Oskari Puumala

OPUX-KOMPONENTTIKIRJASTO JA JSP-SOVELLUSKEHITYS

OPUX-KOMPONENTTIKIRJASTO JA JSP-SOVELLUSKEHITYS

Oskari Puumala
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Oskari Puumala
Opinnäytetyön nimi: OPUX-komponenttikirjasto ja JSP-sovelluskehitys
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: Syksy 2021 Sivumäärä: 45

Tässä päiväkirjamuotoisessa opinnäytetyössä seurattiin ohjelmistokehittäjän työskentelyä Osuuspankin verkkosovellusten kehittäjänä. Työn aikana havainnoitiin kehitystä työskentelytavoissa. Tämän lisäksi analysoitiin suljetun lähdekoodin React-pohjaisen OPUX-komponenttikirjaston mukana ilmeneviä toimintatapoja sekä vertailtiin sitä avoimen lähdekoodin vastaaviin kirjastoihin. Raporttikirjauksia tehtiin päivittäin ja viikoittain kirjoitettiin analyysi viikon työskentelystä.

Työn aikana tärkeimmät käytetyt teknologiat olivat JavaSript, React, Java Spring sekä Jakarta Server Pages (JSP). Työn alkuvaiheessa keskityttiin React-sovellusten kanssa työskentelyyn, mutta loppua kohden työt keskittyivät enemmän JSP- ja Java-sovelluksiin.

Opinnäytetyön aikana havaittiin osaamisen kasvua JSP-pohjaisten sovellusten kehityksessä. OPUX-komponenttikirjasto havaittiin hyödylliseksi työkaluksi React-pohjaisten sovellusten kehityksessä. Lisäksi kirjoittaja analysoi JSP-pohjaisten sovellusten testaushaasteiden vaikutusta Git-versionhallintatyökalun käyttöön.

Avainsanat: Ohjelmistokehitys, JavaScript, Java, päiväkirjamuotoinen opinnäyte

ABSTRACT

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

---

Author(s): Oskari Puumala
Title of thesis: OPUX Component Library and JSP Software Development
Supervisor(s): Eino Niemi
Term and year when the thesis was submitted: Fall 2021
Number of pages: 45

This diary-form thesis follows the work of a software engineer developing web-site applications. Technologies used in development were JavaScript, React, Java and Jakarta Server Pages (JSP). The writer of the thesis also analyzes the advantages and disadvantages of a closed source React component library OPUX while comparing it to open source variants.

At the beginning of the thesis work focuses mostly on developing applications with React. Toward the end of the thesis the focus shifts more into applications developed using JSP.

At the end of the thesis the writer observes improvement in his ability to develop applications using JSP but feels there's still a lot to learn. He also mentions OPUX component library being a helpful tool in developing React applications.

Keywords: Software development, Java, JavaScript, diary-form thesis

SISÄLLYS

1 JOHDANTO

Osuuspankilla on käynnissä projekti sovellusten päivittämiseksi saavutettavimmiksi. Saavutettavuudella tarkoitetaan järjestelmän käytön helpottamista käyttäjille, joilla on esimerkiksi heikompi näkökyky. Saavutettavuuteen kuuluu esimerkiksi ruudunlukija tuen kehittäminen ja sovellusten näppäimistökäytön helpottaminen. Lisäksi saavutettavuuteen kuuluu sovellusten värimaailman kehittäminen sellaiseksi, että toiminnallisuudet olisi helpompi hahmottaa.

Tällä hetkellä tiimillämme on yksi täysin saavutettava sovellus. Lähitulevaisuudessa tulemme aloittamaan saavutettavuuspäivityksiä myös muille tiimimme sovelluksille.

Käytössä on OPUX-komponenttikirjasto sovellusten selaintoteutusten kehittämiseen. Kirjasto on Osuuspankin sisäinen React-komponenttikirjasto, joka sisältää perinteisiä verkkosovelluksissa käytettäviä komponentteja. Komponentteihin kuuluvat esimerkiksi erilaiset syötteet, valintalistat, painikkeet ja listat. Kirjastoa on kehitetty saavutettavuus edellä, jotta saavutettavien sovellusten kehittäminen olisi kehitystiimeille mahdollisimman helppoa.

Tästä johtuen olenkin valinnut opinnäytetyön tavoitteeksi analysoida kirjaston käytön mukana ilmeneviä toimintamalleja. Kirjasto on hyvin keskeinen osa sovellusten selaintoteutusten kehitystä ja se on hyvin tärkeä osa jokaisen yhtiössä toimivan kehittäjän päivittäistä työtä. Tahdon selvittää, mitä hyötyjä tai haittoja kirjastoa käytettäessä tulee esiin. Aion myös vertailla kirjaston käytettävyyttä ja toiminnallisuuksia muihin avoimen lähdekoodin vastaaviin React-kirjastoihin.

2 NYKYTILANTEEN KUVAUS

2.1 Nykytilanne ja osaaminen

Työskentelen OP-Palvelut Oy:ssä ohjelmistokehittäjänä. Olen toiminut työssäni vuoden 2020 toukokuun alusta lähtien. Vuoden 2021 vaihteessa vaihdoin Osuuspankin sisällä toiseen tiimiin, jonka mukana toimin tällä hetkellä. Tiimini vastuualueeseen kuuluu muutamia Osuuspankin verkkosovelluksia, joten työni koostuu näiden sovellusten palvelin- ja selaintoteutusten kehittämisestä.

Sovellusten kehitykseen käytetään Java Spring -viitekehystä palvelintoteutuksessa ja React JavaScript -viitekehystä selaintoteutuksessa. Projekteissa käytetään myös useita eri työkaluja, kuten Maven, Docker ja Git. Käytettävät teknologiat on listattu tarkemmin taulukossa luvussa 2.3.

Työssäni tarvitsen osaamista sovellusten kehittämisessä Javaa ja JavaScriptiä käyttäen. JavaScript-kehityksestä minulla on kokemusta jo kahden vuoden ajalta. Erityisesti React-sovellusten parissa koen osaamiseni olevan hyvällä tasolla, ja suoriudunkin näistä tehtävistä useimmiten itsenäisesti. Palvelintoteutukset ovat minulle uutta ja kokemusta Java-ohjelmoinnista on vähemmän. Usein joudun myös kysymään neuvoja palvelintoteutuksiin liittyvissä tehtävissä. Olen kuitenkin kiinnostunut kehittämään Java-taitojani. Teknistä ymmärrystä rajapinnoista minulla on kehittynyt tietoturvatestaustehtävissä. Osuuspankilla sovellukset ovat laajoja, ja riippuvuuksia löytyy useisiin eri sovelluksiin. Usein työssä täytyykin tehdä yhteistyötä useiden eri tiimien kanssa, jotta sovellukset toimivat yhteen. Sovellusten sekä yhtiön laajuus aiheuttaa sen, että työssä on tiettyjä toimintatapoja, jotka tulee oppia. Vuoden aikana olen hyvin saanut sisäistettyä tärkeimmät toimintatavat. Näihin kuuluvat esimerkiksi sovellusten koonnit ja muutosten julkaiseminen. Viikoittain törmään kuitenkin uusiin toimintatapoihin työkalujen ja ympäristöjen kanssa.

Tiimimme työskentelee kahden viikon sprinteissä ja sprintin suunnittelupalaverit pidetään joka toinen torstai. Daily-palavereita pidetään päivittäin. Töiden etenemistä seurataan Jira-tehtävienhallintatyökalun avulla. Jirasta ja muuta sanastoa tarkemmin luvussa 2.4.

2.2 Sidosryhmät

Tiimimme lisäksi yhtiössä on useita kehitystiimejä, joilla on omat vastuusovellukset. Lisäksi löytyy tiimejä, joiden vastuualueisiin kuuluu testaus- ja julkaisuympäristöjen hallinta sekä Git-projektinhallintasovelluksemme ylläpito. Tärkeimpinä yhteistyötiimeinä ovat kuitenkin ympäristöistä vastaava tiimi, tiettyjä rajapintapalveluja tarjoavat tiimit sekä tiimi, jonka sovellukseen eräästä meidän tiimimme sovelluksesta on linkitys.

2.3 Käytettävät teknologiat

Taulukossa 1 on listattu tärkeimmät projektin aikana käytetyt teknologiat.

Taulukko 1. Opinnäytetyössä käytetyt teknologiat

CSS	Merkintäkieli, jolla luodaan tyylejä ja asetteluita verkkosivuille.
Enzyme	Testaustyökalu, jota käytetään osassa React-sovellustemme testeissä Jest-testausviitekehyksen kanssa. Enzymen avulla komponentteja voidaan renderöidä ja manipuloida. Manipuloinnista esimerkkinä tapahtumien suorittaminen ja komponenttien parametrien muuttaminen.
Git	Versionhallintatyökalu, jonka avulla useat kehittäjät voivat työskennellä samoissa sovelluksissa ja tiedostoissa. Gitissä on muutoshistoria, josta kehittäjien muutoksia voidaan seurata.
Jakarta Server Pages, JSP	Java-pohjainen ohjelmointiteknologia, joka on käytössä tiimimme vanhempien sovellusten palvelintoteutuksissa. Se hallitsee sitä, millaisia HTML-sivuja palvelin palauttaa selaimelle.
Java	Ohjelmointikieli, jota projektin aikana käytetään palvelintoteutusten kehitykseen.
Java Spring	Java-viitekehys, jolla projektin aikana työstettäviä palvelintoteutuksia ja rajapintoja kehitetään.

JavaScript	Ohjelmointikieli, jota projektin aikana käytetään web-sovellusten selaintoteutusten kehityksessä.
Jenkins	Automaatio-ohjelmisto, joka suorittaa sovelluksille testauksen ja koonnin, kun muutoksia viedään Git-versionhallintaan.
Jest	JavaScript-testausviitekehys, jota käytetään testaamaan React-sovellustemme toteutuksia.
Jira	Tehtävienhallintatyökalu, jolla seurataan tiimin jäsenten tehtävien etenemistä.
jQuery	JavaScript-kirjasto, joka on käytössä tiimimme vanhempien sovellusten selaintoteutuksissa. Sovelluksissamme sitä käytetään manipuloimaan JSP:n palauttamia HTML-sivuja.
Maven	Työkalu, jolla voidaan koota Java-sovelluksia.
React	JavaScript-viitekehys, jota projektin aikana käytetään React-pohjaisten web-sovellusten selaintoteutusten kehittämiseen.
Robot Framework	Testiautomaatioviitekehys, jolla toteutetaan automaatiotestit tiimimme sovelluksiin. Robot käyttää sovellusta normaalin käyttäjän tavoin klikkailemalla ja kirjoittamalla syötteisiin.
SonarQube	Koodin laadunvalvontaan käytettävä alusta, joka ympäristössämme suorittaa tarkistuksia Jenkinsajon yhteydessä.
XPath	Kyselykieli, jota käytetään Robot-testiautomaatioviitekehyksen kanssa. Sen avulla HTML-koodin elementtejä voidaan etsiä, jotta Robot voi vaikuttaa niihin.

2.4 Sanasto

Taulukossa 2 on selvennetty muutamia termejä, jotka nousevat esiin useaan otteeseen opinnäytetyön aikana.

Taulukko 2. Työssä esiintyviä termejä

Jira tiketti	Yksittäinen kirjaus Jirassa, jonka etenemistä seurataan. Tiketti voi olla esimerkiksi kirjaus ongelmasta tai jostain kehitysideasta. Tiketti on useimmiten osoitettu jollekin kehittäjälle, mutta osoitettu henkilö voi vaihtua tiketin elinkaaren aikana. Tiketit voivat olla eri tiloissa, kuten esimerkiksi: ”kehityksessä”, ”testauksessa”, ”analysoitavana” tai ”valmis”.
Rajapinta	Sovelluksen osa, jonka kautta toteutetaan tiedonvaihto toisen sovelluksen kanssa.
Commit	Muutos, joka lisätään Git-versiohallintatyökalun historiaan. Commitin alla nähdään tiedostot, joita on muokattu. Committiin on kommentoitu mitä muutoksessa on tehty.

