Opinnäytetyö (AMK)

Tietojenkäsittely

2020

Joel Perasto

WEB-KEHITTÄJÄN PÄIVÄKIRJA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely

2020 | 45 sivua

Joel Perasto

WEB KEHITTÄJÄN PÄIVÄKIRJA

Opinnäytetyö on suoritettu päiväkirjamuodossa. Päivittäisessä raportoinnissa seurataan aloittelevan web-kehittäjän työskentelyä. Opinnäytetyö koostuu kahden web-sovelluksen kehittämisestä, kehityksen työtehtävien päiväkohtaisesta raportoinnista, viikoittaisista edistyksen yhteenvedosta ja pohdinnasta.

Päiväkohtaisessa raportoinnissa opinnäytetyön kirjoittaja dokumentoi päivittäisiä käyttöliittymien kehityksen työtehtäviä, kuten ohjelmointia, ongelmanratkaisua, tiedonhakua ja toteutuksen suunnittelua. Käyttöliittymien kehitykseen ratkaisuja haettiin käytettävän teknologian virallisesta dokumentaatiosta, ohjelmointiyhteisön artikkeleista ja foorumeilta.

Opinnäytetyön kirjoittajan tekninen osaaminen ja web-sovellusten kehityskaaren ymmärtäminen karttui ajanjakson aikana paljon. Projektiosaaminen kehittyi ketterän kehityksen parissa huomattavasti ja oman työajanhallinta parani raportointijakson aikana. Nämä taidot ovat keskeisiä työelämään siirryttäessä ja arvostettuja alan työpaikoilla.

ASIASANAT:

web-kehitys, päiväkirja, react, spring boot, node.js

BACHELOR´S / MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor of Applied Sciences – BASc, Information Technology

2020 | 45 pages

Joel Perasto

A WEB DEVELOPER`S DIARY

This thesis has been done in a form of diary. Work of a novice web developer is followed on daily reports. The thesis consists of the development of two web-applications, work assignments, daily reports, weekly summaries and reflection chapter.

In the daily reports, the author of the thesis documents daily web development work tasks, such as programming, problem solving, information retrieval, and implementation planning. Solutions for development were sought from official documentation of the technology, articles from the programming community and programming forums.

The technical competence of the thesis author and the understanding of the development process of web applications accumulated a lot during the period. Project expertise developed significantly in terms of agile development, and the management of one's own working time improved during the reporting period. These skills are central to the transition to working life and valued in jobs in the industry.

KEYWORDS:

web development, diary, react, spring boot, node.js

SISÄLTÖ

KUVAT

SANASTO

Bitbucket	lähdekoodin ja versionhallintaan tarkoitettu palvelusivusto (Bitcbucket)
CORS	selaintekniikka, jolla pyritään turvallisesti sallimaan Javascript-ohjelmien tekemät palvelupyynnöt eri domainien välillä (CORS)
CSS	tekniikka web-sivustojen fonttien,värien ja muotojen lisäämiseksi, Cascading Style Sheets (CSS)
Fetch	JavaScript-toiminto, joka tarjoaa helpon ja loogisen tavan hakea resursseja asynkronisesti verkon kautta (Fetch)
Git	hajautettu versionhallintajärjestelmä (Git)
GitFlow	Git-versiohallinnan haaroitusmalli (GitFlow)
Header	tiedonsiirto HTTP-protokollan otsikkotiedot (Header)
Hook	React-kirjaston toiminto, joka antaa mahdollisuuden manipuloida funktionaalisen komponentin tilaa ja elinkaarta (Hook)
HTTP	tiedonsiirtoprotokolla selainten ja palvelimien välillä (HTTP)
keskuskone	IBM:n valmistama tietokonejärjestelmä, jota käytetään pääasiassa asiakkaiden tilaus-, rahoitustapahtumiin ja tuotannon ja varaston hallintaan (Keskuskone)
käyttötapaus	kuvaus järjestelmän tai käyttöliittymän toiminnoista (Käyttötapaus)
MVP	ienin julkaisukelpoinen tuote, Minimum Viable Product (MVP)
Node.js	avoimen lähdekoodin alustariippumaton JavaScript runtime-ympäristö JavaScript-koodin suorittamiseen palvelimella (Node)
Openshift	avoimen lähdekoodin kehitysalusta, jonka avulla voidaan kehittää ja ottaa käyttöön sovelluksia pilvi-infrastruktuurissa (Openshift)
PoC	konseptitodistus on tietyn menetelmän tai idean osoittaminen toteuttamiskelpoiseksi, Proof of Concept (PoC)
React	Javascript-kehys käyttöliittymien rakentamiseen
REST	HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen
retrospektiivi	ketterässä kehityksessä käytettävä palaveri projektin arvioimiseen ja kehittämiskohteiden paikantamiseen (Retro)

Spring Boot	avoimen lähdekoodin Java-pohjainen kehys, mikropalveluiden luomiseen (Spring)
Thymeleaf	palvelinpuolen Java-mallimoottori (Thymeleaf)
Token	pääsyavain verkkojen välillä (Token)

1 JOHDANTO

Tämä opinnäytetyö kertoo web-kehittäjän päivittäisestä työstä 10 viikon ajalta. Viimeisessä luvussa on pohdinta, jossa päiväkirjan kirjoittanut web-kehittäjä käy läpi ajatuksiaan tältä 10 viikon ajanjaksolta. Päiväkirja on kirjoitettu 13.1.−20.3.2020.

Toimeksiantajana toimii Kansaneläkelaitoksen (Kela) IT-palvelut-yksikkö. Web-kehittäjä toimii käyttöliittymien toteuttajana etuusalustan kehitystiimissä. Etuus-alustan kehitystiimissä suunnitellaan ja kehitetään nykyisten ratkaisujen tilalle moderneilla teknologioilla toteutettua ratkaisua etuuksien käsittelyjärjestelmiin. Tässä projektissa luodaan ratkaisua yleisen asumistuen käsittelyyn ja tätä ratkaisua käytetään pohjana muiden Kelan etuuksien käsittelyjärjestelmien uudistamiseen.

Työtehtävissä tarvitaan ymmärrystä web-sovellusten käyttöliittymien kehittämisestä Javascript-kirjastolla React.js (React)- ja Node.js (Node) -palvelinympäristöllä sekä Java-pohjaisella Spring Boot (Spring)- ja Thymeleaf (Thymeleaf) -yhdistelmällä. Näiden lisäksi vaaditaan REST-rajapintojen (Representational State Transfer, REST) ja palvelinpuolen tuntemusta.

Web-kehittäjän pääasiallinen tavoite ajanjaksolle on kehittää teknistä osaamista tasolle, jolla itsenäinen kehitystyö ja toimivien sekä oikeaoppisten web-kehitys ratkaisujen tuottaminen onnistuu. Ajanjakson muita tavoitteita on kehittää projektiosaamista ja oman työajanhallintaa. Projektiosaaminen on ketterän kehityksen sisäistämistä, tiimissä työskentelyä ja kehitettävien kohteiden kehitysaikajanan tuntemusta. Oman työajanhallinta on myös projektiosaamista, kuten työtehtävistä suoriutumisen keston arviointia, joka on oleellista ketterässä kehityksessä, kun suunnitellaan käytettävien työtuntien määriä projektin eri vaiheisiin.

2 PROJEKTI

2.1 Työtehtävät

Työskentelen kehitystiimissä, joka suunnittelee ja kehittää teknologiaratkaisua korvaamaan käytössä olevia vanhoja teknologioita nykyaikaisilla teknologiaratkaisuilla. Työtehtäviäni projektissa on kehittää käyttöliittymien osia yleisen asumistuen käsittelyjärjestelmään. Työskentelen nimikkeellä käyttöliittymätoteuttaja.

Käyttöliittymiä kehitetään projektin alkuvaiheessa kahdella eri ohjelmointiympäristöllä. Toinen on minulle tutumpi React ja Node.js-kehitysympäristö ja toinen Java-pohjainen Spring Boot ja Thymeleaf-kehitysympäristö. Kahden eri teknologian käyttämisen tarkoituksena on havainnollistaa ja testata teknologioiden omaksuttavuus ja yhteentoimivuus projektin muiden teknologioiden kanssa. Projektin käyttöliittymien eri osat rakennetaan MVP-laajuuteen, jolloin käyttöliittymistä jätetään pois harvinaisten tapausten käsittelytoiminnot. Työtehtävilleni oleellista on myös ymmärtää REST-rajapintojen toiminta ja käyttäjän autentikoinnin ja auktorisoinnin toimintaperiaatteet.

2.2 Omat tavoitteet

Pääasiallinen tavoitteeni ajanjaksolla ja uran alkuvaiheessa on kehittää teknistä osaamista ja kehitystyökalujen tehokasta käyttöä. Koen, että minulla on hyvä käsitys sovelluskehityksen perusperiaatteista ja oikeista käytännöistä. Kokemusta puuttuu monimutkaisemmista projekteista ja osaamista tuotantotasoisen koodin tuottamiseen itsenäisesti.

2.3 Vuorovaikutus työpaikalla

Käyttöliittymätoteuttajan rooli on suuremmaksi osaksi itsenäistä työskentelyä määriteltyjen käyttötapausten (Käyttötapaus) perusteella. Vuorovaikutusta muiden tiimin jäsenien kanssa kuitenkin tapahtuu paljon selvittäessä ongelmia, ratkoessa virheitä tai kokoontuessa päivittäisiin palavereihin, joissa tiimille kerrotaan oman työskentelyn tilanne. Erityisesti tiiminjäsenien kommunikointi on tärkeää tämän

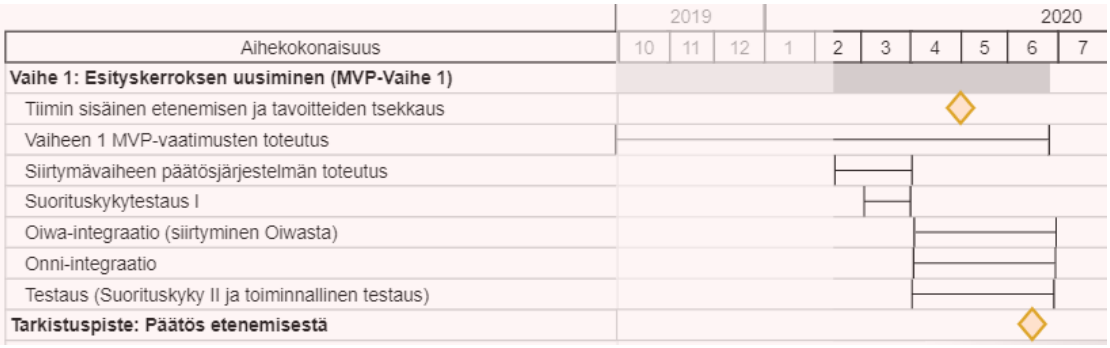
kaltaisessa projektissa, jossa osaaminen ja tieto teknologioista on hajautunutta ja käytettävien teknologioiden määrä suuri.

