Käyttöliittymäkehitys moderneilla web-teknologioilla

Oskari Holopainen

Haaga-Helia ammattikorkeakoulu

Amk-opinnäytetyö

2021

Tradenomin tutkinto

Tiivistelmä

Tekijä(t)
Oskari Holopainen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Käyttöliittymäkehitys moderneilla web-teknologioilla

Sivu- ja liitesivumäärä
49 + 2

Tämä opinnäytetyö kuvaa frontend-kehittäjän arkea palvelutuoteyrityksessä, joka on osa isompaa konsernia. Työssä käsitellään opintojen ja työelämän väliseen siirtymävaiheeseen kuuluvia kehitysaskeleita, mahdollisia haasteita ja oivalluksia. Lisäksi pohditaan erilaisia toimintamalleja ohjelmointiin liittyen yleisesti ja tutustutaan uusiin teknologioihin.

Alituisesti muuttuvassa maailmassa ohjelmistokehittäjiltä edellytetään nopeaa oppimiskykyä ja tiedonhakutaitoja. Oma-aloitteisuuden, motivaation, viestintätaitojen ja yleisten sosiaalisten taitojen merkitys korostuu projektien ollessa niin ratkaisu tosielämän ongelmaan kuin myös ihmisten välisen vuorovaikutuksen tulos.

Keskeisiä työn havaintoja ovat yleisesti tunnistetut ohjelmoinnin hyvät käytännöt, deklaratiivisen React-ohjelmoinnin erilaiset vivahteet, testauksen ja laadunvarmistamisen merkitys ja projektinhallinnalliset näkökulmat suunnitteluvaiheesta toteutukseen. Työn aikana käydään läpi erilaisia projekteja, junior-tason kehittäjän rooliin kuuluvia vaatimuksia ja vastuualueita, henkilökohtaista ammatillista kasvua ja ammatti-identiteetin muodostumista. Opinnäytetyö on toteutettu päiväkirjamuotoisena ja työssä seurataan vuorovaikutusta kehittäjän ja muiden organisaation jäsenten välillä päivittäisellä tasolla.

Asiasanat
Ohjelmistokehitys, web-sovelluskehitys, työssäoppiminen, asiantuntijatyö

Sisällys

1 Johdanto

Tämän opinnäytetyön tavoitteena on kuvata päivittäisellä tasolla tämänhetkistä työnkuvaani frontend-kehittäjänä ja kehittymistäni ammatissa. Työ on päiväkirjamuotoinen, seurantaviikkoja on kymmenen ja ne ajoittuvat aikavälille 06.09.2021 – 15.11.2021.

1.1 Tietoperusta

Tietoperustana työtehtävissäni tarvitsen ymmärrystä ja osaamista useista web-sovellusten käyttöliittymien kehityksessä käytettävistä teknologioista, palvelinteknologioista ja konttiteknologioista. Edellämainittujen lisäksi hyödyksi ovat erilaiset hyväksi todetut käytännöt siistin koodin kirjoittamiseen, luettavuuteen ja ylläpidettävyyteen liittyen. Käytän kirjallisina tiedonlähteinä käyttämiini teknologioihin liittyvää alan ammattikirjallisuutta sekä niiden dokumentaatioita. Ammattikirjallisuudesta käytän mm. Robert Martinin teoksia the Clean Coder (2011) ja Clean Code (2008), Carlos Santana Roldánin React 17 Design Patterns and Best Practices (2021) ja Wilbert O. Galitzin The Essential Guide to User Interface Design. Kirjallisuuden lisäksi käytän verkossa julkisesti saatavilla olevia alan artikkeleita. Reactiin ja JavaScript-kehitykseen liittyvää laadukastakin ajankohtaista aineistoa julkaistaan paljon mm. Mediumissa. Lähdekritiikki on kuitenkin pidettävä mielessä Mediumin ollessa avoin alusta. Jatkuvasti kehittyvässä maailmassa usein ilmenee myös uusia vähemmän kartoitettuja ongelmia tai asioita, joista ei ole ehditty luoda kirjallisuutta. Näissä tapauksissa tiedonetsintätaidot, hakukoneet ja aktiiviset kehittäjäyhteisöt ovat hyödyllisiä, jolloin vastauksia löytyy usein joko Githubin tai Stackoverflow:n keskustelupalstoilta. Usein kuitenkin kannattaa tarkistaa, löytyykö aiheesta virallista dokumentaatiota.

1.2 Työympäristö ja osaamisvaatimukset

Työnantajani on pk-yritys, jonka tuoteperhe tarjoaa älykästä automaatiota, asiakaspalveluratkaisuja sekä erilaisia raportoinnin työkaluja tukemaan ja tehostamaan asiakasyritysten liiketoimintaa. Tuoteperhe koostuu kolmesta palvelutuotteesta, joista kukin on suunniteltu kattamaan yhden edellä mainituista osa-alueista.

Tehtäväni on toimia sovelluskehittäjänä ja vastaan tällä hetkellä frontend-kehityksestä useissa työnantajani tuotekehitysprojekteissa. Tällä hetkellä olen mukana sellaisissa projekteissa jotka liittyvät pääasiassa edellämainituista palvelutuotteista keskimmäiseen, joka on digitaalinen ratkaisu keskitettyyn asiakaspalveluun ja puhelinmyyntiin. Tällä hetkellä se on työnantajani keskeisimpiä palvelutuotteita ja asiakaspalvelun sekä myynnin lisäksi se tarjoaa erilaisia raportoinnin ratkaisuja asiakkaille. Olkoon palvelun nimi kokonaisuudessaan **Customerc.**

Työtehtäväni vaativat useiden sovelluskehityksen osa-alueiden ja työkalujen hallintaa ja käyttöä, luovaa ongelmanratkaisukykyä, viestintä- ja tiedonhakutaitoja ja ennen kaikkea kykyä omaksua uusia asioita.

Projektityö vaatii oma-aloitteisuutta ja itsenäistä kykyä ongelmanratkaisuun. Viestintätaitojen merkitys tuntuu korostuvan monissa tilanteissa, sillä projekteja toteutetaan tiimeinä ja viestin täytyy kulkea moneen suuntaan, jotta ihmiset pysyvät ajan tasalla. Kommunikaatiokatkokset johtavat helposti ongelmatilanteisiin. Työ vaatii myös arvostelukykyä teknisten ratkaisujen osalta ja asiantuntijuutta teknisiltä osa-alueilta.

Keskeisin tekninen osaamisvaatimus on tällä hetkellä Facebookin kehittämä React-kirjasto, sekä muut frontend-kehityksen osa-alueet, kuten HTML ja CSS. Ohjelmointikielenä React-sovelluksissa käytetään kehittäjien keskuudessa suosittua Typescriptiä. Kaikissa React-sovelluksissa on tähän asti käytetty keskitettyyn tilanhallintaan Facebookin Flux -arkkitehtuuriin perustuvaa Redux-kirjastoa. Käytettävien teknologioiden valinnassa minulta odotetaan kykyä muodostaa perusteltuja mielipiteitä ja kehittäjänä vastuullani on selvittää parhaiten kuhunkin tilanteeseen sopivat työkalut. Vallitsevan koronatilanteen takia teen töitä tällä hetkellä pääsääntöisesti etätyönä.

1.3 Keskeiset käsitteet

Opinnäytetyö sisältää teknisiä termejä ja käsitteitä joista monet pohjautuu englanninkieliseen sanastoon. Keskeisimpiä teknisiä termejä ja käsitteitä on kuvattu opinnäytetyön *liitteessä 1.* Työnantajani järjestelmät ovat myös moniosaisia ja sisältävät paljon erilaisia järjestelmän sisäiseen toimintaan liittyviä käsitteitä ja abstraktioita. Monet näistä nimityksistä ovat johdettu englanninkielisestä puhelinkeskuksiin liittyvästä sanastosta, kuten *Agent* tai *Skill.* Kaikille termeille ei ole kuvaavaa suomen kielen käännöstä, jolloin joudun käyttämään anglismeja. Projektien ja järjestelmän osien nimet on tietosuojasyistä kuvattu peitenimillä, jotka ovat esittelyvaiheessa lihavoituja.

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työn analyysi

Työnkuvaani kuuluu etupäässä Reactilla ja TypeScriptillä toteutettujen selainpohjaisten käyttöliittymien vaatimusmäärittely, suunnittelu, toteutus ja testaus.
Teen töitä asiakasprojektissa, jossa kehitämme asiakkaalle telemarkkinointijärjestelmää ulossoittokampanjoihin. Olkoon asiakkaan nimi **Upwire.** Projektin nimi olkoon **AgentUP**. Projekti koostuu kahdesta käyttöliittymästä ja yhdestä rajapinnasta. Käyttöliittymistä toinen on tarkoitettu hallintapaneeliksi ylemmille toimihenkilöille tai tiiminvetäjille kampanjoiden sekä niissä työskentelevien asiakaspalvelijoiden hallintaan ja raportointiin ja toinen asiakaspalvelijoiden käyttöön. Kutsuttakoon tiiminvetäjien hallintapaneelia nimellä **Hallinta UI** ja asiakaspalvelijoiden käyttöliittymää nimellä **Aspa UI**. Tehtävänäni on suunnitella, toteuttaa ja testata näihin käyttöliittymiin uusia ominaisuuksia. Tarvittaessa vien tässä projektissa muutoksia myös tuotantopalvelimelle asiakasympäristöön.

Toinen projekti, jossa olen mukana tällä hetkellä on keskitettyyn järjestelmänhallintaan tarkoitettu käyttöliittymä, jolla hallitaan työnantajani puhelinjärjestelmiin liittyvää puhealustaa. Olkoon tämän projektin nimi **Centry**. Puhealustan nimi taas on **Speakex**. Käyttöliittymää on tarkoitus uudistaa ja tuotokset suunnataan sekä työnantajan sisäiseen käyttöön, että asiakkaille. Joitain toiminnallisuuksia tulee piilottaa asiakkaalta, jotta ei tapahtuisi turhaa vahinkoa. Käyttöliittymästä on syntynyt vuosien varrella erilaisia työpöytäversioita ja sitä olisi tarkoitus uudistaa niin, että käyttökokemus kohentuisi ja manuaalista työtä voitaisiin vähentää. Uudistaminen tapahtuu tekemällä modernimpi selainpohjainen käyttöliittymä, jota olen jo hieman aloittanut viime kesänä ja sitä olisi tarkoitus laajentaa Speakex-puhealustan muihin osiin.

Työni vaatii osaamista selainpohjaisten käyttöliittymien suunnitteluun ja toteutukseen sekä sen lisäksi aloitekykyä, ongelmanratkaisutaitoja sekä vahvaa ohjelmointirutiinia. Pelkästään työnantajan järjestelmien hahmottamiseen tarvitaan myös paljon tietoa sen osista, toiminnallisuuksista ja monimutkaisista kokonaisuuksista. Järjestelmät sisältävät paljon erilaisia televiestintään liittyviä teknisiä osia, joiden ymmärrys puolestaan vaatii tietämystä puhelinjärjestelmistä ja alan termistöstä. Puhelinjärjestelmistä ja televiestinnästä oma tietämykseni on hyvin vähäistä, mutta uskon, että opin ainakin käyttöliittymien kehittämisen näkökulmasta välttämättömiä asioita.

2.2 Osaamisvaatimukset ja oma osaaminen

Työnantaja on ottanut huomioon kokemukseni ja osaamiseni antaessaan työtehtäviä minulle. Työnantajani ja projektipäälliköiden antaman suoran palautteen perusteella olen suoriutunut mielestäni tähänastisista työtehtävistäni kiitettävästi. Olen tehnyt parhaani, ollut mielestäni ahkera ja omatoiminen ja positiivista palautetta on tullut sekä esimieheltä, projektipäälliköiltä ja muilta kollegoilta. Osaamisvaatimukset ja haasteet ovat myös asteittain korkeampia ja työtehtävät vaikeampia. Tämä johtuu normaalisti muuttuvista vaatimuksista joiden mukana tulee aina jotakin itselle uutta. Jos on jotain, mitä en usko pystyväni tekemään pyrin tunnistamaan sen ajoissa ja ilmaisemaan asian suoraan, jotta vältytään väärinkäsityksiltä. Pyrin myös jatkuvasti kehittymään ja kokeilemaan uusia toimintatapoja löytääkseni hyviä ratkaisumalleja. Olen myös yrittänyt olla avuksi muille parhaani mukaan ja auttanut heitä ongelmatilanteissa. Osassa töihini liittyvistä asioista olen hyvinkin vahvoilla, kuten esimerkiksi käyttöliittymien toteutuspuolessa. Muissa asioissa, kuten televiestintäteknologioihin liittyvissä asioissa, tietokantaosaamisessa tai suunnittelutyössä koen olevani vielä aloitteleva toimija.

2.3 Tähänastinen kehittyminen

Olen oppinut tähän asti paljon ja monessa asiassa olen kehittynyt huomattavasti. Välillä huomaan yrittäväni ehkä haalia tietoa asioista liikaakin, jolloin keskittyminen saattaa herpaantua oleellisesta. Kenenkään ei kuitenkaan tarvitse opetella kaikkea eikä se ole mahdollistakaan. Tekemäni työn laatu on harjoittelun aikana parantunut. Palatessani uudelleen jo aiemmin työstämieni koodinpätkien pariin olen huomannut, kuinka ratkaisuistani on tullut siistimpiä, luettavampia ja helpompia ylläpitää. Oppimisen seurauksena myös työn tehokkuus on kasvanut huomattavasti. Asioiden tekemiseen, johon puoli vuotta sitten kului tunteja, kuluu parhaassa tapauksessa tällä hetkellä joitakin kymmeniä minuutteja.

Etätyöskentely on pakottanut miettimään, kuinka saan viestini välitettyä erilaiselle ihmiselle mahdollisimman tehokkaasti. Tässä olen joutunut miettimään viestintätaitojani ja sitä, miten niitä voisi kehittää. Kuvakaappaukset, dokumentit ja videot ovat teknisesti monimutkaisissa tapauksissa ollut tehokkaampia viestinnän keinoja kuin sanat. Etätyö on antanut myös ihmisiin tutustumiseen oman lisähaasteensa. Vaatii enemmän aikaa chatin tai puhelujen välityksellä oppia tuntemaan jotakuta toista ja se, kuinka hyvin onnistun puhumaan jonkun kanssa ”samaa kieltä” on ollut haastavampaa kuin kasvotusten. Olosuhteet ovat pakottaneet kehittymään näissä asioissa.

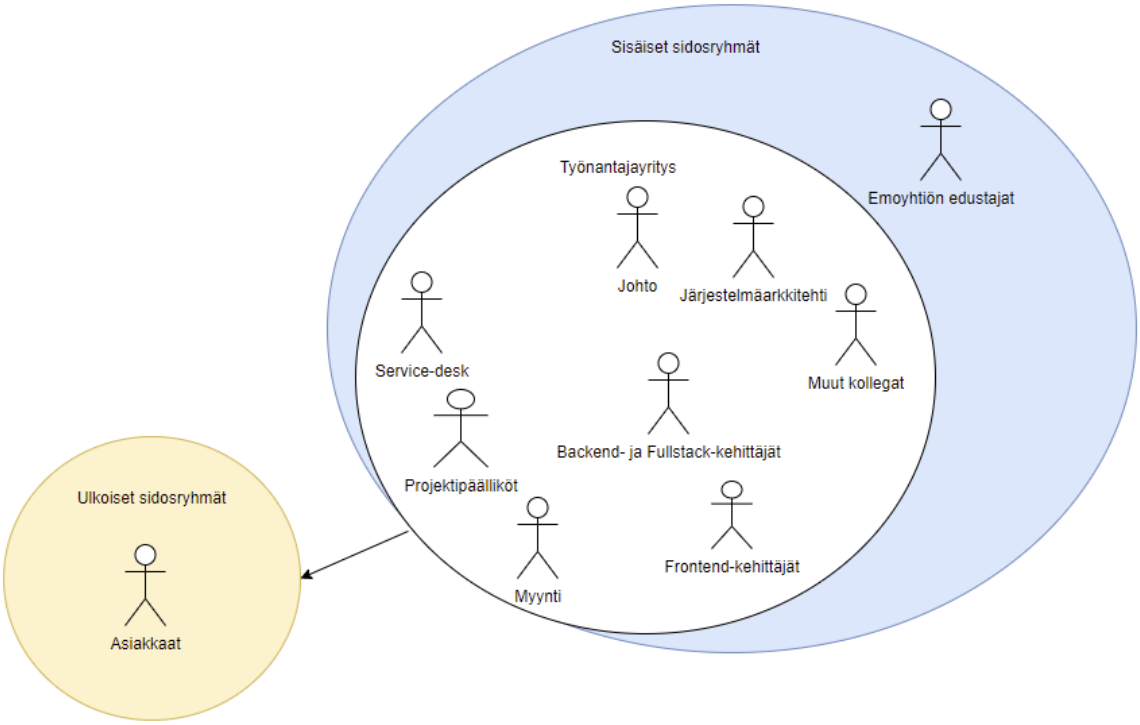
Jatkossa mielestäni minun tulee panostaa suunnittelijana kehittymiseen, teknisten osaamisalueitteni syventämiseen ja laajentamiseen sekä viestintään. Usein projektit ovat

tiimitöitä ja erilaisten ihmisten kanssa toimeen tuleminen ja viestintä on niissä avainasemassa. Opintojenikin aikana on painotettu viestinnän merkitystä paljon ja käytännön projekteissa olen saanut huomata miksi.

2.4 Sidosryhmät työpaikalla

Keskeisimpiä ulkoisia sidosryhmiä työpaikallani ovat alalle tyypillisesti asiakkaat. Sisäisistä sidosryhmistä keskeisimpiä minulle ovat projektipäälliköt, muut sovelluskehittäjät, järjestelmäarkkitehti. Muita keskeisiä työnantajayritykseni sidosryhmiä ovat johdon edustajat kuten palvelujohtaja (esimieheni), teknologiajohtaja, operatiivinen johtaja sekä toimitusjohtaja. Keskeisiä sisäisiä sidosryhmiä ovat myös työnantajayritykseni emoyhtiön osapuolet, jotka järjestävät erilaisia koulutuksia, luentoja ja webinaareja konsernin jäsenyritysten kesken.

Kuva 1: Sidosryhmät



Keskeisimpiä sidosryhmien intressejä asiakasprojekteissa on asiakasyrityksen edustajien tyytyväisyys asiakkaiden edustaessa myös loppukäyttäjien tarpeita. Palveluja räätälöidään asiakkaiden tarpeet huomioon ottaen ja projektien onnistumisen kannalta on tärkeää pitää asiakkaat tyytyväisinä.

Toinen tärkeä sidosryhmä on projektipäälliköt. Raportoin projektien edistymisestä säännöllisesti niiden projektien projektipäälliköille, joissa olen mukana. Projektipäälliköt raportoivat myös yrityksen johdolle ja esimiehille. Jotta osaamiseni tunnistettaisiin johdossa, on

hyvä pitää myös projektipäälliköt tyytyväisinä. Kolmas keskeinen sidosryhmä on muut kehittäjät. Monet kehittäjistä keskittyvät joko frontend- tai backend-kehitykseen, jolloin viestinnän merkitys osapuolten välillä korostuu hieman enemmän kuin esimerkiksi silloin, jos kaikki olisivat fullstack-kehittäjiä.

2.5 Vuorovaikutus työpaikalla

Työ tapahtuu tällä hetkellä pääosin etätyönä. Viestintä työtovereiden kanssa tapahtuu pääosin Microsoft Teamsin kautta. Viestintään sisäisten sidosryhmien välillä kuuluu viikoittaisia tiimipalavereja, päivittäisiä tuotekehityspalavereja tai isompia säännöllisin väliajoin pidettäviä henkilöstöpalavereja.

Päivittäisissä palavereissa minun odotetaan raportoivan senhetkinen tilanne projektieni ja työtehtävieni osalta ja viestivän mahdollisista haasteista tai onnistumisista. Joskus tilanteeseen liittyvät teknisemmät asiat täytyy muotoilla helpommin lähestyttäviksi riippuen osallisista henkilöistä. Jos projektissa ollaan ottamassa uutta teknologiaa käyttöön, minulta odotetaan mahdollisesti siitä perusteltavissa olevaa mielipidettä. Käyttöliittymiin liittyvissä asioissa vastaan parhaan osaamiseni mukaan ja annan näkökulmani asioihin.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 1: 06.-10.09.2021

Maanantai 06.09.2021
Maanantaisin käymme työpaikallani viikkopalaverissa läpi edellisen viikon tapahtumat tiimin kesken, alkavan viikon suunnitelmia sekä tämän hetken tilannetta yleisesti. AgentUP-projektissa on tällä hetkellä käytössä myös päivittäiset palaverit, joissa käydään läpi päivittäiseen projektityöhön liittyviä asioita.

