Haaga-Helia
ammattikorkeakoulu Oy

Tietojärjestelmän rakentaminen startup-yrityksessä full-stack kehittäjänä

Aleksi Tuhkanen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2017

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

Päiväys 25.05.2017

Tekijä(t)
Aleksi Tuhkanen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Tietojärjestelmän rakentaminen startup-yrityksessä full-stack kehittäjänä

Sivu- ja liite-sivumäärä
52 + 0

Opinnäytetyön otsikko englanniksi

Developing an information system in a startup company as a full-stack developer

Tämä portfoliomainen päiväkirjaopinnäytetyö, jossa seurataan full-stack kehittäjän työtehtäviä päiväkohtaisilla merkintojen lisäksi viikkottaisilla analyyseillä. Analyyseissa pohditaan viikon aikana kohdattuja haasteita pureutuen niiden teemoihin ja peilaten niitä teoriaan. Opinnäytetyössä käsitellään startup-yrityksen työntekijän arkea yrityksen ainoana ohjelmistokehittäjän.

Opinnäytetyön seurantaviikkoja edeltää tekijän lähtötilannetta havainnollista osuus, jota kuvataan vaaditun osaamisen, työympäristön, sidosryhmien ja vuorovaikutustaitojen kautta.

Seurantajaksojen oleellisimpia havaintoja käsitellään viikkoanalyyseissä teemojen kautta. Viikkoanalyysien teemat liittyvät vahvasti työkuvan osa-alueisiin kuten ohjelmistokehitykseen ja työskentelytapoihin. Analyyseissä käytettiin apuna teemoihin liittyviä lähteitä havaintojen vertaamiseen ja tukemiseen.

Työn lopussa pohditaan syvällisemmin kehittymistä seurantajakson ajalta ja mietitään tekijän tulevaisuuden tavotteita niiden saralla. Oleellisimmat havainnot kehittymisessä keskittyivät teknisen osaamisen kasvamisen lisäksi projektihallinnallisten taitojen kartuttamiseen.

Asiasanat
tietojärjestelmä, startup, ohjelmistokehitys

Sisällys

1 Johdanto

Opinnäytetyö tuotetaan vuoden 2017 keväällä, 6.2.2017 – 25.5.2017 välisena aikana. Päiväkirjamuotoinen opinnäytetyö sisältää kymmenen viikon mittaisen seurantaosuuden, jonka aikana jokaisena arkipäivänä kirjoitetaan merkintä työtehtävistä ja niiden toteutumisesta. Jokainen viikko päätetään arkipäivien merkinnöistä koostuvaan analyysiin, jossa peilataan kokemuksia ja ongelmia kirjallisuuden teoriaan.

Työssä käsiteltävä projekti on startup-yrityksen SaaS-palveluksi kehitettävä kattava tietojärjestelmä. Olen yksin vastuussa koko projektin teknisestä kehittämisestä. Toimenkuvani tekninen osa-alue sisältää koko full-stack ohjelmoijan työtehtävät sekä kattaa osia palvelinasiantuntija tehtävistä. Teknisen osaamisen lisäksi toimenkuvaani kuuluu projektinhallinta ja järjestelmän tarpeiden puitteissa kattava perehtyminen sen käyttöön ottavaan kohderyhmään.

Työtehtäviini kuuluu päivittäin järjestelmän ominaisuuksien ja toiminnallisuuksien kehitys aina tietokannasta front-endiin. Järjestelmä on rakennettu JHipster –viitekehyksen päälle joka edellyttää hyvää Java, HTML5 ja JavaScript osaamista. JHipster sopii mainiosti yhden miehen kehitystyöhön, sillä se tuottaa valmiiksi paljon Javasta tuttua ”boilerplate” –koodia vähentäen merkittävästi työtaakan määrää. Ohjelmoimisen lisäksi työ vaatii järjestelmän palvelinasennuksen lisäksi MySQL-tietokannan varmuuskopiointiin, migraatioon ja palauttamiseen tarvittavia taitoja.

Työskentelen nuoressa startup-yrityksessä, joka työllistää neljän henkilön lisäksi useita konsultteja jotka työskentelevät yrityksen muiden projektien parissa. Kyseinen yritys keskittyy erilaisiin koiraihmisille suunnattuihin it-palveluihin, joiden ratkaisujen teemat korostavat helppokäyttöisyyttä, yhteisöllisyyttä ja harrastustoimintaa.

1.1 Tietoperusta

WarBurton, R. 2014. Java 8 Lambdas: Functional Programming for the Masses. O’Reilly Media. Kalifornia.

- Teos kertoo Javan 8. version muutoksista ja on suunnattu perustaitoja hallitsevalle lukijalle.

Seshardi, S & Green B. 2014. AngularJS: Up and Running.
O’Reilly Media. Kalifornia.

- Teos antaa kattavat eväät aloittavalle AngularJS-kehittäjälle.

1.2 Keskeiset käsitteet

SaaS-palvelu: *Software as a Service* tarkoittaa järjestelmän hankkimista tai myymistä palveluna yksittäisten ohjelmistolisenssien sijaan.

Back-end ja **front-end:** Back-endillä tarkoitetaan sovelluksen osia palvelimella, jossa yleensä ohjelmistologiikka sijaitsee. Sovelluksen käyttäjä ei näe back-endiä. Front-end on sovelluksen asiakaspääte, joka näyttää asiakkaalle sovelluksen visuaalisen ilmeen.

Full-stack kehittäjä: Full-stack kehittäjä kehittää kaikkia järjestelmän osa-alueita (palvelin, tietokanta, **back-end** ja **front-end**), eikä keskity vain yhteen kokonaisuuteen.

Boilerplate-koodi: Lähdekoodi, joka sijaitsee monessa paikassa lähes muuttumattomana.

Refaktorointi: Refaktorointi eli lähdekoodin uudelleenkirjoitus. Refaktoroinnista olemassa olevasta koodista on tarkoitus tehdä parempaa sen toiminnallisuuden pysyessä ennallaan.

Task: Task tarkoittaa ohjelmistokehityksessä yhtä työtehtävää.

Looppaus: Ohjelmointikielen rakenne joka toistaa syötettyä koodia.

2 Lähtötilanteen kuvaus

Kehitysnimellä ”Ilmo” kulkeva tietojärjestelmän kehitys alkoi vuoden 2016 tammikuussa, kun projektin omistajat esittelivät sen Haaga-Helia ammattikorkeakoulun järjestemällä ”SOFTALA III” –kurssilla. Kurssin jälkeen jatkoin projektin kehitystyötä työharjoittelun muodossa entisen asiakkaan ja nyt nykyisen työnantajan toimistolla. Lokakuun aikana otin vastuun tietojärjestelmän kehityksestä omille harteilleni työharjoitteluni päätyttyä.

Tietojärjestelmä on opinnäytetyön aloitushetkellä sen ensimmäisen tuotantoversion kehityskaaren loppupäässä. Päiväkirjan seurantaosuus pureutuu pääasiallisesti viimeisten keskisuurien toiminnallisuuksien toteutukseen ja koko järjestelmän paranteluun sekä optimointiin. Projektin on määrä saavuttaa versio 1.0 maalis-huhtikuun aikana ja siirtyä sen myötä tuotantoon.

