Henri Jauhiainen

SOVELLUSKEHITTÄJÄN PÄIVÄKIRJA

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2022

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Henri Jauhiainen
Työn nimi	Sovelluskehittäjän päiväkirja
Toimeksiantaja	Kela
Vuosi	2022
Sivut	46 sivua
Työn ohjaaja(t)	Jukka Selin

TIIVISTELMÄ

Toimeksiantajayritys Kela haluaa toteuttavaksi päiväkirjan, jossa seurataan sovelluskehittäjän päivittäisiä töitä. Opinnäytetyö on jaettu kahteen osaan. Ensimmäisessä osassa käydään läpi sovelluskehitykselle olennaisia tekniikoita, sekä sovelluksia, jotka ovat opinnäytetyön toteutusta varten tärkeitä. Toinen osa on neljän viikon pituinen päiväkirja, joka koostuu kahden eri web-sovelluksen kehittämisestä, työtehtävien päivittäisestä raportoinnista sekä viikoittaisista yhteenvedosta ja analyyseistä.

Työn tavoitteena on seurata työnkulkua ja kehitystä sovelluskehittäjänä ja työntekijänä. Tähän kuuluu tekninen osaaminen, yleiset taidot työntekijänä kuten kommunikointitaidot sekä projektityön osaaminen ja työajanhallinta.

Päiväkirjaosuuden aikana raportoitiin neljän viikon ajalta päivittäisiä työtehtäviä käyttöliittymien ja palvelinpuolen kehityksestä. Raportoinnin aikana dokumentoidaan päivittäisessä työssä ilmeneviä ongelmia ja niihin ratkaisuja. Jokaisen viikon lopulla on analyysiosio, jossa kerrataan viikoittaiset työtehtävät ja mietitään, miten opinnäytetyön kirjoittaja kehittyi viikon aikana.

Päiväkirjaosuuden jälkeen on pohdintaosuus, jossa käydään läpi kehitystä sovelluskehittäjänä ja työntekijänä. Pohdinnan aikana mietitään myös, miten työn tekemistä voisi parantaa tulevaisuudessa.

Kirjoittajan tekninen osaaminen ja yleiset taidot työntekijänä kehittyivät ajanjakson ajalta paljon. Myös projektityönteko kehittyi huomattavasti ketterän kehityksen työympäristössä, ja työajanhallinta parani seurantajakson ajalta. Kaikki nämä taidot ovat todella tärkeitä sovelluskehittäjän päivittäisessä työssä.

Asiasanat: ohjelmistokehitys, päiväkirja, Java, Spring boot, Thymeleaf

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Business Administration
Author (authors)	Henri Jauhiainen
Thesis title	Diary of a software developer
Commissioned by	Kela
Time	May 2022
Pages	46 pages
Supervisor	Jukka Selin

ABSTRACT

The thesis's commissioner company Kela wanted a diary that would track the daily work of an application developer. The thesis was divided into two parts. The first part discussed techniques that were essential in software development as well as the applications that were important for the diary part of the thesis. The second part was a four-week long diary which consisted of the development of two different web applications, daily reporting of the work, and weekly summaries and analyses.

The aim of the work was to track the workflow and progress as an application developer and employee. This included technical skills, general skills as an employee such as communication skills as well as project work skills and time management.

During the diary part of the project, daily work tasks were reported for four weeks on the development of front-end and back-end applications. During the reporting period, the problems encountered in daily work and solutions to them were documented. At the end of each week, there was an analysis section to recap the weekly work tasks and to reflect on personal development during the week.

After the diary section, there was a reflection section, which reviewed the development as an application developer and as an employee. This period also included reflection on how the work and skills related to work could be improved in the future.

Personal technical skills and general skills as an employee developed a lot during the reporting period. Project work also improved considerably in an agile development work environment, and time management improved over the period under study. All these skills are very important in the daily work of an application developer.

Keywords: software development, diary, Java, Spring boot, Thymeleaf

SISÄLLYS

Sanasto

Agile board	Scrum-kehitykseen liittyvä ns. taulu, jolla visualisoidaan sprintin aikana tapahtuvat työt ja niiden etenemisprosessit.
Backend	Osa sovelluksesta, joka ei näy käyttäjälle ja jossa toiminnot tapahtuvat.
BitBucket	Versionhallintatyökalu.
Bugi	Virhe tietokoneohjelmassa
CSS	Cascading Style Sheets, tekniikka, jolla muokataan web-sivun ulkoasua.
Ems(em)	Fontin kokoa säätävä skaalautuva yksikkö.
Frontend	Sovelluksen ”etupuoli”, jossa on kaikki mikä käyttäjälle näkyy.
HTML	Hyper Text Markup Language, merkintäkieli, jolla nettisivuja kehitetään
Jenkins	Avoimen lähteen automaatiopalvelin.
Maven	Sovelluksen koontiautomaatiotyökalu.
Responsiviisuus	Nettisivujen ulkoasu, joka mukautuu automaattisesti eri näyttökokoihin ja näkymiin.
Retro	Sprintien lopuksi käytävä ”seremonia”, jossa käydään läpi menneen sprintin tuntemukset ja puhutaan mahdollisista työkaluista seuraaville sprinteille.

SCRUM	Ketterän kehityksen viitekehys, ”työkalu”, jonka avulla rikotaan isommat prosessit pienemmiksi paloiksi.
Sprint	SCRUM -menetelmä, jossa työt jaetaan kolmen viikon pituisiin pätkiin.
Stack	Pino, joka sisältää elementtejä. Sovelluksissa stackilla viitataan sovellusta pyörittäviin kehyksiin.
UI	User Interface eli käyttöliittymä.
URL	Uniform Resource Locator eli websivun osoite.

1 JOHDANTO

Tämä opinnäytetyö kertoo sovelluskehittäjän päivittäisestä työstä. Luvussa kaksi esitellään sovelluskehittäjän työssä olennaisia tekniikoita ja luvussa kolme esitellään opinnäytetyön kannalta tärkeät sovellukset. Luku neljä on päiväkirjamallinen työnseurantaosio. Jokaisen viikon lopuksi on analyysiosio, jossa käydään läpi töitä ja ajatuksia viikon ajanjaksolta. Luku viisi on pohdinta-osuus, jossa käydään läpi kehitystä sovelluskehittäjänä ja työntekijänä neljän viikon ajalta sekä mietintää, miten tulevaisuudessa voisi kehittyä.

Opinnäytetyön toimeksiantajana toimii Kelan eli Kansaneläkelaitoksen IT-palvelujen tulosyksikkö. Sovelluskehittäjä toimii opinto- ja perhe-etuuksien ryhmässä. Opinnäytetyössä kuvataan töitä liittyen suurimmaksi osaksi sekä opiskelijoiden terveydenhuoltomaksu-sovelluksen monipankkitoiminnallisuuden implementointiin että elatustuen hakemus- ja muutosilmoitussovelluksen saavutettavuusmuutosten ja ulkonäköremontin toteuttamiseen.

Työtehtävissä tarvitsee tietoa ja taitoa sovelluskehittämisestä Java-pohjaisilla ympäristöillä, johon kuuluu Spring Boot- ja Thymeleaf -yhdistelmä sekä JSF (Jakarta Server Faces) - ja JSP (Jakarta Server Pages) -yhdistelmä. Edellä mainittujen ympäristöjen lisäksi työssä vaaditaan ymmärrystä palvelinpuolesta (backend) sekä käyttöliittymistä (frontend). Olennaista on myös ymmärtää ketterän kehityksen menetelmät kuten SCRUM.

Opinnäytetyön tavoitteena on seurata työkulkua ja kehittymistä työntekijänä sekä projektityöntekemisen ja oman työajan hallinnan kehitys. Projektityöllä tarkoitetaan tässä tapauksessa ketterää kehitystä ja ketterän kehityksen ryhmässä työskentelemistä. Työajan hallinta yhdistyy myös ketterään kehitykseen, koska ketterässä kehityksessä on tärkeää, että työtunnit suunnitellaan etukäteen ennen uuden sprintin, eli seuraavan kolmen viikon töiden suunnittelun yhteydessä. Koen, että itselläni on näistä jo ihan hyvä pohjakäsitys, mutta tilaa parantaa on loputtomasti.

2 SOVELLUSKEHITYKSESSÄ KÄYTETTÄVÄT TEKNIIKAT

Tässä luvussa käydään läpi tekniikoita, jotka ovat keskeisiä työssä mainittujen sovellusten kehittämisessä. Ennen töiden toteutusta on oleellista, että mainituista tekniikoista on jonkinlainen ymmärrys ja kokonaisnäkemys. Luvun aikana tekniikoita ei esitellä niin laajasti, kun voisi, vaan sen verran mitä on tarve, että niitä pystyy hyödyntämään työkäytössä.

2.1 Spring Boot

Spring Boot on sovelluskehys, joka on variaatio alkuperäisestä Spring-kehyksestä. Pohjimmiltaan Spring Boot on Spring-moduuli, joka tarjoaa Spring-kehykselle RAD (Rapid Application Development) -ominaisuuden (Javatpoint s.a.). Spring Boot kehitettiin helpottamaan sovelluskehittäjiä saamaan nopeammin sovellus alustettua varsinaista kehitystyötä varten, verrattuna alkuperäiseen Springiin, joka vaati paljon enemmän konfigurointia. Spring sai myös kritiikkiä monimutkaisuutensa takia. (Woods 2014.)

Spring Boot tarjoaa mahdollisuudet automaattiseen sovelluksen konfiguraatioon, kolmannen osapuolen kirjastojen käyttöön eikä se tarvitse lainkaan XML-määrittelyjä (Karaman 2014, 3). Spring Bootin toimivuutta varten tarvitaan myös kokoonpanon hallinnan väline, kuten Gradle tai Maven. Spring Boot -sovellukset kehitetään Java-ohjelmointikielellä, ja se vaatii Java Development Kitistä version 1.8 tai myöhemmän (Spring Boot s.a.).

Sovelluskehyksenä Spring Bootia on helppo käyttää Javan mallimoottoreiden kanssa, kuten Thymeleafin tai JSP:n eli Jakarta Server Pagesin kanssa. Spring Boot tarjoaa myös helpon integraation monien sovelluskehitysympäristöjen kanssa kuten Eclipse, Visual Studio Code, IntelliJ IDEA ja Theia. Helpoin sovelluskehitysympäristö Spring Boot -projektien luontia ja opettelua varten on IntelliJ IDEA, koska se tarjoaa sisäänrakennettuna Spring Initializr -projektiavustajan, jolla pystyy helposti luomaan Maveniin tai Gradleen pohjautuvan Spring Boot -projektin.

2.2 Thymeleaf

Thymeleaf on Java-ohjelmointikieleen pohjautuva palvelinpuolen mallimoottori, joka toimii sovelluksissa sekä web-pohjaisessa ympäristössä, että webin ulkopuolella. Thymeleaf tukee XML-, XHTML- ja HTML5-merkintäkieliä, mikä tarkoittaa, että sivumallit voi kehittää näillä kielillä Thymeleafin avulla (Mankale 2015, 227).

Thymeleafin voi käsittää enemmänkin mallimoottorikehyksenä, koska se on hyvin joustava. Thymeleaf perustuu modulaarisiin ominaisuusjoukkoihin, joita sanotaan dialecteiksi. Lisäksi se antaa käyttäjän täysin määritellä omat dialectit, malliattribuutit ja tagit omilla nimillään ja omalla logiikallaan. (Thymeleaf s.a.)

Thymeleafin tarkoitus on korvata web-sovelluksissa vanhentuneet JSP- eli Jakarta Server Pages-teknologiat kokonaan. Sen tarkoituksena on myös tarjota elegantti ja helposti ylläpidettävä tapa luoda sivumallit. (Thymeleaf s.a.) Thymeleaf tarjoaa helpon integraation Spring -kehyksen kanssa lisäämällä thymeleaf-spring -ohjelmakirjaston ja spring-boot-starter-thymeleaf -riippuvuuden projektin Maven- tai Gradle-projektiin, joka sallii projektin kehittämisen ilman enempää konfiguraatiota.

2.3 JSP (Jakarta Server Pages)

JSP eli Jakarta Server Pages (ennen tunnettu nimellä Java Server Pages) on standarditeknologia, joka mahdollistaa datavetoisten ja dynaamisten sivujen luonnin Java-pohjaisiin web-sovelluksiin (Tyson 2019). JSP-sivut koostuvat HTML-merkintäkielestä, johon on upotettu JSP-tageja. JSP-tagien upottaminen mahdollistaa palvelinpuolen koodin ja datan kutsumisen sovelluksen HTML-sovellusnäkymään eli yksinkertaisesti web-sivuun.

JSP-tagit ovat toiselta nimeltään JSP-lausekkeita, jotka on kirjoitettu Expression Language- eli El-kielellä. JSP-sivut pitää ottaa käyttöön Java servlet containerissa (joskus suomeksi viitattu pelkästään sanalla kontti), kuten Tomcat

tai Jetty. JSP-sivuja varten ei tarvitse kirjoittaa lainkaan koodia Java-kielellä, jonka takia JSP-sivujen kehityssykli voi olla hyvin nopea.

Muutokset JSP-sivuihin voidaan tarkistella heti, koska JSP:n kontti kääntää automaattisesti itsensä (Chopra ym. 2005, 11). Lyhyesti kiteytettynä JSP-sivut ovat käytännössä HTML-sivuja erikoisilla merkintätageilla.