Kuten aiemmin mainitsin, AgentUP-projektiin liittyy kaksi eri käyttöliittymää: Hallinta UI sekä Aspa UI. Edellisellä viikolla olin toteuttanut dynaamiset konfiguraatiot Hallinta UI:lle, jotta Docker-perustaisissa toimituksissa sovelluksen voi konfiguroida antamalla käyttöliittymän asetukset erillisestä JSON-tiedostosta mahdollistaen suoraviivaisemman toimituksen uusiin ympäristöihin ilman, että käyttöliittymää tarvitsee rakentaa (engl. build) jokaista ympäristöä varten uudestaan. Aiempi toteutus konfiguroi sovelluksen buildausvaiheessa ympäristömuuttujiin talletetun tiedon perusteella. Tämän päivän aikana tein samankaltaiset konfiguraatiomuutokset myös Centry:n käyttöliittymään sekä Aspa UI:lle.

Toinen tehtävä muutos oli tuki käyttöliittymien hostaamiselle käänteisen välityspalvelimen takaa. Koska työnantajan Docker-perustaiset toimitukset tulee asentaa käänteisen välityspalvelimen taakse ja tuleva HTTP-liikenne ohjataan oikealle kontille osoitteen sisältämän etuliitteen perusteella, Reactin oletuskonfiguraatiot eivät toimi, koska index.html sisältää oletuksena absoluuttiset polut staattisiin tiedostoihin. Yritin ratkaista tätä osittain dokumentaation mukaan määrittelemällä Reactin juuripoluksi ”.”, jolloin index.html etsii staattisia tiedostoja suhteellisen polun alta absoluuttisen sijaan. Tämän kanssa react-router:in BrowserRouter-komponentti aiheutti ongelmia. Käynnistettäessä ohjelma toimi normaalisti ja edettäessä vain juuritason reiteillä kaikki toimi hyvin, mutta edettäessä sovelluksen reiteissä juuritasoa syvemmälle ja päivitettäessä sivua, ohjelma hajoaa sillä index.html etsii staattisia tiedostoja väärästä polusta. Päätin palata asiaan seuraavana päivänä.

Tiistai 07.09.2021
Tänään jatkoin AgentUP:n parissa ja vaihdoin react-routerin käyttämään nk. hash-routeja, joka tarkoittaa, että käyttöliittymä erottelee omat polkunsa http-pyynnöistä ”#”-merkillä, jolloin käyttöliittymän ei tarvitse välittää, #-merkkiä ennen tulevasta osiosta. Tämä ratkaisi aiemmat ongelmat ja sivujen uudelleenlatauksen suhteen. Create-react-app:in dokumentaatiossa oli kyllä asiasta mainittu ja se oli aiemmin tiedossa, mutta yritin turhaan löytää vaihtoehtoista tapaa, jossa ei tarvitsisi käyttää hash-routeja. Kesti aikaa selvittää, mistä ongelmat todella johtuivat ja se oli turhauttavaa, mutta uskon että se oli myös

opettavaista. Ainakin todistin omakohtaisesti, että kannattaa seurata dokumentaatioiden ehdotuksia, sillä niille on syynsä.

Iltapäivän aikana oli palaveri, jossa katsoimme hieman Centry-projektin tilannetta ja pohdittiin, miten sen suhteen voisi edetä ja mitä voisin ottaa työn alle siihen liittyen. Palaverissa päädyimme siihen, että aloitan dokumentoimalla tähänastista toteutusta ja sen ominaisuuksia, jonka jälkeen on helpompi määritellä, mitä ominaisuuksia uudistettuun käyttöliittymään tulee. Jatkan tutustumalla aiempien toteutusten hyödyntämien rajapintojen dokumentaatioihin.

Keskiviikko 08.09.2021
Päivän päätehtävänä ohjelmoinnin suhteen oli toteuttaa asiakkaalta (Upwire) tulleita muutoksia AgentUP-projektin käyttöliittymiin. Muutoksista osa oli ulkoasuun ja tyylittelyyn liittyviä ja osa taas uusia toiminnallisuuksia. Sain ulkoasumuutokset ja osan toiminnallisuuksista jotakuinkin valmiiksi aamupäivän aikana, joka tuotti päivittäisessä palaverissa kehuja projektipäälliköltä nopeasta toiminnasta.

Olin hiljattain ottanut käyttöön Hallinta UI:lle lomakkeita varten react-hook-forms-nimisen kirjaston ja se on mielestäni tehnyt lomakkeiden luomisesta mukavan helppoa. Reactin toiminnan ja sen deklaratiivisen luonteen vuoksi lomakkeiden luominen ei ole aina triviaalia (Bartoli 2017, 126). Käsin lomakkeiden teko on osoittautunut työlääksi, etenkin jos lomakkeet ovat vähänkin monimutkaisempia. Lomakkeiden tekoon suunniteltu kirjasto hoitaa monta asiaa, kuten tapahtumienkäsittelyn, tilanhallinnan, validaation ja virheidenkäsittelyn konepellin alla helpottaen huomattavasti omaa elämääni kirjoitetun koodin määrässä, skaalautuvuudessa ja koodin ylläpidettävyydessä. Luomisen lisäksi myös muutoksien teko lomakkeisiin oli mukavaa ja helppoa peruskomponenttien ollessa paikallaan.

Torstai 09.09.2021
Tänään jatkoin Upwiren pyytämiä muutoksia AgentUP-projektin Aspa UI:lla, jossa yksi aiemmista näkymistä täytyy siirtää omaksi dialogikseen, joka näytetään asiakaspalvelijalle työn sulkemisen yhteydessä, mikäli siihen on asetettu lupa. Näkymässä asiakaspalvelija voi tarkastella ja muokata asiakkaalle lähetettävää viestiä ennen kuin se lähetetään. Näkymässä näkyvät viestit luodaan viestipohjien perusteella, joita voi luoda Hallinta UI:lla.

Aamupäivällä oli myös palaveri Centry-projektiin liittyen, jossa katselmoitiin projektin aiempaa tilaa ja sen ominaisuuksia. Sovellus oli toteutettu työpöytäsovelluksena aiemmin, kuten monet muutkin työnantajani sovellukset. Kokonaisuus vaikutti olevan moniosainen ja laaja ja sisältävän paljon televiestintään ja työnantajan järjestelmiin liittyviä domain-spesifejä teknisiä termejä ja käsitteitä, joista en ole aiemmin kuullutkaan. Joitakin näistä ovat

muun muassa Trunk, VDN (Vector Directory Number), PSTN (Public Switched Telephone Network) ja Vektorit. Projektina Centry vaikuttaa myös kokonaisuutena monimutkaisemmalta ja vaikeimmalta hahmottaa, missä olen koskaan ollut mukana. Sain jo palaverissa lisää ymmärrystä siitä, mitä kaikkia eri osia järjestelmässä on, mutta kokonaisuuden hahmottamiseen menee vielä aikaa. Projekti sisältää paljon minulle uusia asioita ja hiljaista tietoa, joista ei ole paljoakaan dokumentaatiota, ainakaan sellaista, josta olisin tietoinen ja asioita jotka täytyy vain oppia.

Perjantai 10.09.2021

Tänään tein AgentUP-projektin Aspa UI:lla työn alla olleen muutoksen loppuun, jossa viestin esikatselunäkymä eriytettiin omaksi dialogikseen. Viestipohjaan täytyi voida määrittää myös tieto siitä, näytetäänkö esikatseluikkuna työntekijälle vai lähetetäänkö viesti automaattisesti. Aamupäivä sujui hieman takkuisesti, mutta päivän mittaan keskittyminen parani ja sain aikaiseksi asioita ihan kohtalaisen hyvin.

Iltapäivällä käytin runsaasti aikaa Hallinta-UI:n yhden osa-alueen refaktoroimiseen, jossa määritellään puheluille eri lopputulokset. Tehtävää oli koodin siistimisessä, komponenttien eriyttämisessä, tyypityksen parantamisessa ja yleisesti koodin parantelussa. Sieltä täältä löytyi vanhoja pätkiä, jotka on tehty *imperatiivista* ohjelmointiparadigmaa mukaillen, joka on vähemmän eleganttia ja vaikeammin ylläpidettävää, kuin *deklaratiivinen* ohjelmointi, jota React mukailee (Roldán 2021, luku 1.2). Muutin kyseiset kohdat mukailemaan deklaratiivista tapaa, joka vähensi ylläpidettävien arvojen määrää ja teki koodista helpommin luettavaa ja ylläpidettävää.

Viikkoanalyysi

Viikolla on ollut paljon tehtävää ja muistettavia asioita. Yksi kehittämiskohde työssäni tällä hetkellä voisi olla ajanhallinta ja huolellisuus. Usein huomaan tasapainottelevani aikaansaamisen ja laadun tai huolellisuuden kanssa. Ajanhallinnassa voisin kehittyä keskittymällä tekemään huolellisesti yhden asian valmiiksi ennen seuraavaan siirtymistä. Taukojen pitäminen unohtuu myös usein, jos yritän ratkaista ongelmaa, joka ei meinaa ratketa. Usein myös tuntuu helpommalta havaitessaan ongelmia ohittaa ne laiskasti etsimättä ja korjaamatta ongelman juurisyytä, joka usein kostautuu myöhemmin, kun joutuu myöhemmin palaamaan saman koodin pariin ylläpitämään sitä. Kokemukseni on osoittanut tähän mennessä, että mitä aiemmin ja mitä huolellisemmin huonosti ylläpidettävää koodia nähdessään sitä siistii, sen helpompi sen ääreen on myöhemmin palata.

Työtäni AgentUP-projektissa on hankaloittanut se, että olen hypännyt keskelle laajahkoa jo jonkin aikaa käynnissä ollutta projektia mukaan, jossa sain päävastuun käyttöliittymäkehityksen osalta pitkälti heti kun minut oli palkattu. Alkuperäinen vastuuhenkilö lähti tämän

jälkeen piakkoin muihin työtehtäviin. AgentUP-projektin Ongelmakohtien korjaaminen tarkoittaisi paikoitellen kokonaisvaltaisia muutoksia projektin komponenttirakenteessa ja koodityylissä. Olen parhaani mukaan pyrkinyt tekemään parannuksia niiden kohtien osalta, joita olen muokannut. Koko projektin siistiminen taas veisi todennäköisesti enemmän aikaa kuin sen kirjoittaminen kokonaan uudelleen.

Turhauttavin asia työssäni tähän asti on ollut jonkun toisen laiskuudesta tai huonosta ohjelmoinnista aiheutuneiden virheiden korjaaminen. Omien virheiden korjaamiseen on helpompaa suhtautua, koska luonnollisesti koen olevani omasta työstäni ja sen laadusta vastuussa. Toisten työ taas on mielestäni eri asia. Toisaalta olen myös joutunut miettimään monen asian kohdalla uudelleen omaa asennoitumistani ja olen myös huomannut, että kaikki osat eivät ole huonoja vaan projektissa on myös hyviäkin asioita. Muiden koodia lukiessani koodin lukukykyni on myös kehittynyt paljon ja toistenkin virheistä oppii. Eräs tähän liittyvä huomanneeni asia on nimeämisen vaikeus. Huonosti nimettyjä muuttujia on ulkopuolisen näkökulmasta nopeasti lukien käytännössä mahdoton ymmärtää järkevästi. Vastakohtana sille on kuvaavasti nimetty funktio tai muuttuja, jonka merkityksen ymmärtää miltei heti. Omaa koodia lukiessa tälle asialle helposti ikään kuin sokeutuu, koska nimet joita ohjelmoija muuttujille antaa täytyy keksiä itse. Tällöin ne luonnollisesti myös tuntuu itsestä ymmärrettäviltä.

Robert.C.Martin (2011) esittää kirjassaan The Clean Coder lausahduksen: ”Why do most developer fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.”. Tämä kuvastaa omalla kohdallani tilannetta varsin hyvin. En ole nähnyt tähän mennessä yhtäkään testiä työnantajani React-projekteissa. Tällä hetkellä ominaisuuksien testaus isoissakin projekteissa tapahtuu pääosin käsin ja on hyvin aikaa vievää. En ole myöskään kovin kokenut testaaja, mutta aikomukseni on kehittyä vähitellen paremmaksi.

Työnantajani projektien liittyessä televiestintään kuluneella viikolla olen myös ymmärtääkseni paremmin projektin käsitteitä tutustunut televiestintään liittyviin käsitteisiin, kuten PSTN (Public Switched Telephone Network), Trunk, VDN (Vector Directory Number), ja Vektorit. PSTN tarkoittaa julkista puhelinverkkoa, jossa puhelut kulkevat. *Trunk* tarkoittaa jotakuinkin ympäripyöreästi kuvattuna kommunikaatiokanavaa yrityksen omien puhelinvaihteiden ja PSTN-verkkoon kytketyn teleoperaattorin välillä (Wikipedia, 2021).

Vektorit ovat puhelujen ohjelmoituja kulkuja sähköisessä puhelinjärjestelmässä. Puhelun ohjautuessa sähköiseen asiakaspalvelujärjestelmään vektorit määrittävät mm. odotusaikojen pituuden, mahdollisia puhelun aikana tehtäviä IVR (Interactive Voice Response) - valintoja sekä mitä tiedotteita tai musiikkia puhelun aikana soitetaan (Wikipedia, 2021).

VDN on järjestelmään määritettävä *alanumero*, joka ohjaa vaihteeseen saapuvat puhelut oikeisiin vektoreihin. Niiden avulla sähköinen puhealusta osaa yhdistää mm. eri palvelunumerot niille määriteltyihin palvelujonoihin, esimerkiksi asiakaspalvelun tai myynnin puhelut omiin jonoihinsa (Wikipedia, 2021).

3.2 Seurantaviikko 2: 13.-17.09.2021

Maanantai 13.09.2021
Tänään pidettiin viikkopalaveri normaalisti aamulla, jossa on projektipäällikkö ja muita tiimini jäseniä. Ohjelmoinnin osalta jatkoin AgentUP-projektin parissa. Asiakas oli toivonut mahdollisuutta lisätä sähköposti-viestipohjiin liitetiedostoja, kopioita ja piilokopioita. Nämä monimutkaistivat osaltaan lomakkeen rakennetta ja react-hook-forms-kirjaston kanssa tuli myös hieman hankaluuksia liitetiedostojen suhteen, eikä kirjaston toiminta ollut siinä niin intuitiivista kuin yksinkertaisempien lomakkeiden tapauksessa. Kirjaston Typescript-tuki ja dokumentaatio oli myös toistaiseksi hieman hätäisesti tehdyn oloista edistyneempien toimintojen osalta.

Keskipäivällä oli työnantajani emoyhtiön järjestämä luento uupumuksesta ja sen tunnistamisesta. Se muistutti hyvin taukojen pitämisen tärkeydestä ja palautumisen merkityksestä. Työhyvinvointi on mielestäni tärkeä asia ja olen iloinen, että siitä puhutaan työpaikallani. AgentUP:n päivittäisessä statuspalaverissa käytiin läpi tilanne käyttöliittymä- ja API-kehityksen osalta ja päätettiin mitä kukin tekee seuraavaksi. Edellisellä viikolla päätökseen saadut asiat päätettiin myös viedä testiympäristöön. Centry-projektin osalta oli myös iltapäivällä palaveri, jossa katselmoitiin vielä aiempaa käyttöliittymää, keskusteltiin sen puutteista ja kehityskohteista ja päätettiin seuraavia toimenpiteitä, jota projektin valmisteluvaihe vielä vaatii.

Tiistai 14.09.2021
Jatkoin AgentUP:n parissa tehden Hallinta-käyttöliittymän sähköposti-viestipohjien lomakkeisiin kentät kopioille ja piilokopioille. Tämän kanssa tuli hieman hankaluuksia ja meni aika monta tuntia, että sain lomakkeen toimimaan niin kuin halusin, eli niin että kopio- ja piilokopiokenttiä voi dynaamisesti lisätä ja poistaa. Satunnaisesti hiiren klikkaukset aiheuttivat usean funktiokutsun niin, että kun lisäsi yhtä kenttää niin syntyi kaksi tai useampi. Yritin pitkään selvittää mistä ongelma johtuu, mutta en saanut sitä selville. Ongelmaa oli vaikea yrittää korjata, sillä se tuli vain todella satunnaisesti ja harvoin. Selvittelyyn meni niin pitkään, että jouduin luovuttamaan ja jatkamaan muiden töiden parissa, etten jää liikaa jälkeen. Ongelma ei ole kovin akuutti, sillä kenttien lisäys ja poisto toimi sinänsä hyvin. Tietokoneeni on muutenkin reagoinut ajoittain oudosti viime päivinä klikkauksiin ja ajattelin että ehkä ongelma voi olla jossain muussa. Myöhemmin kokeiltuani eri hiirtä huomasin,

että vika oli hiiressäni itsessään, ei sovelluksen toiminnassa. Päivän lopuksi tutustuin Centry-projektin aiempaan käyttöliittymään ja dokumentoin sen ominaisuuksia.

Keskiviikko 15.09.2021

Tänään aika kului oikeastaan lähes kokonaan Centry-projektin parissa, josta oli päivän aikana kaksi palaveria. Palaverien keskiössä oli uuden selainpohjaisen käyttöliittymän toteutus sekä uusia, että aiempia rajapintoja hyödyntäen. Palaverien ulkopuolella keskityin perehtymään aiempiin käyttöliittymiin. Opittavia asioita ja spesifiä terminologiaa on paljon ja toiminnot ja taustaprosessit ovat monimutkaisia. Uskon, että päivän palaverit selkeyttivät osittain asioita ja opin hieman lisää, mutta opittavaa on vielä paljon. Tulee menemään aikaa, ennen kuin ymmärrän edes summittaisesti mitä järjestelmän eri käsitteet tarkoittavat, mitä eri toimintoja aiemmissa toteutuksissa on ja mikä niiden merkitys on käyttöliittymän kannalta. Vanhat toteutukset ovat työpöytäsovelluksia, niitä on useampi ja niistä ulkopuoliselle muodostuva kokonaisuus on todella epäselvä, hajanainen ja monimutkainen. Myös vanhojen ja uusien toiminnallisuuksien yhteensovittaminen on haastavan tuntuista ja tuntuu, ettei kellään muullakaan ole siitä vielä selkeää kokonaiskuvaa.

Torstai 16.09.2021

Päivä kului suurimmaksi osaksi suunnittelutyön ja ideoinnin parissa Centry-projektiin liittyen. Koitin miettiä mahdollisia ratkaisuja Centry-projektin aiempien hajanaisten toteutusten yhtenäistämiseksi ja sulavamman kokonaisuuden muodostamiseksi. Sain joitakin ideoita visualisoitua ja tuntuu siltä, että ne ajatuksen tasolla vaikuttavat ihan hyviltä. Toteutuksessa tuli vastaan vielä yksi itselleni haastava asia, joka on käytettävän rajapinnan hyödyntäminen. Olen saanut siitä joitakin vuosia vanhan englanninkielisen dokumentaation, mutta rajapinnan koodiesimerkit ovat C-pohjaisia kieliä, joita en ymmärrä paljoakaan ja rajapintakutsut vaativat Googlen kehittämää nk. Protocol Buffer (protobuf)-enkoodausta, josta en ole aiemmin kuullutkaan. Jotta voisin toteuttaa ohjelmiston, joka hyödyntää tällaista rajapintaa nämä asiat vaativat vielä lisää opettelua ja perehtymistä.

Iltapäivällä oli palaveri AgentUP-projektiin liittyen ja siinä olisi vielä tekemistä, jota asiakas on toivonut hoidettavaksi. Centry-projektin suunnitteluun käyttämäni tuntien takia en ole ehtinyt hoitaa kaikkia asioita AgentUP-projektiin liittyen. Projektipäällikkö lupasi keskustella Centry-projektin projektipäällikön kanssa siitä, kumpi projekti on prioriteetti tällä hetkellä. Kontekstin vaihtaminen projektien välillä, jota joutuu useassa projektissa työskennellessä tekemään tuntuu välillä kuormittavalta.

