. Nykymallissa back endin puolella on vanhaan toteutukseen nojaava Model, joka sisältää monia tietokenttiä, mitä front endin puolella ei tarvita. Front endin puolella taas on Reduxin tilanhallinnassa tietokenttiä, joita on sinne lisätty tarpeen mukaan tehtäviä tehtäessä. Näiden yhdistämiseen halutaan nyt tehdä uusi Model back endissä, joka voisi sisältää kaikki lomakkeella tarvittavat tiedot.

Tehtävän yhteydessä suunniteltiin myös yhteyspyyntöjen vähentämistä. Ideana oli, että nämä tiedot lähetetään Base64-enkoodattuna JSON-muotoisena piilotietona, kun reseptilomaketta avataan. Tällöin tietoa ei tarvitse jatkossa hakea erikseen Reactin puolella, vaan riittää että se puretaan JSON-muotoiseksi Reactin puolella. Tämän tehtävän toteutus jäi seuraavalle viikolle.

4.7.3 Pohdinta

Teknisiä uudistuksia suunniteltaessa oli puhetta valikoista, joiden valinnat pohjautuvat Kansallisen koodistopalvelun määrittelyihin. Tällä viikolla toteutettu iterointi ja aiemmin toteutettu lääkkeenantoreitin valinta ovat esimerkkejä näistä. Näissä tulisi ottaa huomioon, että koodistot saattavat päivittyä ja muuttua ajan myötä.

Käytännössä tämän varmistamiseen riittää, että pidetään back endin puolella ajantasaista yhteyttä koodistopalveluun ja front endin puolella käytetään back endistä tulevaa tietoa koodistopalveluun nojaavissa tietokentissä. Tässä pyrähdyksessä suunniteltu tiedon mallintaminen soveltuu tähän oikein hyvin. Ongelman voi ratkaista tulevaisuudessa sillä lailla, että valikoiden vaihtoehdot luodaan dynaamisesti back endin puolelta saatujen tietojen perusteella. Tämä mahdollistaa myös sen, että kenttien arvoina voidaan käyttää koodistossa määriteltyä Oidia ja lähettää tämä suoraan back endiin sitten, kun lomakkeella voidaan tallentaa ja lähettää reseptejä.

4.8 Viikko 8

4.8.1 Työtehtävät

Tämän viikon työt keskittyivät React-uudistetun reseptilomakkeen yhteyden mallintamiseen viime viikon suunnitelman perusteella. Toteutus alkoi back endin puolella, missä luotiin uusi C#

ReseptiYhteysModel-tietomalli tietojen säilyttämiseen. Aiempaa ReseptiModel-mallia voitiin hyödyntää siten, että sieltä poimitaan sopivat tiedot uuteen tietomalliin konstruktorissa. Toteutuksen alussa sieltä poimitaan vain lääkevalmisteeseen liittyviä tietoja. Mallin tietosisältöä laajennetaan myöhemmässä vaiheessa, kun on kartoitettu enemmän, minkälaisia tietoja lopullisesti tarvitaan.

Aiemmin React-reseptilomakkeen avaamisen yhteydessä back endissä lähetettiin HTML-koodia, jossa oli pääosin kolme komponenttia: piilotieto valmisteen tyypistä ja tiedoista, React-koodin hallinnoima div-elementti ja viittaus JavaScript-koodiin, jolla React-komponentin renderöinti saatiin käynnistettyä. Piilotiedon valmisteen tyypistä ja tiedoista tilalle asetetaan koko ReseptiYhteysModelin sisältö. Tietomalli muutetaan ensin JSON-muotoiseksi ja sen jälkeen Base64-enkoodataan. Base64-enkoodaukseen voitiin käyttää .NET Frameworkissa jo olevaa System.Encoding-luokkakirjastoa ja projektissa oli jo Newtonsoft-paketti, jota voitiin hyödyntää JSON-muunnoksessa (15).

Reactin puolella oli jo aiemmilla viikoilla tehtyä toteutusta, joissa luettiin piilotiedosta lääkevalmisteen tietoja. Tätä muutettiin siten, että sieltä luetaan vain Base64-enkoodattu tieto lääkevalmisteen sijaan. Base64-enkoodaus voidaan purkaa Base64 MDN-ohjeiden mukaan (16). Koska tietomalli muunnettiin back endin puolella jo valmiiksi JSON-muodoksi, riittää front endin puolella vain jäsentää JSON.parse()-metodia käyttäen purettu JSON-tieto.

