Haaga-Helia
ammattikorkeakoulu Oy

Web-komponenttikirjaston kehittäminen ja viimeistely: Järjestelmäkehittäjän päiväkirja

Nico Tukiainen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2021

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

9.5.2021

Tekijä(t)
Nico Tukiainen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Web-komponenttikirjaston kehittäminen ja viimeistely: Järjestelmäkehittäjän päiväkirja

Sivu- ja liite-sivumäärä
63

Opinnäytetyön otsikko englanniksi
Developing and refining a web component library: Journal of a Systems Developer

Tässä päiväkirjamuotoisessa opinnäytetyössä seurattiin järjestelmäkehittäjän työtä Pagero Oy:llä kahdeksan viikon seurantajakson ajan aikavälillä 15.2.2021 – 9.4.2021. Opinnäytetyön tarkoituksena oli seurata ja reflektoida järjestelmäkehittäjän ammattitaidon kehittymistä päivittäisten ja viikoittaisten analyysien avulla. Työn raportoinnissa keskityttiin työnantajayrityksen sisäiseen käyttöön tulevan web-komponenttikirjaston kehittämiseen ja viimeistelyyn.

Opinnäytetyön tietoperustassa kuvattiin lähtötilannetta eli järjestelmäkehittäjän työympäristöä, analysoitiin oman työn nykytilaa ja perehdyttiin järjestelmäkehittäjän työssä vaadittavaan osaamiseen. Nykytilan analyysissa esiteltiin järjestelmäkehittäjälle kuuluvia työtehtäviä, työpaikan sidosryhmiä sekä kuvattiin työssä tarvittavia vuorovaikutustaitoja.

Raportointiosiossa sekä jokaiselle päivälle että viikolle asetettiin omat tavoitteensa. Päivittäisessä raportoinnissa keskityttiin päivän tapahtumien kuvaamiseen ja kyseiselle päivälle asetetun tavoitteen toteutumisen arviointiin. Jokaisen raportointiviikon päätteeksi kirjoitettiin viikkoanalyysi, jossa analysoitiin kuluneelle viikolle asetetun tavoitteen toteutumista alan lähdemateriaaliin peilaten. Joillakin raportointiviikoilla viikkoanalyysissa käsiteltiin myös viikon työtehtävistä johdettua erillistä teemaa lähdeaineiston avulla.

Pohdinnat ja päätelmät -osiossa käsiteltiin raportointijakson aikana tapahtunutta henkilökohtaista kehitystä. Lisäksi siinä tarkasteltiin, mitä raportointijakson aikana olisi voinut tehdä toisin ja mietittiin, miten järjestelmäkehittäjän työssä kehittyminen voisi edetä tulevaisuudessa.

Asiasanat
Ohjelmistokehitys, järjestelmäsuunnittelu, verkko-ohjelmointi, javascript

Sisällys

1 Johdanto 1
1.1 Keskeiset ammattikäsitteet 3
2 Lähtötilanteen kuvaus 6
2.1 Oman nykyisen työn analyysi 6
2.2 Sidosryhmät työpaikalla 9
2.3 Vuorovaikutustaidot työpaikalla 10
3 Päiväkirjaraportointi 12
3.1 Seurantaviikko 1 12
3.2 Seurantaviikko 2 18
3.3 Seurantaviikko 3 26
3.4 Seurantaviikko 4 31
3.5 Seurantaviikko 5 36
3.6 Seurantaviikko 6 42
3.7 Seurantaviikko 7 47
3.8 Seurantaviikko 8 52
4 Pohdinta ja päätelmät 57
Lähteet 60

1 Johdanto

Järjestelmäkehittäjän työhön voi kuulua hyvin erilaisia työtehtäviä työnantajasta riippuen. Omassa työssäni järjestelmäkehittäjänä olen rakentanut työpaikallani sisäiseen käyttöön tulevaa web-komponenttikirjastoa viime vuoden keväästä saakka. Tässä päiväkirjamuotoisessa opinnäytetyössä seurataan järjestelmäkehittäjän työtehtäviä kahdeksan viikon aikajaksolla. Raportissa kuvataan web-komponenttikirjaston version 1.0 viimeisten puuttuvien komponenttien kehittämistä ja kirjaston viimeistelyä. Alun perin tavoitteena oli kuvata myös web-komponenttikirjaston julkaisu sekä käyttöönotto, mutta projektin viivästymisen vuoksi ne eivät ehtineet raportointijaksolle. Päiväkirjassa raportoidaan kalenteriviikot 7–15, ja se sijoittuu aikavälille 15.2.2021 – 9.4.2021.

Pagero Ab on vuonna 2009 perustettu ruotsalainen kansainvälisten sähköisten dokumenttien välittäjä ja sen kotipaikka on Göteborg. Se tarjoaa pilvipohjaisia sähköisten dokumenttien välitysverkostoja, joiden avulla asiakkaat voivat tavoittaa muita yrityksiä eri puolilla maailmaa riippumatta siitä, kuinka monessa maassa sillä on liiketoimintaa. Pagero huolehtii samalla myös asiakasyritysten dokumenttien maakohtaisista vaatimuksista. Pagerolla Ab:llä on noin 350 työntekijää ja konsernin liikevaihto vuonna 2019 oli 38,4 miljoonaa dollaria. Työnantajani, eli Pagero Ab:n suomalainen tytäryhtiö Pagero Oy on perustettu vuonna 2010. Pagero Oy:n liikevaihto oli vuonna 2019 3,12 miljoonaa euroa. Suomen toimipisteessä on tällä hetkellä 28 työntekijää.

Olen työskennellyt Pagero Oy:llä lokakuusta 2018 alkaen. Tätä ennen työskentelin ravintola-alalla, eikä minulla ollut ollenkaan aiempaa kokemusta IT-alan töistä. Aloitin työskentelyn Pagero Oy:n teknisessä tuessa IT support spesialistina, missä työtehtäviin kuului asiakkaiden auttaminen järjestelmään liittyvien teknisten ongelmien kanssa. Teknisessä tuessa opin työn kautta verkkolaskutukseen liittyvät käsitteet ja menetelmät, mistä on ollut paljon hyötyä myös nykyisessä työssäni.

Nykyinen työni järjestelmäkehittäjänä alkoi tammikuussa 2020, eli olen työskennellyt tehtävässäni reilun vuoden verran. Pagero Ab:llä on 15 tuotekehitystiimiä, joista suurin osa sijaitsee Ruotsissa, Saksassa sekä Sri Lankassa. Työskentelen osana ruotsalaista kahdeksan hengen tiimiä, jonka vastuualueena on pääsääntöisesti uuden web-ulkoasun rakentaminen Pagero Ab:n pilvipohjaiselle dokumenttien välitysjärjestelmälle. Tuotekehitystiimit on nimetty lintujen mukaan ja oman tiimini nimi on Peacock, eli riikinkukko. Koko tiimi työskentelee lähes poikkeuksetta etänä, mutta tapaamme tiimin kanssa päivittäin videon välityksellä yleensä noin 10–40 minuutin pituisissa stand up-kokouksissa. Kehittäjille järjestetään normaalitilanteessa yhteisiä koulutuksia pari kertaa vuodessa Ruotsissa, mutta

nykyisten matkustus- ja tapaamisrajoitusten myötä näitä tapaamisia ei ole ollut mahdollista toteuttaa.

Työssä järjestelmäkehittäjänä tarvittavaan tietoperustaan kuuluvat olennaisena osana hyvät Front-end-kehitystaidot sekä yleisimpien web-teknologioiden, kuten HTML5:en, CSS:n, JavaScript:in ja NodeJS:n käyttäminen sekä Git-versionhallinta. Koodia kirjoitetaan web-standardien mukaisesti, eli sen tulee olla semanttisesti oikeaoppista, validia ja esteetöntä. Front-end- järjestelmäkehittäjänä työskenteleminen vaatii myös JavaScriptin hyvää ymmärrystä, sillä se on verkkoselainten käyttämä kieli ja siksi se on yksi suosituimmista ohjelmointikielistä maailmassa (Crockford 2008, 11.). Crockfordin kirja Javascript: The Good Parts (2008) on eräs alan perusteoksista, jossa on tiivistetty kaikki oleellinen niin alkajille, kuin kokeneemmillekin ohjelmoijille ja sitä käytetään yleisesti kehitystyön tukena.

Alan ehkä merkittävin perusteos on Robert C. Martinin kirja Clean Code: A Handbook of Agile Software Craftmanship (2009). Siinä käsitellään muun muassa koodin puhtautta, mikä onkin Front-end-kehitystaitojen yksi tärkeä osa-alue. Puhtaus tarkoittaa koodin tehokkuutta, luettavuutta sekä minimalistisuutta. Myös virheiden käsittely, koodin formatointi ja kommentointi sekä funktioiden, objektien ja datarakenteiden selkeä nimeäminen on puhtaalle koodille olennaista. (Martin 2009, 7–9.)

Järjestelmäkehittäjän työssä on hallittava vähintään ohjelmoinnin perusteet, mutta olennainen taustakoulutus, kuten IT-tradenomin tutkinto auttaa ymmärtämään toimialaa laajemmin. Substanssiosaamisen lisäksi englannin kielen hyvä hallinta on työssäni välttämätöntä. Alalla on tyypillistä, että täysin suomenkielisissä yrityksissäkin käytetään työkielenä englantia. Oma tiimini Peacock on monikulttuurinen ja olen tiimin ainoa suomenkielinen jäsen, joten työkielenä puhutaan englantia. Koska työ on samanaikaisesti sekä itsenäistä, että tiimityötä, vaatii työ hyvää ajanhallintaa sekä etätyöskentelytaitoja. Lisäksi työtehtävässä menestyäkseen on tärkeää olla ratkaisukeskeinen, utelias sekä kyvykäs omaksumaan jatkuvasti uutta tietoa.

1.1 Keskeiset ammattikäsitteet

Lista työn kannalta keskeisistä ammattikäsitteistä, joita tullaan käyttämään opinnäytetyössä. Lisäksi joitain laajempia käsitteitä tullaan avaamaan tarkemmin työn edetessä.

API (Application Programming Interface), eli ohjelmointirajapinta.

Attribute, eli määrite tuo HTML-elementille ylimääräistä tarkentavaa tietoa. Määritteet ovat usein nimi-arvopareissa, kuten esimerkiksi nimi=”arvo”.

Backlog refinement on ketterään ohjelmistokehitykseen kuuluva menetelmä, jossa tiimin tuoteomistaja katsoo yhdessä tiimin kanssa backlogiin kertyneitä asioita ja varmistaa muun muassa, että tehtäviä priorisoidaan oikeassa järjestyksessä.

Branch on GitHub-repositorioon liitettävä erillinen haara, joka sisältää tyypillisesti johonkin koodin osa-alueeseen tehtyjä muutoksia. Kaikissa GitHub-repositorioissa on master-branch eli master-haara, mihin tehdyt muutokset tyypillisesti liitetään, kun haaran pull request on hyväksytty.

CSS (Cascading Style Sheet) on merkintäjärjestelmä, jolla esitetään internetselaimille dokumenttien ulkoasua koskevia muutosehdotuksia. (Korpela, 2008)

Design tokenit ovat yrityksen sisällä käytettäviä ennalta määriteltyjä arvoja CSS-muuttujille. Niitä käytetään, jotta järjestelmän sisältö näyttäisi yhtenevältä.

DOM (Document Object Model) on dokumenttioliomalli, joka kuvaa internetselaimessa sivun rakennetta puun muodossa.

Event on HTML:n ominaisuus, jonka avulla voidaan laukaista eri toimenpiteitä verkkoselaimessa, kuten käynnistää JavaScript käyttäjän painaessa painiketta.

HTML eli Hypertext Markup Language tarjoaa välineet web-sivujen ja niiden välisten linkkien esittämiseen, jotka esitetään käyttäjälle internetselaimen avulla. HTML on merkkauskieli, jossa tietosisällön sekaan on upotettu tiedon rakennetta ja ulkoasua kuvaavia tageja. Merkkaus alkaa pääsääntöisesti kulmasulkuihin kirjoitetulla alkumerkkauksella <merkkaus> ja päättyy loppumerkkaukseen </merkkaus>. (Hyvönen, 2018)

Framework on yleisesti käytetty termi erilaisista JavaScriptiin pohjautuvista viitekehyksistä, kuten muun muassa Angular, React tai Vue.

Front-end on sovelluksen käyttäjälle näkyvä osa.

Git-versionhallinta tarkoittaa palvelua, johon koodia säilötään. Versionhallinta auttaa pitämään varmuuskopioita ohjelman nykyisestä sekä aiemmista versioista. Lisäksi sen avulla voidaan työstää koodia yhdessä tiiminä. Pagerolla käytetään Git-versionhallintaan GitHub-alustaa.

GUI (Graphical User Interface) eli graafinen käyttöliittymä.

JavaScript on web-sovellusten pääasiallinen ohjelmointikieli. JavaScriptillä voidaan laskea, manipuloida sekä validoida dataa.

Metodeja (Method) käytetään, kun halutaan paljastaa esimerkiksi web-komponentin sisällä oleva funktio ulkopuolelta käytettäväksi.

Mikropalveluarkkitehtuuri (Microservice Architechture) on työskentelytapa, jonka avulla ohjelmistoja suunnitellaan itsenäisinä pienempinä palasina yhden valtavan monoliitin hallitsemisen sijaan. (Fowler, 2014)

NPM (Node Package Manager) on työkalu pakettien hallintaan, jonka avulla julkaistaan tai otetaan käyttöön muiden tekemiä moduuleita.

Polyfill on koodia, jonka avulla voidaan toteuttaa muun muassa JavaScript-ominaisuuksia, joita internetselaimet eivät välttämättä automaattisesti tue.

Pull request on Git -versionhallintaan liittyvä käsite, kun kehittäjä lähettää koodin katselmoitavaksi muille kehittäjille. Katselmoija voi pyytää korjauksia tai hyväksyä muutokset.

Refaktorointi tarkoittaa ohjelman lähdekoodin muokkaamista siten, että ohjelman alkuperäinen toiminnallisuus säilyy, vaikka koodin rakenne muuttuu. Koodia refaktoroidaan muun muassa siitä syystä, että koodi olisi ylläpidettävämpää ja helpommin luettavaa.

SCSS (Sassy CSS) tuo normaaliin CSS-kieleen lisäominaisuuksia, kuten funktioiden sekä mixinien käytön. Projektia rakennettaessa SCSS käännetään CSS-muotoon, joten sitä voi käyttää CSS:n tavoin.

Shadow DOM:n avulla web-komponentti voidaan koteloida omaan shadow DOM-puuhunsa, jolloin sen tyylittelyä ei voida muuttaa web-komponentin ulkopuolelta.

Storybook on työkalu käyttöliittymäkomponenttien kehittämiseen.

Syntaksi tarkoittaa lauseoppia. Järjestelmäkehityksessä sanalla syntaksi viitataan usein ohjelmointikielen kielioppisääntöihin.

TypeScript on Microsoftin rakentama ohjelmointikieli, joka on rakennettu JavaScriptin päälle ja se on täysin JavaScript-yhteensopiva. TypeScript tuo JavaScriptin päälle JavaScriptistä puuttuvia lisäominaisuuksia, kuten muun muassa tyyppijärjestelmän, joka vaatii, että muuttujille ja parametreille asetetaan tyyppi. Tyyppien etuna on, että koodin ylläpidettävyys paranee, kun muut kehittäjät näkevät suoraan, minkä tyyppisistä muuttujista ja parametreista on kyse.

VirtualBox on käyttöjärjestelmien virtualisoimiseen käytettävä ohjelma. Koska työskentelen Mac-tietokoneella, käytän VirtualBoxia päästäkseni testaamaan web-komponentteja muun muassa Windowsin Internet Explorer 11-selaimella.

Web-komponentti tarkoittaa mukautettua HTML-elementtiä, jota voidaan käyttää muun muassa shadow DOM:n sisällä.

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työn analyysi

Työssäni keskityn pääsääntöisesti web-komponenttikirjaston kehittämiseen, testaamiseen, korjaamiseen ja palautteen keräämiseen. Työtä tehdään käytännössä scrum-projektihallinnan viitekehystä noudattaen, vaikka emme tiimissä aktiivisesti sprinteistä puhukaan. Työtehtäviini kuuluu muun muassa

- tutkimustyö käytettävien web-teknologioiden osalta
- uusien komponenttien suunnittelu, kehittäminen ja testaus
- olemassa olevien komponenttien ylläpito ja bugien korjaaminen
- teknisen dokumentaation laatiminen
- tuotekehitykseen tulevien tukipyyntöjen ratkominen
- koodikatselmointi, sekä
- työn esitteleminen kollegoille.

Tutkimustyö ilmenee konkreettisesti työssäni siten, että etsin tietoa käytettävistä web-teknologioista useista alalla yleisesti käytettävistä lähteistä. Näitä tiedonlähteitä ovat muun muassa GitHub, Stackoverflow, W3Schools ja Medium. Lähes päivittäistä tutkimista tai tarkastelua vaativat myös eri selainten tuet, eri kielien syntaksit, eventit sekä metodit.

Uusien komponenttien suunnittelu on tällä hetkellä tauolla, sillä keskitymme olemassa olevat komponenttien viimeistelyyn version 1.0 julkaisua varten. Uusien komponenttien kehitysprosessissa tiimin ulkoasusuunnittelija tekee komponentista prototyypin Zeplin -ohjelmaan, josta kehittäjä katsoo, minkä näköinen komponentista täytyy tehdä. Prototyyppiä ajatellen aletaan rakentaa semanttisesti oikeanlaista komponenttia, joka vastaa toiminnallisuuksiltaan mahdollisimman hyvin natiivia HTML:n vastaavaa elementtiä, kuten esimerkiksi <button>. Komponenteista keskustellaan tiimin kesken ja mikäli niistä löytyy korjattavaa, lisätään korjaukset tehtävälistaan. Valmiiden komponenttien dokumentointiin sekä testaamiseen käytetään Storybookia, joka on kehitysympäristö front-end-käyttöliittymäkomponenteille. Storybookissa on mahdollista muuttaa komponentin parametreja (property), määritteitä (attribute) sekä tarkastella erilaisia tiloja (state) interaktiivisesti, eli muutosten vaikutukset ovat nähtävissä välittömästi (Storybook 2021.).

GitHub -repositoriot ovat säilytyspaikkoja koodille, jotka helpottavat kehittäjien yhteistyötä versionhallinnan pysyessä kontrollissa (TechCrunch 2012.). Web-komponenttikirjaston tekninen dokumentaatio lisätään GitHub -repositorion lisäksi Storybookiin, jotta kaikki tieto olisi helposti yrityksen muiden kehittäjien saatavilla. Komponentin dokumentaatiossa kuvataan selkeästi kaikki komponentin ominaisuudet, eventit ja metodit. Dokumentaatiota laaditaan jo komponentin kehitysvaiheessa, ja täydennetään tarpeen mukaan.

Tuotekehitystiimit ovat vuorollaan vastuussa tuotekehitykseen tulevista tukipyynnöistä, joita tekninen asiakastuki ei pysty itse selvittämään. Vuorossa oleva tiimi jakaa yhteiseen pooliin tulevat tiketit aiheen perusteella niistä vastaaville tiimeille, eli jokaisella tiimillä on omat vastuualueensa. Oman tiimini vastuualueita ovat muun muassa Pageron uusi laskuportaali, liiketoimintaprosessit sekä uuden käyttöliittymän web-infrastruktuuri. Näitä tukipyyntöjä tulee tiimille päivittäin ja niiden kiireellisyys riippuu sekä tapauksen kriittisyydestä, että asiakkaan kanssa sovitusta palvelutasosta.

Kun kehittäjä kirjoittaa koodiin lisäyksiä tai muutoksia, koodi lähetetään katselmoitavaksi pull requestia käyttämällä. GitHub Docsin (2021.) määritelmän mukaan pull requestia voidaan käyttää, kun koodi halutaan katselmoida ennen sen lisäämistä kohteen repositorioon. Koodia ei siis lisätä suoraan web-komponenttikirjaston olemassa olevaan masterhaaraan, vaan se täytyy katselmoida, eli tarkastaa ensin tiimin scrum masterin, tai joku muu seniorin toimesta. Katselmoinnin jälkeen koodi hyväksytään, tai siihen ehdotetaan muutoksia. Katselmointia pidetään tiimissä tärkeänä ja sen etuja ovat muun muassa virheiden minimointi, tasalaatuisuuden ylläpitäminen sekä tietämyksen jakaminen nuoremmille kehittäjille (Brightspot 2021.)

Päävastuullisen kehittäjän pitää valmistautua esittelemään työnsä edistymisen kollegoille. Pidämme tiimin kesken pääsääntöisesti kerran viikossa web-komponenttikirjastoon liittyvän kokouksen, jossa käymme läpi edellisellä kerralla korjattaviksi merkityt kohdat ja katsomme, ilmeneekö uusia korjattavia asioita. Tämän lisäksi pidämme kuukausittain Pageron 14 muun tuotekehitystiimin kesken tiimikokouksia, joissa eri tiimit esittelevät vuorollaan jäsenensä ja näyttävät, minkä parissa tiimi työskentelee. Kokoonnumme myös videon välityksellä kerran kuussa niin kutsuttuun Sigweb -tapaamiseen, johon osallistuvat kaikki Pageron front-end-teknologioiden kanssa tekemisissä olevat tiimit.

Työtehtävissäni tarvitaan sujuvaa englannin kielen taitoa, hyviä tiimityöskentelytaitoja sekä kykyä itsenäiseen etätyöskentelyyn. Omaa aikaa on osattava hallita, jotta työtehtävät tulevat ajallaan valmiiksi. Työtehtävässä menestyäkseen on välttämätöntä osata vähintään ohjelmoinnin perusteet sekä hallita muun muassa HTML, CSS, Node, JavaScript, TypeScript, Git sekä StencilJS -tekniikat.

Web-kehityksessä on tärkeää ymmärtää käyttäjäkokemusta. Käytettävyys on tehtävä mahdollisimman helpoksi, sillä jos jotakin on vaikea käyttää, sitä ei käytetä (Krug 2000, 7). Komponenttikirjastoa kehittäessäni en ole joutunut käytettävyyden kanssa suoraan kosketukseen, koska olen noudattanut suunnittelijan näkemystä prototyypeistä. Menestyksellinen työsuoritus vaatii ymmärrystä natiivi-HTML:n elementtien käyttäytymisestä, SCSS-

tyylittelystä, JavaScriptin eventeistä ja metodeista sekä saavutettavuudesta (accessibility). Kehittäjän työssä tarvitaan vahvoja ongelmanratkaisutaitoja, loogista ajattelukykyä ja samaan aikaan sekä vuorovaikutustaitoja, että itsenäistä työotetta.

Reilun vuoden työskentelyn jälkeen huomaan ohjelmointiosaamiseni syventyneen merkittävästi. Lisäksi GitHubin käyttö versionhallinnan työkaluna on muuttunut rutiiniksi. Lisäksi tietämykseni suurimmista JavaScript-pohjaisista ohjelmistokehyksistä ja eri alustojen välisestä integraatiosta on lisääntynyt. Kun arvioin työtehtävissä tarvittavaa osaamista suhteessa omaan osaamiseeni, luokittelen itseni aloittelevan toimijan ja taitavan suoriutujan välille. Aloitin tehtävät reilu vuosi sitten aivan alkutasolta, sillä minulla ei ollut käytännön ohjelmointikokemusta Haaga-Helian ohjelmointikursseja ja muutamia henkilökohtaisia projekteja lukuun ottamatta. Suoriudun tehtävästä hyvin, mutta joudun toisinaan pyytämään apua kokeneemmilta kollegoilta. Web-komponenttikirjaston kokonaisuuteen liittyvissä asioissa toimin kuin kokenut asiantuntija. Mikäli jotkut web-komponentteja testaavista tiimeistä ovat löytäneet bugeja tai heillä on kysymyksiä komponentteihin liittyen, osaan auttaa heitä itsenäisesti.