Tässä projektissa erityistä on, että projektin sovelluksien arkkitehtuurillista rakennetta ei ole vielä täysin määritelty. Tästä syystä myös vuorovaikutus sovellusarkkitehdin ja vanhemman käyttöliittymätoteuttajan kanssa on tärkeää arkkitehtuurillisten muutoksien tapahtuessa.

Aloitin työni samaan aikaan projektissa toisen käyttöliittymätoteuttajan kanssa. Työtehtävät ovat molemmilla projektissa samankaltaiset, jolloin samoja ongelmia tai ratkaisuja kohdatessa niitä on mahdollista työstää yhdessä.

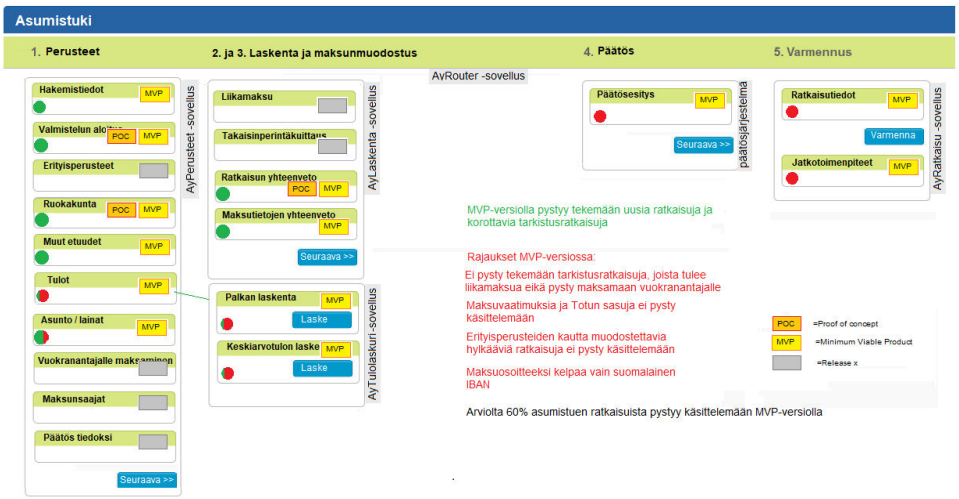
2.4 Projektin kokonaiskuva

Etuusalustan kehitystiimissä suunnitellaan ja toteutetaan nykyaikaisilla teknologioilla MVP-tason toteutusta yleisen asumistuen etuuskäsittelyn käsittelyjärjestelmästä. Projektin tarkoitus on myös luoda ratkaisupohjaa Kelan muiden etuuksien sisäisten käsittelyjärjestelmien teknologioiden uusimiseen. Projektia edistetään vaiheittain ja olen mukana projektin ensimmäisessä vaiheessa toteuttamassa MVP-tason sovelluksia. (Kuva 1.)



Kuva 1. Projektin ensimmäinen vaihe.

Asumistukijärjestelmä koostuu monista paneeleista ja viidestä eri sovelluksesta, joissa paneeleja esitetään. Teen käyttöliittymän Yhteenveto-sovelluksesta React.js ja Node.js tekniikoiden yhdistelmällä ja Perusteet-sovelluksesta Muut Etuudet -paneelia Spring Boot ja Thymeleaf yhdistelmällä. Muut etuudet -toteutuksessa lisään olemassa olevaan sovellukseen paneelin, kun taas Yhteenveto-sovelluksen toteutuksessa kehitin ratkaisua alusta alkaen itsenäisesti. (Kuva 2.)



Kuva 2. Asumistukijärjestelmän tavoitetila.

Projektissa ei rakenneta pelkästään käyttöliittymiä uudella teknologialla. Uudistusprojektin laajuus kattaa myös REST-rajapintojen muodostuksen, liiketoiminnan keskuskonetoteutukset, konttiajoalustan, tunnistus- ja tietoturvaratkaisut, tukijärjestelmät, eräajojen toteutustavat ja mikropalveluarkkitehtuurin. Näiden ratkaisujen kehitystä työstävät tiimin muut jäsenet ja niillä on vaikutusta käyttöliittymien toimintaan. Näiden teknologioiden osaamista ei kuitenkaan minulta käyttöliittymätoteuttajana edellytetä, mutta niiden toiminnan ymmärtäminen on tarpeellista.

2.5 Laskenta-sovellus

Laskenta-sovelluksen on tarkoitus näyttää käyttäjälle käsiteltävän asiakkaan yleisen asumistuen perustetietojen pohjalta lasketut etuusjaksot (Kuva 3.), laskennassa käytetyt perustetiedot ja maksutietojen yhteenveto (Kuva 4.). Laskenta-sovellus ajetaan käsittelyprosessissa Perusteet-sovelluksen jälkeen, joissa esitettävät tiedot määritetään tai niitä muokataan.

Näytettävät tiedot haetaan REST-rajapinnasta, joka on yhteydessä nykyisiin etuuskäsittelyn keskuskonetekniikalla suoritettaviin toimintoihin. PL/1-kielellä toteutetusta keskuskonetekniikasta pois siirtyminen on projektin yksi päämäärä, mutta vaiheessa yksi ne ovat vielä osana toteutuksia.

Web-sovelluksen sivut rakennetaan käyttötapausten perusteella.

Kuva 3. Esimerkki käyttötapauksessa määritellystä Ratkaisun yhteenveto-paneelista.

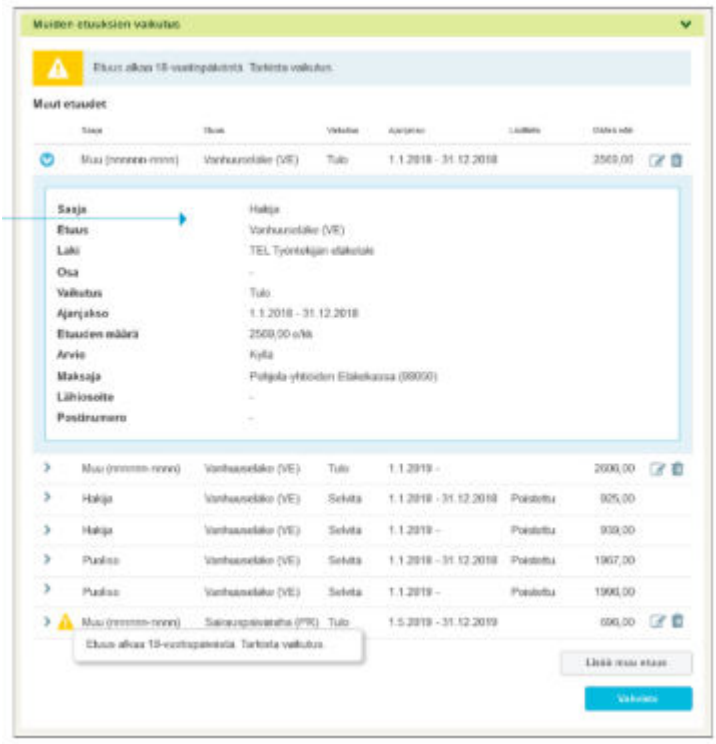
Kuva 4. Esimerkki käyttötapauksessa määritellystä Maksutietojen yhteenveto -paneelista.

2.6 Muut etuudet-paneeli

Perusteet-sovelluksen muut etuudet -paneelissa esitetään ja käsitellään asumistukihakemukseen vaikuttavia muita etuuksia ja korvauksia, joita käsiteltävällä henkilöllä mahdollisesti on (Kuva 5.). Käsiteltävän tapauksen etuuksia voi muokata,

poistaa ja tapaukselle voi lisätä uusia etuuksia. Tapauksen etuudet tarkistetaan ja hyväksytään.

Web-sovelluksen sivut rakennetaan käyttötapausten perusteella.



Kuva 5. Osa käyttötapauksessa määriteltyä ulkoasua.

3 RAPORTOINTI

3.1 Viikko 1 – Kehitystyökalut

Maanantai 13.1.2020

Aluksi kehitysympäristöjen pystyttämistä hankaloitti Kelan vahvat tietosuoja ja tietoturva-toimenpiteet työpisteissä. Esimerkiksi vain valmiiksi määritettyjä sovelluksia voi ladata ja käyttää ja niihin tarvittiin käyttöluvat sisäisessä verkossa. Asennusta kuitenkin lopulta edisti sisäiset kehitysympäristön asennusohjeet ja kysymällä apua sain kehitysympäristön toimimaan.

Tiistai 14.1.2020

Aamupäivä kului rakennellessa ajatusta sovelluksen kulusta. Otin mallia kansiorakenteeseen omista edellisistä React-projekteista ja projektissa aloitetusta toisesta käyttöliittymästä Perusteet-sovellus.

Kun sovelluksen yleinen tyyli on jo CSS-tyylitiedostoissa (CSS) määritelty ja käytin niitä toteuttaakseni Laskenta-sovelluksen näkymän karkeasti ilman toimintoja.

Keskiviikko 15.1.2020

Tänään tutustuin Reactin dokumentaatiosta Hook-ominaisuuksiin (Hook) ja funktionaaliseen ohjelmointiin käsitteenä. Yhdessä Perusteet-sovelluksen esittelyssä vanhempi käyttöliittymätoteuttaja huomautti, että trendinä ja suuntana web-kehityksessä on enenemissä määrin funktionaalinen ohjelmointi.

Tämän lisäksi suunnittelin tietojenkulun toteutusta sovelluksen komponenttien välillä. Iltapäivällä sain haltuuni REST-rajapinnasta palautuvan tietojen muodon. Tämän avulla pystyin rakentamaan yksinkertaisen mockdatan Laskelman taulukon tiedoista. Mockdata on hyödyllinen työkalu tiedonkulun testaamiseen sovellustasolla.

Torstai 16.1.2020

Päivä oli tänään hyvin ohjelmointipainotteinen. Rakensin käyttötapauksessa määritettyjen taulukkojen ulkoasut sovellukseen. Taulukoiden lisäys työvaiheena ei ollut teknisesti haastava mutta työläs, koska taulukkoja on paljon. Sain kuitenkin kaikista paneelin taulukoista perusrakenteen tehtyä. Ajatuksena palata taulukkokohtaisiin yksityiskohtiin myöhemmässä vaiheessa kehitystä. Tässä vaiheessa vielä arvuuttelun varaan jäi, onko kansiorakenteeni oikeanlainen. En ollut saanut teknistä ohjausta vanhemman käyttöliittymätoteuttajan ollessa kiireinen.

Aamupäivällä oli myös arkkitehtuuripalaveri, jossa käsiteltiin Kelan käyttämää pääsynhallintajärjestelmää.

Perjantai 17.1.2020

Tänään ohjelmassa oli koko päivän mittainen versiohallintakoulutus. Koulutuksessa käytiin läpi Git-versiohallinnan (Git) perusteita, työnkulkua ja Bitbucket (Bitbucket) -nimistä graafista työkalua. Gitin perusperiaatteet oli jo ennestään tuttuja, mutta isommassa projektiympäristössä versiohallinnan kanssa työskentelyyn on hyvä saada tarkempia ohjeita. GitFlow ja oikeaoppiset haaroituskäytännöt olivat hyviä keskustelun aiheita koulutuksessa.