2.1 Oman nykyisen työn analyysi

Olen toistaiseksi projektin ainoa tekniseen kehitystyöhön osallistuva työntekijä. Varsinaista työnimikettä minulla ei ole, mutta työkuvani lähestyy vahvasti ”full-stack” kehittäjän kuvausta kattaen mahdolliset ohjelmistokehittäjän työtehtävät.
Järjestelmä nykytilassaan noudattaa kolmitasoarkkitehtuuria, joka on jaoteltu seuraavalla tavalla.

- AngularJS v1 viitekehyksen päälle rakennettu responsiivinen front-end sovellus käytättäen Twitterin Bootstrap 3:sta.
- Spring Boot pohjainen back-end, joka käyttää REST-pohjaista Spring MVC –viitekehystä front-endin kanssa viestimiseen sekä JPA:ta ORM-ratkaisuun tietokannan kanssa.
- MySQL-tietokanta tiedon säilytystä varten.

Järjestelmää kehittäessä uusi isompi kokonaisuus pilkotaan aina pienemmiksi palasiksi ja käsitellään sen sisältämät osa-alueet ja mahdolliset poikkeukset yhdessä projektipäällikön kanssa. Ajallisen arvion toteutettavalle kokonaisuudelle määritän yleensä itse karkeasti, ilmoitan asiasta ja kirjaan sen ylös. Toteuttamisen aloitan itse tarkemmin syväluotaamalla kokonaisuuden, hajottamalla sen pienemmiksi palasiksi ja etsimällä kulmakivet vaatimuksille.

Määrittelen mitä muutoksia tai lisäyksiä järjestelmän tietokanta tarvitsee ja toteutan muutokset JHipster-viitekehystä käyttäen. JHipster generoi määritysten avulla automaattisesti tietokantaan taulut, niitä vastaavat Java-luokat, rajapintaan esimerkki funktioita ja front-endiin vastaavat HTML- ja JavaScript-tiedostot. Tämän jälkeen kirjoitan

vaadittavan ohjelmalogiikan järjestelmän back-endiin ja testaan eri käyttötapauksia lähettämällä kutsuja back-endin rajapintaan. Ollessani tarpeeksi tyytyväinen tulokseen, aloitan front-endillä toiminnallisuuden hahmottamisen ja toteutuksen. Kun front-end näyttää tarpeeksi hyvältä, yhdistän toiminnallisuuden back-endin vastaavaan osaan ja testaan vielä eri käyttötapaukset läpi. Viikon päätteeksi käydään yleensä läpi sen viikon tuotokset ja käydään läpi jos ominaisuudessa tai toiminnallisuudessa on vielä puutteita jotka ovat nousseet pinnalle demoamisen myötä. Jos puutteita on, korjaan ne ja lopulta tallennan muutokset versiohallintaohjelmaa Gittiä käyttäen.

2.2 Osaamiseni taso

Full-stack kehittäminen vaatii monien ohjelmointikielien ja osa-alueiden hallitsemista. Vahvin osaamiseni oli projektin alussa ja on edelleen Javan ohjelmointitaidot. Javaa jo muutama vuoden ohjelmoineena ei toiminnallisuuksien logiikka yleensä tuota suurempia ongelmia. Haluan kuitenkin aina tuottaa aina vain parempaa ja parempaa koodia, välillä myös palaten refaktoroimaan kuukausia vanhaa kirjoittamaani koodia.

Ennen työharjoitteluani projektin parissa, front-end taitojen osaamiseni oli erittäin tyydyttävää ja asenteeni sitä kohtaan vielä heikompaa. Työharjoittelun aikana AngularJS ja HTML5 osaamiseni kehittyessä myös asenteeni kääntyi positiivisempaan päin ja nykytilassa olen paljon itsevarmempi omasta tuotoksestani niiltäkin osin. Varsinaista perinteistä JavaScript pohjaa minulla ei ole, vaan osaamiseni on karttunut juuri AngularJS:n kautta.

Määrittelytaidot ja tarpeiden etsintä kontekstista on kasvanut myös merkittävästi projektin aikana. Nopea ja tarkempi tarpeiden tunnistaminen on johtanut nopeampaan kehitykseen ja tulosten aikaansaamiseen. Ainoastaan graafinen osaaminen on osa-alue jolla tarvitsen vielä paljon kehitystä, jota tulen varmasti tarvitsemaan tulevaisuudessa eri projekteissa.

2.3 Työssä kehittyminen

Koska olen projektini ja yritykseni ainut ohjelmoija, on työni lähes täysin itseohjautuvaa ja itsenäistä. Varsinaisia tilanteita ei ole ollut, joissa työ olisi pysähtynyt täysin ylipääsemättömän ongelman tai esteen takia. Ainoastaan yksi merkittävän järjestelmälogiikan muutos aiheutti ohjelmakoodiin tehtäviä suurempia muutoksia sekä uudelleen kirjoittamista. Koen hallitsevani työni hyvin olosuhteisiin verrattuna ja motivaatio on edelleen yhtä korkealla kuin se oli vuosi sitten projektin alkaessa. Työskentelyni tahtotaso on myös korkealla, sillä harva päivä on samanalainen kuin sitä edeltävä on ollut.