Ennen IT-alalle siirtymistä työskentelin yli kymmenen vuotta ravintolan keittiössä. Ammatillisen kehitykseni näkökulmasta järjestelmäkehittäjänä koen olevani janan alkupäässä. Välttämättömät perusasiat, kuten HTML, SCSS ja JavaScript ovat jo täysin hallinnassa sekä sisäistetty. Pekka Ruohotien mukaan oppiminen on yksilössä tapahtuva muutos ja yksilö on tiedon luoja. Organisaation ei voi ajatella oppivan ilman yksilöitä, eli työntekijöitään (2000, 11.). Oman kokemukseni mukaan organisaatio ja tiimi otti minut hienosti vastaan enkä ole kokenut yhtään syrjintää sen vuoksi, ettei minulla ollut aikaisempaa kokemusta järjestelmäkehityksestä. Teknisen osaamisen lisäksi koen sisäistäneeni IT-alan työelämätaidot ja olen löytänyt paikkani tiimistä. Aikaisemmin saatoin hieman arastella teknisistä asioista puhumista, sillä minulla ei ollut paljoa kokemusta tai tietotaitoa alan termeihin liittyen. Nyt uskallan sanoa omia ajatuksiani ja mielipiteitäni rohkeammin esiin tiimien keskeisissä kokouksissa, sekä oman tiimin kesken. Vaativissa asioissa minulla ei tuota kuitenkaan ongelmia kysyä neuvoa kokeneemmilta kehittäjiltä.

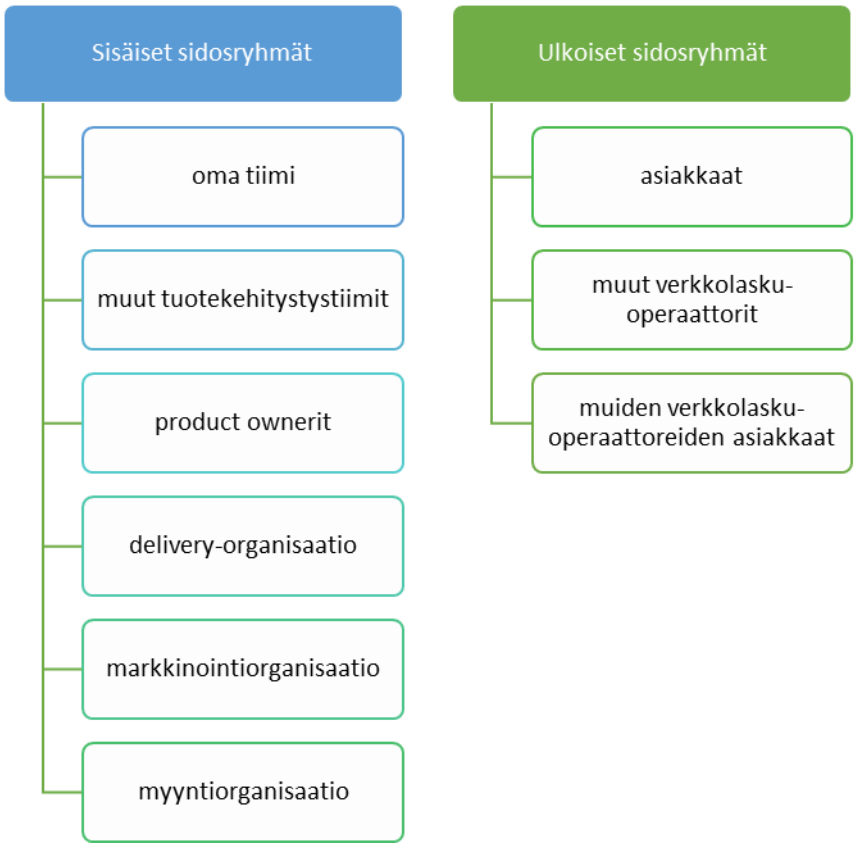
Jatkossa ammatillisen kehitykseni näkökulmasta haluan oppia enemmän mikropalveluarkkitehtuureista, sillä kehittämiäni komponentteja tullaan käyttämään muun muassa siihen, että yrityksen nykyinen valtava monoliittinen järjestelmä puretaan pienemmiksi mikropalveluiksi, joista kokonaisuus kootaan. Mikropalveluiden etuihin kuuluu muun muassa modulaarisuus, jotta monimutkaisesta järjestelmästä saadaan helpommin hallittava (Eberhard 2018, 4.1.). Lisäksi haluan kehittyä hyödyntämään scrumin sprinttejä tehokkaammin,

jonka kautta ajanhallinta paranee entisestään. Tiedän, että minun tulee tulevaisuudessa panostaa käyttäjäkokemukseen sekä saavutettavuuteen (accessibility).

Toistaiseksi kaikki kokemukseni IT-alalta on ketterästä ohjelmistokehityksestä ja vaikka ketterät menetelmät ovat suosiossa, olisi mielenkiintoista päästä jossain vaiheessa kokeilemaan muitakin ohjelmistokehitysmenetelmiä, kuten vesiputousmallia tai prototyyppimenetelmää. Tavoitteenani on tulevaisuudessa oppia suurempia linjoja järjestelmäkehityksessä, kuten esimerkiksi järjestelmän elinkaaren suunnittelua. Kollegoita seuraamalla olen saanut käsityksen, että konttiteknologian hyödyntäminen on tehokas tapa, joten sen opettelu on tehtävien listallani.

2.2 Sidosryhmät työpaikalla

Tuotekehitystiimin tehtävissä toimitaan useiden sidosryhmien kanssa ja niitä havainnollistetaan kuviossa 1. Merkittävimmät sisäiset sidosryhmät ovat oma tiimi, muut tuotekehitystiimit, tuoteomistajat, Delivery-organisaatio sekä markkinointi- ja myyntiorganisaatiot. Ulkoisia sidosryhmiä ovat taas loppuasiakkaat, muut verkkolaskuoperaattorit sekä heidän asiakkaansa.



Kuvio 1. Tuotekehitystiimin sidosryhmät

Tuotekehitystyötä tehdessä työni kannalta keskeisimpiä mielipiteitä ja intressejä edustavat oman tiimini ja tuoteomistajat. Kun web-komponentin versio 1.0 on valmis ja kun se menee laajemmin yrityksen sisällä muiden tuotekehittäjien käyttöön, korostuvat yrityksen sisäisten organisaatioiden sekä ulkoisten sidosryhmien toiveet, palaute ja intressit. Sidosryhmiä on kuunneltava lopullisen tuotteen virheettömyyden saavuttamiseksi ja tuotteeseen on pystyttävä tarvittaessa tekemään muutoksia palautteeseen perustuen.

2.3 Vuorovaikutustaidot työpaikalla

Tiimini tapaa jokaisena työpäivänä kello 10 stand up -kokouksessa, joka pidetään Google Meetin välityksellä. Kokouksen tavoitteena on käydä jokaisen tiimiläisen edellispäivän tekemiset nopeasti läpi. Jokainen kertoo myös vuorollaan kuluvan päivän suunnitelmansa. Päivittäisen kokouksen lisäksi juttelemme tiimin Google Chat -kanavalla niin työ- kuin vapaa-ajan asioista. Jokaisella tiimillä on oma Google Chat -kanavansa ja lisäksi tuotekehityksellä on yksi kaikkien tiimien yhteinen kanava. Mikäli kollega tarvitsee apua esimerkiksi koodaukseen liittyvän asian kanssa, käytämme usein Tandem -ohjelmaa parikoodaukseen. Tämä tapahtuu käytännössä siten, että toinen jakaa ruutunsa toiselle ja koodia käydään yhdessä läpi. Sähköpostitse tai puhelimitse emme juurikaan kommunikoi ja Outlookia käytämme ainoastaan kokouskutsujen lähettämiseen ja ajanhallintaan.

Ennen siirtymistäni tuotekehitystiimiin työskentelin reilun vuoden asiakastuessa, jossa olin suorassa kontaktissa loppukäyttäjärajapinnan kanssa. Nykyisessä roolissani tuotekehityspuolella tällaista kontaktia ei enää ole. Aiemmassa roolissani raportoin asiakkaan ilmoittaman ongelman eteenpäin, ja tieto siirtyi ongelman vakavuuden perusteella hierarkkisesti jopa tuotekehitykseen saakka, eli nykyiseen työhöni ratkaistavaksi. Onkin ollut mielenkiintoista seurata ongelmanratkaisua kehittäjän näkökulmasta. Aiemmassa työssäni tutuksi tullut sanasto ja käsitteet helpottavat tällaista ongelmanratkaisua nykyisessä työssäni tuotekehitystiimin tukipäivystyksen osalta, sillä ongelmatilanteissa olen vuorovaikutuksessa entisten kollegoideni kanssa.

Järjestelmäkehittäjän työssä täytyy toisinaan keskustella eri osastoilta olevien kollegoiden kanssa, jotka eivät välttämättä ymmärrä teknisiä termejä samalla tavalla kuin tuotekehittäjäkollegat. Eräs vuorovaikutuksen kannalta tärkeä taito onkin mukauttaa omaa kommunikaatiota vastaanottajalle sopivaksi. Lisäksi sujuvaan vuorovaikutukseen kuuluu muun muassa muista kulttuureista tulevien erilaisten tapojen ja näkökulmien ymmärtäminen, mikä on tärkeää monikulttuurisessa työympäristössä.

Yksi kehitystiimin sisäinen haaste on kollegoiden keskenään ristiriitaiset neuvot, tai vahvat näkemyserot eri kollegoiden välillä. Konfliktitilanteet ratkaistaan useimmiten keskustelemalla sekä tukeutumalla internetistä löytyviin luotettaviin lähteisiin, jonka jälkeen arvioidaan yhdessä, miten ehdotettu ja toisaalla toimivaksi todettu ratkaisumalli sopisi omaan tapaukseemme. Vaikka keskustelut voivat joskus venyä melko pitkiksi, olemme lopulta löytäneet ongelmiin sopivat ratkaisut.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 1

Viikon tavoitteet

Tuleva työviikko on paluu arkeen talvilomaviikon jälkeen. Jäädessäni lomalle minulla jäi web-komponenttikirjaston Github-repositorioon avoin pull request liittyen Select-komponenttiin. Alkuviikon tavoitteeni on päästä töiden osalta vauhtiin, saada rutiineista taas kiinni sekä yhdistää auki jäänyt pull request master-haaraan. Koko viikolle otan tavoitteeksi kehittää yleistä kehittäjän substanssiosaamistani opettelemalla joka päivä jotain uutta, kuten esimerkiksi uusia käsitteitä tai tapoja tehdä erilaisia asioita.

Maanantai 15.02.2021

Koska palasin juuri talvilomalta, en aseta tälle päivälle paljon tavoitteita, vaan ajattelin vain alkaa tekemään päivittäisiä tehtäviä sekä katsomaan, mitä päivä tuo tullessaan. Aloitin päivän lukemalla viikon sähköpostiviestit sekä lukemalla lokit tiimin Google Chat -kanavalta. Lomani aikana kollegani oli kirjoittanut minulle listan löytämistään bugeista sekä mahdollisista muutoksista web-komponentteihin, joten otin ne tehtävälistalleni.

Suomen Pagero Oy on muuttamassa uuteen toimistorakennukseen ensi viikolla, joten kävin hakemassa henkilökohtaiset tavarani vanhalta toimistolta. Tarkastin avoimeksi jääneen pull requestin koodikatselmoinnin kommentit ja korjattuani bugit, pull request hyväksyttiin. Tämän jälkeen yhdistin Select-komponentin pull requestin haaran repositorion master-haaraan.

Keskustelin kollegoideni kanssa web-komponentteihin liittyvien mukautettujen eventien sisällöstä. Päädyimme tutkimaan erään toisen yrityksen valmista komponenttikirjastoa ja katsomaan heidän ratkaisuaan. Iltapäivällä tiimillä oli joka toinen maanantai pidettävä backlog refinement tuoteomistajan kanssa. Backlog refinementissa puhuttiin muun muassa web-komponenttikirjastoon liittyvien design tokeneiden päivittämisestä seuraavassa sprintissä.

Koska ensimmäiselle raportointipäivälle ei ollut oikeastaan mitään tavoitteita, saavutin ne hyvin. Bugeja korjatessani törmäsin myös ensimmäistä kertaa käsitteeseen Resize Observer API, ja aloin tutkimaan käsitettä tarkemmin. Resize Observer API-rajapintaa voidaan

käyttää elementin koon muutosten seuraamiseen sekä siihen reagoimiseen (The World Wide Web Consortium, 2020.).

Tiistai 16.02.2021

Web-komponenttien testaamiseen käyttämämme Storybookin uudemmat versiot eivät toimi Internet Explorer 11 -selaimen kanssa. Tästä syystä päivän tavoitteena on saada rakennettua erillinen Storybookia käyttämätön testisivu komponenttien testaamiseen Internet Explorer 11 -selaimella. Työskentelen macOS-pohjaisella käyttöjärjestelmällä, ja tästä syystä tarvitsen VirtualBoxin kautta luotua virtuaalikonetta testatakseni komponentteja Internet Explorer 11-selainta käyttäen. Päivän toisena tavoitteenani on alkaa standardisoimaan web-komponenttien käyttämiä mukautettuja eventejä.

En päässyt päivälle asetettuihin tavoitteisiin, sillä rakennettuani sivun Internet Explorer -selaimella testaamista varten, huomasin, etteivät komponentit toimi selaimella kuten haluaisin. Kysyin neuvoa kokeneemmalta kollegaltani ja hän ehdotti, että kirjoittaisin listan löytämistäni ongelmista, jonka jälkeen katsomme yhdessä, mitä asialle voisi tehdä. Vaikka kaikki tavoitteet eivät täyttyneetkään, ei päivä mennyt silti hukkaan. Pääsin lisäämään ensimmäistä kertaa itse StencilJS -projektiin polyfillit ES5 -tukea varten ja sain nähdä käytännössä, miten se vaikuttaa projektin rakentumiseen ja toimivuuteen. Tästä huolimatta Internet Explorer 11-selaimella oli vielä ongelmia komponenttien kanssa, jotka olivat shadow DOM-puun sisällä.

Päivittäisessä stand up-tapaamisessa kuulin kollegoideni puhuvan JavaScriptin interceptoreista, mikä oli minulle jälleen uusi käsite. Aloin tutkimaan interceptoreiden käyttöä yleissivistävästä näkökulmasta ja tulevaisuutta ajatellen.

Keskiviikko 17.02.2021

Keskiviikon tavoitteenani on saada kartoitettua, mitkä kaikki asiat web-komponenteissa eivät toimi Internet Explorer 11 -selainta (IE11) käytettäessä. Listatessani ongelmakohtia, sain ajatuksen erilaisten polyfillien lisäämisestä. MDN Web Docsin (2021.) mukaan polyfillit ovat koodia, jonka avulla voidaan tuoda moderneja toiminnallisuuksia vanhempiin internetselaimiin. Etsin tietoa Googlen kautta vastaavista ongelmista StencilJS -kääntäjän ja IE11-selaimen kanssa ja löysin sopivan näköisen ratkaisun StencilJS:n virallisesta dokumentaatiosta. Lisäsin projektiin uudet polyfillit ja sain shadow DOM:n toimimaan myös Internet Explorerilla.

VirtualBoxin kanssa testaaminen ja kehittäminen on hidasta, sillä virtuaalikone toimii hitaammin kuin normaali tietokone. Tämän lisäksi itse työskentely on hidasta, sillä joka kerta tehdessäni muutoksia web-komponenttikirjastoon, joudun rakentamaan koko kirjaston uudelleen. Tämän jälkeen joudun vielä poistamaan StencilJS-pohjaisesta projektistani aikaisemmin rakennetut kansiot, jonka jälkeen rakennan projektin uudelleen korjatuilla tiedoilla ja käynnistän paikallisen serverin.

Shadow DOM:n tullessa käytettäväksi Internet Explorerilla lista komponenttien ongelmista puolittui. Käsittelin aihetta tiimin scrum masterin kanssa ja hänen mielestään ongelmia näyttäisi olevan niin vähän, että jo olemassa olevat komponentit kannattaa korjata IE11-yhteensopiviksi. Huomasin myös, että FileUploadAdvanced-komponentti, joka toimii Storybookissa, ei toimi erillisessä StencilJS-projektissa. Kyseinen komponentti on rakenteeltaan monimutkainen, joten sen päivittäminen on työlästä.

Päivän tavoitteeni tulivat jossain määrin taas täytettyä, mutta en kuitenkaan päässyt ihanteelliseen tilanteeseen. Komponenttien eventit jäivät tänään ilman huomiota, mutta ehdin alkaa korjaamaan niitä toivottavasti huomenna. Päivän aikana kehityin taas osaamissessani hieman, sillä polyfillien käyttäminen tuli entistä tutummaksi.

Torstai 17.02.2021

Päivän tavoitteena on saada FileUploadAdvanced-komponentti toimimaan IE11-selaimella, alkaa katsoa komponenttikirjaston eventejä, sekä korjata muita löytämiäni IE11 -selaimessa esiintyviä bugeja. Testatessani FileUploadAdvanced -komponenttia ilman Storybookia sain selaimen konsoliin virheviestejä. Aloin etsiä vastaavia virheviestejä Googlen-haun avulla ja löysin osumia alan sivuilta, kuten Stackoverflowsta, jossa joku toinen oli törmännyt samaan ongelmaan (TypeError: ’x’ is not iterable) muutamaa vuotta aikaisemmin. Löysin kommenttikentästä kommentin, jossa suositeltiin FileUploadAdvanced-komponentin käyttämän FileList-objektin taulukointia iteroinnin onnistumiseksi. Suositus oli toimiva ja ongelma ratkesi sen avulla.

Päivän suunnitelmat menivät jossain vaiheessa uusiksi. Julkiselle Peacock-tiimikanavallemme tuli päivän aikana muilta Pageron kehittäjiltä kyselyjä, toiveita ja tarkennuspyyntöjä liittyen heidän testaamiinsa Avatar- ja Checkbox-komponentteihin. Keskusteltuani asiasta tiimin kanssa lisäsin Checkbox-komponenttiin indeterminate-ominaisuuden. W3Schoolsin (2021.) määritelmän mukaan indeterminate-ominaisuus, joka viittaa siihen, että valintaruutu ei ole ”on” tai ”off” asennossa. Lisäksi purin Avatar-komponentin kahdeksi erilliseksi UserAvatar- sekä CompanyAvatar -komponentiksi, joista toista tullaan käyttämään

yritysten logotyypeille ja toista käyttäjien avatareille. Käyttäjän avatarin tulee olla aina pyöreä, kun taas yritysten avatarien tulee olla neliöitä. Lisäksi käyttäjän avatarissa tulee kuvan puuttuessa näkyä käyttäjän nimikirjaimet. Korjausten ja lisäysten jälkeen tein jokaisesta muutoksesta erilliset pull requestit, joista muut hyväksyttiin heti, mutta UserAvatar- sekä CompanyAvatar-komponentteihin tuli vielä muutospyyntöjä.

Päivän tavoitteeni tulivat jossain määrin saavutettua, mutta komponenttien mukautettuja eventejä en ehtinyt katsoa tänäänkään lainkaan. Substanssiosaamiseni kehittyi tänään hieman yleisellä tasolla, sillä en ollut aiemmin tietoinen natiivin HTML:n checkbox-elementin indeterminate -ominaisuudesta.

Perjantai 18.02.2021

Perjantain tavoitteeni on saada UserAvatar- sekä CompanyAvatar-komponenttien muutokset valmiiksi ja yhdistää ne web-komponenttikirjaston master-haaraan. Toinen tavoitteeni on saada lisättyä Icon-komponenttiin tausta, sillä sitä pyydettiin aamulla Peacock-tiimin Google chat -kanavalla ja tarve vahvistettiin tiimin suunnittelijan toimesta. Kolmas tavoitteeni on tutkia toisen verrokkiyrityksen komponenttikirjastoa. Aion katsoa, kuinka verrokkiyrityksessä käsitellään komponenttien mukautetut eventit, sekä ottaa yritykseltä mallia Pageron eventien standardisoimiseen.

Aloitin päivän katsomalla saamani kommentit avatar-haaran pull requestista, jonka jälkeen aloin tehdä muutoksia Avatar-komponentteihin. Saatuani muutokset valmiiksi ja lähetettyäni koodin katselmoitavaksi huomasin, että GitHubin automaattiset tarkastukset eivät menneet läpi. Aloin selvittää ongelman syytä, ja selvityksen tuloksena ilmeni, ettei komponenttien Angular-ohjelmistokehyksen versiot olleet päivittyneet Avatar-komponentin muutoksiin. Jouduin lopulta tekemään muutokset Angular-haaraan manuaalisesti, jonka jälkeen automaattitarkastukset menivät läpi ilman ongelmia.

Avatar-komponenttiin tehtyjen muutosten jälkeen aloin keskustella tiimin scrum masterin sekä suunnittelijan kanssa Icon-komponenttiin liittyvistä muutoksista. Erään toisen tuotekehitystiimin kehittäjien toiveena oli saada ikonille pyöreä tausta, jonka taustavärin vaihtoehtojen valikoima olisi ennalta rajattu. Tiimin suunnittelija teki ikoneista taustoineen mallit, joita seuraamalla jäljensin suunnitelmat koodiin.

Päivän tavoitteeni eivät täyttyneet kokonaan, sillä ikoneiden muokkaaminen osoittautui haastavammaksi, kuin miltä se aluksi vaikutti. Ikonin taustan koon on oltava riippuvainen ikonin koosta, ja ikoneita täytyy myös olla mahdollista käyttää ilman taustaa muun muassa

muiden komponenttien sisällä. Tutkin ongelmaan erilaisia ratkaisumalleja. Päädyin kokeilemaan SCSS:n mixinejä ja koitin saada asian korjattua niitä hyödyntäen. Mixinien avulla kehittäjän on helppo käyttää tietynlaista tyylittelyä eri tilanteissa kirjoittamatta valtavaa määrää koodia tyylitiedostoihin (Sass 2021.).

Perjantain tavoitteet ja suunnitellut tehtävät eivät taaskaan toteutuneet, sillä kehittäjän työssä vastaan tulee toisinaan yllättäviä muuttujia, jotka voivat viedä niin paljon aikaa, että aikaisemmin suunnitellut asiat jäävät tekemättä. Kaikesta huolimatta kehityin tänään SCSS:n mixinien käytössä.

Viikkoanalyysi

Substanssiosaamiseni järjestelmäkehittäjänä kehittyi menneen viikon aikana pieninä palasina. Opin joka päivä jotain uutta etsimällä tietoa internetistä liittyen johonkin kuulemaani tai törmäämääni asiaan. En ollut aikaisemmin ymmärtänyt täysin ResizableContainer-komponentissa käytettyä Resize Observer APIa, mutta etsittyäni siitä tietoa, sen käyttötarkoitus mielessäni selveni. Komponentti on yksi harvoista web-komponenttikirjaston komponenteista, jota en ole itse kirjoittanut, vaan se on kollegani lisäämä. Viikon aikana opin myös, että ajanhallintani tarvitsee vielä parantamista ja minun on opittava asettamaan realistisemmat tavoitteet työpäiville. Seurantaviikon aikana, maanantaita lukuun ottamatta, en yltänyt tavoitteisiini yhtenäkään päivänä.