3 PÄIVÄKIRJARAPORTOINTI

3.1 Ensimmäinen viikko

Maanantai 10.5.2021

Päivän tehtävänä oli purkaa väliaikainen korjaus erään sovelluksen lomakkeesta. Sovellus sisältää useita syötteitä, joista yksi on päivämääräsyöte. Lomake on toteutettu käyttäen OPUX-kirjastoa ja se sisältää perinteisten syötteiden lisäksi pudotuslistoja, sekä kalenterikomponentin. Kaikki komponentit ovat peräisin OPUX-kirjastosta.

Osuuspankin sovelluksilta vaaditaan saavutettavuutta, joten lomake on toteutettu HTML-standardien mukaisesti. Tämä tarkoittaa sitä, että Enter-näppäintä painettaessa lomake lähetetään eteenpäin. Sovelluksessa ongelmana oli OPUX-kalenterikomponentti, jonka arvo päivitetään eteenpäin vain silloin, kun selaimen kohdistus poistuu komponentista. Lähetettäessä lomaketta Enter-näppäimellä selaimen kohdistuksen ollessa kalenterikomponentissa ei syötteen tieto päivity lomakkeeseen. Tämä johtuu siitä, että kohdistus ei poistu syötteestä ja ylemmän tason hallitseva komponentti ei saa kalenterin arvoa. Tällöin lomakkeelta lähetetään kalenterin oletusarvo ”TÄNÄÄN” eikä käyttäjän valitsema päivämäärä.

Ongelma oli hetkellisesti ohitettu poistamalla Enter-näppäimen lähetystoiminto käytöstä. Tämä tuli purkaa, jotta sovellus voidaan toteuttaa standardien mukaisesti.

Tutkimisen ja testauksen pohjalta toimivaksi ratkaisuksi ilmeni blur-tapahtuman pakottaminen kalenterikomponentin saadessa Enter-painalluksen. Blur-tapahtuma poistaa selaimen kohdistuksen syötteestä ja arvo saadaan onnistuneesti päivitettyä hallitsevalle komponentille. Tapahtuma suoritetaan, ennen kuin ylempänä HTML-puussa sijaitsevan lomake-elementin lähetystapahtuma suoritetaan. Tämä johtuu JavaScriptin bubbling-toteutuksesta, joka siirtää elementtien tapahtumat ylöspäin ympäröiville elementeille. Tapahtumat suoritetaan järjestyksessä tapahtuman saaneelta komponentilta ylimmälle komponenttia ympäröivälle elementille (1).

Loppupäivä kului yksikkötestin toteutuksessa Jest-testausviitekehystä käyttäen. Testillä varmistettiin, että selaimen kohdistus poistetaan Enter-painalluksella.

Tiistai 11.5.2021

Seuraavana tehtävänä oli Jest-axe-yksikkötestauskirjaston käyttöönotto kahteen sovellukseen. Jest-axe-kirjastoa käytetään testaamaan React- ja Vue-viitekehyksillä toteutettujen verkkosovellusten saavutettavuutta. Kirjaston on tarkoitus löytää yksinkertaisia saavutettavuusongelmia sovellusten HTML-koodista.

Jest-axe-kirjaston käyttöönotto oli yksinkertaista. Se asennettiin kuin mikä tahansa npm-paketti ja lisättiin yksi rivi testien esivalmistelukoodiin:

```
expect.extend(toHaveNoViolations)
```

Jest-axe lisää Jest-viitekehyksen expect-toiminnallisuuteen uuden ”matcherin”: toHaveNoViolations. Expect-metodi suorittaa parametrinä saadun koodin ja palauttaa sen tuloksen matcherille, joka on jest-axen tapauksessa toHaveNoViolations.

```
expect(axe(component)).toHaveNoViolations()
```

Axe-funktio tarkistaa komponentin saavutettavuuden ja toHaveNoViolations-metodi varmistaa, ettei saavutettavuuspuutteita löytynyt.

Kirjaston käyttö itsessään oli hyvin yksinkertaista: testattava komponentti renderöidään halutulla kirjastolla ja sen jälkeen suoritetaan yllä viitattu koodinpätkä.

Projekteissa on testeistä ja käyttötarkoituksista riippuen renderöintiin käytössä React-testing-library tai Enzyme. Jest-axe-testeihin renderöintiin otettiin käyttöön React-testing-library ja tätä varten toteutettiin apufunktio, joka hoitaa renderöinnin käyttäen oikeaa kielimuuttujaa.

Projekteissa on käytössä kieleistys, joka tietyn kielimuuttujan perusteella muuttaa komponenttien tekstejä oikealle kielelle. Yksikkötestien toteutuksissa kielet ja kielimuuttujien nimet olivat vaihtelevat. Samalla otettiin käyttöön yleinen kielimuuttuja, jotta projektin testien kieli saatiin yhtenäiseksi.

Keskiviikko 12.5.2021

Jest-axe otettiin käyttöön myös toiseen sovellukseen. Operaatio oli sama kuin aikaisemman sovelluksen kanssa, ja siitä selvittiin ongelmitta.

Daily-palaverissa projektin tuotteen omistaja antoi minulle uudeksi tehtäväksi tutkia ongelmaa, joka oli ilmennyt julkaisutestausympäristössä. Ongelmaksi oli ilmennyt linkki, joka tietyssä tilanteessa vie käyttäjän virhesivulle. Sovelluksessamme on linkki toisen tiimin sovellukseen. Toimiessaan linkin tulisi avata toisen tiimin sovellus käyttäen linkin parametreinä tulevia tietoja. Linkki kuitenkin avasi tietyillä tiedoilla virhesivun, eikä käyttäjä päässyt halutulle sivulle.

Siirryin toistamaan virhettä tiimimme omaan testiympäristöön, mutta ongelman toistaminen ei onnistunut. Tiimimme testiympäristössä sovellus aukesi ilman virhettä, mutta sovellusta ei avattu oikeilla tiedoilla. Mietin mahdollisena ongelmana olevan versioerot kahden ympäristön sovellusten välillä. Päätin kysyä tiimin kokeneemmilta kehittäjiltä, miten ympäristöjen sovellusten versioita voisi tarkistaa ja miten niitä voisi muuttaa.

Viikkoanalyysi

Tällä viikolla minulla oli eri tehtävät joka päivälle. Maanantaina työskentelin selaintoteutuksen parissa JavaScriptillä ja toteutin toteutukselle yksikkötestit. Maanantain aikana ei ilmennyt ongelmia ja tehtävistä selvittiin hyvin rutiininomaisesti.

Tiistaina jouduin opiskelemaan hiukan, sillä Jest-axe-kirjasto oli minulle täysin uusi. Jest-axen dokumentaatiot olivat kuitenkin hyvin selkeät ja työkalu saatiin käyttöön ilman suurempia ongelmia.

Keskiviikko koostui pitkälti ongelmanselvittelystä. Ongelman toistamisessa ilmeni haasteita ja oma tietämättömyyteni siitä, miten sovellusten versioita hallinnoidaan testiympäristöissä, hidasti ongelman tutkimista.

OPUX kalenterikomponentin analyysi

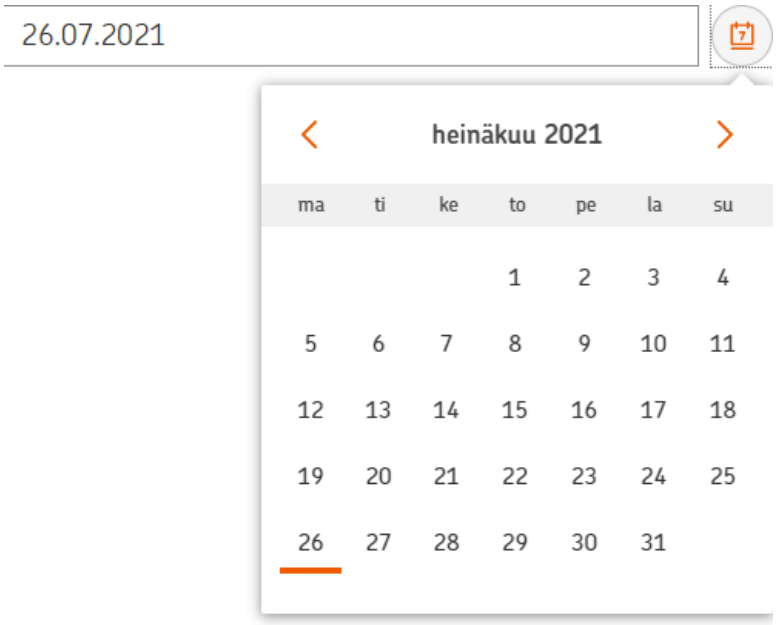
Kuluneen viikon aikana OPUX-komponenttikirjaston parissa ilmeni ongelmaa kalenterisyötteen kanssa. Kalenterisyötteen arvo päivitettiin vain selaimen kohdistuksen poistuessa syötteestä. Olisi loogista, että kalenterisyötteen arvoa päivitettäisiin jokaisella kirjaimen kirjoituksella normaalien syötteiden tapaan. Tällöin ei olisi myöskään ilmentynyt ongelmaa väärän päivämäärän lähetyksen kanssa.

Halusin vertailla OPUX-kirjaston kalenteritoteutusta avoimen lähdekoodin vastineisiin. Löysin kaksi vastaavanlaista toteutusta. Toinen on osa Ant Design React-kirjastoa, toinen on HackerOnen toteuttama React Date Picker-kirjasto.

Vertasin komponenttien onChange-tapahtuman suoritusta. Tapahtuman avulla käyttäjän syöttämiä arvoja voidaan käsitellä. Nimensä mukaisesti tapahtuma suoritetaan kalenterin arvon muuttuessa, mutta tapahtuman käynnistävä toiminta vaihtelee toteutuksen mukaan. Kirjoitin komponentteihin päivämäärän käsin ja testasin, miten onChange-tapahtuma saadaan ajettua. Kaikissa toteutuksissa oli mahdollista valita päivämäärä valikosta, mutta halusin testata vain päivämäärän kirjoittamista.

OPUX-kalenterikomponentti

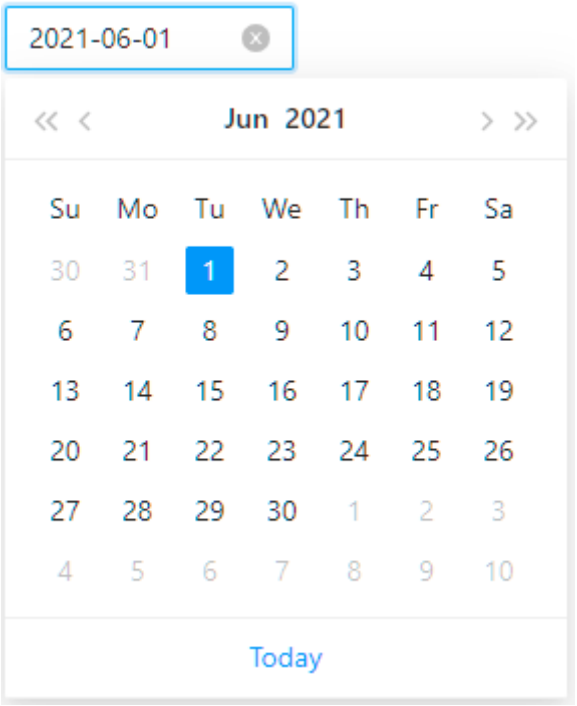
OPUX-kalenterikomponentissa (kuva 1) onChange-tapahtuma suoritettiin selaimen kohdistuksen poistuessa syötteestä. Tämä oli ilmennyt ongelmalliseksi aiemman viikon töissä.



KUVA 1. OPUX-kirjaston kalenteritoteutus

Ant Desing kalenterikomponentti

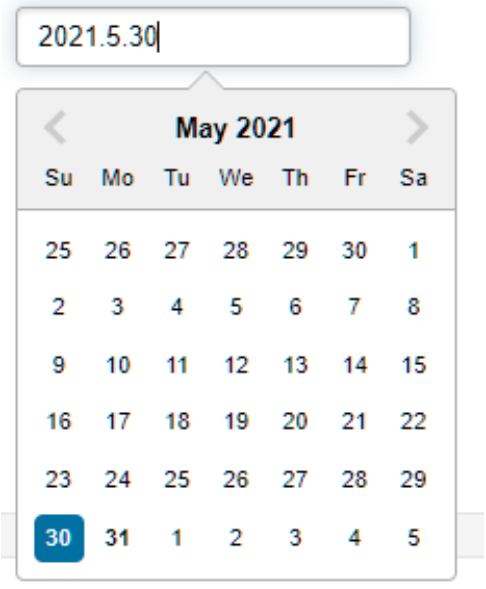
Ant Design kirjaston kalenterikomponentti (kuva 2) suoritti onChange-tapahtuman ainoastaan Enter-painalluksella. Hiirellä pois klikattaessa kirjoitettu arvo pyyhkiytyi pois. Ant Designin toteutus oli mielestäni epäkäytännöllisempi kuin OPUX-kirjaston.