2.4 OpenShift

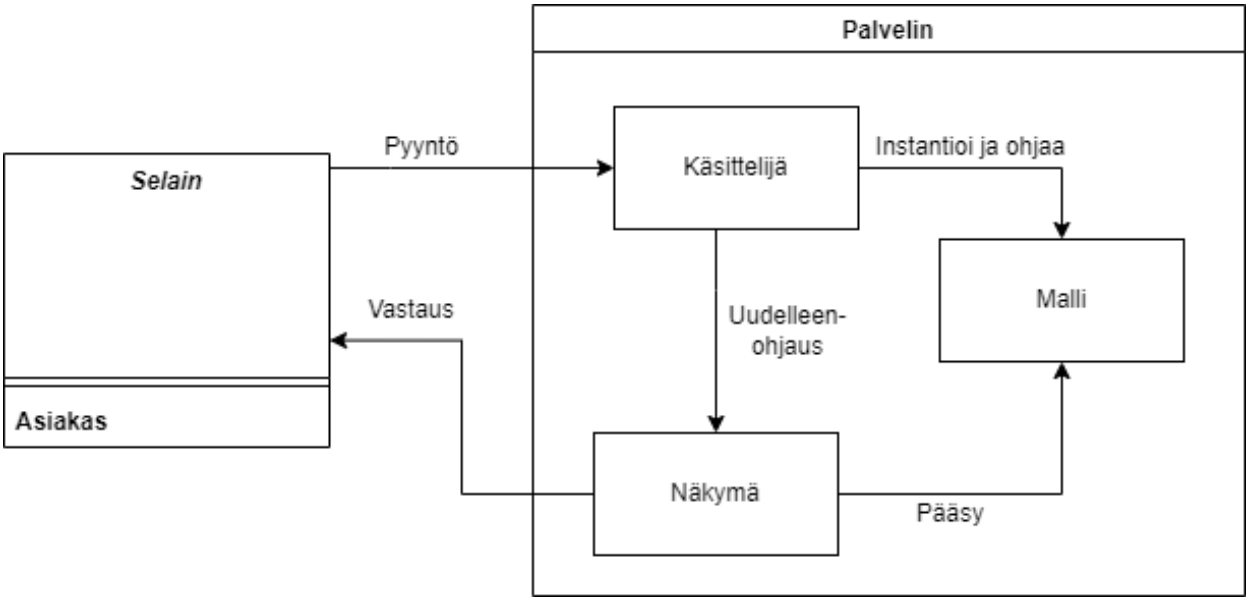
OpenShift on skaalautuva ja monikielinen PaaS (Platform as a Service, sovellusalusta palveluna). OpenShift tukee muun muassa Java-, Ruby-, Python-, Node.js-, PHP- ja Perl-ohjelmointikieliä sekä tietokantoja, kuten MySQL, PostgreSQL ja MongoDB (Gulati 2014, 37). OpenShift auttaa käyttäjää kehittämään ja hallitsemaan konttisovelluksia saumattomasti pilvipalveluissa.

OpenShift mahdollistaa kehittäjälle sen, että kehittäjän täytyy itse vain ohjelmoida, ja OpenShiftin palvelualusta tekee suurimman osan järjestelmän hallinnoinnin tehtävistä automaattisesti (Shipley 2014, 15). OpenShiftin järjestelmä pohjautuu Kubernetes-järjestelmään, joka on avoimen lähdekoodin konttialusta.

OpenShift tarjoaa käyttäjälle automatisoidut ja tiukat tietoturvamääritykset sekä automatisoidut asennukset, päivitykset ja elinkaarihallinnan koko konttipinolle. Konttipino sisältää käyttöjärjestelmän, Kubernetesin, klusteripalvelun ja itse sovellukset. Palvelut tarjotaan täysin pilvessä, tai Kelan tapauksessa privaattipilvessä.

2.5 JSF (Jakarta Server Faces)

JSF eli Jakarta Server Faces (ennen JavaServer Faces) on MVC-arkkitehtuuriin (Model-View-Controller eli malli-näkymä-käsittelijä) perustuva web-pohjainen sovelluskehys. JSF:n tarkoitus on yksinkertaistaa käyttöliittymien rakentaminen Java-pohjaisia web-sovelluksia varten muun muassa yli sadan valmiin UI-tagin avulla (Mkyong 2013).



Kuva 1. Tyypillinen MVC:n perustuvan web-applikaation arkkitehtuuri

Tavallinen MVC-malli on havainnollistettu kuvassa 1, ja se toimii niin, että käyttäjän/asiakkaan selaimesta lähetetään pyyntö palvelimelle, jossa Controller eli käsittelijä instantioi ja ohjaa Modelia eli mallia sekä ohjaa käyttäjän Viewiin eli näkymään. Näkymä on sovelluksen osa, joka näkyy käyttäjälle. Näkymästä käyttäjä saa huomaamattaan pääsyn malliin, joka sisältää kaiken sovelluksen logiikan. Kaikki mitä käyttäjä tekee näkymässä, päivittyy mallin datassa, kuten tietokannat ja syöttökentät. (Kumar 2021.) Käsittelijä nimensä mukaisesti käsittelee kaiken, mitä näkymässä ja mallissa tapahtuu. Käyttäjän pyynnön jälkeen näkymästä lähetetään päivitetty vastaus takaisin selaimelle, joka sisältää käyttäjän pyytämän datan.

JSF-pohjaisissa sovelluksissa malli sisältää Managed Beanit, eli Javan beanluokan, jota JSF hallitsee. Managed Beanit voivat sisältää metodit gettereille ja settereille, sovelluksen logiikkaa tai esimerkiksi arvoja HTML-lomakkeille. Näitä voidaan JSF:n 2.0-versiolla rekisteröidä sovellukseen annotaatioilla, kuten @ViewScoped, joka tarkoittaa sitä, että bean “herää henkiin”, kun käyttäjä siirtyy tiettyyn näkymään ja tuhoutuu, kun käyttäjä siirtyy pois.

Näkymä sisältää käyttäjälle näkyvät sivut, eli JSF-sovelluksissa JSP-sivut. JSP-sivut ottavat dataa mallin beaneista ja yhdistää ne sivujen UI-komponenttien kanssa.

