HAAGA-HELIA
ammattikorkeakoulu

Front end developer työ moderneilla menetelmillä

Markus Kivikoski

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2020

HAAGA-HELIA
ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

25.12.2020

Tekijä tai tekijät Markus Kivikoski	**Ryhmätunnus tai aloitusvuosi** 2017
Raportin nimi Front end developer työ moderneilla menetelmillä	**Sivu- ja liitesivumäärä** 26+1
Opettajat tai ohjaajat Juha Hinkula	

Tämän päiväkirjamuotoisen opinnäytetyön tarkoitus on seurata opiskelijan arkea, kehittymistä ja pohdintaa front-end kehittäjänä. Opinnäytetyö sisältää aikavälillä 7.9.2020 – 16.11.2020 tehdyt päiväkirjamerkinnät, viikkoanalyyseja sekä loppupäätelmät ja pohdinnat seurantajakson ajalta.

Opiskelija toimii yrityksen ensimmäisenä in-house front-end developerina ja on vastuussa yrityksen tarjoaman portaalipalvelun kehittämisestä sekä suunnittelusta yhdessä pääsuunnittelijan kanssa.

Seurantajakson aikana tapahtui paljon kehitystä erityisesti itsenäisen työskentelyn saralla. Uudessa työpaikassa aloittaminen vaatii aina paljon aikaa uuteen projektiin orientoitumiseen, joten suurin osa seurantajaksosta kului uusien työtapojen ja ympäristöjen oppimiseen. Opittavaa on vielä paljon, sillä työtehtävät seurantajakson aikana olivat hyvin paljon toisiaan muistuttavia ja keskittyivät hyvin rajattuun osaan sovelluksen suunnitelluista toiminnallisuuksista.

Asiasanat
Front end, ohjelmistokehitys, javascript, react

Sisällys

1 Johdanto

Opinnäytetyö seuraa työpäiviäni Robot Housella aikavälillä 7.9.2020 – 16.11.2020. Päiväkirjamuotoisen opinnäytetyön raportointi tapahtuu päivittäisellä työtehtävien kuvaamisella, niiden pohdinnalla sekä viikottaisella analyysillä. Seurattava Robot Housen työpaikka on toinen alan työpaikka, jossa olen työskennellyt. Olin kuuden kuukauden pituisessa työharjoittelussa käyttöliittymäkehittäjänä Congrid Oy:llä 2020 tammikuusta heinäkuuhun. Robot Housen työpaikkailmoituksessa haettiin junioritasoista tai jopa vastikään valmistunutta työntekijää, joten työkokemukseni perusteella sovin kuvaan täydellisesti. Työtehtäväni Robot Housella tulevat sisältämään heidän uuden portaalisovelluksen suunnittelu ja kehittäminen React javascript viitekehyksen avulla. Olen opiskellut sekä javascriptiä, että Reactia koulussa ja nämä työkalut olivat käytössä myös kevään harjoittelun aikana. Koen osaavani Reactin ja javascriptin sulavasti, joten niiden osalta työnteon ei pitäisi tuottaa ongelmia. Töiden alussa selvisi kuitenkin, että koodipohja oli rakennettu typescriptillä, eikä javascriptillä, joka tuotti alussa hieman hankaluuksia. Typescript on kuitenkin muutamia ominaisuuksia lukuunottamatta identtinen kieli javascriptin kanssa, joten sen oppiminen ei tuottanut suurempia ongelmia. Aika ajoin kuitenkin tulee törmättyä johonkin uuteen typescript virheeseen, tiedonhaku on kuitenkin helppoa kun ymmärtää ongelman.

Luin seurantajakson aikana Pete Goodliffen teoksen Becoming a better programmer pohitessani miten tehostaisin uuden koodipohjan sisäistämistä. Tietopohjana käytin myös aikaisemmin lukemaani Clean Code (Robert Martin, 2008), joka sisältää paljon pohdintaa siitä, mikä tekee hyvästä koodista hyvää ja huonosta huonoa. Kirja on vaikuttanut omaan työskentelyyni hyvin paljon ja erityisesti Robot Housen ainoana käyttöliittymäkehittäjänä tehtäväni on tuottaa niin hyvää ja ymmärrettävää koodia kuin mahdollista, sillä uusien työntekijöiden palkkaaminen tulevaisuudessa tulee olemaan väistämätöntä ja ajatustyö alussa säästää resursseja myöhemmin.

Robot House on RPA palveluita ja toteutuksia tarjoava startup yritys. RPA, eli robotic process automation tai ohjelmistorobotiikka on teknologia, joka auttaa automatisoimaan tietotyön rutiiniprosesseja, jotka ovat toistuvia ja sääntöpohjaisia. Robot House on kehittämässä asiakaskäyttöön tarkoitettua portaalia, jonka parissa tulen työskentelemään. Robot Housen muu henkilöstö koostuu itse RPA kehittäjistä, myyjistä ja markkinoinnista sekä johdosta. Yrityksellä on toimisto Helsingissä, mutta töitä voi tehdä täysiaikaisesti etätyöskentelynä.

1.1 Käsitteet

Front end: Ohjelmiston visuaalinen käyttöliittymä ja sen loogiset osat (esimerkiksi nappulan avulla kutsuttava funktio).
RPA: Robotic process automation, ohjelmistorobotiikka. Teknologia, joka mahdollistaa toistuvien rutiiniprosessien automatisoinnin.
Back end: Ohjelmiston osa, joka pyörii palvelimella ja hoitaa esimerkiksi käyttöliittymän ja tietokannan välistä rajapintaa.
JavaScript: Pääasiassa web-ympäristöjen rakentamiseen käytetty ohjelmointikieli.
Css: Cascading style sheets, html dokumenteille annettavat tyyliohjeet.
React: JavaScript ohjelmistokehys, eräänlainen komponenttipohjainen työkalupakki helpottamaan web-kehitystyötä Javascript kielellä.
Redux: React ohjelmien tilanhallintaan tarkoitettu lisäkirjasto.
State: Ohjelman tila, käytetään esimerkiksi säilyttämään valitun monivalintakohteen arvo.
Prop / Property: Tietoa voidaan välittää React komponentilta toiselle property arvojen avulla.
Npm: Paketinhallintajärjestelmä jota kautta esimerkiksi asennetaan React sovellukseen lisäkirjastoja kuten Redux.
Yarn: Vaihtoehto Npm:lle.
Power shell: Microsoftin komentolinjatyökalu.
Jira: Projektinhallintatyökalu, virtuaalinen ”tarralapputaulu”.
Wireframe / Rautalanka: Konsepti käyttöliittymän ulkoasusta, suunnittelijan tekemä malli, jonka mukaan front end developer mallintaa käyttöliittymän.

2 Lähtötilanteen kuvaus

Hain työpaikkaa (kutsutaan tätä paikkaa nimellä ”Robot house”) 19.8.2020 Academic Work rekrytointipalvelun kautta. Rekrytointiprosessi eteni nopeasti ja jo seuraavan viikon maanantaina 24.8.2020 pidettiin prosessin ensimmäinen puhelinhaastattelu. Olin edennyt toisessa (Kutsutaan tätä paikkaa nimellä ”Full-Stack road”) Academic Workin kautta haettavassa prosessissa jo viimeiseen vaiheeseen, joten Robot Housen prosessissa haluttiin kiirehtiä ja seuraava haastattelu, tällä kertaa Robot Housen liiketoimintajohtajan kanssa sovittiin jo seuraavalle päivälle. Robot Housen haastattelu meni hyvin ja sain kutsun viimeiseen haastatteluun Robot Housen johtavan ux suunnittelijan kanssa. Tällä välin olin saanut negatiivisen vastauksen Full-Stack Roadilta. Torstaina 27.8.2020 pidetyn lyhyen keskustelun jälkeen ux-johtaja oli vakuuttunut ja seuraavaksi sovittiin töiden aloituspäivä: 7.9.2020.

Kyseessä tulee olemaan toinen koulutustani vastaava työpaikka. Olin palkallisessa työharjoittelussa vuoden 2020 tammikuusta heinäkuuhun. Olen siis hyvin alkuvaiheessa ammatillista kehittymistäni. Hallitsen Reactin perusteet erinomaisesti ja myös Redux on perusteiltaan tuttu. Tunnen Git versionhallinan ja osaan käyttää sitä tehokkaasti. Koen, että eniten kehitettävää minulla on tietorakenteiden ja algoritmien, eli puhtaan tietojenkäsittelytieteen osaamisessa ja se on yksi isoimmista osa-alueista, johon minun täytyy urani edetessä keskittyä hyvin paljon.