KUVA 2. Ant Design-kirjaston kalenteritoteutus (2)

HackerOnen DatePicker-komponentti

Seuraavaksi testasin HackerOnen DatePicker-komponenttia (kuva 3). Komponentti lähetti arvonsa eteenpäin jokaisella painikkeen painalluksella. Koin tämän käytännöllisimmäksi toteutukseksi kaikista kolmesta.



KUVA 3. HackerOnen kalenteritoteutus (3)

HackerOnen toteutus oli mielestäni loogisin tapa toteuttaa kalenterisyöte. Kun komponentit sisältävät perinteisen syötteen, jokaisella kirjoituksella päivittyvä toteutus on selkein. Ant Designin toteutuksen koin hyvin epäkäytännölliseksi, ja erityisesti päivämäärän pyyhkiytyminen oli hyvin hämäävä. OPUX-kirjaston toteutus aiheutti ongelman, joka jouduttiin ohittamaan muuttamalla itse komponentin toimintaa. OPUXin toteutus oli kuitenkin huomattavasti toimivampi kuin Ant Designin. OPUXin kanssa ilmennyttä ongelmaa ei kuitenkaan olisi ilmentynyt käytettäessä HackerOne:n toteutusta.

3.2 Toinen viikko

Maanantai 17.5.2021

Maanantaina jatkoin linkkiongelman tutkimista. Vertaillessa ympäristöjen sovellusten versioita ilmeni, että testiympäristössämme linkattava sovellus oli vanhempi, kuten ajattelin. Sovelluksen päivittäminen testiympäristöömme ei kuitenkaan edelleenkään tuonut virhettä esiin. Lisäksi testausta hidasti ongelma sovelluksessamme, joka esti kutsujen lähettämisen sovelluksesta eteenpäin. Tällöin linkkiin ei päästy käsiksi sovelluksen kautta, vaan se piti luoda käsin. Päätin perehtyä tarkemmin linkattavan sovelluksen toteutukseen ja muutoksiin.

Tiistai 18.5.2021

Linkki-ongelman tutkiminen jatkui. Meidän sovelluksemme kutsuongelman syyksi ilmeni virheellinen rajapinta-avain.

Vertasin sovelluksemme linkitystoteutusta erään toisen sovelluksen vastaavaan toteutukseen. Sovelluksesta linkattaessa salattu parametri oli eri kuin meidän toteutuksessamme. Epäilin, että meidän sovelluksessamme käytetään väärää salattua arvoa parametrinä. Lisäksi ilmeni, että mikäli sovelluksen haluaa avata normaalista poikkeavilla tiedoilla, tulee käyttää poikkeavaa polkua. Olin yhteydessä toiseen tiimiin selvittääkseni, mitä arvoa tässä tilanteessa kuuluu käyttää.

Koodin julkaisutapa oli uudistunut, joten tutustuimme uuteen julkaisuprosessiin tiimimme kehittäjien kesken.

Keskiviikko 19.5.2021

Sain vastauksen toiselta tiimiltä ja epäilykseni olivat oikeassa. Kun sovellus haluttiin avata näillä normaalista poikkeavilla tiedoilla, tuli parametrinä käyttää eri arvoa.

Selvitin, onko arvo saatavilla sovelluksemme toteutuksessa. Arvoa ei löytynyt viitatun nimisenä, mutta löysin lähes samalla nimellä löytyvän arvon. Totesimme kehitystiimin kanssa arvon mitä luultavimmin olevan tarvitsemamme.

Meidän nykyisessä toteutuksessamme arvo salattiin käyttäen väärää salaustoteutusta. Arvoa ei myöskään liitetty linkkiin sitä tarvittavassa tilanteessa. Muutin arvon salattavaksi oikealla algoritmilla, sekä muutin linkin muodostavan toteutuksen käyttämään arvoa tilanteessa, jossa sitä tarvitaan.

Toteutusta ei ollut mahdollista testata ilman testiympäristöä, sillä meillä ei ole mahdollista ajaa linkattavaa sovellusta paikallisesti. Toteutus jouduttiin viemään projektinhallintaan, jotta saimme sen testattavaksi testiympäristöömme.

Torstai 20.5.2021

Toteutus vietiin testiympäristöön ja sovelluksessamme ilmeni tästä johtuen ongelmaa. Sovelluksen rajapintakutsut palauttivat virhettä, kun käytettiin muutokseen liittyneitä tietoja. Tästä syystä sovelluksen lomaketta ei voitu lähettää ja linkkiin ei päästy käsiksi. Kun linkki muodostettiin käsin käyttäen salattua parametriä, linkattu sovellus aukesi oikein ja parametrin todettiin olevan oikein salattu.

Tiimillämme oli sprintin suunnittelupalaveri, joka vei suurimman osan päivän työajasta. Tästä johtuen en ehtinyt perehtymään syvemmin ongelmaan. Sain kuitenkin selville, että salattua arvoa yritetään edelleen purkaa käyttämällä väärää algoritmia.

Viikkoanalyysi

Normaalista poiketen en juurikaan työskennellyt sovellusten selaintoteutusten parissa kuluneen kahden viikon aikana. Työpäivät kuluivat pääsääntöisesti linkkiongelman tutkimisessa. Tutustuin paljon itselleni täysin tuntemattomaan Java-

palvelintoteutusten koodiin. Opin uutta siitä, miten testausympäristöjen sovellusten versioita sekä lokitiedostoja päästään tutkimaan. Uutena minulle tuli myös sovellusten versioiden muuttaminen testiympäristöön.

3.3 Kolmas viikko

Maanantai 24.5.2021

Aloitin työpäiväni suorittamalla kykytestit työnantajani pyynnöstä. Olen siirtymässä toistaiseksi voimassa olevalle työsopimukselle, joten minun haluttiin suorittavan nämä testit.

Tämän jälkeen siirryin korjaamaan ongelmaa, joka ilmeni muutoksessani aiemmalla viikolla. Olin jättänyt huomiotta, että salattavan arvon salauksenpurku tulee myös muuttaa. Tutkin linkattavan sovelluksen salauksenpurkutoteutusta ja toteutin samanlaisen meidän tiimimme sovellukseen.

Tiistai 25.5.2021

Testasin salauksenpurkumuutoksen testiympäristössämme ja sovellus toimi ilman virheitä. Vein muutoksen myös julkaisutestausympäristöön. Päivällä pidimme tiimin kesken palaveria eräästä versiomuutoksesta, joka sovelluksiimme on tulossa. Tämän jälkeen päivittelin erään sovelluksemme selaintoteutuksen kirjastojen versiot uusimpiin.

Daily-palaverissa tuli ilmi, että erään sovelluksen automaatiotesteissä oli tullut epäonnistumisia. Sovelluksen rajapintakutsut olivat epäonnistuneet. Osa virheistä viittasi minun tekemääni muutokseen, joten otin ongelman tutkimisen seuraavaksi työn alle. Toistaessani virhettä tekemääni muutokseen liittyvät rajapintakutsut onnistuivat mutta rajapintakutsuissa tuli kuitenkin muutamia muita virheitä. Selvittelin asiaa sovelluksen lokeilta ja ilmeni, että erästä rajapintaa varten generoidussa asiakassovelluksessamme oli puutteita eräissä vasta-arvoissa. Rajapinta palautti arvon, jolle meidän asiakassovelluksessamme ei ollut vasta-arvoa. Olin yhteydessä rajapinnan toteuttaneeseen tiimiin selvittääkseni, mikä arvo on kyseessä ja löytyykö siihen liittyvää dokumentaatiota. Vastausta odotellessa yritin itse selvitellä asiaa.

Keskiviikko 26.5.2021

En ollut saanut vastausta toiselta tiimiltä. Päätin lisätä vasta-arvon asiakassovelluksemme toteutukseen. Muutos oli hyvin yksinkertainen ja lisäsin vain yhden muuttujan koodiin. Enemmän aikaa kului kuitenkin siihen, kun vein muutoksen projektinhallintaan, päivitin asiakastoteutusta käyttävän sovelluksen version ja vein vielä tämän muutoksen projektinhallintaan. Kun muutos oli testattu, vein sen vielä julkaisutestausympäristöön. Sovelluksen koonnissa julkaisutestausympäristöön ilmeni ongelmia. Jenkins-koontijono antoi virhettä jo koonnin alkuvaiheessa. Virhe ei näyttänyt liittyvän meidän sovellukseemme, joten selvittelin sitä julkaisutestausympäristöstä vastaavalta tiimiltä.

Luppoaikana tutustuin tiimimme Robot-automaatiotestaustoteutuksiin. Seuraavana päivänä oli tarkoitus pystyttää minulle ympäristöt Robot Frameworkin ajoa varten tiimimme testiautomaatiovastaavan kanssa.

Torstai 27.5.2021

Torstaina palaveroin automaatiotestikehittäjän kanssa testaustyökalu Robot Frameworkin käyttöönotosta. Käyttöönotossa ilmeni pieniä ongelmia Python-versioiden ja pakettien asennuksen kanssa. Kävimme myös läpi hyviä käytäntöjä testien kirjoittamisessa ja tutustuimme testikoodeihin.

Meidän sovelluksessamme oli ongelmia julkaisutestausympäristössä. Sovellus ei tullut ollenkaan näkyviin ja esille tuli ainoastaan virhesivu. Yritin selvitellä asiaa lokeilta, mutta en löytänyt sovelluksen lokeja yhdeltäkään palvelinkoneelta. Kaikkien testiympäristöjen palvelinkoneet ovat saatavilla kehittäjille. Koneet on kirjattu sisäiseen wikijärjestelmään, mutta usein ei ole täysin selvää, mikä kone on osa mitäkin testiympäristöä. Sovellukset on myös jaoteltu ympäristön sisällä eri koneisiin, joten ympäristön sisältä tuli löytää myös oikea kone. Tutkin useampaa konetta mutten löytänyt sovelluksen lokeja. Päätin tutkia Jenkins-ohjelmiston sovelluskoonnin luomia lokeja ja sieltä ilmeni, että sovellusta ei olisi koottu testiympäristöön ollenkaan. Olin yhteydessä testiympäristöistä vastaavaan tiimiin ja he selvittivät asiaa. He ajoivat sovelluksen koonnin testiympäristöön uudelleen ja sovellus alkoi toimia normaalisti. Iltapäivän perehdyin tarkemmin Robot Frameworkin käyttöön ja ensimmäisen testin luontiin.

Viikkoanalyysi

Kulunut viikko oli hyvin hajanainen, eikä minulla ollut yhtä selkeää tehtävää, jota työstää. Suoranaista koodaustyötä tein hyvin vähän ja suurin osa työajasta kului tutkimiseen ja selvittelyyn. Pääsin taas tutkimaan sovellusten lokitiedostoja, sekä uutena aiheena perehdyin Jenkins-ohjelmiston lokeihin. Jouduin myös muutamaan otteeseen selvittelemään asioita tiimimme ulkopuolelta. Mukavaa vaihtelua selvittelyyn toi torstain Robot-testien pystytys ja perehtyminen. Olen aiemmin työskennellyt hieman Robot-testien parissa ja olen kiinnostunut taas pääsemään niiden pariin.

3.4 Neljäs viikko

Maanantai 31.5.2021

Aamupäivällä ohjeistin työkaveriani Jest-testausviitekehyksen käytössä. Erääseen komponenttiin oli tehty päivitys ja testit olivat hajonneet. Testeissä ongelmaa aiheutti Enzyme-testaustyökalun setProps-metodi. Emme saaneet setPropsia toimimaan halutulla tavalla. Metodin avulla tulisi voida manipuloida React-komponenteille annettavia arvoja, jotta voidaan testata komponentin reagointia näihin muutoksiin. Ongelman syyksi ilmeni se, että testattavaan komponenttiin oli lisätty kieleistys. Tällöin komponentti joudutaan ympäröimään MessageProvider-komponentin sisään, joka tarjoaa komponentille kielimuuttujat. Enzymen setProps-metodi toimii siten, että se ajaa setPropsin komponenttipuun ensimmäiselle komponentille, joka tässä tilanteessa oli MessageProvider eikä testattava komponentti. Työkaverini joutui muuttamaan testin toteutuksen täysin, sillä setProps-metodia ei ollut mahdollista käyttää.