JSF-sovellusten käsittelijä sisältää myös Faces Servletin ja faces-config.xml-tiedoston. Faces Servlet toimii keskeisenä käsittelijänä, joka alustaa kaikki JSF-komponentit ennen kuin JSP-sivut latautuvat.

```
<managed-bean>
    <managed-bean-name>pc_Hakemus</managed-bean-name>
    <managed-bean-class>pagecode.tilesContent.Hakemus</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
<managed-bean>
    <managed-bean-name>pc_LapsenTulot</managed-bean-name>
    <managed-bean-class>pagecode.tilesContent.LapsenTulot</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
<managed-bean>
    <managed-bean-name>pc_Elatusapu</managed-bean-name>
    <managed-bean-class>pagecode.tilesContent.Elatusapu</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

Kuva 2. Faces-config.xml esimerkki

Faces-config.xml sisältää kaikki Managed Beanit, joihin on määritelty nimet, luokat ja scopet eli annotaatiot. Tässä elatustuen sovelluksen esimerkkikuvassa (kuva 2) beanit ovat request-parametrillä toteutettuja, eli bean on elossa vaan sen ajan, milloin se saa käyttäjältä HTTP-pyyntöjä ja pystyy lähettämään niihin vastauksen.

3 SOVELLUSTEN ESITTELY

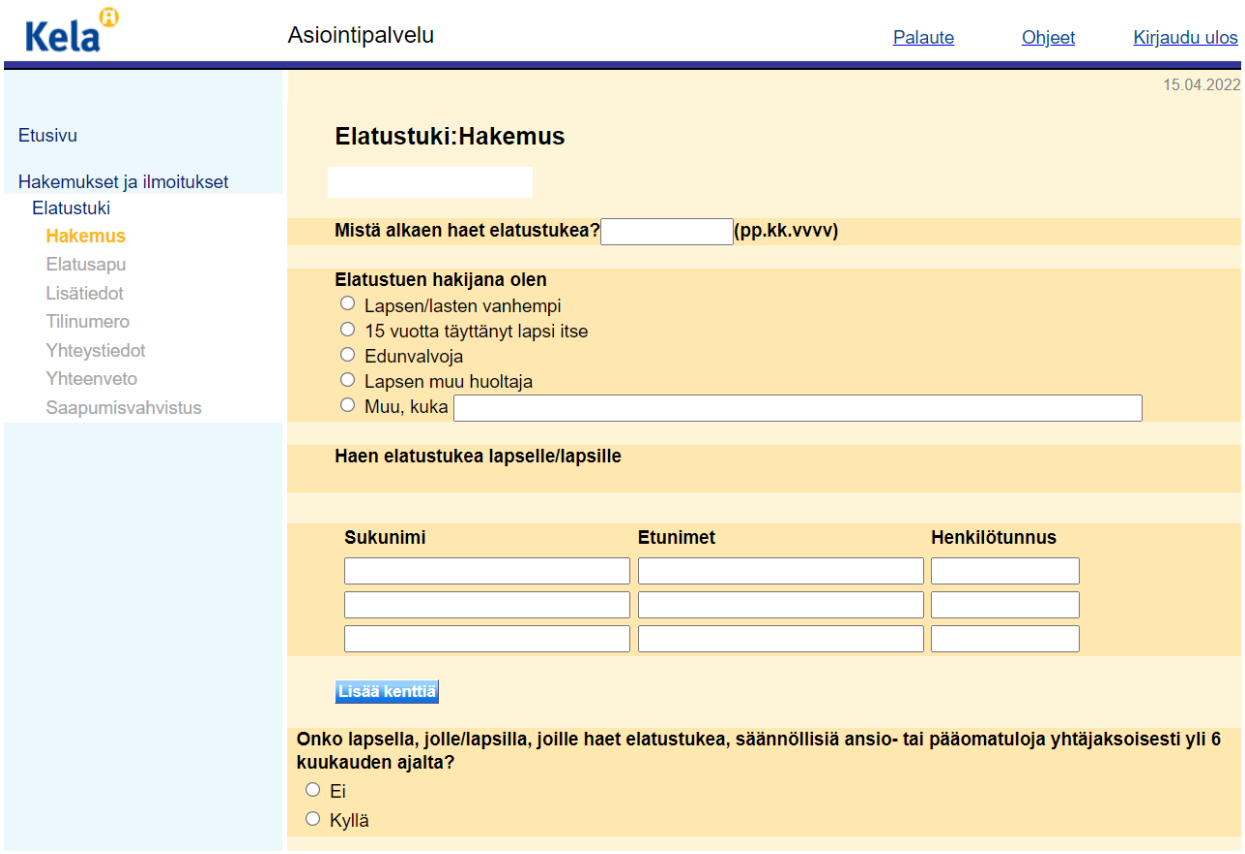
Tämän luvun tarkoituksena on esitellä ne sovellukset, joihin liittyvää kehitystyötä esittelen tähän opinnäytetyöhön sisältyvässä sovelluskehittäjän päiväkirjassa. Nämä kaksi sovellusta ovat siis elatustuen hakemus- ja muutosilmoitussovellus sekä opiskelijoiden terveydenhuoltomaksusovellus. Kummatkin sovelluksista ovat Java-pohjaisia, mutta sisältävät erilaiset sovelluskehykset (kehys-stackit). Kummankin sovelluksen koodieditorina toimii Eclipse, koska se tarjoaa hyvät integraatiomahdollisuudet sekä Mavenille, että Spring Bootille, joilla sovelluksia käytetään lokaalissa testiympäristössä.

3.1 Elatustuen hakemus- ja muutosilmoitussovellus

Ensimmäinen esiteltävä sovellus on elatustukea varten tehty hakemus- ja muutosilmoitussovellus. Sovellus toimii JSP:n ja JSF:n yhdistelmästackilla. Sovellusta voi käyttää lokaalisti Maven-työkalun avulla, ja testi- ja tuotantoympäristöihin se käännetään Jenkins-työkalun avulla.

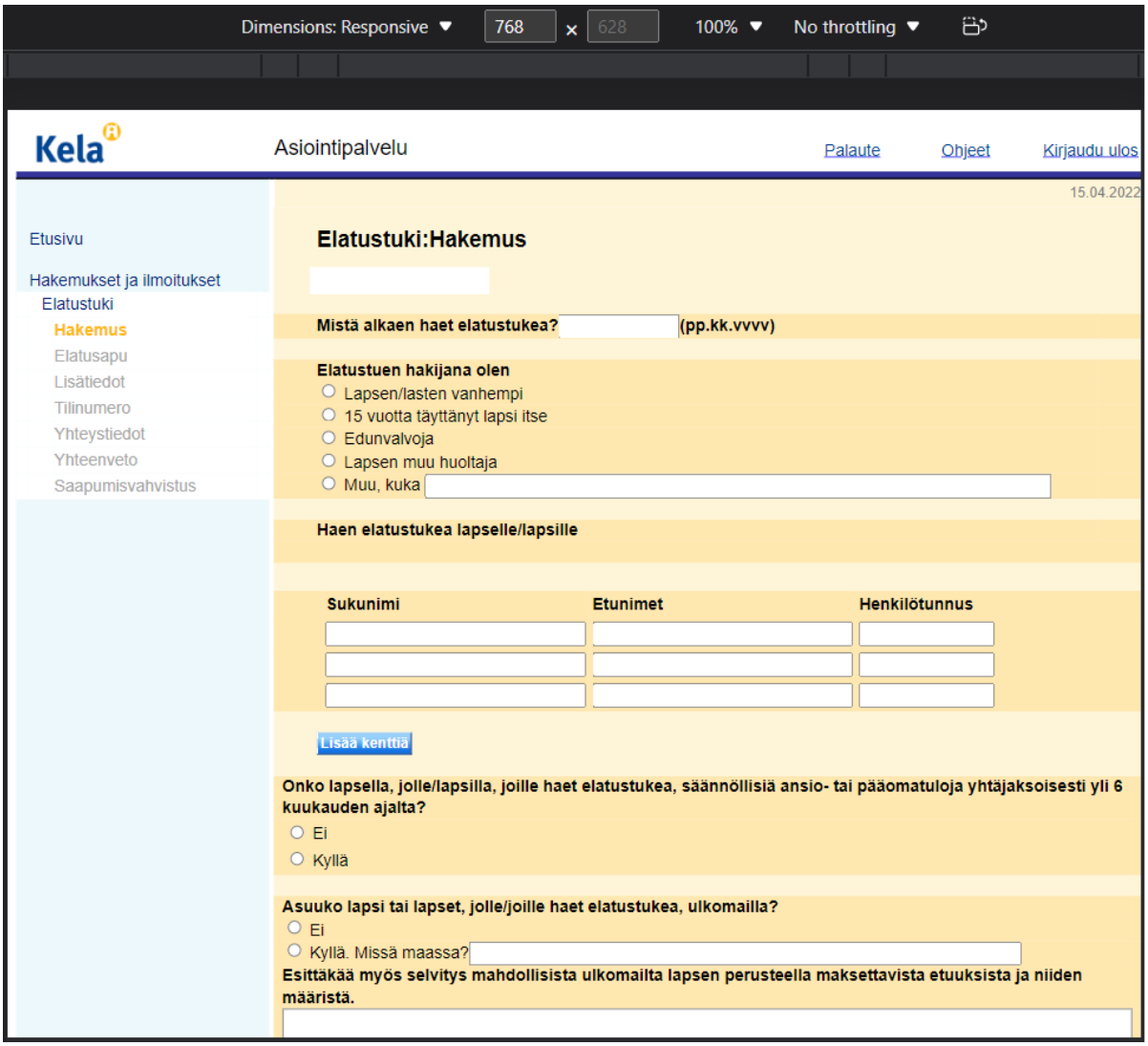
Oma työni sovellukseen liittyen on harjoittelun ja varsinaisen työurani aikana keskittynyt sovelluksen saavutettavuusmuutoksiin. Saavutettavuudella tarkoitetaan sitä, että sovellus kehitetään mahdollisimman monelle ihmisille käytettäväksi. Saavutettavan sovelluksen tulee olla helposti ymmärrettävä, helppokäyttöinen sekä myös tekniseltä tasolta hyvin ja virheettömästi tehty, jotta sivut toimivat eri laitteilla, eri näkymissä sekä avusteohjelmien, kuten ruudunlukijoiden ja puheohjausohjelmien kanssa (Aluehallintovirasto s.a.).

Saavutettavien sovellusten kuuluu seurata WCAG 2.1 eli Web Content Accessibility Guidelines-kriteerejä saavuttaakseen vaatimukset. Kriteerejä on hyvinkin paljon, ja tämän takia projekti on hiukan venähtänyt. Alun perin sovelluksen saavutettavuusmuutokset piti olla valmiina oman harjoittelujaksoni lopussa joulukuussa 2021, mutta sovelluksen koosta ja omista silloisista taidon puutteistani johtuen valmis sovellus menee näillä näkymin tuotantoon vasta toukokuussa 2022.



Kuva 3. Käyttöliittymä ennen saavutettavuusmuutoksia

Suurin muutos sovellukseen on kokonainen ulkonäköremontti. Kuvassa 3 näkyvä käyttöliittymä on ollut todella pitkään Kelalla käytössä, ja on myös eittämättä vanhentunut.

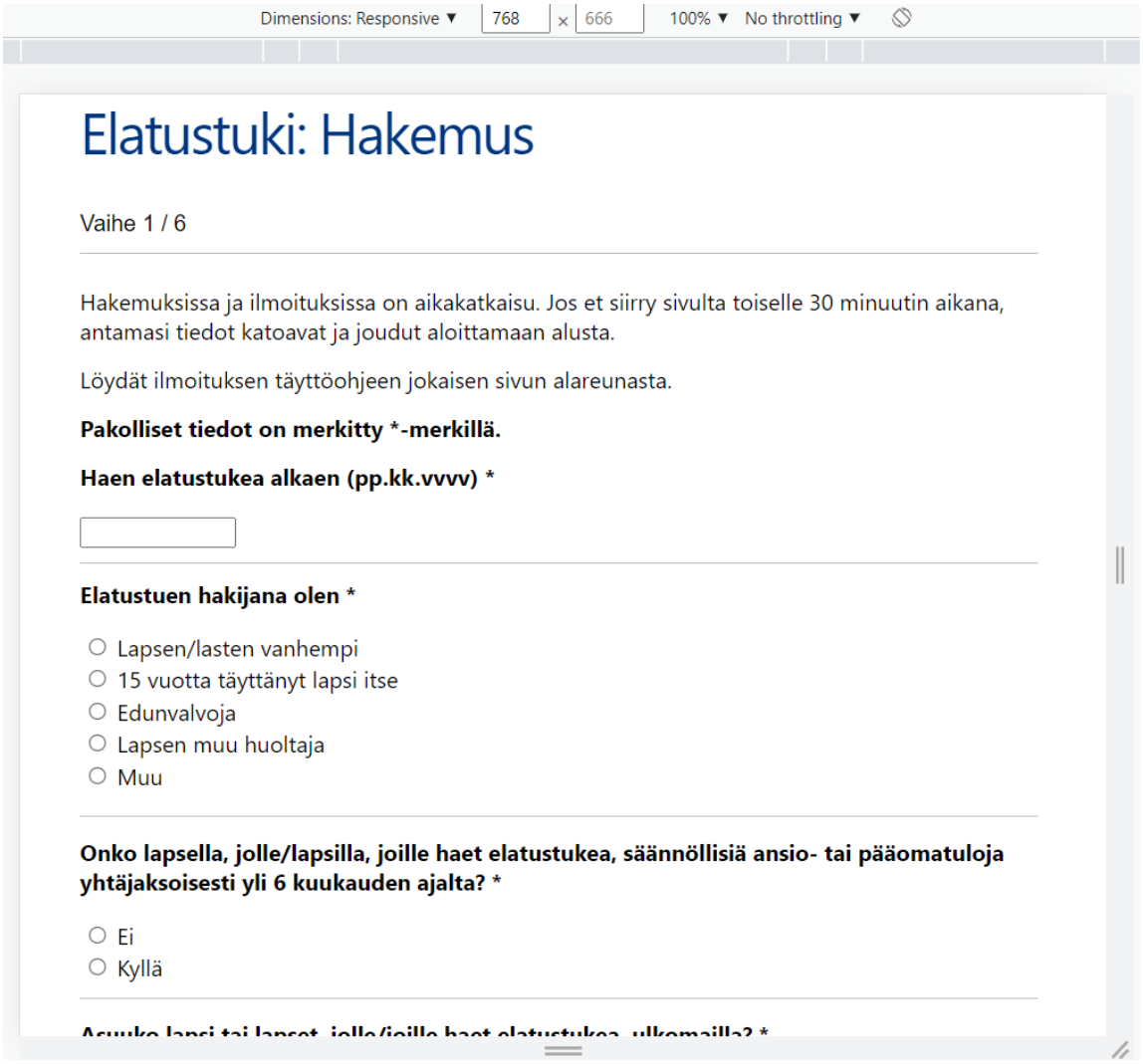


Kuva 4. Epäresponsiivinen käyttöjärjestelmä

Saavutettavien sovellusten yksi tärkeimmistä asioista on myös skaalautua responsiivisesti jokaiselle näyttöpäätteelle, näkymälle tai laitteelle. Sovelluksen vanha käyttöliittymä ei näin tehnyt, ja näytti käytännössä kaikilla laitteilla samalle, miltä työpöytänäkymässäkin (kuva 4). Pienillä näytöillä, kuten mobiililaitteilla, sovelluksesta tulee hyvin vaikeasti luettava ja vaikeasti täytettävä. Myöskään vanhan käyttöliittymän värimaailma ei ole kovin helppokäyttöinen. Keltaisen eri sävyt ja erisävyiset siniset korostusvärit voivat tuntua epämukavalta luettavalta. Responsiivisuusasiat testataan selaimen, tässä tapauksessa Chromen, kehittäjätyökaluilla.

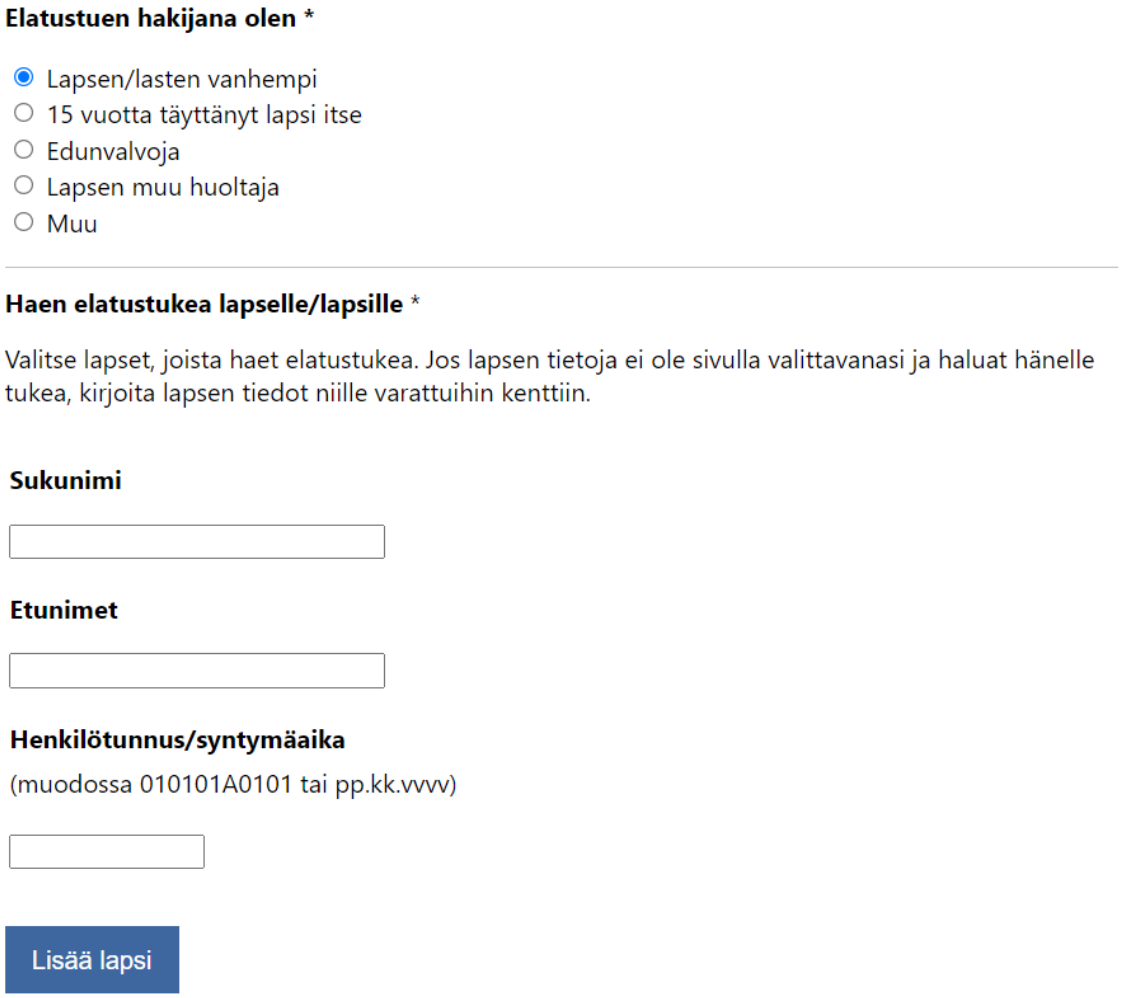
Kuva 5. Saavutettavuusmuutosten jälkeinen käyttöliittymä

Saavutettavuusmuutosten myötä sovelluksen värimaailma on täysin vaihdettu vanhasta keltaisen eri sävyistä, kuten kuvasta 5 voidaan nähdä. Nykyään sovellus on neutraali ja helppolukuinen. Mustaa tekstiä valkoisella pohjalla ja saman sävyisellä sinisellä korostusvärillä.



Kuva 6. Responsiivinen käyttöliittymä

Sovelluksesta on myös tehty responsiivinen (kuva 6). Sivut skaalautuvat oikein jokaiselle laitteelle näytön koosta riippumatta, eikä niihin tule horisontaalista skrollausta. Osa elementeistä vaihtaa myös paikkaa, kun näytön resoluutio pienenee. Navigaatiopalkki siirtyy 768:n tai vähemmän levyisillä näytöillä sovelluksen yläosaan, mikä helpottaa sovelluksen luettavuutta.



Kuva 7. Radiobutton-valinnan yhteydessä aukeava elementti

Muutosten yhteydessä käyttäjiltä on myös piilotettu aluksi turhia elementtiä. Yksi näistä on hakemus -näytöllä oleva elementti, johon syötetään lapsen tiedot, jolle tukea haetaan. Elementti aukeaa kaikilla muilla valinnoilla paitsi, jos hakijana on yli 15-vuotias lapsi itse (kuva 7).