Perjantai 17.09.2021

Tänään AgentUP-projektin rajapinnat oli edenneet siihen pisteeseen, että ensimmäinen versio liitetiedostojen lisäyksestä sähköposti-viestipohjiin tuli valmiiksi. Asiakkaalta oli

tullut joitakin pyyntöjä ulkoasumuutoksiin liittyen ja suurin osa päivän ohjelmointityöstä kului näiden parissa. AgentUP-projektissa on jäljellä vielä joitakin asiakkaan toivomia lisäominaisuuksia joita työstän vielä ensi viikolla.

Aamupäivällä oli palaveri Centry-projektiin liittyen, jossa ideoitiin uutta käyttöliittymää ja sen ominaisuuksia. Customerc-järjestelmä sisältää monia eri abstraktioita eri käsitteille, joka tekee kokonaisuuden hahmottamisesta monimutkaista erityisesti asiakkaiden näkökulmasta ja osasta käsitteitä haluttaisi päästä eroon. Järjestelmässä on samoille asioille useampia abstraktioita riippuen siitä, mistä järjestelmän osasta on kyse. Eri käsitteet tulevat todennäköisesti elämään vielä taustalla, mutta käyttöliittymästä niitä on tavoite piilottaa ja antaa taustaprosessien hoitaa likainen työ. Modernit käyttöliittymäkirjastot ja komponentit taas helpottaa hallintapaneelissa tiedon manuaalista hallintaa, joka taas on ollut vanhan ohjelmiston heikko kohta. Esitin palaverissa omia ideoitani ja näkökulmiani ja punnitsimme yhdessä, mitä eri mahdollisuuksia voisi olla. Tiimin käyttöliittymäsuunnittelija alkaa seuraavaksi työstämään rautalankamallia, jonka pohjalta toteutusta aloitan myöhemmin tekemään. Yksi tiimin kokeneemmista jäsenistä taas kokoaa yhteen käyttötapauksia, joita tulee ottaa huomioon ominaisuuksia suunniteltaessa.

Viikkoanalyysi

Viikko oli mielenkiintoinen ja haastava uutta Centry-projektia varten käynnistetyn käyttöliittymäsuunnittelun näkökulmasta. Sain lisää kokemusta suunnittelutyöstä ja ideoistani tuli positiivista palautetta. Useissa palavereissa oli mukana työntekijöitä, jotka ovat itse käyttäneet aiempia hallintapaneeliratkaisuja. Tämä auttoi minua kehittäjänä ja kollegaani käyttöliittymäsuunnittelijana hahmottamaan aiempien ratkaisujen puutteita ja mahdollisien tulevien loppukäyttäjien ja asiakkaiden tarpeita. Galitzin (2007, osa 2) mukaan loppukäyttäjät kannattaa osallistaa käyttöliittymän suunnitteluprosessiin ja vaatimusmäärittelyyn mahdollisimman aikaisessa vaiheessa, koska heillä on suoraa tietoa järjestelmän tarpeista ja tavoitteista.

Centry-projektin keskitetty hallintakäyttöliittymä on tarkoitus tuottaa aluksi yrityksen omaan testikäyttöön jolloin Service-deskin työntekijät testaavat yrityksen järjestelmien käyttäjien, sovellusoikeuksien ja muiden järjestelmän osien hallintaa. Myöhemmin, kun sovellus on saatu testattua ja toimivaan vaiheeseen se asennetaan asiakkaiden käyttöön, jolloin asiakkaat voivat itse hallinnoida palvelutuotteitaan säästäen ylimääräistä manuaalista työtä Service-desk-työntekijöiltä. Vanhat käyttöliittymät on myös ominaisuuksiltaan hyvin kankeita ja vanhanaikaisia, joka tekee niistä hitaita ja epämukavia käyttää. Uusilla teknologioilla toteutettu käyttötarpeita palveleva käyttöliittymä tekee toivottavasti järjestelmän hallinnoimisesta nopeampaa, helpompaa ja vaivattomampaa.

Käytin viikolla paljon aikaa käyttöliittymän suunnitteluun pohtien käyttäjien tarpeita sekä aikaa järjestelmän hahmottamiseen kokonaisuutena. Sovelluskehittäjien ja loppukäyttäjien välillä vallitsee usein isoja eroja työtehtävissä, asenteissa ja teknisissä valmiuksissa (Galitz 2007, osa 2). Tämän takia on tärkeää irtaantua omista käsityksistä ja yrittää ymmärtää mitä loppukäyttäjät tuotteelta haluavat. Suunnitelmissani keskityin aiempien toteutusten ongelmakohtiin ja siihen, miten hajanaisesta kokonaisuudesta saisi luotua selkeämmän, kompaktimman ja helpommin hahmotettavan kokonaisuuden. Eri osa-alueisiin liittyviä vanhempia ja uudempia ominaisuuksia oli edellisessä käyttöliittymässä ripoteltu hieman pitkin poikin ja päätin tuoda yhteen liittyvät ominaisuudet lähemmäs toisiaan.

Osa uusista asioista on tuntunut myös haastavilta ja olen joutunut totuttelemaan siihen, että projekteissa on usein epävarmuustekijöitä. Joskus uudet asiat voivat tuntua vaikeilta ja on vain lähestyttävä ongelmia ennakkoluulottomalla asenteella. Pragmaattisesti ongelmia ratkomalla asiat yleensä lähtevät selkenemään. Olen myös harjoitellut kysymään kollegoiltani apua ja päässyt haasteista yli uskallettuani kysyä.

3.3 Seurantaviikko 3: 20.-24.09.2021

Maanantai 20.09.2021
Viikko alkoi normaaliin tapaan viikkopalaverilla, jossa katselmoitiin frontend-kehityksen tilannetta eri projekteissa tällä hetkellä. Tämä viikko minun on tarkoitus käyttää pääosin AgentUP-projektin tekemiseen, jotta asiakkaan (Upwire) pyytämät ominaisuudet saataisiin käyttöön.

Uusiin ominaisuuksiin lukeutuu mm. kontaktien tarjoaminen määrätyille asiakaspalvelijoille tietyin ehdoin, esimerkiksi kielen perusteella. Jotta määritykset voisi tehdä, tälle täytyy tehdä oma osuus Hallinta UI:lle kampanjan asetuksiin. Tiimissäni toinen backend-kehittäjä hoitaa uusiin ominaisuuksiin liittyvää rajapintakehitystä.

Myös kampanjoiden asetuksiin asiakaspalvelijoiden osalta on tehtävä muutoksia siihen, miten asiakaspalvelijoita voidaan lisätä kampanjoihin. Asiakaspalvelijoista voi määritellä mm. ryhmiä, johon kuuluvat asiakaspalvelijat on mahdollista lisätä kampanjaan, mutta ryhmistä tulisi voida myös valita yksittäisiä asiakaspalvelijoita.

Tiistai 21.09.2021
Tänään tein muutoksia Hallinta UI:n kampanjoiden asiakaspalvelija-asetuksiin. Aiemmin asiakaspalvelijat ja asiakaspalvelijaryhmät olivat listattuna ja listaa klikkamalla niitä pystyi lisäämään kampanjaan. Lisäsin listoihin checkbox-laatikoita, joita klikkaamalla pystyy valikoimaan joko tietyt ryhmän tai listan asiakaspalvelijat tai vaihtoehtoisesti kaikki kerralla.

Koodi, joka käsitteli kyseistä osuutta ohjelmasta oli vanhentunutta ja siinä oli puutteita. Komponentit olivat liian laajoja, niissä oli liikaa tilamuuttujia ja näiden ylläpitämisestä aiheutuneita sivuvaikutuksia. Isossa osassa komponenttien apufunktioita oli myös käytetty *imperatiivista* ohjelmointiparadigmaa, vaikka React noudattaa *deklaratiivista* paradigmaa (Roldán 2021, luku 1.2). Refaktoroin tällaiset osuudet käyttämään deklaratiivista ohjelmointitapaa ja pilkoin komponentteja pienempiin osiin. Hyödynsin tässä Dan Abramovin (2015) esittämää mallia komponenttien jaottelusta *esittäviin (engl. Presentational)* komponentteihin ja *säiliö (engl. Container)* - komponentteihin. Näin koodista syntyi siistimpiä ja luettavampia kokonaisuuksia. Koodia on helpompaa ylläpitää myöhemmin, koska kaikki tilamuuttujat ja tapahtumankäsittelijät ovat säiliökomponentissa tehden lapsikomponenteista yksinkertaisempia.

Keskiviikko 22.09.2021

Tänään halusin kokeilla jotain uutta ja hyödynsin päivittäisessä työssäni Roldánin (2021, luku 4.6) esittämää suunnittelumallia React-ohjelmoinnissa. Mallin nimi on ”FunctionAsChild”, jonka ideana on, että komponentin lapsi on toisen komponentin sijaan funktio, jolle voi antaa parametreja suoraan. Käytin tätä mallia hakiessani API:lta dataa, jota alikomponentin lomake tarvitsi ja annoin kyseisen datan parametrina komponentin lapsifunktiolle. Koska React-komponentit ovat pohjimmiltaan funktioita, näin voi tehdä.

Päivä menikin pitkälti lomakkeen tekemisessä. Tämä muutos koski Hallinta UI:n kampanjoille asetettavia ehtoja, joiden avulla voidaan määrittää yksittäiset asiakaspalvelijat, keille kontakteja tarjotaan määritellyn ehdon perusteella. Aamupäivästä suunnittelin lomakkeen ulkoasun ja käytin loppupäivän toteutukseen. En ehtinyt täysin saada kaikkea valmiiksi, koska lomakkeen käyttämät komponentit ja toiminnallisuudet ovat sen verran monimutkaisia. Vaiva kannattaa, koska sitten kun lomake on valmis se on käytettävyydeltään erinomaisen mukava ja helppo käyttää. Pääsin kuitenkin hyvään vaiheeseen ja päätin jatkaa huomenna.

Torstai 23.09.2021

Tänään jatkoin eilisen lomakkeen ja sen validaation parissa AgentUP-projektin HallintaUI:lle. Lomakkeilla on tarkoitus luoda ja muokata asiakaspalvelijoille tai niiden ryhmille liitettäviä ehtoja, joiden mukaan muodostetaan lista kohteita soitettavaksi.

Kesti aikaa saada lomake toimimaan halutulla tavalla sen lopulta monimutkaisen toiminnan takia. Yhden lomakkeen kentän sisältö vaikutti siihen, mitä vaihtoehtoja toisessa lomakkeen kentässä näytetään ja sisältö täytyi saada vaihdettua oikea-aikaisesti. Myös sisällön vaihdon yhteydessä kentän sisältämän datan rakenne muuttui. Sain lopulta kaikki

toimimaan niin kuin halusin. Lomakekomponentista tuli lopulta pakostikin aika laaja ja sitä voisi myöhemmin refaktoroida osiin. Joskus kuitenkin aika tulee vastaan ja täytyy saada asioita tehdyksi, vaikka niitä voisi hioa loppuun tolkuttoman kauan.

Upwire oli keskiviikkona testannut uusia ominaisuuksia ja heiltä oli tullut lista korjaustoivomuksia. Tämä päivä kului pitkälti lomakkeen parissa ja sen testauksessa, joten ehdin paneutua muutospyyntöihin vasta huomenna.

Perjantai 24.09.2021

Tämä päivä kului pitkälti Upwiren muutospyyntöjen toteuttamiseen. Muutokset koskivat sekä HallintaUI:ta että AspaUI:ta. Kampanjan viestipohjien liitetiedostojen tallennustapaan oli tehty muutoksia, jotka aiheuttivat muutostarpeita käyttöliittymässäkin. Jotkut asiakkaan raportoimat virhetilanteet johtuivat rajapinnan häiriöistä ja jotkut käyttöliittymän häiriöistä. Osa asiakkaan pyytämistä muutoksista liittyi ulkoasuun ja ne olivat erittäin helppo korjata. Projekti alkaa olla jo niin laaja, että toiminnallisuuksien osalta muutostyöt vievät kohtuullisen paljon aikaa. Varsinkin, kun jotkut uudet ominaisuudet aiheuttaa muutostarpeita useammassa paikassa täytyy usein punnita, miten saisi aikaan halutun lopputuloksen mahdollisimman vähillä sivuvaikutuksilla.

Sekä Aspa UI ja Hallinta UI on jonkun muun kuin itseni alullepanema ja muiden kirjoittamaa koodia on projekteissa paljon pohjalla. Jotkut ratkaisuista on hankalasti ylläpidettäviä ja niiden uudelleenkirjoittaminen veisi niin paljon aikaa, että jäisin pahasti jälkeen nykyisistä töistäni. Vaikka välillä houkuttelee kirjoittaa jotkut asiat kokonaan uudelleen, yritän toteuttaa muutoksia mahdollisimman modulaarisesti ja vähillä sivuvaikutuksilla niin, että siistin koodia pala kerrallaan sieltä täältä silloin kun se ei aiheuta kohtuuttoman paljon aikaa vaativia muutoksia.

Viikkoanalyysi

Tällä viikolla opin lisää deklaratiivisesta ohjelmointitavasta ja sen noudattamisesta React-ohjelmoinnissa. Myös Dan Abramovin ajatusten komponenttien jaottelusta toiminnallisiin (*engl. presentational*) ja ulkoasullisiin (*engl. container*) komponentteihin tuntui olevan hyötyä. Yritän jatkuvasti kehittyä ja löytää helpommin ylläpidettäviä tapoja kirjoittaa koodia ja etsin vielä omaa kultaista keskitietäni. Erityisesti haluan oppia pitämään koodia ylläpidettävänä laajoissa projekteissa, koska se on minulle vielä suhteellisen uutta.

Lomakkeiden toteuttaminen on ollut Reactilla haastavaa, koska React ei tarjoa niiden tekemiseen mitään valmista ratkaisua vaan enemmänkin vain työkalut niiden tilojen ylläpitämiseen. Lomakkeet, joita projektini tarvitsevat ovat myös suhteellisen monimutkaisia tarpeiltaan ja käyttötarkoituksiltaan. Niitä on myös lukumäärältään paljon.

Monimutkaisempien lomakkeiden tekemiseen lomakekirjastoista on ollut apua. Olen kokeillut eri kirjastoja ja päätynyt siihen, että react-hook-forms-kirjasto palvelee parhaiten käyttötarpeitani monipuolisuutensa ja joustavuutensa ansiosta.

Haluaisin kuitenkin vielä oppia toteuttamaan uudelleenkäytettävämpiä komponentteja, jotta en joutuisi kirjoittamaan asioita niin usein uudelleen. Usein projektissa, jossa työskentelen ilmenee uusia tarpeita eri ominaisuuksille. Tarpeet ovat aina tapauskohtaisia ja usein päädyn kirjoittamaan paljon uutta koodia, vaikka pyrinkin parhaani mukaan hyödyntämään olemassaolevaa. Tässä koen, että minulla on kehittymisen varaa ja haluaisin tulevaisuudessa voida kasvattaa koodini uudelleenkäytettävyyttä.

Toinen asia, jossa haluaisin kehittyä sekä itse sekä jota haluaisin kehittää työpaikallani yleisestikin on testaus. Tällä hetkellä käyttöliittymien testaus tapahtuu pääosin manuaalisesti käsin. En ole työskennellessäni nähnyt React-projekteissa yhtäkään yksikkö- integraatio- tai end-to-end -testiä. Testauskirjastoja ja työkaluja automaattitestaukseen on kuitenkin olemassa ja usein minusta tuntuu, että ne olisivat hyväksi. Työnantajan taholta kuulen usein lausahduksia testauksen osalta siitä, kuinka siihen ei tällä hetkellä vain ole tarvittavia resursseja tai vakiintuneita käytäntöjä. Usein minua myös turhauttaa rajapintakehityksen testauksen puutteellisuus. Osa kehittäjistä ei joko ehdi tai osaa testata kehittämiään toiminnallisuuksia kyllin kattavasti.

Monesti manuaalinen testaus tuntuu työläältä ja aikaa vievältä etenkin laajojen sovellusten kohdalla. Olen aiemmin tutustunut Cypress-nimiseen työkaluun, jota käytetään Javascript-kehityksessä end-to-end - testaukseen. Olen harkinnut ottavani sen jossain vaiheessa käyttöön projekteissa, joissa manuaalista testausta joutuu tekemään paljon. Cypress-testit simuloivat selaimen käyttäjää suorittamalla selainta omassa ikkunassaan ja tekemällä hiirellä ja näppäimistöllä asioita, joita sovelluksen loppukäyttäjäkin tekisi. Näin myös saadaan testattua tehokkaammin käyttöliittymän ja palvelinohjelmiston välistä vuorovaikutusta ja sen toimivuutta.

3.4 Seurantaviikko 4: 27.09.-01.10.2021

Maanantai 27.09.2021
Tänään oli normaaliin tapaan viikkopalaveri, jossa käytiin tiimini kesken läpi mitä töitä kenelläkin on käynnissä tällä hetkellä ja projektien tämänhetkinen tilanne. Osa kehittäjistä tekee usein töitä samaan aikaan useampaan projektiin itseni mukaan lukien ja joskus projektien välillä tulee konflikteja. Iso osa ajastani menee tällä hetkellä asiakkaalta tulleiden mukautusten tekoon AgentUP-projektissa ja Centry-projektia täytyisi toisen

projektipäällikön mukaan saada eteenpäin. Sovin tekeväni osan viikosta AgentUP-projektin muutoksia ja loppuviikosta Centry-projektia.

Tänään ohjelmoinnin osalta jatkoin AgentUP-projektin Hallinta UI:n sähköposti-viestipohjien viimeistelyä. Lisäsin toiminnallisuudet liitetiedoston lisäämiseen olemassaoleviin viestipohjiin ja mahdollisuuden liitetiedoston poistoon. Tein myös korjauksia Aspa-UI:lle lomakekenttien validaatioon. Asensin uusimpia muutoksia iltapäivästä testipalvelimelle ja testasin niitä.

Tiistai 28.09.2021
Tänään jatkoin asiakkaan muutospyyntöjen toteutusta kampanjoiden lisäasetusten osalta. Kampanjoille voi määritellä erilaisia lopputuloksia puheluille. Näistä erillinen toiminnallisuus on myös määritellä soittokiintiöitä tiettyjen määreiden mukaan, joiden täyttyessä kontaktien antaminen kampanjassa lopetetaan. Muutokset koskivat lopputuloksiin liittyvien kiintiöiden määrittelyä. Kiintiöistä oli aiempi toteutus jo valmiina ja lopputulokset näin ollen oli helppo lisätä joukkoon.

Käyttöliittymämuutosten jälkeen testasin muutosten toimivuutta. Käytin AgentUP-projektin projektipäällikön tekemää ohjelmaa, joka tekee automatisoitua *end-to-end* -testausta, varmistaakseen käyttöliittymän ja rajapinnan yhteistoimivuutta.

Iltapäivällä tein käyttöliittymiin pieniä bugikorjauksia ja osallistuin päivittäiseen AgentUP-projektin seurantapalaveriin. Työpäivän lopuksi asensimme kollegani kanssa uusimmat muutokset asiakkaan testiympäristöön, jossa testasimme muutosten toimivuutta. Asiakkaan on tarkoitus seuraavana päivänä testata uusia ominaisuuksia. Kaikki vaikutti toimivan ja olin kohtalaisen tyytyväinen päivän aikaansaannoksiin.

Keskiviikko 29.09.2021
Eilen AgentUP-projektissa saimme uusimpia ominaisuuksia vietyä asiakkaan testattavaksi, joten tänään minulla oli aikaa käytettäväksi Centry-projektiin. Projektista oli pari palaveria aamupäivällä ja osallistujilla oli kehityksen suunnasta eriäviä mielipiteitä. Osa teknisistä asiantuntijoista oli projektipäällikön kanssa eri mieltä asioista ja projektin epävarmuustekijöihin ei tuntunut löytyvän yhteistä säveltä.

Centry projektin kesällä aloitettuun osaan oli myös tehty testausta, johon liittyen minulle oli laitettu muutospyyntöjä sähköpostiin ja käytin päivän pääasiassa niiden toteuttamiseen. Iltapäivällä oli AgentUP-palaveri, jossa tällä hetkellä muutokset painottuu rajapinnoille, joten sovimme, että jatkan Centry-projektin työstämistä. Centry projektin käyttöliittymässä tällä hetkellä haastavimmalta tuntuu monimutkaisten lomakkeiden teko. Olin aiemmin

kesällä käyttänyt Formik-nimistä kirjastoa lomakkeiden hallintaan, mutta huomattuani viimeaikoina react-hook-form-kirjaston hyödyt päätin vaihtaa projektiin kirjaston kokonaan. Sain perustoiminnallisuuden vaihdettua kirjastosta toiseen suhteellisen nopeasti, mutta validaatiota en ehtinyt saada tämän päivän aikana toimimaan.