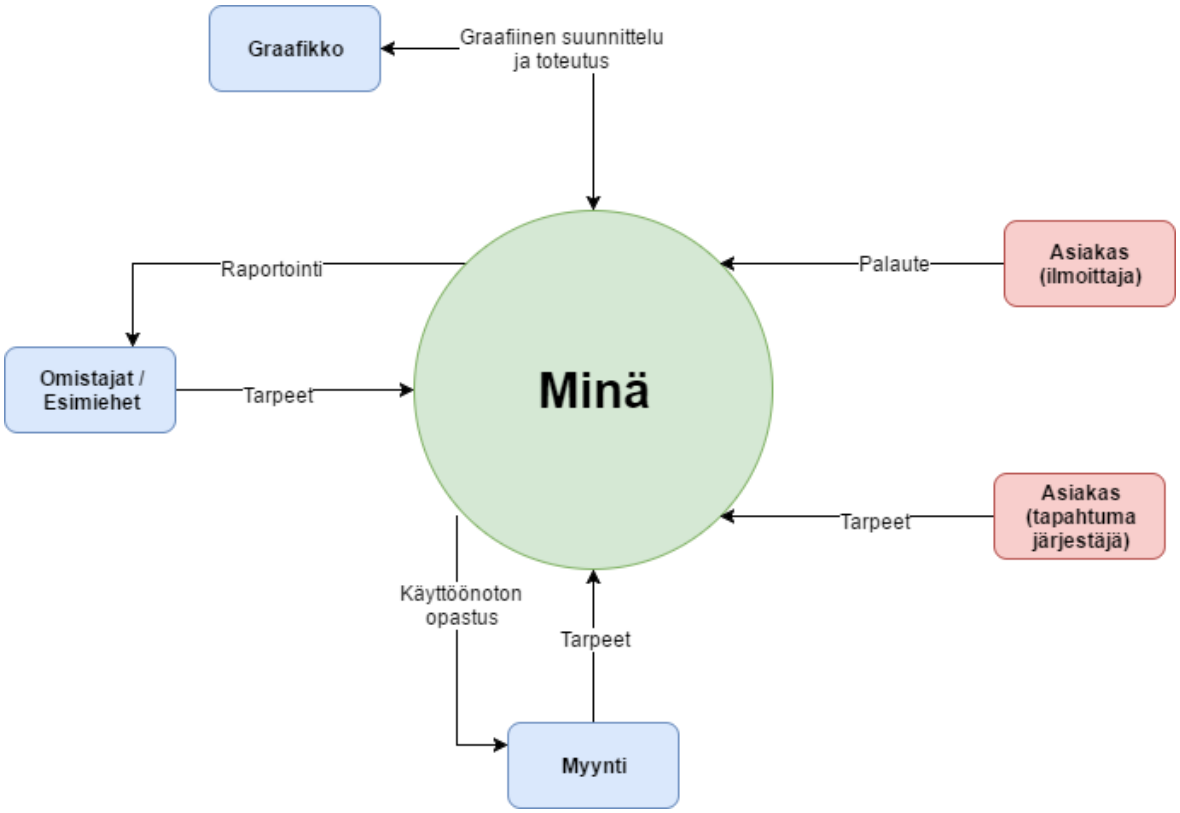
Uusissa ominaisuuksissa tulee lähes aina väistämättä vastaan jotain joka vaatii edelleen lisää opiskelua. Uuden oppinen tapahtuu yleensä etsimällä samantyyppisiä ongelmia Internetistä ja soveltamalla niiden ratkaisuja omaan työhöni. Syksyn aikana olen myös itsenäisesti suorittanut eri oppimisalustojen tarjoamia verkkokursseja. Koen osaamiseni ja sen tuoman itsevarmuuden sijoittamaan minut ”taitavan suoriutujan” kategoriaan. Koen selittyväni hyvin nykyisistä sekä tulevista työtehtävistä, mutta aina on mahdollisuus oppia lisää.

Ammatillisessa kehityksessä olen mielestäni hyvässä vaiheessa työkokemuksen määrässä ja omien projektien pohjalta suhteutettuna opintojeni määrään. Olen mielestäni saavuttanut tarvittavan osaamisen Java-ohjelmointikielessä, jotta voisin tulevaisuudessa työskennellä itsevarmasti sitä käyttäen. Front-end taidot kehittyvät jatkuvasti ja ne alkavat rutinoitua, nykyisin pelkästään dokumentaatiosta asian katsominen riittää hakukoneiden käyttämisen sijasta. En kuitenkaan koe että uuden oppiminen loppuisi missään vaiheessa. Olen asennoitunut oppimaan aina uutta niin kauan kuin ohjelmointi arjessa on pääasiallinen työtehtäväni. Minulle ei riitä että koodi toimii, vaan sen tulee olla myös kaunista. Tämä ei kuitenkaan aina toteudu työelämässä aikarajojen puskiessa jatkuvasti vastaan. Haluan panostaa tällä hetkellä front-end taitojen parantamiseen ja Java 8:n tuoman funktionaalisuuden opettelemiseen.

2.4 Sidosryhmät työpaikalla

Työpaikan sisäiset sidosryhmien määrä on vielä toistaiseksi vähäistä, kun kyseessä on nuori start-up yritys. Järjestelmälle ja minulle merkittävin sidosryhmä on esimies, jolta suurin osa järjestelmän tarpeista ja määrittelyistä on peräisin. Lähitulevaisuudessa tulevia sisäisiä sidosryhmiä tulee olemaan mm. myynti ja graafikko. (Kaavio 1)

Järjestelmässä erilaisia asiakasrooleja on kaksi: tapahtumajärjestäjä ja tapahtumaan ilmoittaja. Tapahtumajärjestäjä pystyy luomaan tapahtumia ja osallistuja pystyy ilmoittamaan koiriaan niihin. Tulen todennäköisesti toimimaan tiiviimmässä yhteistyössä tapahtumanjärjestäjäasiakkaiden kanssa, sillä järjestelmä tarjoaa tälle sidosryhmälle kattavampia ja monipuolisempia toimintoja. Tulevaisuuden tärkeimmät tarpeet järjestelmälle tulevat myös varmasti ko. sidosryhmältä, koska jos tapahtumia ei järjestetä niin ei niihin myös voida osallistua.



Kaavio 1 Sidosryhmäkaavio sisäisistä (sininen tausta) ja ulkoisista (punainen tausta) sidosryhmistä ja niiden vuorovaikutuksista työhöni.

2.5 Vuorovaikutustaidot työpaikalla

Vuorovaikutus työpaikka tapahtuu joko toimistolla tai Slack –ohjelmaa hyväksikäyttäen. Demotilaisuuksissa käydään esimiehen kanssa yksityiskohtaisesti läpi mitä ollaan tehty sitten viime demon jälkeen ja määritellään mitä tulen tekemään seuraavalla viikolla. Demo kestää yleensä tunnista kahteen, riippuen ominaisuuden koosta ja tarvittavat muutokset kirjataan ylös Google Docsiin, joka toimii eräänlaisena työlistana.

Demotilaisuuksien ulkopuolella vuorovaikutus näkyy normaalina keskusteluna joko aiheesta tai sen ulkopuolella. Välillä tulee myös uusia ehdotuksia erilaisista toiminallisuuksista, jotka voisivat asiakasta viehättää. Tulevaisuudessa tulee varmasti tilanteita asiakaspalvelun puolelta järjestelmän toiminnasta ja sen käyttämisestä.

Ollessani ainut ohjelmoija, vuorovaikutusteknisissä aiheissa tuottaa ajoittain hieman vaikeuksia esimerkiksi ongelmatilanteiden, logiikan tai toiminnallisuuksien aikataulujen kanssa. Helpolta näyttävä yksinkertainen muutos tai kaavakkeen lisäkenttä saattaakin vaikuttaa moneen paikkaan järjestelmän lähdekoodissa ja näin kasvattaa työmäärää.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 1

Maanantai 13.02.2017

Aamulla aloitan päivän jatkamalla viime perjantaina keskenjäänyttä ilmoittautumisen määrien seurantaa ja sen toiminnallisuuksia. Toteutan MorrisJS-kirjastolla ympyrädiagrammeja, jotka kuvaavat kuinka paljon jokaiseen kilpailuluokkaan on ilmoitettu koiria ja missä hintaluokassa. Pidämme iltapäivälle demotilaisuuden, jossa esittelen tämän keskeneräisen seurantasivun ja sen jälkeen määrittelemme erilaiset maksupaketit yrityssivulle, joilla näyttelyjärjestelmän ominaisuuksia voi ostaa.