```
<script>
    window.addEventListener("load", function() {
        avaaMaa('subview4:lomake:radioAsuukoLapsiUlkomailla');
        avaaMuu('subview4:lomake:radioElatustuenHakijanaOlen');
        avaaHaenElatustukeaLapsille('subview4:lomake:radioElatustuenHakijanaOlen');
    });
</script>
```

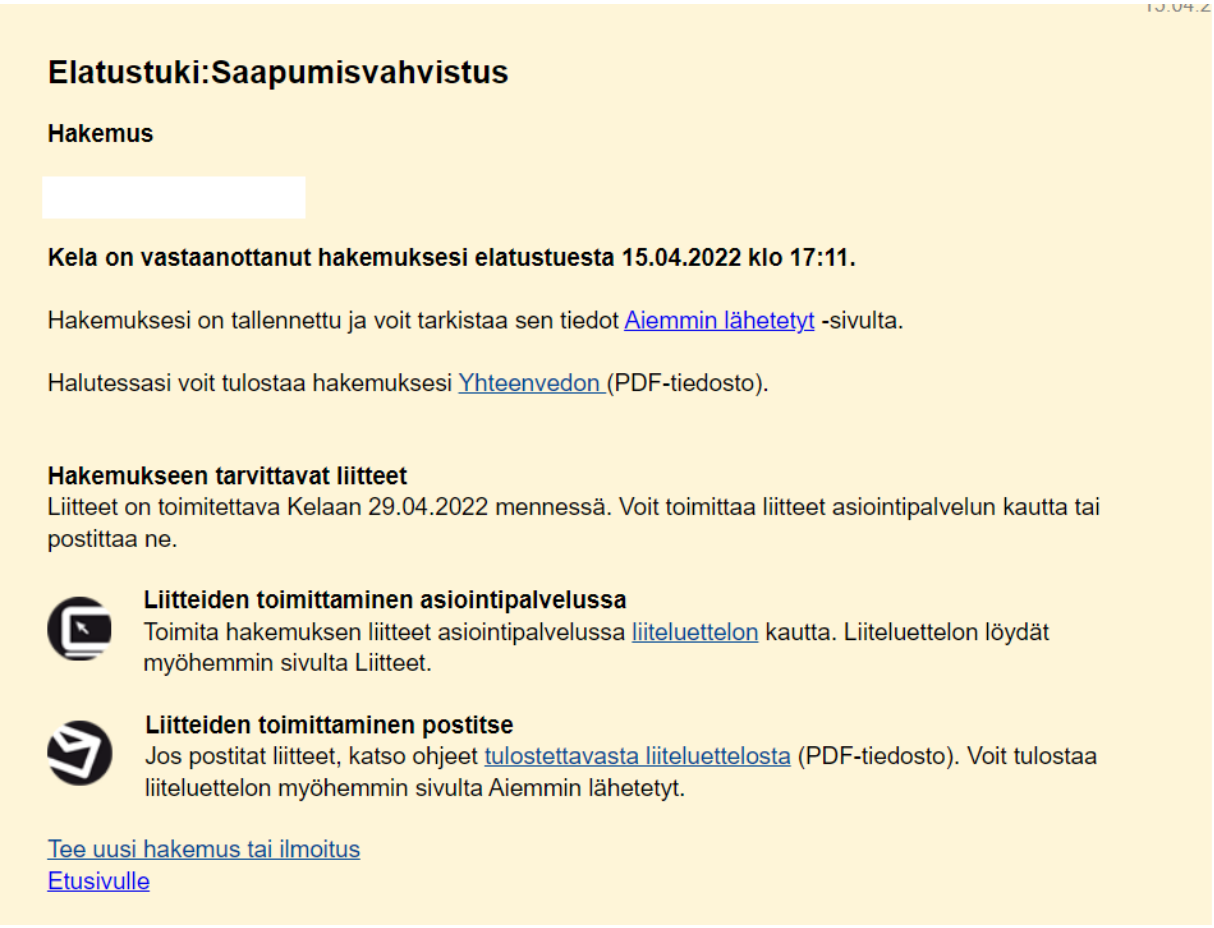
Kuva 8. Valinnasta riippuvan elementin aukeamisen kuuntelija

Koodissa (kuva 8) asia on implementoitu niin, että hakemus -sivulle on asetettu addEventListener -metodin avulla kuuntelija, joka lataa funktiosta riippuen oikean elementin.

```
function avaaHaenElatustukeaLapsille(radionappiId) {
    var varradionappiId = document.getElementsByName(radionappiId);
    var haenElatustukeaLapsille = document.getElementById("haenElatustukeaLapset_div");

    for(i = 0; i < varradionappiId.length; i++) {

        if(varradionappiId[i].checked) {
            var apu = varradionappiId[i].value;

            if(apu != "L"){
                haenElatustukeaLapsille.style.display = "block";
            }else{
                haenElatustukeaLapsille.style.display = "none";
            }
        }
    }
}
```

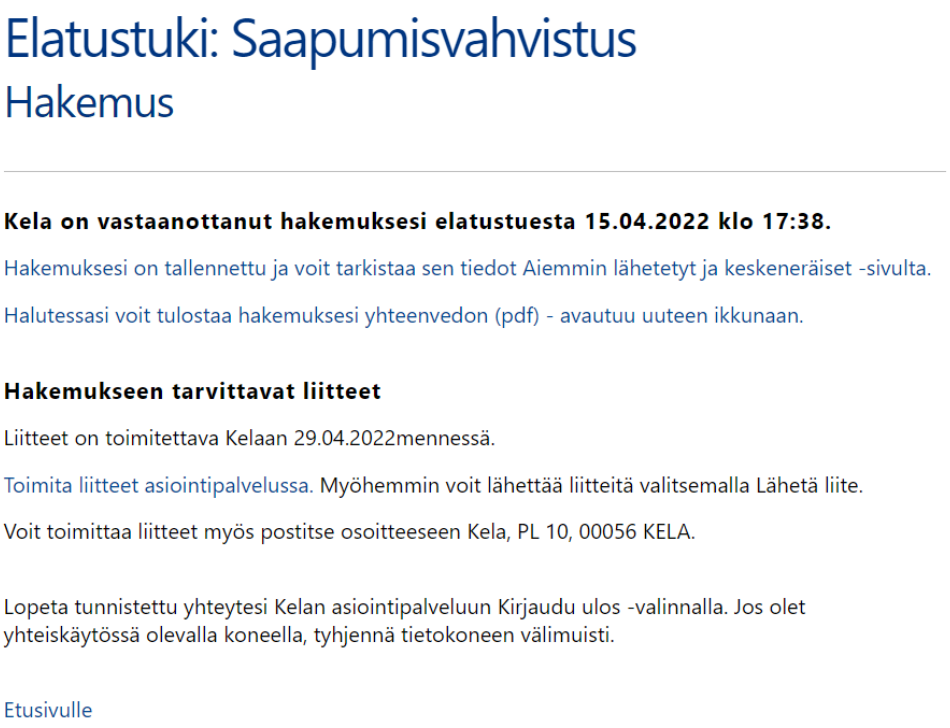
Kuva 9. Funktio, jolla valinnasta riippuva elementti aukeaa

Erillisessä Javascript -tiedostossa on määritetty, mitä funktiot tekevät. Kyseisessä funktiossa (kuva 9) valintanapin id on otettu document.getElementsby-Name-metodilla, minkä jälkeen for-lauseella tarkistetaan, että funktiota varten keksitty laskuri eli i on 0. Laskuri i saa isomman arvon, kun nappia painetaan. Tämä on koodattu muuttujalla apu. Jos apumuuttujan arvo on L, sovellus vaihtaa piilotetun elementin tyylin tyhjästä tyyliin block, eli näkyväksi. Samanlaisia funktioita on aseteltu ympäri sovellusta eri elementeille tilanteissa, joissa käyttäjälle ei haluta näyttää hänelle tarpeettomia elementtejä.



Kuva 10. Saapumisvahvistus-sivu ennen muutoksia

Saavutettavuutta varten osaa sivuista piti myös selventää. Hakemuksen tai muutosilmoituksen lähettämisen jälkeen näkyvällä saapumisvahvistus -sivulla (kuva 10) oli tekstejä, joihin oli sijoitettu linkit yhteen pieneen osaan lausetta, vaikka syytä vain osan tekemisestä linkiksi ei varsinaisesti ollut. Sivulla oli myös turhia kuvia, mikä voi vaikeuttaa lukemista sekä saattaa aiheuttaa ongelmia ruudunlukijaohjelmistojen kanssa.

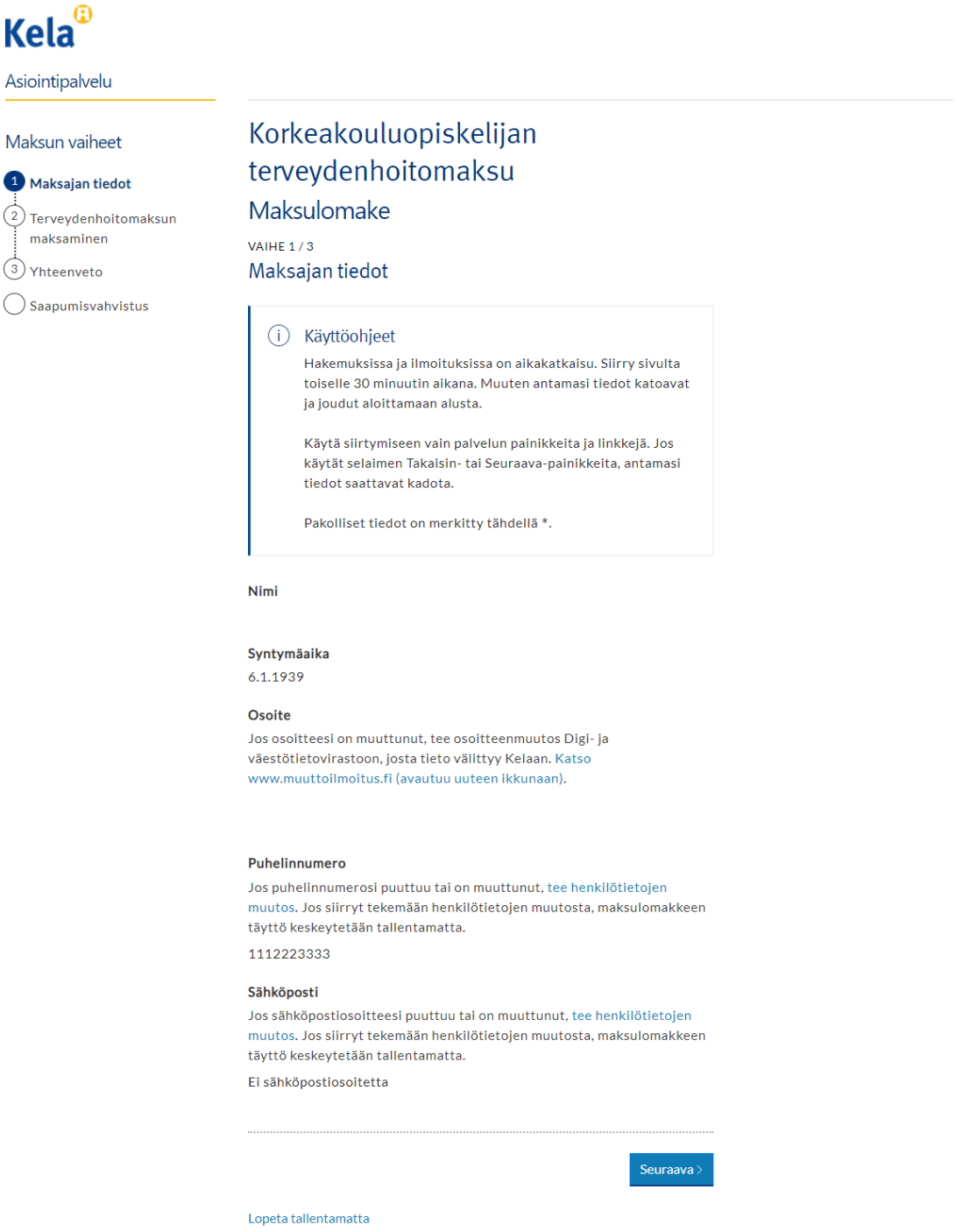


Kuva 11. Saapumisvahvistus-sivu muutosten jälkeen

Sivua selkeytettiin asettamalla linkit koko lauseen pituiseksi sekä poistamalla kuvat (kuva 11). Myös vanhentuneita tekstejä, kuten ohjeet tulostettavasta liiteluettelosta on poistettu. Liitteiden tulostaminen postitse lähettämistä varten tapahtuu nykyään Kelan asiointipalvelun kautta, jonne yksi linkeistä ohjaa jo valmiiksi.

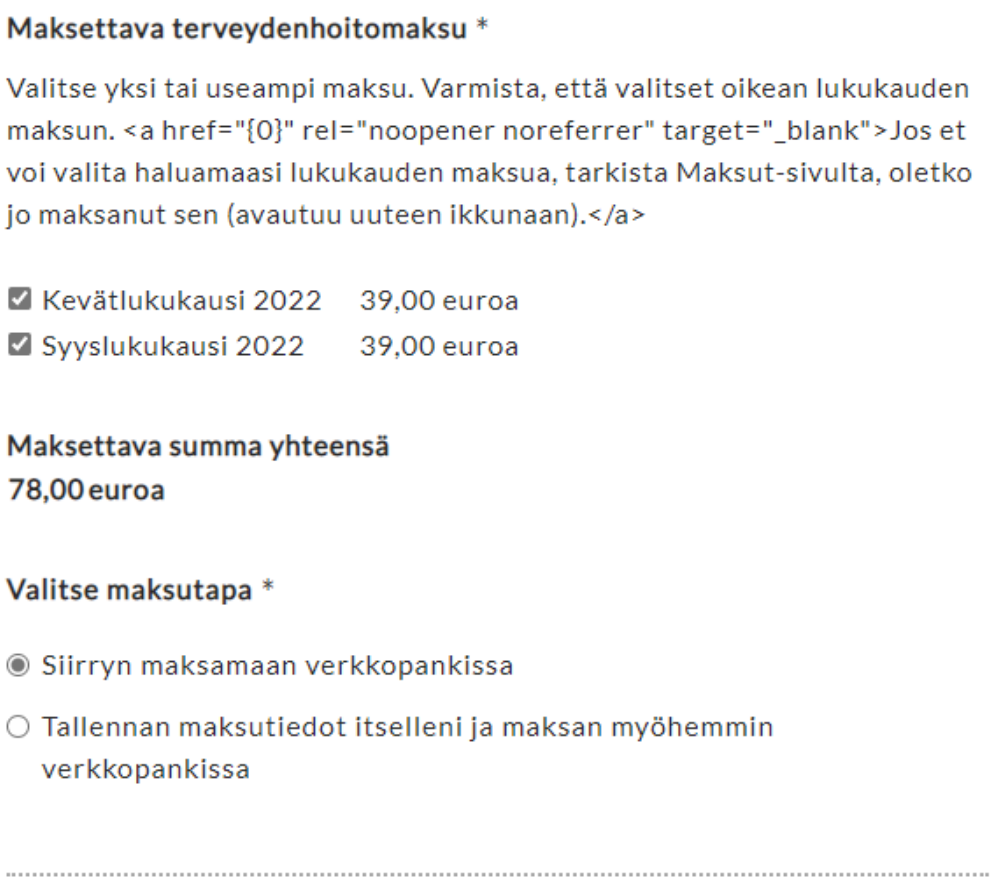
3.2 Opiskelijoiden terveydenhuoltomaksu-sovellus

Opiskelijoiden terveydenhuoltomaksu-sovellus (kuva 12) toimii Spring Bootin ja Thymeleafin yhdistelmästackillä. Sovellusta voi pyörittää lokaalissa testiympäristössä Spring Bootin avulla ja testiympäristöihin sovellus käännetään OpenShift-sovellusalustalla.



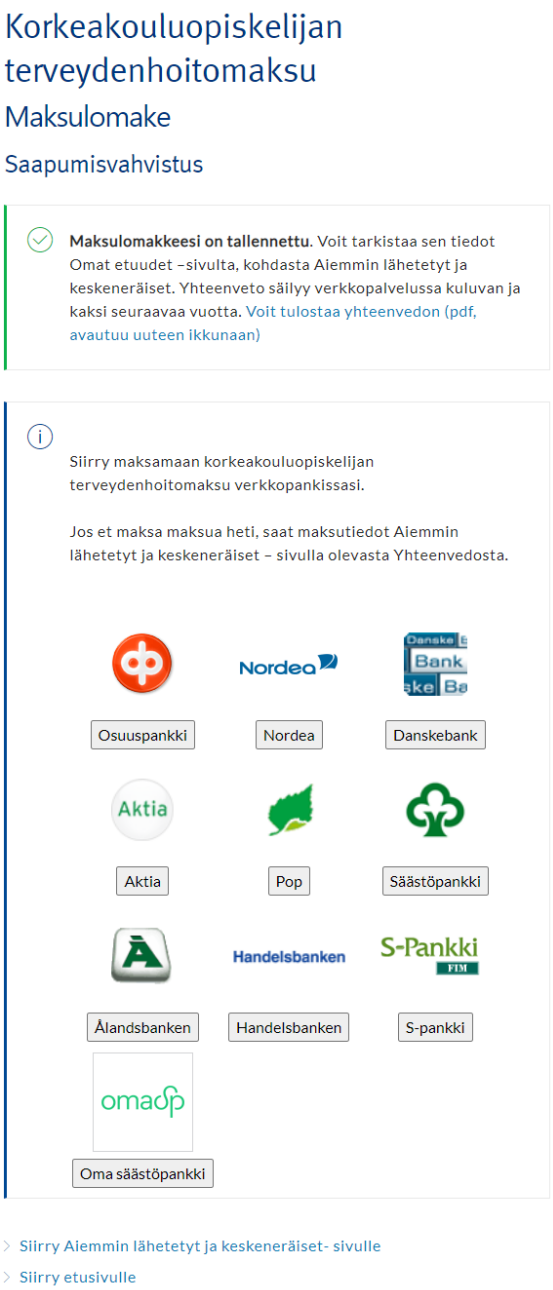
Kuva 12. Käyttöliittymäesimerkki ja navigaattorin demonstraatio

Sovellus on huomattavasti pienempi kooltaan, kuin elatustuen sovellus. Navigaattori sisältää vain 4 sivua, jotka ovat sisällöltään melko pieniä. Koko sovellus sisältää yhteensä 3 pakollista valintaruutua (kuva 13).



Kuva 13. Opiskelijoiden terveydenhuoltomaksu-sovelluksen pakolliset valintaruudut

Valintaruuduilla valitaan, mitkä lukukaudet maksetaan ja halutaanko maksu maksaa suoraan verkkopankin kautta, vai tulostettavan maksulomake-PDF:n kautta myöhemmin. Jos sovelluksella siirrytään maksamaan verkkopankissa, mennään sinne yhteenveto -sivun kautta ja siirrytään tämän jälkeen saapumisvahvistus-sivulle (kuva 14).



Kuva 14. Saapumisvahvistus-sivun näkymä

Saapumisvahvistus-sivulla ilmoitetaan, että maksulomake on tallennettu, jotta sen voi käydä tekemässä uudelleen siltä varalta, että esimerkiksi internet-yhteys katkeaa kesken kaiken. Maksu pankeille hoidetaan pankkikohtaisesti. Sivulta löytyy maksupainikkeet kaikille Suomen pankeille.