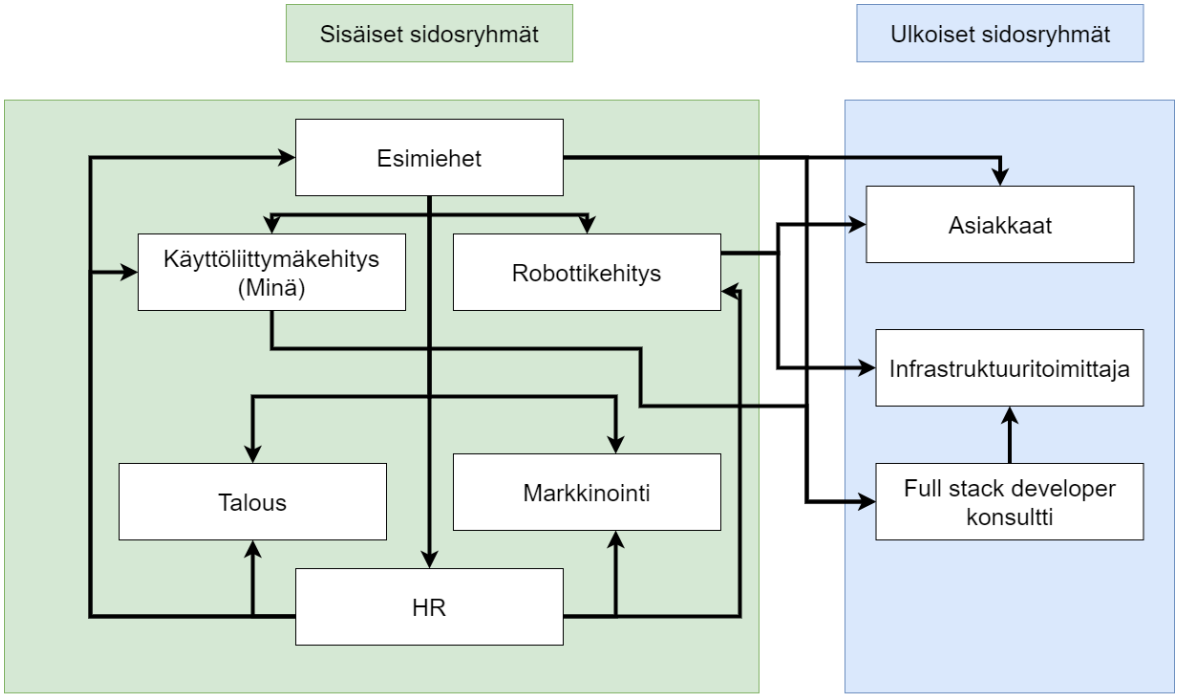
2.1 Pääasiallinen työnkuva

Tulen olemaan Robot housen ensimmäinen talon sisäinen front end kehittäjä. He ovat aikaisemmin hankkineet front end kehittäjiä ulkoisilta toimijoilta. Pääsen muotoilemaan työskentelyäni ja asemaani haluamaani suuntaan. ”Osaston” ainoana henkilönä pääsen myös vaikuttamaan paljon toteutustapoihin ja suunnitteluun. Toistaiseksi kuitenkin Robot Housella on edelleen töissä yksi konsultti, joka toimii full stack developerina, eli hänen vastuullaan on sekä back endin, että front endin kehittäminen. Oma työnkuvani tulee sisältämään Robot housen portaali-aplikaation käyttöliittymäkehitys, uusien toiminnallisuuksien suunnittelu ja toteutus sekä ulkoasun suunnittelu. Suurin paino tulee kuitenkin olemaan toiminnallisuuksien ja ulkoasukehtiysken toteuttaminen, sillä pääasiassa Robot housen UI/UX designer hoitaa suunnittelutyön. Minun vastuullani tulee olemaan näiden suunnitelmien paras mahdollinen tekninen toteutus, on myös tärkeää tuoda oma näkemys suunnittelijan kuuluviin parhaan mahdollisen lopputuloksen aikaansaamiseksi. Työnkuvaani kuuluu myös odotetusti viikottaisiin yrityskokouksiin,

kuukausittaisiin tiimikokouksiin sekä päivittäisiin (joskus harvemmin) kehittäjien välisiin scrum kokouksiin.

2.2 Sidosryhmät työpaikalla

Tulen työskentelemään hyvin itsenäisesti, ainoat tahot suorassa vuorovaikutuksessa kanssani tulevat olemaan suorat esimieheni, full stack developer konsultti, sekä HR johtaja. Robot Housen muut kehittäjät toimivat RPA toteutuksien parissa, joten minulla tuskin tulee olemaan suoraa yhteistyötä heidän kanssaan missään vaiheessa.



3 Päiväkirjaraportointi

3.1 Seurantaviikko 1

Maanantai 07.09.2020

Ensimmäiselle päivälle ei ole selkeitä maaleja tai tavoitteita.
Olin sovittuun aikaan aamulla paikalla Robot Housen toimiston aulassa odottamassa teknologiajohtajan saapumista. Hänen saavuttuaan siirryimme ensimmäiseksi tutustumaan toimistorakennuksen tiloihin ja tietenkin rakennuksen omaan kahvilaan, josta haimme kupilliset lämmintä juomaa siivittämään ensimmäistä perehdytystä.
Ensimmäinen perehdytys käsitteli hyvin tiiviisti Robot Housen toimintaa ja sen tarjoamia palveluita ja tuotteita. Robot Housen pääasiallinen liiketoiminta on RPA-robottien ohjelmointi ja myynti. Itse en tule olemaan osallinen robottien ohjelmointiin. Rbot House on lanseeraamassa robottien ja virtuaalikoneiden tietojen seurantaan tarkoitettua ”portaalia”. Tämä portaali on juurikin se asia, mihin minun työntekoni tulee keskittymään.
Päivän toisessa perehdytyksessä tutustuttiin portaalin code baseen sen rakentaneen konsultin johdolla. Esittely oli aika pintapuoleinen, enkä oikein osannut sillä hetkellä mitään kovin teknistä kysyäkään. Päälimmäisenä mielen päälle jäi se, että portaali oli tehty käyttäen TypeScriptiä JavaScriptin sijasta, mistä ei ollut mitään puhetta rekrytointiprosessin aikana. TypeScript on itselleni tuttu vain konseptitasolla, mutta olen varma, että opin sen nopeasti.

Tiistai 08.09.2020

Asetin tiistaina ensimmäiseksi tavoitteekseni uuteen code baseen tutustumisen, joka tulee varmasti viemään useamman viikon.
Toisena päivänä minulle oltiin osoitettu jo ensimmäiset taskit, joita aloin itsenäisesti tekemään. Tällä kertaa kotitoimistolta. Tehtävänäni oli yksinkertaisten rautalankamallien täydentäminen seuraavan viikon demoa varten. Rautalanka jota tiistaina työstin sisälsi yksinkertaisen listanäkymän sekä klikkaamalla paljastuvaa lisätietoa valittavista listakohteista, ei mitään monimutkaista. Tehtävän suorittamista vaikeutti kuitenkin uusi tuntematon code base. Uuden code basen sisäistäminen on aikaa vievä prosessi, jonka aikana yksinkertaistenkin taskien tekeminen on huomattavasti hitaampaa ja kömpelömpää, kuin se olisi työntekijälle, joka on työskennellyt code basen parissa pidempään. Pidimmekin pari puhelua tuon aiemmin mainitun konsultin kanssa, joka selitti taas vähän tarkemmin eri komponenttien toimintaa ja käyttötapoja. Näiden neuvojen siivittämänä rautalankamallin rakentaminen sujui helposti loppuun.

Keskiviikko 09.09.2020

Keskiviikon aamupalaverissa lyötiin lukkoon asiat, jotka on saatava toimimaan ensi viikon tiistain demoa varten. Tiukka aikataulu siis luvassa, joka tuntuu omalta kohdaltani muita tiukemmalta, sillä aloitin kehitysympäristössä työskentelyn vasta eilen. Sain tehtäväkseni portaalin aikajananäkymän tooltippien luomisen. Tooltipeissä on tarkoitus laskea ja näyttää dataa valituista aikajanatapahtumista. Tehtävä oli yksinkertainen, sillä haluttu data oli jo valmiiksi haettu Redux stateen. Ainoa laskentaa kaipaava data oli aikaleimojen formatointi ja aikavyöhyke-erojen kiertäminen.

Torstai 10.09.2020

Torstaina jatkoin saman task pinon kanssa ja sain eilen aloittamani tooltip taskin tehtyä loppuun. Seuraavana oli vuorossa olemassa olevien rautalankamallien täyttäminen mock datalla demoa varten, yksinkertaista kovakoodaamista valmiiden määrittelyiden mukaan. Aloitin myös hieman aiempaa monimutkaisempaa taskia, jonka tavoittena oli näyttää aikajananäkymän sisältämää dataa diagrammeissa Recharts kirjaston avulla. Kaikki tarvittava data oli taas valmiiksi Redux statessa, jolloin tehtäväksi jäi tuon datan muovaaminen muotoon, joka voidaan antaa Recharts diagrammille.
Tänään en päässyt diagrammin kanssa perus rautalankaa ja mock dataa pidemmälle ajanpuutteen takia.