Ympyrädiagrammeissa on vielä bugi, jossa kirjasto tulostaa selaimen konsoliin virheilmoituksia. Bugi johtuu koska näyttelyyn ei ole vielä ilmoittautuneita koiria niin piirakkaa ei voi piirtää. Bugin korjausprioriteetti on aika matala, sillä tälle kyseiselle sivulle ei ole mahdollista päästä ennen kuin näyttelyn ilmoittautumisen on virallisesti alkanut (ja näin ollen dataa piirtää diagrammi on..). Tavoitteena on korjata ko. bugi, muuttaa järjestelmään rekisteröitymisen lomaketta ja demotilaisuuden jälkeen tehdä maksupaketeista alustavat tietokantataulut.

Sain työpäivän aikana enemmän aikaan mitä olin suunnitellut. Rekisteröinnistä poistui kokonaan käyttäjätunnuksen syöttäminen (järjestelmän käyttäjätunnuksena toimii sähköposti) ja sähköpostikentän muoto validoidaan tarkalla ”regular expression”-mallilla. Aloitin myös ylimääräisenä hommana näyttelynjärjestäjän seurantaan "kehänäkymän", jossa koirien tulostensyöttämistä voidaan seurata kokonaisnäkymästä. Piirakkadiagrammin bugia en saanut vielä korjattua front-endin puolella, uskon vahvasti että virheilmoituksilta vältytään, jos logiikka tehdäänkin back-endiin. Alustavat tietokantakaaviot uusista tauluista annoin esimiehelleni, ja aloitan niiden työstämisen JHipsterillä huomenna tiistaina saatuani hyväksynnän. Tänään opin tarkemmin miten ”regex”-validointi tapahtuu ja mistä se koostuu. Opin myös piirtämään MorrisJS:llä erilaisia diagrammeja.

Tiistai 14.02.2017

Järjestelmä tulee käyttämään kolmannen palveluntarjoajan maksuintegraatiota rahaliikenteeseen. Päädyimme yhdessä tähän maksutavan malliin, sillä sen toteuttaminen

on helpompaa ja tietoturvallisempaa. Palveluntarjoajaksi valitsimme Maksukaistan, jonka rajapinnan dokumentaatioon tutustun tänään. Työskentelen etänä kotoa, sillä tämä työpäivä sisältää vain teorian oppimista ja pientä testailua. Päivän tavoitteena on tutustua perin pohjin Maksukaistan rajapintaan ja ymmärtää maksuintegraation sekvenssikaavio. Iltapäivällä katson eilistä piirakkadiagrammi -bugia mikäli aikaa riittää.

Sain päivän päätteeksi hyvän selkeyden Maksukaistan rajapinnasta ja maksuntapahtuman rakenteesta. Rajapintaan tulee lähettää määrämuotoinen HTTP Post pyyntö, joka sisältää tilauksen, tuotteiden, maksutyypin ja asiakkaan tiedot JSON-formaatissa. Jos pyyntö on oikeassa muodossa, vastaus sisältää merkkijonon, jota käytetään URL-osoitteena asiakkaan ohjaamiseksi maksuun. Maksun onnistumisen tai epäonnistumisen seurauksena järjestelmä vastaanottaa erilaisia virhekoodeja. Ne tulee huomioida huomenna aloittaessani implementoimaan maksujärjestelmää.

Tilasin aamupäivällä Maksukaistalta ilmaiset testitunnukset, jolla rajapinnan kanssa pääsee harjoittelemaan. Tunnukset saapuivat jo iltapäivällä, mutta en päässyt niitä vielä kokeilemaan. Yksi tilauksen attribuutti on nimeltään "authcode", joka on SHA-246 salakirjoitettu merkkijono. Kenttä sisältämä teksti tulee salakirjoittaa tunnusten mukana tulleella salausavaimella ennen maksupyynnön lähettämistä Maksukaistan rajapintaan.

Opin tänään Maksukaistan rajapinnan perusteet sekä tavan jolla datan autenttisuus voidaan tarkistaa eri järjestelmien integroinnissa. Salaaminen näyttää luontevalta ja on tärkeää etenkin kun kyseessä on rahaliikenteeseen liittyviä toiminnallisuuksia.

Keskiviikko 15.02.2017

Tänään päämäärä on saada maksuintegraation pyyntö lähtemään Maksukaistan rajapintaan. Teen aluksi pienen erillisen ohjelmakoodin, jolla testaan pyynnön vaatimaa salakirjoitettua kenttää. Kun saan oikean näköisen salakirjoituksen sen tekemästä metodista, kokeilen pyynnön lähettämistä rajapintaan Maksukaistan omalla esimerkillä. Korvaan esimerkin objektista tarvittavat attribuutit vastaamaan generoimaani salakirjoitusta, jotta rajapinta hyväksyisi pyynnön autenttisuuden. Tämän jälkeen luon Java-luokan vastaamaan pyynnöstä palautuvan vastauksen rakennetta ja refaktoroin koiran ilmoittautumisen metodia kattamaan maksupyynnön luonnin.

Sain salakirjoituksen toimimaan hyvin nopeasti. Tällä hetkellä rajapinnan testikäyttäjän API-tunnus ja salasana on kovakoodattu konfigurointitiedostoon. Nämä tiedot tulevat

kuitenkin siirtymään tietokantaan tämän tai ensiviikon aikana turvallisuussyistä. Pyynnön rakenteen luominen ja liittäminen koiran ilmoittautumista käsittelevään ohjelmakoodiin oli aikaa vievää puuhaa muttei vaikeaa. Ainoastaan HTTP-pyynnön lähettämisestä sain kokeilla useamman kerran, koska maksurajapinta ei hyväksynyt lähettämääni objektia. Objekti oli epäonnistumisista huolimatta täysin validi, sillä sen lähettäminen komentoriviltä onnistui. Lopulta selvisi että pyynnössä (eng. ”request”) tarvittavia ylätunnisteita puuttui (viestin sisällön tyyppi = json). Onnistuneesta maksutapahtuman pyynnöstä palautuvaa vastaus pitää sisällään merkkijono ”token”, jolla asiakas ohjataan maksutapahtuman URL-osoitteeseen. Päivän lopussa ehdin vielä luomaan tämän osoitteen sekä palauttamaan sen front-endiin asiakkaan uudelleenohjausta varten.

Osaamiseni ja tietämykseni maksujärjestelmistä (ainakin Maksukaista) kehittyivät päivän aikana. Uskon vahvasti tulen varmasti tulevaisuudessakin implementoimaan kolmannen palveluntarjoaman maksuintegraatioita. Edellä mainitusta syystä, aiheen kiinnostavuudesta ja Maksukaistan erittäin hyvästä dokumentaatiosta johtuen työ sujui erittäin sulavasti.