Kuluneen viikon aikana selvitystyötä vaati esimerkiksi komponenttien yhteen sovittaminen IE11-selaimen kanssa. Vuodesta 2015 lähtien suurin osa web-kehitykseen liittyvästä koodista on kirjoitettu ES6-standardia noudattaen. ES-standardi on lyhenne JavaScriptin oikeasta virallisesta nimestä ECMAScript, ja version numero merkitään ES-lyhenteen loppuun. Jotta ES6-standardilla kirjoitettu koodi toimisi ES5-standardia tukevalla Internet Explorerilla, täytyy koodiin lisätä polyfilleja, jotka lisäävät puuttuvat ES6-ominaisuudet ES5-pohjan päälle (W3Schools 2021.). Selvitin, kuinka polyfilleja käytetään, ja miten ne vaikuttavat konkreettisesti kehitystyöhön.

IE11-selainta ei tarvitse kuitenkaan tukea enää kovin kauan. Sovimme alkuvuodesta tiimin kanssa, että kun Microsoft lopettaa IE11-tuen, niin mekin voimme lopettaa selaimen tukemisen. Tom Warrenin (2020) mukaan Internet Explorerin kehittäjä Microsoft lopettaa tuen selaimelle tämän vuoden kolmannella neljänneksellä (The Verge 2020.). Google Analytics -seurantatyökalun mukaan tällä hetkellä Pageron järjestelmän aktiivisista käyttäjistä noin 12 % on IE11-käyttäjiä, ja heitä varten sivun ylälaitaan on lisätty banneri, jossa kehotetaan vaihtamaan tuoreempaan verkkoselaimeen. Uskon kuitenkin, että polyfillien käytön

oppimisesta on hyötyä myös jatkossa muiden kehityshaasteiden yhteydessä. Tämän viikon suurimmat ongelmat koskivat Internet Exploreria, joka on kehittäjien keskuudessa tunnettu murheenkryyni. Sain onneksi suurimmat haasteet ratkaistua ja komponentit toimivat ainakin joiltain osin myös IE11-selaimella. Käytin ongelmanratkaisuun hyvin perinteisiä keinoja, eli etsin ratkaisuja Googlen lisäksi Stackoverflowsta sekä GitHubista. Toinen viikon aikana ilmennyt ongelma koski FileUploadAdvanced-komponenttia, joka ei toiminut Storybookin ulkopuolella missään selaimessa. Käytin reilusti aikaa vian etsintään, kunnes lopulta ymmärsin, etten ollut määritellyt komponentille pakollista attribuuttia ”allowedFileTypes” StencilJS-projektissani oikein. Komponentti ei siis tiennyt, mitä tiedostoja sen tulisi hyväksyä, ja hylkäsi sen vuoksi kaikki tiedostot, joita yritin lähettää sen kautta. Ratkaisu löytyi siis lopulta yrityksen ja erehdyksen kautta, ja kaikki internetistä löydetyt ratkaisuehdotukset olivat turhia. Tämä on usein hyvin tyypillistä ongelmanratkaisussa.

Todd Miller ja Ryan Ripley esittävät kirjassaan Fixing Your Scrum (2020, 14.), että scrumin sprinttien jälkeen niistä tulisi ehdottomasti järjestää jälkikatsaus. Vaikka olen tästä tavasta kuullut Haaga-Helian oppitunneillakin, niin käytännössä en ole koskaan nähnyt tiimini tällaista järjestävän. Eräänlainen jälkikatsaus järjestetään aina web-komponenttikirjaston jokaisen iteraation välissä, jossa tarkastetaan, onko aiemmin löydetyt bugit korjattu ja katsotaan, mitä onko uusia korjaus- tai kehitystarpeita ilmennyt. Mielestäni pienimuotoinen jälkikatsaus voisi olla hyödyllinen jokaisen työpäivän jälkeen. Tuolloin voisi pohtia, mitä olisi voinut tehdä omassa työssään toisin kuluneen päivän aikana.

Toimiessaan alati vaihtuvassa ympäristössä, jossa vaatimukset, toiveet sekä komponenttien ominaisuudet saattavat vaihtua hyvin tiheästi, täytyy kehittäjän pystyä työskentelemään ketterää ohjelmistokehitystä noudattaen. Muita projektihallinnan malleja olisi muun muassa vesiputousmalli, jossa yhden vaiheen on päätyttävä ennen seuraavaan siirtymistä. Kyseinen malli ei kuitenkaan omassa työympäristössäni voi toimia, sillä joudun palaamaan jopa joihinkin vuosi sitten tehtyihin komponentteihinkin yhä uudelleen vaatimusten ja toiveiden muuttuessa. Vesiputousmallia käytetään perinteisesti suuriin kertaluontoisiin ohjelmistoprojekteihin (Agendium 2020.).

Päivittäinen tekemiseni on aika usein päivästä riippumatta enemmän tai vähemmän samankaltaista, eikä uusia asioita tule opeteltua komponenttikirjaston rakentamisen ja käyttämisen ulkopuolelta. Minun on kuitenkin mahdollista päästä osallistumaan erilaisiin ja uusiin työtehtäviin tänä keväänä, kun yksi kollega tiimistäni lähtee maaliskuussa Pagerolta ja web-komponenttikirjaston versio 1.0 julkaistaan. Tällöin joitakin tiimin sisäisiä työnjakoja laitetaan hieman uusiksi, ja toivon, että pääsen kartuttamaan itselleni laaja-alaisempaa työkokemusta.

3.2 Seurantaviikko 2

Viikon tavoitteet

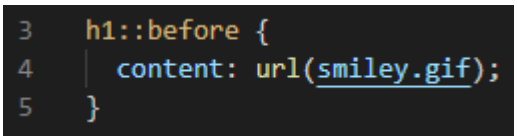
Toisen seurantaviikon tavoitteisiin kuuluvat edellisellä viikolla kesken jääneet asiat, kuten web-komponenttikirjaston mukautettujen change-eventien standardisointi. Eventien läpikäyminen ja korjaaminen oli tehtävälistallani lähes joka päivä, mutta en saanut edes aloitettua sen tekemistä yhtenäkään päivänä. Tehtävää helpottaa myös, että eräs vanhempi kollegani, jonka kanssa olen IE11-yhteensopivuusasioihin yhdessä tutustunut, palaa lomalta töihin. Uuden oppimisen kannalta aion keskittyä työn ohessa ammattitaidon kehittämiseen kokonaisuutena, eikä ainoastaan substanssiosaamisen näkökulmasta.

Maanantai 22.02.2021

Päivän tavoitteeksi asetan perjantaina aloittamani ikonien muutosten valmiiksi saattamisen. Lisäksi palaan tiimin suunnittelijan toiveesta Avatar-komponentteihin. Hän toivoi, että avatareihin upotetaan läpinäkyvä reunustus, ja että avatarien kokoja muutetaan. Mikäli vastaan ei tule yllättäviä ongelmia, ehdin ehkä alkaa katsoa viimein komponenttien eventien käsittelyä.

Päivän suunnitelma työtehtävien osalta on seuraava: Aloitan päivän kartoittamalla ikonien kanssa tarvitsemiani SCSS:n mixinejä ja kirjoitan tarvittavan koodin Icon-komponentin tyylitiedostoon. Luen tiimin suunnittelijan kirjoittamat muutospyynnöt koskien Avatar-komponentteja, kysyn tarvittaessa tarkennuksia, toteutan korjaukset ja avaan muutoksista erilliset pull requestit. Päivän kolmas tehtävä on tutustua verrokkiyrityksen komponenttikirjastoon ja verrata yrityksen mukautettuja eventejä Pageron web-komponenttikirjaston eventeihin. Mikäli aikaa jää, alan toteuttamaan eventien muutoksia.

Päivälle asettamani tavoitteet ja tehtävät onnistuivat kerrankin erinomaisesti, eikä suurempia ongelmia ilmennyt. Lisäsin ikoneihin taustan rikkomatta prosessissa muita komponentteja, joissa ikonia käytetään. Avatar-komponenttien muutokset valmistuivat ja samalla kartutin lisää kokemusta CSS:n pseudo-elementtien käytöstä. W3Schools (2021.) määrittelee pseudo-elementin niin, että sillä voi muun muassa lisätä sisältöä elementin eteen tai taakse. Kuvassa 1 on kuvattu CSS:n before-pseudoelementti, joka lisäisi jokaisen <h1>-elementin, eli ison otsikkoelementin eteen kuvan.



Kuva 1. CSS:n ::before-pseudoelementti (mukaillen W3Schools 2021)

Tutustuin myös verrokkiyrityksen komponenttikirjaston eventien käsittelyyn lukemalla heidän virallista dokumentaatiotaan. Heidän event-käsittelynsä on suurilta osin sama kuin Pagerolla. Standardoidessani web-komponenttikirjaston mukautettuja eventejä kaikki sujui hyvin, kunnes Input-komponentin kanssa eteen tuli ongelma. Input-komponentti ei lähetä ulos change-eventiä, vaikka se on kirjoitettu aivan samalla lailla, kuin kaikissa muissakin komponenteissa.

Tiistai 23.02.2021

Päivän ensimmäinen tavoite on saada Input-komponentti lähettämään mukautettu change-event käyttäjän kirjoittaessa jotakin input-kenttään. Haluan hoitaa asian heti aamulla pois päiväjärjestyksestä, jotta saan hyvissä ajoin standardoitua komponenttien eventit ja pääsen avaamaan muutoksista pull requestin. Tämän jälkeen olen suunnitellut pyytäväni vanhemman kollegan Tandem-ohjelman kautta parikoodaussessioon koskien muutoksia, jotta saisimme kaikki komponentit toimimaan hyvin myös IE11-selaimella. Vaikka ne tällä hetkellä jossain määrin toimivatkin polyfillien ansiosta, ei niiden tyylittely ole niin huoliteltua Internet Explorerilla, kuin muilla selaimilla.

Aamulla avattuani työkoneen huomasin, että julkisella Peacockin Google chat-kanavalla oli keskusteltu Select-komponentista löytyneestä bugista. Eräs Woodpecker-tiimin, eli Pageron toisen tuotekehitystiimin kehittäjä oli käyttänyt komponenttia React-pohjaisessa sovelluksessaan, mutta hänen käyttämänsä Select-komponentin sisältämät vaihtoehdot (option) tulostuivat jostain syystä komponentin ulkopuolelle. En onnistunut tuottamaan bugia omassa kehitysympäristössäni, joten pyysin häntä toisintamaan ongelman jossain pienessä irrallisessa repositoriossa, jotta voisin tutkia sitä tarkemmin. Päivittäisessä stand up-kokouksessa tiimiläiset muistuttivat, että tänään on myös kuukausittainen tiimikokous, jossa tekemisensä esittelevät tuotekehitystiimit Puffin, Scrooge sekä Woodpecker.

Kokouksen jälkeen tiimin suunnittelija sekä seniorit halusivat lisätä Avatar-komponentteihin vielä yhden uuden ominaisuuden. Jotta Avatar-komponenttia voidaan käyttää yritysten logotyyppien vaihtamiseen, sen täytyy olla tarvittaessa klikattava. Komponenttiin täytyy lisätä mahdollisuus klikkaukseen, selkeä visuaalinen ikoni sekä käyttäjän kursorin kanssa

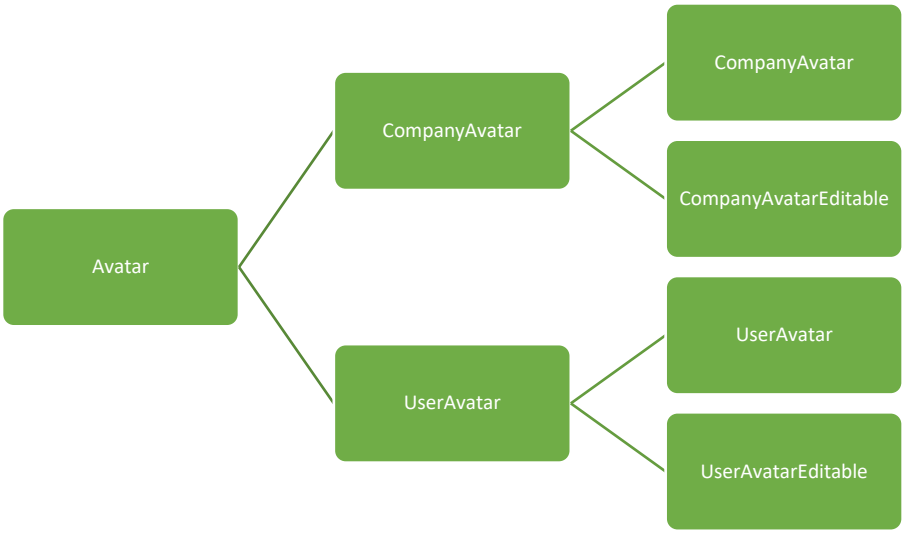
vuorovaikutuksessa toimiva palaute. Lisäksi komponenttiin on lisättävä mukautettu change-event, jonka tulee käynnistyä käyttäjän painaessa ”muokkaa”-painiketta.

Päivän tavoitteet ja suunnitelmat täyttyivät vain osittain. En saanut vieläkään selville, miksi Input-komponentin change-event ei toimi. En ehtinyt käydä Tandem-keskustelua vanhemman kollegani kanssa IE11-muutoksista, enkä ehtinyt alkaa korjata IE11-selaimeen liittyviä ongelmia. Päivän hyvä puoli on, että sain yhtenäistettyä komponenttien eventit ja pääsin avaamaan niistä pull requestin. Lisäksi sain tehtyä Icon-komponentin pull requestiin pyydetyt muutokset, joten toivottavasti pääsen aamulla yhdistämään ne master-haaraan ja siirtymään seuraaviin tehtäviin.

Keskiviikko 24.02.2021

Keskiviikkona otan ainoaksi tavoitteekseni edes aloittaa IE11-selaimeen liittyvät korjaukset komponentteihin. Lisäksi minulle jäi tiistailta web-komponenttikirjaston GitHub -repositorioon kolme avointa pull requestia, joita ei ole vielä hyväksytty, mutta jotka haluaisin saada yhdistää master-haaraan. Tiedän, että ainakin Avatar-komponentteihin täytyy vielä tehdä muutoksia, sillä komponenttien change-eventeissä on samanlaisia ongelmia, kuin Input-komponentissa.

Tarkistin pull requestit heti avattuani työkoneen ja Icon-komponentin pull request oli hyväksytty. Avatar-komponentin pull requestiin oli tullut muutamia muutospyyntöjä, sekä huomautus onClick-funktiokutsun lopusta puuttuvista sulkeista, minkä vuoksi event ei käynnistynyt oikein. Korjasin Avatar-komponentteja, kunnes päivittäinen stand up-kokous alkoi. Kokouksessa päätettiin, että aiemmin Avatar-komponentista irtautuneet CompanyAvatar sekä UserAvatar -komponentit puretaan vielä kahteen osaan, joista toisen tulee olla muokattava, ja toisen normaali avatar. CompanyAvatar komponentista tehdään CompanyAvatar-, sekä CompanyAvatarEditable -komponentit. Pidimme stand up-kokouksen jälkeen erillisen kokouksen tiimin Front-end-teknologioiden parissa työskentelevien henkilöiden kanssa, jossa tarkemmat yksityiskohdat Avatar-komponenttien purkamisesta sovittiin. Avatar-komponenttien kehityskulku on kuvattu kuviossa 2.



Kuvio 2. Avatar-komponentin evoluutio

Päivän suunnitelmat menivät jälleen uusiksi jo ennen lounasta. En ehtinyt edes vilkaista tavoitteeksi asettamiani IE11-korjauksia, vaan työstin koko päivän CompanyAvatar-komponentin kahtiajakoa. Lähetin Avatar-komponenttien purkamisesta syntyneen koodin katselmoitavaksi ja aloin tarkastaa kommentteja event-käsittelyn pull requestista, jonka jälkeen aloin tekemään event-käsittelyyn toivottuja muutoksia.

Torstai 25.02.2021

Päivän tavoite on saada uusi CompanyAvatarEditable-komponentti julkaisuvalmiiksi. Aloitan päivän katsomalla pull requestin kommentit, jonka jälkeen alan työstää pyydettyjä muutoksia. Päivän tehtävälistalla on myös eventien pull requestin muutospyyntöjen korjaaminen.

Aloin tehdä eventien muutoksia heti aloitettuani työt ja sain muutokset valmiiksi lounaaseen mennessä. Lounaan jälkeen aloin korjata CompanyAvatarEditable-komponenttiin pyydettyjä muutoksia, joiden spesifikaatiot menivät jälleen uusiksi. Uusien spesifikaatioiden myötä komponentin tulee automaattisesti hakea järjestelmästä kohdeyrityksen logo ja näyttää se avatarissa, kun komponenttiin syötetään ”companyId”-attribuutin arvoksi Pageron Company API-rajapinnassa oleva yrityksen id-numero.

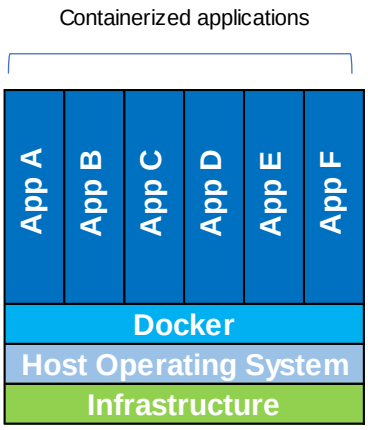
Päivästä jäi kokonaisuutena hyvä fiilis, vaikka en koko tavoitteeseen yltänytkään spesifikaatioiden mennessä uusiksi. Pääsin työskentelemään uuden asian parissa, eli hakemaan yrityksen sisäisistä API-rajapinnoista yritysten tietoja ja hyödyntämään niitä web-komponentissa. Lisäksi opin uuden tavan hyödyntää CSS-luokkia, eli opin, kuinka pystyn

muokkaamaan esimerkiksi kahden elementin ulkonäköä samaan aikaan käyttäjän viedessä kursorin toisen elementin päälle.

Perjantai 26.02.2021

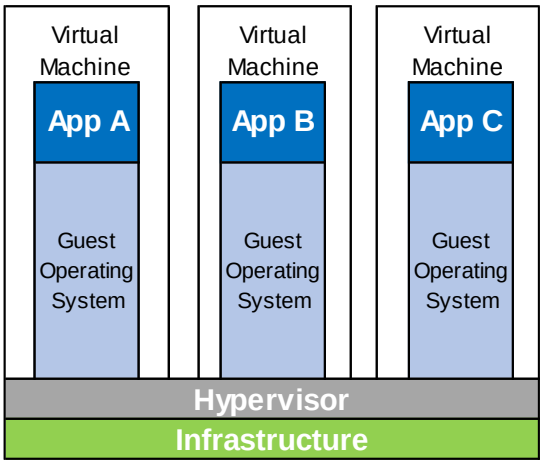
Sovin vanhemman kollegani kanssa eilen, että pidämme Tandem-session aamulla liittyen Docker- ja Postman -ohjelmien käyttöön, ja lisäksi katsomme mahdollisesti myös IE11-bugeja. Olin ladannut ohjelmat koneelleni jo aikaisemmin, mutta en ollut juuri koskaan käyttänyt niitä. Docker on teknologia, jonka avulla IT-ympäristön uudistamista voidaan nopeuttaa ja jonka avulla kehittäjä voi laittaa sovelluksia omiin kontteihinsa, sekä ajaa niitä näistä erillisistä konteista (Wallenius 16.11.2020.).

Dockerin toimintaperiaate havainnollistetaan kuviossa 3. Kuviosta näkyy, että yhdellä koneella (infrastructure) olevalla käyttöjärjestelmällä (host operating system) voidaan ajaa montaa eri konttia (app A-F) Dockerin sisällä.



Kuvio 3. Dockerin toimintaperiaate (mukaillen Fong 2018)

Kontit eroavat virtuaalikoneista siten, että ne eivät varaa resursseja etukäteen, vaan pyytävät käyttöjärjestelmältä resursseja vain itse varsinaista ajoa varten (Wallenius 16.11.2020.). Virtuaalikoneiden toimintaa havainnollistetaan kuviossa 4. Toisin kuin Docker-konttiteknologiassa, jokainen virtuaalikoneessa pyörivistä sovelluksista (app A-C) vaatii oman tietokoneen (infrastructure). Virtualisointialustaa (hypervisor) tarvitaan virtuaalikoneen sisällä olevien käyttöjärjestelmien (guest operating system) ajamiseen.



Kuvio 4. Virtuaalipalvelimien toimintaperiaate (mukaillen Fong 2018)

IE11-bugeihin katsominen on jäänyt kaikkien uusien asioiden jalkoihin koko viikon. Päivän tavoitteeni on säätää Docker-asetukset kuntoon sekä edistää CompanyAvatarEditable-komponenttia.

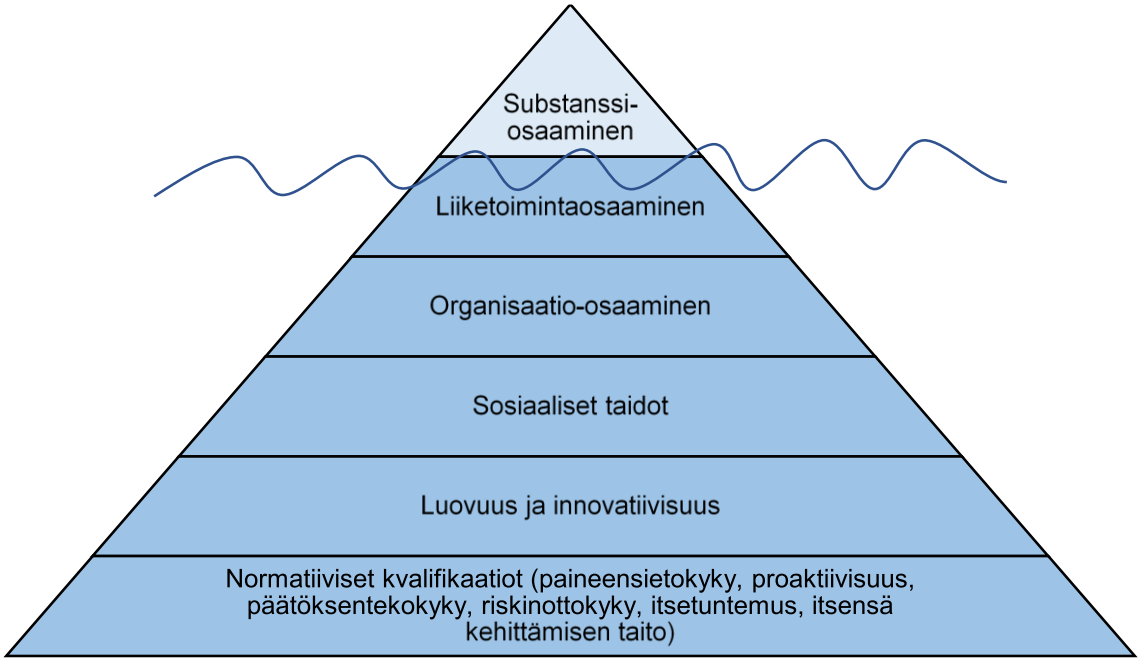
Päivän tavoitteet onnistuivat siinä määrin hyvin, että sain Dockerin asennettua sekä konfiguroitua toimintavalmiiksi. Kollegani opetti minulle myös Dockerin käyttöä ja sain selkeämmän kuvan sen käyttötarkoituksesta, sekä kokemusta varsinaisesta käytöstä. Olen toki lukenut Docker-konttimenetelmistä aikaisemminkin, mutta tänään pääsin ensimmäistä kertaa konkreettisesti ajamaan 30 konttia samaan aikaan.