Perjantai 11.09.2020

Perjantaina jatkoin diagrammi taskin parissa. Rautalanka oli valmis eilisen jälkeen, joten pystyin keskittymään Redux staten datan saamiseen oikeaan muotoon. Data oli yksinkertaisessa muodossa, joten oikeanlaisen algoritmin kirjoittaminen ei vaatinut paljoa: yksinkertainen listan mappaus ja nodejen lajittelu uusiin listoihin. Tehtävää kuitenkin vaikeutti tapa, jolla eroteltiin mitkä nodet haluttiin erotella listasta valitun kentän perusteella. Tarvitsin apua konsultilta, mutta päivä alkoi olla tässä vaiheessa jo niin pitkällä, että oli aika yrityksen viikkopalaverille. Viikkopalaveri on ainoa koko yrityksen yhteinen kokous, jossa käydään läpi taloustilanne, henkilöstöosaston asiat, asiakasasiat sekä pidetään yllä yhteishenkeä ja sosialisoidaan työkavereiden kanssa. Viikkopalaveri on

iso osa Robot housen työkulttuuria, sillä työt ovat täysin etätöitä myös koronatilanteen ulkopuolellakin. Robot housella on kyllä toimistotilat Espoossa, mutta työntekijät käyvät siellä todella harvoin. Työntekijät asuvat ympäri Suomea, joten fyysistä yhdessäoloa ei järjestetä usein. Tässä ensimmäisessä viikkopalaverissani olin osana HR-osiota ja pääsin esittelemään itseni toisille. Viikkopalaverin jälkeen pidimme vielä kehitystiimin kesken tilannekatsauksen ja näytti, että kaikki asiat ovat ajallaan joten lopetimme hommat tältä päivältä.

Viikkoanalyysi: Uuden codebasen nopea oppiminen

Uudessa työpaikassa on aina uusi codebase ja uudet työskentely ja ohjelmointitavat. Törmäsin tähän kevään harjoittelussa sekä nykyisessä työpaikassani ja se tulee pitämään paikkaansa jokaisessa tulevaisuuden työpaikassani.

Tällä viikolla pääsin heti ohjelmoimaan, joten uuteen codebaseen tutustuminen alkoi sukeltamalla välittömästi oikeiden töiden pariin. Codebase ei ole massiivinen, verrattuna esimerkiksi kevään harjoittelussa kehittämääni codebaseen. Koodia on kuitenkin paljon ja se luonnollisesti jonkun muun kuin itseni kirjoittamaa, joten sen opettelussa tulee menemään aikaa. Pete Goodliffe on listannut seitsemän kohtaa mitä uuteen codebaseen tutustuessa tulisi ottaa huomioon kirjassaan Becoming a better programmer:

- Löydä oikea paikka aloittaa koodin tarkastelu
- Selvitä mitä koodin jokainen osio tekee ja miten se saavuttaa sen
- Mitoita koodin laatu
- Selvitä miten sovelluksessa navigoidaan
- Sisäistä koodissa käytetty kieli, jotta muutoksesi sopivat joukkoon
- Löydä jokaisen toiminnallisuuden looginen sijainti
- Sisäistä koodin ja sen tärkeimpien ”satelliittiosien” suhde (esimerkiksi koodin testit ja dokumentaatio)

(Pete Goodliffe, Becoming a better programmer, 2014.)
Listan jälkeen Goodliffe sanoo, että nämä asiat on opittava nopeasti, sillä ”et halua, että ensimmäiset muutoksesi ovat liian noloja, vahingossa toista tehtyä työtä tai riko jotain jossain muualla”. Tähän codebaseen tutustumisesta erityisen vaikeaksi tekee lähes täysi dokumentaation puute. Konsultti, joka on tähän mennessä rakentanut aplikaation lähes täysin itse ei ole kirjoittanut minkäänlaista dokumentaatiota (rajapintadokumentaatiota lukuunottamatta) eri komponenttien toiminnasta. Tämä tarkoittaa sitä, että minun on selvitettävä itse miten kaikki toimii, varsinkin kun nyt alkuun vaikuttaa siltä, ettei konsultilla ole aikaa vastailla kysymyksiin tiukkojen aikatauluvaatimusten takia.

Goodliffe sanoo, että projektin riippuvuuksien tarkastelu kertoo paljon: käyttääkö sovellus paljon kolmannen osapuolen kirjastoja? Kuinka paljon ne vaativat opiskelua? ”Kaikkea ei voi oppia kerralla, varsinkin kun osa kirjastoista on massiivisia.” Toinen Goodliffen vinkeistä codebasen opetteluun on koodin laadun tutkailu. Hän kehottaa tarkastelemaan kommentoinnin määrää ja laatua. Tätä vinkkiä voi tarkastella kahdesta eri näkökulmasta: hyvälaatuiset kommentit ovat hyviä ja huonolaatuiset ovat huonoja, tai kommentointi missään muodossa on huono asia. Jälkimmäistä näkökulmaa tukee lukemani Clean Code (Robert C. Martin, 2008), jossa argumentoidaan, että koodin tulisi olla niin suoraviivaista, sisältää selkeästi nimettyjä muuttujia kuin mahdollista ja sisältää mahdollisimman yksinkertaista abstraktiota. Näin voidaan teoriassa välttyä kommentoinnin tarpeelta kokonaan. Pyrin itsekin työskentelemään tällaisella clean code periaatteella. On huomattavasti helpompaa aloittaa uuden tehtävän parissa, kun jokainen abstraktio tai muuttujan nimeäminen ei tarvitse erillistä palaveria alkuperäisen ohjelmoijan kanssa. Clean code periaate auttaa sekä itseäni, että kaikkia jotka tulevat työskentelemään code basen parissa. Olen useassa kohtaa Robot Housen codebasea törmännyt yhdellä kirjaimella nimettyihin muuttujiin, joka tekee vähääkään monimutkaisemman funktion tulkitsemisesta hyvin hankalaa ja hidastaa kenen tahansa työskentelyä huomattavasti.

Goodliffen kirja ei ole web-developer näkökulmasta kirjoitettu, mutta suuri osa sen sisällöstä kääntyy myös web-kehitykseen. Selaimen React- ja Redux kehittäjätyökalut ovat elintärkeitä niin normaalissa työskentelyssäkin, kuin uuteen codebaseen tutustuessa. React kehitystyökalujen ”components” työkalu on äärimmäisen hyödyllinen. Se toimii kuin inspect-element, mutta se kertoo hyvin tarkasti komponentin tiedostonimen, sekä mikä sen parent component on.

3.2 Seurantaviikko 2

29.9.2020 tiistai

Päivän tavoite: Saa rautalankamallien toteutukset valmiiksi.

Tänään jatkoin keskeneräisten rautalankamallien rakentamista. Eilen kesken jäänyt lisäasetusten konditionaalirenderöinti oli toistaiseksi viimeinen rautalankoihin liittyvä työ. En pidä siitä, että ongelmanratkaisu jää kesken päivän päätteeksi, sillä seuraavana aamuna kestää tavallista kauemmin päästä kiinni eilen katkenneisiin ajatuksiin. Tähän

tarkoitukseen työpäiväkirjan loppupäivän mietteet, on hyvä osio. Nyt ongelmana oli, että staten muuttuessa koko ohjelma kaatuu. Kaikki näytti olevan oikein kirjoitettu ja event handlerit ja logiikka oli suoraan kopioitu eilen tekemästäni samanlaisesta toiminnallisuudesta. Jonkin aikaa pohdittuani kävi ilmi, että tekemäni sisällön renderöinnistä huolehtiva funktio suoritettiin lomake-wrapper-komponentin ulkopuolella. Tämä aiheutti ohjelman kaatumisen. Ongelma korjaantui yksinkertaisesti siirtämällä render funktio lomake-wrapperin sisälle. Nyt kaikki renderöityy oikeaan aikaan ja oikeaan paikkaan. Tehtävä suoritettu.
Seuraavaksi otin työn alle n. viikko sitten aloittamani dataikkunan full screen näkymän. Olin aloittanut featuren tekemistä vähäsen, mutta ainoa konkreettinen asia oli nappula, jonka oli määrä avata uusi näkymä. Näytettävän ikkunan pohjaksi valitsin Material UI:n Dialog komponentin. Sen demosivun esimerkkien perusteella komponentti sopi täydellisesti tämän featuren tekemiseen.
Event handler funktio piti huolen komponentin ”showFullscreen” staten muutoksista, joka puolestaan kontrolloi milloin Dialog komponentti näytetään. Graafin lisääminen Dialog komponentin sisään oli helppoa: kopioin jo olemassa olevan pienen dataikkunan sisällön Dialogin sisään. Kopioin kuitenkin aluksi liikaa koodia ja myös Dialog komponentin sisään renderöitiin Dialog komponentin avaava nappula, jolloin niitä voitiin avata loputon määrä. Wrappereiden poistaminen ratkaisi ongelman. Nyt näkymässä oli oikea graafi oikealla datalla, mutta huonosti skaalattuna isolle näytölle.

Päivän tavoite saavutettiin. Tänään aloittamani featuren työstö tulee todennäköisesti viemään seuraavat 2 päivää.

30.9.2020 keskiviikko

Päivän tavoite: Tee valmiiksi taulukko/graafi näkymän vaihto, responsiivisuus kuntoon, graafin skaalaus kuntoon.
Jatkoin eilen aloittamaani ”full screen dialog” featuren tekoa. Olin saanut oikean graafin näyttämään haluttua dataa koko ruudun kokoisella Dialog komponentilla, seuraavaksi oli tarkoitus saada sama data näkymään taulukkomuodossa. Ensimmäinen vaihe tämän saavuttamiseksi, oli luoda nappulat eri näkymien vaihtoa varten. Yksinkertainen komponentin statea muuttava event handler funktio hoiti tämän. Seuraavaksi aloin tutkimaan datan näyttöä varten sopivia npm moduuleja. Päädyin lukemaan uuden Material UI:n oman datataulukomponentin dokumentaatiota. En ollut ennen Material UI:ta käyttäessäni törmännyt tähän komponenttiin, joten sen täytyy olla suhteellisen uusi. Ehdin jo innostumaan komponentista, mutta kävi ilmi, että se oli maksullisen lisenssin takana.