Torstai 16.02.2017

Torstain päämäärä on saada maksuintegraatiota edistettyä. Asiakkaan saapuessa Maksukaistan sivulle maksulinkin kautta toteutuu yksi neljästä käyttötapauksesta:

- Asiakas suorittaa maksun onnistuneesti
- Asiakas keskeyttää maksun painamalla ”Peruuta”
- Asiakas sulkee selaimen
- Maksukaistan järjestelmä on huoltotilassa

Listan kahden ylimmän käyttötapauksen toteutuessa Maksukaistan palvelu ohjaa asiakkaan ennalta määritettyyn verkko-osoitteeseen takaisin Ilmon sivuille. Verkko-osoite pitää sisällään parametreja (vastauskoodi, tilausnumero, autentikointikoodi). Näitä parametreja käyttäen selvitän saapuneen kutsun autenttisuuden ja jatkan tilauksen kulkua.

Tein uuden HTML-sivun järjestelmään ottamaan vastaan tilauksen toteutumisen parametrit. Sivu näyttää asiakkaalle tilauksen onnistumisen tai epäonnistumisen riippuen siitä, minkä vastauskoodin Maksukaistan järjestelmä palauttaa. Kyseinen sivu näkyy vain hetken, sillä sitä käytetään ainoastaan parametrien välittämiseen back-endiin

jatkotoimenpiteitä varten. Pienen hetken kuluttua asiakas ohjataan automaattisesti hänen omiin tilauksiinsa.
Parametrien saapuessa back-endille järjestelmä selvittää autentikointikoodin oikeellisuuden tallennetulla salausavaimella. Tämän jälkeen järjestelmä päivittää tilausrivin maksetuksi, jos maksu oli hyväksytty. Tilauksen peruuntuessa tilausrivi poistetaan ja tilauksen sisältämät ilmoittautuneet koirat vapautetaan uudelleen ilmoittautumista varten.

Tänään opin kuinka maksutapahtuman palautuminen järjestelmään tulee käsitellä front- ja back-endissä. Tapauksessa on paljon huomioitavia tilanteita ja virhekoodeja jotka tulee käsitellä ennen kuin järjestelmä voidaan julkaista.

Perjantai 17.02.2017

Perjantain tarkoituksena on luoda metodi, jonka järjestelmä ajaa automaattisesti määritetyn syklin välein (eng. ”cron job”). Ohjelmakoodin tarkoitus on hakea kaikki tilaukset, joita ei vielä ole maksettu ja jotka ovat yli tunnin vanhoja tilaushetkestä, sekä poistaa ne tietokannasta. Jokaisen haun tuottaman tilauksesta lähetetään kysely Maksukaistan rajapintaan, tarkoituksena selvittää maksun sen hetkinen tila. Jos maksu on vanhentunut tai sitä ei ole maksettu tunnin sisällä järjestelmä poistaa tilausrivin. Toisena tavoitteena on tarjota ”Omat tilaukset”-sivulla mahdollisuus palata maksuun, tapauksessa jossa käyttäjä sulki selaimen ennen maksun toteutumista.

Tein refaktorointia ohjelmakoodiin, jonka seurauksena maksulinkin pyytämistä käsittelevä koodi ei toimi enää. Refaktoroinnissa tein pyynnönlähettämistä käsittelevästä koodin oman metodin, sillä sitä käytetään toistaiseksi kahdessa paikassa. Kyseinen koodi ottaa vastaan yhteyden luomista varten tarvittavan verkko-osoitteen (Maksukaistan eri osoitteita) ja yhteyden sisältämän JSON –muotoisen kehon. En onnistunut miksi refaktoroinnin jälkeen pyynnön lähettäminen ei enää onnistu.

Maksuun palaaminen vaatii alkuperäisen verkko-osoitteen, jonka Maksukaista lähettää vain kerran tilauksen luomisen yhteydessä. Tietokannassa ei kuitenkaan ollut tälle osoitteelle omaa kenttää, josta se voitaisiin palauttaa myöhemmin. Tänään opin määrittämään metodin ajamaan itsensä automaattisesti ja kasvatin osaamistani maksuintegraation tekemisestä. Löysin myös mahdollisuuksia refaktroida ohjelmakoodia paremmaksi vähentäen samanlaisen koodin toistuvuutta.

3.1.1 Viikkoraportti

Viikko kului erittäin nopeasti uuden asian parissa. Kyseessä on ensimmäinen koskaan implentoimani maksujärjestelmä. Ennen ohjelmoimisen aloittamista olin syksyllä onnistunut miettimään jo valmiiksi millaisia tietokantakenttiä tarvitaan ja tästä syystä pääsin siirtymään suoraan toteuttamiseen. Ainoastaan maksulinkin tallentava kenttä puuttui, mutta sen lisääminen onnistui nopeasti JHipsterin avulla.

Toteutus eteni sujuvasti ja mielestäni loogisessa järjestyksessä, mutta toiminnallisuuksien toteutus oli keskimääräistä hitaampaa. Syynä tähän oli käsiteltävän asian uutuus ja vaadittavan koodin laatu. Rahaliikenteen ollessa ominaisuudessa mukana, koen tarpeen erityisen laadukkaalle ja hyvin testatulle koodille olevan edellytys. Erilaisia virhe- ja toteutuskoodeja on paljon ja niihin kaikkiin tulee varautua. Tulen varmasti refaktoroimaan maksamiseen liittymistä ohjelmakoodia myös tulevilla viikoilla, kun jatkan sen työstämistä.

Osaamiseni kehittyi maksuintegraation ulkopuolelta refaktoroinnin tarpeen tunnistuksessa sekä toteutuksessa. Harjoittelin myös perianteinteisten for-looppien sijasta käyttämään Java 8. version tuomaa ”stream” data prosessointia ja ylipäätään tuottamaan funktionaalisempaa ohjelmakoodia. Kaikki maksamiseen liittyvät Java-luokat on toteutettu ”fluent setters” –menetelmällä, jossa olioon pystyy asettamaan useampia attribuutteja samalla rivillä ilman että ko. olion nimeä kirjoittaa jokaisen asetuksen kohdalla uudestaan. (Kuva 2)

```
CapturePaymentRequest checkStatus
  = new CapturePaymentRequest()
              .setVersion(PaymentConstants.API_VERSION)
              .setApi_key(PaymentConstants.API_KEY)
              .setOrder_number(request.getOrderNumber())
              .setAuthcode(calculateAuthCode(request.getOrderNumber()));

CapturePaymentRequest checkStatus2
  = new CapturePaymentRequest();
              checkStatus2.checkStatus.setVersion(PaymentConstants.API_VERSION)
              checkStatus2.setApi_key(PaymentConstants.API_KEY)
              checkStatus2.setOrder_number(request.getOrderNumber())
              checkStatus2.setAuthcode(calculateAuthCode(request.getOrderNumber()));
```

Kuva 2. Ylempi ohjelmakoodi käyttää ’checkStatus’-olioon attribuuttien asettamiseen ”fluent setters” käytäntöä ja alempi perinteisempää lähestymistapaa asettaessaan attribuutteja ”checkStatus2”-olioon.