Torstai 30.09.2021

Tänään jatkoin Centry-projektia muutospyyntöjen osalta ja tärkeimpänä muutoksena järjestelmän käyttäjille täytyi voida asettaa tiettyjä ominaisuuksia manuaalisesti. Iso osa päivästä meni myllätessä lomakkeita uusiksi uutta lomakekirjastoa (engl. *react-hook-form*) varten. Sain lopulta kuitenkin lomakkeet toimimaan niin kuin pitääkin. Syy, miksi aikaa menee niin paljon muutoksiin on se, että rajapinta joka käyttöliittymälle on tehty ei ole kovinkaan helppokäyttöinen. Se on syystä tai toisesta rakenteeltaan hyvin rakeinen joka tekee siitä hankalan käyttää. Rakeisuuden takia käyttöliittymästä joutuu tekemään useita kutsuja ja samojen käsitteiden datan rakenne vaihtelee taustalla olevien teknisten yksityiskohtien takia. Tämä tekee tiedon käsittelystä ja tilanhallinnasta käyttöliittymässä monimutkaista ja muutosten tekemisestä koodiin aikaavievää.

Kirjoitin apufunktioita jotka helpotti datan käsittelyä ja tein niille yksikkötestit päivän lopuksi. Testaaminen oli virkistävää, koska olen miettinyt testauksen puutteellisuutta jo jonkin aikaa eri projekteissa. Päätin aloittaa pienistä yksikkötesteistä, koska itsenäisistä apufunktioista oli suhteellisen helppo muodostaa luotettavasti testattavia kokonaisuuksia.

Perjantai 31.09.2021

Tämän päivän AgentUP-projektin palaveri oli poikkeuksellisesti aamulla. Asiakas ja projektipäällikkö oli testannut ohjelmistoa ja kävimme läpi jäljellä olevia asioita sekä muutamia löytyneitä bugeja. Näistä osan korjasin aamupäivän aikana. Aamupäivällä oli myös palaveri Centry-projektiin liittyen, jossa sain hyvää palautetta aikaisemmin viikolla tekemistäni muutoksista. Kuulin, että sovelluksen nykyinen versio aiotaan piakkoin asentaa asiakkaan käyttöön. Sovimme asioista, joita kukin vie eteenpäin projektin osalta.

Iltapäivällä hoidin AgentUP-projektiin työpöydällä olevia asioita. Korjasin aiemmin ilmenneitä vikoja ja sain suurimman osan tehtävälistalla olleista asioista tehtyä. Jäljellä on vielä kontaktilistoille tehtävä sivutus, jotta suorituskyky ei huonone, vaikka kontakteja olisi kampanjassa satoja tuhansia. Tähän tulee todennäköisesti kulumaan enemmän kuin yhden päivän työtunnit, joten jatkan sen parissa ensi viikolla. Pidimme iltapäivällä kollegoitteni kanssa myös pienen palaverin Speakex-puhealustaan hallintaan käytettäviin rajapintoihin liittyvän palaverin. Rajapinnat käyttävät tiedonvälitykseen Googlen 2000-luvun alkupuolella kehittämää Protocol Buffers-serialisointitapaa.

Viikkoanalyysi

Tällä viikolla opin lisää itselleni täysin ennalta tuntemattomasta formaatista. Protocol Buffers-serialisointitapa perustuu siihen, että ennen lähetystä kyselyt enkoodataan binäärimerkkijonoiksi jotka lähetetään palvelimelle, joka vuorostaan purkaa enkoodauksen ja vastaanottaa kyselyn (Abuelenain, Doyle, Karneliuk & Jain 2021, luku 15.4). Google käyttää serialisointitapaa korkean rasituksen alaisena olevien järjestelmien tiedonvälitykseen. Binäärimuotoon pakkaaminen tekee kyselyjen tekemisestä hyvin tehokasta. (Abuelenain, Doyle, Karneliuk & Jain 2021, luku 15.4.) Formaatin oppimiskäyrä itselleni tuntuu olevan hyvin jyrkkä, mutta näen sen samalla myös mahdollisuutena oppia uusia asioita. Uskon myös oppivani tarpeelliset asiat, kunhan pääsen alkuun.

Tällä viikolla kirjoitin myös ensimmäiset varsinaiset yksikkötestini. Sekin tuntuu askeleelta eteenpäin, sillä selainpohjaisten käyttöliittymien testaukseen ei tunnu olevan tällä hetkellä työpaikallani mitään vakiintuneita käytänteitä. Kaiken manuaalinen testaaminen vie myös paljon aikaa, varsinkin kun projektit ovat aika laajoja. Olen myös harkinnut tekeväni cypress-nimisellä kirjastolla end-to-end testejä, jotta aikaa vieviä manuaalisia testejä voisi automatisoida. Torstaina kirjoittaessani testejä havaitsin testauksen konkreettisia hyötyjä, sillä testien kirjoittamisen prosessi paljasti puutteita yhdessä funktioita, joita testasin. Puutteen havaittuani se oli helppo ja nopea korjata ilman tuskallista vianmääritysprosessia, koodin lukemista ja pohdintaa siitä, mikä ongelman mahtaa aiheuttaa.

Projektien välillä vaihtaminen tuntuu myös haastavalta. Mieli voi olla aluksi toisessa projektissa ja kontekstin vaihtaminen tuntuu kuormittavalta. Tällä viikolla olen myös alkanut käyttää muistiinpanoja organisoidummin. Käytän sovellusta jossa on värikoodillisia muistilappuja joihin kirjaan eri projekteihin liittyviä ajankohtaisia asioita ja tehtäviä. Muistilappujen värit helpottavat erottamaan eri projekteihin liittyvät asiat toisistaan. Usein projektipäälliköille raportoitaessa on hyvä olla ylhäällä, mitä konkreettista on muuttunut ja minne. Tämä on tuntunut hyödylliseltä ja aion jatkaa sitä. Olen myös harkinnut ostavani oikeita muistilappuja ja tekeväni niistä seinälleni kanban-taulun. Yrityksessäni käytetään projektienhallintaan Jira-nimistä ohjelmistoa, mutta kunnolliset yhteiset pelisäännöt puuttuvat parhaan hyödyn saamiseksi ja käytön taso on vaihtelevaa ja ylimalkaista. Osa syystä on mielestäni Jiran järkälemäisyys joka pikemminkin hankaloittaa elämää kuin helpottaa sitä, mutta se on vain oma mielipiteeni. Näkökulmia on varmasti monia.

3.5 Seurantaviikko 5: 04.-08.10.2021

Maanantai 04.10.2021

Tänään oli aamulla normaaliin tapaan viikkopalaveri, jossa katsottiin mitä kenelläkin oli ollut työn alla viime viikolla ja mitä kukin aikoo jatkaa tällä viikolla. Projektipäällikkö, joka

vetää AgentUP-projektia ei toistaiseksi osallistu näihin viikkopalavereihin. Projektipäällikkö, joka vetää Centry-projektia taas vetää viikkopalaverit ja haluaa, että minä ja kollegani viemme AgentUP-projektin mahdollisimman pikaisesti loppuun, jotta muita projekteja saataisiin eteenpäin.

AgentUP-projektilla taas ei ole tuntunut olevan oikein selkeää päätepistettä, vaan tehtäviä asioita on vain tullut aina lisää. Tätä myötä projektin laajuus tuntuu paisuvan ja aikataulu venyvän. Aikapaine tuntuu kasvavan, mutta kukaan projektipäälliköistä ei tunnu ottavan AgentUP-projektin kokonaisvaltaisesta hallinnasta tarpeeksi vastuuta. Päävastuu tuntuu olevan liian jakautunut tai sysätty kehittäjien harteille, eikä projektipäällikkö edes osallistu päivittäisiin palavereihin.

Ohjelmointipuolella jatkoin HallintaUI:n muutosten parissa. Kampanjoiden kontaktit listataan kampanjoiden asetuksissa. Näihin kontaktilistoihin tuli toteuttaa sivutus niin, ettei jouduta hakemaan kaikkia kampanjan kontakteja kerralla. Joissain tapauksissa kampanjoihin täytyy voida ladata puolikin miljoonaa kontaktia, joten kaikkien kerralla hakeminen kestää liian kauan. Sivutukseen oli jo olemassa rajapintakutsut, joiden implementoimiseen ja testaamiseen käyttöliittymässä käytin tänään aikani.

Tiistai 05.10.2021
Tänään implementoin HallintaUI:lle kontakteihin liittyviä muutoksia. Kontaktitaulukon sivutus vaati uudet massatoiminnot, joilla voi hallita kaikkia kampanjan kontakteja kerralla. Kollegani oli saanut tähän liittyvät rajapintakutsut valmiiksi, joten ne täytyi ottaa käyttöön myös käyttöliittymässä. Kun olin saanut muutokset valmiiksi ne täytyi testata ja korjata virheet joita esiintyi.

Iltapäivällä pidimme kollegani kanssa palaverin, jossa katsottiin tämän hetken tilanne ja pohdittiin mitä kukin vie seuraavaksi eteenpäin. Suunnittelimme seuraavaksi toteutettavia ominaisuuksia ja sitä, kuinka aiomme testata nykyisiä uusimpia muutoksia. Loppupäivän käytin projektin koodin siistimiseen. Kokeilin konfiguroida koodin tyylisääntöjä säätelevän ESLint-työkalun avustamaan koodityylin määrittelyssä, mutta päätin olla lopulta ottamatta kaikkia uusia tyylisääntöjä käyttöön, koska projekti on jo niin laaja ja kokonaisvaltaiseen koodityylin muuttamiseen menisi tuhottoman paljon aikaa. Tällaiset asiat olisi tullut ottaa huomioon projektin alkuvaiheessa niiden henkilöitten toimesta ketkä projektia aloittivat. Päätin tehdä muutoksia hieman pienimuotoisemmin ja jatkaa vähittäistä siistimistä sitä mukaa kun projektia täytyy työstää.

Keskiviikko 06.10.2021

Tämän päivän tavoitteena minulla on jatkaa koodin siistimistä Hallinta UI:lla. Aamupäivällä on myös palaveri Centry-projektiin liittyen, jossa katsotaan tämän hetken tilanne uuden järjestelmänhallintapaneelin kehityksen suhteen.

Rajapinta, jolla hallitaan Speakex-puhelustaa käyttää Googlen protocol buffers-enkoodausta ja käytin aamupäivällä aikaa rajapintakyselyiden tekemisen selvittelyyn. React-ohjelmasta protobuf-kyselyiden tekeminen ei vaikuta olevan se kaikista yleisin käytäntö ja asia vaati hieman tarkempaa tutkimista. Kokeilin paria eri tapaa tehdä kyselyitä ja päätin näistä lähestymistavan, joka tuntui itselleni helpoiten lähestyttävältä ja johon kehitystyökalut antoivat parhaiten automaattitäydennysvihjeitä.

Rajapintaa varten on sen luomisen yhteydessä luotu protobuf-teknologialle tyypillinen ”proto”-tiedosto, joka sisältää tiedon käsiteltävien dataobjektien ja kyselyiden rakenteesta toimien pohjapiirrustuksena tehtäville kyselyille. Tämän tiedoston pohjalta on generoitu Javascript-tiedosto, joka sisältää tarvittavat funktiot datan käsittelyyn. Tehdessäni kyselyitä React-ohjelmasta käsin käytän tätä generoitua tiedostoa apuna datan enkoodauksessa, dekoodauksessa ja muussa käsittelyssä. Käytin kyselyiden testaukseen ja datarakenteen tutkimiseen pitkälti koko päivän.

Torstai 07.10.2021
Tämän päivän jatkoin Centry-projektin tutkimisen ja siihen liittyvien kokeilujen parissa. Käyttöliittymässä on tarkoitus käyttää Reactille kehitettyä Material-UI – komponenttikirjastoa. Material UI sisältää maksullisen datataulukon (Material -UI X-Data-Grid-Pro), jota aiemmin olen ehdottanut Centry-projektissa hyödynnettäväksi. Monet työnantajani ohjelmista sisältää paljon taulukoita ja erilaisia vaatimuksia taulukoiden ominaisuuksia ja käytettävyyttä koskien on ilmennyt eri projekteissa paljon.

Sen sijaan, että kehittäjien aikaa käytetään tekemään jokaiseen käyttöliittymään kaikki räätälöidyt komponentit erikseen käsin, kannattaa mielestäni hyödyntää sitä, mitä on valmiiksi saatavilla. Kuukausimaksu monipuolisia toimintoja tarjoavaan kirjastoon tulee työnantajalle suhteessa halvemmaksi kuin ohjelmoijien tuntipalkat, joita käytetään taulukoiden rakentamiseen.

Olen myös keskustellut uudemman kollegani kanssa, että olisi hyvä rakentaa uudelleenkäytettäviä komponentteja sisältävä keskitetty komponenttikirjasto Storybook-nimistä käyttöliittymäkehitystyökalua hyödyntämällä, joiden osia voisi tulevissa käyttöliittymissä hyödyntää. Tämä mahdollisesti voisi säästää aikaa uusissa projekteissa keskittäen työnantajani brändi-ilmeen tuomisen uudelleenkäytettäviin komponentteihin. Aika ei kuitenkaan ole

projektityöltä toistaiseksi tähän riittänyt tai aloite ei ole ollut kyllin vahva sen toteuttamiseksi.

Iltapäivällä minulle annettiin pienehkö tehtävä AgentUP-projektiin liittyen, jonka toteutin. Huomasin samalla pari vahingossa jäänyttä bugia ja korjasin ne. Korjauksien päätteeksi asensin uusimmat muutokset testipalvelimelle.

Perjantai 08.10.2021
Tämän päivän käytin Centry-projektiin. Aamupäivällä pidimme palaverin projektiin liittyen. Käyttöliittymäsuunnittelija oli sairaslomalla, joten päätimme jatkaa ulkoasusuunnitelmien osalta myöhemmin. Esittelin teknisestä näkökulmasta Material UI:n DataGrid Pro:ta ja sen ominaisuuksia, joita voisimme projektissa hyödyntää. Muut paikallaolijat pitivät ehdotustani hyvänä ja päätimme, että jatkan kokeilujani ja yritän ottaa komponentin käyttöön tähänastisten Centry-projektin ominaisuuksien osalta.

Projektin käyttöliittymään on tähän asti toteutettu järjestelmän käyttäjien ja niihin liittyvien oikeusluokkien hallinta. Tällä hetkellä ulkoasun suunnitteluvaihe on käynnissä muiden Speakex-puhealustan objektien hallintaan liittyvien asioiden kannalta. Näitä ovat mm. palvelutehtävät, joita puhealustan käyttäjille eli agenteille asetetaan. Jotta asiakaspalvelija voisi hoitaa tietyntyyppisiä puheluita, täytyy sille asettaa kyseinen palvelutehtävä. Näiden hallintaa käyttöliittymäsuunnittelija paraikaa työstää.

Olen myös jo jonkin aikaa pohtinut React-sovellusten tilanhallintaa ja sitä, kuinka sen voisi laajemmissa sovelluksissa toteuttaa mahdollisimman selkeästi, modulaarisesti ja helposti skaalattavasti. Aiemmissa projekteissa tilanhallintalogiikka on toteutettu Redux-kirjastoa hyödyntäen yhdellä tavalla, mutta tiedän että modernimpiakin tapoja on olemassa. Myös Material-UI-käyttöliittymäkirjastosta on hiljattain julkaistu uusi pääversio, joka on monilta osin hieman erilainen kuin aiemmat. Päätin, että Centry-projektia varten voisi olla helpompi alustaa kokonaan uusi projekti, kuin yrittää muokata vanhaa täysin uuteen uskoon. Käytin loppupäivän uuden projektin alustamiseen.

Viikkoanalyysi
Tällä viikolla olen oppinut lisää protobuf-rajapinnoista, tutustunut uusiin datataulukkokirjastoihin ja selvitellyt niiden välisiä eroja. Uusiin asioihin tutustuminen, joista muilla ei ole ennestään paljoa antaa ammatillista vastuuta ja haastaa muodostamaan rajatussa aikaikkunassa ammatillisen mielipiteen jostakin itselle vähemmän tutusta asiasta. Asiantuntijalausuntojen antamiseen on vähitellen alkanut tottua ja tämä on myös pakottanut arvioimaan toisistaan poikkeavia tuotteita kriittisesti ja mahdollisimman objektiivisesti. Se on ollut

hyvää harjoitusta ja karaisevaa itsessään. Perustellun mielipiteen osoittaminen auttaa löytämään itsevarmuutta ja tukee ammatillisen identiteetin muodostumista.

Kollegani mainitsi aiemmin Storybookin hyödyntämisestä brändinmukaisen komponenttikirjaston kehittämiseksi. Tämä on ollut silloin tällöin mielessäni jo jonkin aikaa. Selainpohjaisia käyttöliittymiä tuntuu alkavan kertyä jonkin verran työnantajani eri projektien välillä ja niihin on alettu tuoda työnantajani emoyhtiön design-suuntaviivojen mukaista muotokieltä, värimaailmaa, logoja, typografiaa ja käyttöliittymäelementtejä. Uusia projekteja aloittaessa designin säilyessä muotokieleltään jotakuinkin samana monesti tuntuu siltä, että joutuu tekemään samoja asioita useasti tai kopioimaan tyylejä paikasta toiseen. Omaan tarkoitukseen sopiva lähinnä ulkoasuun keskittyvä komponenttikirjasto jota kaikki voisivat hyödyntää vaikuttaisi ainakin ajatuksen tasolla hyvältä.

Olen toisaalta keskustellut kokeneempien kehittäjäystävieni kanssa valmiiden komponenttikirjastojen käyttämisestä ja kuullut myös tarinoita kalliista ja pitkään kestäneistä projekteista, joissa on lähdetty tekemään valmiin komponenttikirjaston päälle omaa design-järjestelmää kaikkine mukautuksineen eikä loppua näy. Olen myös kuullut useammalta taholta, että valmiit käyttöliittymäkirjastot ovat hyviä, mikäli halutaan saada siedettävän näköisiä ratkaisuja suhteellisen nopeasti, mutta mikäli ratkaisuja aiotaan ylläpitää pidempään tai mikäli valmiiseen käyttöliittymäkirjastoon jouduttaisiin tekemään paljon mukautuksia kannattaa rakentaa komponentit itse.

Hyvä esimerkki pitkittyneestä projektista sekä siitä, kuinka projektin laajuus voi paisua mukautusten laajasta määrästä johtuen ja kuinka suunniteltuja käyttötapauksia vastaan toiminen voi johtaa pitkällä aikavälillä katastrofiin on Lidlin surullisenkuuluisa ERP-projekti vuosien 2011 ja 2018 välillä. Projektin tarkoituksena oli ottaa käyttöön SAP-järjestelmä, mutta mukautusten laajan määrän ollessa osatekijä projektin laajuuden kasvuun ja aikataulun venymiseen Lidlin johto päätti lopulta lakkauttaa projektin sen korkeiden kustannusten takia (Agiles, 2018). Projektin päättyessä projektin kokonaiskulut oli asiantuntijoiden mukaan yli 500 miljoonaa euroa (Agiles, 2018).

3.6 Seurantaviikko 6: 11.-15.10.2021

Maanantai 11.10.2021

Tämä päivä alkoi yllättäen Centry-projektin palaverilla, jota en varsinaisesti osannut odottaa. Käyttöliittymäsuunnittelija oli palannut töihin ja pohdimme tiimin kesken yhdessä edellisen viikon löydöksiäni, kävimme läpi käyttöliittymäsuunnitelmia ja sovimme tehtävistä

muutoksista. Sovimme myös, että jatkan uuden viime viikolla alustetun prototyypin työstämistä ja hyödynnän siinä uutta Material UI:n Data-gridiä, jota viime viikolla esittelin.

Aamupäivällä oli myös normaaliin tapaan viikkopalaveri. Palaveri oli lyhyehkö nopea katsaus tähän hetkeen ja kesti vain muutaman minuutin. AgentUP-projektia ollaan siirtämässä vihdoin pilottivaiheeseen, jossa asiakas (Upwire) ottaa projektin laajempaan testikäyttöön. Päivystän muutospyyntöjen varalta ja teen korjauksia mikäli tarpeita ilmenee.