Päivällä ehdin tutustumaan lisää Robot-automaatiotestien toteutukseen. Tehtävänäni oli toteuttaa testi, joka tarkistaa erään lomakkeen tiedon olevan oikealla kielellä. Testissä tulee luoda lomakkeella tieto ja poistaa se sen jälkeen, jotta testi on mahdollista suorittaa uudelleen. Testi on riippuvainen siitä, että tietoa ei ole luotu aiemmin, jotta lomake tarjoaa erästä painiketta painettavaksi. Yritin testailla toiminnallisuutta käsin, mutta minulla oli ongelmia saada painiketta näkyviin. Painikkeen tulisi ilmestyä näkyviin, kun kirjoitetaan tieto syötteeseen. Sovellus antoi

kuitenkin virheen ja painike ei ilmestynyt. Yritin poistaa mahdollisia olemassa olevia tietoja, mutta en saanut painiketta näkyviin. Toteutin kuitenkin testin lisäämään tiedot lomakkeeseen.

Etätyöaikana tiimimme on pitänyt ”perehtymiskahveja” uusien tiimiläisten kanssa. Uudet tiimiläiset ovat varanneet 30 minuutin ajan jokaisen tiimiläisen kalenterista ja olemme rupatelleet Teams-puhelussa. Tänään minulla oli perehtymiskahvit tiimimme uuden tuoteasiantuntijan kanssa.

Iltapäivällä pidimme pidennetyn daily-palaverin. Tiimimme eräs kehittäjä oli vaihtamassa työpaikkaa ja pidimme hänelle läksiäiskahvit.

Tiimimme toinen kehittäjä pyysi minua tutkimaan erästä ongelmaa. Erääseen uuteen ikkunaan aukeavassa sovelluksessa oli ilmestynyt näkyviin yläpalkki. Palkki tulee Osuuspankin verkkosovelluksen yleisteemasta ja se tulisi piilottaa sovelluksesta.

Tiistai 1.6.2021

Perehdyin tarkemmin yläpalkin piilotukseen sovelluksesta. Sovellus on vanhempaa toteutusta ja se on toteutettu käyttäen jQuery-kirjastoa. Teknologia ei ollut minulle tuttu, joten minun täytyi tutustua sovelluksen toteutukseen tarkemmin. Sovellusta ei ollut mahdollista ajaa paikallisesti, joten jouduin tekemään muutoksen, viemään muutoksen projektinhallintaan ja testaamaan sen testausympäristössämme. Yritin piilottaa yläpalkkia muutamalla eri tavalla. Testasin jQueryn hide-metodia, jonka tulisi piilottaa annettu elementti, mutta palkki ei piiloutunut. Tämän jälkeen kokeilin asettaa yläpalkin CSS-luokan näkyvyyden piilotetuksi käyttäen jQuerya. Sekään ei toiminut. Tämän jälkeen havahduin, että voisin kokeilla lisätä sovelluksen HTML-koodiin suoraan tyylitagin alle saman CSS-määrityksen, jota yritin suorittaa jQuerylla. Tämän jälkeen yläpalkin sai piiloon. Havahduin, että sovellukseen oli ilmestynyt myös alapalkki. Selvitin tiimimme tuotteenomistajalta, kuuluisiko sekin piilottaa. Sain myönteisen vastauksen ja piilotin alapalkin samalla tekniikalla.

Iltapäivällä ehdin jatkaa Robot-testejä. Nyt onnistuin saamaan painikkeen esiin ongelmitta ja lisäsin testiin painikkeen painalluksen sekä aloitin toteutusta tekstin tarkistukselle.

Keskiviikko 2.6.2021

Keskiviikkona toteutin Robot-testiin kaikki lomakkeen lähetykseen liittyvät osat: tekstien lisäykset, painikkeen painalluksen, tekstin tarkistuksen sekä lomakkeen lähetyksen. Lomakkeen lähetyksen jälkeen tuli myös tarkistaa, että lomakkeen lähetys onnistui, joten lisäsin tarkistuksen myös sille. Töitäni hidastivat ajoittaiset kirjautumisongelmat testiympäristöömme, sillä Robot-testit suoritetaan tiimimme testiympäristöä vasten. Toisena hidasteena oli se, että lomakkeella lähetettävä tieto tulee aina poistaa käsin testin ajon jälkeen. Seuraavaksi minun tulisi toteuttaa tämän tiedon poisto automaatiotestillä.

Torstai 3.6.2021

Tiimillämme oli sprintinsuunnittelupalaveri. Suurin osa päivästä kului retrospektiivissä sekä tulevan sprintin suunnittelussa. Iltapäivällä minulla oli hetki aikaa jatkaa Robot-testejä. Sain lomakkeella lisätyn tiedon poistamisen automatisoiduksi.

Viikkoanalyysi

Viikkoni kului Robot-testien parissa, kuten viime viikolla odotin. Tein myös yhden korjauksen tiimimme vanhemman sovelluksen toteutukseen. Sovelluksen toteutus ja jQuery olivat minulle uutta. Minut yllätti se, että sovelluksen selaintoteutusta ei ollut mahdollista testata paikallisesti. Jouduin viemään testaamatonta koodia versionhallintaan, jotta sitä päästiin testaamaan. En tiennyt vanhempien sovellustemme selaintoteutusten ylläpidettävyyden olevan näin haastavaa.

Olen päässyt hyvin vauhtiin Robot-testien parissa ja vanhat osaamiset Robotin parissa ovat alkaneet palautua. Työläin osuus Robotin kanssa työskentelyssä on oikeanlaisten paikantimien rakentaminen. Robotissa käytetään HTML-elementtien hakuun XPath-kyselykieltä. Hakuun käytetään HTML-elementtien nimiä ja attribuutteja, mutta monimutkaisemmissa tapauksissa joudutaan paikantimet ra-

kentamaan käyttäen XPathin akseleita (axes). Näiden avulla voidaan hakea elementin ympäröiviä elementtejä, jos tarvittavaa elementtiä ei voida hakea esimerkiksi luokan nimen perusteella (5).

3.5 Viides viikko

Maanantai 7.6.2021

Eräässä sovelluksessamme on käytössä OPUX-kirjaston kalenterikomponentti. Komponentissa on automaattisesti arvo: “TÄNÄÄN”, mutta se hyväksyy normaalien päivämäärien lisäksi myös arvot nyt, heti ja huomenna. Tilanteessa, jossa kalenteriin kirjoitetaan arvo “nyt” tai “heti”, muuttuu kalenterin arvo “TÄNÄÄN”-muotoon. Toiminnallisuus voi olla hämäävä käyttäjälle. Tarkoituksena olisi muuttaa kalenteri toimimaan siten, että teksti pidettäisiin käyttäjän antamana, jos annetaan jokin näistä sallituista termeistä.

Kalenterikomponentti tarjoaa formatToday-arvon, jota muuttamalla voidaan muuttaa syötteen tekstiä, kun päivämääräksi on valittu sen hetkinen päivä. Päätin kokeilla toteuttaa toiminnallisuutta vaihtamalla formatToday-arvoa silloin, kun komponentin onChange-tapahtumassa havaitaan jokin sallituista teksteistä. Ongelmaksi ilmeni kuitenkin, että teksti muuttui yhden renderöinnin myöhässä. Arvo siis muuttui vasta sen jälkeen, kun syötteeseen annettiin jo uusi arvo. Tutkin asiaa jonkin aikaa, mutta en onnistunut ratkaisemaan ongelmaa, joten siirryin muihin tehtäviin. Katselmoimme automaatiotestikehittäjän kanssa tekemiäni Robot-testejä. Ilmeni, että olimme toteuttaneet yhtä aikaa samantapaisia testejä ja olin toteuttanut omat testini eri tiedostoon. Yhtenäistin meidän testimme. Testeissä ilmeni vielä ongelmia erään tiedon kanssa. Tietoa syötettäessä tulisi erään painikkeen tulla näkyviin. Annetulla tiedolla tuli kuitenkin esiin virhettä. Vaihdoin syötettävän tiedon ja painike saatiin näkyviin. Tällöin kuitenkin muuttuivat syötettävät tiedot, joita vaaditaan painikkeen painalluksen jälkeen, joten jouduin tekemään pieniä muutoksia nykyiseen toteutukseen. Lisäsin muutokset katselmoitavaksi.

Tiistai 8.6.2021

Jatkoin kalenterikomponentin tutkimista. Tarkemmin dokumentaatiota tutkittuani huomasin, että OPUX-kirjaston kalenterikomponentti sisältää formatFunction-nimisen apufunktion. Funktiolla saadaan tietoon päivämäärä ja päivämäärän tekstimuoto aina kalenterin arvon muuttuessa. Funktio palautti myös tekstit “nyt”, “heti” ja “huomenna”, kun ne syötettiin kalenteriin. Toteutin funktion palauttamaan syötetyn arvon silloin, kun se vastasi jotain näistä sallituista termeistä. Tällä sain ratkaistua eilisen ongelman ja syötetty teksti saatiin säilytettyä toivotulla tavalla. Toteutin toiminnallisuuteen myös testejä, jotka tarkistavat, että syötetyt termit eivät nosta virhettä. Yllätyksekseni huomasin tekstien nostavan virheen. En kuitenkaan saanut selville, mistä ongelma johtui.

Keskiviikko 9.6.2021

Jatkoin eilen toteuttamiini testeihin liittyvän ongelman tutkimista. Syyksi selvisi se, että renderöin testissä pelkkää kalenterikomponenttia hallinoivaa komponenttia enkä koko sovellusta hallinoivaa komponenttia. Kalenterikomponenttia hallinoivalla komponentilla on riippuvuus ylemmän komponentin funktioon, joka tarkistaa syötetyn päivämäärän sallittavuuden. Funktion puuttuessa testistä komponentti salli vain nykyisen päivämäärän. Muutin testin renderöimään koko sovellusta hallinoivan komponentin ja virheitä ei enää noussut. En ollut aiemmin toteuttanut Jest-viitekehyksellä tarkistuksia sille, että jokin elementti ei ole näkyvissä. Jouduin käyttämään aikaa dokumentaation tutkimiseen, mutta lopulta tarkistuksen sai toteutettua yksinkertaisesti käyttämällä jest-dom-kirjastoa. Jest-dom tarjoaa “toBeInTheDocument”-vertailijan, joka etsii annettua elementtiä HTML-koodista. Kun vertailijan kanssa käytettiin ”not”-metodia oli mahdollista varmistaa, ettei virhettä ole HTML-koodissa.

```
expect(queryByText(”Virhe”)).not.toBeInTheDocument();
```

Torstai 10.6.2021

Aiemman korjaamaani ylä- ja alapalkin piilotusmuutosta varten tuli toteuttaa automaatiotestit. Suunnittelin tarkistavani Robotilla, että ylä- ja alapalkki löytyvät sivun html-koodista mutta eivät ole näkyvissä. Sovellus, josta palkit piti tarkistaa, avautuu uuteen ikkunaan. Robot Framework ei huomaa uuden ikkunan aukeamista automaattisesti, joten sille pitää kertoa ikkuna, johon kohdistaa. Yritin toteuttaa tämän käyttäen Select Window-avainsanaa. Jostain syystä tämä kuitenkin herätti virheen. Avainsana näytti olevan käytössä myös muualla automaatiotestiemme lähdekoodissa. Vertailin omaa käyttötapaani muualla lähdekoodissa käytettyyn eikä siinä ollut eroa.

Huomasin, että lähdekoodissa Select Window-avainsanaa käyttävät avainsanat, eivät olleet käytössä yhdessäkään testissä. Tutkin Robot Frameworkin dokumentaatiota ja huomasin, että dokumentaatio oli vanhemmalle versiolle. Uusimmassa Robot versiossa avainsana oli muuttunut Switch Window'n. Tällä avainsanalla sain ikkunan vaihtamisen toimimaan. Muodostin paikantimet ylä- ja alapalkille sekä toteutin tarkistukset sille, että ne löytyvät HTML-koodista eivätkä ole näkyvissä.