```
<div id="pankkipainikkeet" class="bd-callout au-bd-callout-info"
    th:if="${terveydenhoitomaksu.maksutapa.equals('1')}" hidden="true">
    <span class="au-bd-callout-icon" aria-hidden="true"></span>
    <div class="au-bd-callout-body"
        th:switch="${@applicationWebConstant.getApplicationType().toString()}">

        <p th:utext="#{296410}" />

        <div class="pankkipainikeDiv">

            <div th:replace="~{${localFragment + 'op_form'} :: op_form}" >
            </div>

            <div th:replace="~{${localFragment + 'nordea_form'} :: nordea_form}" >
            </div>

            <div th:replace="~{${localFragment + 'danske_form'} :: danske_form}" >
            </div>

            <div th:replace="~{${localFragment + 'aktia_form'} :: aktia_form}" >
            </div>

            <div th:replace="~{${localFragment + 'pop_form'} :: pop_form}" >
            </div>

            <div th:replace="~{${localFragment + 'saastopankki_form'} :: saastopankki_form}" >
            </div>
```

Kuva 15. Esimerkki koodista, johon pankkipainikkeet ovat sijoitettu

Pankkipainikkeiden sijoittaminen on toteutettu Thymeleafin fragment-ominaisuuden avulla (kuva 15). Fragmentit ovat osa mallia ja niitä voi sijoittaa eri malleihin (Frontbackend, 2020). Käytännössä sivut toimivat koodia siistivinä osina. Sovelluksen esimerkissä pankkipainikkeet ovat sijoitettu fragmenteilla ja fragmentien sivurakenne itsessään sijaitsee eri tiedostossa.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" lang="fi"
    layout:decorate="~{'classpath:/layout/layout-common'}">
<head>
 <meta charset="UTF-8" th:if="${templating != null}">
</head>
  <body>
    <div class="pankkipainikeDivChild" th:fragment="op_form">
                    <p/>
                    <table border='0' cellspacing='0' cellpadding='0' align='center'
                    <td align='center' valign='center' height='60'>
                    <input class='pankkipainike' onclick="ohjaaOP('op')" type='image'
                    </td> </tr> <tr> <td align='center' valign='top'>
                    <input type='submit' onclick="ohjaaOP('op')" value='Osuuspankki'
                    <p/>
    </div>


  </body>
</html>
```

Kuva 16. Fragmentin toteutus

Fragment-tiedosto on .html-tiedosto (kuva 16), johon voidaan määritellä kaikki samat osat, mitkä muihinkin .html-tiedostoihin. Tässä tapauksessa fragment-

tiedosto sisältää lomakkeen, jonka sisälle on toteutettu Osuuspankin pankkipainike. Input-luokka, joka sisältää sekä klikattavan kuvan, että napin, toimii onClick-ominaisuudella. onClick ohjaa napin tai kuvan painalluksella käyttäjän Osuuspankin omaan maksupalveluun.

Oma työni tähän sovellukseen seurantajakson ajalta tulee olemaan Osuuspankin monipankkitoiminnallisuuden implementointi. Tällä hetkellä jokainen pankkipainike johtaa suoraan valittavan pankin maksupalveluun. Tekemieni muutosten jälkeen jokainen pankkipainike ohjautuu ensin Osuuspankin monipankkitoiminnallisuus-rajapintaan, joka vastaavasti ohjaa sen eteenpäin pankin maksusivulle.

4 PÄIVÄKIRJA

Päiväkirjaosuuden aikana seuraan ja kirjaan ylös omia työtehtäviäni sovelluskehittäjänä. Haluan työseurannan ajanjaksolta kirjata omat työtehtäväni ylös konkreettisemmin, kuin mitä pelkästään Agile boardilla (tehtävälistalla) saa seurattua. Toinen tavoite on seurata omaa kehittymistäni työntekijänä näinkin lyhyellä ajanjaksolla. Oletan kuitenkin, että nelisen viikkoa riittää hyvin oman kehittymisen seuraamiseen. Pidän päiväkirjamallista työtä itselleni sopivana sekä rajoitetusta ajasta johtuen että mielenkiintoisen toteutuksen takia. Myös työnantaja ehdotti aihetta. Päiväkirjaluvun aikana työn teksti vaihtuu akateemisesta tekstistä enemmän blogityyliseen tekstiin.

4.1 Viikko 1

Tällä viikolla luvassa tulee olemaan enimmäkseen opiskelijoiden terveydenhuoltomaksu-sovelluksen monipankkitoiminnallisuuden toteuttamista sekä Kelan järjestämää Spring Boot -koulutusta. En usko, että saan omin avuin monipankkitoiminnallisuutta toimimaan, kun olen kyseisen sovelluksen tietotaidon kanssa vielä todella alkutekijöissäni.

Maanantai 28.3.2022
Päivä meni opiskelijoiden terveydenhuoltomaksu-sovelluksen monipankkitoiminnallisuusmuutokseen ja siihen vaadittavien töiden tutkimisessa. Loppupäivästä pääsin hyvään ymmärrykseen, että mistä on kyse, mutta pääsen tätä kokeilemaan vasta loppuviikosta.

Tiistai 29.3.2022

Tänään koko päivä meni Kelan sisäisessä Spring Boot -koulutuksessa, jossa käytiin läpi Spring Bootin toiminta ihan tyhjän Spring -projektin luomisesta lähtien. Kurssi on kaksiosainen, joten tänään jäätiin virheiden käsittelyyn ja niiden näyttämiseen projektissa. Koulutuksen ohessa pääsin myös tekemään harjoituksia, joihin kuului Hello Worldin toteuttaminen, alkukantainen CRUD (Create, Read, Update, Delete) -pohjainen sovellus sekä virheiden käsittelyn lisääminen sovellukseen.

Keskiviikko 30.3.2022

Päivä meni myös Spring Boot -koulutuksessa. Tänään jatkettiin siitä, mihin eilen jäätiin. Päivän ohjelmassa oli opetusta siitä, kuinka sovelluksen sisällä kutsutaan APIa eli ohjelmointirajapintaa, sovelluksen yksikkötestausta sekä SQL-tietokantojen lisäämistä ja testaamista sovelluksessa. Kurssi tuli omasta mielestä hyvin tarpeeseen, koska olen vielä aika Spring Boot -aloittelija.

Torstai 31.3.2022

Päivä kului opiskelijoiden terveydenhuoltomaksu-sovelluksen parissa, jotta saisin Osuuspankin monipankkitoiminnallisuuden toimimaan. Työtä varten alustin Thymeleafin pankkikohtaiset fragmentit siten, että klikattaessa pankkinappia nappi ei vie enää kaikkia tietoja pankkikohtaisesti, vaan ohjaa pelkästään määrittelemäni pankki-ID:t URL-parametrina Osuuspankin maksupalveluun ohjaavaan funktioon. Monipankkitoiminnallisuuden testaamista paikallisella palvelimella varten piti lisätä sovelluksen .properties -tiedostoon uusia ympäristömuuttujia, joihin kuului pankkikohtaiset ID:t ja uusi URL johon monipankkitoiminnallisuus ohjaa.

Sovelluksen ns. ”controllerin” koodiin täytyi tehdä erillinen funktio, joka hakee monipankkitoiminnallisuuden URL:n kaikkien muiden pankkien napeilla, paitsi Osuuspankin napilla. Funktio toimi niin, että funktio ottaa parametrinä pankki-ID:n, joka tulee klikattaessa pankin nappia. Tämä yhdistetään uuteen ohjattavaan URL:ään. Pankki-ID:t saatiin funktioon siten, että sivun uudelleenohjausfunktiossa käytin Spring Bootin @RequestParam-ominaisuutta, jotta ID:t haetaan napin painalluksella koodiin.

Näin saatiin sovellus toimimaan testiympäristöissä lokaalisti. Huomenna homma jatkuu sillä, että sovellus laitetaan toimimaan myös Kelan sisäisen intran testipalvelimella.