Tiistai 12.10.2021

Tämän päivän aikana oli useita palavereja Centry-projektiin liittyen. Olin ehdottanut aiemmin projektissa käytettävän Material UI:n maksullista DataGrid Pro-kirjastoa, jolle kollegani ehdottivat toista vaihtoehtoa. Kyseessä on Bulgarialaisen Telerik-nimisen yrityksen käyttöliittymäkirjaston nimeltä Kendo-React, joka itselleni ei ollut entuudesta tuttu. Vertailimme näiden kahden ominaisuuksia ja päädyimme sittenkin valitsemaan Telerikin. Kävi myös ilmi, että työnantajallani on olemassa ennestään lisenssejä viime vuodelta Telerikin käyttöliittymäkirjastoon. Komponenttikirjasto sisältää myös suhteellisen kattavasti React-komponentteja. Ihmettelin hieman, miksi niitä ei ole otettu käyttöön työnantajani React-pohjaisissa käyttöliittymissä aiemmin. Sain tehtäväkseni ottaa uuteen projektiin käyttöön Telerikin datataulukon.

Pääosin käytin päivän AgentUP-projektin HallintaUI:n viimeistelyyn. Käyttöliittymän raportointiominaisuuksiin täytyi tehdä vielä muutoksia. Raportteja tulee voida hakea asiakaspalvelijoiden mukaan sekä asiakaspalvelijoista muodostettujen ryhmien mukaan. Näitä varten kollegani oli tehnyt rajapintaan tarvittavat muutokset ja ne täytyi vielä implementoida käyttöliittymässä.

Keskipäivällä minulle tuli vielä Centry-projektin käyttäjähallintaan pieni muutospyyntö, joka hieman keskeytti muut työt, mutta oli nopeasti hoidettu.

Keskiviikko 13.10.2021

Tänään tavoitteenani on palaverien ohella tutustua ja asentaa aiemmin mainitsemani Telerik Kendo React-käyttöliittymäkirjasto ja kokeilla kyseisen kirjaston datataulukon käyttöönottoa uudessa Centry-projektissa. Aamulla käytiin aiheesta palaveri ja sain tehtäväkseni vertailla erilaisia datataulukoita ja selvittää niiden hyviä ja huonoja puolia.

Material UI:n ja Telerikin komponenttien lisäksi kollegani oli ehdottanut AG-Grid-nimistä kirjastoa. Käytin aamupäivän tähän. Kiinnitin huomiota viikoittaisiin latausmääriin,

Githubissa kirjastoille annettuihin tähtiin, päivitysten määrään ja tiheyteen, avoimien bugi-ilmoitusten määrään ja ylipäätään kehityksen aktiivisuuteen. Kävi ilmi, että näillä mittareilla AG-Grid vaikutti olevan näistä kaikista ylivoimaisesti suosituin ja vakain. Ilmoitin löytämäni asiat projektipäällikölle joka vei viestiä eteenpäin. Iltapäivällä korjasin AgentUP-projektiin projektipäällikön löytämiä korjattavia asioita. Käyttöliittymästä oli löytynyt joitakin bugeja ja käytin iltapäivän pääosin niiden setvimiseen.

Torstai 14.10.2021

Tänään aamulla käytiin lisää keskusteluja siitä, mikä datataulukko yritykselle hankitaan ja kävi ilmi, että AG-Gridin lisenssimaksut ovat työnantajalleni liian kovia. Projektipäällikkö käski minun vielä vertailla jäljelle jääneitä Material-UI:n ja Telerikin datataulukoita käyttämällä esimerkkidataa yrityksen omasta projektista, jotta saataisiin selville kummankin hyviä ja huonoja puolia.

Telerikin datataulukon virtualisaatiosta löytyi joitakin epäkohtia, mutta se sisälsi enemmän työnantajani taulukolta toivomia perustoimintoja sarakkeiden ryhmittelyyn, suodatukseen ja järjestelyyn liittyen kuin Material UI:n datataulukko. Projektipäällikkö piti Telerikin ominaisuuksista enemmän joten päädyimme valitsemaan sen. Kummankin käyttöliittymäkirjaston arkkitehtuuri on hieman erilainen ja molemmissa oli omat hyvät puolensa.

Perjantai 15.10.2021

Tänään aamulla ilmoitimme palaverissa muille kollegoille projektipäällikön kanssa johtopäätökset, joihin olimme datataulukoiden suhteen tulleet ja sovimme Telerikin käyttöliittymäkirjaston lisenssien uusimisesta. Kokeilujen aikana oli käynyt ilmi myös, että vanhoilla lisenseillä ei enää juurikaan tee mitään, koska niiden umpeutumisen jälkeen on tullut niin paljon korjauksia, ettei huonommin toimivaa vanhempaa versiota ole järkevää käyttää.

Aamupäivällä oli henkilöstöpalaveri, jossa käytiin läpi yrityksen taloustilanne, tilanne koronarajoitusten suhteen, yhteiset pelisäännöt työtapojen suhteen ja tulevaisuudennäkymiä eri projektien suhteen laajemmassa mittakaavassa. Kävimme läpi myös tulevia yrityksen jäsenilleen järjestämiä tapahtumia.

Tämän jälkeen oli myös Centry-projektin palaveri, jossa käyttöliittymäsuunnittelija näytti tekemiään ulkoasusuunnitelmia ja sain tehtäväkseni alkaa toteuttamaan suunnitelmia käytännössä. Palaverissa päädyimme asentamaan tähänastisen käyttäjien hallinnan yhden asiakkaan palvelimelle. Matkassa tuli hieman mutkia vastaan ja palaveri venyi aika pitkäksi niitä selvitellessä. Iltapäivällä asensin asiakkaan ympäristöön uusimmat versiot AgentUP-projektin käyttöliittymistä.

Viikkoanalyysi

Usean projektin välillä vaihteleminen tuntuu ajoittain hyvin kuormittavalta. Viestintä tuntuu kulkevan myös hyvin paljolti ylhäältä alaspäin ja johtamistyyli on pääsääntöisesti delegoivaa tukevan tai ohjeistavan sijaan. Projektipäälliköt odottavat minulta tuloksia ja viestivät odotuksensa suoraan minulle. Tällöin myös paine tuntuu kasautuvan minulle kehittäjänä.

Projektipäälliköitä ei tunnu kiinnostavan niinkään tekniset yksityiskohdat tai se, *miten* asiat saadaan tehtyä tai valmiiksi. Tuloksia on vain saatava jotenkin. Tuntuu vaikealta pitää omia puoliaan silloin kun muut kauemmin yrityksessä olleet ja kokeneemmat vaativat tuloksia jatkuvasti yhä kiihtyvään tahtiin. Olen keskustellut kokeneemman ohjelmistoalalla olevan ystäväni kanssa tilanteestani, jossa joudun olemaan kahden tai useamman tulen välissä eri projekteissa. Hänen mukaansa välissä kuuluisi olla joku teknisesti kokeneemmista ottamassa painetta vastaan ja asettamassa eri osapuolten odotuksia realistiseksi, jolloin uudemmat ja vähemmän kokeneet saisivat paremman rauhan keskittyä työhönsä.

AgentUP-projektin aikataulu on venynyt mielestäni laajuuden venymisen takia. Asiakkaalta on tullut muutospyyntöjä toisensa jälkeen ja ne on otettu mukisematta vastaan. Projektin aikana sen vetäjä on vaihtunut useaan otteeseen, eikä mielestäni sen kokonaisuuden hallinnasta ole otettu tarpeeksi kokonaisvaltaista vastuuta. Projektia vetämään on ylennetty henkilö, joka on toiminut kehittäjänä pitkään ja omaa paljon teknistä kokemusta, mutta jolla ei ole projektin johtamisesta paljoa kokemusta.

Oman haasteensa AgentUP-projektiin on etenkin Hallinta-UI:n kohdalla tuonut myös ennestään projektissa valmiina olleen koodin määrä ja sen huono laatu. Huonolla laadulla tarkoitan mm. muuttujien ja tiedostojen epäselvää ja epäjohdonmukaista nimeämistä, liiallista TypeScriptin any-tyypin käyttöä joka altistaa virhetilanteille, epäjohdonmukaista tilanhallintaa, *imperatiivisen* ohjelmoinnin käyttöä React-projektissa ja komponenttiarkkitehtuurin sekavuutta. Edellä mainitut asiat hankaloittaa koodin ylläpitämistä ja muutosten tekemistä ja toisinaan muutosten tekeminen minimissäänkin on vaatinut refaktorointia, joka vie aikaa.

Eräs juurisyy syntyneelle tilanteella on nähdäkseni puutteellinen laadunvalvonta. Projekteissa ei käytetä esim. koodikatselmointeja, joissa tarkasteltaisiin kirjoitettua koodia tiimin kesken laadunvarmistamiseksi, vaan kehittäjien annetaan kirjoittaa koodia vapaasti itsekseen. Tämänkaltainen työtapa projekteissa antaa mielestäni ikävästi hyvän kasvualustan huonolaatuiselle koodille, johon pääsee pesiytymään sudenkuoppia.

3.7 Seurantaviikko 7: 18.-22.10.2021

Maanantai 18.10.2021

Tänään aloitin työt korjaamalla AgentUP-projektin yhtä bugia, joka liittyi raportoinnin ominaisuuksiin. Käyttöliittymästä voi hakea erilaisia raportteja valitsemilleen ajanjaksoille tuottavuuteen, kontakteihin, tilauksiin, tilastoihin ja viestintään liittyen. Raportteja voi myös suodattaa, viedä Exceliin tai hakea kampanjakohtaisesti.

Olin tehnyt raporttien hakuun viime viikolla muutoksia, joissa annetaan mahdollisuus hakea raportit asiakaspalvelijakohtaisesti tai niistä muodostettujen ryhmien mukaan. Asetusten valintaan oli tullut muutoksia ja ne eivät toimineet ihan niin kuin piti. Ongelmat liittyivät vanhan alkuperäisen koodin sekavuuteen. Komponenttien sisältämä tila oli hajallaan komponenttipuun eri tasoilla tehden koodista vaikeaselkoista ja muutoksista hankalia. Normaalin viikkopalaverin ohella käytin aamupäivän koodin siistimiseen ja tilanhallinnan selkeyttämiseen.

Iltapäivällä jatkoin Centry-projektin uuden käyttöliittymän alustavan työn kanssa. Käytän tilanhallintaan Redux-nimistä kirjastoa. Iltapäivä kului tilanhallintaan liittyvien funktioiden ja rajapintakutsujen kirjoittamiseen. Päätin hieman uudistaa tapaa, jolla sovelluksen tilan hallinta toteutetaan tässä projektissa. Otin käyttöön Redux Toolkit-nimisen kirjaston, joka ehostaa Reduxin perustoiminnallisuuksia hieman moniulotteisemmaksi ja mahdollistaa mm. asynkronisten rajapintakutsujen tekemisen tilanhallinnasta käsin. Tila on myös ns. ”viipaloitu” (*engl. slice*) eri osiin, jolloin eri osa-alueisiin liittyvät asiat ovat omissa paikoissaan.

Tiistai 19.10.2021

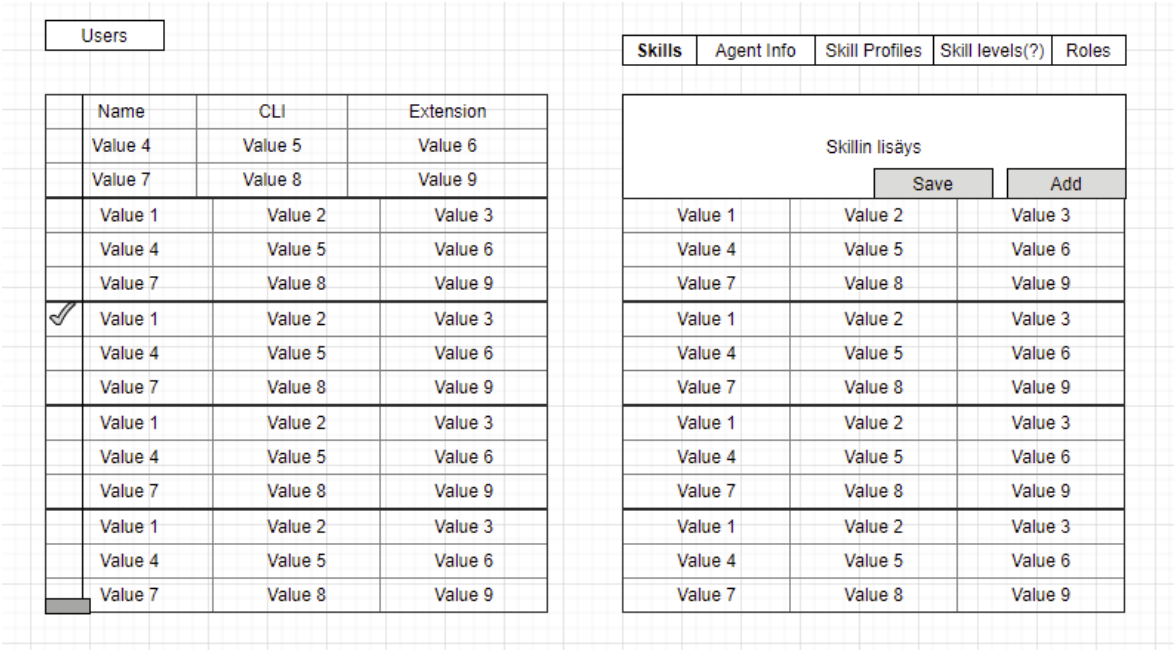
Tämän päivän käytin pääasiassa ohjelmointiin. Redux Toolkitin käyttöönotto ei sujunut täysin ongelmitta ja jouduin käyttämään hieman aikaa ongelmien selvittelyyn. Virheviestit eivät kertoneet paljoa ja jouduin selvittelemään virhetilanteita käsin kohtuullisen kauan.

Sain kuitenkin lopulta tilanhallinnan ja rajapintakutsut toimimaan niin kuin halusin. Syyksi osoittautui lopulta sovelluksen tilan lukemisen yrittäminen väärällä tavalla rajapintakutsuja tehdessä. Nyt ongelmalliseksi havaitsemani tapa toimi edellisten projektien arkkitehtuureissa, mutta tämä toteutus toimii hieman eri tavalla.

Saatuani asynkroniset rajapintakutsut toimimaan tilanhallinnasta käsin niin kuin halusin, käytin lopun iltapäivän käyttöliittymäkomponenttien hahmotteluun. Yritän tehdä komponenteista niin uudelleenkäytettäviä kuin mahdollista ja aloitin toteuttamaan datataulukkoa, jota voi käyttää useiden eri muotoisten objektien listaamiseen ja käsittelyyn. Typescriptin ominaisuudet tarjoavat mm. *geneerisiä* tyyppejä, jotka voidaan antaa parametrina taulukkokomponentille, jolloin komponentin itsensä ei tarvitse ottaa kantaa siihen minkä tyyppistä dataa se käsittelee.

Keskiviikko 20.10.2021

Tänään otin uudessa Centry-projektissa käyttöön protobuf-kyselyitä joita rajapinta, joka hallitsee Speakex-puhealustaa hyödyntää. Olkoon rajapinnan nimi myöhemmin **Management-API**. Näillä kyselyillä hallitaan muuttujia, kuten puhelinjärjestelmän käyttäjiä (nk. *Agentteja)* ja palvelukanavia (nk. *Skillejä).* Käyttäjälle lisätyt palvelukanavat määrittävät sen, millaisia töitä käyttäjälle ohjautuu.



Kuva 2: Hahmotelma Centry-projektin käyttäjähallinnasta

Käyttäjien hallintaan keskittyvä näkymä on suunniteltu kokonaan uudestaan ja näkymä on jaettu kahteen osaan, joissa toisessa listataan käyttäjät datataulukossa ja toisessa hallitaan käyttäjän ominaisuuksia. Käyttäjään liittyvät tiedot ovat jaoteltu eri välilehdille. Tämän uuden käyttöliittymän tarkoitus on yhdenmukaistaa sitä tapaa, jolla puhejärjestelmän elementtejä hallinnoidaan. Vanhoissa työnantajani työpöytäsovelluksissa ominaisuudet ovat hieman hajallaan toisistaan tehden kokonaisuudesta hyvin epäselvän.

Olin päivän aikaansaannoksiin tyytyväinen. Uskon, että mikäli keskityn kirjoittamaan niin hyvää koodia kuin osaan, vältän koodin kopioimista ja karsin laiskuudesta johtuvaa

huolimattomuutta, projekti skaalautuu turvallisesti ja ylläpidettävästi. Ihannetilanne jota tavoittelen on se, että koodin laatu paranisi ajan myötä paisumisen ja kasautuvien ongelmien sijaan. Uutta projektia työstäessäni yritän myös keskittyä havaitsemaan ongelmallisia rakenteita mahdollisimman aikaisessa vaiheessa, koska alkuvaiheessa ne on helpompi purkaa tai muotoilla uudelleen.

Torstai 21.10.2021
Tällä viikolla yksi projektipäälliköistä on lisännyt minut aamuisiin palavereihin, johon osallistuu suurin osa kehittäjistä tällä hetkellä. Aamupalaverin jälkeen pidimme projektipäällikön ja käyttöliittymäsuunnittelijan kanssa pienemmän palaverin Centry-projektiin liittyen.

Esittelin hieman tähänastista työtä, keskustelimme pikaisesti seuraavista askelista ja sovimme että alan seuraavaksi tuomaan käyttäjähallinnan ominaisuuksia projektiin. Erona aiempaan kesällä tehtyyn käyttäjähallinnan versioon on se, että Telerikin datataulukko mahdollistaa suoraan taulukkoon tietojen muokkaamisen. Aiemmassa käyttöliittymässä tietoja hallittiin erilliseen dialogiin aukeavalla lomakkeella. Suoraan datataulukkoon tietojen muokkaaminen tekee muutoksien tekemisestä suoraviivaisempaa ja käyttäjälle helpompaa.

Mahdollisia haasteita muutoksiin liittyen on datataulukon validoiminen. Taulukon ominaisuudet eivät vaikuttaneet suoraan tarjoavan helppoa tietä arvojen syöttökenttien validoinnille. Validointia tapahtuu toisaalta myös rajapinnalla, joten päätin huolehtia tästä myöhemmin. Mikäli tarpeita tiukempaan oikeellisuuden varmistamiseen käyttöliittymässä ilmenee, on taulukkoon mahdollista asettaa mukautettuja komponentteja tai muokata datataulukon arvoja myös lomakkeilla. Lähetettävien objektien validoinnin voi toteuttaa myös datataulukosta erillään ennen lähetystä siihen tarkoitetulla kirjastolla nimeltä *Yup*.
Päivän päätteeksi joidenkin käyttöliittymäominaisuuksien lisäksi olin toteuttanut käyttäjien lisäämisen ja niiden muokkauksen. Olin aikaansaannoksiin suhteellisen tyytyväinen.

Perjantai 22.10.2021
Tänään päivä alkoi normaaliin tapaan aamupalaverilla, jossa käydään läpi projektien tilanteet ja mitä kukakin edistää. Päästessäni jatkamaan ohjelmointia lisäsin Centry-projektin datataulukkoon käyttäjien oikeudet sovelluksiin. Taulukkoa voi muokata ja näin ollen myös sovellusoikeudet voi kytkeä käyttäjille joko päälle tai pois helposti. Keskustelin kollegani kanssa, kuka hoitaa rajapinnan kehitystä ja sovimme muutoksista, joita rajapintaan on tehtävä, jotta käyttöliittymän vaatimukset täyttyisi.

Olen ottanut tässä projektissa käyttöön Eslint-työkalun avulla hieman tiukemmat vaatimukset koodin tyylin suhteen. Käytän JavaScript-yhteisön keskuudessa tunnettua

konfiguraatiota, jonka *Airbnb* on kehittänyt. Tiukempi Eslint-konfiguraatio auttaa pitämään koodia siistimpänä ja luettavampana, jolloin pitkän ajan kuluessa koodiin ei kerry niin paljoa sotkua.