Lopuksi kartoitin ensi viikon tekemiselle valmiiksi suunnan. Tarkoituksena oli perehtyä lisää Reactin props- ja Hook-ominaisuuksiin.

Viikko 1 yhteenveto

Toinen viikko alkoi ensimmäisen viikon informaatiotulvan sulattelulla ja kehitysympäristön asentamisella. Kehitysympäristöä asettaessa valmiiksi Kelan vahvat tietosuoja ja tietoturvatoimet eivät jääneet huomaamatta. Työtä aiheutti käyttöoikeuksien rajoitukset ja valmiiksi määritellyt asennustoimenpiteet. Työpisteelle pystyin asentamaan vain valmiiksi sisäisessä verkossa olevia sovelluksia, ja niiden käyttämiseen tarvittiin erilliset käyttöluvat.

Viikon edetessä pääsin tutustumaan lisää ensimmäiseen työtehtävään. Tehtävässä oli määritelty toteuttaa Reactilla asumistuen käsittelyprosessin Laskenta-sovellus. Tehtävään sain tueksi käyttötapauksen ja toisen Reactilla toteutetun sovelluksen malliksi. Näiden avulla pääsinkin hyvin nopeasti aloittamaan ja lisäksi Reactilla projektin aloittaminen on myös erittäin helppoa. Aloitin kansiorakenteen ja teknisen toteutuksen suunnittelulla, johon sain kokemattomana kulumaan paljon aikaa. Kun toteutuksen idea oli valmiina, loppuviikko oli hyvin suoraviivaista ulkoasun paikalleen asettelua käyttötapauksessa määritellyn kuvan perusteella.

Sovelluksen sain rakennettua ulkokuorelta vaadittuun muotoon ja taulukoiden rakenteen oikeaksi. Kaikki taulukoissa näkyvä tieto on vielä kovakoodattua eli staattista, ei käyttäjän tai järjestelmän muokattavissa olevaa. Sovelluksen toiminnallisuudet ovat ensi viikon aiheena.

Huomioita omaan työskentelyyn liittyen tuli yllättävän paljon jo ensimmäisellä viikolla. Muistiinpanoja pitäisi tehdä enemmän ja parempi laatuisia. Niistä tulee helposti kryptisiä nopeasti sutaistuna. Ajattelutyötä pitäisi muistaa tauottaa ja aivoja virkistää välillä taukojumpilla, muuten seinä tulee vastaan vähän puolenpäivän jälkeen. Tehokkaaksi ja toimivaksi koodaamistyyliksi on todettu iteraatiot, jota pyrin harjoittelemaan. Käytännössä tällä tarkoitetaan asteittaista koodin kehittämistä ”raameista” valmiiseen sovellukseen eikä kirjoittamaan täydellistä sovellusta ensimmäisellä kerralla.

3.2 Viikko 2 – Suunnittelu

Maanantai 20.1.2020

Aloitin päivän suunnittelemalla toiminallisuuksia Laskenta-sovelluksen Laskelma-paneelien taulukoihin. Taulukoiden yksinkertaisen toteutuksen sain valmiiksi viime viikolla. Tämä vaati hieman syvempää tutustumista Reactin props- ja Hook-ominaisuuksiin. Aamupäivä kului hyvin pitkälti taulukoiden ”kovakoodattujen” taulukon tietojen korvaamisella muuttujilla. Iltapäivällä sain taulukot täytettyä dynaamisella mockdatalla jotka vastaavat REST-rajapinnasta haettavaa tietoa. Seuraavana tehtävänä olisi saada kutsuttua oikeaa rajapintaa.

Aiemmissa tekemissäni React-sovelluksissa olen käyttänyt Class-rakennetta ja nyt käytössä on uudet Hook-ominaisuudet (React 2020b). Vanhemman käyttöliittymätoteuttajakollegan mukaan Hookeissa käytettävä funktionaalinen ohjelmointi on suosiotaan kasvattava ohjelmointityyli. Funktionaalinen rakenne vähentää koodin ylimääräisiä toistoja, parantaa modulaarisuutta ja parantaa koodin huollettavuutta (Palakollu 2019).

Tiistai 21.1.2020

Aloitin päivän paneutumalla oman sovelluksen ja rajapinnan väliseen dynamiikkaan. Aamupäivä kului sovelluksen REST-rajapintojen kutsuja koestaessa sovelluksesta. (Kuva 6.) Yritykseni osoittautuivat turhiksi. Taustapalveluiden arkkitehtuuri oli rakennettu siten, että Cross-Origin Resource Sharing (CORS) -ongelmia rupesi ilmenemään kutsuja tehdessä. Kokeneemman kehittäjän vinkeistä löytyi ratkaisu. Minun täytyy tehdä palvelin, jolla voin lisätä http-palvelinpyynnön headereihin (Header) tarvittavan tunnistautumis-tokenin (Token). Tämän toteutusta pääsin jatkamaan seuraavana päivänä.

```
async function fetchData() {
  setLoading(true)
  const res = await fetch(url)
  res
  .json()
  .then(response => {
    setData(response.payload);
    setLoading(false);
  })
  .catch(error => setError(error));
}
useEffect(() => {
  fetchData();
}, []);

return (
  <div id="container">
    {loading === true ?
    <Loader></Loader>
    :
    <Paneeli response={data}></Paneeli>
    }
  </div>
  )
```

Kuva 6. Sovelluksesta lähtevän kutsun ensimmäinen versio.

Keskiviikko 22.1.2020

Aamulla aloitin tutustumisen Laskelma-paneelin REST-rajapintojen yhteyksiin ja pyrin kartoittamaan tarkemmin sovellukseen vaadittavat toiminnot. Tämänhetkisten tietojeni mukaan täytyi rakentaa pieni paikallispalvelin. Tämän palvelimen avulla luodaan turvallinen yhteys REST-rajapintaan. Tein Node.js -palvelinympäristöllä esimerkin mukaisesti perustoiminnollisen palvelimen, johon otin Perusteet-sovelluksen toteutuksesta mallin. Sovelluksesta palvelinta kutsuttaessa palautti palvelin Laskelma REST-rajapinnan mukaista mockdataa. Mockdata mukaili REST-rajapinnan palauttamaa tietoa, jolloin pystyin testaamaan palvelimen ja sovelluksen välisiä toimintoja ilman, että pitää kutsua itse REST-rajapintaa. Yksinkertaisen palvelimen kasaaminen oli suoraviivainen ja ”helppo” työvaihe.

Sovelluksen puolella harmaita hiuksia aiheutti kutsu omaan lokaaliin palvelimeen. Uuden funktionaalisen ohjelmointisyntaksin käyttö palvelimelle tehdyssä fetch-kutsussa (Fetch) osoittautui haastavaksi. Täytyi tutkia monia eri esimerkkejä, ennen kuin löytyi omaan käyttöön sopiva ei-synkroninen funktio. Tässä tilanteessa Reactin useEffect-ominaisuus on hyödyllinen sillä se ajetaan vasta kun kaikki muut sovelluksen komponentit ja osat ovat latautuneet täysin. Tämä osoittautui erittäin käteväksi ei-synkronisia kutsuja soveltaessa. UseEffectin käyttö estää kutsusta palautuvien tietojen ajamista sovelluksen latautuessa vielä rakentuviin komponentteihin ennekö ne ovat valmiit (Hooks Effect 2020).

Torstai 23.1.2020

Aamupäivän rakensin yksinkertaista ratkaisua kutsuun sovelluksesta rajapintaan asti. Koko iltapäivä kului Laskelman tietojen fetch-kutsun, ja sen käsittelyn ja yhteensovittamisen jo käytössä olevan koodin ja kutsun kanssa. Tämä työ jäi kesken mutta suunnittelu ja ajatus sopivasta toteutuksesta eteni hyvin paljon.

Perjantai 24.1.2020

Tänään tein sovelluksen fetch-kutsuja ohjaavan välityspalvelimen refaktorointia. Nodella rakennetulla välityspalvelimella on refaktoroinnin jälkeen sovelluksen GET- ja

POST-kutsuille omat kutsuja ohjaavat toteutukset. Tämä muutos toteutettiin, kun sovellukseen lisättiin POST-kutsu, jolla aloitetaan laskenta. Uuden kutsun myötä jouduin refaktoroimaan ei-synkroniset kutsut toimimaan yhdessä synkronisesti, jotta laskenta ehtii suoriutua ennen tuloksien hakua.

Viikko 2 yhteenveto

Tällä viikolla olen päässyt työskentelemään paljon sovelluksen palvelimen parissa eli backendissä, joka ei itselleni ole niin tuttu ympäristö. Viikko on sen myötä sisältänyt paljon suunnittelua ja uuden opettelua. React 16.8 -versiossa tulleet Hook-ominaisuudet useState ja useEffect ovat vaatineet paljon harjoittelua. Eniten aikaa kului kuitenkin sovelluksen ja rajapinnan välisen yhteyden toteutuksen suunnitteluun ja tekemiseen.

Ensimmäinen yritys hakea tietoa rajapinnasta päätyi CORS-ongelmiin. Yrityksessäni kutsua rajapintaa tein sen selaimelta suoraan. Tämä ei kuitenkaan ollut oikea menettelytapa vaan sain ohjeena tehdä välityspalvelimen, jonka kautta rajapintaa voi kutsua turvallisesti. Tähän teknologiana käytetään Node.js-alustaa, jonka käyttö esimerkkiä seuraten onnistui mutkattomasti ja yhteys palvelimelta rajapintaan oli nopeasti valmiina. Vaikein osuus oli saada sovelluksen ja oman palvelimen välinen yhteys toimimaan tunnistuksella, jolloin tunnistukseen vaadittavat tokenit pitää saada liikkumaan oikein. Lopulta päätin jättää tunnistautumisen tässä sovelluksen kehitysvaiheessa vielä sovelluksesta pois, kun testirajapintapalvelin toimi myös ilman tunnistusta. Tähän käytetään Kelan pääsynhallintajärjestelmää, joka on aivan uusi myös monelle muulle kehitystiimissä.

Loppuviikolla myös huomasin myös työn ulkoisilla asioilla olevan suuri vaikutus työtehokkuuteen, kun niskakipujen kiusaamana torstai- ja perjantaipäivä olivat erittäin haastavia keskittymisen kannalta. Tähän uskoisin olevan hyötyä Break Pro -työpistetaukoliikuntaohjelmasta, jonka ajattelin ottavani käyttöön ensi viikolla.

3.3 Viikko 3 – Palvelinpuoli

Maanantai 27.1.2020

Tämä viikko alkoi uuden sprintin suunnittelulla ja edellisen sprintin retrospektiivillä (Retro). Retrossa oli paljon asiaa arkkitehtuurillisista kysymyksistä ja projektihallinnallisista käytännöistä. Tekemiseeni suoraan vaikuttavia päätöksiä oli tässä vaiheessa vielä vähän.