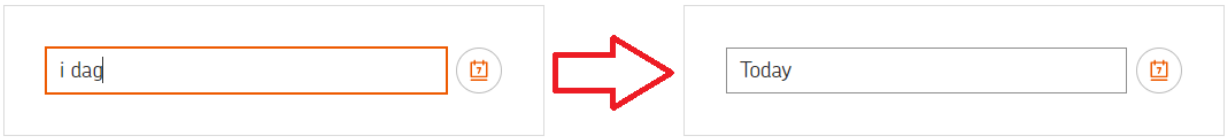
Viikkoanalyysi

Tällä viikolla pääsin pitkästä aikaa työskentelemään React-pohjaisen sovelluksen sekä OPUX komponenttien parissa. OPUXin kalenterikomponentin formatFunction yllätti minut positiivisesti. Se helpotti ongelman ratkaisua suuresti. Kuluneella viikolla hankalaksi OPUXin parissa huomasin kalenterin sallitut termit: "Tänään", "huomenna", "heti" ja "nyt". OPUX-dokumentaatio ei kertonut, mitä nämä termit ovat eri kielillä. Termien mainittiin olevan "Now", "Today" ja "Tomorrow" mutta ei kerrottu, mitkä ne ovat ruotsiksi tai suomeksi. Suomen "heti"-termistä ei ollut mitään mainintaa. Jouduin varmistamaan sallitut termit kalenterikomponentin lähdekoodista.

OPUX kalenterikomponentin “sallitut termit”-toiminnallisuuden analyysi

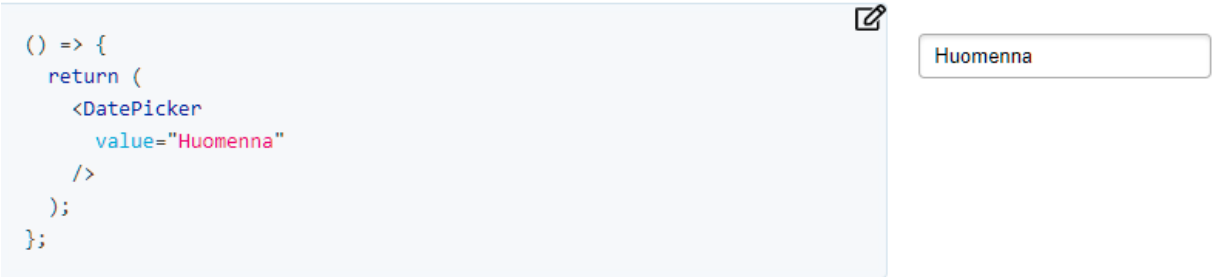
Halusin vertailla OPUX kalenterikomponentin ”sallitut termit”-toiminnallisuutta avoimen lähdekoodin vastineisiin. Valitsin vertailukohteiksi HackerOnen React-Datepicker-komponentin sekä react-day-picker-komponentin.

OPUX-kalenterikomponentti mahdollisti päivämäärän valitsemisen termeillä ”Today”, ”Tomorrow” ja ”Now” sekä samat suomeksi ja ruotsiksi. Syötteen teksti muuttui kuitenkin aina muotoon ”Today”, vaikka syötteeseen olisi kirjoitettu ”i dag” (kuva 4) tai “tänään”. Komponentti tarjosi formatFunction-metodin, jonka avulla syötteessä voitiin näyttää tekstejä päivämäärän sijaan. Tällä termien toiminnallisuutta voitiin laajentaa. Logiikan kehittäjä joutui kuitenkin tekemään itse.



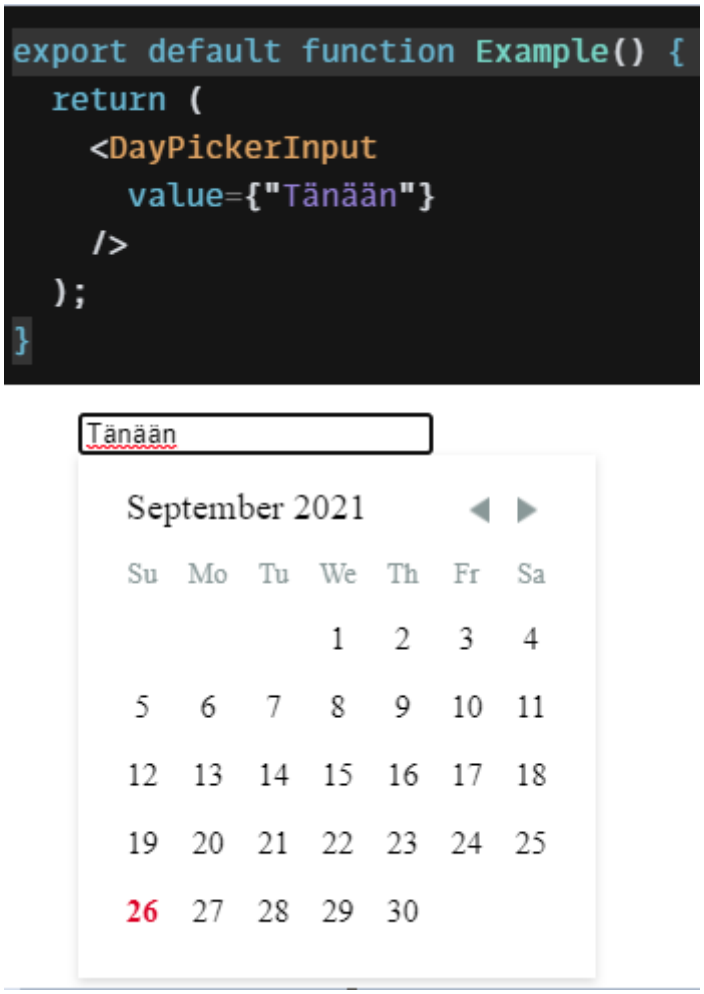
KUVA 4. OPUX kalenterikomponentti, kun syötetään termi "i dag".

HackerOnen Datepicker ei tarjonnut valmiina samantapaista toiminnallisuutta. Dokumentaatiossa esimerkit kuitenkin esittelivät tilanteen, miten päivämäärä voidaan asettaa syötetyn tekstin mukaan. Syötteen näyttämää arvoa voidaan myös käsitellä ”value”-propin arvolla (kuva 5). Tällöin olisi itse mahdollista laajentaa komponentin toiminnallisuutta siten, että tietyissä tilanteissa näytettäisiin tekstejä päivämäärien sijaan. HackerOnen toteutus oli suppeampi ja vaati kehittäjältä enemmän komponentin muokkaamista halutun toiminnallisuuden saavuttamiseksi.



KUVA 5. HackerOne kalenterikomponentin tekstin kustomointi. (3)

React-day-picker ei myöskään tarjonnut OPUX kalenterikomponenttia vastaavaa toiminnallisuutta. React-day-pickerin toiminnallisuus oli hyvin pitkälti vastaavanlainen HackerOnen toteutuksen kanssa. React-day-pickerin dokumentaatiosta ei kuitenkaan löytynyt mitään tilanteeseen sovellettavia esimerkkejä. Komponentti kuitenkin tarjosi saman ”value”-propin kuin HackerOnen komponentti, jolloin toiminnallisuutta olisi mahdollista laajentaa samaan tapaan (kuva 6).



KUVA 6. react-day-pickerin tekstin kustomointi. (4)

OPUX-kirjaston toteutus on paljon la ajempi ja yksityiskohtaisemmaksi räätälöity kuin avoimen lähdekoodin vastineet. OPUXin kalenterikomponentti on räätälöity Osuuspankin kehittäjien tarpeisiin ja laajempi toteutus takaa sen, etteivät tiimit lähde tekemään omia ratkaisuja vaan kaikki kalenterikomponenttiratkaisut pysyvät yhtenäisinä. Avoimen lähdekoodin vastineet taas on tarkoitettu pohjatoteutuksiksi, koska niiden käyttötarkoitusta ei haluta rajata liian suppeaksi.

3.6 Kuudes viikko

Maanantai 14.6.2021

Maanantai oli normaalista poikkeava päivä, sillä vietimme läksiäisiä tiimin kanssa, josta olin poistunut vuodenvaihteessa. Läksiäisiä siirrettiin koronapandemian takia, mutta nyt oli mahdollista kokoontua viettämään virkistyspäivää. Tein aamulla töitä vain muutaman tunnin. Tutkin tilannetta, jossa sovellus tuo esiin sovelluksesta poistumisesta varoittavan ilmoituksen painettaessa “takaisin ylös”-painiketta sovelluksen alapalkista. ”Takaisin ylös”-painike ei kuitenkaan poistu sovelluksesta vaan vierittää selaimen ikkunaa ylöspäin. Ilmoituksen on tarkoitus tulla esiin vain silloin kun painetaan toiselle sivulle vieviä linkkejä. ”Takaisin ylös”-painikkeessa käytetään HTML-linkkielementtiä, jonka osoitteena on pelkkä “#”. Tämä vie automaattisesti silloisen sivun alkuun. Tämä kuitenkin herättää sivulta poistumista valvovan toiminnallisuuden ja tuo esiin ilmoituksen.

Tiistai 15.6.2021

Jatkoin ”takaisin ylös”-painikkeen ongelman tutkimista. En ollut aikaisemmin perehtynyt sovellukseen, jossa ongelma ilmeni. Jouduin siis käyttämään paljon aikaa lähdekoodin opiskeluun. Lopulta löysin sivulta poistumista vahtivan toiminnallisuuden. Ongelman pystyi korjaamaan lisäämällä painikkeen käyttämän linkin listaan, jotka eivät herätä sivulta poistumistoiminnallisuutta.

Tein myös lisämuutosta kalenterikomponenttimuutokseen, jossa haluttiin säilyttää teksti, kun annettiin jokin sallituista termeistä. Testaajan mielestä tekstiä ei tulisi säilyttää vaan siitä pitäisi nousta virhe, jos se on syötetty muulla kuin valitulla kielellä. Muutin toiminnallisuuden säilyttämään vain käytössä olevan kielen termit.

Testatessa huomasin kuitenkin, että väärän kielen termit eivät herättäneet virhettä, toisin kuin oli oletettu. Kerroin testaajalle, ettei vääränkielisestä päivästä nouse virhettä, sillä se on peräisin OPUX-kirjaston toteutuksesta. Kalenterikomponenttiin on OPUX-kirjaston toteutuksessa määritetty jo sallitut termit, jotka eivät ole riippuvaisia käytettävästä kielestä. Otimme asian puheeksi daily-palaverissa ja totesimme tiimin kesken, että väärällä kielellä kirjoitettua termiä ei ainakaan

tule säilyttää, mutta sen olisi hyvä nostaa virhe. Päätimme tehdä OPUX-kirjastolle kehitysehdotuksen aiheesta ja muuttaa meidän toteutustamme siten, että väärällä kielellä annettu termi muuttuu päivämääräksi. Loppupäivästä aloitin automaatiotestejä ”takaisin ylös”-painikkeeseen liittyen.

Keskiviikko 16.6.2021

Jatkoin testejä liittyen ”takaisin ylös”-painikkeeseen. Toteutin testin siten, että ensin yritetään poistua sivulta linkin kautta, jotta ilmoitus saadaan esiin. Tämän jälkeen klikkasin ”takaisin ylös”-painiketta ja vertasin, tuleeko ilmoitus esiin. Haastavin osuus testin toteutuksessa oli luoda paikannin ilmoituksen takaisinpainikkeelle. Jostain syystä painiketta löytyi kaksi kappaletta sivun HTML-koodista eikä paikantaminen pelkästään painiketta etsimällä onnistunut. Jouduin ensin paikantamaan ilmoituksen sen sisältävän tekstin perusteella. Tämän jälkeen ilmoituksen sisältä etsittiin tekstin viereinen komponentti, jonka sisältä painike löytyi.