Postman on monitoimityökalu, jonka avulla voidaan suunnitella, testata, dokumentoida, seurata sekä julkaista rajapintakuvauksia (Postman 2021.). Konttiteknologian hyödyntämisen lisäksi opettelin käyttämään Postman -sovellusta kuvien lähettämiseen Pagero Ab:n developer-testiserverille POST-kutsulla, sekä käyttämään Postman-sovellusta kuvien hakemiseen GET-kutsulla. Lisäksi sain rakennettua CompanyAvatarEditable -komponentin ainakin teoriassa valmiiksi. En pääse kuitenkaan testaamaan sen käyttöä käytännössä tietokoneellani olevalta paikalliselta testiserveriltä Storybookin kautta, sillä sekä POST-, että GET-kutsut Company API-rajapinnassa vaativat käyttäjän olevan kirjautuneena Pageron järjestelmään. Maanantaina aloitan päivän testaamalla projektia Pageron testiserverin kautta kontteja hyödyntäen.

Viikkoanalyysi

Valitsin maanantaina koko viikon tavoitteeksi ammattitaidon kokonaisuuden kehittämisen. Garavan ja McGuire (2001.) vertaavat työntekijän ammattitaidon kokonaisuutta jäävuoreen, jossa vain ylin huippu eli taidot ja tietämys voivat olla näkyvissä. Jäävuoren veden

pinnan alle jäävä osaa taas koostuu edellisten ilmenemistä ja kehitystä säätelevistä tekijöistä, joita ovat esimerkiksi sosiaalinen rooli, minäkäsitys, motiivit ja luonteenpiirteet. Viimeksi luetellut ominaisuudet kehittyvät ihmisenä kasvamisessa, kun taas jäävuoren ylemmissä kerroksissa olevia asioita voidaan kehittää koulutuksilla. Jäävuori havainnollistetaan kuviossa 5.



Kuvio 5. Ammattitaidon kokonaisuuden jäävuorimalli (mukaillen Viitala 2007, 146.)

Koen työtehtäviin liittyvän substanssiosaamisen kehittyneen viikon aikana merkittävästi, sillä opin uutta CSS:n pseudo-elementeistä. Vielä tärkeämpänä koen kuitenkin Docker- ja Postman -ohjelmien käytön oppimisen. Pääsin vihdoin kunnolla kiinni yrityksen sisäiseen kehitysympäristöön sen sijaan, että työstäisin vain erillisiä komponentteja omalla paikallisella palvelimellani.

> Dockerin etuina virtuaalikoneiden käyttöön verrattuna on, että virtuaalikoneen käyttäessä omaa käyttöjärjestelmäänsä ja toimiessa ylimääräisenä tietokoneena, Docker jakaa isäntäkoneen käyttöjärjestelmän resursseja tarvittaessa itselleen ja käyttää niitä ainoastaan tarvittavan määrän. Tämän ansiosta Docker-kontit ovat virtuaalikoneita nopeampia, helpompia käyttää sekä monipuolisempia, sillä kontteja voi käyttää monelta työasemalta samanaikaisesti. (Thomas 2.4.2019)

Viikon aikana selvitin muun muassa tapoja saada Avatar-komponentteihin osittain läpinäkyvä reunustus, sillä normaaleilla CSS:n border, box-shadow tai outline -metodeilla se ei onnistunut. Selvitin myös muun muassa Docker-konttiteknologian sekä Postman -rajapintasovelluksen käyttötarkoituksia.

Kuluneen viikon alussa minulla oli haasteita Input-komponentin mukautetun eventin lähettämisen kanssa, sillä en nähnyt mitään syytä, miksi komponentti ei lähettäisi eventiä. Ongelman taustalla oli Storybookin runScript-lisäosa, jota jouduimme aikaisemmin käyttämään, jotta voisimme käyttää <script>-tageja komponenttien tarinoissa. Viime vuoden lopulla päivitimme Storybookin uudempaan 6.0 versioon ja päivityksen myötä skriptejä pystyi kirjoittamaan tarinoihin ilman lisäosaakin. Lisäosa runScript jäi kuitenkin jo olemassa oleviin tarinoihin käyttöön ja jostain syystä se esti kuuntelijan toiminnan Input-komponentin tarinassa. Poistettuani lisäosan kuuntelija sai mukautetusta eventistä kiinni aivan normaalisti ja ongelma korjautui.

Pohdimme tiimin kanssa, onko tietoturvan kannalta suuri riski, jos asiakasyrityksen markkinointitarkoitukseen tehtyjä logoja on mahdollista nähdä kirjautumatta palveluun CompanyAvatarEditable-komponentista. Ymmärsimme kuitenkin, että koska käyttäjien henkilökohtaisiin UserAvatarEditable-komponentteihin liitetään mahdollisesti käyttäjien kuvia, tietoturva-näkökulma on otettava avatareissa huomioon. Edellä mainittuun tietoturva-näkökulmaan liittyvä ongelma oli, että kuinka saan haettua CompanyAvatarEditable-komponenttiin yrityksen logon pelkällä id-numerolla. Löysin ratkaisun twelve21-sivun tietoturvaan liittyvästä blogista. Kirjoituksessa esiteltiin erilaisia tietoturvallisia tapoja kuviin käsiksi pääsemiseen OAuth 2.0-autorisointiprotokollan avulla. Valitsemassamme data URI-menetelmässä valittava kuva haetaan dynaamisesti järjestelmästä haetun id-numeron perusteella, mutta haun onnistumiseen vaaditaan, että käyttäjä on kirjautunut sisään palveluun. Mikäli kuvan haku epäonnistuu, eli jos käyttäjä ei ole kirjautunut sisään tai kuvaa ei ole määritelty, avatarissa näkyy pelkkä ikoni. Menetelmän etuja on, että se on kaikista tietoturvallisin vaihtoehto, ja että kuva pystytään tallentamaan käyttäjän selaimen välimuistiin. Huonoja puolia puolestaan ovat menetelmän monimutkaisuus sekä data URI:n mahdollinen kokorajoitus, vaikka Internet Engineering Task Forcen (IETF) kirjoittamassa ”data” URL scheme -spesifikaatiossa (1998.) lukee, ettei tarkkaa maksimikokoa ole tiedossa (Burman 11.3.2019.). IETF on johtava internetin standardien luoja. Yhteisö on perustettu tammikuussa 1986 ja se koostuu vapaaehtoisista, jotka työskentelevät avoimen lähdekoodin parissa erilaisissa ryhmissä. Heidän missionaan on luoda kaikille käyttäjille parempi internet korkealaatuisten relevanttien teknisen dokumentaation kautta, mikä vaikuttaa siihen, kuinka ihmiset kehittävät, käyttävät ja hallinnoivat internetiä (IETF 2021.).

Kokonaisuutena kulunut viikko oli mielestäni menestys, sillä opin paljon uusia ja tulevaisuutta ajatellen hyödyllisiä taitoja. Odotan mielenkiinnolla tulevan viikon haasteita sekä mahdollisuuksia oppia uutta. Maanantaina tiimissämme aloittaa myös uusi jäsen, vaikkakin jossain vaiheessa keväällä tai kesällä tiimejä tullaan järjestämään jonkin verran uusiksi.

3.3 Seurantaviikko 3

Viikon tavoitteet

Kolmannen raportointiviikon tavoitteeni on oppia Dockerin monipuolinen käyttö sekä työstää CompanyAvatarEditable- sekä UserAvatarEditable-komponentit julkaisuvalmiiksi. Mahdollisuuksien mukaan haluan myös syventää osaamistani Postman-työkalun käytössä.

Maanantai 01.03.2021

Päivän tavoitteena on saada kuvien vastaanottaminen toimimaan CompanyAvatarEditable-komponentin sisäisellä GET-kutsulla Pageron Ab:n Appstore-palvelun dev-kehitysympäristössä, minkä paikallista versiota ajan Docker-kontissa. Mikäli saan komponentin ominaisuudet toimimaan spesifikaatioiden mukaisesti, avaan muutoksista pull requestin. Iso osa päivästä tulee kulumaan kokouksissa, sillä päivittäisen stand up-kokouksen lisäksi tänään on kahden viikon välein pidettävä backlog refinement.

Aloitin päivän testaamalla CompanyAvatarEditable-komponenttia Docker-kontissa ajettuna. Ilokseni huomasin, että perjantaina tekemäni muutokset näkyvät kehitysympäristössä. Komponentin muokkaaminen oli todella hidasta, sillä aina kun tein komponenttiin pienenkin muutoksen, jouduin rakentamaan neljä eri projektia uudelleen nähdäkseni muutoksen vaikutukset. Tähän uudelleenrakentamiseen meni aina noin 35 minuuttia kerrallaan. Pidin vanhemman kollegani kanssa Tandem-tapaamisen, jossa katsoimme koodia ja prosessia yhdessä. Pienen pohdinnan jälkeen saimme kirjoitettua skriptin, jolla rakentamisaika saatiin lyhennettyä noin 35 minuutista 15 minuuttiin. Muutos nopeutti prosessia, mutta en kuitenkaan päässyt aivan päivän tavoitteeseeni kokousten viedessä niin ison osan päivästä. Tiimissä aloitti tänään myös uusi työntekijä. Helpottaaksemme hänen perehtymistään kävimme kokouksissa kaikki asiat läpi alusta lähtien ja tästä syystä kokoukset venyivät normaalia pidemmiksi.

Sain CompanyAvatarEditable-komponentin näkyviin Docker-kontissa paikallisesti ajetussa Pagero Ab:n Appstore-projektissa. Komponentin on tarkoitus lähettää käyttäjän lataama logo vastaanottavalle palvelimelle. Kuva lähetetään base64-enkoodattuna, jotta vastaanottavan palvelimen API-rajapinta ymmärtäisi tiedoston sisällön, mutta valitettavasti kuva, jonka pitäisi näkyä komponentissa staattisen ikonin tilalla, palautuu serveriltä ”undefined”-tilassa. Base64-enkoodauksessa tiedosto muutetaan binäärimuodosta ASCII-muotoon, jotta data voitaisiin tarvittaessa siirtää esimerkiksi sähköpostin kautta (MDN Web Docs

2021.). Epäilen ongelman liittyvän komponentin elinkaareen, sillä komponentti renderöityy, eli ilmestyy näkyväksi sivulle ennen komponentin sisällä tapahtuvaa GET-kutsua, mutta palvelimelta haettu kuva ei enää päivity avatariin kutsun vastauksen saapuessa.

Tiistai 02.03.2021

Päivän tavoitteeni on saada CompanyAvatarEditable-komponentin kuvan lataaminen toimimaan testiympäristössä. Kollegani Albatross-tiimistä pyysi Close-eventin lisäämistä Alert-komponenttiin ja tiimi hyväksyi pyynnön. Päivän toinen tavoitteeni onkin lisätä Alert-komponenttiin mukautettu Close-event, joka käynnistyy loppukäyttäjän painaessa komponentin sulkupainiketta.

Huomasin aamulla työkoneessani macOS-päivityksen ja ajattelin, että koska stand up-kokoukseen on vielä tunti, ehdin hyvin päivittää käyttöjärjestelmän. Laitoin päivityksen pyörimään ja siihen menikin 95 minuuttia, enkä ehtinyt tiimin päivittäiseen kokoukseen lainkaan. Kirjoitin asiasta Peacockin sisäiselle Google chat-kanavalle ja ilmoitin, mitä aion tulevana päivänä tehdä.

Koska käyttöjärjestelmä päivittyi ja kone täytyi käynnistää prosessissa uudelleen, kaikki aiemmin päälle asettamani Docker-kontit menivät pois päältä. Käytännössä käytin koko päivän siihen, että saan kaiken taas päälle ja toimintakuntoon. Päivän päätteeksi kaiken toimiessa normaalisti pääsin vihdoin tekemään muutoksia komponenttiin ja testaamaan muutoksia kehitysympäristössä. Lopulta CompanyAvatarEditable-komponentti haki avatariin kuvan Pageron Company API-rajapinnasta ”companyId”-attribuutin arvon perusteella, mutta kuva rajautui hieman omituisesti, sillä koko logon pitäisi olla näkyvissä pelkän keskiosan sijaan.

Keskiviikko 03.03.2021

Päivän tavoitteenani on rajata CompanyAvatarEditable-komponentissa käytettävä yrityksen logotyyppi oikein. Ulkoasusuunnittelijan toiveesta aion myös lisätä avatariin logotyypin tiedoston koon tarkastuksen. Käyttäjän ladatessa yrityksen logotyypin palvelimelle latautuu logosta automaattisesti kaksi eri kokoista versiota. Mikäli avatarin koko on pieni (sm) tai todella pieni (xs), voidaan yrityksen logona näyttää pienempi logotyyppi, jolloin loppukäyttäjälle tulee vähemmän dataa ladattavaksi selaimen kautta.

Tavoitteeni on myös ainakin aloittaa Dialog-komponentin lisääminen CompanyAvatarEditable-komponenttiin. Dialogi-ikkuna avautuu käyttäjän klikatessa CompanyAvatarEditable-

komponenttia ja dialogin kautta loppukäyttäjä voi vaihtaa avatarissa näkyvän yrityksen logotyypin. Mikäli aikaa jää, lisään Alert-komponenttiin pyydetyn Change-eventin.

Tänään päivän tavoitteet täyttyivät hienosti. Kuva näkyy avatarissa haluamallani tavalla ja sen lisäksi komponenttia klikkaamalla aukeaa dialogi-ikkuna, jonka kautta loppukäyttäjä voi ladata Company API-rajapintaan yritykselleen uuden logotyypin. Lisäksi minulla oli aikaa kirjoittaa Alert-komponenttiin pyydetty Close-event. Avasin Close-eventin muutoksesta uuden pull requestin ja se hyväksyttiin saman päivän aikana, kuin myös pull request muidenkin eventien standardoinnista. Pääsin vihdoin yhdistämään avoimet haarat master-haaraan.

Torstai 04.03.2021

Tavoitteenani tänään on korjata CompanyAvatarEditable-komponentin uuden kuvan lähetys toimimaan niin, että kuva menee tietokantaan Pageron dev-testiyrityksen logotyypiksi. Päämääränä on, että uutta logotyyppiä pystyy esikatselemaan komponentista avautuvan dialogi-ikkunan sisältä ennen logotyypin muutoksen lopullista hyväksyntää.

Päivän tavoite ei täyttynyt odotetusti, sillä vastaanottavassa Company API-rajapinnassa oli jotain vialla. Postman-sovelluksella tekemäni testilähetykset onnistuivat, mutta tuntemattomasta syystä testisivustolta lähettämäni POST-kutsu palautti virhekoodin 400, joka on yleinen koodi ”Bad request”-virheelle. Virhekoodi tarkoittaa sitä, että vastaanottava palvelin ei pysty, tai suostu käsittelemään kutsua (MDN Web Docs 2021.). Ilmoitin asiasta tiimiläiselleni, joka on rakentanut vastaanottavat API-rajapinnat ja hän lupasi tutkia asiaa. Hänen kirjoittamansa dokumentaation perusteella kutsuni olisi pitänyt toimia sellaisenaan.

Torstai ei kuitenkaan mennyt aivan hukkaan, sillä opin paljon POST-kutsujen lähettämisestä yrittäessäni etsiä ratkaisua ongelmaan. Opin muun muassa, että voin muuntaa koodissa käyttäjän valitseman kuvan string-muotoon FileReader-Web API-työkalua hyödyntäen. Kyseinen skenaario on hyödyllinen esimerkiksi silloin, kun haluan lähettää kuvan JSON-muodossa tietokantaan. FileReader Web API-työkalussa on monia tahdistamattomia, eli asynchronous-metodeja, joiden avulla kehittäjä voi muokata käyttäjän valitsemia tiedostoja haluamallaan tavalla (MacDonald 2014, 335.).

Perjantai 05.03.2021

Perjantain tavoite on saada vihdoinkin CompanyAvatarEditable-komponentti valmiiksi web-komponenttikirjaston version 1.0 julkaisua varten. Toivon, että API-rajapintoja

rakentava kollegani olisi keksinyt, miksi POST-kutsu epäonnistuu ja palauttaa ainoastaan virhekoodin 400.

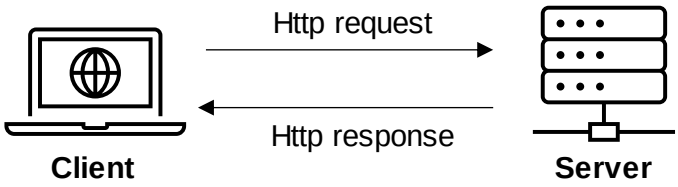
Aamun aikana löysin syyn POST-kutsun epäonnistumiseen. Syyoli inhimillisessä virheessä, tarkemmin sanottuna logotypeType-parametrin arvon virheellisessä kirjoitusasussa. Company API-rajapinnan dokumentaatiossa mainittiin, että arvo kirjotetaan normaalisti käyttämäämme camelCase-kirjoitusasua käyttäen. Virheen vuoksi rajapintaan oli kuitenkin laitettu kirjoitusasuksi PascalCase-kirjoitusasu, joka aiheutti kutsun epäonnistumisen. Kirjoitusasut eroavat toisistaan siten, että camelCasessa ensimmäinen kirjain on pienellä alkukirjaimella ja sen jälkeen jokaisen sanan ensimmäinen kirjain kirjoitetaan isolla alkukirjaimella. PascalCasessa puolestaan myös ensimmäisen sana kirjoitetaan isolla alkukirjaimella, jolloin parametrin olisi toimiakseen pitänyt olla muodossa LogotypeType.

Saatuani logotyypin palvelimelle lataamiseen tarvittavan POST-kutsun toimimaan, ryhdyin työstämään varsinaista CompanyAvatarEditable-komponenttia. Toimivuuden kannalta suuri ongelma oli, että komponentin koodin alkupäässä esittelemäni globaalit muuttujat eivät suostuneet toimimaan funktioiden sisällä, eli ne olivat funktion scopen ulkopuolella. Etsin syytä internetistä Googlen kautta lähes koko päivän. Vihdoin löysin eräästä Stackoverflow:n keskustelusta vinkin, jossa kehotettiin käyttämään syntaksiltaan tuoreempia nuolifunktioita normaalien funktioiden sijaan. Nuolifunktioita käyttämällä sain komponentin vihdoin toimimaan haluamallani tavalla pieniä muutostarpeita lukuun ottamatta.

Opin paljon uutta FileReader-työkalusta, POST-kutsuista sekä funktioista tänäänkin, eli päivä ei mennyt hukkaan. Osa asioista oli sellaisia, jotka olin opetellut jo vuosia sitten, mutta olin unohtanut ne, sillä niille ei ollut aikaisemmin ollut konkreettista käyttöä.

Viikkoanalyysi

Seurantaviikon aikana työskentelin pääsääntöisesti vain yhden komponentin parissa, mutta opin silti paljon uutta. CompanyAvatarEditable-komponentin toimintatapa pakotti minut pois mukavuusalueeltani ja opettelemaan paremmin HTTP-metodien, kuten GET- ja POST-kutsujen hyödyntämistä. HTTP toimii kutsu-vastaus (request-response) protokollan mukaan käyttäjän ja palvelimen välillä. Prosessi on kuvattu yksinkertaisesti kuviossa 6.



Kuvio 6. HTTP kutsu ja vastaus (mukaillen BytesofGigabytes 2021)

Käyttäjän kirjautuessa Pageron alustaan omilla käyttäjätunnuksillaan selaimen välimuistiin tallentuu väliaikainen access token, eli käyttöoikeustunnus. Kun loppukäyttäjä valitsee graafisen käyttöliittymän kautta uuden kuvan yrityksen logotyypiksi, laukaisee muutos komponentissa Change-eventin. Change-eventin huomatessaan komponentin sisäinen tapahtumakuuntelija laukaisee ketjun toimintoja komponentin sisällä. Ensin tiedosto käännetään FileReaderia hyödyntäen string-muotoon. Tämän jälkeen komponentti varmistaa, että käyttäjällä on oikeudet vaihtaa yrityksen logotyyppi palveluun tarkistamalla internetselaimesta käyttöoikeustunnuksen kelvollisuuden. Mikäli oikeudet ovat kunnossa, tiedosto lähetetään palvelimelle POST-kutsun (request) sisällä JSON-muodossa ja lopulta palvelimelta tulee kutsuun vastaus (response), kuten kuviossa 6 on kuvattu. Mikäli vastauksen sisältö on kelvollinen ja virhettä ei tapahtunut, logotyyppi vaihdetaan uuteen.

Viikon aikana joudun selvittämään enimmäkseen asioita, jotka liittyivät HTTP-metodeihin tai ylipäätään tiedostojen käsittelyyn. Aiheet ovat sellaisia, joita olen käsitellyt työssäni joskus aiemminkin, mutta osaamiseni on syventynyt kuluneen viikon aikana. Viitala (2007, 147.) kuvailee osaamisen kehittämistä loppumattomaksi prosessiksi, joka on havainnollistettu kuviossa 7.

Kuvio 7. Osaamisen kehittämisen prosessi (mukaillen Viitala 2006, 87.)

Kuviossa 7 havainnollistetun prosessin mukaan oman osaamisen tunnistamisen sekä arvioinnin jälkeen osaamista aletaan vertailemaan tulevaisuuden tarpeisiin. Tämän jälkeen osaamista kehitetään, kunnes taas tullaan tunnistamisen sekä arvioinnin vaiheeseen ja sykli alkaa alusta. Koen samaistumista ympyrän prosessin kanssa pohtiessani omaa toimintamalliani muun muassa HTTP-metodien opettelussa. Jokaisen kerran syklin läpi käytyäni koen osaamiseni syventyneen ja osaan katsoa aihetta erilaisista näkökulmista.

Myöhemmin ajateltuna ongelmat, joihin jäin viikon aikana jumiin, olivat todella yksinkertaisia. Kokemukseni ja kollegoilta kuulemani perusteella on tyypillistä, että retrospektiivistä katsottuna triviaalinkin ongelman selvittämiseen voi mennä aikaa. Syy on usein väärä kirjoitusasu, tai esimerkiksi koodista puuttuva merkki. Kun eri tyyppisten yksinkertaisten ongelmien kanssa on kerran jo hakannut päätä seinään, niin seuraavalla kerralla saman ongelman luultavasti osaa sulkea pois laskuista jo varhain ongelmanratkaisua tehdessä.

Tällä viikolla tekemäni kehitystyö on ollut normaalia hitaampaa, sillä olen joutunut testaamaan komponenttia Pageron dev-ympäristössä paikallisen Storybookin sijaan. Tämän seurauksena pelkkään testiprojektin rakentamiseen menee yhteensä noin 10–15 minuuttia. Lisäksi kahden seuraavan viikon kehitystyötä tulee hieman hidastamaan se, että tiimin scrum masterin lomaillessa kehittämäni koodia tarkastavat hänen sijaansa enimmäkseen tiimin muut seniorikehittäjät. Toisaalta taas on kiinnostavaa saada palautetta myös muilta kehittäjiltä heidän omista näkökulmistaan.

3.4 Seurantaviikko 4

Viikon tavoitteet

Viikon tavoite on saada CompanyAvatarEditable-komponentti valmiiksi, jotta voin siirtyä korjaamaan loppuja bugeja web-komponenttikirjaston version 1.0 julkaisua varten. Lisäksi viikon aikana on minun vuoroni hoitaa tiimille tulevia tukitikettejä kokeneemman seniorikehittäjän kanssa.