Viikon aikana opiskelin paljon Maksukaistan maksuintegraatioon liittyvää dokumentaatiota, jota myös seurasin tiiviisti ohjelmoidessani toiminnallisuuksia. Maksukaistalla asiakas pystyy maksamaan tilauksensa joko pankkimaksulla, luottokortilla tai laskulla. Suoraveloitus on yleisin tapa maksaa ja helpoin tapa toteuttaa joten tuntui luonnolliselta toteuttaa sen toiminnallisuudet ensin. Tutustuin myös Maksukaistan käyttämään SHA (Secure Hash Algorithm) kryptograafiseen tiivistefunktioon, jolla

kehittämäni järjestelmän ja Maksukaistan välinen viestintä autentikoidaan ja salataan. En kirjoittunut salauksen tekevää ohjelmakoodia itse, sillä valmiita ohjelmapätkiä löytyy ja kryptausta ei myöskään yleensä kannata kirjoittaa myöskään itse. (Cox 2015)

Suurimman ongelman kohtasin perjantaina, kun maksupyynnön lähettävä koodi lakkasi toimivasta, vaikka pyyntö lähtee oikeaan osoitteeseen ja sen sisältämä keho on oikea. Bugi ilmeni refaktoroinnin aikana ja en saanut sitä perjantain aikana selvitettyä. Sen korjaaminen on kuitenkin erittäin kriittistä, sillä maksun toteutumista ei voi testata ilman korjaamista.

3.2 Seurantaviikko 2

Maanantai 20.02.2017

Aloitan maanantain jatkan selvitystä siitä, miksi perjantaina ilmestynyt bugi HTTP Post pyynnössä tapahtuu. Bugin aiheuttama koodi debugattu hyvin pitkälle, joten syyn pyynnön epäonnistumiselle pitäisi selvitä tänään. Minulla on vanha epäilys siitä, että ohjelmakoodi ei anna tarpeeksi aikaa vastauksen saapumiselle ja sen takia rajapinta palauttaa pyynnöstä validoinnin virhekoodin. Käytän kutsun lähettämiseen Apachen "CloseableHttpClient"- nimistä luokkaa. Jos saan bugin korjattua, on tänään tarkoituksena myös muuttaa maksutapahtuman uudelleenohjauksen siirtymään suoraan tilausvahvistuksen yhteenvetoon.

En kuitenkaan saanut vielä tänäänkään selvitettyä bugin johtumista ja lähetin sähköpostia Maksukaistan asiakaspalveluun mahdollisista korjausehdotuksista. Bugi aiheutti töiden hidastumista, sillä jokaisen tilaus pitää lähettää Linuxin komentoriviltä ”curlia” käyttäen. Sain tilausvahvistuksen HTML-sivun tehtyä päivän aikana. Parantelin myös metodia, joka maksun epäonnistuessa tai puuttuessa poistaa tietokannasta tilauksen. Aikaisemmin ohjelmakoodi poisti yksi kerrallaan tilaukseen liittyvät alaluokkaan järjestyksessä. Lisäämällä annotaation "orphanRemoval" jokaiseen tilauksen sisältämään alaluokkaan poistaa ne automaattisesti, jos itse tilaus poistetaan tietokannasta.

Tiistai 21.02.2017

Päivän tarkoituksena on korjata pitkäkestoinen maksulinkin pyyntöön liittyvä bugi. Jos bugi ei selviä pikaisesti, aion kokeilla toista HTTP-pyyntöihin liittyvää kirjastoa. Tämän

jälkeen testaan pyynnön ja siirryn tekemään maksun peruuttamiseen ja maksuun palaamiseen tarvittavia toiminnallisuuksia. Maksun peruuttaminen vaatii uuden REST:in päätepisteen pyynnön vastaanottamiseen sekä sitä käsittelevän logiikan luomisen "PaymentService"- nimiseen luokkaan.

Vaihdoin Apachen kirjaston "OkHttpClient" nimiseen kirjastoon. Tehtyäni muutokset pyynnön lähettävään lähdekoodiin pitkään vaivannut bugi katosi kokonaan. Testasin tätä pyyntöä vielä erilaisilla tilauksilla kuitenkin ilman ongelmia. Bugi oli kadonnut ja työn sujuvuus parani huomattavasti.

Keskiviikko 22.02.2017

Päivän tarkoituksena on refaktoroida tilausta käsittelevää metodia. Koodi on kokenut paljon muutoksia ja lisäyksiä viime kuukausina. Esimerkiksi sähköpostin luominen tilauksesta ja uusi maksuintegraatio on kasvattanut kyseisen toiminnallisuuden koodirivien määrää merkittävästi. Aion luoda uuden Java-luokan, joka sisältää ainoastaan tilauksen käsittelyssä tarvittavia metodeja. Refaktoroinnin merkitys tänään on tehdä lähdekoodista modulaarisempaa, jotta sitä olisi jatkossa helpompi ylläpitää ja kehittää jatkossa.

Tilauksen käsittelyn refaktorointi onnistui hyvin. Lähdekoodin määrä väheni huomattavasti ja back-endin vastausaika tilauksen käsittelyssä nopeutui. Muutin myös useita tilauksen sisältämiä Java-luokkia käyttämään "fluent setters"-tyyliä. Tämän lisäksi uutin tilauksen luokkaa JHipsterillä sisällyttämään maksulinkin luonnissa palautuvan "tokenin". Onnistuin korjaamaan vanhaa koodia paremmaksi ja selkeämmäksi. Laajennettu rakenne on nyt arkkitehtuurillisesti parempaa, sillä kaikki tilaukseen käyttämät metodit sijaitsevat omassa luokassaan.

Torstai 22.03.2017

Torstain tarkoitus on muuttaa tietokannasta koirien rekisteröitymisiä sisältävää taulua. Tietokantataulusta on tarkoitus vastata tarpeita paremmin, sekä sisältää useampia viittauksia muihin tauluihin. Yhdistän taulussa muutamia kolumneja yhteen, koska niiden sisältämiä tietoja ei koskaan tarvitse erotella. Viittausten tarkoitus on tehdä koirien ilmoittautumisen seurannasta helpompaa ja vähemmän resursseja kuluttavampaa. Ilmoittautumisrivit voidaan muutosten jälkeen hakea suoraan yhdestä taulusta esimerkiksi rodun, roturyhmän tai näyttelypäivän perustella.

Sain muutettua taulun sekä korjattua kaikki lukuisat virheet, jotka johtuivat uusista muutoksista. Lisäsin myös tilausrivi-olioon uuden attribuutin, joka tulee sisältämään uniikin generoidun merkkijonon. Merkkijonoa voi käyttää myöhemmin esimerkiksi QR-koodin luomiseen ilmoittautuneen koiran numerolappuun.