Tiedon voisi tämän jälkeen tallentaa sellaisenaan Reduxin säilöön. Käytännössä tässä vaiheessa kannattaa kuitenkin miettiä tilanteita, joissa tietomalli ei kuitenkaan vastaisi sitä muotoa, mikä vaaditaan front endin puolella. Toisen kehittäjän esimerkin mukaan tehtiin Reduxiin käsittelijäksi funktio, joka varmistaa, että Reduxiin säilötään aina oikean muotoista tietoa. Back endissä olevaa tietomallia ja samassa yhteydessä tätä käsittelijäfunktiota laajentamalla voidaan varmistaa, että Reduxissa tallennettu tila tulee olemaan tulevaisuudessa yhteensopivassa muodossa. Laajennusta on tarkoitus jatkaa tulevaisuudessa siten, että nykyisten tietokenttien tila tultaisiin jatkossa tallentamaan back endissä määriteltyyn muotoon. Nykytilanteessa uusi mallinnettu tieto on vain aiempien rinnalla.

4.8.2 Pohdinta

Reduxin käsittelijäfunktio oli mielestäni kätevä tapa varmistaa, että Reduxin tilaan tulee jatkossa hallittu tietomalli. Funktio toimii vain sillä lailla, että sille annetaan parametrina JSON-mallin mukainen JavaScript-olio ja siitä poimitaan halutun tietomallin mukaiset tietosisällöt. Jos niitä puuttuu tai funktiolle ei anneta mitään parametria, funktio alustaa niille kuitenkin järkevät oletusarvot, jolloin tila tulee olemaan kuitenkin ennustettavissa.

5 POHDINTA

Päiväkirjamuotoinen opinnäytetyö oli minulle luontainen valinta toteutusmuodolle, sillä se sopi hyvin toteutettavaksi täysipäiväisen työn ohessa. Vaikeinta toteutusvaiheessa oli raportointiosiossa tehtävänantojen ja toteutuksien kuvaaminen sillä lailla, että teksti olisi ymmärrettävää lukijalle, jolla ei välttämällä ole taustaa teknisestä osaamisesta, lääketieteellisestä tietotaidosta tai Esko-potilastietojärjestelmän käytöstä. Tätä vaikeutti myös se, että ratkaisujen teknisen toteutuksen kuvaamisessa piti tietoturvasyistä harkita tarkkaan, kuinka tarkalla tasolla yksityiskohtia pystyi kuvaamaan tässä raportissa. Seurantajakson aikana onnistuin kuitenkin pääosin kirjaamaan pyrähdyksen aikaiset osuudet pian toteutuksen jälkeen, kun asiat olivat vielä tuoreessa muistissa.

Seurantajakson aikana tein aluksi tehtäviä alaikäisen puolesta asioinnin ominaisuuteen ja myöhemmillä viikoilla reseptilomakkeen uudistukseen. Suurelta osalta tehtävissä korostui, että tehtävän suorittaminen vaati oma-aloitteista tutkimista. Tehtävä oli ehkä kuvattu toiminnallisuuden tasolla, mutta tekninen toteuttaminen vaati järjestelmän uuden osa-alueen toiminnan selvittämistä ja toteutuksen suunnittelua. Tämä oli paikoin haastavaa, mutta selvittelyn jälkeen toiminnasta jäi mieleen hyödyllisiä asioita tulevia tehtäviä varten. Käytin myös tiimistä löytyvää osaamista tarpeen mukaan selvityksissä, mutta minulle on ominaista yrittää ensin itsenäisesti ratkaista asioita.

Useamman kerran keskellä pyrähdystä jouduttiin käymään suunnittelupalavereja, joissa täsmennettiin tehtävän kuvausta ja mietittiin tarkemmin toteutusta. Viimeisessä pyrähdyksessä itse asiassa jatkokehitettiin aikaisempien viikkojen tehtäviä sen verran, että osa niiden tuotoksista tehtiin seurantajakson viimeisen pyrähdyksen aikana uudelleen eri tavalla. Reseptilomakkeen teknisen uudistuksen suunnittelussa ei onnistuttu kovin hyvin noudattamaan myöskään Scrum-mallin periaatteita seurantajakson aikana. Niiden mukaan suunnittelussa tehtävät tulisi olla määritelty sen verran tarkasti, että niitä varten ei tarvitse suunnitella tarkemmin pyrähdyksen aikana. Nämä ongelmat olivat kyllä tiedossa, mutta niihin ei ehditty seurantajakson aikana puuttumaan.