Maanantai 08.03.2021

Tavoitteeni on rakentaa viime viikolla kesken jäänyt CompanyAvatarEditable-komponentti valmiiksi. Nykytilassa käyttäjän ladattua uuden logotyypin komponentin kautta ilmestyy uusi logo näkyviin sekä esikatseluruutuun, että dialogi-ikkunan taustalla näkyvään klikattavaan avatariin. Haluan komponentin toimivan niin, että uusi logotyyppi tulee näkyviin esikatseluun, mutta vanha logotyyppi, joka näkyy dialogi-ikkunan takana, vaihtuu uudeksi

vasta käyttäjän painaessa OK-painiketta. Lisäksi kollegani pyysi minua lisäämään komponenttiin latauksen edistymistä kuvaavan palkin, eli progress barin, tai pyörivän spinnerin. Käytännössä tämä tarkoittaa, että tehtäväni on rakentaa komponenttiin molemmat edistymistä kuvaavat mallit. Näistä malleista valitsemme yhdessä suunnittelijan kanssa jommankumman lopulliseen versioon.

Päivän tavoitteet täyttyivät osittain, sillä sain kuvan näkymään esikatselussa ja vaihtumaan vasta käyttäjän painaessa OK-painiketta dialogin oikeasta alakulmasta. Edistymistä kuvaavaa palkkia tai spinneriä en ehtinyt saada toimimaan. Tiimin tukeen tuli päivän aikana vain yksi tukipyyntö, johon löysimme ratkaisun yhdessä vanhemman kollegani kanssa Tandem-ohjelman välityksellä.

Tiistai 09.03.2021

Tiistain tavoitteenani on refaktoroida eilen ja viime viikolla kirjoittamani koodi CompanyAvatarEditable-komponenttiin, jotta koodista tulisi luettavampaa ja se olisi ymmärrettävämpää muillekin kehittäjille. Refaktorointi tarkoittaa, että koodin alkuperäinen toiminnallisuus säilyy, mutta sen sisäinen rakenne muuttuu (Burchard 2017, 1.). Lisätavoitteeni on jatkaa edistymistä kuvaavan palkin sekä spinnerin toteuttamista ja lisätä nämä toiminnallisuudet komponenttiin. Päivän päätteeksi toivon voivani laittaa koodin katselmoitavaksi, jotta voin siirtyä komponentin parissa työskentelystä seuraaviin tehtäviin.

Valitettavasti spinnerin tai edistymistä kuvaavan palkin lisääminen CompanyAvatarEditable-komponenttiin ei onnistunut haluamallani tavalla, sillä ominaisuuden testaaminen oli hankalaa. Logotyypit latautuivat esikatseluun niin nopeasti, ettei spinneriä ollut järkevää lisätä. Sain päivän tavoitteeksi asettamani koodin refaktoroinnin kuitenkin tehtyä ja nyt koodin pitäisi olla kaikille kehittäjille ymmärrettävää ja helposti luettavaa. Lähetin refaktoroidun koodin katselmoitavaksi ja odotan katselmoinnin tuloksia.

Keskiviikko 10.03.2021

Päivän tavoitteeni on aloittaa CompanyAvatarEditable-komponenttiin liittymättömien, kesken jääneiden muutosten korjaaminen komponenttikirjaston version 1.0 julkaisua varten. Lisäksi pyrin hoitamaan CompanyAvatarEditable-komponentin pull requestiin tulleet korjausehdotukset, jotta kaikki komponentin parannukset voitaisiin lisätä master-haaraan.

Päivän tavoitteeni täyttyivät tänään osittain, sillä tein pull requestiin tulleet korjausehdotukset, mutta en ehtinyt tehdä muutoksia muihin komponentteihin. Työpäivän aikana eniten

ongelmia aiheutti CompanyAvatarEditable-komponentista aukeava dialogi-ikkuna. Komponentti käyttää taustalla erillistä Dialog-komponenttia. Ongelmat johtuivat dialogi-ikkunan päällysteestä eli overlay:sta, jonka tyylittelyä en onnistunut muuttamaan. Käyttäjän klikatessa CompanyAvatarEditable-komponenttia aukeaa sivun päälle dialogi-ikkuna sekä päällyste. Päällyste aiheuttaa selaimessa olevan vierityspalkin (scrollbar) katoamisen ja päällysteen takana oleva nettisivu ikään kuin hyppää vierityspalkin verran sivulle. Bugi ei haittaa varsinaista toiminnallisuutta, mutta on käyttäjälle ärsyttävä, joten se on ehdottomasti korjattava.

Lisäksi hoidin päivän aikana seniorikollegani kanssa muutamaa tukipyyntöä, jotka olivat tulleet Peacockin tukikanavaan hoidettavaksi. Opin tänään lisää CSS-tyylittelystä ja päällysteiden käytöstä, vaikka en saanutkaan toiminnallisuutta vielä kuntoon.

Torstai 11.03.2021

Tiimin tuoteomistajalta oli tullut eilen ilmoitus Peacockin Google Chat-kanavalle. Hän ilmoitti, että aikoo tehdä julkaisun järjestelmän Web 2.0 -ensimmäisestä versiosta, joka tulee käyttöön viidelle prosentille asiakasyrityksistä. Kyse on käytännössä sivujen uudesta käyttöliittymästä. Sivujen nykyinen käyttöliittymä on yli kymmenen vuotta vanha ja uutta järjestelmää on rakennettu taustalla jo vuosia. Prosessissa vanha monoliittinen rakenne puretaan pienemmiksi mikropalveluiksi. Päivän tavoitteenani on tarkastaa, että kaikki Web 2.0:n sisältämät komponentit ovat varmasti toimintakunnossa, sillä vaikka uusi sivu tulee käyttöön vasta viidelle prosentille käyttäjistä, tarkoittaa se silti yli 2500 käyttäjätunnusta. Lisäksi tarkoitukseni on korjata dialogi-ikkunan päällysteen aiheuttama taustan hyppääminen CompanyAvatarEditable-komponentissa.

Päivän tavoitteeni täyttyi ja kaikki komponentit toimivat testeissä kuten niiden kuuluukin. Päivällä pidettiin yrityksen laajuinen esitys Web 2.0 -version julkistuksesta, jota en valitettavasti nähnyt. Muistin kokouksen alkavan klo 13:30, mutta sille oli varattu aika 13:00-13:30 ja olin tuolloin vielä lounaalla. Onneksi esitys tallennettiin ja ehdin katsoa sen myöhemminkin. Toisaalta en saavuttanut ihan kaikkia tavoitteita, sillä en saanut CompanyAvatarEditable-komponentin dialogi-ikkunan päällystettä toimimaan vieläkään haluamallani tavalla.

Perjantai 12.03.2021

Päivän tavoitteeni on korjata aikaisemmin löytämiämme bugeja muissa kuin CompanyAvatarEditable -komponenteissa. Olemme kirjanneet löydettyjä bugeja sekä

korjattavia asioita web-komponenttikirjastosta yhteiseen dokumenttiin tammikuusta lähtien. Dokumenttia on säilytetty Google Docs-palvelussa.

Suunnitelmistani huolimatta työskentelin taas koko päivän CompanyAvatarEditable-komponentin parissa. En tiedä mitä on tapahtunut, mutta web-komponenttikirjaston komponentteihin tekemäni muutokset eivät päivity millään Pageron dev-ympäristöön. Ongelman ikävyyttä lisää, että projektin rakentamiseen menee noin 20 minuuttia kerrallaan, joten muutosten testaaminen on todella hidasta. Työpäivän lopussa olin kollegani kanssa Tandem-parikoodaussessiossa lähes kolme tuntia, emmekä yhdessäkään keksineet, mistä vika johtuu.

Viikkoanalyysi

Kulunut viikko meni ohi tosi nopeasti. Tuntuu siltä, että en saanut mitään aikaiseksi koko viikon aikana. Vaikka en päässyt edistämään niin kutsuttuja normaaleja työtehtäviäni juurikaan, opin silti uutta viikon tapahtumista. Opin lukuisia uusia ratkaisuja ongelmiin, joita tiimin tukeen toisinaan tulee. Esimerkkinä voi mainita tapauksen, jossa järjestelmässä olevalle yritykselle haluttiin uudet käyttöoikeudet API-rajapintaan. Tapaus oli erityisen haastava, sillä yrityksen sisäinen dokumentaatio aiheesta oli hieman epäselvää. Dokumentaatiossa ja käytännössä asiat oli nimetty eri nimillä ja useissa tapauksissa oli arvattava, mahtaisiko muun muassa asiakkaan ”authorization_code” tarkoittaa samaa kuin ”secret”? Hoidimme asian lopulta puoliksi arvailemalla ja toimitimme käyttöoikeudet asiakkaan tarvitsemaan API-rajapintaan.

Viikon ajalta mieleeni jäi myös Google Analyticsin (GA) tutkiminen. Sain oikeudet yrityksen Google Analytics-rajapintaan ja pohdimme kollegani kanssa, miten saisimme samalle GA-tilille lisättyä verkkotunnukselle (domain) alidomainin. Tarkoituksena oli pystyä seuraamaan käyttäjän verkkokäyttäytymistä hänen siirtyessään vanhasta käyttöliittymästä uuteen Web 2.0-käyttöliittymään, jotka sijaitsevat eri osoitteissa. Google Analyticsin hyödyntäminen on varmasti myös hyvä taito hallita tulevaisuutta ajatellen ja aiempi kokemukseni sen käytöstä rajoittuu henkilökohtaisiin projekteihini.

Epäonnistuneet yritykset lisätä spinneriä CompanyAvatarEditable-komponenttiin eivät juurikaan harmita, sillä Shakirin (28.2.2017.) mukaan spinnereiden käyttö kannattaisi lopettaa kokonaan. Hänen mukaansa, vaikka lataus-spinnerit ovat käytetyimpien edistymisen indikaattoreiden joukossa, on niissä omat haittapuolensa, jotka yleensä jätetään huomioimatta. Spinnereiden ensimmäinen ongelma on, etteivät ne näytä pyörimisen lisäksi minkäänlaista merkkiä etenemisestä. Pyörivät spinnerit aiheuttavat epävarmuutta, kun

käyttäjä ei tiedä kuinka kauan sitä joutuu katsomaan. Kolmas ongelma spinnereissä on, että latausaika tuntuu mahdollisesti käytännössä pidemmältä mitä se oikeasti on, sillä käyttäjä näkee spinnereissä ainoastaan odotellessa hukkaan menevän ajan. Suosittu ehdotus spinnereiden korvaajaksi on sivun luurankonäkymä, joka muistuttaa lopullista latauksen tulosta ja sen päälle lopullinen latauksen tuotos rakennetaan. (Shakir 28.2.2017.) Muun muassa Facebook käyttää luurankonäkymää ladatessaan sivua.

Mikäli etenemispalkin lisääminen CompanyAvatarEditable-komponenttiin ei onnistu, yksi mahdollisuus olisi näyttää kuvasta aluksi heikkolaatuista pikselöityä tai sumennettua versiota ja korvata se valmiilla kuvalla heti latauksen valmistuttua. Googlen käyttämä metodi on täyttää kuvalle varattu tila kuvassa vallitsevalla päävärillä, ja korvata värikäs ruutu lopullisella kuvalla heti latauksen valmistuessa.

Mainitsin viikon aikana, että CompanyAvatarEditable-komponentti käyttää pinnan alla toista komponenttia, joka on nimeltään Dialog. Aina kun on mahdollista käyttää olemassa olevaa komponenttia toisen komponentin sisällä, se kannattaa tehdä, jotta koodiin tulisi mahdollisimman vähän toistoa. Koodin toistaminen on hyvin suunnitellun järjestelmän päävihollinen. Se tuottaa ylimääräistä työtä, lisää riskejä sekä tarpeetonta kompleksisuutta. On olemassa monenlaista koodin toistamista, kuten muun muassa tismalleen samanlaisia koodirivejä. Toinen esimerkki on jonkin toteutuksen toistaminen, vaikka useissa tapauksissa samankaltaiset koodirivit voitaisiin refaktoroida esimerkiksi yhteen funktioon ja seurauksena koodin toiminnallisuus olisi samanlainen, kuin aikaisemmin, mutta koodi olisi helpommin luettavissa. (Martin 2009, 173.)

```
3   int size() {...}
4   boolean isEmpty() {...}
5
6   boolean isEmpty() {
7     return 0 == size();
8   }
```

Kuva 2. Koodin toistamisen minimointi (mukaillen Martin 2009, 173.)

Kuvassa 2 kuvataan koodin toistamisen minimointia. Ylempänä esitetyt ”size” ja ”isEmpty” voisivat olla esimerkiksi kaksi erillistä metodia. Tässä tapauksessa isEmpty()-metodi tarkkailisi boolean-arvoa, kun taas size-metodi tarkkailisi laskuria. Koodin toistamisen voi kuitenkin eliminoida kytkemällä ”isEmpty”-boolean arvon size-muuttujaan kuten kuvan 2 alariveillä on kuvattu. (Martin 2009, 173.)

3.5 Seurantaviikko 5

Viikon tavoitteet

Viidennellä raportointiviikolla päätavoitteenani on edelleenkin saada CompanyAvatarEditable-komponentti valmiiksi. Sama tavoite oli myös edellisellä viikolla, mutta se ei täyttynyt silloinkaan. Peacockin kiertävän tukiaikataulun mukaan minulla ei ole tukitehtäviä seuraavaan kuuteen viikkoon, joten luultavasti pystyn tällä viikolla keskittymään komponentteihin tarkemmin kuin edellisellä viikolla.

Maanantai 15.03.2021

Aamupäivän tavoitteeni on saada käynnistää kaikki tarvitsemani Docker-kontit ja Pageron dev-kehitysympäristö toimintakuntoon CompanyAvatarEditable-komponentin kehittämistä sekä bugien korjausta varten. Mikäli ylimääräistä aikaa jää, katson viime viikolla nauhoitetun videon Web 2.0-julkaisusta, jotta pysyn paremmin kartalla tulevista muutoksista ja mahdollisista tarpeista. Iltapäivällä stand up-kokouksen ja lounaan jälkeen korjaan Google Docs-dokumenttiin kirjattuja bugeja myös muista web-komponenttikirjaston komponenteista.

Saavutin päivän tavoitteeni lähes kokonaan, sillä sain Docker-kontit ja kehitysympäristön toimimaan moitteetta ennen lounasta, mutta Web 2.0-julkaisuvideota en ehtinyt katsoa. CompanyAvatarEditable-komponentin pull requestiin oli tullut muutospyyntöjä, joita en ehtinyt tehdä. Lisäksi olin kokonaan unohtanut, että tänään oli joka toinen maanantai pidettävä Backlog refinement, josta onneksi mainittiin myös aamupäivän stand up-kokouksessa. Tuoteomistajan toiveesta CompanyAvatarEditable-komponentin lisäksi toimintakuntoon halutaan vielä UserAvatarEditable-komponentti web-komponenttikirjaston version 1.0 julkaisuun. Lopullisessa UserAvatarEditable-komponentin versiossa loppukäyttäjällä on oltava mahdollisuus rajata valitsemansa käyttäjäkuva sopivaksi, mutta ensimmäisestä versiosta rajaus voidaan tuoteomistajan mukaan jättää pois. En voi vielä alkaa kuitenkaan työstää komponenttia, sillä User API-rajapintaa ei ole vielä rakennettu. Rajapintoja tekevä kollegani ei osallistunut Backlog refinement-kokoukseen, joten hän ei tiedä komponentista tai puuttuvasta API-rajapinnasta vielä mitään. Pitkän Backlog-kokouksen ja uusien muutospyyntöjen vuoksi kaikki muut web-komponenttikirjaston komponentit kuin CompanyAvatarEditable jäivät tänään huomiotta.

Tiistai 16.03.2021

Tiistain tavoitteena on korjata viimeinen bugi CompanyAvatarEditable-komponentista ja saada haaran pull request hyväksyttyä, jotta voin yhdistää muutokset master-haaraan. Mikäli kollegani ehtii tehdä päivän aikana UserAvatarEditable-komponenttia varten User API-rajapinnan, alan työskennellä sen parissa. Urakkaa onneksi helpottaa tieto, että voin kopioida suurin osan CompanyAvatarEditable-komponentista, joten uutta komponenttia ei tarvitse rakentaa aivan tyhjästä. Jos edellä mainittujen töiden lisäksi aikaa jää, alan korjata muista komponenteista aikaisemmin löydettyjä bugeja.

Tavoitteeni täyttyi tavallaan, sillä korjasin päivän aikana viimeisenkin tiedossa olevan bugin CompanyAvatarEditable-komponentista, vaikkakin tämän jälkeen siihen toivottiin lisää tarpeellisia ominaisuuksia. Komponentti haki vielä aamulla yrityksen logotyypin Company API-rajapinnasta käyttäen kehittäjän lisäämää companyId-attribuutin arvoa. Vanhempi kollegani halusi komponenttiin lisättävän vielä imageUrl-attribuutin, jonka arvo syrjäyttäisi companyId-tunnusta käyttäen rajapinnasta haetun logon. Lisäsin komponenttiin toivotun ominaisuuden ratkottuani ensin pari ongelmaa, jotka ilmenivät kehitystyön aikana. Muutosten jälkeen komponentti tarkastaa ensin, onko imageUrl-attribuutilla olemassa arvoa ja jos arvo on null tai undefined, yrittää komponentti hakea yrityksen logotyypin Company API-rajapinnasta companyId-attribuutin arvoa hyödyntäen. Mikäli companyId-arvokin on null tai undefined, komponentti näyttää avatarissa ainoastaan ennalta määritellyn ikonin, mikä kuvastaa tehdasta. Lisäksi korjasin muutaman aiemmin löydetyn pienemmän bugin muista komponenteista.

Keskiviikko 17.03.2021

Päivän tavoitteena on viimeistellä CompanyAvatarEditable-komponentin uuden imageUrl-attribuutin lisäys. Tällä hetkellä vanha logo ei vaihdu automaattisesti uuteen käyttäjän ladatessa uuden logotyypin yritykselleen komponenttia käyttäen, mikäli imageUrl-attribuutilla on arvo. Jos rajapinnoista vastaava kollegani saa päivän aikana User API-rajapinnan rakennettua, alan kehittää UserAvatarEditable-komponenttia. Muussa tapauksessa alan korjata suurempaa bugia FileUploadAdvanced-komponentista, sillä tällä hetkellä komponentti ei osaa tarkastaa, onko käyttäjä kirjautuneena sisään. Mikäli käyttäjä ei ole kirjautuneena sisään ja hän yrittää ladata tiedostoa File API-rajapintaan, kutsu hylätään, sillä rajapinta vaatii kutsuun voimassa olevan käyttöoikeustunnuksen.

En saavuttanut työpäivän aikana tavoitettani, sillä en keksinyt toimivaa tapaa ratkaista ongelma imageUrl-attribuutin arvon ja companyId-attribuutin arvon kanssa. Lisäksi päivän

aikana vastaan tuli muita yllättäviä tilanteita. Esimerkiksi muiden tiimien kehittäjillä, jotka ovat ottaneet kirjaston testikäyttöön, ilmeni ongelmia web-komponenttikirjaston tiettyjen komponenttien käytössä. Ratkoin testikäyttäjien ongelmia ja julkaisin komponenttikirjastosta version 0.9.0, jossa löydetyt bugit oli korjattu.

UserAvatarEditable-komponentti saa vielä odottaa, sillä User API-rajapinta ei ollut vielä valmis. Olisin halunnut pitää tänään Tandem-kokouksen vanhemman kollegani kanssa imageUrl-attribuutin ongelmanratkaisua koskien, mutta hän oli varattu koko päivän, kun tiimin muillakin kehittäjillä oli hänelle ongelmia ratkottavaksi. Sain päivän aikana kuitenkin kokemusta jälleen bugien korjaamisesta ja vuorovaikutuksesta web-komponenttikirjaston käyttäjien, eli muiden kehittäjien kanssa.

Torstai 18.03.2021

Päivän tavoite on vihdoinkin viimeistellä useita viikkoja tekeillä ollut CompanyAvatarEditable-komponentti. Yritän saada seniorikollegani Tandem-parikoodaussessioon ennen stand up-kokousta ongelmanratkaisua varten. Mikäli saamme CompanyAvatarEditable-komponentin toimintakuntoon, alan työstää joko UserAvatarEditable-komponenttia tai korjata bugeja FileUploadAdvanced-komponentista.

Tavoite ei täyttynyt tänäänkään. Pidin kyllä seniorikollegani kanssa Tandemissa parikoodaussession, mutta sen aikana huomasimme, ettei CompanyAvatarEditable-komponentti tule toimimaan kunnolla CompanyAvatar-komponentin kanssa. Molemmat komponentit vaativat suuria muutoksia, jotta ne toimisivat yhdessä. Tällä hetkellä CompanyAvatar-komponentin logotyyppi ei lataudu, mikäli käyttäjä ei ole kirjautuneena sisään, vaikka ainoastaan CompanyAvatarEditable-komponentin pitäisi vaatia sisäänkirjautumista. Käytännössä tämä tarkoittaa sitä, että molempien komponenttien koodista täytyy kirjoittaa noin 50 % uudestaan. Onneksi voin kuitenkin käyttää CompanyAvatarEditable-komponentin koodia uudelleen CompanyAvatar-komponentin rakentamisessa.

Perjantai 19.03.2021

Päivän tavoitteeni on ainakin aloittaa CompanyAvatar- ja CompanyAvatarEditable-komponenttien koodin kirjoittaminen uudelleen. Toivon mukaan kaikki menee sujuvasti ja molemmat komponentit ovat valmiita työpäivän päätteeksi. Kokemukseni mukaan matkaan tulee kuitenkin mutkia, enkä usko, että kaikki valmistuu tämän päivän aikana. Olen oppinut kehitystyön myötä, että yksinkertaisempiakaan asioita ei kannata pitää itsestään

selvyyksinä, sillä koodista voi paljastua yllättäviä ongelmia muun muassa yhteensopivuuden kanssa.

Tänään saavutin osittain tavoitteeni. Ehdin juuri alkaa kirjoittaa CompanyAvatar-komponenttien koodia uudelleen ja testasin paria muutosta, kunnes uusi kollegani otti yhteyttä ja pyysi apua tiedostojen lataus-, eli file upload-ominaisuuden lisäämiseen Pageron Businessprocess-Web -ympäristöön. Seniorikollegani oli kertonut hänelle, että olen juuri rakentanut FileUpload-komponentin ja käyttänyt ominaisuutta myös uudessa CompanyAvatarEditable-komponentissa, joten tiedostojen kääntäminen eri muotoihin on minulla tuoreessa muistissa.

Valitettavasti minulla ei ollut Businessprocess-Webin vaatimien Docker-konttien kehitysympäristöä asennettuna omalle koneelleni, joten suurin osa päivästä kului toimintaympäristön kasaamiseen ja konfigurointiin. Sain lopulta toimintaympäristön käyntiin ja pääsin tutkimaan järjestelmän businessprocess osa-aluetta. Koodi on kirjoitettu AngularJS -sovelluskehystä käyttäen, josta minulla ei ole käytännössä lainkaan oikeaa kokemusta. Opettelin siis loppupäivän lukemaan AngularJS-frameworkin syntaksia- Lisäksi pohdimme uuden kollegani kanssa, kuinka saisimme tiedostot muutettua haluamaamme muotoon, mikäli loppukäyttäjä yrittää lisätä esimerkiksi laskuunsa liitteitä. Vaikka en edennyt käytännössä yhtään Avatar-komponenttien kanssa, päivä ei mennyt kuitenkaan täysin hukkaan. Asensin koneelleni kuitenkin Businessprocess-Web -kehitysympäristön, jonka olisin joutunut luultavasti asentamaan koneelleni joka tapauksessa jossain vaiheessa.