Perjantai 01.04.2022
Tänään työpäivä alkoi kokonaisarkkitehtuurin eli KARKKI-seminaarilla. Näitä pidetään melkein joka perjantaina, ja tällä kertaa aihe liittyi sosiaaliturvauudistukseen ja digitalisaatioon. Ei ollut niin mielenkiintoinen mitä yleensä, mutta näillä lähtee hyvin aina perjantait käyntiin.

Seminaarin jälkeen jatkui työt terveydenhuoltomaksu-sovelluksen parissa. Sovelluksen testaamista intran testipalvelimella piti alustaa siten, että eilen remontoituun controlleriin piti tehdä funktio, joka tunnistaa onko sovellus lokaalissa testauksessa vai netissä. Se onnistui suhteellisen yksinkertaisella if-else lausekkeella, jossa testataan, onko sovellus lokaali vai ei, ja sitä myöten annetaan asiaan kuuluvat parametrit pankki-ID:n ja URL:n muodossa. Näin saatiin lähes kaikki koodimuutokset valmiiksi lukuun ottamatta muutaman testiparametrin poistoa.

Koodimuutosten jälkeen siirryttiin lokaaliin testaukseen, joka oli käytännössä sitä, että käydään jokainen pankkimaksu läpi ja katsotaan palauttaako sovellus oikeat parametrit ja pankki-ID:t. Tämä tehtiin system.out-komentojen eli lokittamisen avulla. Testaus meni hyvin läpi lokaalisti, minkä jälkeen sovellus piti saada OpenShift-konttialustan avulla testipalvelimelle.

Tämä oli ensimmäinen kerta, kun pääsin itse OpenShiftiä käyttämään. Prosessi sujui työkaverin avulla ihan hyvin. Sovelluksen koonti onnistui muokkaamalla OpenShiftin omia parametrejä, eli asettamalla BitBucketin oikea kehityshaara mistä sovelluksen versio kootaan ja palvelin, jolle sovellus halutaan asentaa. Sovellus kääntyi palvelimelle hyvin, mutta itse pankkimaksut eivät toimineetkaan. Tähän oli pienen ihmettelyn jälkeen helppo ratkaisu, eli URL-muodostajafunktiosta puuttui testipalvelimen URL:tä parametrit. Tämän jälkeen myös testipalvelimella kaikki maksut menivät onnistuneesti läpi. Päivän lopuksi poistin OpenShiftin omasta secretistä (pelkästään OpenShiftissä sijaitsevassa tiedostossa, jota ei ole fyysisenä tiedostona) vanhat käyttämättömät pankkimuuttujat, ja lisäsin uudet muuttujat tilalle.

Viikon 1 analyysi

Ensimmäinen kirjattava viikko oli kieltämättä yksi parhaimmista työviikoista mikä hetkeen on ollut. Sain sisäistettyä ja opittua paljon uutta asiaa Spring Boot-kurssin avulla sekä myös työstä monipankkitoiminnallisuuden parissa. Tästä kiitos myös työkavereille. Olisin hakannut päätäni seinään huomattavasti pidempään ellen olisi konsultoinut apua. Ensi viikolla siirryn taas tutumpaan ympäristöön, eli elatustuki -sovelluksen saavutettavuusmuutosten virhekorjausten pariin. On myös sprintin viimeinen viikko, joten koitan saada oman agile boardini mahdollisimman tyhjäksi.

4.2 Viikko 2

Toinen viikko jatketaan todennäköisesti elatustuen virheiden parissa, joita vielä jonkin verran tulee. Vähän huolestuttaa näin pääsiäisen ja muiden pääsiäislomien lähentyessä, että mitenhän itselläni riittää työt seuraaville muutamille viikoille. Ei se tietenkään haittaa, ettei tarvitse koko ajan hiki päässä olla suorittamassa, mutta toivottavasti ei tarvitse ihan toimettomana olla.

Maanantai 4.4.2022

Päivä kului elatustukisovelluksen parissa. Omalle kohdalle sattui kaksi todella vaikeaa virhettä, jotka liittyivät .css-tyylitiedostoon. Ensimmäinen virhe oli, että sovellus ei skaalaudu responsiivisesti mobiililaitteilla tai työpöytänäkymässä, mikä johtaa sivun horisontaaliseen skrollaukseen. Saavutettavuusmuutosten määrittelyissä mainitaan, että sivulla saa olla pelkästään vertikaalista TAI horisontaalista skrollausta, eikä molempia. Toinen virhe on taas sellainen, että ruudunlukijalla tai näppäimistöllä selaamalla uloskirjautumisnapin päälle keskittäessä ei napin ympärille tule kehystä ollenkaan, mikä pitäisi myös määrittelyiden mukaan piirtyä jokaisen painettavan elementin ympärille. Kumpaakaan virhe en saanut tänään vielä ratkaistua, mutta tuli päivän aikana aloitettua isomman puoleinen .css-remontti. Olin työharjoittelun alkuaikana jättänyt .css-tiedoston kieltämättä aivan hirveään kuntoon. Paljon duplikaattityylejä, responsiivisuustyylit ympäri tiedostoa, eikä nätisti peräkkäin ja myös täysin tyhjiä elementtityylejä. Sain ihan hyvin päivän aikana järjesteltyä tiedostoa, mutta tästä riittää varmaan koko loppuviikoksi tekemistä.

Tiistai 5.4.2022

Elatustuen parissa jatkettiin tänäänkin. Tällä kertaa heti aamupäivästä sain responsiviisuusvirheen ratkaistua yhden toisen entisen harjoittelijan ja nykyisen työntekijän vinkillä. Hänellä oli ollut oman sovelluksen saavutettavuusmuutosten parissa täysin sama ongelma ja hän vinkkasi, että ongelma johtuu responsiivisuustyylien väärästä leveydestä. Osittainen ratkaisu olikin siis siinä, että ruudut, jotka ovat maksimissaan 1020-pikselin levyisiä skaalasivat elementit 99 % :n levyisiksi. Vaihdettuani näiden leveydeksi 90 %, horisontaalinen skrollaus poistui työpöytänäkymästä isommilta mobiililaitteilta, mutta pienemmillä mobiililaitteilla horisontaalisuus säilyi. Parin testityylin lisäämisen jälkeen huomasin, että tekstisyöttökentät leviävät ns. ruudulta pois muutaman pikselin verran, mikä aiheutti horisontaalisuuden. Tämän ratkaisin poistamalla syöttökentistä kovakoodatut leveydet ja lisäsin kenttiä varten tyylin, joka pienentää niitä 80 %:n maksimissaan 576 pikselin levyisillä näytöillä. Kokoustettiin myös lounaan jälkeen elatustuen tuoteomistajan kanssa siitä, missä tilanteessa sovellus on tällä hetkellä tekeillä. Loppupäivän jatkoin .css -tiedoston uudelleenjärjestelyä.

Keskiviikko 6.4.2022

Tänään oli aika toimeton päivä, ei omasta panoksesta johtuen, vaan työn puutteesta. Elatustuen sovelluksen testaaja oli toista päivää kipeänä, joten virherintamalla oli tyhjää, eikä omalla Agile boardillanikaan ollut enää virheitä ratkottavaksi. Päätin sitten, että jatkan puolet päivää .css -tiedoston järjestelyä ja toisen puolen käytän oman osaamiseni laajentamiseen. Jokaiselle jaettiin tammikuussa työaikaa 30 tuntia osaamisen laajentamista varten, johon kuuluu itseopiskelu sekä kurssit ja koulutukset. Katsoin siis puolet päivästä erään työkaverin viime vuonna pitämän OpenShift-kurssin dioja ja tallenteita. Jos testaaja ei pääse huomennakaan paikalle, jatkan OpenShift-materiaalit loppuun, jos kaikkien kokousten ohella ehtii.

Torstai 7.4.2022

Tänään jatkui taas melkein täysin toimettomana päivä, kun testaaja oli vielä kipeänä. Alkupäivästä jatkoin .css -tiedoston järjestelyn loppuun asti. Nyt se on hyvässä luettavassa kunnossa. Sen jälkeen lähes koko loppupäivä oli erinäisiä palavereita. Kävin muun muassa kuuntelemassa ja katsomassa opintotuen verkkohakemuksen muutoksien demoa sekä opinto- ja perhe-etuusryhmän

ryhmäpalaveria, jossa käytiin ajankohtaisia asioita läpi. Päivän lopulla pidettiin meidän ryhmämme retro, jossa käytiin läpi Agile boardille tehtäviä ”Definition of Ready” -muutoksia. Kerkesin myös työpäivän aikana käymään OpenShift-kurssin materiaalit kokonaan lävitse.

Perjantai 8.4.2022

Tänään pääsi jatkamaan oikeita töitä, kun testaaja oli palannut takaisin töihin. Korjasin siis alkupäivästä muutaman elatustuen hakemuksesta löytyneen pienen virheen liittyen teksteissä oleviin puuttuviin pakollisuusmerkintöihin.

```
<h:outputText value="#{tekstit.ulkomaanpankkiyhteys_BIC}">
</h:outputText>
<span aria-hidden="true" title="Pakollinen" class="musta">*</span>
<span class="sr-only">Pakollinen</span>
```

Kuva 17. Pakollisiin kenttiin lisättävä merkintä

Kaikki pakolliset kentät pitää merkitä tähtimerkinnällä, joka näkyy vaan visuaalisesti eli ruudunlukijaohjelmat eivät niitä lue. Tämä mahdollistetaan aria-hiddenillä, joka piilottaa tähden ruudunlukijalta, ja sen sijaan luetaan sr-only luokkaan tehty ”Pakollinen”-teksti (kuva 17).

Virheiden korjaamisen jälkeen siirryttiin ryhmän sprintin katselmointiin ja seuraavan sprintin suunniteluun. Omalta kohdaltani palaverin lopuksi tultiin päätökseen, että jatkan elatustuen virheiden ratkomista niin kauan, kun niitä riittää, eli toivottavasti suurin osa sprinttiä. Omasta ryhmästä jää ensi viikon alussa niin sanottu mentori ja muita kokeneimpia toteuttajia pääsiäislomalle, joten ei varsinaisesti löydy seuraavaan viikkoon tai kahteen ketään perehdyttämään minua uusiin ohjelmistoihin.

Työpäivän lopuksi vielä opiskelijoiden terveydenhuoltomaksu-sovelluksen testaaja otti yhteyttä, ja pyysi minua korjaamaan yhdestä pankkinappulasta löytyneen pienen virheen. S-pankin napissa luki Spankki, joka ei estä sovelluksen käyttöä, mutta saattaisi ärsyttää osaa käyttäjistä. Kävin siis nopeasti korjaamassa sen ja OpenShiftissä käänsin sovelluksen testipalvelimille ennen työpäivän loppua.

Viikon 2 analyysi

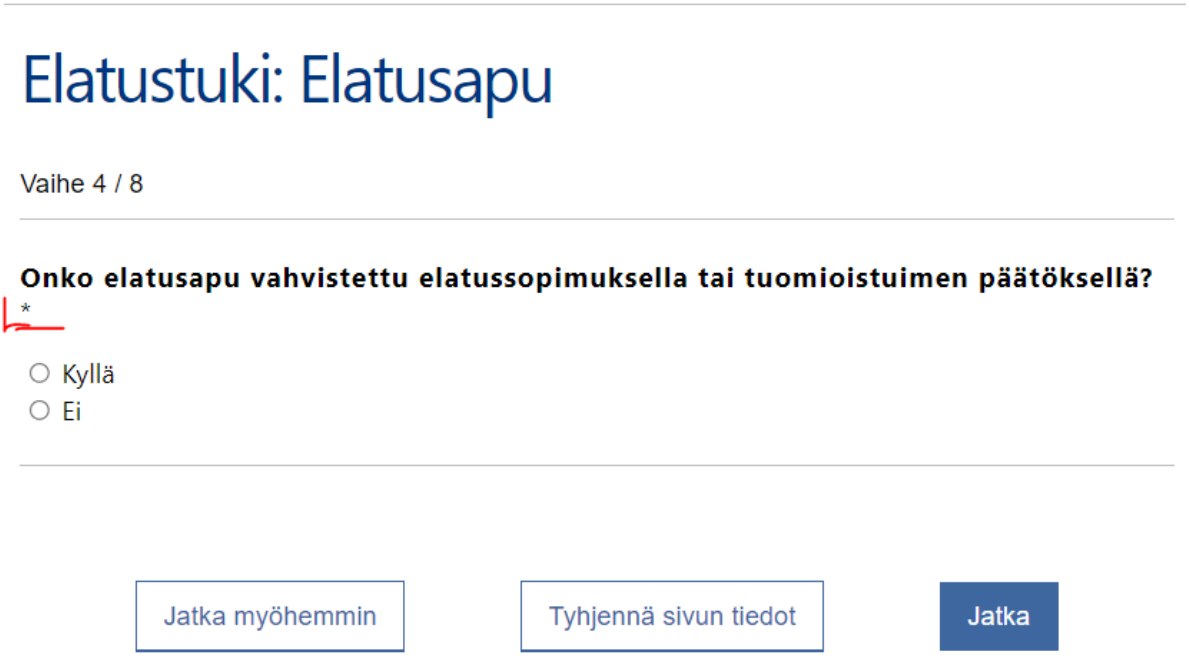
Toinen seurantaviikko ei mennyt ihan niin toimeliaasti, kun mitä oletin. Tekeminen loppui äkkiä kesken elatustuen sovelluksen testaajan jäädessä sairaslomalle. Ei viikko siltikään kylmäksi jättänyt, oli opettava kokemus päästä tekemään työtä responsiivisuuden parissa ja .css -tiedoston järjestely auttaa varmasti muita tekijöitä tekemään muutoksia nopeammin tulevaisuudessa, kun tiedosto ei ole enää niin hirveässä kunnossa. OpenShift -kurssi tuli myös tarpeeseen, kun joudun todennäköisesti seuraavan parin viikon aikana itsenäisesti pyörittelemään sovelluksia OpenShiftissä.