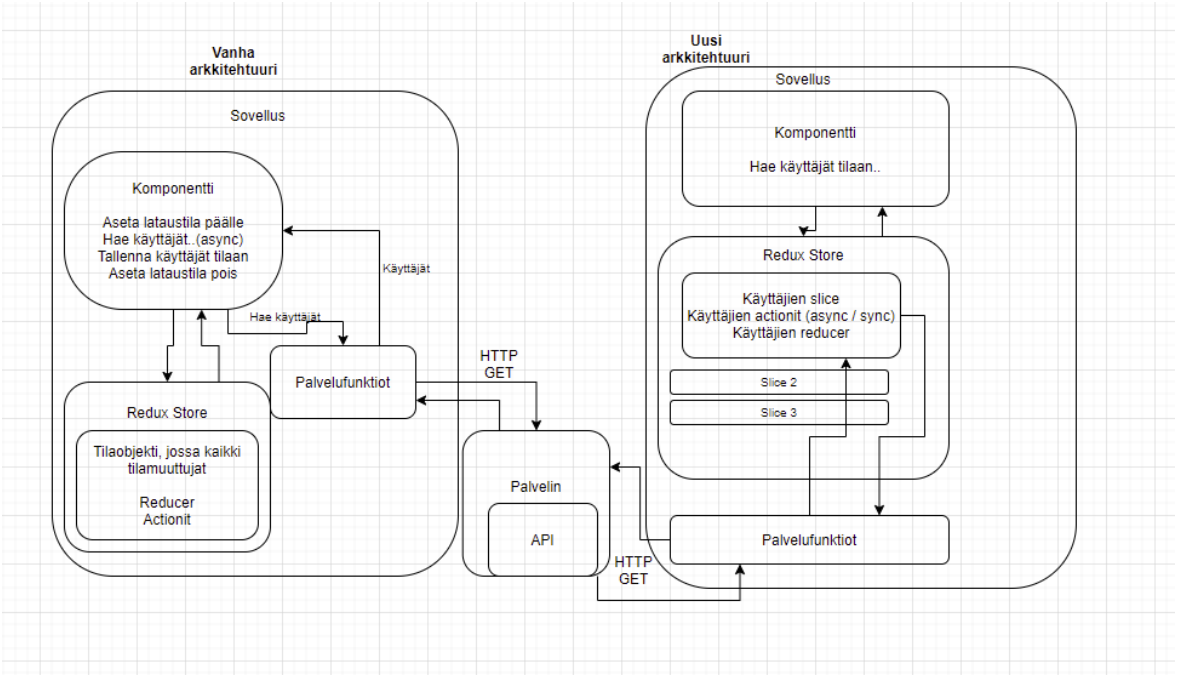
Tyylisäännöt ohjaavat myös mahdollisia tulevia muita projektin kehittäjiä kirjoittamaan tyylipuhdasta koodia. Varmistaakseni sen, että sääntöjä noudatetaan asensin projektiin myös husky-nimisen kirjaston. Husky tunnistaa tilanteet, joissa koodia kommitoidaan (*engl. commit*) joko versionhallintaan tai lähetetään repositorioon (*engl. push*). Ennen tallennusta Husky ajaa sille määritetyt komennot. Asetin Huskyn ajamaan tallennuksen yhteydessä Eslint-tarkistuksen joka tarkistaa, että koodissa ei ole tyylivirheitä. Jos tyylivirheitä on, tallennus keskeytetään. Lähetyksen (*push*) yhteyteen asetin Huskyn ajamaan projektin testit. Näin projektin kehitystyökalut estävät huonolaatuisen koodin päätymistä repositorioon ehkäisten omasta ja muiden huolimattomuudesta johtuvia virheitä.

Kirjoitin myös joitakin yksikkötestejä sekä sovelluksen komponenttipuun ylimmälle tasolle nk. savutestin, joilla joitakin sovellukseen mahdollisesti päätyviä suorittamisen keskeyttäviä virheitä saadaan kiinni. Aiheutin tietoisesti virhetilanteita, jotka saivat käyttöliittymän kaatumaan ja kirjoitin testin, joka paljasti virheet.

Viikkoanalyysi

Olin suhteellisen tyytyväinen viikon aikaansaannoksiin Centry-projektin suhteen. Olen yrittänyt kirjoittaa niin huolellista ja hyvin jäsenneltyä koodia kuin osaan. Redux-Toolkitin käyttö on mahdollistanut Rajapintakutsuihin liittyvän tiedon lähetyksen ja vastaanottamisen kirjoittamisen tilanhallinnan yhteyteen ja asynkronisten tilapäivitysten (*engl. action*) tekemisen.

Asynkronisen tilanhallinnan lisääminen on auttanut pitämään käyttöliittymän komponenttien koodia paljon siistimpänä, koska asynkronisten rajapintakutsujen tekemiseen liittyvää koodia ei näin ollen täydy kirjoittaa komponenttien sekaan. Tilan jakaminen omiin ”viipaleisiin” (*engl.slice*) myös auttaa pitämään arkkitehtuuria modulaarisempana, jolloin tila säilyy selkeämpänä, itsenäisiä sovellustilan osa-alueita on helpompi ylläpitää ja uusien tilamuuttujien lisääminen on helpompi toteuttaa sotkematta tai paisuttamatta vanhaa koodia.



Kuva 3. Aiemman Centry-projektin käyttöliittymän Redux-arkkitehtuuri ja uusi Redux-Toolkitin tarjoama arkkitehtuuri

Sovelluksen kokonaisarkkitehtuuri säilyy myös selkeämpänä ja suoraviivaisempana. Kun rajapintakutsuihin liittyvää logiikkaa tai tilanhallinnan logiikkaa ei tarvitse sotkea komponenttien koodin sekaan on komponentit helpompi pitää ”tyhmempinä” ja vapaampina sivuvaikutuksista. Redux Toolkitin tarjoama monipuolisempi tilanhallinta myös vähentää komponenteissa ylläpidettävän tilan tarvetta, jolloin niissä voi keskittyä puhtaammin käyttöliittymään liittyviin asioihin.

Olen myös onnistunut pitämään mielestäni tähän asti komponenttien vastuualueet selkeämmin jaoteltuina ja tehnyt komponenteista uudelleenkäytettävämpiä. AgentUP-projektin kohdalla tuntui, että jokaiseen matkan varrella ilmenneeseen tarpeeseen tuli kehitettyä omat komponenttinsa ja näkymänsä sen sijaan, että olisi alun perin suunniteltu yksi useita samankaltaisia käyttötarpeita palveleva näkymä.

Tasapainottelu uudelleenkäytettävyyden ja mukautettavuuden välillä on vielä asia, jota työskennellessäni hieman etsin. Lähestyn asioita siitä lähtökohdasta, että pyrin tekemään React-komponenteista mahdollisimman uudelleenkäytettäviä, jotta samaa työtä ei tarvitsisi tehdä montaa kertaa uudestaan. Tässä projektissa olen myös hyödyntänyt Typescriptin *geneerisiä* tyyppejä, jotka mahdollistaa mm. sen, että funktionaalisen komponentin ei tarvitse tietää kaikkea sen käsittelemien arvojen tyypistä vaan ne voidaan antaa nk. *tyyppiparametreina.*

Geneeristen tyyppien käyttö on mahdollistanut mm. saman datataulukon käyttämisen useiden erimuotoisten objektien käsittelyyn, luomiseen, muokkaukseen ja poistoon.

3.8 Seurantaviikko 8: 25.-29.10.2021

Maanantai 25.10.2021
Viikko alkoi tavalliseen tapaan aamuisilla palavereilla. Tähän asti käytössä on ollut frontend-kehittäjille yhteiset viikkopalaverit. Esimiehet ja tiiminvetäjät haluavat keskittää palavereita vähentääkseen niiden määrää, ja tällä hetkellä aamupalavereihin osallistuu muitakin kuin frontend-kehittäjiä. Kävimme pikaisesti läpi kaikkien kuulumiset jonka jälkeen jäin parin muun kanssa puhumaan hetken Centry-projektin tilanteesta ja tehtävien asioiden tärkeysjärjestyksestä.

Projekti on lähtenyt yllättävän hyvin mielestäni käyntiin. Projektin kokonaiskuva ja vaatimukset voisivat silti olla mielestäni selvempiä. Kävimme aamupalaverin jälkeen läpi seuraavia toteutettavia asioita ja palasin sitten ohjelmoinnin pariin.

Puhejärjestelmän käyttäjille täytyi voida hallinnoida palvelukanavia nk. *skillien* kautta. Käytin tämän päivän niiden hallinnoimisen toteuttamiseen. Operaatiot tapahtuvat protobuf-kyselyillä aiemmin mainitsemalleni Management API:lle. Protobuf-kyselyjen tekeminen tiedon hakuun aiemmin oli suhteellisen yksinkertaista, mutta tiedon muokkaus ei ollutkaan niin itsestäänselvää. Pyysin kokeneemmalta kollegaltani apua ja hän auttoi ongelmien selvittelyssä. Päivän päätteeksi olin saanut kyselyt onnistumaan ja vaikein osuus palvelukanavien muokkauksessa oli ohi.

Tiistai 26.10.2021
Tänään aamupalaverin jälkeen palasin jatkamaan puhejärjestelmän käyttäjien palvelukanavien hallintaa. Eilen olin saanut protobuf-kyselyt tehtyä ja nyt toiminnot täytyi vain lisätä käyttöliittymään. Sain toiminnot toimimaan niin kuin halusin käyttöliittymässä. Lopputulos oli mielestäni paljonkin näppärämpi käyttää, kuin aiempi työpöytäkäyttöliittymä, jolla on hallittu Speakex-puheluastaa.

Protobuf-kyselyt toiminnassa osoittautuivat myös hyvin nopeiksi ja tehokkaiksi. Binäärimuotoinen data on pienemmässä koossa ja kutsut paljon nopeampia lähettää kuin perinteinen JSON. Sain keskipäivään mennessä palvelukanavien hallinnan valmiiksi.

Iltapäivän käytin seuraavaan ominaisuuteen, joka oli käyttäjäkohtaisten sovellusoikeuksien hallinta. Oikeuksia hallitaan niistä muodostettujen oikeusluokkien kautta, joita sitten

lisätään käyttäjille. Toteutin siis sen, että käyttäjille voi lisätä tai poistaa oikeusluokkia. Sain ominaisuuden valmiiksi päivän loppuun mennessä.

Keskiviikko 27.10.2021

Tänään tavoitteenani on lisätä Centry-projektiin toiminnallisuudet varsinaisten oikeusluokkien luomiseen, muokkaukseen ja poistoon. Oikeusluokat sisältävät autentikaatiopalveluun luotuja nk. *rooleja (engl. roles)*, jotka taas puolestaan sisältävät niihin asetettuja oikeuksia ominaisuuksiin (*engl. permissions*). Roolien ja oikeuksien ollessa taulukoita ne muodostavat sisäkkäisiä listarakenteita. Tämä on lomakkeen tekemisen kannalta haaste.

Kuten aiemmilla viikoilla on tullut ilmi, monimutkaisten lomakkeiden teko ei ole Reactissa aina helppoa. Olen kuitenkin toteuttanut näiden oikeuksien hallintaan lomakkeen, jota päätin hyödyntää tässä projektissa myös. Käytin kesällä lomakkeen tekemiseen useita tunteja ja oli työn takana saada se toimimaan niin kuin piti. Kahden sisäkkäisen listan muokkaaminen ei ole kovin yleinen käyttötarve, joten mielestäni ei kannata käyttää aikaa tämän ominaisuuden uudelleentoteuttamiseen turhaan. Käytin tämän päivän pääasiassa oikeusluokkien listausnäkymän tekoon ja oikeusluokiin kuuluvien roolien muokkauslomakkeen tekemiseen.

Torstai 28.10.2021

Tänään aamulla jatkoin oikeusluokkien hallinnan toteuttamista Centry-projektissa. Aamupalaveri oli poikkeuksellisesti hieman myöhemmin, koska eilen oli ollut virkistyspäivä. Jatkoin oikeusluokkien tekoa palaverin jälkeen. Hyödynsin käyttöoikeuksien muokkaamisessa aiemmin tekemääni lomaketta, jonka siirsin tähän projektiin. Minun oli kuitenkin tehtävä jonkin verran muutoksia, että sain lomakkeen toimimaan.

React itsessään tarjoaa lomakkeiden tekemiseen sekä niiden sisäisen tilan hallintaan kyllä työkaluja, mutta lomakkeiden ollessa monimutkaisempia koodista tulee helposti toisteista ja sen kirjoittamisesta ikävää. Suositut lomakekirjastot kuten Formik ja React hook form tarjoavat mielestäni hyviä ja joustavia ratkaisuja monipuolisten lomakkeiden tekoon hieman vaivattomammin. Olen kokeillut monia eri tapoja, mutta huomannut samalla, ettei ole olemassa yhtä oikeata tapaa, joka sopisi kaikkeen. Lomakkeessa, jota tähän projektiin otin käyttöön on käytössä Formik. Iltapäivällä kohensin projektin ulkoasua hieman enemmän designin mukaiseksi. Pienetkin muutokset saivat aikaan suhteellisen positiivisia vaikutuksia kokonaisuuteen.

Perjantai 29.10.2021

Tänään aamupalaverissa projektipäällikkö antoi minulle tehtäväksi muuttaa Centry-projektin datataulukkojen toimintoja niin, että rivien muokkaamisen sijaan käytettäisiin solumuokkausta, jonka jälkeen kaikki rivit joissa on muutoksia tallennetaan.

Rivikohtainen muokkaus oli yksinkertaisempi toteuttaa, sillä taulukoissa listataan rajapinnalta haettuja objekteja, joista jokainen on yksi rivi. Rajapintaan päivitetään myös objekteja yksi kerrallaan, joten rivikohtainen muokkaus käy helpommin rajapintakutsujen kanssa yksi yhteen. Kerroin, että solukohtainen muokkaus teettää kyllä jonkin verran lisätyötä, mutta se ei haitannut muita.

Olin tähän mennessä toteuttanut datataulukkokomponentin, joka oli nyt käytössä neljässä eri sovelluksen kohdassa, jossa muokattiin asioita rivi kerrallaan. Muutosten kanssa kävi nyt hieman niin, että kun yksi puu kaatuu toisaalla, niin toinen toisaalla.

Käytin tämän päivän ohjelmoinnin osalta kokonaan solukohtaisen muokkauksen toteuttamiseen. En saanut sitä kokonaan valmiiksi, mutta hyvään vaiheeseen kuitenkin. Saadessani ominaisuuden suurimmalta osin toimimaan, olen yhtä mieltä projektipäällikön kanssa siitä, että käyttökokemus kuitenkin koheni. Huomaan toisinaan ajattelevani hieman puolueellisesti ja puoltavani ratkaisuja, jotka vaikuttavat helpommilta toteuttaa.

Viikkoanalyysi
Tällä viikolla Centry-projekti on edennyt hyvän matkaa eteenpäin. Olen oppinut myös, kuinka enkoodataan Protocol buffers (Protobuf) -kutsuja binäärimuotoon ja lähetetään niitä palvelimelle tiedon muokkausta varten. Tämän lisäksi työskentely havainnollisti käytännössä, kuinka protobuf-kutsut tarjoavat hyvin tehokkaan tavan viestintään sovelluksen ja palvelimen välillä. Kyselyiden ollessa muunnettuna binäärimuotoon suoraan ihmiselle luettavien JSON-objektien sijaan protobuf lisää myös ylimääräisen kerroksen suojaa lähetettäville tiedoille, vaikkei se suojaakaan kyselyjä ihan kaikelta.

Centry-projektin käyttöliittymään on nyt saatu jo toteutettua käyttäjähallinnan ominaisuuksista suurin osa, eli käyttäjien luominen, muokkaus ja poisto. Tämän lisäksi on lisätty käyttäjien palvelukanavien, eli *skillien* hallinta.

Olen myös huomannut, kuinka tiukempi tyypitys Typescriptiä käytettäessä kannattaa, sillä muutosten yhteydessä kehitystyökalut osaavat ilmoittaa muutoksista aiheutuneiden virheiden olemassaolosta. Tämä tekee refaktoroinnista monta kertaa helpompaa verrattuna siihen, että ei ole jaksettu tyypittää koodia kunnolla ja dynaamista tyypitystä sallitaan any-tyypin avulla, jolloin koodi käyttäytyy enemmän kuten perinteinen Javascript.

Dynaamisen tyypityksen käyttö voi joissain tilanteissa helpottaa kehitystyötä, varsinkin tehdessä nopeita kokeiluja, mutta laajojen projektien kohdalla sitä kannattaa käyttää harkiten ja vain silloin kun on pakko. Liiallinen käyttö johtaa siihen, että virheitä pääsee pesiytymään helpommin tehden koodista paljon alttiimpaa niille. Silloin muutoksia tehdessä kehitystyökalut eivät osaa varoittaa siitä, että muutokset vaikuttavat myös toisaalla.

Laajempi ja monipuolisempi tyyppien käyttö projektissa opettaa myös ottamaan enemmän Typescriptin tarjoamia hyötyjä irti ja pakottaa oppimaan lisää erilaisten tyyppien käytöstä erilaisiin tilanteisiin. Typescriptin paras ja toisinaan myös ikävä puoli on se, että kehittäjä saa halutessaan säädellä sitä, kuinka tiukkaa tyypitystä käytetään. Vastuu tyypityksen hyödyntämisestä on siis kehittäjällä itsellään. Tyyppijärjestelmä tarjoaa mielestäni mahdollisuuden parhaimmillaan erittäin miellyttävälle kehittäjäkokemukselle ja hyvän tukirangan sovelluksen toiminnalle. Toisinaan tyypityksen opettelemiseen menee hieman aikaa ja se voi tuntua vaivalloiseltakin, mutta ajan myötä tämä on helpottanut. Olen sitä mieltä, että pitkällä juoksulla vaivannäkö kuitenkin kannattaa, koska se myös säästää paljon aikaa ja ehkäisee useita virhetilanteita, joiden selvittelyyn voi kulua pitkiäkin aikoja.

Telerikin datataulukon käyttöönotto on myös ollut omalta osaltaan opettavaista. Uusien teknologioiden käytön opettelu kehittää ja syventää osaamista, koska itselle ennalta tuntemattomien teknologioiden käyttö on yleensä aluksi kaikista haastavinta. Uuden opettelu myös virkistää ja tekee yleensäkin mielelle hyvää.

3.9 Seurantaviikko 9: 01.-05.11.2021

Maanantai 01.11.2021
Viikko alkoi normaalisti aamuisella tuotekehityspalaverilla.
Centry-projektin datataulukoihin täytyi toteuttaa solukohtainen muokkaus. Olin viime viikon puolella aloittanut toiminnallisuuden tekoa ja jatkoin siitä mihin viimeksi jäin.

Solukohtainen editointi vaikutti myös rajapintakutsuihin. Aiemmin yhtä riviä muokattaessa kyseinen rivi tallennettiin ja lähetettiin objektina rajapinnalle päivitettäväksi HTTP PUT-kutsun mukana. Nykyinen toiminnallisuus vaati sen, että tallennuksen yhteydessä tarkistetaan, mitkä rivit ovat muuttuneet ja lähetetään jokainen niistä päivitettäväksi. Yhden kutsun sijaan täytyy siis tehdä useampia kutsuja, koska tällä hetkellä rajapinta tukee vain yksittäisten objektien päivitystä. Sain toiminnon pitkälti valmiiksi aamupäivän aikana.

Esittelin iltapäivällä käyttöliittymää projektipäällikölle ja sain todella tyytyväistä palautetta työni tuloksista. Olin kohentanut ulkoasua hieman, joka teki positiivisen vaikutuksen.

Tiistai 02.11.2021
Päivä alkoi tuotekehityspalaverilla. Esittelin käyttöliittymää aamun tuotekehityspalaverissa myös muille kollegoille. Samalla kävi pieni demoefekti, joka paljasti yhden käyttöliittymään jääneen bugin. Käytin aamupäivän sen selvittelyyn ja korjaamiseen.

Syy löytyi sovelluksen tilasta. Selaimessa suoritettavalla ohjelmalla on tila, johon rajapinnalta haettu data tallennetaan. Tässä vaiheessa sovelluksessa hyödynnetään kahta rajapintaa: Käyttäjien hallintaan tarkoitettua rajapintaa ja aiemmin mainitsemaani nk. Management-rajapintaa, jolla hallitaan puhealustaa (Speakex) ja sen osia.

Käyttäjähallinnan rajapinta on kehitetty helpottamaan käyttäjien luomista vähentäen manuaalista työtä. Tämä on toteutettu niin, että rajapinta keskustelee autentikaatiopalvelun kanssa, jossa varsinaiset sovellusten käyttäjät sekä käyttöoikeudet sijaitsee sekä management apin kanssa, jossa puhejärjestelmän käyttäjät (*Agentit*) sijaitsee.

Kun käyttäjähallinnan rajapinnalle tehdään kutsu, joka sisältää muutoksia käyttäjän tietoihin, myös agentti päivitetään, mikäli käyttäjälle on liitetty sellainen. Agenteilla on tieto versiosta, joka kertoo, onko kyseinen objekti ajan tasalla kun sitä yritetään muokata. Käyttöliittymään oli käyttäjään tehtyjen muutosten jälkeen jäänyt siis vanhentunut versio Agentista. Käyttäjään liitetty Agentti täytyi siis päivittää sovelluksen tilaan myös käyttäjää muutettaessa.