Viikkoanalyysi

Kuluneen viikon päätavoitteena oli saada CompanyAvatarEditable-komponentti vihdoin valmiiksi, mutta sen sijaan joudunkin ensi viikolla kirjoittamaan komponentin lähes kokonaan uudelleen. Mielestäni ilmiössä tulee hyvin esiin ketterän ohjelmistokehityksen haittapuoli, sillä käytännössä kahden viikon työ komponentin parissa menee lähes hukkaan, kun asiaa ei osattu huomioida suunnittelussa alusta alkaen kunnolla. Havaintoani mukailee myös Fridmanin artikkeli ketterästä kehityksestä ja sen ongelmista. Artikkelissaan Fridman listaa ketterän kehityksen haittapuolia ja listan kärjessä on ennalta arvaamattomuus (Fridman 2016.).

Viikon aikana jouduin tutkimaan paljon erinäisiä CompanyAvatarEditable-komponenttiin liittyviä ongelmia. Tutkin muun muassa miten saisin komponentin tunnistamaan loppukäyttäjän selaimessa olevan vierityspalkin leveyden, sillä leveys vaihtelee selaimesta riippuen. Kun käyttäjä klikkaa CompanyAvatarEditable-komponenttia, aukeaa sivun päälle dialogi-

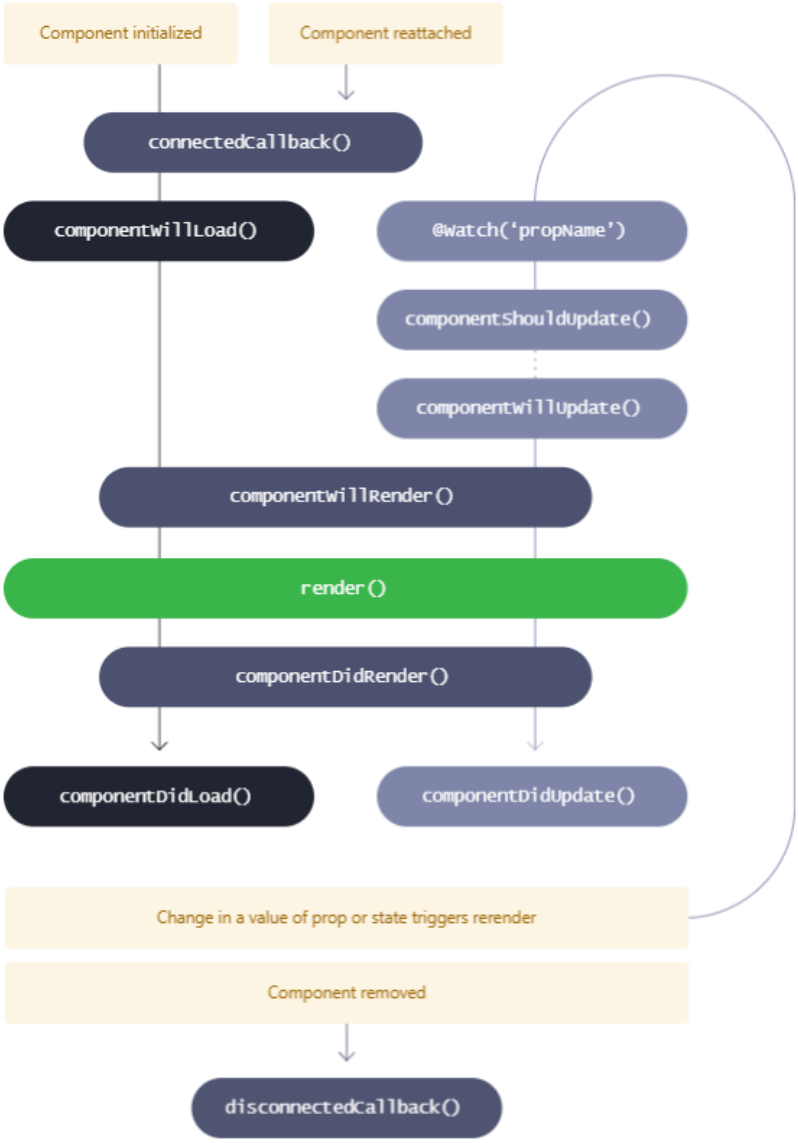
ikkuna, joka estää käyttäjää olemaan vuorovaikutuksessa dialogin takana olevan sivun kanssa. Komponentin ongelma oli kuitenkin, että dialogin auetessa takana oleva sivu hyppäsi sivusuunnassa vierityspalkin leveyden verran ja se oli käyttäjälle todella häiritsevää. Ongelma ratkesi lopulta pienellä koodin pätkällä, jonka löysin sattumalta Googlen kautta Bytutorialin (2016.) blogikirjoituksesta. Paras tapa tarkastaa vierityspalkin olemassaolo on tarkastella käyttäjän internetselaimen ikkunan korkeutta, sillä jos ikkunan sisäinen korkeus on pienempi kuin sivun HTML-bodyn korkeus, voidaan olettaa, että sivulla on vierityspalkki. Viiden vuoden takaa blogikirjoituksesta löytämäni tilanteen pelastanut yksinkertainen koodin pätkä on havainnollistettu kuvassa 3.

```
3   if (window.innerWidth > document.body.clientWidth) {...}
```

Kuva 3. Tarkastuslauseke sivun vierityspalkkia varten (mukaillen Bytutorial 2016.)

Yksi suurimmista ongelmista kehitystyössä viime viikkoina on ollut myös järjestelmän hitaus etenkin silloin, kun olen jonkun kanssa Tandem-parikoodaussessiossa. Tandem-ohjelman avulla ruudun jakaminen vie lähes yhtä paljon resursseja prosessorilta kuin 35 Docker-kontin yhtäaikainen ajaminen. Kollegani epäili hitauden johtuvan siitä, että Tandem joutuu pakkaamaan videokuvaa jatkuvasti reaaliajassa yhteyden ollessa päällä. Ratkaisimme asian viikon aikana yleensä siten, että suljimme Tandemin siksi aikaa, kun projekteja rakennettiin muutosten tekemisen jälkeen, ja otimme uuden yhteyden aina kun uusi buildi valmistui. Ratkaisu on toimiva, mutta raskas molemmille osapuolille, sillä mitään uutta ei oikein voi aloittaa, kun hetken päästä toinen jo soittaa Tandemilla, että buildi on valmis. Yrityksen näkökulmasta lyhyellä tähtäimellä tämä ei ehkä ole kovin tehokasta, sillä yhteen tehtävään käytetään kahta resurssia, eli ohjelmoijaa. Toisaalta pitkällä tähtäimellä yritys hyötyy siitä, että lopulta yrityksellä on yhden osaajan sijaan kaksi osaajaa.

Viikon toinen mieleen jäänyt ongelma johtui StencilJS:n komponenttien elinkaarimetodeista. CompanyAvatarEditable-komponentti, kuten kaikki muutkin komponenttikirjaston komponentit, kirjoitetaan StencilJS-sovelluskehyksen syntaksilla. Komponentin elinkaari on kuvattu alla kuviossa 8.



Kuvio 8. StencilJS-komponentin elinkaari (Stencil 2021)

Komponentin toimintoja ja funktioita täytyy kutsua elinkaaren oikeassa vaiheessa, jotta niiden järjestys pysyisi oikeana. Viikon aikana erään bugin korjaamisyrityksen tuloksena CompanyAvatarEditable-komponentin internalCompanyId-tilaa, eli statea manipuloitiin vuorotellen componentWillRender()- sekä componentDidRender() -vaiheissa. Staten manipulointi aiheuttaa StencilJS-komponenteissa aina uudelleen renderöinnin, jonka seurauksena komponentti jäi jumiin ikuiseen looppiin. StencilJS:n dokumentaatiossa varoitetaan juuri tapahtuneesta ikuisesta kierteestä ja ratkaisuksi esitetään tarkastamaan tila componentDidUpdate()-metodissa (Stencil 2021.).

Komponenttien elinkaaren suunnitteleminen on erityisen tärkeää ennen varsinaista rakentamista. Toisaalta kokemukseni mukaan ketterässä kehityksessä ei kannata suunnitella liian tarkkaan etukäteen, sillä usein komponenttiin joudutaan tekemään aiempia suunnitelmia rikkovia muutoksia. Elinkaaren eri metodeista on kuitenkin hyvä olla tietoinen, jotta

tunnistaa heti muutostarpeiden ilmetessä, miten koodi kannattaa strukturoida muutosten jälkeen uudelleen. Elinkaaren selkeä noudattaminen on tärkeää myös käyttäjien, eli yrityksen muiden web-kehittäjien näkökulmasta. Heidän täytyy voida päätellä johdonmukaisesti, kuinka komponentin elinkaari käytettäessä toimii.

3.6 Seurantaviikko 6

Viikon tavoitteet

Kuudennen seurantaviikon tavoitteisiin kuuluu edellisillä viikoilla työstettyjen Avatar-komponenttien koodin uudelleenkirjoitus. Tiimin scrum master palaa tänään kahden viikon lomalta, joten hänen täytyy kysyä hänen mielipidettään CompanyAvatarEditable-, sekä CompanyAvatar-komponenttien rakenteesta. Viikon teemaksi otan asioiden tarkastelun erilaisista näkökulmista.

Maanantai 22.03.2021

Päivän tavoitteeksi asetan CompanyAvatarEditable- ja CompanyAvatar-komponenttien koodin rakenteen uudistamisen. Lisäksi koetan tarvittaessa auttaa uutta tiimiläistä tiedostojen latauksen lisäämisessä Businessprocess-Webiin.

Maanantaille asettamani tavoitteet onnistuivat erinomaisen hyvin, sillä sain molemmat komponentit toimimaan. Edellisellä viikolla kokeilin komponenttien sisällä hieman kyseenalaisia viritelmiä, joiden avulla olisin voinut laittaa komponentit renderöimään itsensä haluamiini aikoihin. Vanhempi kollegani kuitenkin ehdotti, että lisäisin komponenttikirjastoon vielä yhden uuden Avatar-komponentin kirjaston sisäiseen käyttöön. Tämän uuden Avatar-komponentin tehtävä on ainoastaan renderöidä ruudulle komponentille syötetty sisältö. CompanyAvatar-, sekä CompanyAvatarEditable-komponentit käyttävät Avatar-komponenttia sisällön näyttämiseen, mutta itse API-rajapintakutsut, koon määritykset ja vastaavat asiat tapahtuvat CompanyAvatar-, sekä CompanyAvatarEditable-komponenttien sisällä.

Avasin kaikista uusista muutoksista pull requestin, mutta en oleta muutospyyntöjen menevän läpi katselmoijalle aivan sellaisenaan. Koodikatselmoinnin yksi etu onkin, että koodiin saadaan erilaisia näkökulmia ja nuoremmat kehittäjät oppivat jatkuvasti uutta katselmointien kautta. Avatareiden purkaminen osiin on hyvä asia jokaisen sidosryhmän näkökulmasta ja se lisää kompleksisuutta ainoastaan minulle itselleni komponenttia kehittäessä. Komponenttien käyttäjille eli muille kehittäjille sekä loppukäyttäjille (Pageron asiakkaille)

muutokset eivät näy mitenkään ulospäin. Tiimin uusin jäsen oli päässyt myös itsekseen eteenpäin ongelmansa kanssa, johon hän pyysi apua perjantaina, joten minun ei tarvinnut auttaa häntä tiedostojen lataamisen lisäämisessä.

Tiistai 23.03.2021

Huomasin aamulla työpuhelimeen tulleesta sähköposti-ilmoituksesta, että pull requestiin on tullut kymmenen kommenttia ja muutospyyntöä. Päivän tavoitteeni on käydä kommentit läpi ja korjata komponentit pyyntöjen mukaisiksi. Mikäli aikaa jää ja User API-rajapinta alkaa olla valmis, voin alkaa jo työstää UserAvatar- ja UserAvatarEditable-komponentteja. Muussa tapauksessa alan tehdä muutoksia FileUploadAdvanced-komponenttiin.

FileUploadAdvanced-komponenttiin asti en tänään ehtinyt, mutta sain kyllä pääosin kaikki Avatar-komponentteihin tulleet muutospyynnöt tehtyä. Aikaisemmin jokaisessa komponentissa, jossa API-rajapinnan kutsuja käytettiin, oli omat kutsunsa. Kopioin komponenttien käyttämän GET-kutsun yhteen tiedostoon, jota kaikki neljä Avatar-komponenttia pystyvät hyödyntämään. Näin samaa koodia ei toisteta monessa paikassa, vaan kaikki kutsuja tarvitsevat komponentit pystyvät hyödyntämään kerran kirjoitettua koodia.

Suurin osa päivästä kului muutosten tekemiseen, sillä koodissa oli huomioitava, että kutsut voivat erota toisistaan käyttötapauksesta riippuen. Stand up-kokouksessa tiimin scrum master otti puheeksi, etteivät nykyisessä muodossaan Base64-enkoodatut logotyypit tallennu selaimen välimuistiin, vaan ne täytyy ladata palvelimelta joka kerta erikseen, kun avatar-komponentteja käyttävät sivut latautuvat. Tämä seikka ei ole tullut aikaisemmin mieleeni, joten asiaa täytyy tutkia lähipäivinä.

Tänään oli normaalin stand up-kokouksen lisäksi kuukausittainen tiimikokous, jossa itsensä ja työnsä esittelivät tiimit Albatross, Condor sekä Eagle. Albatrossin esityksestä jäi mieleen mielenkiintoisia vaihtoehtoja Webpackin käytölle, joiden pitäisi nopeuttaa Front-end projektien rakennusaikoja merkittävästi. Webpack on moduulien niputtaja, eli bundler JavaScript-pohjaisille sovelluksille. Webpackin prosessoidessa sovellusta, se rakentaa projektista sisäisen riippuvuusgraafin, joka kartoittaa kaikki tarvitut moduulit ja paketoi projektin kompaktiksi paketiksi (Webpack 2021.).

Keskiviikko 24.03.2021

Päivän tavoitteeni on viimeistellä CompanyAvatar-komponenttien normaalitoiminnot sekä tutkia mahdollisuutta lisätä base64-enkoodattu kuva käyttäjän selaimen välimuistiin, ettei

sitä tarvitsisi ladata palvelimelta joka kerta sivun latautuessa. En usko, että ehdin päivän aikana lisätä toteutusta kuvan välimuistiin lisäämisestä, mutta se on silti ylimääräinen tavoitteeni.

Saavutin päivälle asetetun päätavoitteen ja sain CompanyAvatar, CompanyAvatarEditable sekä UserAvatar-komponenttien toiminnot valmiiksi. Lisäksi korjasin kaikki pull requestiin tulleet muutospyynnöt. Base64-enkoodattujen kuvien käsittely selaimen välimuistissa oli hieman monimutkaisempi ongelma ja pidimmekin siitä erikseen kokouksen tiimin scrum masterin ja seniorikollegan kanssa. Kokoukseen oli kutsuttu myös API-rajapintoja tekevä kollega tiimistä, mutta valitettavasti hän sairastui ennen kokousta. Pohdimme ongelmaan erilaisia ratkaisuja, mutta ongelma oli sen verran haastava, ettei kukaan osannut sillä hetkellä ehdottaa toimivaa ratkaisua. Ongelma API-rajapinnan headerin välimuistin käytössä on, että loppukäyttäjän ladatessa yritykselle uuden logotyypin, se ei välttämättä päivity heti näkyviin, sillä komponentti näyttäisi käyttäjälle yhä välimuistissa olevan vanhan logotyypin uuden logotyypin sijaan. Kokouksen lopuksi päätimme, että emme tee asialle vielä mitään muutoksia, mutta koitamme tutkia vaihtoehtoisia ratkaisuja.

Nykytilassa CompanyAvatar-komponentti lähettää palvelimelle kutsun joka kerta kun loppukäyttäjä avaa sivun, jossa on yhden tai useamman yrityksen logotyyppi. Yhden yrityksen kohdalla tämä ei ole ongelma, mutta jos loppukäyttäjät avaavat sivuja, joissa voi olla näkyvillä esimerkiksi 20 yrityksen logotyypit, palvelimelle tulee hirveästi ylimääräistä liikennettä. Sattumalta kokouksen aikana tiimin tuoteomistaja kyseli myös CompanyAvatar- ja CompanyAvatarEditable-komponenteista ja hän vaikutti innokkaalta päästä kokeilemaan sekä julkaisemaan komponentit. Vilkaisin päivän lopuksi käännösten lisäämistä komponenttikirjastoon, sillä ainakin kolmeen komponenttiin täytyy olla käännökset saatavilla, kun web-komponenttikirjaston versio 1.0 julkaistaan. Komponentit pyritään pääsääntöisesti rakentamaan niin, ettei erillisiä komponenttikohtaisia käännöksiä tarvita, mutta tiettyihin komponentteihin on silti lisättävä tekstiä, joka täytyy kääntää.

Torstai 25.03.2021

Työsähköpostiini oli tullut viesti, joka sisälsi uuden Pageron sertifikaatin, joka tulee asentaa käyttämiini internetselaimiin. Vanha sertifikaatti oli vanhentunut, joten se ei enää toimi. Sertifikaatin tarkoituksena on estää sivuston ylläpitotoimintojen väärinkäyttö. Tavoitteeni on asentaa uusi sertifikaatti heti aamulla saadessani tekstiviestinä sertifikaatin vaatiman salasanan. Lisäksi toinen tavoitteeni on tutkia, kuinka web-komponenttikirjaston käännökset ovat järkevintä toteuttaa. Pyrin myös lisäämään web-komponenttikirjaston osalta uuden projektin käyttämäämme käännöspalveluun Weblateen. Weblate on selaimella

käytettävä käännöstyökalu, joka yhdistetään suoraan projektin Github-repositorioon ja se hoitaa automaattisesti käännösten lokalisoinnin, laatutestit sekä versionhallinnan.

Saavutin torstain tavoitteet, sillä asensin sertifikaatin onnistuneesti jo aamulla ja loin Weblateen projektin web-komponenttikirjastolle käännöksiä varten. Sain myös luvan yhdistää CompanyAvatarEditable-komponentin sekä muut avatar-komponenttien muutokset master-haaraan, joten suoritin liittämisen. Lisäksi konfiguroin komponenttikirjaston yhteensopivaksi Weblaten kanssa ja lisäsin pari pientä käännöstä CompanyAvatarEditable-komponenttiin. Testasin käännöksiä paikallisesti ja ne toimivat pienen hienosäädön jälkeen hienosti omalla työkoneellani paikallisella palvelimella, mutta testatessani niitä dev-kehitysympäristössä ne eivät toimineet lainkaan. Kehitysympäristössä on käytössä Header-komponentti, joka ei ole osa web-komponenttikirjastoa ja se käyttää omaa Weblate-projektiaan. Epäilen ongelmien johtuvan jotenkin Weblate-projektien päällekkäisyyksistä. En kuitenkaan ehtinyt tutkia asiaa tarkemmin, sillä löysin CompanyAvatarEditable-komponentista bugin dev-kehitysympäristössä testatessani käännöksiä. Jostain syystä mukautettu change-event antaa erilaisen tuloksen nyt kuin aikaisemmin, enkä keksi lainkaan missä vika on. Asian selvittäminen jää seuraavalle päivälle.

Perjantai 16.03.2021

Luulin jo päässeeni hetkeksi CompanyAvatarEditable-komponentista eroon, mutta eilen löytämäni bugi on saatava korjattua. Se onkin päivän ensimmäinen tavoitteeni. Kollega toisesta tiimistä raportoi julkiselle Peacockin Google Chat-kanavalle, että hän oli löytänyt bugin Button-komponentista. Button-komponentti sijaitsee shadow DOM:ssa ja kun Button-komponenttia käytetään lomakkeessa, sen pitäisi lomaketta lähettäessä luoda loppukäyttäjälle näkymätön väliaikainen natiivi HTML:n button-elementti, jonka avulla lomake lähetetään eteenpäin. Jostain syystä väliaikaisen painikkeen luomisessa on joku bugi ja päivän toinen tavoitteeni on löytää bugi sekä saada se korjattua.

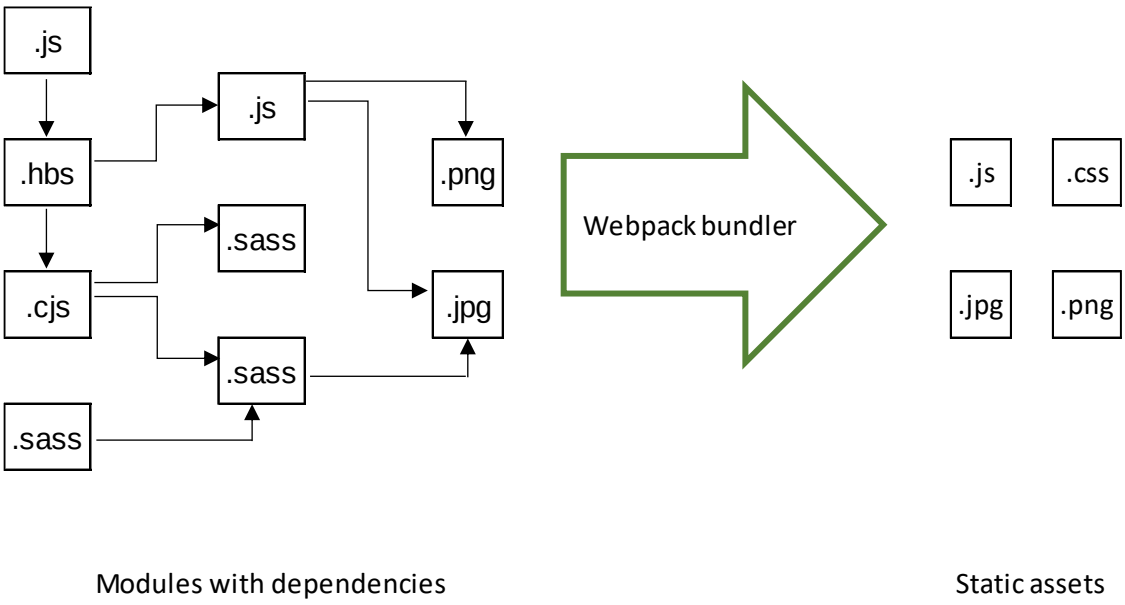
Korjasin päivän aikana CompanyAvatarEditable-komponenttiin ilmaantuneen bugin muuttamalla mukautetun change-eventin käsittelyä erilaiseksi kuin aikaisemmin. Loppukäyttäjälle toiminnallisuus näyttää aivan samalta kuin aiemminkin. Peacockin julkisella Google Chat-kanavalla erään toisen tiimin jäsen raportoi Icon-komponentin aiheuttamasta bugista, jossa ikonit eivät pysyneet samalla rivillä tekstin kanssa menuvalikossa. Olin korjannut aikaisemmin Icon-komponenttia, sillä se ei toiminut oikein Select-komponentin sisällä ja tämä muutos aiheutti virheen muualla. Palautin tekemäni muutoksen takaisin ja tein uuden muutoksen Select-komponenttiin, jonka jälkeen julkaisin web-komponenttikirjastosta version 0.10.0. Uudella versiolla kaikki toimii jälleen normaalisti.

Viikkoanalyysi

Kulunut viikko sujui mielestäni kokonaisuutena oikein hyvin. Yhdistin vihdoin Avatar-, CompanyAvatar- sekä CompanyAvatarEditable-komponentit web-komponenttikirjaston master-haaraan ja pääsen korjaamaan muita asioita, jotka täytyy tehdä ennen version 1.0-julkaisua. Valitettavasti API-rajapintoja rakentava kollegani sairastui keskiviikkona ja hän oli koko loppuviikon poissa, joten User API-rajapinta ei vielä tullut valmiiksi. Tarvitsen User API-rajapintaa, että voin rakentaa UserAvatar- sekä UserAvatarEditable-komponentit valmiiksi. Tämä ei kuitenkaan tule olemaan mikään valtavan suuri urakka, sillä voin uusiokäyttää paljon CompanyAvatar- sekä CompanyAvatarEditable-komponenttien koodista.