Maksulliset moduulit ovat myös uusi asia itselleni, ihan oletettavaa, että sellaisia on, en vain ole ennen nähnyt niitä. Maksullisuuden ja ennen kaikkea moduulin ”early-access” statuksen vuoksi käännyin datataulukosta pois ja päädyin työstämään featurea code-basesta löytyvän listauskomponentin avulla. Harmikseni huomasin, ettei komponentti voi näyttää samaa graafille syötettävää dataa suoraan saman muotoisena, vaan se vaatii hieman työstämistä toimiakseen. En päässyt tähän vaiheeseen asti tänään, sillä iltapäivän tilannekatsaus alkoi ja työaika oli lopuillaan.
Päivän tavoitteet saavutettiin osittain: taulukkonäkymä vaatii odotettua enemmän työtä. Muut asiat sain valmiiksi tähän vaiheeseen riittävälle tasolle, kaikki tarvitsee vielä lopullista viilaamista, mutta haluan saada ensin kaikki toiminnalisuudet valmiiksi. Arvioin eilen tämän featuren viemän ajan väärin, feature valmistunee viimeistään perjantaina.

1.10.2020 torstai

Fullscreen featuren työstäminen jatkui tänäänkin, tein taas muutoksen listauskomponentin kanssa ja päädyin Material UI:n Table-komponenttiin. Valintaani vaikutti pääasiassa sen valinnainen kompakti koko, joka saavutetaan ilman CSS-määrityksiä. Komponentista myös löytyy sisäänrakennettu paginaatio toiminnallisuus, joka helpottaa käytettävyyttä suurten datamäärien selailussa.
Loppupeleissä en joutunutkaan muokkaamaan taulukolle syötettävän datan luovaa funktiota, toisin kuin aikaisemmin ennustin. Data voitiin antaa yksinkertaisesti map metodilla suoraan taululle. Loppupäivästä palasin responsiivisuuden viilaamisen pariin. Graafi ei tällä hetkellä skaalaudu oikein. Varsinkin suuren datasetin piirto on ongelmallista, kun listan viimeiset osat jäävät kokonaan avautuvan Dialog komponentin ulkopuolelle. Iltapäivän tilannekatsaus oli alkamassa ja päätin jättää loput työt huomiselle.

2.10.2020 perjantai

Päivän tavoite: Saa valmiiksi fullscreen dialog feature.

Feature on toiminnallisuuksiltaan täysin valmis, jäljellä oli enää muotoilua ja responsiivisuuden parantelua. CSS kuulostaa front-end kehityksen yksinkertaisimmalta asialta, pelkkää asioiden liikuttelua ja värien muutoksia. Tosiasiassa yksinkertaiselta kuulostavakin CSS-task voi olla hyvin monimutkainen, varsinkin kun DOM elementtejä on useita päällekäin. Selaimen inspect-element toiminto on erinomainen työkalu tällaisissa tilanteissa, huomattavasti parempi, kuin IDE. Inspect elementiä käyttämällä ei tarvitse odottaa hot-reloadia jokaisen uuden määrityksen kirjoittamisen jälkeen vaan muutokset näkee suoraan kuten ne olisivat koodiin kirjoitettuna. En saanut koko näytön kokoista

graafia ihan täysin skaalaamaan oikein pienemmillä näytöillä, mutta asia ei ole niin suuri ongelma, sillä sovellusta käytetään vain työpöytäkoneilla. Päivän tavoitteisiin siis periaatteessa yllettiin.

Viikkoanalyysi: Valmiiden pakettien hyödyntäminen

React.js:n yksi suurimmista vahvuuksista web kehityksen frameworkina on sen komponenttipohjaisuus. React sovelluksen näkymiä ei rakenneta yhtenä staattisena kokonaisuutena (vaikka niinkin voidaan tehdä), sen sijaan sivu jaetaan pieniin, mielellään vain yhtä toimintoa suorittaviin komponentteihin. Esimerkiksi kalenterisovellus voitaisiin jakaa itse kalenteriin, tekstikenttään merkintöjä varten, muistiinpanolistaan sekä muistiinpanon tallennus ja poistonappi komponentteihin. Jokainen näistä komponenteista on oma javascript tiedostonsa, joita kutsutaan yhdessä niinsanotussa ”parent komponentissa”. Kun toiminnallisuudet on jaettu komponentteihin, niitä voidaan käyttää helposti myös muualla, kuin esimerkin kalenterinäkymässä. Sovelluksessa voi olla myös toinen näkymä, jossa käytetään samoja nappulakomponentteja kuin kalenterinäkymässä. Näin toimittuna sovelluksen visuaalinen ilme on helppo pitää yhtenäisenä ja säästytään toistuvasta työstä, kun nappuloita ei tarvitse tehdä aina puhtaalta pöydältä.

React.js on avoimen lähdekoodin framework, mikä tarkoittaa sitä, että kuka tahansa voi osallistua sen kehittämiseen tai julkaista Reactille rakentamiaan komponentteja muiden käyttöön esimerkiksi Npm paketinhallintasysteemin avulla (Aikaisemmissa päiväkirjamerkinnöissä mainitsemani Material Ui (1 503 855 viikottaista latausta) on yksi npm:n kautta asennettava komponenttikirjasto)(Npmjs.com). Aikaisemmassa esimerkissä puhuin kalenterista. Kalenterin voisi tietenkin rakentaa täysin tyhjästä itse, mutta siihen tarkoitukseen löytyy tuhansia käyttövalmiita npm paketteja (npmjs.com, 4017 pakettia). Kehittäjä voi asentaa haluamiaan paketteja ohjelmaansa käyttäen komentorivin npm install komentoa. Npm:stä löytyy lähes mitä tahansa kuviteltavissa olevaa toiminnallisuutta suorittavaa tai ulkoasullista komponenttia. Npm pakettien käyttö vauhdittaa kehitystyötä huomattavasti, esimerkiksi Robot housella tekemäni dataikkuna toiminallisuuden valmiiksi saaminen olisi kestänyt äärimmäisen paljon kauemmin, mikäli diagrammit olisi täytynyt tehdä alusta itse. Recharts kirjasto on hyvin tehokas datan visualisointiin tarkoitettu työkalu, jolta löytyy myös hyvin kattava dokumentaatio. Recharts käytännössä hoitaa kaiken itse, kunhan sille antaa oikeanmuotoisen datasetin. Sisäänrakennettujen propertyjen, kuten legend, joka lisää diagrammiin värien tai akselien selityksn, ansiosta diagrammin ulkoasun ja

toiminnallisuuden muokkaus on äärimmäisen helppoa. Suurimmilla ja suosituimmilla kirjastoilla on yleensä erinomainen dokumentaatio, jolloin uusien pakettien käyttöönotto sujuu nopeasti.

Npm pakettien varjopuolena on se, että projektin kasvaessa pakettien määrä lisääntyy ja niitä päivitetään eri tahtiin. Kalenteripaketti voi vaatia toimiakseen kolme muuta riippuvuutta, eli dependencyä, jotka päivittyessään voivat rikkoa jonkin toisen projektiin asennetun paketin täysin. Versionhallinnan tärkeys korostuu etenkin silloin, kun paketteja on paljon. Hyvä käytäntö on päivittää paketit tasaisin väliajoin, jolloin suurilta vaikeasti korjattavilta muutoksilta vältytään. Pakettien päivittämättä jättäminen voi tarkoittaa pitkällä tähtäimellä sitä, ettei jonkin uuden kriittisen toiminnallisuuden implementointi onnistu lainkaan, koska se vaatii uusimpia versioita joistain paketeista ja pakettien päivittäminen puolestaan hajoittaisi jotain perustavanlaatuisia toiminnallisuuksia. Tämä voi pahimmillaan aiheuttaa pitkiä taukoja muussa kehityksessä, kun resursseja on siirrettävä päivitysten tekemiseen ja niistä aiheutuvien bugien korjaukseen. (Npm, Dependency hell.)

3.3 Seurantaviikko 3

5.10.2020 maanantai

Sain viime perjantaina fullscreen graafinäkymän valmiiksi, joten sain vasta aamupalaverissa tietää mitä päivän työt tuovat tullessaan. Sovelluksesta puuttuu vielä muutama rautalankamalli, joten jatkoin niiden parissa. Yksinkertainen tehtävä, mutta ensimmäinen ongelma ilmaantui, kun rautalangan mukaan haluttiin alasvetovalikko, johon pystyy lisäämään vaihtoehtoja itse. Sopiva komponentti kyllä löytyi nopeasti Material UI:sta sisäänrakennettuna, mutta sovellus käyttää vanhaa versiota kirjastosta, joka ei tätä Autofill komponenttia tue. Aloin siis päivittämään paketteja, eli yksinkertaisesti nostin Material-ui/lab paketin versionumeroa, poistin node_modulet sekä package-lockin ja suoritin npm install komennon uudestaan. Asian pitäisi olla sillä selvä, mutta paketteja päivittäessä on aina varauduttava siihen, että joku menee pieleen. Sain ”failed to compile”

viestin jostain täysin muun paketin node_module tiedostosta ja aikani asiaa pyöriteltyäni huomasin, etten pystynyt enää palauttamaan vanhaa toimivaa versionumeroa jostain tuntemattomasta syystä. Myös master branch oli omalla koneellani samanlaisessa solmussa, vaikka olin tehnyt kaikki muutokseni uuteen feature development branchiin. Jonkun ajan jälkeen tulin siihen tulokseen, että koko repository on helpointa vain poistaa ja kloonata uudestaan. Otin tekemäni muutokset talteen erilliseen tiedostoon ja kloonasin repositoryn uudestaan Githubista. Ongelma oli korjaantunut ja sovelluksen pystyi jälleen kokoamaan. Haluamani komponentti sensijaan ei edelleenkään toiminut. Päätin jättää jatkon huomiseen.