Keskiviikko 03.11.2021
Päivä alkoi tuotekehityspalaverilla. Minua pyydettiin tekemään muutoksia yhteen projektiin, jota olin tehnyt viimeksi kesällä. Kyseinen tuote ollaan viemässä asiakkaalle lähiaikoina ja toimitus tehdään Docker-pohjaisena.

Kyseessä on chat-sovellus, jota asiakaspalvelijat käyttävät palvelukanavana mahdollistaen WhatsApp-yhteensopivan chat-asiakaspalvelun. Yhteys WhatsAppin palvelimiin tapahtuu ulkoisen palveluntarjoajan rajapintojen kautta.

Käyttöliittymän konfiguraatioihin oli tehtävä muutoksia samaan tapaan, kuin muihinkin Docker-yhteensopiviksi tehtyihin käyttöliittymiin. Sovelluksen konfiguraatiot on voitava antaa dynaamisesti JSON-tiedostosta, jotta frontendia ei tarvitse paketoida erikseen eri ympäristöjä varten, vaan samaa versiota voi käyttää eri ympäristöissä vaihtamalla vain JSON-tiedoston konfiguraatiotietoja. Docker-perustaiset toimitukset tulee voida myös hostata käänteisen välityspalvelimen takaa. Tällöin tuleva http-liikenne ohjataan oikeaan ohjelmistoon selaimesta saatavan osoitteen perusteella, jolloin käyttöliittymän täytyy tukea

etuliitettä käyttämässään osoitteessa. Oletuksena React-sovellukset etsii tarvitsemiaan staattisia tiedostoja osoitteen juuripolun alta.

Torstai 04.11.2021

Aamu alkoi tuotekehityspalaverilla. Tämän päivän käytin ohjelmoinnin osalta pitkälti Centry-projektiin. Puhealustan käyttäjille on tähän mennessä toteutettu palvelukanavien hallinta, jossa käyttäjille voi lisätä palvelukanavia ja poistaa niitä. Näitä hallitaan Management-apin kautta. Palvelukanavista on voitava muodostaa erilaisia profiileita (Skill profile), joita asiakaspalvelijat voivat hyödyntää itsenäisesti töittensä ohella. Asiakaspalvelijan on voitava valita tilanteeseensa sopiva profiili, joka sisältää tilanteeseen sopivat palvelukanavat (esim. vain puhelut, sähköposti ja puhelut, vain chat).

Kehittämästäni käyttöliittymästä on siis voitava asettaa käyttäjille kyseisiä profiileja ja poistaa niitä käyttäjiltä. Tämän lisäksi itse profiileja on voitava hallita, eli hakea, luoda, muokata ja poistaa omassa näkymässään. Nämä ominaisuudet backendin osalta eivät ole toteutettu käyttäjähallinnan rajapintaan eikä management rajapintaan, vaan nämä ominaisuudet on toteutettu kolmanteen rajapintaan, nk. settings-rajapintaan. Ominaisuudet eivät ole rajapinnankaan osalta aivan valmiit, mutta siinä vaiheessa, että voin alkaa toteuttaa käyttöliittymän ominaisuuksia. Käytin loppupäivän niiden aloittamiseen.

Perjantai 05.11.2021

Aamuisen tuotekehityspalaverin jälkeen jatkoin töitäni nk. *Skill profiilien* ominaisuuksien parissa. Eilen olin toteuttanut itse profiilien hallintaan liittyviä käyttöliittymän ominaisuuksia. Tänään keskityin käyttäjään liittyvien profiilien hallintanäkymään, jossa käyttäjälle voi lisätä olemassa olevia profiileja tai poistaa niitä. Käyttäjän profiilien hallintanäkymään täytyi lisätä myös linkit profiilien muokkausnäkymään kunkin profiilin kohdalle. Hyödynsin tässä sovelluksessa jo valmiiksi käytössä olevaa React-routeria, joka hoitaa sovelluksen reitityksen eri näkymien välillä sovelluksen tilaan ja osoitekentän tietoihin perustuen. Näkymän vaihdon yhteyteen on mahdollista antaa parametrina muokattavan profiilin yksilöivä tieto eli id, joka ohjaa oikeaan paikkaan.

Iltapäivällä jatkoin Skill-profiilien hallintanäkymää profiilien sisältämien palvelukanavien (*engl. Skill)* osalta. Toteutin toimintoja, joilla profiilien sisältämiä palvelukanavia voi listata, valita poistettavaksi tai lisätä. Sain toiminnot iltapäivän aikana alustavaan vaiheeseen. Joudun jatkamaan niitä ensi viikolla.

Viikkoanalyysi

Centry-projektin käyttöliittymä käyttää nyt kolmea eri rajapintaa (Management API, User Management API, Settings API), joten liikkuvia osia on kertynyt jo jonkin verran. Tästä on

aiheutunut jo joitakin ongelmia tähän asti, mm. sovelluksen tietojen ajantasaisena pitämisen vaikeus, koska tietoja pitää ylläpitää useaan paikkaan.

Googlasin hieman aiheesta, ja yksi mahdollinen ratkaisu tällaisten tilanteiden välttämiseksi olisi nk. Backend For Frontend (lyh. BFF). Se tarkoittaisi kokonaisarkkitehtuurin osalta sitä, että käyttöliittymää varten olisi kehitetty *yksi* rajapinta, johon käyttöliittymästä käsin oltaisiin yhteydessä. Tämä rajapinta olisi sitten yhteydessä tarvittaviin muihin rajapintoihin tai mikropalveluihin, joka tekee kokonaisarkkitehtuurista suoraviivaisemman.

Nykyiset ongelmat ovat mielestäni seurausta siitä, että ensinnäkin sovelluksista käytettävät rajapinnat eivät kaikki ole toisistaan riippumattomia, jolloin muutokset yhteen paikkaan aiheuttaa sivuvaikutuksia toisaalla. Eri henkilöiden eriävät mielipiteet (kuten projektipäällikön ja järjestelmäarkkitehdin) ovat nähdäkseni myös muodostaneet eräänlaisen kuilun viestintään, jolloin teknisesti optimaalisin ratkaisu on ehkä jäänyt löytämättä. Ongelmien juurisyyt toisaalta sijaitsee myös vanhojen ja uusien projektien yhteensovittamisessa. Eri asiakkailla voi olla myös eri versioita sovelluksista käytössä, jotka sisältävät järjestelmän eri osia ja erilaisia ominaisuuksia.

Useat liikkuvat osat tekevät kokonaisuudesta vaikeamman hahmottaa ja ylläpitää. Tämä tekee tilanteesta hieman sellaisen, että ongelmat tuppaavat kasautumaan Frontend-kehittäjien selvitettäviksi. Eri rajapinnoilla on erilaiset tietotarpeet ja ne käsittelevät erimuotoista dataa. Osa rajapinnoista myös sisältää keskinäisiä riippuvuuksia tiedon suhteen ja viestii myös keskenään (Management API & User management api).

Vuosien varrella työnantajani järjestelmiin on kasautunut erilaisia muuttuvia osia, joista osa on enemmän ja toiset vähemmän ajan tasalla. Muutoksia on tapahtunut myös käsitteellisellä tasolla jonkin verran. Vaihteleviin tilanteisiin on kehitetty abstraktioita jotain tiettyä toiminnallisuutta varten. Myöhemmin on ilmennyt uusia tarpeita, jolloin aiempiin abstraktioihin perustuen on kehitetty uusia abstraktioita ratkomaan muuttuneiden tarpeiden ongelmia. Ajan myötä tällaiset asiat ovat päässeet nähdäkseni hieman kumuloitumaan, jolloin näistä osista koherentin kokonaisuuden muodostaminen on vähintäänkin haastavaa ellei miltei mahdotonta.

Johto ja projektien vetovastuussa olevat henkilöt tuntuvat usein puoltavan kaikkein nopeimpia ja vähätöisimpiä ratkaisuja. Tämä luonnollisesti siksi, etteivät projektit tulisi liian kalliiksi. Ongelmia on näkemykseni mukaan päässyt kasautumaan todennäköisesti siksi, että on yritetty päästä useasti sieltä, missä aita näyttää matalimmalta. Projekteja on myös lähdetty viemään eteenpäin nopeiden tulosten toivossa, mutta se on tapahtunut osittain suunnittelun ja vaatimusten määrittelyn kustannuksella. Kuluja on säästetty myös

testausresurssien osalta. Pitkällä aikajänteellä helpot ja halvat ratkaisut tulevat kuitenkin kalliiksi. Puutteellinen vaatimusmäärittely, testaus tai epärealistiset aikataulut kostautuvat helposti pitkällä aikavälillä.

Tähänastinen kokemukseni on osoittanut, että väliaikaisratkaisuja ei ole olemassa. Väliaikaisratkaisut tuppaavat tulemaan usein jäädäkseen. Mieleeni on jäänyt tähän liittyen Robert Martinin kirjassaan Clean Code (2008, Luku 1) mainitsema LeBlancin laki: Myöhemmin on yhtä kuin ei koskaan (engl. *Later equals never*). Olen todistanut tämän omakohtaisesti monien asioiden kohdalla. Olen koittanut ottaa opikseni ja tehdä asioita hieman eri tavalla. Tämä on tuottanut positiivisia kokemuksia. Huomatessani ajattelevani tekeväni jotain myöhemmin, olen tunnistanut ajatukset ja päättänyt tehdä asian saman tien. Monessa kohtaa tämä on ollut myöhemmin hyödyksi. Erityisesti pikkuasiat tuppaavat kasautumaan kaikkein pahiten. Yhtenäisen koodityylin noudattamista helpottamaan työkalut kuten Eslint ja Husky ovat olleet Centry-projektissa tähän asti hyödyksi.

3.10 Seurantaviikko 10: 08.-12.11.2021

Maanantai 08.11.2021
Viikko alkoi tavalliseen tapaan tuotekehityspalaverilla, jossa käytiin läpi tämänhetkisten projektien tilanne. Ilmoitin Centry-projektin tämänhetkisen tilanteen ja sen, että eteneminen riippuu tällä hetkellä paljolti siitä, kuinka projektin API-kehitys saadaan etenemään. Projektiin on lähiaikoina tulossa työn alle myös muita toimintoja. Palaverin jälkeen palasin jatkamaan ohjelmointia siitä mihin viimeksi jäin, eli nk. Skill-profiilien muokkauksesta.

Tein toiminnallisuudet niin pitkälle, mitä rajapintojen tämänhetkinen tilanne sallii ja siirryin lounaan jälkeen tekemään uutta näkymää. Asiasta täytyi pitää lyhyt palaveri projektipäällikön ja toisen kollegan kanssa. Sain työkalut, jotta pääsin tehtävän kanssa eteenpäin, kuten pääsyn uuteen rajapintaan ja kuvan ulkoasusuunnitelmasta.

Uudessa näkymässä on tarkoitus käsitellä asiakaspalvelijoiden valitsemia nk. työkoodeja. Päättäessään jonkin työtehtävän asiakaspalvelijoiden täytyy valita tehtävään liittyvä koodi, joka määrittää työn lopputuleman ja jatkotoimenpiteet, kuten lähetetäänkö tyytyväisyyskysely vai ei. Käytin loppupäivän näkymän alustukseen ja rajapintakutsujen luomiseen.

Tiistai 09.11.2021
Tänään jatkoin työkoodinäkymän suunnittelua ja näkymän työstämistä. Datan rakenne osoittautui taulukkojen kannalta hieman haasteelliseksi, koska se sisältää paljon sisäkkäisiä listarakenteita. Jotta datan saisi näytettyä taulukossa sen rakennetta tulee muuttaa datataulukkoon sopivaksi, joka poikkeaa alkuperäisestä rakenteesta lähes täysin. Konsultoin

asiassa kollegaani, kuka on tehnyt käyttöliittymäsuunnitelmat. Ehdotin, että datan rakenteen voi säilyttää ennallaan, jos tiedon näyttäisi puumaiseen tietorakenteeseen sopivammalla tavalla taulukon sijaan. Kollegani kuitenkin ehdotti, että pyrin muokkaamaan dataa niin, että se sopii taulukkoon.

Lähdin yrittämään ratkaisua. Ajatuksenani oli toteuttaa funktio, joka rekursiivisesti purkaa sisäkkäiset taulukot sellaiseksi rakenteeksi, jonka voi näyttää taulukossa riveinä. En päivän aikana löytänyt toimivaa ratkaisua, enkä ole varma onko sellaisen toteuttaminen edes tarkoituksenmukaista tai järkevää. Päätin ottaa asian seuraavana päivänä esille.

Keskiviikko 10.11.2021
Mainitsin tämän päivän aamuisessa tuotekehityspalaverissa ongelmasta, johon eilen törmäsin ja pidimme kyseisen palaverin jälkeen aamupäivällä käyttöliittymäsuunnittelijan ja kokeneemman backend-kehittäjän kanssa toisen lyhyehkön palaverin, jossa vilkaisimme ongelmaa tarkemmin.

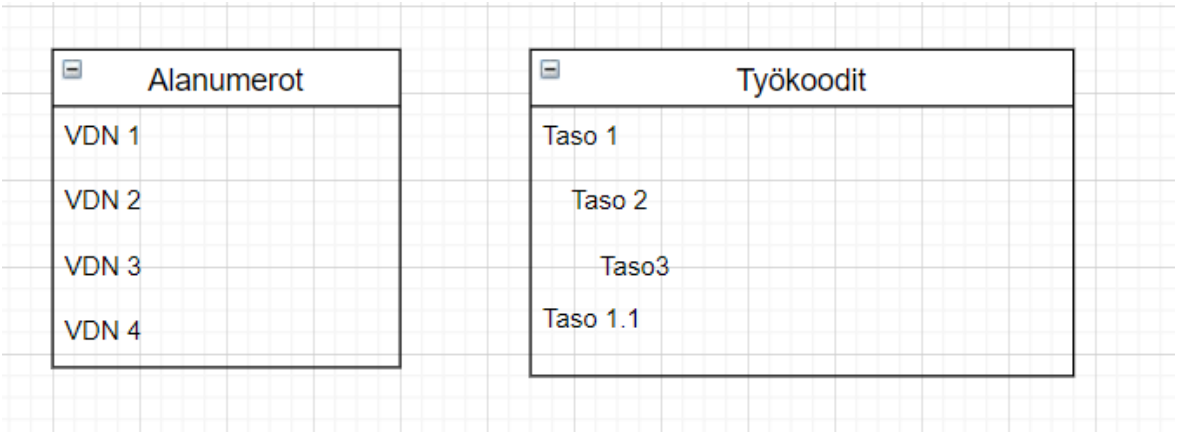
Käyttöliittymäsuunnittelija löysi käyttämästämme Telerik Kendo React-käyttöliittymäkirjastosta *puurakenteita* varten suunnitellun muokattavissa olevan taulukkokomponentin, joka sopi tähän tarkoitukseen hyvin. Olin tyytyväinen, että ongelmaan löytyi juuri sopivan oloinen ratkaisu läheltä. Lähdin jatkamaan työkoodinäkymän työstämistä uudella komponentilla.

Iltapäivällä minulle tuli pieni muutospyyntö chat-sovellukseen, jonka konfiguraatioita olin muuttanut viime viikolla. Kävi ilmi, että edellisviikon konfiguraatiomuutosten yhteydessä koodiin oli epähuomiossa jäänyt vanhaa konfiguraatiotapaa käyttävää koodia, joka aiheutti ongelmia. Tämä oli suhteellisen nopeasti korjattu.

Centry-projektin käyttäjien hallintaan tuli myös yksi muutospyyntö liittyen käyttöliittymän toimintaan uutta käyttäjää lisätessä. Tässä kyseessä ei ollut niinkään bugi, vaan mukautus taulukon solujen muokattavuustilaan, kun kyseessä on uusi taulukon rivi.

Torstai 11.11.2021
Tämän päivän käytin pitkälti työkoodinäkymän työstämiseen. Työkoodeihin liittyviä tietoja ovat muun muassa alanumero (VDN), jolle koodi on liitetty, sekä asetus voiko koodin valitessa valita useampia sisäkkäisiä tasoja. Aiemmin viikolla mainitsin tiedon puurakenteesta ja sen sopimattomuudesta taulukoihin. Tiedon puurakenne johtuu siitä, että eri työkoodeihin voi liittyä sisäkkäisiä työkoodeja, jotta asiakaspalvelijat voivat määritellä tarkemmin,

mitä työn aikana tapahtui. Lopputuloksia voi olla tietty määrä ja tiettyihin lopputuloksiin voi liittyä tarkentavia valintoja.



Kuva 4. Hahmotelma Centry-projektin työkoodien hallintanäkymästä

Itse työkoodinäkymä koostuu kahdesta osasta, joista vasemmalla puolella listataan eri alanumerot (VDN) ja oikealla puolella työkoodien eri tasot puurakenteiseen taulukkoon listattuna (Kuva 4). Molempien taulukoiden alkioita voi lisätä, muokata, poistaa ja muutokset tallentaa tai peruuttaa.

Puurakenteen tiedon käsittelytoiminnot vaati rekursiivisia apufunktioita, joiden avulla sain koko puurakenteen kaikki alkiot käytyä läpi halutessani muokata tiettyä arvoa tai poistaa niistä jonkin. Tämä ei ollut täysin yksinkertaista, mutta oppimani funktionaalisen ohjelmoinnin tekniikat mahdollistivat datan oikean käsittelyn. Meni hetki löytää toimiva ratkaisu, mutta lopulta tiedon lisäys, muokkaus ja poisto-operaatiot toimi haluamallani tavalla.Päivän päätteeksi olin tyytyväinen saadessani tähän näkymään liittyvät ydintoiminnot testausta varten valmiiksi.

Perjantai 12.11.2021
Päiväkirjamerkintä puuttuu sairasloman takia.

Viikkoanalyysi
Tällä viikolla olen ratkonut ongelmia liittyen työkoodinäkymän datan puurakenteeseen. Datapuun yksittäisiä alkioita täytyi pystyä lisäämään, muokkaamaan ja poistamaan. Myös puun tasoja täytyi voida olla mielivaltainen määrä. Yksittäisen puurakenteessa sijaitsevan objektin käsittely käyttöliittymän puurakenteessa vaati toimintoja koko puurakenteen läpikäymiseen ja tässä hyödynsin funktionaalisesta ohjelmoinnista tuttuja tekniikoita.

```
{
  "attribute": "Level 0",
  "children": [
    {
      "attribute": "Level 1a",
      "children": []
    },
    {
      "attribute": "Level 1b",
      "children": [
        {
          "attribute": "Level 2a",
          "children": []
        },
        {
          "attribute": "Level 2b",
          "children": [
            {
              "attribute": "Level 3a",
              "children": [
                {
                  "attribute": "Level 4a",
                  "children": []
                }
              ]
            },
            {
              "attribute": "Level 3b",
              "children": []
            }
          ]
        }
      ]
    }
  ]
}
```

Kuva 5. Esimerkki JSON-objekteista koostuvasta puurakenteesta.

JSON-puurakenne, jota olen viikon aikana käsitellyt sisältää JSON-olioita, jonka yksi attribuutti sisältää objektin lapsiattribuutit. Jokainen lapsiattribuuteista taas voi sisältää myös omat lapsensa ja niin edespäin muodostaen puumaisen haarautuvan tietorakenteen (Kuva 5).

Yksi käyttämistäni ratkaisuista edellä mainitun puurakenteen käsittelyyn liittyviin ongelmiin on rekursio. Rekursio on yksi funktionaalisen ohjelmoinnin keskeisistä tekniikoista (Kereki 2020, luku 9.). Käytännössä rekursio on funktio, joka kutsuu itseään useita kertoja tietyyn pisteeseen asti, jolloin jonkin ehdon täyttyessä kutsut lopetetaan (Kereki 2020, luku 9). Puurakenteen muokkaus- ja poisto-operaatioissa kävin puurakenteen läpi funktiolla, joka saa parametrina yhden puurakenteen JSON-olioista. Funktion sisällä tarkistin löytyikö objektilta etsimäni arvo tai onko sillä lapsiobjekteja. Mikäli lapsiobjekteja on, täytyi funktioni käydä läpi lapsiobjektit silmukassa ja kutsua itseään jokaisen objektin kohdalla antamalla kyseinen objekti itselleen parametrina.