Tein myös kehitysehdotuksen OPUX-kirjaston Jiraan, liittyen kalenterikomponenttien termeihin ja niiden rajoittamiseen tiettyyn kieleen. Iltapäivällä oma työlistani alkoi näyttämään tyhjältä, joten kysyin automaatiotestaajalta jotain hänen tehtävistään itselleni. Hän oli ollut kiireellinen kuluneella viikolla tiedonsiirrosta johtuen. Häneltä löytyi yksi testi, jota pystyin alkaa toteuttamaan.

Torstai 17.6.2021

Suurin osa päivästä kului tulevan sprintin suunnittelupalaverissa. Iltapäivällä ehdin aloittaa testiautomaatiototeutusta, jonka otin työkaveriltani työn alle. Toteutuksella tulisi testata, että erään sovelluksen syötteiden lähetykset toimivat onnistuneesti tietyillä tiedoilla.

Viikkoanalyysi

Kuluvalla viikolla perehdyin taas uuteen jQuery-pohjaiseen sovellukseen. Minulla ei ollut aiempaa kokemusta sovelluksesta ja jQuerystäkin vain hiukan aiemmilla viikoilla, joten jouduin käyttämään aikaa taas koodin opiskeluun. Pääsin kuitenkin nopeasti jyvälle siitä mistä ongelma johtui ja sain ongelman korjatuksi.

Työskentelin viikon aikana myös hiukan OPUX-komponenttien parissa, erityisesti kalenterikomponentin. Testaaja oli havainnut puutteen meidän kalenterikomponenttimme käyttötilanteessa. Lopulta tämä kuitenkin ilmeni puutteeksi itse kalenterikomponentissa, sillä toiminnallisuutta ei ollut mahdollista toteuttaa sellaiseksi kuin testaaja sen toivoi. Teimme asiasta kehitysehdotuksen OPUXia kehittävälle tiimille. Loppuviikosta tein automaatiotestejä.

3.7 Seitsemäs viikko

Maanantai 21.6.2021

Jatkoin aiemmalla viikolla aloitettua automaatiotestiä. Testissä tarvittiin erästä tietoa, joka ei toiminut oikein. Ongelmasta oli tehty jo erillinen tehtävä. Otin myös tämän itselleni työn alle, sillä se korjaantuisi nykyisten töiden mukana. Vaihdoin viallisen tiedon toiseen vastaavaan ja muutin automaatiotestit toimimaan oikein, kun tieto lisätään sovellukseen. Sovellus muuttaa vaadittavia syötteitä tämän tiedon mukaan, joten automaatiotestit pitää toteuttaa näiden tietojen pohjalta. Tilanteesta riippuen tiedon syöttämisen jälkeen voidaan vaatia yhtä tai kahta lisätietoa syötettäväksi. Toteutin robotilla metodin, joka pystyy hallitsemaan tilanteen riippumatta siitä, tuleeko tietoa syöttää yhteen vai kahteen syötteeseen. Toteutettiin myös tarkistukset sille, että tiedon lähetys onnistui sekä sille ettei samoilla tiedoilla olevaa lomaketta voi enää lähettää uudelleen.

Tiistai 22.6.2021

Tiistaina tein vielä jatkomuutosta liittyen kalenterikomponentin “tänään”-tekstin säilyttämiseen. Pohdimme toiminnallisuutta tiimimme kanssa ja totesimme, että kalenterikomponentin tulisi säilyttää vain silloisen käytössä olevan kielen termit: “tänään”, “huomenna”, “heti” ja “nyt”. Muista kielistä tulisi muuttaa päivämäärämuotoon. Muutos oli hyvin yksinkertainen. Muutin aiempaa tekstin sallittavuuden päättelevää toiminnallisuutta siten, että rajasin sallittavat tekstit kielen mukaan. Näin väärällä kielellä annettu termi muutettiin automaattisesti päivämääräksi. Yksikkötestipuolella muutos oli suurempi. Kaikille kielille tuli toteuttaa testit tarkistamaan, että termit säilytetään oikein ja että väärillä kielillä termejä ei säilytetä.

Myös erääseen toiseen sovellukseen piti toteuttaa esto ”takaisin ylös”-painikkeen aiheuttamalle keskeytykselle. Painikkeen painaminen avaa ilmoituksen, joka varoittaa, että ollaan poistumassa sivulta, vaikka näin ei ole tapahtumassa. Korjaus tapahtui samalla tavalla kuin aiemmin. Painikkeen käyttämä linkki lisättiin keskeytysilmoituksen ohituslistalle.

Keskiviikko 23.6.2021

Alapalkin ylös-painikkeen korjausta varten toteutettiin myös automaatiotestit. Toteutus oli hyvin samanlainen kuin aiemmassa testissä. Sovellukseen syötettiin tieto ja painettiin painiketta, jonka jälkeen tarkistettiin, että ilmoitus ei aukea. Samassa tiedostossa oli testi liittyen uloskirjautumiseen sovelluksesta, jonka paikantimet olivat vanhentuneet. Aiheesta oli tehty jo tiketti, joten korjasin tämän ongelman samalla. Testissä tarkistettiin eräs teksti uloskirjautumisen yhteydessä. Teksti oli muuttunut ja sen paikannin tuli päivittää.

Viikkoanalyysi

Viikon työni ovat olleet monipuolisia ja joka päivälle on ollut jotain uutta tehtävää. Maanantaina työskentelin Robot automaatiotestien parissa. Töissä ei tullut vastaan erityisempiä haasteita. Testasin käsin sovellusta uusilla tiedoilla ja tämän pohjalta mietin miten testit tulisi rakentaa. Suurimmalle osalle sovelluksen toiminnallisuuksista löytyi valmiit avainsanat, joita pystyi käyttämään testeissä. Esimerkiksi painikkeiden painamisille ja syötteiden täyttämiselle löytyi jo olemassa olevat avainsanat. Tein muutoksia vain niihin avainsanoihin, jotka olivat riippuvaisia näistä uusista tiedoista.

Tiistaina työskentelin React-sovelluksessa OPUX kalenterikomponentin parissa, sekä lisäksi toteutin yksikkötestejä Jest-testausviitekehyksellä. React-sovelluksen muutos onnistui minulta sujuvasti, mutta enemmän aikaa jouduin käyttämään yksikkötesteihin. Aiemmissa kalenterikomponenttiin liittyvissä testeissä testattiin nostaako kalenterikomponentti virheen sallituista termeistä. Tämän testaaminen meidän sovelluksessamme oli täysin turhaa, sillä sovelluksemme ei määrittänyt mikä termi nostaa virheen vaan se tulee täysin OPUX kalenterikomponentin toteutuksesta. Uusimman muutoksen myötä meidän toteutuksessamme oli oikeasti jotain testaamisen arvoista.

Keskiviikkona tein taas automaatiotestejä Robotilla. Tein automaatiotestit erään toisen sovelluksen ”takaisin ylös”-painike virhetilanteeseen. Samalla törmäsin virheeseen liittyen uloskirjautumiseen sovelluksesta ja korjasin myös tämän. Testin korjaaminen vaati ainoastaan uuden paikantimen muodostamisen. Paikantimen selvittäminen oli kuitenkin yllättävän hankalaa, koska haluttu toiminnallisuus oli esillä vain 15 sekuntia. Yritin monenlaisia HTML-koodin lokitus-avainsanoja selvittääkseni miltä sivun lähdekoodi näytti sillä hetkellä. En kuitenkaan saanut mitään niistä toimimaan. Tästä johtuen minulla kului rutkasti aikaa paikantimen selvittämiseen ja sen testaamiseen.

3.8 Kahdeksas viikko

Maanantai 12.7.2021

Palasin kahden viikon kesälomalta, joten minun tuli saada itseni taas ajan tasalle tiimin sen hetkisestä tilanteesta. Suurin osa tiimistäni oli myös jäänyt lomalle oman lomani aikana, joten hirveästi ei ollut ehtinyt tapahtumaan. Aloitin työt taas kevyellä muutoksella, jossa mahdollisuus kohdistaa näppäimistöllä erääseen otsikkoon tuli poistaa. Muutos tapahtui poistamalla elementistä tab-index-attribuutti. Enemmän aikaa kului muutoksen yksikkötestien toteutuksessa Jest-testausviitekehystä käyttäen. Testissä ongelmaksi ilmeni varmistuminen siitä, että otsikko ei saa kohdistusta. Jest ei tarjonnut suoraa tarkistusta sille, että jokin elementti ei ole kohdistettuna. Päätin siis toteuttaa tarkistuksen siten, että varmistetaan koko sivun saavan kohdistus, kun otsikkoon yritetään kohdistaa. Loppupäivän kulutin katselmoimalla työkaverini Robot-automaatiotesti muutoksia, joita hän oli toteuttanut useamman lomani aikana.

Tiistai 13.7.2021

Tutustuin erääseen käyttäjätarinaan, jonka tavoitteena olisi tehdä eräästä ilmoituksesta selkeämpi. Kyseessä on eräs sovellus, jonka toteutus on vanhempi ja minulle myös uutta lähdekoodia. Tutustuin sovelluksen toteutukseen ja siihen, miten ilmoitus voitaisiin saada näkyviin. Sovellusta ei ollut mahdollista testata paikallisesti, joten kaikki toiminnallisuus tuli päätellä suoraan lähdekoodista. Minulla oli hankaluuksia löytää tekstimuuttujia, joita sovelluksessa käytettiin. Kysyin asiasta työkaveriltani ja selvisi, että nämä muuttujat löytyvät toisesta sovelluksesta.

Muuttujat haetaan tästä sovelluksesta työstettävän sovelluksen käyttöön. Päästyäni selville muuttujien teksteistä sain selville, kuinka ilmoitus saadaan tuotua esiin. Ilmoituksen esiin tuonti tapahtui yhden funktion ajamalla, jolle annettiin parametriksi näytettävän ilmoituksen HTML-elementin id-attribuutti. Selvitin vielä tarvittavan paikan, jossa funktio kuuluisi ajaa. Kyseessä oli ilmoitus automaattisesta muutoksesta, jonka sovellus tekee syötteisiin. Selvitin missä tämä automaattinen muutos tapahtuu ja lisäsin funktion suorituksen sinne. Näin sovelluksen yläpalkkiin saatiin esiin ilmoitus, joka ilmoitti tapahtuneesta muutoksesta. Vein muutoksen testiympäristöömme ja testasin muutosta. Huomasin yläpalkin ilmoituksen olevan hyvin epäkäytännöllinen, sillä sitä ei nähnyt automaattisen syötemuutoksen tapahtuessa. Päätin keskustella asiasta päivän daily-palaverissa.

Lomani aikana oli myös ilmennyt ongelmia sovellustemme Jenkins-koontijonoissa. Eräs sovellus ei ollut käynnistynyt testiympäristössämme viikkoon. Tutkimme tätä työkaverini kanssa ja hän lopulta lähti selvittämään asiaa ympäristötuelta.

Aloitin myös tutustumisen uuteen bugiin, jossa erässä sovelluksessa ei voinut muuttaa valintalistan arvoa, jos sovelluksen syötteeseen syötettiin eräs tietty tieto. Kyseessä oli taas minulle uusi sovellus, jonka lähdekoodi oli minulle tuntematonta. Loppupäivä kului tähän koodipohjaan tutustuessa. Daily-palaverissa toin esiin ajatukset, jotka minulla heräsi ilmoitusmuutokseen liittyen. Käyttäjätarinan vaatimukset olivat epäselvät ja tiketin luonut oli tällä hetkellä lomalla, joten yritimme päätellä mitä muutosta haluttiin. Tulimme siihen tulokseen, että ei ollut tarkoitus tuoda ilmoitusta yläpalkkiin vaan teksti tuli tuoda syötteen alapuolelle esiin.

Torstai 15.7.2021

Suurin osa päivästä kului suunnittelupalaverissa. Iltapäivällä perehdyimme työkaverini kanssa siihen, miten eräiden sovellusten julkaisu julkaisutestausympäristöön toteutetaan. Emme olleet aiemmin toteuttaneet julkaisuja näille sovelluksille, joten jouduimme tutkimaan asiaa. Ympäristöissä sovellukset on sisällytetty

tiettyihin paketteihin. Meidän tuli selvittää, mihin pakettiin nämä sovellukset kuuluvat, jotta julkaisu voitiin toteuttaa. Lisäksi meidän tuli lisätä muutosten Jira-tiketit ja pakettien versionumerot julkaisun kirjanpitotiketille. Selvitimme sovellusten paketit aikaisemmista Jenkins-julkaisuista.