4.3 Viikko 3

Kolmas seurantaviikko tulee olemaan vähän lyhyempi. Pääsiäisen takia torstai on lyhyempi työpäivä ja pitkäperjantai tietysti on vapaana. Työn kuva on suurimmaksi osaksi elatustuen virheiden korjaamista ja mahdollisesti pari kappaletta opiskelijoiden terveydenhuoltomaksu -sovellukseen liittyvää virhettä.

Maanantai 11.4.2022

Maanantai alkoi Agile boardin tarkistamisella, että oliko perjantai -iltapäivästä oman lähdön jälkeen tullut lisää virheitä korjattavaksi. Kaksi virhettä oli ennen viikonloppua tullut, joten otin ne itselleni päivän tekemiseksi. Suurin osa päivästä kuluikin yhden asetteluvirheen ratkomiseen.



Kuva 18. Pakollisuusmerkintä rivittyy väärin

Elementin pakollisuusmerkintä kuuluisi rivittyä heti tekstin perään, eikä rivinvaihdon omaisesti elementin alle kuten kuvassa 18. Pienen testailun jälkeen tämä näytti tapahtuvan vain näytöillä, jotka ovat yli 1020:n pikselin levyisiä. Oletin siis alkujaan, että tämä on responsiivisuustyyleistä kiinni ja kävin niihin käsiksi. Pienen tuskailun jälkeen oli pakko tulla siihen tulokseen, että tämä on jostain muusta johtuvaa. Huomasinkin, että tyyleissä oli otsikkoa varten oma tyyli, jonka fonttikoko oli 1.05em. Pienensin sen arvoon 1.04em:n ja pakollisuusmerkintä rupesi rivittymään oikein.

```
.labelotsikko {
    font-weight: 700;
    font-style: normal;
    font-size: 1.05em;
    margin-bottom: .4rem;
    font-weight: 700;
    letter-spacing: .05rem;
}
.labelotsikkoalt {
    font-weight: 700;
    font-style: normal;
    font-size: 1.04em;
    margin-bottom: .4rem;
    font-weight: 700;
    letter-spacing: .05rem;
}
```

Kuva 19. Alkuperäinen tyyli ja uusi vaihtoehtotyyli

Tein siis tätä kyseistä otsikkoa varten erillisen tyylin (kuva 19), jossa fontti on 0.1em pienempi ja asetin sen tälle otsikolle, jotta sain virheen ratkaistua vaikuttamatta muihin elementteihin.

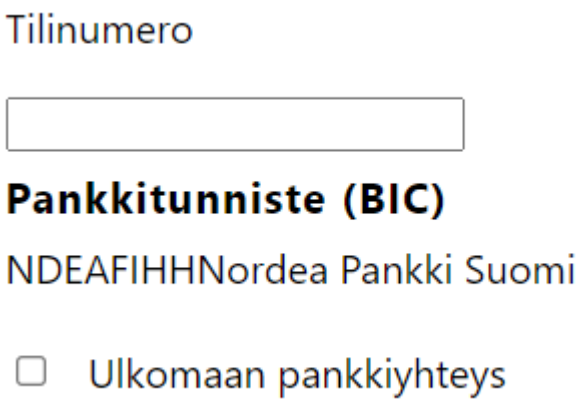
Toinen virhe oli niinkin yksinkertainen, että olin vahingossa tekstien lisäämisen yhteydessä pistänyt itse tekstiin pakollisuustähden (*), minkä takia ruudunlukija luki sen tähtenä. Tähän tein samanlaisen ratkaisun kuin aiemminkin, eli spanilla lisätään tähti, jota ruudunlukija ei näe ja lukee sen tilalla sanan ”Pakollinen”.

Tiistai 12.4.2022

Päivä alkoi sillä, että huomasin Agile boardille tulleen uuden teksteihin liittyvän virheen opiskelijoiden terveydenhuoltomaksu -sovelluksessa. Epäonnistuneiden maksujen jälkeen tulevassa ”Verkkomaksu epäonnistui” -tekstistä puuttui piste perästä, joka pitäisi korjata. Rupesin siis täten tutkimaan vähän syvällisemmin sovelluksen rakennetta, että millä sivulla kyseinen tekstinpätkä ilmaantuu ja mistä pääsen itse tekstiä vaihtamaan. Aamupäivän perehtymisen jälkeen tuli seinä vastaan, kun sovelluksen tekstit eivät ole sovelluksessa itsessään missään tiedostossa, vaan verkossa tekstirekisterissä, johon minulla ei ole käyttöoikeuksia. Otin yhteyttä sovelluksen määrittelijälle, joka kävi nopeasti korjaamassa tekstirekisterin puolella puuttuvan pisteen ja piste näkyikin sen jälkeen automaattisesti testiympäristöissä. Harvinaista kyllä, kyseessä oli virhe, johon itse ei tarvinnut tehdä oikeasti yhtään työtä, mutta oli hyödyllistä sentään käydä sovelluksen rakenne läpi siltä varalta, jos tulevaisuudessa tarvitsee samaan sovellukseen tehdä työtä. Loppupäivän jatkoin yksinkertaisilla elatustuen sovelluksen tekstivirheiden ratkomista.

Keskiviikko 14.4.2022

Päivän aikana tuli kaksi virhettä työn alle. Ensimmäinen virhe oli, että sovelluksen tilinumero -näytöllä ohjeistetaan, että jos käyttäjä haluaa ilmoittaa ulkomaisen tilinumeron, tulee tyhjentää pohjatietona oleva tilinumero, valita Ulkomaan pankkiyhteys ja painaa Jatka -painiketta.



Kuva 20. Käyttöliittymä, jossa tilinumero on muokattava syöttökenttä ja BIC-koodi kovakoodattu

Kuitenkin, jos tukea hakee käyttäjä, jolla on tilinumero jo pohjatietona ja hän pyyhkii manuaalisesti tilinumeron pois syöttökentästä ja valitsee ulkomaan pankkiyhteyden, antaa sovellus virheilmoituksen jo olemassa olevasta BIC-

koodista, joka haetaan automaattisesti maksupalvelusta, eikä sitä saa poistettua (kuva 20). Ulkomaan pankkiyhteys -näytölle kuitenkin pääsee, jos painaa sivun alareunassa löytyvää Tyhjennä sivun tiedot -painiketta, joka pyyhkii tilinumeron ja BICin pois, ja jatkaa siitä.

Aluksi ajattelin koodin puolelta ruveta ratkomaan asiaa, mutta BIC-koodin tyhjennys tilinumeron poistuessa reaaliajassa osoittautui hyvin monimutkaiseksi, eikä omat taidot siihen riittäneet. Soitin varmuuden vuoksi elatustuen tuoteomistajalle ja määrittäjälle. Hän oli heti sitä mieltä, että ulkomaiseen pankkiin maksavat ihmiset ovat todella marginaalitapauksia.

Kuva 21. Käyttöliittymä, jossa päivitetyt tiedot painikkeella jatkamisesta

Sovittiin siis virheen ratkaisusta niin, että sovelluksen teksteihin lisätään teksti, jossa kerrotaan, että Tyhjennä sivun tiedot -painikkeella saa tyhjennettyä pohjatietona oleva tilinumero (kuva 21).

Saman puhelun aikana totesin, että aikaa löytyy toteuttaa yhteystiedot -näytölle toisen tekstivirheen korjaamisen ohessa teksti ja linkki, joka ohjaa Digi- ja

Väestötietoviraston sivuille muuttoilmoitusta varten, sillä sovelluksesta itsessään ei voi vaihtaa omaa osoitetta, koska osoitetiedot tulevat automaattisesti väestötietojärjestelmästä.

```
<div class="sivun_sisalto_margin">
    <div class="mb3">
    <div class="sisalto mb3">
    <h:outputText value="#{tekstit.yhteystiedot_alku}">
    </h:outputText>
    <a target="_blank"
    href="#{tekstit.yhteystiedot_alku_linkki}">#{tekstit.yhteystiedot_alku_linkkiteksti}</a>
    </div>
    <div class="sisalto mb3">
    <h:outputText value="#{tekstit.yhteystiedot_alku2}">
    </h:outputText>
    </div>
```

Kuva 22. Toteutus perinteisestä uudelleenohjaus -linkistä

Tämä ratkaistiin JSF-tagilla h: outputText, johon upotettiin linkki muuttoilmoitussivulle. Pidempiä tekstejä ei ikinä kirjoiteta suoraan sovellukseen, vaan ne haetaan erillisestä tekstitiedostosta (kuva 22).

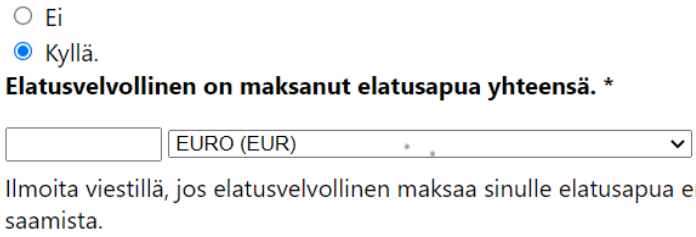
Torstai 14.4.2022

Päivä oli lyhyempi kiirastorstain vuoksi, mutta ehdin silti korjaamaan kaksi uutta Agile boardille ilmestynyttä virhettä. Kummatkin suhteellisen yksinkertaisia virheitä, toinen puuttuvaan tekstiin ja toinen tyyleihin liittyvä. Ensimmäinen virhe oli, että Maksamaton elatusapu -näytöllä on syötekentät, joihin pitää kirjoittaa mistä lähtien elatusvelvollinen on jättänyt elatusavun maksamatta. Kysymys sisältää siis kummatkin syötekentät alku- ja loppupäivämäärälle, mutta pelkästään alkupäivämäärä on pakollinen tieto, ja tätä ei ollut sovelluksessa mainittu ollenkaan.

```
<div class="mb1 sisalto">
    <h:outputText value="#{tekstit.laiminlyonti_alkaen}">
    </h:outputText>
</div>
<div class="mb1 sisalto">
    <h:outputText value="#{tekstit.elatusapu_eipakollinen}" />
</div>
```

Kuva 23. Toteutus uusista teksteistä

Virhe ratkesi lisäämällä tekstin alle myös edellisillä näytöillä ollut ”Päättymisaika ei ole pakollinen tieto” -teksti, minkä johdosta tekstikoodi osoittaa elatusapu -näytölle, eikä laiminlyönti -näytölle (kuva 23.). Toinen virhe oli myös samassa näytössä.



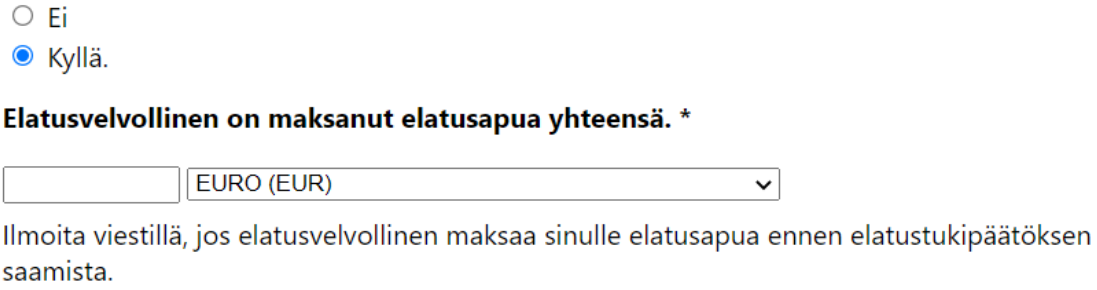
Kuva 24. Käyttöliittymä ennen korjausta

Näytöllä olleen lisäkysymyksen radiobutton -valintojen ja niistä ilmestyvän elementin otsikon välillä oli liian vähän väljyyttä, mikä haittasi kognitiivista saavutettavuutta (kuva 24).

```
<div class="mb1 sisalto mt2" id="maksettuYhteensa_div">

    <div class="sisalto_bold" class="mb1">
        <h:outputText value="#{tekstit.laiminlyonti_maksanut_yhteensa}"></h:outputText>
        <span aria-hidden="true" class="musta">*</span> <span
        class="sr-only">#{tekstit.pakollinen}</span>
    </div>
```

Kuva 25. Toteutus väljyyden lisäämisestä

Väljyyttä lisättiin lisäämällä elementin tyyliluokkaan luokka mt2 (kuva 25), joka lisää elementin yläosaan 2 prosenttia lisää väljyyttä (kuva 26).



Kuva 26. Käyttöliittymä korjauksen jälkeen

Tämän jälkeen käyttöliittymän kyseinen kohta näytti huomattavasti paremmalta lukea, eli kognitiivisen saavutettavuuden kannalta onnistuin.Työpäivän lopuksi vielä käänsin uuden version sovelluksesta Jenkinsin avulla testiympäristöihin ensi viikoksi testaajalle uudelleentestattavaksi.

Viikon 3 analyysi

Kaiken kaikkiaan olin tämän viikon suoritukseeni hyvin tyytyväinen. Virheitä sai korjattua sitä mukaan, kun niitä tuli, eikä viikonlopuksi jäänyt yhtään tehtävää Agile boardille roikkumaan. Eniten ylpeyttä aiheutti kuitenkin oman itsenäisyyden lisääntyminen. Olen yleensä ennen ottanut yhteyttä omaan työharjoitteluajan mentoriin, josta on samalla tullut muutenkin mentori, johon otan oletusarvoisesti aina ongelmien tullessa ensin yhteyttä. Hän kuitenkin oli tämän viikon lomalla ja on vielä ensi viikonkin, joten jouduin itse ottamaan asioihin liittyviin henkilöihin yhteyttä, eikä se ollutkaan niin paha, kun kuvittelin. Ehkä ei jatkossakaan tarvitse turvautua vaan yhteen henkilöön niin paljoa.