3.2.1 Viikkoraportti

Tämän vajaan viikon työskentely eteni tahmaisemmin verrattuna edelliseen viikkoon. Viikon alkua vaivannut bugi maksutokenin pyynnössä hidasti muuta työskentelyä selvästi. Bugin ratkaiseminen olikin viikon tärkein tehtävä. Kun tämä ohjelmavirhe ei ratkennut laajan debuggaamisen ja refaktroinnin jälkeen, päätin että eteneminen on tärkeämpää kuin selvittää virheen syytä. OkHttpClient-kirjaston avulla pyyntö onnistui heti ensimmäisellä yrityksellä Maksukaistan rajapintaan. Aina työelämässä ei ole aikaa jäädä selvittämään kriittistä, mutta pientä bugia aikataulun puitteissa. Uusi kirjasto ratkaisi tämän ongelman, mutta tulevaisuudessa sillä voi olla mahdollisuus luoda niitä lisää. Kirjasto voi tuoda lisää päivitettävää ja mahdollisia yhteensopivuusongelmia jatkokehityksessä. JHipster sisältää useita Javan kehitystyökalun ulkopuolisia kirjastoja, ja vastaa niiden versioista ohjelmaa päivittäessä.

Viikon aikana opin tekemään maksuintegraation Javalla back-endiin, sekä syvempää teoriaa mitä kolmannen maksuintegraatio palveluntarjoajien kulissien takana tapahtuu. Maksuintegraatiota toteuttaessa todella tärkeää on luoda järjestelmään valmius eri virhekoodeille, joita maksutapahtumassa tai asiakkaan toiminnasta voi aiheutua (Babora API Docs 2017). Suurin riski on, että järjestelmä poistaa tilauksen maksamattomana aikarajan puitteessa, vaikka asiakas on juuri sen maksanut. Tähän epätodennäköiseen käyttötapaukseen on kuitenkin varauduttava esimerkiksi lisäämällä tietokannan lukitsemisia tärkeiden toimenpiteiden aikana tai hakemalla tilausrivi uudelleen juuri ennen sen poistoa. Poistettava tilauksen voisi myös tallentaa erilliseen tauluun joka sisältää vain poistettuja tilauksia ilman suhteita muihin tauluihin. Nämä toiminnallisuudet vaativat kuitenkin lisää työtä eivätkä varsinaisesti edistä järjestelmän etenemistä tuotantoon, mutta ovat kriittisiä sekä vähentävät mahdollista työtä tulevaisuudessa.

3.3 Seurantaviikko 3

Maanantai 27.02.2017

Maanantain tarkoitus on määritellä näyttelyjärjestelmän tilaukseen tarvittavat tallennettavat tiedot. Näyttelyn ostaja voi valita pakettiin erilaisia ominaisuuksia hinnan määräytyessä niiden mukaan. Näitä varten tarvitaan:

- HTML-sivu, jossa näyttelyn ominaisuudet valitaan
- HTML-sivu, jossa näyttelyn nimi, tyyppi ja päivämäärät asetetaan
- HTML-sivu tilausvahvistusta varten
- Maksusivuun siirtyminen tilausvahvistuksesta

Kun näyttelypalvelu on tilattu ja maksettu, pääsee asiakas luomaan yhden näyttelyn valitsemillaan ominaisuuksilla. Päivän tavoite on luoda pohjat näille sivuille sekä tehdä hahmotelma tietokantaan lisättävistä tauluista. Tärkeää näyttelyn ominaisuuksien tallentamisessa on tehdä rakenne siten, että sitä on helppo laajentaa kun lisäominaisuuksia kehitetään. Myöskin pakettien hinnoitteluun tulee varmasti muutoksia tulevaisuudessa ja niiden tarjottavuus voi vaihdella (A/B-testaus).

Tein tehtyä alustavia rautalanka-muotoisia hahmotelmia HTML-sivuista. Näyttelypaketit esitellään sivulla vierekkäin ja niistä voidaan checkboxeilla valita ominaisuuksia hinnan päivittyessä dynaamisesti. Tietokannan taulujen määrittelemiseen meni aikaa hieman enemmän mitä alunperin suunnittelin. Päädyin kuitenkin tekemään ratkaisun joka koostuu neljästä uudesta taulusta. Räätälöinti mahdollisuuksien takia suunnitteleminen ei ollutkaan suoraviivaista, mutta uskon että tämä ratkaisu täyttää tarpeet toistaiseksi.

Keskiviikko 01.02.2017

Tänään tarkoitus on luoda näyttelyjärjestelmän tilauksen luokat JHipsterin entity-generaattorilla. Generaattoriin syötetään terminaalista tulevalle luokalle nimi, attribuutit, niiden tietotyypit ja suhteet muihin jo generoituihin luokkiin. Generaattori tämän jälkeen luo luokalle tietokantataulun, Java-luokan, controllerin, servicen sekä front-endiin vastaavat tiedostot. Kun saan kaikki luokat tehtyä generaattorilla, ohjelmoin tarvittavan logiikan niihin. Tilauksen olion koostumus on melkein samanlainen kuin koiran ilmoittautumisessa käytettävä rakenne.

Sain kaiken generoitua onnistuneesti, ainoastaan luokkien ja attribuuttien nimeämisessä kului hetki. Tarkoitukseni oli luoda mahdollisimman itse itseään kommentoivaa lähdekoodia, jotta sen työstämiseen tulevaisuudessakin olisi helpompaa palata. Luokkia generoidessani huomasin, että asiakkaille näytettävät mallipaketit täytyy myös lisätä kantaan. Mallipaketteja ei voida kuitenkaan luoda ilman uutta tietokantataulua, koska jokainen räätälöity paketti liittyy yhteen 'EventOrder'-olioon. Mallipakettien tietoja tulee voida myös muuttaa, joten front-endiinkään niitä ei voida kovakoodata. Ratkaisin ongelman luomalla muista olioista irrallisen luokan, joka sisältää attribuutteja eri näyttelytyyppien ja ominaisuuksien hinnoittelusta. Tätä taulua voidaan myöhemmin päivittää adminpaneelista, jossa myös myyntisivulla esitettävien mallipakettien luonti tapahtuu. Asiakkaan valitessa räätälöidyn paketin, sen hinta tulee muodostumaan tämän taulun senhetkisen hinnoittelun mukaan.

Torstai 02.02.2017

Huomasin aamulla puutteen tilauksen rakenteessa. Vaikka näyttelyjärjestelmän tilaussivu on suunnattu uudelle asiakkaalle, tulee myös vanhan asiakkaan pystyä tilaamaan uusi näyttely. Tämä tarkoittaa rakennemuutosta omistussuhteisiin, näyttelytilaus ei omista tilaajaa vaan tilaaja voi omistaa usean näyttelyn. Tämä myös tarkoittaa, että jo näyttelyn tilannut asiakas tarvitsee erillisen sivun, jossa uuden näyttelyn voi tilata.