Viikkoanalyysi

Lomalta paluun jälkeen tein pienen muutoksen React-pohjaiseen sovellukseemme ja katselmoin työkaverini Robot-automaatiotesti muutoksia. Tiistaina minulla oli hiukan hankalampi tehtävä, sillä minun tuli tutustua minulle täysin uuteen koodiin. Pääsin yllättävän nopeasti kuitenkin sisälle sovellukseen ja onnistuin saamaan muutoksen aikaan. Tekemistä hankaloitti se, että sovellusta ei ollut mahdollista testata paikallisesti. Muutokset piti viedä versionhallintaan, jotta sitä voitiin testata testiympäristössä.

Lisäksi tutkimme työkaverini kanssa mitä Jenkins-automaatiosovelluksessamme on tapahtunut, sillä useat koontijonot epäonnistuivat outoihin virheisiin.

Tämän jälkeen syvennyin taas minulle uuteen sovellukseen, jossa oli ongelmia valintalistan kanssa.

Torstaina meillä oli tulevan sprintin suunnittelupalaveri, joka söi suuren osan päivästä. Iltapäivällä tutustuimme työkaverini kanssa meille vähemmän tuttujen sovellusten julkaisemiseen julkaisutestausympäristöön. Kokeneempien kollegojen ollessa lomalla tutkimme asiaa kahden, mutta pääsimme yllättävän hyvin asiasta selville. Saimme itse selvitettyä mihin pakettiin sovellukset tulee päivittää.

3.9 Yhdeksäs viikko

Maanantai 19.7.2021

Tänään tutustuin tarkemmin ongelmaan, jossa erään sovelluksen valintalistan arvoa ei voitu muuttaa, kun sovellukseen syötettiin eräs tietty tieto. Sovelluksen lähdekoodi oli minulle uutta, joten tutustuin sen toteutukseen ja yritin selvittää, miten valintalista toimii. Havaitsin syötteen suorittavan rajapintakutsun arvon muuttuessa. Rajapinnan palauttamat tiedot aiheuttivat muutoksia sovelluksen muihin syötteisiin.

Siirryin testiympäristöömme tutkimaan sovelluksen toimintaa tarkemmin ja seuraamaan sovelluksessa liikkuvaa dataa Chrome-selaimen debuggeritoiminnallisuudella. Testiympäristöissä sovellusten muuttujien ja funktioiden nimet on muunnettu, jotta sovellusten takaisinmallinnus olisi vaikeampaa. Tämä hidasti oikean koodinpätkän löytämistä debuggerin pysäytyspisteitä varten, mutta vertailemalla muutettua koodia oikeaan lähdekoodiin onnistuin löytämään tarvittavat paikat.

Lisäsin pysäytyspisteen paikkaan, jossa rajapintakutsu toteutettiin. Tämän avulla näin, mitä arvoja palautettiin. Arvojen perusteella pystyin tulkitsemaan paremmin, mikä ongelman aiheuttaa. Ilmeni, että rajapinta palauttaa tiedolla virheellisiä tietoja, jotka sovelluksen toteutuksessa poislukevat toisensa. Yhden tiedon perusteella valintalista piilotetaan, mutta toinen tuo valintalistan uudelleen esiin. Piilotus aiheutti sen, että valintalistan arvo ei muuttunut, koska piilotuksen yhteydessä se palautettiin aina oletukseen. Tämä operaatio tapahtui niin nopeasti, että selaimessa valintalista ei ehtinyt piiloutumaan eikä tätä kyennyt silmin näkemään.

Pitkän tutkimisen jälkeen korjaus oli lopulta yhden rivin lisäys ehtoon, jonka perusteella valintalista piilotettiin. Ehtoon lisättiin tarkistus sille, että toisen poislukevan tiedon tuli olla epätosi, kun piilotusta vaativa tieto on tosi. Iltapäivällä ehdin aloittaa automaatiotestejä korjaukselle.

Tiistai 20.7.2021

Jatkoin automaatiotestejä edellisen päivän korjaukseen. Testeissä on tarkoituksena syöttää tieto syötteeseen, joka suorittaa rajapintakutsun. Tämän jälkeen tarkistetaan, että valintalistan arvoa voidaan muuttaa. Sama testi toteutetaan useaan kertaan käyttäen eri tietoa rajapintakutsulle, jotta toiminnallisuus voidaan testata erilaisilla rajapinnan palauttamilla arvoilla. Ongelmia aiheutti valintalistan arvon tarkistaminen, sillä sitä ei näe suoraan HTML-koodista.

Työkaverillani ilmeni ongelmia eräässä muutoksessaan ja hän tarvitsi apua. Hän ei tiennyt miten tiimimme toisen sovelluksen uusin versio saatiin käyttöön eräälle toiselle sovellukselle. Kyseessä oli sovellus, joka sisältää aputoiminnallisuuksia ja kielimuuttujia eräille sovelluksistamme. Yleensä tällaiset päivitykset tapahtuvat siten, että uusi sovellusjulkaisu suoritetaan Jenkins-automaatiosovelluksella ja

Jenkinsin luoma sovellusversio päivitetään sitä tarvitsevaan sovellukseen. Sovelluksen Jenkins-putki kuitenkin puuttui tiimimme Jenkins-projektista. Kokeneemmat kehittäjät olivat kaikki lomalla, joten olimme ongelman kanssa kahden.

Muistelin, että Jenkins-automaatiosovelluksen versiota oli päivitetty jokin aika sitten ja että vanha versio löytyi vielä eri osoitteen alta. Yritin löytää tämän vanhan osoitteen ja tarkistaa, jos sovelluksen putki löytyisi sieltä. Sovelluksen putki löytyi vanhan Jenkinsin alta ja pienen tutkimisen jälkeen selvisi myös, miten sovellusjulkaisu saatiin toteutettua vanhaa Jenkinsiä käyttäen.

Palasin automaatiotestien pariin ja päätin tutkia, löytyisikö Robotilta avainsanaa valintalistojen tarkistamiseen. Robotin SeleniumLibrary-kirjaston alta löytyi avainsana “List Selection Should Be”, jonka avulla listan arvo oli helppo tarkistaa.

Toinen ongelma ilmeni valintalistojen paikantimien luonnissa. Yritin löytää valintalistaa käyttämällä XPath-kyselykielen contains-funktiota. Ongelmana oli se, että HTML-koodista löytyi kaksi valintalistaa, joiden id:t olivat lähes samat ja sisälsivät osittain saman id:n. Tästä syystä etsiessäni valintalistoja contains-funktiolla se palautti molemmat elementit. Contains-funktio hakee HTML-koodista elementtejä sille annetun attribuutin perusteella ja palauttaa elementit, joiden attribuuttien arvot sisältävät osan annettua parametriä.

Esimerkkinä seuraavat HTML-elementit:

```
<select id=”lista” />

<select id=”toinenlista” />
```

Kun muodostetaan seuraavanlainen XPath-kysely löydetään molemmat elementit.

```
//select[contains(@id,’lista’)]
```

Tajusin että minun tulee käyttää XPath-kyselykielen attribuuttivalitsimia contains-funktion sijaan.

Valitsimien käyttö tapahtui seuraavasti:

```
//select[@id="lista"]

//select[@id="toinenlista"]
```

Tällöin pystyin löytämään yksittäiset elementit. Jouduin tällöin tekemään valintalistojen hakuun kaksi erillistä avainsanaa.

Iltapäivällä avustin työkaveriani käännöstiedostojen enkoodauksen kanssa. Vanhempien sovellustemme käännöstiedostojen enkoodaukseen on käytössä windows1252. Tästä johtuen useimmat ohjelmointiympäristöt, kuten Visual Studio Code, eivät automaattisesti osaa avata näitä tiedostoja oikein. Ne muuttavat tiedoston sisältöä UTF-8 enkoodaukselle sopivaksi. Kun tiedostojen muutoksia katsotaan Visual Studio Coden sisäisellä muutosvertailulla ei tiedostossa näy ylimääräisiä muutoksia, ainoastaan käsin tehdyt. Kuitenkin kun muutokset viedään versionhallintaan, voidaan ohjelmointiympäristön nähdä yrittäneen muuttaa tiedoston enkoodausta ja lähes joka rivillä on muutoksia. Katsoimme työkaverilleni ohjelmointiympäristön asetukset kuntoon, jotta tätä ei enää tapahtuisi.

Keskiviikko 21.7.2021

Aamulla viimeistelin vielä automaatiotestit. Loin paikantimet kahdelle eri valintalistalle ja testasin niiden toimivuuden. Tämän jälkeen aloin tutkimaan toista ongelmaa samassa sovelluksessa. Sovellukseen tallennetaan tietoja lomakkeella ja tallennus tapahtuu onnistuneesti. Kuitenkin kun tietoja mennään muokkaamaan, on eräs valintalista piilossa. Kyseessä on sama valintalista kuin minkä kanssa työskentelin aiemman ongelman kanssa. Kyseessä on lähes sama ongelma kuin aiemmin. Tällä kertaa tieto saadaan kuitenkin tallennettua oikein, mutta muokkauksessa se on virheellinen. Tämä johtuu sitä, että tällä kertaa erään annetun tiedon perusteella rajapinnalta palautetut tiedot ovat erit. Käytin päivän

perehtymällä sovelluksen Java palvelintoteutukseen, sekä jQuery selaintoteutukseen. Yritin muodostaa palvelinpuolelle yksikkötestiä, jonka avulla olisi voinut tarkistaa tapahtuuko lomakkeen tallennuksen yhteydessä jotain virheellistä datan muokkausta. En kuitenkaan saanut mitään tällaista selville yksikkötestien perusteella.

Torstai 22.7.2021

Jatkoin saman ongelman selvittelyä. Huomasin, että sovelluksessa käytetään piilotettuja HTML-syötteitä, joissa talletetaan rajapinnalta palautetut arvot. Nämä arvot otetaan HTML-koodista myös JavaScriptin käyttöön. Nämä arvot lisätään HTML-koodiin jo lomaketta luotaessa, joten pystyin sen perusteella tarkistamaan ongelmat sovelluksen lomakkeen tallennuksessa. Arvot olivat kuitenkin samat ennen ja jälkeen tallennuksen. Tämä viittasi siihen, että tiedot ovat virheelliset jo rajapinnalta palautuessa. Tällöin minun tuli keskittyä vain käsittelemään tätä virheellistä dataa siten, että se ei riko meidän sovellustamme. Eräs palautetuista tiedoista on kriittinen ja muuttaa laajasti sovelluksen syötteitä. Muutama muu arvo muuttaa yksittäisten syötteiden arvoja, joka tässä tapauksessa aiheuttaa ongelman valintalistan piiloutumisessa. Pohdimme asiaa tiimimme kanssa daily-palaverissa ja päätimme korjata ongelman siten, että kun tämä kriittinen havaitaan todeksi, muutetaan muut arvot sellaisiksi kuin niiden yleensä kuuluisi olla tämän kriittisen arvon ollessa tosi. Toteutin korjauksen siten, että kun rajapinta palauttaa arvot ja niitä käsitellään, toteutetaan datan muokkaus. En kuitenkaan saanut vielä toteutusta toimimaan.

Viikkoanalyysi

Koko viikko on kulunut erään tietyn sovelluksen ongelmien ja niihin liittyvien automaatiotestien parissa. Suuri osa työnkuvasta on ollut koodiin tutustumista ja sen toiminnan ymmärtämistä. Chrome selaimen Debugger-toiminnallisuus ilmeni loistavaksi työkaluksi koodin toiminnan selvittämiseen erityisesti, kun sovellusta on pakko testata testausympäristössä.

Robotin kanssa havahduin eroon XPathin contains-haun ja suoran attribuuttihaun välillä. Olin normaalisti käyttänyt paikantimien luontiin contains-metodia sitä sen

kummemmin ajattelematta, mutta tiistaiset ongelmat toivat minulle esiin haasteet mitä sen turhasta käytöstä saattaa nousta.

JSP-teknologian tapa hoitaa muuttujat selaintoteutukselle piilotetun lomakkeen kautta oli minulle yllättävä. Olen tottunut siihen, että arvot saadaan JavaScriptissä toteutetun rajapintakutsun kautta ja että arvot tulevat siistissä JSON-muotoisessa datassa. Tämä havahtuminen kuitenkin auttoi minua ymmärtämään paremmin mitä koko sovelluksessa tapahtuu.