Ennen palaveria ehdin palautellaa mieleen viikonlopun jälkeen toteutuksen tilaa ja kartoittaa puuttuvia ominaisuuksia ja etsiä tietoa projektissa käytettävistä teknologioista, kuten OpenShift-ajoalustasta (Openshift), SOA-palveluista ja Kelan käyttämästä pääsynhallintajärjestelmästä. Ensimmäisien viikkojen aikana teknologioita mainittiin niin monia, ja nyt päädyin kirjoittamaan niitä muistiin. Tänään oli sopiva hetki käydä kirjoittamaani muistilistaa läpi ja tutustua teknologioihin paremmin.

Iltapäivän käytin Laskenta-sovelluksen Spring Boot -toteutuksen tutkimiseen ja hakien esimerkkiä taulukkojen tietojen käsittelyyn. Tämän lisäksi muokkasin CSS-tyylien tiedostosijaintia keskitetymmäksi. Keskitetyt CSS-tyylitiedostot on todettu paremmaksi tavaksi tallentaa tyylejä. Muokattavat tyylit löytyvät yhdestä sijainnista, jolloin niiden päivittäminen tapahtuu myös yhteen tiedostoon.

Tiistai 28.1.2020

Aamulla kokeilin saada Kelan käyttämän pääsynhallintajärjestelmän tunnistuksen lisäämistä Laskenta-sovellukseen siinä kuitenkaan onnistumatta. Ongelmia ilmeni välityspalvelimen konfiguroinnissa. Palvelinpuolen työvaiheet vaativat vielä opettelua ja pureskelua.

Toinen aamun askare oli väliarviointi esimiehen kanssa. Väliarvioinnissa käsiteltiin perehdytyksen vaiheita ja harjoittelun tilannetta. Palaute oli positiivista ja iltapäivällä pidetyssä Laskenta-sovelluksen demotilaisuudessa palaute oli hyvää.

Päivä oli hyvin katkonainen eikä sovellus edennyt muuta kuin suunnittelun tasolla. Uusia ratkaisuja olemassa oleviin ongelmiin ei ehtinyt kuin hieman kokeilemaan.

Ensimmäisessä sprintissä olen listaillut sovelluksesta puuttuvia ilmeisiä osia listaan, joita käsittelen tästä eteenpäin. Puuttuvista ominaisuuksista olivat isoimpia virhekäsittely, tyylien hiominen, taulukoiden tietojen tarkka validointi ja inputtien muotoilu.

Keskiviikko 29.1.2020

Aamu kului taulukoiden tietojen oikeanmuotoisessa formaatissa esittämiseen ja sen toteuttamisen suunnitteluun. Tässä kompuroin liian monimutkaisen ratkaisun yrittämiseen ja opastusta kysyttyäni sain ohjeen oikeaan ratkaisuun. Oikea ratkaisu tällä kertaa oli tehdä yleisesti käytettävä funktio taulukoiden päivämäärien ja rahamäärien muotoiluun. Iltapäivällä tein taulukon tyylien hiomisia, kuten tekstien samaan reunaan tasaamisia ja taulukon sivun koon mukaan dynaamisesti muuttuvat solukot.

Torstai 30.1.2020

Sovelluksen kehittäminen jatkui tyylien hiomisella ja yrityksiin saada välityspalvelimelta yhteys sovellukseen. Tarvittaessa olen kysellyt tai minua on informoitu outputtien tarkemmista muodoista ja määrityksistä. Euromäärille tein alustavan yleisesti käytettävän CurrencyOutput-komponentin, joka palauttaa syötetyn numeron 2 desimaalin tarkkuudella takaisin. (Kuva 7.) Funktion tekemiseen sain ohjeistuksen sovellusarkkitehdiltä React-toteutusten palaverissa ja esimerkin kollegan DayOutput-komponentista, jolla päivämäärien muotoa muokataan.

```
export default function CurrencyOutput(props) {

    if (props === undefined || props === null || isNaN(props) === true) {
        return(<span>-</span>);
    }

    let decimalNumber = props.toFixed(2).replace(".", ",");

    return (
        <span>{decimalNumber}</span>
    )
}
```

Kuva 7. CurrencyOutput-komponentin ensimmäinen versio.

Perjantai 31.1.2020

Rakensin mockdataa taulukon toiminallisuuksien testaamiseen. Vanhemman käyttöliittymätoteuttajan kanssa käsiteltiin Kelan käyttämän pääsynhallintajärjestelmän palvelun lisääminen toteutukseen ja saimme sen toimimaan Laskenta-sovelluksessa.

Viikko 3 yhteenveto

Tällä viikolla alkoi uusi Sprint ja sen suunnittelulla ja edellisen retrospektiivillä. Sprintin suunnittelussa käytiin läpi paljon arkkitehtuurillisia kysymyksiä ja tehtävien priorisointia. Omaan tekemiseen ei tullut muutoksia vielä tässä vaiheessa vaan jatkoin Laskenta-sovelluksen toteutuksen työstämistä. Retrospektiivissä palaute oli hyvää projektiin mukaan pääsystä ja itse tekemisestä.

Viikko on pitänyt sisällään paljon selvitystyötä REST-rajapinta kutsujen, välityspalvelimen, tunnistukseen ja käyttöoikeuksien määritykseen käytettävän Keycloak-ohjelmiston käyttöön. Sovelluksen on tarkoitus kutsua välityspalvelinta, palvelin lisää headeriin pääsynhallintajärjestelmän vaatiman tokenin kutsuun, joka määrittää käyttäjän rooli. Näin pyritään minimoimaan selain päässä käsiteltävän tiedon määrä.

Tämän tehtävän toteutukseen sain esimerkkiä valmiista toteutuksesta Perusteet-sovelluksesta, jossa pääsynhallinta implementointi oli toteutettu toimivasti. Ongelmia aiheutti eniten sovelluksen ja välityspalvelimen välinen yhteys. Ongelman syyksi viikon lopussa paljastui, että pääsynhallintajärjestelmään ei ollut määritelty Laskenta-sovellusta, jolloin sen toimintoja ei sallittu. Tämä ongelma selvisi työkaverin avustuksella.

Muita viikon tehtäviä on ollut sovelluksen taulukoissa näkyvien tietojen oikeamuotoisuus. Euromäärien esittämiseen tein erillisen funktion, joka muuntaa tiedon haluttuun muotoon ja päivämäärien esittämiseen vastaavanlainen.

3.4 Viikko 4 – JavaScript ominaisuudet

Maanantai 3.2.2020

Tänään jatkoin Tulo-taulukon tietojen ehtojen toteutusta. Taulukossa otetaan huomioon tietojenkäsittelijälle tarpeelliset tiedot ja pyritään sulkemaan pois tarpeettomat tulojaksot. Tämä osana tietojenkäsittelijän työn nopeuttamista. Tietojen erittelyä varten rakensin erillisen funktion taulukon käsittelyyn. Funktio käsittelee ja poistaa käsittelijälle turhat tiedot käsiteltävästä objektista ja palauttaa uuden objektin, joka päivittyy taulukkoon.

Käytin ison osan työskentelyajasta oikeanlaisen metodin löytämiseen. Pienimuotoisiin testeihin käytin muutamaa eri tietojen erittelytapaa ja tietojen objektista poistamista. Parhaaksi totesin splice() metodin jolla voin poistaa tietyn solun Array-tietorakenteesta. Yksinkertaisimman ja kattavan malliesimerkin elementin poistoon Array-tietorakenteesta löysin Mozillan Developer dokumentaatiosta, jota hyödynsin ratkaisussa. (Mozilla 2020)

Tiistai 4.2.2020

Tänään jatkoin muiden taulukoiden tietojen ehtojen toteutusta. Otin työn alle Tulot-taulukon lisäksi myös Asumismenot-taulukon sillä ehdoissa on yhteneväisyyksiä. Rakensin uuden apufunktion joka päivämäärien käsittelyyn käytettävän DateOutput-funktion tavoin käsittelee taulukoiden päivämäärätietoa. Uusi funktio EndDateOutput käsittelee objektin, jos se pitää sisällään myös loppupäivämäärän. Ajattelin myöhemmin yhdistää nämä kaksi funktiota yhden alle mutta nyt keskityn muihin asioihin.

Osaan käyttötapauksessa määritetyistä ehdoista jouduin kysymään määrittelijöiden tekijöiltä tarkennusta, kun asumismenojen käsittely asiaan perehtymättömälle oli vaikea selkoista. Tarkennuksen kysyminen oli kuitenkin hyvin ajoitettu, kun määrityksistä oltiin joka tapauksessa laatimassa uusi versio. Kysymykseni myös herättivät keskustelua, onko laaditut luonnokset käyttöliittymän näkymästä ajan tasalla muilta osin. Aina kannattaa siis kysyä.

Kun tarkennuksia käyttötapauksessa määritettyjen taulukkojen ehtoihin päivitettiin, oli hyvä aika perehtyä ketterään kehitykseen ja testausautomaatioon. Näistä löytyi tietoa ja lyhyet kurssit kelan sisäisestä perehdytysmoduulista. Testauksen ja testausautomaation huomioon ottaminen tulee olemaan osana omaa kehitystä, joten oli hyvä ajoitus hankkia perusteet haltuun. Iltapäivällä pääsin jo käyttötapauksen tarkennuksiin käsiksi ja hieman miettimään toteutusta. Tekemisen aloittaminen jää kuitenkin seuraavaan päivään.

Keskiviikko 5.2.2020

Tänään jatkoin asumismenot ja tulot taulukon ehtojen mukaan määrittelyä. Eilen iltapäivällä saamien tarkennettujen käyttötapausten perusteella sain taulukot päällisin puolin valmiiksi mutta hienosäätöä varmasti löytyy. Oikeiden ehtojen ja niiden implementoinnin lisäksi olen saanut pähkäillä JavaScriptin omien ominaisuuksien kanssa. Yksi isoimmista kompastuskivistä tässä taulukoiden täytössä on ollut objektiin viittaukset. Tähän ongelman ratkaisin kloonauksella, johon ohjeen löysin Stackoverflow sivustolle postatusta kysymyksestä JavaScript objekteihin liittyen. (Stackoverflow 2009)

Päivän muita töitä olivat DateOutput-komponentin ja EndDateOutput-komponentin yhdistäminen yhdeksi ja siihen sisälle ehtolauseet määrittämään onko syötetty tieto vain alkupäivämäärä vai onko siinä myös loppupäivämäärä. (Kuva 8.)

```
export default function DateOutput(sdate, edate){

    if (sdate === undefined || sdate === null || typeof sdate !== 'string' || sdate.length !== 10 ) {
        return("-");
    }

    let token1 = sdate.split("-");
    let date1 = token1[2] + "." + token1[1] + "." + token1[0] + " -"

    if (edate === undefined || edate === null || edate.length !== 10 || typeof edate !== 'string'){
        return(
            date1
        )
    }

    let token2 = edate.split("-");
    let date2 = token1[2] + "." + token1[1] + "." + token1[0] + " - " + token2[2] + "." + token2[1] + "." + token2[0];
    return (
        date2
    )

}
```

Kuva 8. Uusi DateOutput-komponentti.