6.10.2020 tiistai

Päivä alkoi taas aamupalaverilla, tällä kertaa eilisen päivän tekniset ongelmat tuoreessa mielessä. Aamupalaverissa ei tullut mitään uutta tietoa ja seuraavana vuorossa oli uusi tuotekehitystiimin ensimmäinen kuukausikokous. Tuotekehityspalaverin jälkeen koitin uudestaan eilistä Material-ui/lab paketin päivitystä. En oikeastaan odottanut mitään muuta lopputulemaa, kuin toisinnon eilisestä. Ja niinhän siinä kävi: joku node_moduleista hajosi totaalisesti ja ratkaisuna toimi jälleen repositorion uudelleenkloonaaminen.

Hylkäsin toistaiseksi Autocomplete komponentin ja keskityin rautalankamallin loppuosaan ja sainkin sen valmiiksi päivän loppuun mennessä. Juttelimmekin konsultin kanssa mahdollisia syitä. Tulimme sellaiseen tulokseen, että mahdollinen syypää olisi käyttämämme eri paketinhallintajärjestelmä. Minä olen tottunut käyttämään npm:ää ja konsultti käyttää yarnia. En vielä testannut hypoteesia, mutta epäilen, että se voi hyvinkin pitää paikkaansa.

8.10.2020 torstai

Tänään jatkoin http kutsujen tekemistä. Eilen sain yhteen komponenttiin yksinkertaisen listauksen tehtyä ja tänään oli tarkoitus tehdä listausta eräänlaiseen monivalintakomponenttiin. Toteutusta pohtiessani huomasin, ettei API palauta kaikkia monivalintakomponentin tarvitsemia parametreja. Soitin suunnittelijalle ja totesimme, että asia tosiaan on näin ja ettei taskia voida suorittaa tällä hetkellä loppuun. Työ meni jäihin odottamaan API päivitystä.

Siirryin jälleen rautalankahommiin, joka alkaa sujua rutiininomaisesti. Sainkin työn valmiiksi hyvin nopeasti, toiminnallisuuksia lukuunottamatta, joiden viimeistelyn päätin jättää huomiselle. Olen lähipäivinä tehnyt sen verran ylitöitä, että voi hyvällä omatunnolla lopettaa työt ajoissa.

9.10.2020 perjantai

Tänään jatkoin eilen aloittamaani rautalankamallia, tai sen puuttuvaa toiminnallisuutta. Tähän tarvitsin kuitenkin komponentin luoneen konsultin apua, sillä pelkkä yksinkertainen datan passaaminen ei toiminut suoraan. Odotellessa konsultin vapautumista aloin työstämään uutta API-kutsu taskia, jossa oli taas tarkoitus hakea dataa ja näyttää se listalla. Suoraviivaista tekemistä eilisen mallin mukaan.

Konsultin vapauduttua hän selitti monivalintakomponentin toiminnallisuutta ja kävi ilmi, että sille passattava data on oltava osa sen wrapper komponentille passattavaa data objektia. Tämän jälkeen oli helppo lisätä tarvittava informaatio dataobjektin luovan funktion palautukseen ja passata tämä lomakewrapperille. Toiminnallisuutta ei kuitenkaan ollut täysin mahdollista tehdä tässä vaiheessa loppuun, sillä se vaatii muutoksia sovelluksen back-endiin.

Viikkoanalyysi: Huijarisyndrooma

Huijarisyndrooma on psykologinen ilmiö, joka saa ihmisen epäilemään omia taitojaan ja saavutuksiaan. Huijarisyndroomasta kärsivä uskoo, että hänen saavutuksensa ovat vain ja ainoastaan hyvän tuurin ansiota, tai että hän on onnistunut huijaamaan muut uskomaan hänen olevan parempi kuin todellisuudessa onkaan. Huijarisyndroomasta kärsivä kysyy itseltään:

- Olenko tarpeeksi älykäs ollakseni ohjelmoija?
- Kuulunko tänne?
- Teinkö virheen valitessani tämän alan?
- Onko minun edes mahdollista oppia kaikki mitä muut osaavat?
- Olenko vain onnekas?

(Psychology today, Impostor syndrome.)
Huijarisyndroomaa esiintyy kaikilla aloilla, eikä se rajoitu pelkästään ammatillisiin tilanteisiin. Haluan kuitenkin tarkastella huijarisyndroomaa erityisesti ohjelmistokehtiyksen näkökulmasta.

Huijarisyndrooman kehittyminen voi alkaa jo lapsuudessa. Psychology todayn artikkelin mukaan lasten vanhemmat antavat kahdenlaisia viestejä lapsilleen, jotka aiheuttavat riittämättömyyden tunnetta. Ensimmäinen on ylikriittisyys. Kun lapsi kuulee jatkuvaa arvostelua siitä mikä ei suju tai ole täydellistä he oppivat ettei millään muulla kuin täydellisyydellä ole väliä. Tällaisissa tilanteissa vanhemmat huomaavat lapsessa ainoastaan täydellisestä oletusarvosta poikkeavat asiat. Tohtori Suzanne Lawryn mukaan

perfektionismi yhdistetään huijarisyndroomaan, mutta ne eivät ole sama asia. ”Monet perfektionistit päätyvät matalasti suoriutuviksi, koska he valitsevat vähemmän haastavia töitä joissa he voivat olla täydellisiä. Kontrastina huijarisyndroomasta kärsivät ovat perfektionisteja jotka ovat todistetusti menestyviä ja silti pitävät itseään huijarina.” Artikkelin mukaan toinen lapsien huijarisyndrooman syntymiseen vaikuttava tekijä on liiallinen kehuminen ilman yksityiskohtia. ”Olet fiksuin lapsi maailmassa.”, ”Olet koulusi paras matematiikassa.”, ”Olet taitavin piirtäjä ikinä.” Hyvää tarkoittavat vanhemmat antavat lapsilleen viestin, että he odottavat lapsilta aina näitä asetettuja oletusarvoja ja kaikki vähäisempi on epäonnistumista. (Psychology today, How to prevent impostor syndrome, 2019)

Miksi huijarisyndrooma on niin esillä erityisesti ohjelmistokehityksen parissa työskentelevillä? Yksi syy on varmasti ohjelmoinnin laajuus. Ohjelmointiin liittyviä töitä on lähes loputtomasti erilaisia ja kilpailu on kovaa. Työpaikkailmoituksissa on pitkiä listoja työntekijältä haluttavista taidoista, joista tosiasiassa vain kourallinen on vaatimuksena työn tekemiselle. Ohjelmistoala kehittyy joka päivä ja voi tuntua vaikealta pysyä muutoksen perässä. Tämä voi synnyttää monessa ihmisessä epävarmuuden ja riittämättömyyden tunnetta.

Huijarisyndroomasta yli pääseminen vaatii oman ajattelutavan muutosta omia kykyjään kohtaan. Ihmiset vertaavat itseään jatkuvasti muiden paljon menestyneempiin ihmisiin vaikka pitäisi keskittyä vain omien saavutusten ja taitojen mittaamiseen. Uutisartikkelit ja sosiaalinen media nostaa jatkuvasti pinnalle uskomattomia menestystarinoita ja erityislaatuisia visionäärejä, jolloin on helppo ajatella, ettei omat uran alussa olevat taidot tule ikinä yltämään sille tasolle, eikä todennäköisesti tulekaan. Osa huijarisyndroomasta irti pääsemisen prosessia on hyväksyä se, että jokainen voi tehdä vain henkilökohtaisen parhaansa.

3.4 Seurantaviikko 4

12.10.2020 maanantai

Sain viime perjantaina kaikki keskeneräiset työni valmiiksi, joten heti aamusta pystyin käyttämään aikaa uuden työkoneen käyttöönottoon. Minulle oli aluksi tilattu tehoiltaan todella alimitoitettu tietokone (8gb ram, intel core i5 jne.). Asennuksissa ei ole sinänsä mitään erikoista, mutta taas lyhyen ajan sisään puhtaalta pöydältä aloittaminen muistuttaa siitä, miten paljon hyödyllisiä lisäosia olen opiskelu- ja työurani aikana kerännyt Visual Studio Coden plug-in lataamosta. Sellaisia pieneltä kuulostavia asioita, kuten ”auto close tag”, ”auto rename tag”, Prettier formatointityökalu jne. Kaikki nämä ja lukuisat muut ovat olleet käytössäni jo niin pitkään, että ilman niitä työskentely tuntuu todella kömpelöltä. Tuntuu oudolta ajatella, että koulussa ensimmäisiä ohjelmointitehtäviä tehtiin notepadilla ja notepad++:lla. Vasta toisen vuoden puolivälissä tuli vastaan kurssi, jossa oli käytössä Atom, mutta sielläkään ei mainittu sanallakaan IDE lisäosia tai muutenkaan puhuttu ns. workflowsta ja sen parantamisesta.