Kehityin viikon aikana ainakin Weblaten käytössä, sillä aikaisemmin en ollut tutustunut sen käyttöön juuri lainkaan. Weblatessa käännökset organisoidaan projekteihin ja käännöskomponentteihin. Yksi projekti voi sisältää useita komponentteja ja jokainen komponentti voi sisältää useita käännöksiä eri kielille. Weblaten käännöskomponentti voidaan yhdistää kohdesovelluksen GitHub-repositorioon, jonka jälkeen muutokset päivittyvät vuorovaikutteisesti molempiin suuntiin repositorion ja Weblaten välillä.

Lisäksi tutustuin kuluneen viikon aikana paremmin Webpack-niputukseen, eli bundlingiin. Olen toki käyttänyt Webpackia niputukseen koko web-komponenttikirjaston kehityskaaren ajan, mutta en ole tutustunut sen toimintaan tarkemmin. Webpackin toimintaperiaate on havainnollistettu kuviossa 9.



Kuvio 9. Webpackin toimintaperiaate (mukaillen Webpack 2021)

JavaScriptiä voidaan ajaa internetselaimessa kahdella tavalla. Ensimmäinen vaihtoehto on lisätä sivulle erillinen skripti jokaiselle toiminallisuudelle, mutta tämä ratkaisu on haastavaa skaalata, sillä liian monen skriptin yhtäaikainen lataaminen voi aiheuttaa verkon ruuhkautumisen. Toinen vaihtoehto on käyttää yhtä suurta JavaScript-tiedostoa, joka sisältää kaiken projektissa käytettävän koodin, mutta tämä vaihtoehto aiheuttaa haasteita koodin luettavuuteen, ylläpidettävyyteen, laajuuteen sekä kokoon. (Webpack 2021.) Edellä mainitut haasteet voidaan ohittaa niputtamalla moduulit yhdeksi paketiksi, joka käy jokaisen tiedoston erikseen läpi rekursiivisesti ja luo riippuvuuksista riippuvuusgraafin, jonka avulla funktiot suoritetaan oikeassa järjestyksessä (Srivastava 10.4.2018.).

Raportointiviikon aikana oli mukavaa saada bugi-ilmoituksia käytössä olevista web-komponenteista Peacockin julkiselle Google Chat-kanavalle, sillä bugien löytyminen tarkoittaa, että komponentteja käytetään. Suurimmat bugit täytyy korjata ennen version 1.0-julkaisua, minkä jälkeen rikkovia muutoksia (breaking changes) täyttyy pyrkiä välttämään, sillä komponentit saattavat olla jo tuotantoympäristössä käytössä.

Valitsin alun perin viikon tavoitteekseni asioiden tarkastelun eri sidosryhmien näkökulmista, mutta viikon aikana totesin, että nykyisessä tehtävässäni näkökulmia on niin paljon ja tekemäni asiat niin spesifejä, ettei useista asioista saa järkevä kuvaa monesta eri näkökulmasta. Erilaisia näkökulmia web-komponenttien tarkasteluun olisivat olleet esimerkiksi kollegoiden, tuoteomistajan, muiden kehittäjien sekä loppukäyttäjien näkökulmat. Uskoakseni web-komponenttikirjaston version 1.0 virallisen julkaisun jälkeen kokonaisuutta on helpompi tarkastella eri näkökulmista sen sijaan, että yksittäisiä komponentteja alettaisiin tarkastelemaan usealta kannalta ennen julkaisua.

3.7 Seurantaviikko 7

Viikon tavoitteet

Seitsemäs seurantaviikko sisältää maanantaista torstaihin neljä normaalia työpäivää, jonka jälkeen perjantai on vapaa pääsiäisen vuoksi. Kyseessä on ensimmäinen vajaa viikko raportointijakson aikana. Viikon tavoitteeni on saada käännösten osalta kaikki valmiiksi web-komponenttikirjaston version 1.0 julkaisua varten. Mikäli aikaa jää, korjaan komponenteissa jäljellä olevia pieniä bugeja.

Maanantai 29.03.2021

Päivän tavoitteeni on korjata edellisenä perjantaina tiimin Google Chat-kanavalle raportoitu Button-komponentin bugi, jotta komponenttia voitaisiin käyttää lomakkeissa, eli formeissa. Kunhan Button-komponentti toimii, julkaisen web-komponenttikirjastosta uuden version ja ilmoitan versionumeron bugin raportoineelle kehittäjälle, jotta hän voi päivittää riippuvuuden, eli dependencyn tuoreen versionumeron omaan projektiinsa. Mikäli API-rajapinnoista vastaava kollegani on tervehtynyt ja pääsee tänään töihin, kysyn User API-rajapinnan tilanteesta. Mikäli User API on valmis tai lähellä valmista, alan työstää UserAvatarEditable-komponenttia. Tänään kalenterissa on stand up-kokouksen lisäksi myös backlog refinement, jossa käydään läpi osaltani muun muassa CompanyAvatarEditable-komponentin tilannetta. Kyseinen tapaus on merkitty minun vastuulleni JIRA-tehtävienhallintaohjelmistossa.

Heikolta näyttävästä alusta huolimatta saavutin päivän tavoitteet hienosti. Lähes koko alkupäivä meni korjatessa bugia, jossa shadow DOM:n sisällä oleva Button-komponentti ei luonut loppukäyttäjälle näkymätöntä HTML:n button-elementtiä, jota komponentti käyttää sisäisesti lomakkeen lähettämiseen. Löysin lopulta ongelman tarkastuksesta, joka tarkistaa onko komponentin ”form”-attribuuttiin määritelty arvo vai ei. Eristin tarkastuksen omaan funktioonsa, jonka jälkeen komponentti toimi jokaisessa paikassa, jossa sitä testasin eli paikallisesti omalla koneellani sekä dev-kehitysympäristössä Docker-kontin sisällä. Lisäksi lisäsin toimivat käännökset CompanyAvatarEditable-komponenttiin, vaikkakaan en vielä kääntänyt minkään kielen tekstejä, vaan lisäsin ainoastaan vakiona käytettävät englanninkieliset versiot. Logiikka on kuitenkin toimiva ja käännökset tulevat hoitamaan Weblaten käyttöliittymän kautta jotkut muut henkilöt. Lisäksi lisäsin CompanyAvatarEditable-komponenttiin uuden logotyypin lisäämisen yhteyteen pakotetun välimuistin tarkastuksen, jolla vältetään edellisellä viikolla pohtimani ongelma välimuistiin jäävän logon kanssa toimimisesta. Company API-rajapintaa ylläpitävän kollegani täytyy vain lisätä rajapintaan viikon mittainen automaattityhjennys, jonka jälkeen kaiken pitäisi toimia suunnitellusti.

Päivän aikana avasin kolme uutta pull requestia ja olen tyytyväinen aikaansaannoksiini. Backlog refinementissa kuulin uutena tietona, että mikäli haluamme käyttää Google Analytics-työkalua, järjestelmään on lisättävä uusi kehote, jossa loppukäyttäjiltä on kysyttävä lupa evästeiden käyttöön. Tämä tarkoittaa, että loppukäyttäjien on hyväksyttävä kaksi lähes vastaavaa kehotetta, sillä tallennamme asiakkaiden tietoja jossain muussa vaiheessa eri syystä. Aiempi kehote ei kuitenkaan kata Google Analytics -palvelun käyttöä.

Tiistai 30.03.2021

Tiistai on lyhyen seurantaviikon toinen päivä ja tämän jälkeen viikko onkin jo puolivälissä. Päivän tavoitteeni on tarkastaa kolmeen eilen avaamaani pull requestiin tulleet kommentit ja tehdä komponentteihin tarvittavat korjaukset. Lisätavoitteeni on lisätä käännökset muihin niitä tarvitseviin komponentteihin. Lopuksi alan työstää UserAvatarEditable-komponenttia, mikäli User API-rajapinta alkaa olla käyttövalmis.

Saavutin päivän aikana aamulla asettamani pääasialliset tavoitteet. Korjasin pull requesteihin useissa eri vaiheissa tulleita muutospyyntöjä. Tehdessäni korjauksia tuli uusia muutospyyntöjä lähes koko ajan. Kokonaisuutena työpäivä oli hauska. Button-komponentti meni muutospyynnön katselmoinneista läpi ja yhdistin sen master-haaraan. Refaktoroin päivän aikana GET- ja POST-kutsut yhteen tiedostoon, joka on nimeltään LogotypeService.tsx. LogotypeService-tiedoston POST-kutsu on kuvattu kuvassa 4.

```
20  public static async postLogotype(companyId: string, base64Logo: string, mediaType: string): Promise<Response> {
21    const postOptions = {
22      method: 'POST',
23      headers: { Authorization: 'Bearer ' + LogotypeService.bearerToken, 'content-type': 'application/json' },
24      body: JSON.stringify({
25        companyId: companyId,
26        logotypeType: 'WebLogo',
27        image: base64Logo,
28        mediaType: mediaType
29      }),
30    };
31
32    return fetch(`${this.apiHost}/company/v1/companies/${companyId}/logotypes`, postOptions)
33    .then(res => LogotypeService.handleErrors(res))
34  }
```

Kuva 4. Geneerinen funktio logotyypin API-rajapintaan lähettämistä varten

Tiimin scrum masterin mukaan tällainen geneerinen logotyypin API-rajapintakutsut hoitava tiedosto on monikäyttöisempi kuin aiemmat kaksi erillistä tiedostoa. Lisäksi yhden tiedoston etuna on, että tarvittaessa tiedostoon voi rakentaa auttavia helper-funktioita, jos niille tulee tarvetta jossain muussa ympäristössä. Näin koodin rakenne on selkeämpää ja luettavampaa.

Keskiviikko 31.03.2021

Päivän tavoite on yhdistää CompanyAvatarEditable-komponentin välimuistin tallentamiseen liittyvät muutokset master-haaraan. Pull requestiin oli tullut uusia kommentteja ja ehdotuksia koodin refaktorointiin, mitkä toteutan aamupäivän aikana. Iltapäivällä stand up-kokouksen jälkeen lisään käännökset komponentteihin, joissa olisi käännettävää tekstiä mutta käännöksiä ei ole vielä lisätty. Käännösten lisäämisen jälkeen alan työstää

UserAvatar- ja UserAvatarEditable-komponentteja riippumatta User API-rajapinnan valmiustilasta.

Valitettavasti en saavuttanut päivän tavoitteita. Tein eilen koodikatselmointiin tulleet muutospyynnöt sekä lisäsin käännökset puuttuviin komponentteihin, mutta tiimin scrum master oli löytänyt menetelmän, jonka avulla voimme luopua Base64-formaatin käytöstä CompanyAvatarEditable-komponentissa käyttäjän ladatessa palvelimelle uuden logotyypin. Korjasin komponentin lataamaan Base64-formaatin sijaan kuvista tehtyjä Blobeja. Blobin etu verrattuna Base64-formaatissa olevaan kuvaan on, että selaimen ei tarvitse suorittaa kuvan kääntämistä lukujonoksi ja sitten takaisin kuvaksi erikseen. Lisäksi kuva mahtuu pienempään kokoon, joten menetelmä on monelta kannalta aiempaa hyödyllisempi.

Yksi suuri hidastava tekijä päivän työssä oli väärin dokumentoitu Company API-rajapinta. Ihmettelin kauan, että miksi kutsu ei mene läpi Company API-rajapintaan, vaikka kaikki oli dokumentaation mukaan oikein. Lopulta keksin testata kutsua Postman-ohjelmassa, jossa selvisi, että dokumentaatiossa kuvatun logotypeType -parametrin kuuluisi olla muodossa logoTypeType, jotta kutsu toimisi. Ilmoitin asiasta eteenpäin ja virhe tullaan jatkossa korjaamaan dokumentaatioon.

Torstai 01.04.2021

Torstai on seurantaviikon viimeinen päivä, sillä huomenna on pitkäperjantai. Tänään on Front-end tiimien välinen Sigweb-kokous, johon liittyen tiimin scrum master oli laittanut minulle viestiä, että hän aikoo puhua kokouksessa web-komponenttikirjaston muutoksista sekä asioista, joita kirjastoon on lisätty edellisen Sigweb-kokouksen jälkeen. Hän halusi ainakin muutaman kuvakaappauksen uudesta CompanyAvatarEditable-komponentista, jotka pystyn ottamaan Pageron dev-ympäristössä testatessani komponenttia Docker-kontissa. Lisäksi hän oli löytänyt pari pientä bugia Icon-komponentin tyylittelystä, silloin kun komponentin background-attribuutilla on arvo. Päivän tavoitteeni on ottaa tarvittavat kuvakaappaukset CompanyAvatarEditable-komponentista, korjata Icon-komponentin bugit sekä katsoa avoimeen pull requestiin tulleet muutospyynnöt ja tehdä asianmukaiset muutokset. Joudun tänään lopettamaan työt jo puolen päivän aikaan, sillä lähdemme mökille pääsiäiseksi.

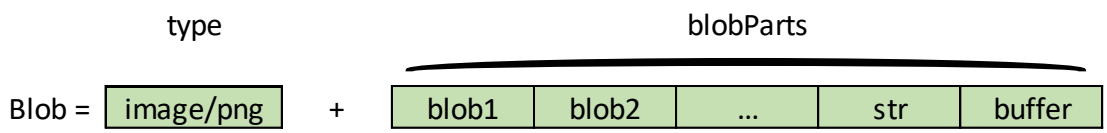
Korjasin aamulla Icon-komponentin tyylittelyvirheet, mutta en ehtinyt tehdä oikein mitään muuta stand up-kokouksen ja Sigweb-tapaamisen lisäksi, sillä lopetin työt aikaisin. Ilokseni CompanyAvatarEditable-komponentin pull request hyväksyttiin ja yhdistin kauan työstämäni komponentin web-komponenttikirjaston GitHub-repositorion master-haaraan.

Perjantai 02.04.2021

Vapaa

Viikkoanalyysi

Seitsemäs seurantaviikko oli työajaltaan normaalia viikkoa lyhyempi, mutta opin siitä huolimatta viikon aikana uusia asioita. En ole esimerkiksi aikaisemmin perehtynyt kovin paljon erilaisiin menetelmiin kuvien lähettämiseksi palvelimelle käyttäen natiivia HTML:n input-elementtiä, jonka tyyppi on tiedosto. Tällä viikolla tutustuin kuitenkin base64-enkoodauksen lisäksi Blobien, eli Binary large objectien käyttöön. Blob-objekti koostuu valinnaisesta merkkijonotyypistä, joka on yleensä tiedoston MIME-tyyppi, sekä blobParts-taulukosta, joka koostuu Blob/BufferSource/String-arvoista (JavaScript.info 2020.). Blobin rakenne on kuvattu kuviossa 10.



Kuvio 10. Blob-objektin rakenne (mukaillen JavaScript.info 2021)

Blobia voi käyttää URL-osoitteena kuvan näyttämiseen sellaisenaan myös muun muassa <img>-, ja <a>-tageissa lisäämällä osoitteen eteen etuliitteen blob://. Blobin URL-osoite luodaan käyttämällä staattista URL.createObjectURL() -metodia. (JavaScript in Plain English 2019.). Type-ominaisuuden ansiosta kuva voidaan myös ladata palvelimelle, tai käyttäjän tietokoneelle palvelimelta, jolloin se muutetaan automaattisesti Content-Type-muotoon tietoverkon request-kutsuissa. Blob-objektit ovat muuttumattomia, joka tarkoittaa, että Blobin sisällä olevaa dataa ei voida muokata. Tästä huolimatta Blob-objekteja voidaan tarvittaessa viipaloida ja näistä viipaleista voidaan luoda uusia Blob-objekteja. Käytös on saman kaltaista kuin JavaScriptin merkkijonoilla eli stringeillä. Vaikka merkkijonon sisällä olevaa kirjainta ei voida muuttaa, voidaan luoda kokonaan uusi merkkijono, joka sisältää halutun muutoksen. (JavaScript.info 2020.) Blob-objektin viipalointi on havainnoillistettu kuvassa 5.

```
3   Blob.slice([bytesStart], [byteEnd], [contentType]);
```

Kuva 5. Blob-objektin viipalointi (mukaillen JavaScript.-info 2020)

Blob-objektin viipalointi tapahtuu määrittelemällä arvot byteStart-, sekä byteEnd -argumenteille. byteStart-argumentti määrittelee viipaloinnin aloituskohdan ja sen oletusarvo on

0. byteEnd-argumentin oletusarvo on tiedoston viimeinen tavu. Lisäksi Blob-objektin viipaleelle voidaan määritellä contentType, joka on uuden viipaloidun Blobin mediatyyppi. Oletusarvoisesti uuden Blobin contentType on sama kuin lähdeobjektin. (JavaScript.info 2020.)

Kuluvan viikon tavoitteeksi olin asettanut itselleni rakentaa web-komponenttikirjaston käännökset koodin osalta valmiiksi version 1.0 julkaisua varten. Saavutin tavoitteen ja myös koodin osalta kaiken pitäisi toimia, vaikka itse varsinaiset käännökset ovat vielä tekemättä. Ensi viikolla pyrin selvittämään, kenen tehtävä on hoitaa käännökset Weblate-järjestelmään ja kuinka asia saataisiin vietyä eteenpäin.

3.8 Seurantaviikko 8

Viikon tavoitteet

Kahdeksas ja viimeinen seurantaviikko on edellisviikon tapaan yhden päivän normaalia lyhyempi, sillä pääsiäisvapaiden vuoksi työviikko alkaa tiistaista. Viikon sisältöä en osaa ennakoida etukäteen, mutta oletan aloittavani viikon rakentamalla UserAvatarEditable-, sekä UserAvatar-komponentteja samanlaisella logiikalla kuin CompanyAvatarEditable- ja CompanyAvatar-komponentit on rakennettu. Lisäksi viikon tavoitteeni on selvittää, kuinka Weblate-järjestelmään lisätään käännökset komponenteille ja ketkä henkilöt kääntämisen hoitavat.

Maanantai 05.04.2021

Vapaa

Tiistai 06.04.2021

Päivän ensimmäinen tavoitteeni on lisätä web-komponenttikirjaston riippuvuutena käyttämään pagero-design-tokens -repositorioon pieni muutos koskien Icon-komponentissa käytettäviä pag-size-icon -design tokeneita. Design tokeneita käytetään komponenttien ja nettisivujen tyylittelyyn, jotta muun muassa mitta- ja kokosuhteet pysyisivät yhtenäisinä koko järjestelmässä. Toinen tavoitteeni on alkaa työstää UserAvatar-, sekä UserAvatarEditable-komponentteja ja lisäksi selvittää, keneen Weblaten käännöksistä täytyy olla yhteydessä.

Saavutin tänään tavoitteeni. Avasin pagero-design-tokens -repositorioon pull requestin, joka sisälsi muutoksen pag-size-icon -design tokeneihin. Kun muutos oli katselmoitu ja

hyväksytty, yhdistin muutoksen sisältäneen haaran master-haaraan. Lisäksi korjasin web-komponenttikirjaston käännösten asetuksia Weblate-järjestelmästä ja nyt englantia käytetään lähdekielenä kaikille kielille. Tämä tarkoittaa sitä, että aina kun Weblate-projektiin lisätään uusi käännettävä kieli, kaikkia määrityksiä ei tarvitse tehdä erikseen alusta asti, vaan englanninkielistä tekstiä käytetään pohjana uudelle kielelle.

Lisäksi työstin päivän aikana UserAvatar-, ja UserAvatarEditable-komponentteja, mutta en voi rakentaa niitä valmiiksi ennen kuin User API-rajapinnan dokumentaatio on valmis. Tiedustelin dokumentaatiota API-rajapinnoista vastaavalta kollegaltani aamulla, mutta hän sanoi, ettei ole vielä ehtinyt edes aloittaa. Hän lupasi rakentaa dokumentaatiota tänään ja palata asiaan iltapäivällä. Toivottavasti dokumentaatio on valmis huomenna aamulla, jotta voin työstää komponentteja eteenpäin.

Päivän viimeinen tehtäväni oli Button-komponentista löytyneen bugin korjaaminen. Button-komponentti ei vieläkään löydä itseään lähintä lomaketta, sillä komponentti on shadow DOM -puun sisällä. Lisäsin komponenttiin ominaisuuden, joka tarkastaa, onko dokumentin aktiivinen elementti shadowRoot:n sisällä. Avattuani muutoksesta pull requestin lopetin työskentelyn.

Keskiviikko 07.04.2021

Aloitan päivän tarkastamalla Button-komponentin avoimeen pull requestiin tulleen katselmoijan kommentin ja teen tarvittaessa asianmukaisen korjauksen. Päivän pääasiallinen tavoite on kuitenkin avata uusi pull request UserAvatar ja -UserAvatarEditable-komponenteista. API-rajapinnoista vastaava kollegani oli tehnyt User API-rajapinnasta oman pull requestin viime yönä. Vaikka koodi ei vielä menisikään läpi katselmoinnista, uskon saavani pull requestin koodista melko hyvän kuvan uuden User API-rajapinnan käytöstä, josta näkisin ainakin kutsuissa käytettävät päätepisteet, eli endpointit.

En saavuttanut päivän tavoitteitani. En ehtinyt tehdä UserAvatar-, tai UserAvatarEditable-komponenteille mitään, sillä korjasin koko päivän yhtä ainoata bugia Button-komponentista. Tehtävä oli saada shadow DOM:n sisällä oleva komponentti tunnistamaan lähin lomake-elementti. Tehtävä olisi muuten helppo, mutta lomake-elementti on mahdollisesti myös oman shadow DOM-puun sisällä, jolloin ongelma muuttuu haastavaksi. Tutkin asiaa enimmäkseen Stackoverflow:n keskustelupalstalta ja lopulta löysin toimivan keinon, jonka avulla pääsen kulkemaan shadow DOM-rajojen läpi. En saanut päivän aikana juuri mitään konkreettista aikaiseksi, mutta opin toki paljon uutta shadow DOM-puuhun pääsystä toisen shadow DOM-puun sisällä olevasta komponentista käsin.

Torstai 08.04.2021

Torstain tavoitteeni on saada avattua pull requestit UserAvatar- ja UserAvatarEditable-komponenteista. API-rajapinnoista vastaava kollegani oli avannut pull requestin uudesta User API-rajapinnasta, mutta siihen tuli muutospyyntöjä pitkin päivää, eikä hän valitettavasti ehtinyt viimeistellä dokumentaatiota.

Tänään saavutin tavoitteeni. Avasin uuden pull requestin UserAvatar, sekä UserAvatarEditable-komponenttien haarasta. User API-rajapinnan pull requestissa ei tänään tapahtunut mitään muutoksia, joten dokumentaatiotakaan ei ole vielä saatavilla. En pääse etenemään UserAvatar-, tai UserAvatarEditable-komponenttien kanssa ennen User API-rajapinnan dokumentaation näkemistä tai avoimen pull requestin katselmointia. Käytin ylimääräistä työaikaani korjaamalla vihdoin tiedettyjä bugeja FileUploadAdvanced-komponentista.

Päivän päätteeksi tiedustelin vanhemmalta kollegaltani, voisimmeko käyttää FileUploadAdvanced-komponentissa File API-rajapintaa. Tällä hetkellä komponentissa on pakollinen url-attribuutti, jonka avulla määritellään ladattavan tiedoston päätepiste. Mielestäni pakollista url-määritettä parempi vaihtoehto olisi, että komponentti käyttäisi Pageron File API-rajapintaa, mikäli se on mahdollista. File API-rajapinnan käyttäminen helpottaisi tunnistautumista sekä vaadittujen rajapinnan parametrien määrittämistä FileUploadAdvanced-komponentin sisäisesti. Valitettavasti kollegani oli koko loppupäivän kiinni kokouksissa, emmekä ehtineet keskustella asiasta.