Torstai 6.2.2020

Tänään otin yhteyttä ratkaisuarkkitehtiin, joka oli työskennellyt saman sovelluksen mutta eri teknologia toteutuksen parissa. Häneltä halusin tietää rajapintakutsuista palautuviin tietoihin liittyviä tarkennuksia. Ennen lounasta oli myös palaveri koskien rajapintoja sovellusarkkitehtuurillisesta näkökulmasta. Palaverissa käytiin läpi rajapinta teknologian mahdollisuuksista päivityksien suhteen ja keskusteltiin muutoksista.

Iltapäivällä tein pieniä korjauksia CSS-tyyleihin, jotta taulukon ehdoissa laaditut ilmoitustekstit näkyisivät kokonaisina taulukossa. Sovelluksessa kutsuttuja rajapintoja on tässä sprintissä muokattu ja rajapintojen uuden ratkaisun yhdistäminen sovellukseen vaatii hieman suunnittelua. Tämän ratkaisun kehittämisen aloitin iltapäivällä ja sen toteutus jatkuu seuraavana päivänä.

Perjantai 7.2.2020

Jatkoin muutettujen REST-rajapintojen muutostöitä. REST-rajapintayhteyttä muokataan siten, että rajapintaan lähetään erillinen palvelupyyntö taulukon tietojen laskemiseksi. Palvelupyynnön palautettua onnistunut vastaus tehdään toinen palvelupyyntö, joka hakee lasketut tiedot.

Viikko 4 yhteenveto

Viikolla 4 olen työskennellyt paljon taulukoiden ominaisuuksien parissa. Taulukoiden tietojen määrittelyyn viikon aikana tuli rakennettua monta eri versioita. Jotkut olivat turhia ja jotkut hyödyllisiä, mutta opin tällä viikolla paljon JavaScript-ominaisuuksista, kun selvitin parasta mahdollista tapaa käsitellä taulukoiden tietoja.

3.5 Viikko 5 – Yksikkötestit

Maanantai 10.2.2020

Tutustuin tänään yksikkötesteihin ja käytettäviin teknologioihin ja yksikkötestien luomiseen Laskenta-sovellukseen. Koestin testausta CurrencyOutput-komponentilla, johon loin testitapauksia eri mahdollisista päivämäärämuodoista.

Iltapäivällä tiimissä käytiin keskustelua koko projektin tavoitteista ja aikataulusta.

Tiistai 11.2.2020

Perehdyin syvemmin, miten tehdään hyödyllisiä yksikkötestejä. Tähän löytyi artikkeleita Googlesta, kuten mm. Medium-sivuston artikkeli aiheesta, kuinka tehdään hyödyllisiä yksikkötestejä. (Hodges 2019) Artikkelista sain pohjan miten lähden testaamaan sovelluksen komponentteja. Testejä on hyvä luoda ensin komponentin yksinkertaisimmille toiminnoille ja testata että kaikki suunniteltujen virhetilanteiden ilmoitukset todella ilmoittavat virhetilanteissa.

Tähän lisäksi työstin Laskenta-sovelluksessa käytettävien koodien ja tekstien lokalisointia. REST-rajapinnasta palautuvalle koodille esitetään käyttöliittymässä tekstimuotoinen vastine.

Keskiviikko 12.2.2020

Aamupäivällä tein lakkautus- ja hylkäyskomponentteihin toteutuksen. Asumistuen hylkäys- tai lakkautuspaneeli näytetään tilanteessa, joissa tapauksen käsiteltävä ajanjakso on hylätty. Tämän hylkäys- tai lakkautustiedon tarkistukseen tein erillisen funktion, joka käy läpi käsiteltävän etuuden ja poimii tietueen, jos käsiteltävän aikajakson laatu on hylätty tai lopetettu. (Kuva 9.)

```
//Tarkasta onko viimeisin etuusajanjakso hylätty tai lopetettu
function kasiteltavaEtuus(obj){
    if(obj === undefined || obj === null || obj.length === 0){
        return undefined
    }
    const key2 = "laatu"
    let cloned = obj.map(a =>({...a}))
    const tarkastettava = cloned.pop();
    if(tarkastettava[key2] === "LOP" || tarkastettava[key2] === "HYL"){
        return tarkastettava
    }

    return undefined
}
```

Kuva 9. Hylkäyksen ja lakkautuksen tarkastuskomponentti.

Torstai 13.2.2020 ja Perjantai 14.2.2020

Kipeänä

Viikko 5 yhteenveto

Alkuviikosta perehdyin yksikkötesteihin. Testaukseen ja yksikkötestaukseen olin ennen tätä perehtynyt hyvin vähän. Pääasiassa tarkoitus projektiin liittyen oli testata tekemäni sovelluksen uudelleen käytettävät komponentit kuten päivämäärien ja euromäärien käsittelyfunktiot. Tällöin testattuja ja toimivia ratkaisuja ei tehdä uudelleen jokaisessa päivämäärä ja euromääriä käytettävässä kohdassa jatkossa. Testaukseen perehtymisessä luin artikkeleita kuten Mediumin blogipostauksen (Medium 2019) saadakseni käsitystä miksi ja miten tehdä hyödyllisiä yksikkötestejä.

Keskiviikosta eteenpäin oli valitettavasti pahasti flunssassa ja sairauslomalla. Ennen sairauslomaa sovellukseen yksikkötestauksen lisäksi olin ehtinyt lisäämään toteutukseen etuuksien hylkäys- ja lakkautuskomponentit, ”seuraava sivu”-painikkeen ja tietojen käsittelyssä käytettävien koodien lokalisoinnin. Käytettäviä koodeja tietojen käsittelyssä kertyy projektin aikana tiedettävästi paljon. Ensin koodeille ja koodien mukaan määritellyille teksteille tehdään sovelluskohtaiset lokaalit toteutukset mutta myöhemmin koodit ja koodien merkityksen haetaan tietokannasta. Näiden koodien myötä sain myös viimein taulukoiden ja hylkäys- ja lakkautuskomponenttien toimimaan.

.

3.6 Viikko 6 – REST-rajapinta

Maanantai 17.2.2020

Sprint-suunnittelupäivä

Tiistai 18.2.2020

Eilinen sprintin suunnittelu jatkui vielä aamulla tämän päivän puolelle, jolloin itse tekeminen jäi vähemmälle. Keskustelua käytiin tämän sprintin uudesta työtehtävästä eli eri teknologian käyttämiseen seuraavan Perusteet-sovelluksen paneelin kehittämiseen. Tähän asti olen rakentanut Laskenta-sovellusta Reactilla ja nyt rakentaisin myös Perusteet-sovellukseen paneelin Java-pohjaista Spring Boot-kehitysympäristöä käyttäen.

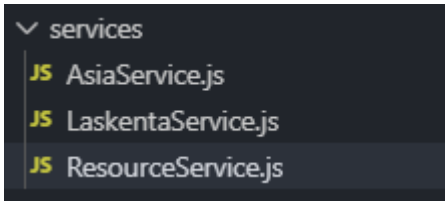
Sain kuitenkin keskustelujen jälkeen jatkettua Laskenta-sovellusta siten että liitin Redis-sessionhallinta järjestelmän Laskenta-sovellukseen. Ehdin myös lisäämään automaatiotestauksessa tarvittavia id-tietoja inputelementteihin, joiden toimintaa testataan. Redis-sessionhallinta ulkoistaa sovelluksen sessionhallintaa, joka mahdollistaa sovelluksen käytön kahdessa tai useammassa kontissa Openshift-ajoalustassa. Tämä mahdollistaa virhetilanteista palautumisen ilman että sillä on vaikutusta asiakkaan käyttökokemukseen.

Keskiviikko 19.2.2020

Tänään aloitin Spring Boot toteutuksen suunnittelun kollegoiden avustuksella. Aamulla harjoittelijapari perehdytti nykyisen Java-pohjaisen Perusteet-sovelluksen toimintaan ja kuinka hän oli sitä rakentanut. Iltapäivällä ratkaisuarkkitehti perehdytti lyhyesti vielä itse Javan perustoimintoihin Perusteet-sovelluksessa. Ja lisäksi itsenäisesti tutkinut toteutusta ja koodia. Yhtenä tämän kolmen viikon sprintin tehtävinä olisi siis rakentaa Thymeleaf+Spring boot teknologialla Perusteet-Thymeleaf toteutukseen Muut Etuudet -paneeli.

Torstai 20.2.2020

Eilen aloittamani toisella teknologialla toteutus asetettiin tänään jäihin, kun sovellussuunnittelijat antoi ohjeet sovelluksien yhtenevään virheenkäsittelyyn. Virheenkäsittelyä en ollut aikaisemmin ottanut huomioon Laskenta-sovelluksen toteutuksessa tarkempien käytännön ohjeiden vielä puuttuessa. Ohjeiden mukainen virheidenkäsittely vaatii paljon refaktorointia REST-rajapinta kutsujen toteutuksessa. Nykyinen toteutus vaihdetaan Service-malliseksi, (Kuva 10.) jolloin uusien kutsujen vaadittu koodimäärä vähenee ja sovelluksen rakenne selkenee ja siihen on helpompi lisätä uusia toimintoja kuten REST-kutsuja ja virheenkäsittelyä. (Kuva 11.)



Kuva 10. Service-mallin kansiorakenne.

```
export default class LaskentaService extends ResourceService {

    async aloitaLaskenta(asiaId) {
        return await this.post(`/asumistuki/asia/${asiaId}/laskelma/laskenta`);
    }

    async aloitaMaksu(asiaId, muutosaikaleima) {
        return await this.post(`/asumistuki/asia/${asiaId}/laskelma/maksunmuodostus`, muutosaikaleima);
    }

    async haeLaskentatiedot(asiaId) {
        return await this.get(`/asumistuki/asia/${asiaId}/laskelma`);
    }


}
```

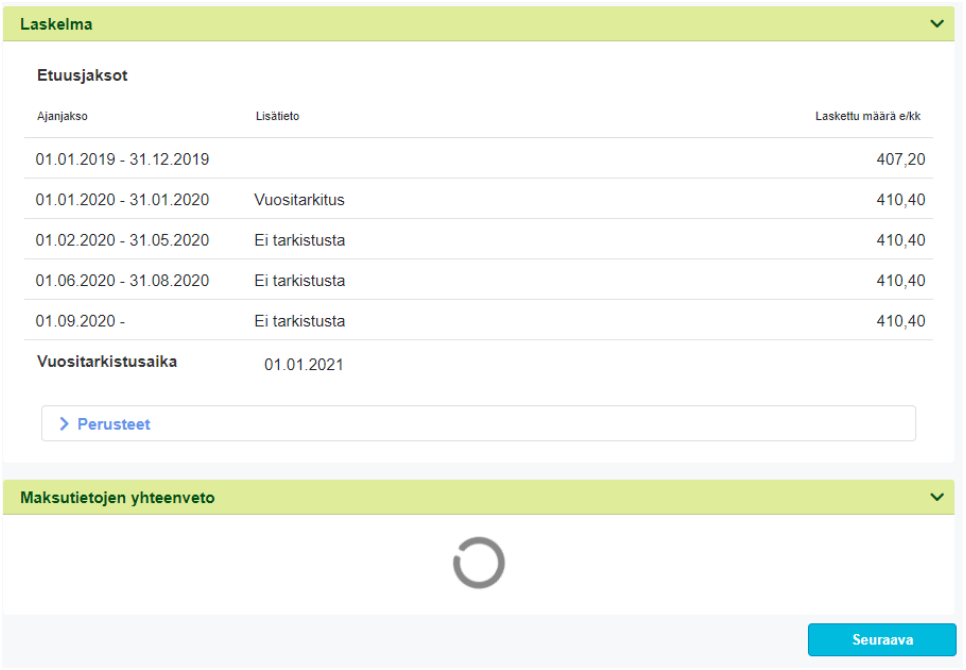
Kuva 11. LaskentaServicen kutsut.