Aamupalaverissa minulle annettiin tehtäväksi luoda portaalin dashboardin muillekin dataikkunoille koko näytön kokoinen valinnaisnäkymä (sellainen, jonka tein n. 2 viikkoa sitten dashboardin pylväsdiagrammille). Lähdin liikkeelle ympyrädiagrammi-dataikkunasta. Aluksi oli pohdittava miten eri diagrammeille toimitettava data oli parasta toimittaa full screen komponentille. Päädyin tilamuokkauksen sijaan passaamaan kaikki kolme datasettiä heti komponentin rederoinnin yhteydessä. Ratkaisu vaikutti yksinkertaisimmalta toteuttaa, sitä ei myöskään lasketa full screen komponentin avautuessa uudestaan, joka voisi aiheuttaa pitkähköjä latausaikoja laskettavan datan määrästä riippuen.

Seuraavaksi ongelmaksi muodostui se ettei ympyrädiagrammille luomani data sopinut nykyisellään full screen dialog komponentin taulukolle. Tässä en vielä päässyt ratkaisuun asti, mutta muutin aluksi palautettavan datan nykyisestä kahden objektin listasta yhdeksi objektiksi. Alunperin listalla oli siis kaksi objektia joilla oli ”name” ja ”value” atribuutit, yhdistin namen ja valuen kahdeksi key-value pariksi. Tämä vaikutti mielestäni yksinkertaisemmalta ja paremmalta tietorakenteelta.

13.10.2020 tiistai

Jatkoin tänään dataikkunoiden toteuttamista. Jätin eilen ympyrädiagrammin tekemisen kesken ja päätin tänään aloittaa janadiagrammin tekemisellä. Graafien saaminen isommalle näytölle tuntuu tällä kertaa jotenkin kankealta, vaikka yhden koko näytön diagrammin olen jo tehnyt (toki sekin vaatii vielä hiomista). Janadiagrammin error count data kääntyi helposti taulukkomuotoon, jonka jälkeen jätin diagramminäkymän odottamaan. Siirryin ympyrädiagrammin datan taulukoimiseen, joka eilen aiheutti oudon

paljon ongelmia. Tänään palautin tietomallin samanlaiseksi kuin aloittaessani ja sain taulukon näyttämään datan oikein ensimmäisellä yrittämällä.

Tämä on tuttu efekti ja toistuu lähes joka kerta, kun ongelman ratkominen jää kesken edellisenä iltapäivänä. Päivän päätteeksi alkaa olla väsynyt ja uupunut, jolloin ajatukset jäävät helposti pyörimään ympyrää. Pystyy ajattelemaan ongelmaa vain yhdeltä kantilta ja ratkaisu muuttuu täysin mahdottomaksi. Joskus lyhyt tauko riittää, mutta joskus on parempi jättää jatko seuraavalle päivälle, jolloin pääsee aloittamaan ikään, kuin puhtaalta pöydältä ja raktaisu löytyy erittäin nopeasti.

14.10.2020 keskiviikko

Dataikkunoiden viimeistelyn ja pienten ulkoasu hiomisten jälkeen sain tehtäväksi toteuttaa robotteja pyörittävien virtuaalikoneiden hallintaa suorittavan komponentin toiminnallisuus. Yksinkertainen datan haku kahdesta eri API endpointista, jonka jälkeen piti suodattaa vapaat koneet käytössä olevista. Päivän tunnit alkoivat olemaan täynnä, joten en päässyt kovinkaan paljoa suunnittelutyötä pidemmälle.

15.10.2020 torstai

Päivän tavoite: saa valmiiksi datan suodattava funktio.
Torstaina jatkoin virtuaalikone-tehtävän parissa työskentelyä. Algoritmien kirjoittaminen ei ole suurin vahvuuteni ja olen koittanut panostaa tämän taidon kehittämiseen. Uusi työpaikka ja koulutyöt rajoittavat kuitenkin ”ylimääräiseen” opiskeluun käytettävää aikaa ja Helsingin yliopiston tietorakenteet ja algoritmit kurssin käynti on jäänyt taka-alalle. Ymmärsin vaadittavan logiikan heti, mutta sen toteuttaminen koodissa vaati muutaman yrityksen ja jonkin verran aikaa.

Sain funktion kuitenkin valmiiksi, mutta sen palauttama data näkyi väärällä tavalla monivalintakomponentissa, jolle data annettiin. Otin yhteyden jälleen konsulttiin, joka totesi funktioni olevan periaatteessa ihan oikein, mutta suodatin valmiiksi liikaa tietoa. Monivalintakomponentti osaa itse jäsennellä datan vaadittavalla tavalla, joten funktio vaati muokkausta. Kirjoitimme konsultin kanssa funktion loppuun yhdessä ja lopputulos toimi halutulla tavalla.

16.10.2020 perjantai

Aamupalaverissa kävi ilmi, että en ollut huomannut eilen tehtyyn koodiin jäänyttä bugia. Virtuaalikoneet näkyivät oikein vain yhden työtehtävän kohdalla. Pitkän mietinnän ja tutkinnan jälkeen ongelmana oli se, ettei monivalintakomponentti ehtinyt saada dataa, vaan sille annettiin tyhjä array. Ongelma ratkesi lisäämällä monivalintakomponentin efekti hookin suoritusehtoihin valintavaihtoehtojen muutos, tällöin komponentti ei enää tyydy tyhjään listaan, vaan päivittyy kunhan kaikki asynkroniset tehtävät on suoritettu.

Viikkoanalyysi: Etätöiden hyödyt ja haitat

Robot House on pandemiatilanteen ulkopuolellakin täysin etätyöskentelyyn mahdollistava yritys. Sillä on kyllä toimisto Helsingissä, mutta suurin osa työntekijöistä työskentelee täysiaikaisesti etänä ja asuu muualla, kuin pääkaupunkiseudulla. Teknologia on mahdollistanut etätöiden tekemisen monella alalla, mutta silti ennen vuoden 2020 COVID-19 pandemiaa etätyön mahdollisuutta tarjottiin hyvin konservatiivisesti Suomessa, jopa ohjelmistokehityksen alalla. Koronatilanteen takia 59% Suomalaista työssäkäyvistä ihmisistä siirtyi etätöihin ja TTL:n kyselyn mukaan ”Puolet vastaajista haluaa ja myös pystyisi jatkamaan etätyöskentelyä kokonaan tai osittain myös jatkossa” (Työterveyslaitos, 2020). Etätyöt mahdollistavat paljon, mutta nämä mahdollisuudet eivät tule ilman varjopuolia.

Etätöiden ehkä selkein hyöty on työmatkojen puute. Kun toimisto on kotona, tarve pitkille (tai lyhyille) työmatkoille katoaa täysin. Suomalaisen työntekijän työmatkan keskimääräinen pituus vuonna 2018 oli 14 kilometriä ja työmatkojen pituus on puolitoistakertaistunut vuodesta 1990 (findikaattori.fi, 2018). Pääkaupunkiseudulla edestakainen työmatka kestää keskimäärin 54 minuuttia päivässä (Proliitto, 2013). Viikossa pelkkiin työmatkoihin siis käytetään keskimäärin 4,5 tuntia. Etätyöt siis säästäisivät työntekijän aikaa keskimäärin 4,5 tuntia viikossa, olettaen, että he ovat täysiaikaisesti etätöissä, kuten Robot Housella. Työnteko ei siis ole paikkasidonnaista, vaan työntekijä voi tehdä työnsä kotinsa lisäksi vaikka mökiltään tai jopa ulkomailta, kunhan paikasta löytyy toimiva internetyhteys. Konkreettisemmin sijainnin vapaus mahdollistaa asumisen huomattavasti halvemmalla alueella, kuin esimerkiksi Helsingissä työskentelevä ilman etätöitä voisi asua.

Toinen suuri hyöty etätöissä on työnteon joustavuus, työntekijä voi itse päättää missä ja mihin aikaan hän tekee työnsä, poislukien pakolliset kokoukset tai mahdolliset asiakastapaamiset. Työntekijä voi itse muodostaa itselleen toimivimman aikataulun, joka ei välttämättä ole 7,5h töitä yhteen putkeen. Moni henkilöistä, joiden kanssa olen keskustellut asiasta pitää työpäivästä, joka on jaettu kahteen osaan.

Esimerkiksi klo 8-12 töitä, klo 12-14 taukoa, jolloin he yleensä urheilevat tai käyvät ulkona ja klo 14-17:30 taas töitä. Pitkä tauko keskellä päivää auttaa heitä kognitiivisen uupumuksen sekä yleisen jaksamisen kanssa.

Suurimmat haitat etätyössä ovat mielestäni sosiaalisuuden puute sekä häiriötekijöiden välttely. Etätöissä ei mennä toimistolle, jossa normaalin työnteon yhteydessä näkee ja sosialisoi työkavereiden kanssa. Sosiaalisuus on työntekijän jaksamisen kanssa avainasemassa ja etätyötilanteessa on nähtävä erityisesti vaivaa tämän saavuttamiseksi. Robot House on koittanut panostaa etätyön sosiaalisuuteen asettamalla Microsoft Teams kalenteriin joka päivälle kaksi (vapaaehtoista) 15 minuuttia pitkää kahvitaukoa, joilla kannustetaan työntekijöitä viettämään taukoa yhdessä ja keskustelemaan kaikesta muusta, kuin töihin liittyvistä asioista. Kahvitauoille osallistumiseen kannustamisesta huolimatta työntekijän on itse oltava aktiivinen. Etätöiden tekeminen myös vaatii itsekuria, sillä kotoa löytyy yleensä paljon huomion varastajia. Kotona kynnys töihin keskittymiseen voi myös olla pienempi, kun ympäriltä puuttuu kolleegoiden luoma työilmapiiri.