4.4 Viikko 4

Viimeinen seurantaviikko menee myös elatustuen sovelluksen virheiden korjaamisessa. Epäilen, että viikon aikana ei tule enää niin paljoa virheitä mitä edellisillä viikoilla on tullut, kun testaaja on arviolta kahden sivun päässä valmistumisesta. Näiltä sivuilta voi tietenkin tulla pieni kasa virheitä, ja myös vanhoja virheitä voi tulla uudelleen korjattavaksi, jos niissä on jäänyt jotain korjaamatta.

Tiistai 19.4.2022

Pääsiäisloman jälkeinen ensimmäinen työpäivä alkoi sähköpostin läpikäymisellä. Sähköpostin lukemisen ja aamukokousten jälkeen kävin vilkaisemassa Agile boardia, jossa löytyi pari monimutkaisemman oloista virhettä. Otin yhden näistä päivälle työn alle. Kyseisessä virheessä yhteenveto -näytölle tulevaa tekstiä näkyy ruotsinkieliselle käyttäjälle suomeksi ja suomenkieliselle käyttäjälle ruotsiksi. Oletin, että virhe liittyy kieliparametrin välittymiseen yhteenveto -näytölle, mutta en päivän aikana saanut virhettä ratkaistua. Toivottavasti huomenna tulee läpimurto ja tajuan mistä on kyse.

Päivän aikana tuli myös tehtyä töitä OpenShiftin puolella. Käänsin opiskelijoiden terveydenhuoltomaksu -sovelluksen uudelle testipalvelimelle ja tein aikaisemmin mainittuun secretiin pankkimuuttujamuutokset. Tutkin myös yhtä testi-

palvelinta, joka ei suostunut aukeamaan selaimeen koska palvelimeen ei saanut yhteyttä. Tulin siihen tulokseen muutaman uudelleenkäynnistyksen jälkeen, että palvelin ei taida olla vielä valmis käytettäväksi, tai vaihtoehtoisesti siellä on joku ongelma mihin ei omat amatöörin OpenShift -taidot riitä.

Keskiviikko 20.4.2022

Jätin eilisen kieliparametrivirheen tältä päivältä väliin, kun Agile boardille oli tullut iso kasa muita virheitä tälle päivälle tehtäväksi. Suurin osa virheistä oli yksinkertaisia teksti- tai tyylivirheitä, mutta mukaan mahtui myös muutama virhe, jossa elementit näkyivät väärin tai duplikaatteina yhteenvedossa. Yksinkertaisia nekin. Suurin ongelma on selata yhteenveto -näyttöä läpi, kun se sisältää käytännössä kaiken syötetyn tiedon mitä sovellukselle on annettu.

Päivästä meni myös pari tuntia aikaa teknologian CoP -kokoukseen, jossa käytiin läpi sovelluskehityksen tuotealueen roadmap, digitaalisten palvelujen kehittämistä sekä UUID:n käyttöä henkilön tunnisteena.

Torstai 21.4.2022

Päivä meni tänäänkin yksinkertaisten yhteenveto -näytön virheiden ratkomisessa. Jatkoin myös kieliparametrivirheen ratkomista, ilman tuloksia tosin. En ymmärrä, mistä sovellus nappaa yhteen tekstin pätkään väärän kieliparametrin, vaikka muualla se toimii oikein. Motivaatio taidon puutteeseen meni sen verran, että taidan jättää sen tältä viikolta kesken ja jatkaa ensi viikolla.

Perjantai 22.4.2022

Tänään tuli vieläkin korjailtua yksinkertaisempia virheitä viimeisiltä näytöiltä. Monimutkaisin virhe näistä oli saapumisvahvistus -näytöllä yksi, jossa sivun väliotsikko ei ollut määrittelyiden mukainen. Aikaa kului paljon, kun kävin otsikon funktiotiedostoa läpi, että löytäisin virheellisen koodin kohdan. Väliotsikon muodostajafunktiossa oli laitettu tekstiä hakevaan funktioon väärä tekstinpätkä.

```
public String getHakemusVaiIlmoitus() {

    String valiotsikko = "";

    if(Vakiot.HAKEMUKSEN_TYYPPI_HAKEMUS.equals(this.getLomakeTiedot().getHakemus().getHak
        valiotsikko  = OmaUtil.haeTeksti(this.getFacesContext(), "etuus_hakemuksennimi");
    } else {
        valiotsikko  = OmaUtil.haeTeksti(this.getFacesContext(), "ilmoitus_otsikko");
    }

    return valiotsikko;
}
```

Kuva 27. Toteutus väärän väliotsikon korjaamisesta

Kuten kuvassa 27 voidaan nähdä, sovellus tunnistaa .equals -funktiolla, onko hakemuksen tyyppi hakemus vai muutosilmoitus. Jos tyyppi on hakemus, asettaa funktio väliotsikoksi eri tekstin, kun muutosilmoituksessa. Vaihdoin hakemuksen väliotsikoksi sen, mitä määrittelyissä kerrottiin, ja kävin myös ruotsinkielisestä määrittelystä vaihtamassa ruotsinkielisiin teksteihin oikean version. Loppupäivästä käänsin Jenkinsillä uuden version sovelluksesta, että testaaja pääsee testaamaan uudelleen korjatut virheet.

Viikon 4 analyysi

Viimeinen seurantaviikko jätti hieman kylmäksi. Olen tyytyväinen siihen määrään virheitä mitä sai korjattua, mutta viikonlopun yli roikkumaan jäävä kieliparametri -virhe jäi kaivamaan. Toivottavasti saan ensi viikolla ratkottua asian. OpenShiftin kanssa säätäminen tuli myös hyvin tarpeeseen. Sitä tulee kuitenkin todennäköisesti tulevaisuudessa tarvitsemaan enemmän.

5 POHDINTA

Päiväkirjamallinen opinnäytetyö osoittautui hyväksi ideaksi. Ajanhallinnallisesti katsottuna työ oli oikea ratkaisu itselleni, koska tein opinnäytetyön kirjoittamisen yhteydessä täysipäiväisesti töitä. Päiväkirjan toteutus opetti itseäni kartuttamaan omaa osaamista sovelluskehittäjänä sekä tuki omaa oppimista päivittäisten töiden kirjaamisen ja kertaamisen yhteydessä. Työnseurannan jakson aikana tarkoitus oli kehittää omaa osaamista teknisellä tasolla ja yleisesti työntekijänä, jonka koin onnistuvan hyvin.

Opinnäytetyön toteuttamisen aikana tein monipuolisia sovelluskehittäjän töitä. Opiskelijoiden terveydenhuoltomaksu -sovelluksen muutosten myötä pääsin tutustumaan Spring Boot- ja Thymeleaf -yhdistelmään. En ennen ole kyseisiin kehyksiin perehtynyt omien ohjelmointiprojektien myötä, joten koin työt sovelluksen parissa hyvinkin mielekkääksi ja opettavaiseksi. Työt olivat myös virkistävästi vaihtelevia, kun pääsin pitkästä aikaa tekemään palvelinpuolen ohjelmointia.

Elatustuen sovelluksen parissa työskentelin jo harjoittelujaksolta entuudestaan tutulla JSF- ja JSP-yhdistelmällä. Vaikka omat työtehtävät vaikuttivat työnseurannan jakson ajalta samanlaisilta, oli monen virheen korjaamisen toteuttaminen kuitenkin erilaista toisiinsa verrattuna. Huomasin ilokseni neljän viikon ajalta, että oma nopeus virheiden korjaamisessa kehittyi huomattavan paljon. Jos vertaa siihen, millä tahdilla tein töitä esimerkiksi oman harjoittelujakson aikana, on ero kuin yö ja päivä. Kuitenkin osaamattomuuteni tietyillä Java-kehityksen puolilla aiheutti hetkellistä turhautumista, mutta tämän voi käsittää myös positiivisena asiana. Aina on tilaa oppia enemmän.

Omat taidot yleisesti työntekijänä parantuivat myös paljon. Kommunikaatiotaidot paranivat sen verran, ettei enää tarvitse turvautua vain yhteen tuttuun työkaveriin, vaan pystyn ottamaan matalalla kynnyksellä yhteyttä myös muihinkin ei niin tuttuihin ihmisiin ja kysymään heiltä apua tai ajatuksia töihin liittyen. Etätöissä varsinkin tuntui, ettei kehtaa häiritä muita ihmisiä omiin töihin liittyen, koska tietokoneiden välityksellä ei koskaan tiedä onko toisella kiire. Opinnäytetyön aikana opin sen, että kaikilla on joskus aikaa, ja toivon myös, että minuun otettaisiin yhteyttä, jos uudemmille työntekijöille tulee ongelmia.

Yksi tavoitteista opinnäytetyötä varten oli oman ajanhallinnan kehittäminen. Se parantui hieman, mutta en vieläkään osaa ihan täysin arvioida kuinka kauan työtehtävien toteuttamisessa kestää. Tämä johtuu siitä, että yksinkertaiselta vaikuttava virhe voikin olla monen tunnin urakka, johon en osannut varautua aikaisemmin. Kehitystä kuitenkin tapahtui siinä mielessä, että en käytä monta päivää peräkkäin yhden virheen kanssa taisteluun, vaan siirryn suosiolla tekemään muita hommia ja palaan virheen pariin siinä vaiheessa, kun ensimmäinen mieleen tuleva asia ei ole enää hermoilu siitä, että virhe ei ole ratkennut vielä.

Koin myös kehittyneeni projektityönteossa hyvin. Seurantajakson aikana en kokenut enää ongelmaksi saada tehtyä sprintin ajalta kaikkia työtehtäviä valmiiksi, vaan sain työt hyvin tehtyä, eikä ensi sprintille jäänyt Agile boardille roikkumaan vanhoja töitä. Koen, että pystyisin ottamaan tiukemmilla deadlineilla olevia töitä vastaan tästä edespäin.

Seuraava tavoitteeni tästä edespäin on kehittyä lisää kommunikaatiotaidoissa ja omassa teknisessä osaamisessa. Itsenäistyin sovelluskehittäjänä hyvin, mutta olen kuitenkin vasta ”junior” -sovelluskehittäjä, eli puhekielellä ns. keltanokka, joten oppimista on vielä paljon jäljellä.

LÄHTEET

Aluehallintovirasto. s.a. Yleistä saavutettavuudesta. WWW-dokumentti. Saatavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/ [viitattu 15.4.2022]

Chopra, V., Li. S., Jones, R., Eaves, J., Bell, J. 2005. Beginning JavaServer Pages. Indianapolis: Wiley Publishing. E-kirja. Saatavissa: https://ebookcentral.proquest.com/ [viitattu 14.3.2022]

Frontbackend. 2020. How to work with Fragments in Thymeleaf. WWW-Dokumentti. Saatavissa: https://frontbackend.com/thymeleaf/how-to-work-with-fragments-in-thymeleaf [viitattu 20.4.2022]

Gulati, S. 2014. OpenShift Cookbook. Birmingham: Packt. E-kirja. Saatavissa: https://ebookcentral.proquest.com/ [viitattu 1.4.2022]

Javatpoint. s.a. Spring Boot Tutorial. WWW-dokumentti. Saatavissa: https://www.javatpoint.com/spring-boot-tutorial [viitattu 30.4.2022]

Karaman, R. 2018. Spring: Microservices with Spring Boot. Birmingham: Packt. E-kirja. Saatavissa: https://ebookcentral.proquest.com/ [viitattu 11.3.2022]

Kumar, N. 2021. How the Model View Controller Architecture Works – MVC explained. WWW-dokumentti. Saatavissa: https://www.freecodecamp.org/news/model-view-architecture/ [viitattu 15.4.2022]

Mankale, A. 2015. Mastering Spring Application Development. Birmingham: Packt. E-kirja. Saatavissa: https://ebookcentral.proquest.com/ [viitattu 13.3.2022]

Mkyong. 2013. JSF 2.0 Tutorial. WWW-dokumentti. Saatavissa: https://mkyong.com/tutorials/jsf-2-0-tutorials/ [viitattu 15.4.2022]

Shipley, G. 2014. Learning OpenShift. Birmingham: Packt. E-kirja. Saatavissa: https://ebookcentral.proquest.com/ [viitattu 1.4.2022]

Spring Boot. s.a. Building an Application with Spring Boot. WWW-dokumentti. Saatavissa: https://spring.io/guides/gs/spring-boot/ [viitattu 11.3.2022]

Thymeleaf. s.a. Getting started with the Standard dialects in 5 minutes. WWW-dokumentti. Saatavissa: https://www.thymeleaf.org/doc/articles/standarddialect5minutes.html [viitattu 13.3.2022]

Thymeleaf. 2018. Tutorial: Using Thymeleaf. WWW-dokumentti. Saatavissa: https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html [viitattu 13.3.2022]

Tyson, M. 2019. What is JSP? Introduction to JavaServer Pages. Artikkeli. Päivitetty 29.1.2019. Saatavissa: https://www.infoworld.com/article/3336161/what-is-jsp-introduction-to-javaserver-pages.html [viitattu 14.3.2022]

Woods, D. 2014. Exploring Micro-Frameworks: Spring Boot. Artikkeli. Päivitetty 18.3.2014. Saatavissa: https://www.infoq.com/articles/microframeworks1-spring-boot/ [viitattu 11.3.2022]

KUVALUETTELO