Perjantai 21.2.2020

Tänään jatkoin REST-rajapinta kutsujen refaktorointia. Lähdin toteuttamaan tilanhallintaa Reactin Reducer-ominaisuudella. Kun tietojen hakeminen on lataus tilassa, näytetään Loader-elementti. Loader-elementti on lataustilaa esittävä animoitu pyörivä ympyrä. Otin myös Panel-komponentin käyttöön Perusteet-sovelluksesta. Laskenta-sovelluksessa myös kutsujen täytyy toimia synkronisesti.

Viikko 6 yhteenveto

Aloitin tällä suunnittelujen ja palaverien täyteisen alkuviikon jälkeen Laskenta-sovelluksen refaktorointia enemmän yhteneväksi Perusteet-sovelluksen toteutuksen kanssa. Yhteneväisyyttä lisää service-mallinen keskitetty REST-kutsujen toteutus. Myös Perusteet-sovelluksessa käytetty Panel-komponentti joka ottaa vastaa parametreinä paneelissa käytettävät tiedot ja luo paneelin raamit ja logiikan. (Kuva 12.) Tähän tarvitsee sen jälkeen vain lisätä paneelin sisäiset komponentit kuten taulukot.



Kuva 12. Panel-komponentti paneelit ja Loader-elementti.

3.7 Viikko 7 – Virhe- ja tilanhallinta

Maanantai 24.2.2020

Viikko alkoi React Hook-ominaisuuksiin perehtymisellä ja niiden implementoinnilla Laskenta-sovellukseen. Mallia olen saanut myös tähän Perusteet-sovelluksesta mutta joudun itse päättelemään tarpeellisuuden, kun Laskenta-sovelluksen toimintaperiaate ja tiedonkulku toimii eriävästi. Laskenta-sovelluksen tietoja ei muokata mutta taustalla ei-synkroniset REST-rajapinta kutsut ovat riippuvaisia toistensa palautuvista tiedoista.

Näin ollen tilanhallintaa tarvitaan mutta miten se toteutetaan toimivasti, on ollut ongelmana ja aiheessa opeteltavaa.

Sovellus on tavoitteena ajaa Openshift 4 ympäristössä tulevaisuudessa ja tällä hetkellä valmiit järjestelmän sovellukset pyörivät Openshift 3 versiossa. Aiheesta oli lyhyt palaveri tiimin sovellusarkkitehtuurista vastaavilla, jossa olin mukana.

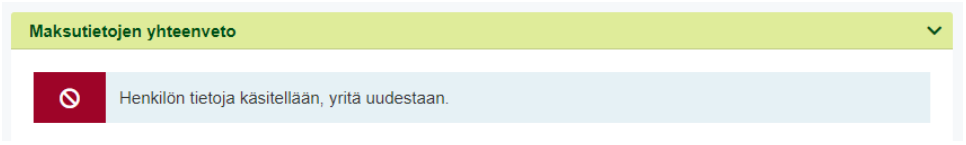
Tiistai 25.2.2020

Tänään päädyin yksinkertaiseen ratkaisuun komponenttien tilanhallintaan useState-Hook ominaisuutta hyväksi käyttäen. Komponentissa kutsuttavat REST-rajapinta kutsut päivittävät tilansa kutsujen onnistuttua tilanhallintaan. Tällä tavoin voidaan määrittää ehdollisuuksia kuten että toisen kutsun täytyy olla valmistunut ennen kuin seuraava kutsu lähetetään.

Sopivan tilanhallinnan päättämiseen sain apua mediumin artikkelista, jossa käsiteltiin -React Hook-ominaisuuksia ja niiden kustomointia. Artikkelissa käsiteltiin monia eri tasoisia tilanhallinta ratkaisuja ja niiden riskialttiutta ja tarpeellisuutta. (Dolidze 2019)

Keskiviikko 26.2.2020

Tänään sain virheenkäsittelyn implementoitua Laskenta-sovelluksen REST-rajapinta kutsuille ja valmistelin logiikan tulevalle Maksujenyhteenveto-paneelille. Mahdolliset virheilmoitukset näkyvät nyt sovelluksessa Alert-komponentin avulla. (Kuva 13.) Alert-komponentti toimii myös huomautuksien esittämiseen, jotka ovat REST-rajapinta kutsussa palautuvia käsittelijälle suunnattuja ohjeita. Lisäsin siis myös huomautukset samalla Laskelma-paneeliin.



Kuva 13. Alert-komponentti virhetiedolla.

Torstai 27.2.2020

Tänään jatkoin toisella teknologialla tehtävää toteutusta, jonka aloitin viime viikolla. Muut etuudet-paneeli on Perusteet-sovelluksen osa ja sen valmiit paneelit toimivat mallina ja toteutuksen pohjana.

Aamupäivä oli pitkälti Perusteet-sovelluksen koodin tutkimista ja omassa paneelissa tarvittavien toimintojen kartoittamista. Ruokakunta-paneelissa oli hyvin paljon yhtäläisyyksiä ominaisuuksilta ja käytin sitä hyväkseni Muut etuudet-paneelin perusrakenteen luomiseen. Valmistelin POJO-malliset REST-kutsuista palautuvia objekteja varten ja liitin vielä tyhjän muut etuudet-sivun Perusteet-sovelluksen jatkoksi.

Java-pohjainen Spring Boot ja Thymeleaf yhdistelmän kehitysympäristö on hyvin erilainen verrattuna Reactiin. Eclipse-kehitysympäristön käytön ja Javan syntaksin totutteluun menee varmasti aikaa mutta sovellus etenee varmasti, kun apua on luvattu isompien ongelmien vastaan tullessa.

Perjantai 28.2.2020

Tämän päivä kului oleellisimpien Java-luokkien rakentamisessa, kun koitin ratkoa työkaverin avustuksella, kuinka ottaa talteen Muut-etuudet REST-rajapinta kutsulta palautuva tieto. Ongelmaksi ilmeni, kun olin katsonut palautuvan datan rakenne on sama kuin Ruokakunta REST-rajapinta kutsussa mutta näin ei ollut. Ruokakunnassa rivien tiedot palautuivat jostain syystä objekteina eikä Array-tietoluokkana. Ongelman selätyksen jälkeen siirryin esittämään haettua tietoa Muut etuudet-paneelin rakentamaani taulukkoon.

Tähän ryhdyttyäni huomasin, että kehitysympäristöni staattisten tiedostojen ajonaikainen päivitys ei toiminut. Näin ollen joutuisin rakentamaan sovelluksen kehitysympäristössä aina muutoksia tehdessäni kokonaan ja tämä vie aikaa. Tämän ongelman syytä ei loppupäivän aikana löytynyt.

Viikko 7 yhteenveto

Tällä viikolla sain paljon aikaan Laskenta-sovelluksen parissa ja Muut etuudet-paneelin Thymeleaf toteutuksen hyvin alkuun. Viime viikolla alkaneessa sprintissä tavoitteena oli refaktoroida Laskenta-sovellus virheenkäsittelylle sopivaan muotoon ja aloittaa Muut etuudet-paneelin toteutus.

Laskenta-sovelluksen refaktoroinnin tavoite oli pystyä tarkkailemaan Laskelma-paneelin REST-kutsujen tiloja. Kutsujen tilojenhallinnan avulla kutsut voidaan suorittaa oikeassa järjestyksessä ja paneelikohtaiset virheilmoitukset ovat mahdollisia. Laskenta-sovellukseen lisätään Maksutietojen yhteenvetopaneeli, kun lopulliset määritykset ja käyttötapausten kuvaukset ovat valmiit. Maksutietojen yhteenvetopaneelilla on omat REST-rajapinta kutsut, joten kutsujen tilojen hallinta on edellytys toimivan koodin ylläpitämiseen sovelluksen kasvaessa. Tällä hetkellä tilojen hallinta toteutetaan useState-ominaisuutta käyttäen. Tämä ratkaisu voi vielä muuttua seuraavassa refaktorointi iteraatiossa. Toimintamalli pysyy todennäköisesti samana, vaikka koodin muuttaisi muotoa.

Suunnitellun Muut etuudet-paneelin toteutuksen aloitin virhekäsittelyn valmiiksi saatuani. Java-pohjainen Spring Boot ja Thymeleaf teknologiayhdistelmä vaati hieman totuttelua uuden kehitysympäristön ja syntaksin takia. Kehitysympäristössä ilmenneet ongelmat hieman hidastaneet tahtia. Kehitysympäristön ongelmien selvitys jatkuu ensi viikkolla.

3.8 Viikko 8 – Spring Boot

Maanantai 2.3.2020

Aamulla ensimmäisenä selvitin kehitysympäristössä ilmenneen ongelman ja pääsin jatkamaan kehittämään itse sovellusta. Ongelmaksi ilmeni, etten ollut sallinut Eclipsen automaattista päivitysominaisuutta. Etsin ongelmaa pitkään sovelluksen ja Spring bootin ominaisuuksista ja asetuksista.

Tämän jälkeen koestin Muut etuudet-paneeliin REST-kutsun oikealla datalla toimivaksi ja aloitin ”Lisää muu etuus”-moduulin työstämisen. (Kuva 14.) Näiden lisäksi alustin

muita paneelissa käytettäviä toimintoja kuten tallenna, poista ja peruuta MuutEtuudetController-luokkaan joka ohjaa taustapalvelujen kutsuja ja näin ollen sovelluksen toimintaa.

```
@RequestMapping("/muutetuudet/lisaaetuus")
public String muutetuudetLisaaEtuus(Model model) {
    sessionBean.clearVirheet();

    PaneeliOhjain ohjain = sessionBean.getPaneeliOhjain();
    ohjain.getPaneeli("muutetuudet").setTila("lisaa");
    model.addAttribute("etuus", new Etuus());
    modelConfig.alustaPaneeli(model, "muutetuudet");
    System.out.println("muutetuudet/lisaaetuus kutsuttu ja tila on " + ohjain.getPaneeli("muutetuudet").getTila());
    return "perusteetMuutEtuudet :: muutetuudet-panel";
}
```

Kuva 14. MuutEtuudetControllerin Lisää etuus -funktio.