3.5 Seurantaviikko 5

19.10.2020 maanantai

Päivän tehtäväksi muodostui käännöstyöt. i18next on npm paketti, jonka useTranslation hook tekee käyttöliittymän käännöstyöt erittäin helpoksi. Translate funktiolle passataan käännösavain, jotka ovat arvopareineen tallennettu sovelluksen tietokantaan. Itse käännöstöiden tekeminen on hyvin mekaanista ja itseään toistavaa: kaikki käyttöliittymään renderoitavat stringit on korvattava i18n:n t-funktiolla ja käännösavaimelle on annettava arvo tietokantaan.

20.10.2020 tiistai

Jatkoin käännöstöiden parissa myös tänäänkin.

21.10.2020 keskiviikko

Kävimme konsultin kanssa läpi uuden React-table komponentin käyttöä helpon paginaation lisäämiseksi pitkiin listoihin. Olen joskus aikaisemmin käyttänyt react-tablea,

mutta paketti on sittemmin päivitetty kirjoitushetkellä uusimpaan versionumeroon 7. Versio 7 poikkeaa hyvin merkittävästi aikaisemmin käyttämästäni versiosta. Koko react-table perustuu uusimmassa versiossa uuteen useTable hookiin, jonka avulla voidaan käyttää mitä tahansa jsx elementtiä tai esimerkiksi Material ui:n tauluelementtejä luomaan datanäkymiä.

22.10.2020 torstai

Jatkoin tänään react-table demon luontia. Sain myös ilmoituksen lomautuksesta, joka astuu voimaan heti marraskuun alussa. Kaikki käytännön asiat eivät ole kirjoitushetkellä selviä, sillä en ole ollut asiaan liittyen yhteydessä Academic Workiin.

Viikkoanalyysi: Kognitiivinen uupumus ja häiriötekijät etätöissä.

Uupumus on fysiologinen tila, jossa fyysinen tai henkinen kapasiteetti on alentunut. Uupumusta voi aiheuttaa mm. liiallinen työmäärä, unenpuute ja stressi (Saurabh Sarkar, 2015).
Moni ohjelmistokehittäjä törmää aika ajoin tilanteeseen, jossa yhtäkkiä päättely- ja keskittymiskyky heikkenee radikaalisti ja työn alla olevan pulman ratkaisu tuntuu täysin mahdottomalta. Tämä aiheutuu kognitiivisesta uupumuksesta tai tuttavallisemmin aivosumusta.

Ohjelmistotyön tekemisessä on välillä vaikea määritellä tehokas työntäyteinen työaika, sillä töiden tekeminen vaatii jatkuvaa ajattelua ja tehtävät asiat eivät ole pelkkiä konkreettisia rivejä koodissa. Kognitiivinen uupumus iskee usein työpäivän loppupuolella, jolloin yksinkertaisimmatkin sovelluksen staten hallintaan liittyvät asiat tuntuvat kuin vieraalta kieleltä. Kognitiivista uupumusta voi välttää pitämällä päivän aikana säännöllisiä taukoja, mielellään poissa tietokoneen ääreltä, jaloitellen tai jumppaillen. Tärkeintä on saada jotain täysin sen hetkiseen työhön liittymätöntä tekemistä, joka ns. alustaa aivot ja ajatukset, jolloin pulmanratkontaa pääsee jatkamaan uusin silmin.

3.6 Seurantaviikko 6

26.10.2020 maanantai

Päivän työksi valikoitui uusien ”collectioneiden” implementointi. Yksinkertaisesti ne ovat tietokantaan tallennettuja kokoelmia monivalintakentille annettavista arvoista. Koodissa ne tallennetaan custom hookin avulla muuttujaan ja otetaan käyttöön javascriptin map metodin avulla. Näiden tietokantaan tallennettavien collectioneiden etu kovakoodattuihin valintavaihtoehtoihin verrattuna on se, että niitä voidaan päivittää ilman uuden ohjelmistoversion julkaisua. Tällöin myös kuka tahansa yrityksen henkilö, jolla on admin oikeudet, pystyy päivittämään kokoelmia suoraan käyttöliittymästä.

27.10.2020 tiistai

Päivän tavoite: saa valmiiksi collectioneiden implementointi sekä käännökset.
Datataulu on edelleen tauolla, kunnes käännökset ja kokoelmat saadaan lisättyä. Työ on yksinkertaista ja mekaanista ja sainkin ne viimeisteltyä päivän aikana. Loppupäivästä palasin react tablen pariin. Tehtävä on aiheuttanut paljon päänvaivaa, muunmuassa react tablen v7:n ja typescriptin välisten pienten bugien takia. Lähdin työstämään datataulua tällä kertaa puhtaalta pöydältä uudessa komponentissa, mutta päädyin type-errorien ja muiden virheiden takia hylkäämään react table paketin ja siirryin käyttämään Material Ui:n Table komponenttia.

28.10.2020 keskiviikko

Päivän tavoite: saa datataulu valmiiksi.
Puhtaalta pöydältä aloittaminen selvästi selkeytti ajatukseni. Jonkin aikaa Material Ui pöytää kasattuani tulin siihen tulokseen, että on helpointa tehdä taulun tarvittavat muutokset suoraan portaalin code basesta löytyvään taulukomponenttiin. Alunperin ajattelin, ettei tehtäviä muutoksia tarvittaisi kuin vain yhdessä paikassa, jolloin kokonaan uusi komponentti olisi hyvä vaihtoehto. Asian kehittyessä totesin, että uudet ominaisuudet on kaikista helpoin lisätä olemassa olevaan koodiin, varsinkin nyt, kun päätin luopua react table paketin käytöstä. Työ sujui hyvin suoraviivaisesti loppuun ilman ongelmia. Hieman turhautumistakin aiheuttaneella pulmalla olikin hyvin helppo ratkaisu. Paginaatio oli helppo

lisätä vanhaan komponenttiin ja se tarvitsi vain uuden property arvon, joka kertoo milloin paginaatiota halutaan käyttää ja kuinka monta kohdetta yhdellä sivulla näkyy kerrallaan.

Taulukomponentti tarvitsee vielä hakukentän sekä lajittelutoiminnallisuuden, mutta ne toteutetaan vasta myöhemmin.

29.10.2020 torstai

Seuraavana tehtävänä oli kanban taulun mukaan dashboardin dataikkunoiden uusi minimointitoiminnallisuuden toteuttaminen. Dashboardilla on tällä hetkellä kolme eri dataikkunaa, mutta tulevaisuudessa niitä tulee olemaan N-määrä. Ikkunoiden määrän kasvaessa tilaa on säästettävä jotenkin, tähän ongelmaan vastaa uusi minimointitoiminto. Material Ui tarjoaa jälleen valmiin komponentin juuri tätä tehtävää varten: Collapse. Collapse komponentti toimii hyvin yksinkertaisesti: ensin tarvitaan uusi state kertomaan onko Collapse auki vai kiinni, seuraavaksi komponentin return metodissa määritellään mitä näytetään, kun Collapse on auki.

Toiminnallisuus tuli valmiiksi yhden päivän aikana ja huomenna siirryn seuraaviin hommiin. Minimoitujen ikkunoiden sijoittelu tulee todennäköisesti vaatimaan lisää työtä tulevaisuudessa.

30.10.2020 perjantai

Tänään ainoa työ oli Microsoft Power Apps työkaluun tutustuminen ja pienen käyttöliittymädemon luominen. Microsoft Power Apps on Microsoftin kehittämä ”visual coding” työkalu, jolla voidaan helposti luoda käyttöliittymiä tai yksinkertaisia ”sovelluksia”. Power Appsilla tehty sovellus voisi olla esimerkiksi lomake yhdistettynä tietokantaan Excel muodossa. Power Appsin suurin etu on sen helppo integrointi Microsoft Teamsiin.

3.7 Seurantaviikko 7

2.11.2020 maanantai

Power Apps jätettiin tänään sivuun ja palasin portaalin kehityksen pariin. Uusien asiakkuuksien lisäystä varten tarvittiin uusi lomake. Helpolta kuulostava tehtävä, mutta en ole vielä syventynyt code basesta löytyvään lomakekomponenttiin kovin tarkasti. Olen tehnyt pieniä visuaalisia muutoksia, mutta en ole perehtynyt komponentin toimintaan tai tarkemmin edes Formik kirjastoon, johon lomake perustuu. Aloitin työskentelyn tutustumalla Formikin omaan dokumentaatioon, sekä youtube videoon, jossa luotiin juuri Formikin ja Material Ui:n avulla uusi lomake. Tein uuden komponentin, joka hoiti lomakkeen renderöinnin Dialog komponentin sisälle. Sovellus compilasi oikein, mutta ei renderöinyt muuta kuin Dialog komponentin varjon. Pitkän pohdinnan ja ratkaisujen kokeilun jälkeen kävi ilmi, että valinnaiseksi propiksi merkitty validation schema puuttui ja sen lisättyäni lomake lähti toimimaan. Huomiselle jää visuaalisen ilmeen hiominen.