Perjantai 09.04.2021

Seurantaviikon ja -jakson viimeisenä päivänä tavoitteeni on selvittää kollegani kanssa, mikä olisi järkevin tapa toteuttaa API-rajapinnan päätepisteen määrittely FileUploadAdvanced-komponenttiin. Lisäksi tarkastan UserAvatar-, sekä UserAvatarEditable-komponenttien pull requestiin tulleet kommentit ja teen tarvittavat korjaukset.

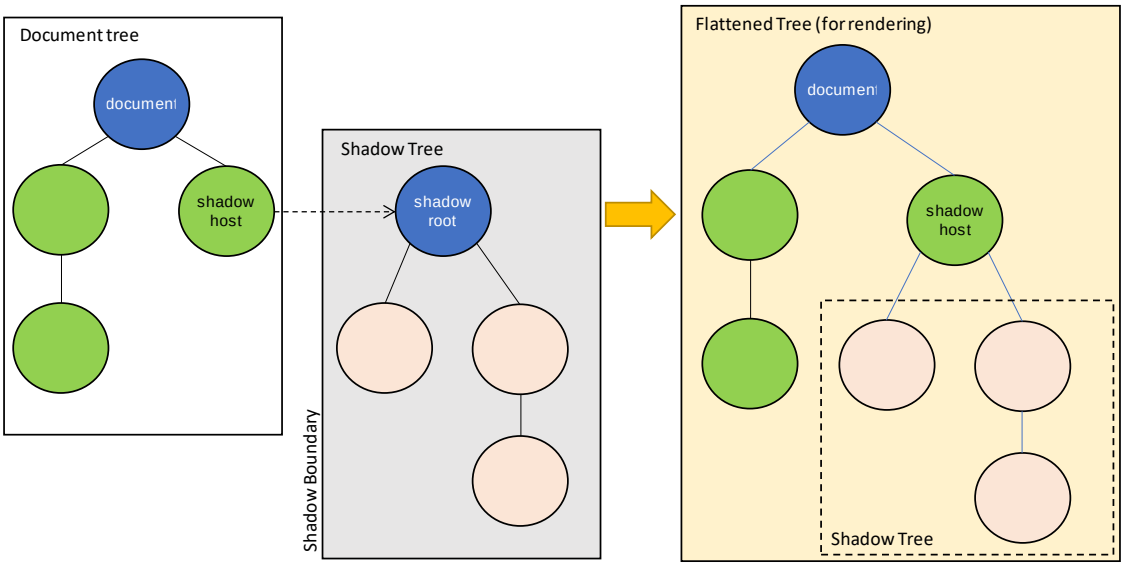
Perjantain työpäivä oli sarja erilaisia vastoinkäymisiä ja sattumuksia, joiden vuoksi en päässyt tänään tavoitteisiini. Ennen stand up-kokousta tarkastelin FileUploadAdvanced-komponentin API-rajapinnan käsittelyä itsekseni ja sovin seniorikollegani kanssa parikoodaussession stand up-kokouksen jälkeen. Aloitimme session ja ideoimme uutta toteutusta, jonka jälkeen testasimme API-kutsuja File API-rajapinnan päätepisteeseen Postman-ohjelmalla. Valitettavasti API-kutsut eivät toimineet dokumentaatiossa esitellyllä tavalla. Lähdin tässä välissä lounaalle ja lupasin kollegalleni palata asiaan.

Lounaan jälkeen sain kutsun toimimaan Postman-ohjelmassa, jonka jälkeen yritin kirjautua www-dev.pageroonline.com testiympäristöön tarkistamaan, näkyykö lähettämäni tiedosto testiympäristössä. Macin käyttöjärjestelmä sisältää keychain-salasanantallennuspalvelun, joka ei hyväksynyt selaimeen asennettua uutta sertifikaattiani, eikä salasanaani, joten en päässyt testiympäristöön lainkaan. Tästä alkoi tuntien mittainen taistelu sertifikaattien, Docker-konttien sekä toimintaympäristön recipe-algoritmien konfigurointitiedostojen kanssa.

Korjasin loppupäivän ongelmia, kunnes lopulta sain kaiken toimimaan. Lähetin uuden kutsun Postman-ohjelman kautta ja löysin lähettämäni tiedoston www-dev.pageroonline.com-ympäristöstä. Valitettavasti en ehtinyt edes vilkaista UserAvatar- ja UserAvatarEditable-komponenttien pull requesteihin tulleita kommentteja. Toisaalta User API-rajapintakaan ei ole vielä valmis, joten en olisi muutenkaan päässyt kovin pitkälle UserAvatar-komponenttien kanssa. Uuden oppimisen kannalta päivän positiivinen puoli on, että opin uusia asioita Docker-konttien konfiguroinnista.

Viikkoanalyysi

Seurantajakson viimeisen viikon tavoitteeni oli rakentaa UserAvatar- sekä UserAvatarEditable-komponentit sekä avata niistä uusi pull request web-komponenttikirjaston GitHub-repositorioon. Toinen tavoitteeni oli selvittää Weblate-järjestelmän käännösasioita. Saavutin kuitenkin viikon tavoitteeni, vaikka viikko oli päivän normaalia lyhyempi. Substanssiosaamisen näkökulmasta seurantaviikon aikana ymmärrykseni shadow DOM:n käytöstä lisääntyi merkittävästi. Yhden raportointihetkellä pieneltä tuntuneen bugin vuoksi jouduin etsimään internetistä valtavasti tietoa shadow DOM:n rakenteesta, sekä eri shadow DOM-puiden välisestä liikenteestä. Shadow DOM:n rakenne on selitetty kuviossa 11.



Kuvio 11. Shadow DOM:n suhde tavalliseen DOMiin (mukaillen MDN Web Docs 2021)

Shadow DOM tuntui minusta käsitteenä todella sekavalta, kun aloin rakentaa web-komponenttikirjastoa. Pikkuhiljaa ymmärrykseni sen toiminnasta on kasvanut pala palalta, mutta tunnen tällä viikolla vihdoin ymmärtäneeni kunnolla, mistä shadow DOM:n käytössä on kyse. Shadow DOM:n keskeisiä käsitteitä ovat

- **shadow host** on tavallinen DOM-node, josta shadow DOM alkaa
- **shadow tree** on DOM-puu shadow DOM:n sisällä
- **shadow boundary** on tavallisen DOM:n ja shadow DOM:n välinen raja, ja
- **shadow roo**t on shadow DOM-puun juuri, josta shadow DOM:n sisäiset elementit lähtevät.

Piilottamalla web-komponentteja shadow DOM:n sisään varmistetaan, ettei komponentin sisään vuoda tyylittelyjä ympäristöstä, sillä shadow DOM:n sisällä olevat komponentit ovat kapseloituina omaan sisäiseen DOM-puuhunsa. (MDN Web Docs 2021.) Kuvion 11 vasemmassa laidassa näkyy dokumentista haarautuva dokumenttipuu, eli DOM, jota kutsutaan tietyssä kontekstissa myös nimellä light DOM. Dokumentista lähtee shadow host-node, joka kapseloi shadow DOM:n sisäisen elementin omaan shadow DOM-puuhunsa. Kuvion 11 oikeanpuoleisessa ruudussa näkyy selaimeen renderöity lopputulos, jossa shadow DOM:n sisältö on rajattu katkoviivoilla. Shadow treen sisään ei vuoda tyylittelyjä muualta light DOM:sta, eikä sen sisäisiä elementtejä pysty manipuloimaan esimerkiksi internetselaimen tarkastelutyökalua käyttäen.

Viikon aikana myös Weblate-järjestelmän käyttö tuli tutummaksi ja opin asettamaan Weblate-projektiin lähdekielen, jotta muiden kielien lisääminen on helpompaa. Lähdekielenä on luontaisesti helpointa käyttää englantia. Viikon aikana minulle selvisi, että tiimin tuoteomistaja on ilmoittanut eri maiden kääntäville tahoille, jotta uuden järjestelmän käännökset tulisi hoitaa lähiaikoina, sillä uuden järjestelmän versio 1.0 julkaistaan pian. Uuden

järjestelmän käyttöliittymissä käytetään rakentamiani web-komponenttikirjaston komponentteja.

Seurantaviikon aikana ehdin korjata bugeja FileUploadAdvanced-komponentista, joka on jäänyt hieman muiden asioiden alle lähiaikoina. Tällä hetkellä komponentti ei ole vielä aivan valmis tuotannossa käytettäväksi, mutta uskoisin tilanteen muuttuvan lähiaikoina. Peacockin julkiselle Google Chat-kanavalle on tullut komponentista tiedusteluita muilta tiimeiltä ja he haluaisivat saada komponentin käyttöönsä.

Kokonaisuutena kahdeksas seurantaviikko oli lyhyydestään huolimatta antoisa ja pääsin edistämään monia asioita, joita olen joutunut aiemmin lykkäämään jostakin syystä. Näitä asioita ovat muun muassa shadow DOM-puiden välillä liikkuminen, FileUploadAdvanced-komponentin bugit, UserAvatar- ja UserAvatarEditable-komponenttien pohjakoodit.

4 Pohdinta ja päätelmät

Kahdeksan viikon seurantajakso tuntui menevän ohi nopeasti. Koen ammatillisen osaamiseni kehittyneen valtavasti kuluneen kahden kuukauden aikana. Tiesin jo kirjoitusprosessin alussa, että yksi suurimmista haasteista raportoinnin aikana tulee olemaan työstä kirjoittaminen suomeksi, sillä työkieleni on sataprosenttisesti englanti. Toisinaan tuntuu, etten osaa ilmaista asiaani kunnolla, koska useille töissä käytettäville termeille on vaikeaa löytää suomenkielisiä vastineita. Asiaan varmasti vaikuttaa myös, etten ole koskaan työskennellyt järjestelmäkehitystehtävissä suomenkielisessä työympäristössä. Tästä huolimatta huomaan, että myös kirjoitustaitoni on kehittynyt sujuvammaksi raportoinnin edetessä.

Tarkastellessani päiväkirjamerkintöjäni huomaan, miten yksitoikkoista ja itsenään toistavaa työ voi toisinaan olla, vaikka se ei tekohetkellä sellaiselta tunnukaan. Rakensin koko seurantajakson ajan vain muutamaa komponenttia, joiden spesifikaatiot muuttuivat raportoinnin aikana. Lisäksi rakennusprosessin aikana saattoi ilmetä asioita, joiden vuoksi ainakin osa tehdystä työstä täytyi tehdä uudelleen. Jos jälkeenpäin ajatellen tekisin jotain toisin päiväkirjan kirjoittamisen aikajaksolla, niin suunnittelisin komponentit tarkemmin ennen rakentamisen aloittamista. Pitkällä tähtäimellä se säästäisi työaikaa.

Seurantajakson aikana kehitin substanssiosaamistani monella eri osa-alueella ja hahmotan työtehtäviin kuuluvaa kokonaisuutta huomattavasti paremmin kuin seurannan alussa. Osaamiseni kehittyi muun muassa palvelimelle lähetettävien POST- ja GET-kutsujen käytössä, lähetettävän tiedoston datan manipuloinnissa sekä shadow DOM:n rajojen yli

kulkemisessa, mitkä ovat varmasti hyödyllisiä taitoja tulevaisuutta ajatellen. Kirjoitusprosessin ja päivittäisen raportoinnin myötä koen ajan ja tehtävien hallinnan kehittyneen kuin itsestään. Vaikka opinnäytetyön päivittäinen raportointiosuus onkin jo päättynyt, otin tavakseni kirjoittaa muistiin työpäivän aikana tavoitteita sekä ottaa muistiinpanoja, joihin voin tarvittaessa palata.

Kirjoitusprosessin aikana olen löytänyt erilaisia menetelmiä työssä ilmenevien ongelmien ratkaisuun. Stackoverflow:n tai GitHubin selailun sijaan luotettavaa tietoa löytää muun muassa tieteellisistä asiaa koskevista artikkeleista. Olen myös alkanut hyödyntää MDN Web Docsia spesifien JavaScript -metodien sekä eventien tutkimiseen, sillä MDN Web Docseista löytyvä tieto on ainakin sataprosenttisen paikkansapitävää ja luotettavaa. Koen myös päivittäisten työtehtävien ennakoinnin ja Microsoft OneNote-ohjelmaan kirjoitettujen muistiinpanojen olevan hyödyllinen menetelmä työtehtävien hallinnassa.

Päiväkirjan kirjoittamisen ja viikonloppuisten reflektointien kautta opin asettamaan itselleni realistisempia päivätavoitteita sekä hyväksymään, että toisinaan itsestä riippumattomienkin syiden vuoksi työ ei etene halutulla tavalla. Työn aktiivinen analysointi ja pohtiminen auttoi jäsentämään uudet asiat selkeämmin. Kirjoitusprosessin aikana mielekkäintä työssä oli päästä mukaan kehitystyöhön koko yrityksen laajuisessa dev-ympäristössä pelkän paikallisen, omalla koneella käytettävän Storybookin kanssa toimimisen sijaan. Storybook on oiva työkalu, mutta on tärkeää, että kehittäjä ymmärtää yrityksen kehitysympäristöä laajemminkin.

Opinnäytetyön kirjoittamisen aikana oli kiinnostavaa huomata, että raportointijakson loppupuolella ymmärsin jo suurimman osan asioista, joista yrityksen sisäisten kehitystiimien välillä keskustellaan. Aikaisemmin osa asioista on ollut turhauttavasti ymmärrykseni ulkopuolella. Näitä asioita ei oppia voinut googlaamalla, sillä kyse oli yrityksen sisäisistä asioista sekä termeistä. Olisin toki voinut avata suuni ja kysyä, mutta en halunnut olla se henkilö, joka kyselee jatkuvasti kaikesta. Olen lopulta tyytyväinen, että opin asiat orgaanisesti. Lisäksi tarkastelujakson loppupuolella oli todella innostavaa huomata, että pystyi toimimaan jo aika lailla itsenäisesti verrattuna raportointijakson alkuun, vaikka aikaa ei ollut kulunut kuin kahdeksan viikkoa. Uskon, että päiväkirjaa kirjoittaessa tehdyistä muistiinpanoista tuli hyvä tapa jatkoa ajatellenkin. Asiat jäävät usein paremmin mieleen, kun ne kirjoitetaan ylös. Lisäksi niihin voi tarvittaessa palata.

Odotan innolla web-komponenttikirjaston julkaisua. On kiinnostavaa nähdä, miten muut tuotekehitystiimit ottavat komponenttikirjaston vastaan. Yksi harmittava seikka on, että monta viikkoa työstämäni FileUploadAdvanced-komponentti pudotetaan pois

komponenttivalikoimasta ainakin ensimmäisen version julkaisussa. Pudotus johtuu siitä, että tiimin scrum masterin mielestä komponentin ominaisuus, joka lataa valitut tiedostot suoraan palvelimelle ennen käyttäjän lähettäessä lomaketta, olisi liian hankala ja monimutkainen käytettäväksi muille kehittäjille ja sotisi web-komponenttikirjaston helppokäyttöisyyttä vastaan. Jatkossa tarkoitus on siirtää mahdollisesti joitain FileUploadAdvanced-komponentin sisäisiä ominaisuuksia FileUpload-komponenttiin, joten työ ei välttämättä mene täysin hukkaan. Vaikka ominaisuudet eivät siirtyisikään, koen silti saaneeni komponentin kehittämisestä valtavasti arvokasta kokemusta ja oppia.

Kun web-komponenttikirjaston version 1.0 julkaisu on hoidettu onnistuneesti ja suurimmat bugit on korjattu, on aika katsoa tulevaisuuteen. Jatkossa uusille komponenteille tulee varmasti kysyntää ja tiedänkin jo muutaman, jotka on sovittu kehitettäviksi version 1.0 julkaisun jälkeen. Tulevaisuudessa, kun web-komponenttikirjasto on vakaa ja täysin kaikkien yrityksen kehittäjien käytössä, aikomukseni on tutustua tarkemmin Front-endin lisäksi Back-end -kehityspuoleen. Muun muassa tietokantojen kanssa toimimisesta minulla ei ole juuri lainkaan oikeaa työelämän kokemusta, vaan tämän hetkiset tietoni perustuvat enimmäkseen Haaga-Helian SQL-kursseihin sekä omissa projekteissani käyttämääni ilmaiseen MongoDB-tietokantaan.

Työn analysointi on mahdollistanut hyödyllisten havaintojen tekemisen työprosessin menetelmiä koskien. Olen asettanut itseäni eri sidosryhmien asemiin ja ajatellut työni jälkeä toisista näkökulmista, kuten muun muassa niiden tiimien kehittäjien perspektiivistä, jotka komponentteja lopulta tulevat käyttämään. Opinnäytetyön struktuuri tarjosi minulle pitkällä tähtäimellä oivan työkalun itseni kehittämiseen, sillä innostuin muistiinpanojen tekemisestä niin paljon, että teen niitä varmasti jatkossakin.

Lähteet

Agendium 2020. Agile, Waterfall, Kanban ja muut: 6 yleistä menetelmää projektityöhön – ja miksi sinun kannattaa valita omasi? Luettavissa: https://www.agendium.com/projektinhallinta/menetelmat-projektityohon. Luettu: 21.2.2021.

Brightspot 2021. Developer life: 5 reasons why the code review process is critical for developers. Luettavissa: https://www.brightspot.com/products/developer-life-5-reasons-why-the-code-review-process-is-critical-for-developers. Luettu: 12.2.2021.

Burchard, E. 2017. Refactoring JavaScript. O'Reilly Media Inc. Sebastopol.

Burman, B. 11.3.2019. How To Access Images Securely with OAuth 2.0. Luettavissa: https://www.twelve21.io/how-to-access-images-securely-with-oauth-2-0/. Luettu: 26.2.2021.

BytesofGigabytes 2021. How HTTP request and response works. Luettavissa: https://bytesofgigabytes.com/networking/how-http-request-and-response-works/. Luettu: 7.3.2021.

Bytutorial 2016. Simple way detecting if a page has a scroll bar. Luettavissa: https://bytutorial.com/blogs/javascript/simple-way-detecting-if-a-page-has-a-scroll-bar. Luettu: 15.3.2021.

Crockford, D. 2008. JavaScript: The Good Parts. Yahoo Press. Palo Alto.

Eberhard, W. 2018. Microservices. Dpunkt. Heidelberg.

Fong, J. 30.8.2018. Are Containers Replacing Virtual Machines? Luettavissa: https://www.docker.com/blog/containers-replacing-virtual-machines/. Luettu: 26.2.2021.

Fridman, A. 6.5.2016. The Massive Downside of Agile Software Development. Luettavissa: https://www.inc.com/adam-fridman/the-massive-downside-of-agile-software-development.html. Luettu: 21.3.2021.

GitHub Docs 2021. About pull requests. Luettavissa: https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests. Luettu: 25.2.2021.

Hyvönen, E. 2018. Semanttinen web: Linkitetyn avoimen datan käsikirja. Gaudeamus. Tallinna.

Internet Engineering Task Force 2021. About. Luettavissa: https://www.ietf.org/about/. Luettu: 28.2.2021.

Internet Engineering Task Force 1998. The ”data” URL scheme. Luettavissa: https://tools.ietf.org/html/rfc2397. Luettu: 28.2.2021.

JavaScript.info, 25.11.2020. Blob. Luettavissa: https://javascript.info/blob. Luettu: 4.4.2021.

JavaScript in Plain English, 31.10.2019. What are Blobs used for in JavaScript? Luettavissa: https://javascript.plainenglish.io/javascript-blob-why-is-it-useful-20c372dfca00. Luettu: 4.4.2021.

Korpela, J. 2008. CSS verkkosivujen muotoilussa. WSOY. Jyväskylä.

Krug, S. 2000. Don’t make me think! A common sense approach to web usability. New Riders. Berkeley.

MacDonald, M. 2014. HTML5: The missing manual. O’Reilly Media Inc. Sebastopol.

Martin Fowler 2014. Microservices. Luettavissa: https://martinfowler.com/articles/microservices.html. Luettu: 11.2.2021.

Martin, R. 2009. Clean Code: A Handbook of Agile Software Craftmanship. Pearson education, Inc. Boston.

MDN Web Docs 2021. 400 Bad Request. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400. Luettu: 04.03.2021.

MDN Web Docs 2021. Base64. Luettavissa: https://developer.mozilla.org/en-US/docs/Glossary/Base64. Luettu: 07.03.2021.

MDN Web Docs 2021. Polyfill. Luettavissa: https://developer.mozilla.org/en-US/docs/Glossary/Polyfill. Luettu: 25.2.2021.

MDN Web Docs 2021. Using shadow DOM. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM. Luettu: 7.4.2021.

Miller, T., Ripley, R. 2020. Fixing Your Scrum. Pragmatic Bookshelf. Raleigh.

Pagero 2021. Tietoja Pagerosta. Luettavissa: https://www.pagero.fi/. Luettu: 10.2.2021.

Postman 2021. What is Postman? Luettavissa: https://www.postman.com/. Luettu: 28.2.2021.

Ruohotie, P. 2000. Oppiminen ja ammatillinen kasvu. WSOY. Juva.

Sass 2021. Mixins. Luettavissa: https://sass-lang.com/guide. Luettu: 21.2.2021.

Shakir, S. 28.2.2017. Stop Using A Loading Spinner, There's Something Better. Luettavissa: https://uxdesign.cc/stop-using-a-loading-spinner-theres-something-better-d186194f771e. Luettu: 13.3.2021.

Srivastava, U. 10.4.2018. Webpack – why and what. Luettavissa: https://medium.com/js-imaginea/webpack-why-and-what-4948433cc2d3. Luettu: 28.3.2021.

Stencil 2021. Component Lifecycle Methods. Luettavissa: https://stenciljs.com/docs/component-lifecycle. Luettu: 21.3.2021.

Storybook 2021. What's a Story. Luettavissa: https://storybook.js.org/docs/react/get-started/whats-a-story. Luettu 12.2.2021.

Tadić, S. 19.3.2018. Uploading files using 'fetch' and 'FormData'. Luettavissa: https://muffinman.io/blog/uploading-files-using-fetch-multipart-form-data/. Luettu: 13.4.2021.

TechCrunch 2012. What Exactly Is GitHub Anyway? Luettavissa: https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/. Luettu: 25.2.2021.

The Verge 2020. Microsoft will bid farewell to Internet Explorer and legacy Edge in 2021. Luettavissa: https://www.theverge.com/2020/8/17/21372487/microsoft-internet-explorer-11-support-end-365-legacy-edge. Luettu: 21.2.2021.

The World Wide Web Consortium 2020. Resize Observer. Luettavissa: https://www.w3.org/TR/resize-observer/#api. Luettu: 15.2.2021.

Thomas, G. 2.4.2019. A beginner’s guide to Docker – how to create your first Docker application. Luettavissa: https://www.freecodecamp.org/news/a-beginners-guide-to-docker-how-to-create-your-first-docker-application-cc03de9b639f/. Luettu: 28.2.2021.

Viitala, R. 2007. Henkilöstöjohtaminen: Strateginen kilpailutekijä. Edita. Helsinki.

W3Schools 2021. Input Checkbox indeterminate Property. Luettavissa: https://www.w3schools.com/jsref/prop_checkbox_indeterminate.asp. Luettu: 25.2.2021.

W3Schools 2021. JavaScript Versions. Luettavissa: https://www.w3schools.com/js/js_versions.asp. Luettu: 21.2.2021.

Wallenius, N. 16.11.2020. Mikä on Docker ja mitä hyötyä siitä on? Luettavissa: https://niklaswallenius.fi/mika-on-docker/. Luettu: 27.2.2021.

Webpack 2021. Concepts. Luettavissa: https://webpack.js.org/concepts/. Luettu: 28.3.2021.