Tiistai 3.3.2020

Muut etuudet taulukon tietoja hakevan REST-rajapinta kutsun palautusta oli muokattu, joten aamupäivä kului Javan palautuksen vastaanottavien POJO-luokkien muuntamista soveltuviksi uuteen palautuksen malliin. Tämän jälkeen jatkoin paneelin komponenttien työstöä.

Keskiviikko 4.3.2020

Tänään jatkoin Muut etuudet-paneelin komponenttien ja kokonaisuuden työstämistä. (Kuva 15.) Sovellusarkkitehdin kanssa käytiin keskustelua sovelluksen ensi sprintissä tulevista palavereista Thymeleaf toteutuksia koskien. Palavereissa käsitellään refaktorointia ja koodin yleistä näkymää.

```
<a id="perusteet-ruokakunta-lisaa-etuus"></a>
<th:block th:if="${ohjain.paneelit.get('muutetuudet').tila == 'lisaa'}">
    <th:block th:replace="/muutetuudet/lisaaEtuusForm.html :: lisaaEtuus"></th:block>
</th:block>
```

Kuva 15. Lisää uusi etuus -lomakkeen lisäys Thymeleafillä.

Torstai 5.3.2020

Tänään jatkoin käyttöliittymän lomakkeiden työstämistä Thymeleaf template-moottorilla ja kun toteutusten tekeminen takkusi ajattelin lukea lisää dokumentaatioita. Dokumentaatioita tutkiessa tuli vastaan tieto, että nykyisessä toteutuksessa käytetty Spring Bootin RestTemplate-ominaisuus ”vanhenee” tulevissa Spring Boot -versioissa. Tästä sovellusarkkitehdille huomautettuani pidimme lyhyen palaverin Perusteet-sovelluksen nykyisestä tilasta ja todettiin että ensi sprintissä täytyy tehdä refaktorointia sovelluksen nykyiseen toteutukseen.

Perjantai 6.3.2020

Tämän päivän olen suunnitellut ja yrittänyt tehdä RestTemplaten korvaavaa WebClient toteutusta Perusteet-sovelluksen jo olemassa olevalle Hakemistiedot-paneelin REST-rajapinta kutsulle. Tämän toteutukseen olen etsinyt mallia Spring Bootin dokumentaatioista (Spring 2020) ja blogeista ja artikkeleista.

Viikko 8 yhteenveto

Alkuviikosta työstin komponentteja ja moduuleja Muut etuudet-paneeliin. Testasin perustoimintoja mockdatalla, jota käytin paneelin perustoimintojen testaamiseen.

Lisätoiminallisuuksia rakennellessa koin nykyisen toteutuksen oleva hieman vajavainen toiminnoilta käytettäväksi ja kehittää. Sovellusarkkitehdin kanssa keskustellessa aiheesta todettiin tarpeelliseksi tulevassa sprintissä tehdä muutoksia nykyiseen toteutukseen. Tämän jälkeen koitin perehtyä lisää Spring Bootin ja Thymeleaf dokumentaatioon.

3.9 Viikko 9 – Thymeleaf

Maanantai 9.3.2020

Sprintin suunnittelupäivä

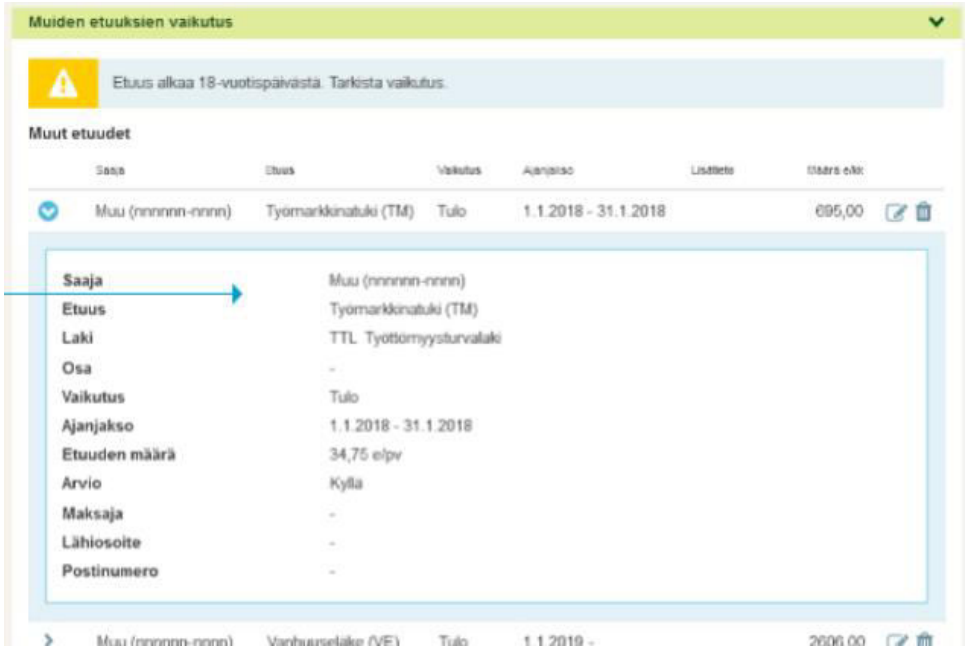
Tiistai 10.3.2020

Tänään tein Muut etuudet -paneelin Muokkaus-lomakkeiden rakentamista ilman täyttä toiminallisuutta. Taustapalveluiden puuttuessa täysien toiminnollisuuksien toteutus on haastavaa, joten pääosin työstän tällä hetkellä vain komponenttien ulkoasua.

Keskiviikko 11.3.2020

Koestin Thymeleafin onClick-attribuuttia jonka avulla olisin toteuttanut taulukossa etuuskohtaisten tietojen tarkastelun. (Kuva 16.) Etuuskohtainen tietojen tarkastelu on käyttötapauksessa määritetty niin että riviä painettaessa aukeaa kuvan näkymä. (Kuva 17.) Tämän implementoinnissa ilmeni ongelmia JavaScriptin ja sovelluksen komponenttien esittämisjärjestyksien ajoituksissa, joiden selvittelytyö jatkuu.

Kuva 16. Nykyinen keskeneräinen toteutus.



Kuva 17. Etuuskohtaisen tarkastelun tavoite ulkoasu.

Iltapäivällä pidimme refaktorointi aiheisen palaverin Perusteet-sovelluksen suunnasta. Sovellusarkkitehti oli paneutunut tarkemmin nykyiseen toteutukseen ja ehdotti korjaus toimenpiteitä. Toimenpiteet kuulostivat kovin haastavilta suorittaa omalla kokemuksella. Sovimme tiimin kesken edetä asteittain refaktoroinnin kanssa. Ensimmäisenä toimenpiteenä on sovelluksen poikkeustilojen käsittely. Tämä aihe oli itselleni hieman tuntematon, joten aloitin tutkimalla aihetta.

Torstai 12.3.2020

Tänään aamupäivän jatkoin Javan poikkeuksien tutkimista ja koestamalla, kuinka poikkeustilanteet rakentuvat Java-sovelluksissa.

Iltapäivällä palasin jälleen komponenttien toimintojen pariin edistääkseni myös niitä.

Perjantai 13.3.2020

Spring Boot ja Thymeleafillä toteutettua Perusteet-sovelluksen REST-rajapinta kutsujen refaktorointia suorittaa tänään Java-kielen paremmin tunteva tiimiläinen. Hän luo REST-rajapinta kutsuihin sapluunan, jota voin hyödyntää myös Muut etuudet-paneelissa.

Perusteet-sovelluksen sapluunaa odotellessani aloitin Laskenta-sovelluksen Maksutietojen yhteenveto-paneelin rakentamisen. Paneelin olin jo alustanut edellisessä sprintissä valmiiksi ja siitä puuttui vain taulukot ja toimivat taustapalvelut. Aloitin luomalla taulukoita. Maksutietojen yhteenveto-paneeli on taustapalveluilta ja toiminnoilta Laskelma-paneelin toteutusta vastaava.

Viikko 9 yhteenveto

Tällä viikolla työskentelin Perusteet-sovelluksen, että Laskenta-sovelluksen parissa. Perusteet-sovellusta refaktorointiin ja toteutusta työstettiin tiimin kanssa. Laskenta-sovelluksen parissa työstin Maksutietojen yhteenveto-paneelia.

Tällä viikolla osaaminen on tullut vastaan Java-kielen kanssa ja vaativampia töitä on ollut ohjattu muille tiimiläisille. Tekemistä on silti riittänyt molempien sovelluksien ulkoasun parissa, jonka kehitystä olen edistänyt osaavampien henkilöiden paneutuessa haastavampiin ominaisuuksiin ja sovellusarkkitehtuurillisiin ongelmiin.

3.10 Viikko 10 – Iteratiivinen kehittäminen

Maanantai 16.3.2020

Aloitin viikon Perusteet-sovelluksen Muut etuudet-paneelin REST-rajapinta kutsujen toteutuksen refaktoroinnilla. Sovellusarkkitehti oli luonut service-mallisen REST-rajapinta kutsujen toteutuksen pohjan Hakemistiedot-paneelin kutsuihin. Service-mallia käytetään myös React.js toteutuksissa ja se on todettu toimivaksi. Service-mallin toteutus vastaa paremmin oikeaoppista sovelluksen rakennetta, jossa vältetään koodin toistamista monessa eri kohteessa. Sain muutoksen valmiiksi suurimmalta osin ja käsiteltävän tapauksen tietojen haku toimii. (Kuva 18.) Seuraavana työlistalla on muokkaus, tallennus ja poisto toiminnot samaan paneeliin.

```
@Service
class MuutEtuudetServiceImplementation extends  RestTemplateService implements MuutEtuudetService {

    @Autowired private PerusteetProperties properties;
    @Autowired private PerusteetSessionBean sessionBean;

    @Override
    public Response<Etuudet> haeEtuudet() {
        return get(properties.getMuutetuudetHakuRest(), null, Etuudet.class, sessionBean.getAsiaId());
    }
```

Kuva 18. Service-mallinen REST-rajapinta kutsujen toteutus.

Tiistai 17.3.2020

Olen viime aikoina vaihdellut kahden toteutettavan teknologian välillä useasti, ja tämä on häirinnyt työskentelyäni huomattavasti. Välttääkseni viimeaikojen teknologioiden välillä vaihtelua ja päätin keskittyä viemään yhden tehtävän loppuun asti kerrallaan. Aloitin Laskenta-sovelluksen Maksutietojen yhteenveto-paneelista. Tähän vaikutti myös Muut etuudet-paneelin taustapalveluiden keskeneräisyys ja Maksutietojen yhteenveto-paneelin taustapalvelut saatiin juuri valmiiksi.