3.11.2020 tiistai

Päivä alkoi bugi-ilmoituksella joka piti saada korjattua niin pian kuin mahdollista. Bugin aiheutti datataulukomponenttiin tekemäni paginaatio ja sille annettava rivimäärän määräävä property. Kun paginaatio halutaan ottaa käyttöön, pitää antaa rivimäärä sivua kohti, sekä itse showPagination = true. Kun paginaatiota ei käytetä rivimäärä sivua kohden on vakio 15, tämän johdosta ilman paginaatiota pidemmät kuin 15 kohdetta sisältävät listat eivät näy kokonaan. Bugi pääsi syntymään epähuomioidessa mahdollisia käyttötapauksia. Se oli helppo korjata asettamalla vakiopituudeksi datasetin pituus, tällöin koko lista mahtuu yhdelle sivulle ilman paginaatiota.

Seuraavaksi viimeistelin eilen aloittamani asiakaslomakkeen, jonka jälkeen pääsin työstämään samankaltaista lomaketta työvaiheiden luontia varten.

3.8 Seurantaviikko 8

9.11. – 11.11.2020 maanantai-keskiviikko

Sairasloma

3.9 Seurantaviikko 9

16.11.2020 maanantai

Palasin töihin ensimmäistä kertaa sairasloman jälkeen ja aamu alkoi palaverilla. Sain työkseni toteuttaa jälleen uusi dataikkuna. Konsultti oli viime viikon aikana päivittänyt rajapintaa, jotta uuden mallisen datan hakeminen onnistuu. Rajapinnassa kuitenkin paljastui pieniä ongelmia, eikä data ollutkaan käyttötarkoitukseen sopivaa. Pidimme iltapäivällä uuden palaverin designerin ja rajapintakehittäjän kanssa. Tulimme siihen tulokseen, että tehtävä jätetään odottamaan jatkoa toistaiseksi.

17.11.2020 tiistai

Tänään tehtäväksi valikoitui eilistä tehtävää vastaava työ. Uudella rajapinnalla pystyy palauttamaan kaksi uutta atribuuttia joita tarvitaan uusia dataikkunoita varten. Myös tämän työn tekemiseksi tarvitaan muutos rajapintaan. Ongelmien vuoksi sain varata tunnin pituisen konsultoinnin konsultilta.

18.11.2020 keskiviikko

Tunnin konsultoinnin aikana keskustelimme edellisten päivien ongelmista, niiden syistä ja potentiaalisista ratkaisuista. Konsultin mielestä nyt pitäisi pysähtyä suunnittelemaan tarkasti ennen uusia implementointeja. Rajapintamuutokset ovat hänen mukaansa todennäköisesti välttämättömiä sillä nykyisellä tilallaan suunnitellut muutokset voitaisiin toteuttaa ainoastaan hyvin kömpelöllä tavalla. Lopullista vastausta ei tänään vielä saatu pääsuunnittelijan hyvin kiireisen päivän vuoksi.

3.10 Seurantaviikko 10

23.11.2020 maanantai

Viikko alkoi viime viikolla aloitettujen dataikkunoiden työstämisen kanssa. Toistaiseksi työ tehdään vain yhdelle valinnalle kerrallaan rajapinnan rajoitusten vuoksi. Tehtävä on hyvin suoraviivainen, eikä siinä ole mitään uutta. Redux storeen tallennettua dataa piti taas muotoilla oikeanlaiseksi, jotta se voidaan antaa taulukkokomponentille piirrettäväksi. Graafi ei tämän datasetin kanssa ole tarpeellinen, joten se jätetään käyttöliittymästä kokonaan pois.

4 Pohdinta ja päätelmät

Seurantajakso tuntui kuluneen todella nopeasti ja jakson aikana on tapahtunut todella paljon. Olen oppinut paljon, erityisesti typescriptin saralla. Täysi uppoutuminen typescriptin käyttöön tuntui alkuun hyvin uhkaavalta ja suurelta haasteelta, mutta jälkikäteen ajateltuna se oli erittäin tehokas tapa oppia uusi taito. Sain oppimiseen toki tukea konsultilta ja osasin javascriptiä valmiiksi suhteellisen sujuvasti. Olen myös onnistunut saamaan koodipohjan haltuun suhteellisen hyvin. En ole päässyt vielä käsiksi sovelluksen Redux osuuteen, joten en osaa kommentoida asiaa siltä osin. Koodipohja sisältää vielä paljon asioita, joita en täysin ymmärrä ja joiden selittäminen siirtyy pitkälle tulevaisuuteen marraskuussa alkaneiden lomautusten vuoksi. Lähes täysin yksin työskentelyssä on ollut sekä hyviä, että huonoja puolia. Huonot puolet tähän mennessä ovat tähän mennessä liittyneet pääasiassa siihen, etten tunne sovellusta vielä riittävän hyvin tai siihen, ettei dokumentaatiota ole tarjolla vaan umpikujassa on turvauduttava apuun konsultilta, joka ei aina kiireiltään ehdi auttamaan (ja lomautuksen takia ei ole käytettävissä lainkaan). Tämän johdosta töiden tekeminen ja ratkaisujen löytäminen hidastuu joissain tapauksissa huomattavan paljon. Kontrasti on hyvin suuri tämän työn ja kevään harjoittelun kanssa, jonka frontend tiimissä työskenteli minun ja toisen harjoittelijan lisäksi kaksi seniorityöntekijää, jotka tunsivat kehitettävän sovelluksen läpikotaisin. Hyvä puoli on erityisesti se, että asioita on pakko saada tehdyksi itse. Typescriptin opettelu on konkreettinen esimerkki tästä.

Projekti jonka työstämiseen minut palkattiin on hyvin alussa, joten työskentely ei ole ollut ihan niin järjestelmällistä verrattuna kevään harjoitteluuni. Kaikki on vasta asettumassa paikoilleen ja muotoutumassa oikeanlaiseksi. En vielä tämän seurannan aikana päässyt vaikuttamaan prosessien suunnitteluun tai edes isommin itse sovelluksen suunnitteluun, mutta tulevaisuudessa minulla tulee olemaan vaikutusvaltaa näissäkin asioissa. Tämä marraskuussa alkanut lomautus on tietenkin vaikuttanut suuresti työskentelyyn sekä sen määrän, että sisällön suhteen. Myös etätöiden vaikutus työntekoon ja viihtyvyyteen on yllättänyt itseni. Kun täysiaikaiset etätyöt ovat yrityksen normaalitila, vaatii kolleegoiden kanssa sosialisointi erityistä panostusta ja aktiivista osallistumista. Normaalissa toimistoympäristössä työkavereiden välisiä kontakteja on mahdotonta välttää. Kevään harjoitteluni oli myös koronaepidemian takia suurimmalta osin täysin etätöinä tehtävää. Harjoittelujakson alussa ehti kuitenkin olemaan toimisto-oloissa kaksi kuukautta, jonka aikana ehti tutustumaan suurimpaan osaan yrityksen työntekijöistä.

Kaikenkaikkiaan olen oppinut jälleen paljon ja päässyt tekemään koulutustani vastaavia töitä mielenkiintoisen projektin parissa. Aika näyttää miten paljon luvattua vaikutusvaltaa projektin ja työnkuvan suhteen pääsen käyttämään. Odotan innolla, että lomautus loppuu ja pääsen taas työskentelemään täysipäiväisesti.

Lähteet

Findikaattori. Työmatkan pituus. luettavissa: https://findikaattori.fi/fi/70 Luettu: 9.11.2020

Goodliffe Pete, 2014. Becoming a better programmer: a handbook for people who care about code. O'Reilly.

Imposter syndrome. luettavissa: https://www.psychologytoday.com/intl/basics/imposter-syndrome Luettu: 24.11.2020

Martin Robert, 2008. Clean Code. Pearson.

NPM. Dependency hell. luettavissa: https://npm.github.io/how-npm-works-docs/theory-and-design/dependency-hell.html Luettu: 11.11.2020

Npmjs.com. luettavissa: https://www.npmjs.com/ Luettu: 18.10.2020

Proliitto. Suomalainen työmatka kestää keskimäärin 46 minuuttia. luettavissa: https://www.proliitto.fi/uutiset/tyomarkkinat/suomalainen-tyomatka-kestaa-keskimaarin-46-minuuttia Luettu: 9.11.2020

Psychology today. How to prevent impostor syndrome. luettavissa: https://www.psychologytoday.com/us/blog/shouldstorm/201910/how-prevent-impostor-syndrome-in-your-child Luettu: 24.11.2020

Recharts. luettavissa: https://recharts.org/en-US/api Luettu: 11.11.2020

Saurabh, S. 2015. Investigation of the Effects of Mental Fatigue on Programming Tasks. North Carolina State University. Luettavissa: https://repository.lib.ncsu.edu/bitstream/handle/1840.16/10358/etd.pdf?sequence=1

TTL. Miljoona suomalaista loikkasi etätöihin. luettavissa: https://www.ttl.fi/miljoona-suomalaista-loikkasi-etatoihin/ Luettu: 9.11.2020