Toinen käyttämäni tekniikka aiempien lisäksi oli nk. *Curry*aminen (*engl. currying*). Sen ajatuksena on muuttaa useita parametreja ottava funktio ketjuksi funktioita palauttavia funktioita, joista jokainen ottaa sisäänsä parametrina yhden alkuperäisen funktion parametreista. Ketjun viimeinen funktio palauttaa lopullisen halutun arvon. Tällä tekniikalla funktioiden *muuttujamäärää* voidaan pienentää. (Kereki 2020, luku 7.) Siitä oli hyötyä rekursiivisten kutsujen yhteydessä, jolloin tarvitsin toisinaan muuttujia useista nimiavaruuksista, mutta silti funktioita, jotka ottivat alkujaan vain yhden parametrin. Puurakenteen käsittely oli hieman haastavaa, mutta myös opettavaista ja kiinnostavaa. Olen kiinnostunut funktionaalisen ohjelmoinnin tekniikoista, sillä ne tarjoavat mielestäni elegantteja ratkaisuja monimutkaisiin ja haastaviin ongelmiin. Myös React ja Redux sisältävät funktionaalisesta ohjelmoinnista tuttuja malleja (Kereki 2020, luku 1).

Ylläolevan puurakenteen JSON-oliot muistuttavat hyvin paljon myös rakennetta, johon Reactin nopean sisäisen toiminnan mahdollistava tietomalli, Virtual DOM liittyy (Raja, 2020). VirtualDOM on muistissa säilytettävä mallinne varsinaisesta selaimessa ajettavasta käyttöliittymän tilasta (Facebook 2021). Käyttöliittymäelementtien tila on muistissa tallennettu objekteihin, joita kutsutaan React-elementeiksi (Facebook 2021).

```
{
    type: Title,
    props: {
        color: 'red',
        children: [
            {
                type: 'h1',
                props: {
                    children: 'Hello, H1!'
                }
            },
            {
                type: 'h2',
                props: {
                    children: 'Hello, H2!'
                }
            }
        ]
    }
}
```

Kuva 6. Kuva React elementistä (Mukaillen Roldán 2021, kpl 1)

Kuten kuvassa 5, myös kuvan 6 React-elementit sisältävät lapsielementtejä, joista Reactin komponenttipuu rakentuu. React-elementit kuvaavat sitä, mitä selaimen kuuluu näyttää (Roldán 2021, kpl 1). Virtual DOM mahdollistaa Reactin deklaratiivisen luonteen eli sen, että sovelluskoodin tilassa määritetään se tila, mitä halutaan käyttöliittymässä nähdä ja React varmistaa että selaimen näkymä vastaa sitä (Facebook 2021).

4 Pohdinta ja päätelmät

Opinnäytetyön aikana olen kasvanut kehittäjänä ja kartuttanut lisää työkokemusta käytännön projekteissa. Erilaisten teknologioiden lisäksi lisäksi olen oppinut lisää suunnittelutyöstä, kehittynyt mielestäni viestijänä ammattimaisessa viitekehyksessä ja tullut sekä työssäni että henkilökohtaisessa kehityksessäni itseohjautuvammaksi. Oppimani ja lukemani asiat ovat saaneet minut myös haastamaan kehittämään itseäni. Mielestäni on hyvä välillä haastaa omia toimintamallejaan ja näkemyksiään ja asennoitua uusiin asioihin oppimiskokemuksina

Kirjoittamisprosessi on ollut sekä matka ammatilliseen itseen, että myös pohdinnan paikka siirtymävaiheessa opinnoista työelämään. Kirjoittamisen aikana tapahtunut oman kehittymisen reflektointi on myös selkeyttänyt omaa käsitystäni siitä, missä vaiheessa uraa nyt olen ja mihin suuntaan haluaisin edetä. Omien toimintamallien pohdinta ja arviointi on ollut arvokasta itsereflektointia, joka on kypsyttänyt opintojen aikana kerättyä tietoa ja työn ohella tapahtunutta oppimista. Useimmista aluksi hyvin vaikealta ja pelottavaltakin tuntuneista asioista olen selvinnyt parhaiten lähtemällä pilkkomaan ongelmia sen verran pieniin osiin, että ne ovat realistista toteuttaa. Pieniä ongelmia on helpompi ratkoa yksi kerrallaan kuin valtavalta tuntuvaa ongelmien vyyhtiä kaikkinensa. Ajan kuluessa ja asiasta oppiessa lisää ongelmanratkaisuprosessi nopeutuu ja aluksi vaikealta tuntunut asia voi lopulta osoittautua hyvinkin yksinkertaiseksi ja suoraviivaiseksi, vaikkei sitä olisikaan. Myös tarpeen vaatiessa kollegoilta avun pyytäminen sekä mahdollisten esteiden ilmaiseminen ovat olleet tärkeitä taitoja.

Olen myös huomannut, että vaikeimmilta tuntuneiden asioiden vaikeus on pohjautunut enemmänkin omiin uskomuksiin ja tunnepohjaisiin käsityksiin kuin rationaaliseen ajatteluun. On mielestäni luonnollista säikkyä uusia asioita, koska selviytymisvaistomme ovat pitäneet lajimme tähän asti hengissä. Ilman sitä olisimme varmasti kuolleet sukupuuttoon jo kauan sitten. On kuitenkin muistettava, että jatkuva kehittyminen muuttuvassa maailmassa on myös rikkaus, joka pitää mielen virkeänä.

Olen tutustunut ohjelmoinnin hyviin käytäntöihin, React-kirjaston sisäiseen toimintaan, erilaisiin ohjelmointiparadigmoihin ja konfiguraatiotapoihin, käyttöliittymäsuunnitteluun, televiestintäjärjestelmiin, testaukseen, vaatimusmäärittelyyn ja tuotantoonsiirtoon. Olen oppinut lisää itselleni uusista Protocol Buffer-kyselyistä ja saanut lisää käytännön työkokemusta asiantuntijaroolissa toimimisesta. Olen oppinut tekemään ratkaisuistani siistimpiä, helpommin ylläpidettäviä ja skaalattavia sekä komponenteista uudelleenkäytettävämpiä. Olen löytänyt uusia mielenkiintoisia modernimpia arkkitehtuurimalleja Redux-tilanhallinnan hyödyntämiseen React-sovelluksissa ja aion hyödyntää oppimiani työkaluja myös

jatkossa. Olen oppinut lisää Typescriptin ominaisuuksista hyödyntämällä laajemmin Typescriptin tarjoamia erilaisia tyyppejä. Olen huomannut staattisen tyypityksen ominaisuuksien tuomia hyötyjä laajan skaalan Javascript-projekteissa, joissa työskentelen. ESLint-työkalun tarjoaman koodityylin tarkistuksen seurauksena omasta koodistani on tullut yhdenmukaisempaa ja hyväksi todetut ESLint-konfiguraatiot auttavat tekemään minusta paremman ohjelmoijan. Tyylisääntöjen paremmasta noudattamisesta on tullut itseään ruokkiva positiivinen kierre. Tyylisääntöjen ja paremman Typescript-tyypityksen ansiosta koodiin päätyy vähemmän virheitä tai yllätyksiä. Koodi pysyy myös luettavampana.

Tämänhetkisessä yrityksessäni ei tunnu olevan vakiintuneita prosesseja koodin katselmoimiseen, vaan laadunvarmistus tapahtuu enemmänkin tuotosten manuaalisen testaamisen ja katselmoinnin kautta. Ohjelmistoalalla käytetään laajalti myös pull-requesteja ja ohjelmoijat katselmoivat aktiivisesti toistensa kirjoittamaa koodia. Hyviksi todettuihin testausprosesseihin kuuluu mielestäni myös erilaisia yksikkö- integraatio- ja end-to-end-testejä. En ole päässyt vielä näkemään työpaikallani ihan niin paljon testejä tai koodikatselmointeja kuin olisin toivonut. Siispä olenkin yrittänyt hyödyntää Typescriptin ominaisuuksia mahdollisimman kattavasti, koska ne pienentävät virhemarginaalia ja parhaimmillaan vähentävät yksikkötestien tarvetta. Olen myös päättänyt alkaa kirjoittaa testejä enemmän ja olen ottanut tavoitteeksi lähteä ajamaan parhaani mukaan asiaa eteenpäin työpaikallani. Nykyisessä tilanteessa automaattitestaukseen ei vaan tunnu aina riittävän tarpeeksi resursseja. Käyttöliittymätestien huonoja puolia ovat huono sietokyky muutoksille, jolloin testit menevät helposti rikki. Nykyisessä projektissa olen käyttänyt joitakin yksikkötestejä varmistamaan, että sovelluksen kriittisimmät osat käynnistyvät odotetulla tavalla pyrkien varmistamaan, että sovellus lähtee käyntiin. Lisäksi olen tehnyt joitakin end-to-end-testejä ja aion jatkaa niihin tutustumista.

Työn aikana olen myös saanut tilaisuuden tutustua sovellusten tuotantoonsiirtoon. Olen tutustunut mm. Jenkins-nimiseen devops-työkaluun, jolla sovellusten siirtoa kehityksestä tuotantoon on mahdollista automatisoida. Työkalu hakee pilvipalveluna hostatusta versionhallinnasta sovellusten lähdekoodin, rakentaa koodin pohjalta toimitettavan paketin ja asentaa sen kohdeympäristöön. Olen rakentanut React-sovelluksistani konfiguraatioiltaan yhteensopivia sekä konttipohjaisiin Docker-toimituksiin, että Windows Server-palvelinympäristöihin sopiviin toimituksiin. Sovellukset on myös mahdollista konfiguroida JSON-tiedoston avulla niin, että saman paketin voi asentaa eri ympäristöihin joutumatta kasaamaan sovellusta erikseen eri ympäristöjä varten. Myös sovellusten hostaaminen käänteisen välityspalvelimen takaa onnistuu. Kaikki edellä mainitut asiat ovat tuoneet lisää arvokasta kokemusta.

Hankittu osaaminen ja puhdas tieto on arvokasta, sillä sitä tarvitsee IT-alalla paljon. Välillä silti tuntuu, että oppimaan oppiminen on vielä melkeinpä tärkeämpää. Taustalla jo hankittu osaaminen toki helpottaa uuden oppimista entisestään, koska osaamista on helpompi siirtää alueelta toiselle. Uuden asian nopea haltuunotto on kuitenkin asia, jossa myös kehittyy ja jota voi opetella. Teknologia-alalla tuskin voi jäädä lepäilemään laakereille kovinkaan pitkäksi aikaa. Välillä tuntuu siltä, että kun opettelee hyvin yhden asian, se on todennäköisesti pian jo vanhentunutta tietoa. Tämä tekee uusien asioiden oppimistaidosta entistä tärkeämpää. Alan työmarkkinat kehittyvät myös nopeasti ja erilaiselle osaamiselle on eri aikoina eri tavalla kysyntää. Käytännön työelämässä ja projektityössä ilmenevät tarpeet voivat olla hyvinkin nopealla aikataululla ilmeneviä ja vaatia riittävää mukautumiskykyä, jotta projektien aikana ilmenneet haasteet saadaan selätettyä. Siispä on olennainen selviytymistaito osata lukea dokumentaatioita ja etsiä tietoa käsillä olevasta aiheesta eri lähteistä ja hyödyntää mahdollisten olemassa olevien kehittäjäyhteisöjen tarjoamia ratkaisuja. Usein on myös tilanteita, joihin valmista ratkaisua ei löydy, jolloin ongelmanratkaisutaidot ja vahva ohjelmointirutiini ovat välttämättömiä taitoja.

Lista asioita, joita haluaisin oppia on pitkä. Funktionaalisen ohjelmoinnin perusteet ovat myös nykyään tärkeitä hallita. Olen React-kehityksen myötä jonkin verran jo funktionaaliseen ohjelmointiin päässyt tutustumaan, mutta tavoitteenani on lähiaikoina syventyä aiheeseen vielä lisää. Javascriptia käytetään funktionaaliseen ohjelmointiin paljon, vaikka Javascript ei olekaan pelkästään funktionaaliseen ohjelmointiin tarkoitettu kieli, kuten vaikkapa Haskell. Funktionaalinen ohjelmointi, rekursiot ja erilaisten algoritmien tuntemus avaavat ovia tehokkaaseen ongelmanratkaisuun ja vapauttaa mielen kapasiteettia varsinaiseen ajatteluun.

Pitkällä tähtäimellä olen kiinnostunut etenemään frontend-kehitykseen painottuvista tehtävistä lähivuosien aikana fullstack-kehittäjän rooliin. Jatkan uuden opettelua työssäni sekä laajennan osaamistani mahdollisuuksien mukaan rajapintakehityksen ja tietokantojen osalta. Olen jo kartuttanut osaamistani hyvän matkaa enkä usko, että on enää varsinaisesti asioita, joita ei olisi mahdollista oppia työn kautta. Yritän kehittää osaamistani parhaani mukaan myös vapaa-ajalla harrasteprojektien kautta. Opinnäytetyön kirjoittamisprosessi ja pohdinta työn lomassa on selkeyttänyt ajatukseni sen suhteen, että haluan työstäni tulevaisuudessa mahdollisimman monipuolisen. Kenenkään on tuskin mahdollista oppia kaikkea, mutta pyrin todennäköisesti työnkuvaan, jossa sekä käyttöliittymän että palvelinohjelmistojen parissa tapahtuvia asioita on sopivassa suhteessa.

Lähteet

Wikipedia. 2021. Puhelinverkko. Luettavissa: https://fi.wikipedia.org/wiki/Puhelinverkko. Luettu: 10.09.2021

Wikipedia. 2021. Interactive voice response. Luettavissa: https://en.wikipedia.org/wiki/Interactive_voice_response. Luettu: 10.09.2021

Wikipedia. 2021. Trunking. Luettavissa: https://en.wikipedia.org/wiki/Trunking. Luettu: 10.09.2021

Wikipedia. 2021. Vector directory number. Luettavissa: https://en.wikipedia.org/wiki/Vector_directory_number. Luettu: 10.09.2021

Roldán, C.S. 2021. React 17 Design Patterns and Best Practices – Third Edition. Packt Publishing Ltd. Birmingham. E-kirja. Luettu: 22.09.2021.

Galitz, W.O. 2007. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, 3rd Edition. John Wiley & Sons Inc. New Jersey. E-kirja. Luettu: 17.09.2021.

Abramov, D. 23.03.2015. Presentational and Container Components. Luettavissa: https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0. Luettu: 22.09.2021.

Abuelenain, Doyle, Karneliuk & Jain. 2021. Network Programmability and Automation Fundamentals. Cisco Press. Indianapolis. E-kirja. Luettu: 01.10.2021

Martin, R.C. 2011. The Clean Coder: A Code of Conduct for Professional Programmers. E-kirja. Luettu: 07.10.2021

Martin, R.C. 2008. The Clean Code: A Handbook of Agile Software Craftsmanship. E-kirja. Luettu: 07.11.2021

Sangoma. 2021. How does sip trunking work. Luettavissa: https://www.sangoma.com/articles/how-does-sip-trunking-work/. Luettu: 10.11.2021

Agiles. 2018. Why ERP Projects fail. Luettavissa: https://agiles.com/en/why-erp-projects-fail/. Luettu: 10.11.2021

Kereki, F. 2020. Mastering JavaScript Functional Programming – Second edition. Packt Publishing Ltd. Birmingham. E-kirja. Luettu: 16.11.2021.

Facebook. 2021. Virtual DOM and Internals. Luettavissa: https://reactjs.org/docs/faq-internals.html. Luettu: 16.11.2021

Raja, K. 2020. What is Virtual Dom? And Why is it faster? Luettavissa: https://dev.to/karthikraja34/what-is-virtual-dom-and-why-is-it-faster-14p9. Luettu: 16.11.2021

Liitteet

Liite 1. Käsitteet

React: Facebookin luoma avoimen lähdekoodin JavaScript-kirjasto, jolla luodaan selainpohjaisia käyttöliittymiä.

TypeScript: Microsoftin kehittämä kieli, joka tarjoaa staattisen tyypityksen ominaisuuksia perinteisen dynaamisesti tyypitetyn JavaScriptin tueksi.

HTTP: Lyhenne sanoista hypertext transfer protocol. Tarkoittaa protokollaa, jota käytetään selainten ja palvelinten väliseen tiedonsiirtoon.

Docker: Käyttöjärjestelmätason virtualisaatiota hyödyntävä teknologia joka mahdollistaa erilaisten ohjelmistojen ajamisen konteissa (*engl. Containers*).

Imperatiivinen ohjelmointi: Yleinen ohjelmointiparadigma, jossa keskeistä on se, että komennoilla muutetaan suorituksen tilaa yksi kerrallaan muokkaamalla muuttujiin talletettuja arvoja. Imperatiivisen ohjelmoinnin voi ajatella kuvaavan sitä, miten asiat toimivat. Se käsitetään usein deklaratiivisen ohjelmoinnin vastakohtana.

Deklaratiivinen ohjelmointi: Käsitetään imperatiivisen ohjelmoinnin vastakohtana. Deklaratiivisessa ohjelmoinnissa usein imperatiivisen ohjelmoinnin piirteet ovat implisiittisiä, jolloin ongelmia ratkaistaan kuvaamalla sitä mitä halutaan saavuttaa varsinaisten algoritmien sijaan. Funktionaalinen ohjelmointi on yksi deklaratiivisen ohjelmoinnin alalaji.

API: Ohjelmointirajapinta. Lyhenne englannin kielen sanoista Application Programming Interface. Rajapinnat määrittävät, miten eri ohjelmat voivat vaihtaa tietoja keskenään.

Protocol Buffers: Googlen kehittämä avoimen lähdekoodin dataformaatti, jolla serialisoidaan strukturoitu dataa. Sitä käytetään verkon yli viestimiseen binäärimuotoon enkoodatuin kyselyin.

Frontend: Sovelluksen tai tietojärjestelmän käyttöliittymä tai osa, joka on lähimpänä käyttäjää, kuten vaikkapa selainpohjainen sovellus.

Backend: Sovelluksen tai tietojärjestelmän osa, joka on kauimpana käyttäjästä, eli esimerkiksi tietokantaan lukeva ja sinne kirjoittava rajapinta eli API.

Full-stack: Full-stackista puhuttaessa tarkoitetaan frontendin ja backendin muodostamaa kokonaisuutta. Sana stack viittaa järjestelmän kerroksittaiseen arkkitehtuurirakenteeseen.

VDN: Vector Directory Number. Ohjaa sähköisessä puhelinjärjestelmässä puhelut oikeaan Vektoriin

Vector: Sähköisen puhelinjärjestelmän Vektori, eli puhelulle ohjelmoitu kulku.

IVR: Interactive Voice Response. Puhelun aikana tehtävät näppäimistövalinnat.

Trunk: Kommunikaatiokanava yrityksen puhelinvaihteen ja operaattorin välillä

Agent: Sähköisen contact center-järjestelmän käyttäjä, tyypillisesti asiakaspalvelija

Skill: Sähköisen contact-center järjestelmän asiakaspalvelijalle asetettu palvelujono, josta työt asiakaspalvelijalle ohjautuu, kuten vaikkapa eri puhelinkanavat, chat tai sähköpostiasiakaspalvelu.

UI: Lyhenne englannin kielen sanoista User Interface. Tarkoittaa käyttöliittymää.

Github: Avoin pilvipalvelu, joka tarjoaa tallennustilaa Git-versionhallintaa käyttäville ohjelmistoprojekteille.

Stackoverflow: Keskustelufoorumi, joka käsittelee ohjelmointia, arkkitehtuureja ja muita teknisempiä IT-aiheita.

PSTN: Lyhenne englannin kielen sanoista Public Switched Telephone Network. Tarkoittaa julkista puhelinverkkoa.

JSON: Tiedostomuoto tiedonvälitykseen (.json). Lyhenne sanoista JavaScript Object Notation.

Redux: Avoimen lähdekoodin JavaScript-kirjasto, joka on kehitetty React- ja Angular-sovelluksien keskitettyyn tilanhallintaan. Alun perin Dan Abramovin ja Andrew Clarkin kehittämä.

Jenkins: Palvelinohjelmisto, jota käytetään DevOps-toimintamallien mukaisten jatkuvan integraation ja jatkuvan toimituksen palvelimissa

Devops: Toimintamalli, joka pyrkii automatisoimaan sovellusten saattamisen sovelluskehittäjien käsistä tuotantoympäristöihin. Automatisointiin liittyy testaamiseen ja ylläpitoon liittyviä toimintoja.

Jira: Australialaisen yrityksen Atlassianin tehtävienhallintaohjelmisto, jota käytetään mm. erilaisten IT-projektien hallintaan.