Työtehtävien toteutuksissa sovellettiin jo olemassa olevia osaamisalueita. Pääosin niissä ei tullut teknisesti mitään erityisen uutta, mitä ei onnistunut selvittämään internet-haulla tai ohjelmistokehyksen dokumentointia lukemalla. Ongelmanratkaisukyvyssä ja virheenetsintätyökalun käytössä sen sijaan tuli enemmän kehitystä.

Seurantajakson aikaiset työtehtävät olivat aikaisempiin viikkoihin verraten hieman erilaisia. Tämä on kuitenkin tietyllä tavalla tyypillistä. Laajan potilastietojärjestelmän samanaikainen kehittäminen ja ylläpitäminen asiakkaille vaatii mm. uusien ominaisuuksien kehittämistä, virheiden korjaamista, tuotannon ongelmien selvittelyä ja kehitysympäristön toiminnan parantamista. Työn seurantajaksolla korostui, että yksi haastavin asia potilastietojärjestelmän kehittämisessä on tehtävien suunnittelu.

LÄHTEET

1. Esko Systems Oy 2022. Yritys. Hakupäivä 1.4.2022 https://eskosystems.fi/yritys/
2. Esko Systems Oy 2022. Esko Systems Oy Organisaatio 21.2.2022 alkaen. Yrityksen sisäinen dokumentti.
3. Esko Systems Oy 2022. Esko Systems tuotekehitystiimit 01-2022. Yrityksen sisäinen dokumentti.
4. Esko Systems Oy 2022. Tuoteperhe. Hakupäivä 3.4.2022. https://eskosystems.fi/tuoteperhe/
5. Kela 2021. Kansallinen lääkityslista etenee – rakenteisen annostuksen käyttöönotto 2022. Hakupäivä 3.4.2022. https://www.kela.fi/-/kansallinen-laakityslista-etenee-rakenteisen-annostuksen-kayttoonotto-2022
6. Kanta 2021. S1 Kirjaa ja muodosta rakenteinen annostusohje. THL ja Kela. Hakupäivä 2.5.2022. https://www.kanta.fi/documents/20143/1465348/S1+Kirjaa+ja+muodosta+rakenteinen+annostusohjeV3.00.pdf/03bfee0e-5f82-a512-20e1-ed64b8c65389?t=1620802636004
7. Kanta 2022. Reseptikeskukseen tallennettavien asiakirjojen määrittelyjen versiokäytännöt. Kela. Hakupäivä 3.4.2022. https://www.kanta.fi/documents/20143/796372/Reseptikeskukseen+tallennettavien+asiakirjojen+m%C3%A4%C3%A4rittelyiden+versiointik%C3%A4yt%C3%A4nn%C3%B6t+1.6.pdf/9e2d537b-e402-bcd0-ddae-eb0f2eddb853?t=1646308234109
8. Bootstraps team 2022. Alerts. Hakupäivä 6.4.2022. https://getbootstrap.com/docs/4.0/components/alerts/
9. Material UI SAS 2022. Alert. Hakupäivä 7.4.2022. https://mui.com/components/alert/
10. Kanta 2021. Sähköinen lääkemääräys vaatimusmäärittely. Kela. Hakupäivä 6.4.2022. https://www.kanta.fi/documents/20143/1848363/Resepti-palvelu_Maarittely_Vaatimukset_Potilastietoj%C3%A4rjestelm%C3%A4_v2.95.pdf/b7389553-3c30-753c-ed2c-7765513efd0b?t=1636105018586
11. Figma 2022. Creative tools meet the internet. Hakupäivä 6.4.2022. https://www.figma.com/about/
12. Material UI SAS 2022. Progress. https://mui.com/components/progress/
13. Jasonsaayman. Axios. Hakupäivä 7.4.2022. https://github.com/axios/axios
14. Kela 2022. Kansallinen koodistopalvelu. Hakupäivä 06.04.2022. https://koodistopalvelu.kanta.fi

15. Newtonsoft 2022. Json.NET. Hakupäivä 6.4.2022. https://www.newtonsoft.com/json
16. Mozilla Corporation 2022. Base64. Hakupäivä 6.4.2022. https://developer.mozilla.org/en-US/docs/Glossary/Base64