Laskenta-sovelluksen Maksutietojen yhteenveto-paneelin taustapalveluiden valmistuttua pystyn toteuttamaan sovelluksen kokonaisuudessaan valmiiksi. Tietooni tuli myös, että Laskenta-sovelluksen pitää toimia kahdessa eri tilassa, rekisteröinti- ja kyselytila. Rekisteröintitilassa asumistietoa käsitellessä ja kyselytilassa tarkastelua varten, kun käsittely on varmennettu.

Keskiviikko 18.3.2020

Tänään olen tehnyt refaktorointia Laskenta-sovellukseen pitäen mielessä Rekisteröinti ja kyselytilan lisäämisen. Olen pyrkinyt poistamaan ehdollisuuksia, joissa nojataan Laskennan ja Maksumuodostuksen käynnistyksiin. (Kuva 19.) Rekisteröinti kulku ja kysely kulku eroavat siten että kyselyssä tietoja ei lasketa vaan pelkästään haetaan tieto.

Refaktoroinnin kohteena oli ensimmäisenä sovelluksen palvelinkutsut ja tilojenhallinta. (Kuva 20.)

```
service.haeLaskentatiedot(asiaID)
    .then(result => {
        if(result.virhetieto){
            console.log("haelaskenta virhetieto")
            setLaskelmaErrors(result.virhetieto)
            setLaskelmaLoad(false)
        }
        if(result.selite){
            console.log("haelaskenta selite")
            setLaskelmaErrors(result.selite)
            return
          }
        setLaskelma(result.payload)
        setHylkays(etuudenTarkistus(result.payload.maarajaksot))
        setLaskelmaLoad(false)
        console.log("haelaskenta done")
    })
```

Kuva 19. Fetch-kutsu ennen refaktorointia.

```
    service.haeLaskentatiedot(asiaID)
    .then(res => {
        setLaskentaState({
        laskentaTiedot: res.payload,
        suojaukset: res.suojaukset,
        laskentaLoading: false,
        laskentaError: res.selite,
        kasittelynKesk: kasiteltavaEtuus(res.payload.maarajaksot)
        });
    })
}
```

Kuva 20. Fetch-kutsu refaktoroinnin jälkeen.

Torstai 19.3.2020

Tänään lähdin toteuttamaan mahdollisuutta toimia kysely- ja rekisteröintitilassa. Toisena työnä oli yhdistää Router-sovellus ja Laskenta-sovellus Web Component -teknologialla tehdyn Router-navigator-komponentilla. (Kuva 21.) Router-navigator-komponentti on rakennettu uudella Web components -teknologialla (Web Components 2020), jolla voi laajentaa HTML-elementtejä. Laskenta-sovelluksen kysely- ja rekisteröintitila määritetään Router-sovelluksesta. Laskenta-sovellus saa kyselytilan määrityksen parametrinä Router-sovelluksesta tulevassa kutsussa.

Kuva 21. Router-navigator-komponentti yhdistettynä sovellukseen.

Perjantai 20.3.2020

Poissa

Viikko 10 yhteenveto

Aloitin viikon refaktoroimalla Muut-etuudet paneelin REST-rajapinta kutsujen toteutuksen Service malliin. Tiimin kokeneemman Java-osaajan toimesta toteutettu malli refaktorointi Hakemistiedot-paneelin REST-rajapinta kutsuihin toimi esimerkkinä. Kun sain Muut etuudet-paneelin refaktoroinnin valmiiksi tietojen haun osalta päätin palata tekemään Laskenta-sovelluksen maksutietojen yhteenveto-paneelin loppuun.

Maksutietojen yhteenveto-paneelin taustapalvelut olivat tulleet valmiiksi ja Laskenta-sovelluksen haluttuun toimintaan uutta tietoa kuten sovelluksen ajo rekisteröinti- ja kyselytila.

Eri tiloissa ajamisen lisäksi tällä viikolla lisäsin sovellukseen Web Components -teknologialla tehdyn router-navigator-komponentin. Router-sovellus toimii kelan sisäisen työnhallintajärjestelmän ja asumistukikäsittelyyn rakennettujen sovellusten yhdistävä tekijä. Router-sovelluksen tehtävä on ajaa sovellukset oikeassa järjestyksessä.

4 POHDINTA

Oppimispäiväkirjamuotoisen opinnäytetyöni tavoite viimeiselle opintovuodelle oli kartuttaa teknistä osaamista. Päiväkirjamuotoisessa opinnäytetyössä keskityin tekemiseen ja kokemuksen kerryttämiseen käytännön työtehtävistä. Päiväkirjamuotoinen toteutus tukee myös oppimista päivittäisen työtehtävien kertaamisen ja dokumentoinnin myötä. Ajanjakson aikana tein töitä kokopäiväsesti, joten ajanhallinnallisesti päiväkirjamuotoinen toteutus oli myös kohdallani oikea ratkaisu.

Opinnäytetyön kirjoittamisajanjakson aikana pääsin tekemään monipuolisesti web-kehittäjän työtehtäviä ja kehittämään teknistä osaamistani. Laskenta-sovelluksen parissa pääsin työstämään toteutusta alusta loppuun asti React- ja Node.js-ohjelmointiympäristössä. Tässä tehtävässä pääsin siis näkemään sovelluksen kehityksen kaaren ja siihen vaikuttavat tekijät. JavaScript-kehyksillä ennenkin työskennellessä mutta en tuotantotasoisessa ympäristössä koin tämän työtehtävän erittäin opettavaiseksi ja mielekkääksi.

Muut etuudet-paneelia tehdessäni pääsin tutustumaan palvelinpuolen web-kehitysympäristöön Spring Boot- ja Thymeleaf-yhdistelmällä. Java-pohjainen kehitysympäristö toi mukanaan haasteita syntaksin ja Java-sovellusten kehittämiskäytännöiden myötä. Kokemusta web-sovellusten kehittämisestä minulla oli aiemmin vain selainpuolen JavaScript-kehyksillä. Osaamattomuuteni Java-kielen kanssa toi pientä turhautumista ajoittain. Koin kuitenkin mahdollisuuden kehittää kahdella eri teknologialla eduksi. Spring Boot on yleinen Java-pohjaisten web-sovellusten kehitysympäristö, joissa palvelinyhteys on suuressa roolissa sillä Spring Boot on luotu nopeaan kommunikointiin palvelimen kanssa, kuten REST-rajapintatoteutukset, jotka hakevat tietoa palvelimilta.

Projekti oli mielenkiintoinen kokonaisuus, jossa eri teknologioita oli paljon. Projektin ollessa vielä kehitysvaiheessa pääsin seuraamaan monia arkkitehtuurillisia keskusteluita ja oppimaan moderneista teknologioista, kuten Openshift, Redis-sessionhallinta ja Web Components. Kehitysvaiheessa olevassa projektissa oli myös huonot puolet kokemattomalle. Projekti oli altis muutoksille, kuten tavoitteiden ja sisäisten resurssien muuttumiselle. Resurssien muuttuessa tavat päästä projektin tavoitteeseen täytyy myös muuttua, joten niistä tavoista on monta eri mielipidettä ja

ajatusta tiimin sisällä, tämä johtaa projektin pirstaloitumiseen, jos yhteinen linja puuttuu. Itse koin tämän pirstaloitumisen sekoittavan omaa työskentelyä, kun yleinen käsitys normaalista projektien kulusta on vielä rajallista. Näistä hankaluuksista huolimatta koin projektiosaamiseni kehittyneen huomattavasti lähtötilanteeseen nähden. Projektin tehtävälistalla ja backlogilla navigoiminen ja edistyksen seuraaminen on paljon helpompaa, kun ketterän kehityksen terminologia on tullut tutuksi ja tietää mistä tietoa voi etsiä.

Oman työskentelyn ja ajanhallinnan kehittäminen oli yksi tavoitteeni ajanjaksolle. Omien työtehtävien työntuntien arviointi vaatii vielä työstämistä, mutta tähän vaikuttaa myös itse annettujen työtehtävien tarkkuus. Vielä en siis osaa tarkalleen sanoa kuinka monta työtuntia kuluu työtehtävän tekemiseen sen kuultuani mutta kehitystä on tapahtunut paljon omassa työskentelyssä. Koin että ajattelutyössä on tärkeä pitää mieltä virkeänä tauoilla, liikkumisella ja venyttelyllä, tämä joskus unohtuu uppotuessa ongelmien ratkaisuun tai ns. flow-tilassa. Yksi iso kehityskohde jatkossa on avun kysyminen ongelmatilanteissa. Sain huomata, että saatan jumittua pitkiksi ajoiksi ratkomaan ongelmaa, josta kysymällä voisi selvitä nopeammin. Avun kysymiseen ajoissa ohjeistettiin myös tiimini puolesta. Tärkeää olisi siis oppia oikea aika kysyä apua.

Monissa kohtaamissani ongelmissa ja vastoinkäymisissä ajanjakson aikana oli usein syynä kokemattomuus. Siksi näin jälkeenpäin ajatellen oli loistava idea keskittyä käytännön tekemiseen. Vaikka en täysin itsenäiseen tuotantotasoiseen sovellusten kehitykseen pystykään teknisesti, koin saavuttaneeni asettamiani tavoitteita hyvin ajanjakson aikana. Tätä ajatusta vahvisti positiivinen palaute tiimiltä ajanjakson aikana.

LÄHTEET

Dolidze, S. 2019. The Iceberg of React Hooks. 26.3.2019 Medium. Viitattu 25.2.2020 https://medium.com/@sdolidze/the-iceberg-of-react-hooks-af0b588f43fb .

Hodges, N. 2019. 13 Tips for Writing Useful Unit Tests. 7.10.2019 Medium. Viitattu 11.2.2020 https://medium.com/better-programming/13-tips-for-writing-useful-unit-tests-ca20706b5368 .

Mozilla 2020. Array.prototype.splice(). Viitattu 3.2.2020 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice .

Oracle 2020. How to Throw Exceptions. https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html .

Palakollu, S. M. 2019. How and why to use Functional Programming in modern JavaScript. 15.4.2019 #JavaScript. freeCodeCamp. Viitattu 20.1.2020 https://www.freecodecamp.org/news/how-and-why-to-use-functional-programming-in-modern-javascript-fda2df86ad1b/ .

React 2020a. Using the Effect Hook. https://reactjs.org/docs/hooks-effect.html .

React 2020b. Using the State Hook. Viitattu 22.1.2020 https://reactjs.org/docs/hooks-state.html .

Spring 2020. Building a Reactive RESTful Web Service. Viitattu 6.3.2020 https://spring.io/guides/gs/reactive-rest-service/ .

Stackoverflow 2009. How do you clone an Array of Objects in Javascript? Viitattu 5.2.2020 https://stackoverflow.com/questions/597588/how-do-you-clone-an-array-of-objects-in-javascript .

Thymeleaf Onclick 2019. Using thymeleaf variable in onclick attribute. https://stackoverflow.com/questions/32650536/using-thymeleaf-variable-in-onclick-attribute .

Web Components 2020. What are web components?. Viitattu 19.3.2020 https://www.webcomponents.org/introduction .