3.10 Kymmenes viikko

Maanantai 26.7.2021

Jatkoin viimeviikkoisen ongelman parissa. Huomasin ongelman viimeviikkoisessa muutoksessani. Muutokseni oletti, että rajapinta palauttaa arvoja mutta kun lomake avataan muokkaukseen, tiedot haetaan sovelluksen HTML-koodista. Tällöin datan muokkaukseni ei koskaan ehtinyt vaikuttaa sovelluksen toimintaan, koska se yli kirjoitettiin HTML-koodista haetuilla arvoilla. Siirsin muokkauksen toteutettavaksi vasta sen jälkeen, kun tiedot on haettu HTML-koodista. Korjaus toimi nykyistä ongelmaa varten, muttei korjannut toista samankaltaista ongelmaa, joka minun tuli korjata seuraavaksi. Ilmeni että ongelmassa lomakkeen tiedot eroavat aiemmin korjatusta eikä toteutus muokannut näitä tietoja ollenkaan. Lomake sisältää tietoa, joka määrittää sen tietyn tyyppiseksi lomakkeeksi. Lomakkeen tyypin muuttaminen yhdestä toiseen on estetty ja tämä on alkuperäinen ongelman aiheuttaja. Erään lomakkeen tiedoista tulisi sallia molempien tyyppien käyttö, joten ongelma halutaan korjata poistamalla tämä rajoitus. Tutustuin siihen, miten tämä tyyppi muutoksen sallittavuus voitaisiin toteuttaa ja mihin sillä on vaikutuksia. Muutos vaikuttaa toteutettavalta mutta sen vaikutukset päästään todentamaan vasta testiympäristössä. Tarkoitus muuttaa toiminnallisuutta siten, että kun aiemmin tyyppimuutosta tehtäessä on näytetty virheilmoitus ja estetty muutos, nyt poistetaan ilmoitukset ja suoritetaan muutos.

Tiistai 27.7.2021

Poistin eräästä sovelluksesta ilmoituksen, joka näytettiin, kun käyttäjä yritti suorittaa lomakkeen tyyppimuunnoksen. Ilmoituksen näyttämisen sijaan toiminnallisuutta muutettiin siten, että tyyppimuunnos oli mahdollista suorittaa. Tyyppimuunnokselle oli olemassa valmiit metodit, joten näitä metodeja tarvitsi vain kutsua tässä tilanteessa. Vietiin muutos testiympäristöön ja testattiin sen vaikutuksia. Tietyissä tilanteissa lomakkeen tallentaminen ei onnistunut. Myöskään tyypin muuttaminen toiseen suuntaan ei toiminut. Ongelmana näytti olleen aiempi korjaus. Poistettiin vanha korjaus ja testattiin mitä tapahtuu. Vielä ei kuitenkaan saatu lomakkeen tallennusta toimimaan oikein.

Keskiviikko 28.7.2021

Muutettiin vanhaa korjausta siten, että se osaa huomioida tarkistukset suorittavan metodin tyyppi parametrin. Lomaketta ei voitu muuttaa toisesta tyypistä enää takaisin toiseen koska metodi tarkisti vain HTML-koodista löytyvän tyypin ja muokkasi datan tämän perusteella. Tällöin tyyppiä ei koskaan voitu muuttaa takaisin. Nyt kun tarkistetaan, eroaako parametrinä tuleva tyyppi HTML-koodista löytyvästä tyypistä, saadaan tyyppi muutettua ja data muokattua onnistuneesti. Metodissa oli toteutettu myös palautuksia niissä paikoissa, joissa tyyppiä olisi yritetty muuttaa, jotta metodin suorittaminen saataisiin keskeytettyä. Näitä ei enää tarvittu koska metodi haluttiin ajaa loppuun tyypin muuttamiseksi. Tämän jälkeen lomakkeen tallennus ja tyypin muutos toimivat.

Iltapäivällä avustin työkaveriani JSP-pohjaisen sovelluksen kanssa. Hänellä oli epäselvyyttä siitä, miten teknologia toimii enkä itsekkään ollut teknologiasta täysin perillä. Yritimme yhdessä tutkia, miten sovellus toimii. Suurin epäselvyys meille oli se, miten sovelluksen palvelintoteutuksen arvoja käytetään sovelluksen selaintoteutuksessa. Ilmeni, että ”addAttribute”-nimisellä metodilla palvelintoteutuksessa merkataan Java muuttujia käytettäväksi selaintoteutukseen. Merkkauksen jälkeen muuttujat palautetaan sovelluksen HTML-koodiin piilotetun lomakkeen sisään. Sovellukseen tuli toteuttaa integraatio toisen sovelluksen kanssa. Sovelluksessa tehtävät lomakkeet tuli voida tallentaa toiselle sovellukselle käyt-

töön. Sovellukset olivat jo integroidut toisiinsa, joten ehdotin, että toisen sovelluksen lomakkeen tallennus siirrettäisiin integroitujen sovellusten käytettäviin toiminnallisuuksiin.

Lisäsin vielä ilmoituksen sovellukseen, johon tein korjausta siihen tilanteeseen, kun tyyppiä vaihdetaan.

Torstai 29.7.2021

Aamulla aloitin automaatiotestejä edellisenä päivänä tekemääni korjaukseen. Testejä toteuttaessa huomasin, että erään tiedon sisältävän lomakkeen tietoja ei voitu tallentaa, jos erääseen valintalistaan oli valittu tietty arvo. Tällöin vaadittiin syötettäväksi tietoa, jonka tulisi olla vaadittu vain silloin kun lomakkeen tyyppi on toinen. Tämä johtui siitä, että metodi, joka suorittaa tyyppimuutoksen ei poistanut HTML-koodista lomakkeen required-attribuuttia. Loppupäivä kului tulevan sprintin suunnittelupalaverissa.

Viikkoanalyysi

Työt ovat kuluneet taas samoissa merkeissä kuin aiemmalla viikolla. Viikko on sisältänyt paljon koodin opiskelua ja ongelman ratkomista Chromen debuggerityökalulla. Viikon ongelman ratkomiset ovat tuoneet esille sen, kuinka vähän ymmärrän JSP sovellusten toiminnasta. Olen tottunut React-sovellusten kanssa toimimiseen ja vaikka molemmissa teknologioissa työskennellään JavaScriptiä käyttäen, ovat käyttötavat hyvin erilaiset. Tästä johtuen olen joutunut käyttämään paljon aikaa sovelluksen toiminnan ymmärtämiseen. Tilannetta ei myöskään helpota se, että sovellusta ei ole mahdollista testata paikallisesti. Kun jatkossa tulee tehtäviä liittyen JSP-sovelluksiin, minun tulee ottaa aikaa opiskellakseni teknologian toimintaa. Työajastamme on varattu 10 % opiskeluun, joten tämä olisi hyvä tila käyttää JSP:n perehtymiseen syvemmin.

4 POHDINTA

Kuluneiden viikkojen aikana työnkuvani oli hyvin vaihteleva. Työskentelin sovellusten selain- ja palvelintoteutusten sekä automaatiotestien parissa. Opinnäytetyön alkuperäinen tavoite oli analysoida OPUX-komponenttikirjaston vaikutusta React-pohjaisten sovellusten kanssa työskentelyyn. Loppujen lopuksi en juurikaan kuluneiden viikkojen aikana kehittänyt React-pohjaisia sovelluksia, joten OPUX-kirjastoon liittyvät analyysit jäivät vähäisiksi.

OPUX-kirjaston voidaan kuitenkin vähäiselläkin käytöllä todeta antavan suurta hyötyä React-sovellusten kehitykseen. Se sisältää kaikki peruskomponentit, joita verkkosovelluksia rakentaessa tarvitsee. Komponenttien ulkoasut on valmiiksi suunniteltu Osuuspankille sopivaksi, joten tyyleihin ei tarvitse juurikaan koskea. Ainoat ongelmat opinnäytetyön aikana ilmenivät kalenterikomponentissa, ja sen kanssa on tiimissämme ollut haasteita jo ennen työn aloittamista. Ongelmat ovat kuitenkin lopulta olleet pikku seikkoja, eikä mitään kriittisempiä ongelmia OPUXin kanssa ole ilmennyt.

Työt ovat odotettua poiketen keskittyneet vanhempiin ei-React-pohjaisiin sovelluksiin sekä sovellusten palvelintoteutuksiin. Lisäksi tehtäviin on kuulunut automaatiotestien kehitystä. Kaikki nämä aiheet ovat olleet minulle jollain tasolla uusia, ja niissä on ollut kehitystä kuluneiden viikkojen aikana. Suurimman kehityksen olen itse havainnut JSP- ja jQuery-pohjaisten selaintoteutusten kehityksessä. Neljännellä viikolla työstin ensimmäistä kertaa tällä teknologialla toteutettua sovellusta. Muutos oli hyvin yksinkertainen ja sain sillä kevyen ensikosketuksen näihin sovelluksiin. Tulevina viikkoina työni keskittyivät yhä enemmän JSP-pohjaisiin sovelluksiin. Tietämättömyyteni teknologiasta aiheutti haasteita kehityksen aikana. Jouduin käyttämään paljon työajastani lähdekoodin tutkimiseen ja sen toiminnan ymmärtämiseen. Pienetkin muutokset veivät paljon aikaa, koska en ymmärtänyt, mitä koodissa tapahtuu. Kolmen viimeisen viikon aikana perehdyin JSP-pohjaisiin sovelluksiin syvemmin ja ongelmien ratkominen alkoi onnistua entistä sujuvammin.

Opinnäytetyön loppua kohden työskentely JSP:n kanssa kehittyi rutiininomaisemmaksi. Sovellusten lähdekoodi tuli tutummaksi ja toimintatavat selkeytyivät. Huomasin myös edistystä kehitystiimin katselmointiputken sujuvuudessa. JSP-pohjaisia sovelluksia ei voitu testata paikallisesti, joten uutta koodia tuli aktiivisesti viedä katselmoitavaksi, jotta se saatiin testattavaksi testiympäristöön. Kaikki kehittäjät katselmoivat koodeja aktiivisesti useampaan otteeseen päivän aikana ja pullonkaulaa testaukseen ei päässyt syntymään.

Omalta osaltani huonoksi tässä toimintatavassa havaitsin sen, että pienien virheiden määrä kasvoi verrattuna normaaliin. Toteutusten testaaminen paikallisesti mahdollistaa yksinkertaisten kirjoitusvirheiden ja pienien huolimattomuuksien kiinni jäämisen nopeasti. Hyvänä puolena todettakoon kuitenkin se, että kynnys viedä koodia versionhallintaan laski. Usein olen päätynyt tilanteeseen, jossa en ole vienyt koodia versionhallintaan jo kehityksen aikana ja viennit ovat kasvaneet hyvin suuriksi. Tällöin muutosten lajittelu committeihin on ollut työlästä ja muutokset raskaita katselmoida. Nykyisellään JSP-muutosten katselmoinnit ovat hyvin kevyitä tiheän viennin ansiosta.

Olen ollut tyytyväinen JSP-taitojeni kehitykseen, mutta opittavaa riittää vielä paljon. Lisäksi haluaisin päästä syventämään taitojani myös Javan kanssa. Opinnäytetyön aikana palvelintoteutuksia kehitettiin vain muutamaan otteeseen, ja nekin olivat vain pieniä korjauksia. Haluaisinkin tulevaisuudessa päästä kehittämään Java-palvelintoteutuksia laajemmin.

LÄHTEET

1. JavaScript.info 2020. Bubbling and capturing. Saatavissa: https://javascript.info/bubbling-and-capturing. Hakupäivä 15.5.2021.

2. Ant Design. DatePicker. Saatavissa: https://ant.design/components/date-picker/. Hakupäivä 15.5.2021.

3. ReactDatePicker Crafted by HackerOne. Saatavissa: https://reactdatepicker.com/. Hakupäivä 15.5.2021.

4. react-day-picker Date Picker component for React. Saatavissa: http://react-day-picker.js.org/.Hakupäivä 26.9.2021.

5. MDN Web Docs Axes. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/XPath/Axes. Hakupäivä 3.10.2021.