Loin uuden taulun ’yritysasiakas’, joka yhdistää käyttäjän useaan näyttelyntilaukseen. Taulu sisältää myös asiakaskohtaiset maksukaistan api- ja salausavaimen. Näyttelyn tilaus onkin monipuolisempi ja enemmän käyttötapauksia sisältämä kokonaisuus kuin aikaisemmin kuvittelin. Huomaan ohjelmoidessani joidenkin tarpeiden ilmentyvän vasta kun käyttötapaus ilmenee ja tämän seurauksena tehdä muutoksia tietokantaan. Voisin keskittyä paremmin suunnittelemiseen ja määrittelemiseen, vaikka siihen ei paljoa olekaan käytettävissä aikataulun puitteissa. Muutin näyttelytilauksen tilauksen kaikki hinnat ”double”-tietotyypistä ”BigDecimal”-tietotyyppiin. Törmäsin kirjoitukseen, jossa painotettiin rahamäärien oikean tallennuksen tärkeyttä. Rahamäärät tulee tallentaa ainakin kolmen desimaalin tarkkuudella ettei pyöristysvirheitä synny laskutuksessa tai veroihin liittyvissä asioissa. Tulen korjaamaan myös koiran ilmoittautumisen hintojen tietotyypit vastaamaan BigDecimal-tietotyyppiä.

Perjantai 03.02.2017

Tänään töissä jatkan maksupakettien HTML sivujen luomista. Lisään mallipaketit ominaisuuksineen ensimmäiselle sivulle, josta mahdollinen asiakas voi valita haluamansa. Jos asiakas valitsee jonkin paketeista, ohjataan hänet uudelle sivulle jossa paketin sisältämiä ominaisuuksia voidaan muokata. Ohjelmoin siis sivun, joka muodostuu edellisellä sivulla valitun paketin mukaan. Ominaisuuksien lisäksi sivulla syötetään asiakkaan tiedot ja näyttelyn luomista varten tarvittavat perustiedot.

Sain ohjelmoitua molemmat sivut loppuun. Kun aloin luoda lomakkeita sivulle jossa asiakkaan tiedot syötetään, huomasin että tietokannan ’näyttelyasiakas’-taulusta puuttuu näyttelyjärjestäjän asiakastiedot. Lisäsin tarvittavat asiakastietoihin liittyvät attribuutit luokkaan JHipserillä ja loin lomakkeen tilausriville. Poistin näiltä "yrityssivuilta" saman navigointipalkin mikä näkyy ohjelman tavallisilla sivuilla. Nämä sivut tulevat saamaan tulevaisuudessa myyntisivuja vastaavaan graafisen ilmeen.

3.3.1 Viikkoraportti

Viikon työt edistyivät mallikkaasti, sain näyttelymyyntisivun hyvin alkuun. Näiden myyntisivujen ulkonäön on tarkoitus olla muusta sivusta poikkeavaa. Muun sivuston tyylin tulee olla suoraviivaista ja erittäin käyttäjäystävällistä, sillä järjestelmän merkittävin kohderyhmä ei ole teknisesti osaavin. Myyntisivujen näyttävyys ja houkuttelevaisuus vaikuttavat ehdottomasti mahdollisten asiakkaiden ostopäätöksiin, joten nämä sivut vaativat visuaalisesti enemmän työtä. (Andew 2015) Tarkoituksenani oli luoda vain rautalankamalli ja pohjan myyntisivuille, jossa voin tulevalla viikolla testata näyttelyn ostoon liittyviä toiminnallisuuksia. Varsinaisen myyntisivun tulee varmaankin suunnittelemaan yrityksen ulkopuolinen graafikko.

Muutoksia eri olioiden rakenteeseen tulee edelleen usein, vaikka koen suunnitelmani tietorakenteet kohtalaisen hyvin. Pieni perfektionismi ei anna minun jättää rakennetta virheelliseksi vaan se tulee heti korjata paremmaksi tarpeiden ilmaantuessa. Kriittinen ja välitöntä korjausta kaipaava asia oli tilauksen hintojen tietotyppien muutos BigDecimaliin, josta mainitsinkin torstain kirjoituksessa. Hintaa käsitellessä numeron muoto tulee olla kahta desimaalia pidempi pyöristyksestä johtuvien ongelmien takia. (Toniut 2015) En tee uusien attribuuttien lisäyksiä tai muutoksia suoraan lähdekoodiin vaan muutan koko luokan rakennetta komentoriviltä JHipsteriä käyttäen. Pelkkä bean-tasoisen luokan tietotyypin muutos ei riitä, vaan muutos täytyy tehdä myös tietokannan XML-

konfiguraatioon sekä front-endin controllereihin, serviceihin ja kielitiedostoihin. Muutosten jälkeen poistan ja ajan tietokannan taulujen luomisessa käytettävät scriptit uudestaan muutosten kanssa. Kehitysvaiheessa en koe tietokannan migraatiota tarpeelliseksi, vaan enemmänkin työnkulkua hidastavaksi tekijäksi. Työnkulkua kuitenkin hidastaa uusien muutosten tuottama testauksen tarve, jonka suoritan manuaalisesti. Ainoana kehittäjänä ja osaamisen puutteen vuoksi minulle ei jää aikaa luoda JUnit-testejä jokaiselle toiminnallisuudelle.

Front-endin puolella erillisten myyntisivujen luominen oli miellyttävää puuhaa. AngularJS toimii ”yksisivuinen”-appi periaatteella, jossa kaikki applikaation alasivut eli näkymät (eng. ”views”) upotetaan pääsivun elementtien sisään. Sivustoa navigoidessa sivua ei ladata kokonaan uudestaan vaan elementtien sisältämä data muuttuu. (AngularJS 2017) Pääsivulla on kolme elementtiä joiden sisälle ladataan järjestyksessä navigaatiopalkki, sivustonäkymät ja alatunniste. Haasteena oli piilottaa muun sivuston käyttämä navigaatiopalkki ja näyttää kokonaan toisenlainen palkki.

Projektin front-end ja back-end yhdistetään rakennusvaiheessa yhdeksi war-tiedostoksi, joka voidaan ajaa palvelimella. Tätä arkkitehtuurimallia kutsutaan monoliittiseksi, jolle vaihtoehtona on tuoreempi microservice-malli. Microservice-mallissa järjestelmä on rakennettu tuotannossa useampaan eri suoritettavaan pienempään järjestelmään. Nämä pienet osat toimivat itsenäisesti omana applikaationaan ja voivat keskustella keskenään esimerkiksi yhteisen rekisteröintipalvelimen kautta. Tätä arkkitehtuuria käyttäen Ilmon voisi jakaa useampaan pienempään järjestelmään:

- Käyttäjä/kirjautuminen/rekisteröityminen
- Maakohtainen näyttelyyn ilmoittautuminen
- Tapahtumanjärjestäjän hallintapaneeli
- Tulospalvelu

Monoliittisuudessa kriittinen ongelma on järjestelemän vakaus, järjestelmä kaatuessa kaikki toiminta lakkaa. Myöskin applikaatiota päivittäessä tulee se asentaa kokonaan uudestaan. (Richardson 2017) Microservicen rakennetta käyttäessä yhden applikaation liikenteen ruuhkaantuminen tai totaalinen kaatuminen ei vaikuta muihin järjestelmän osiin yhtä merkityksellisesti. Esimerkiksi käyttäjän rekisteröinnin tai koiran lisäämisen eivät ole keskenään riippuvaisia ja näin ollen niiden ei tulisi vaikuttaa toisiinsa.

Koska microservice-arkkitehtuuri koostuu monesta pienemmästä applikaatioista, niitä integroidessa eri applikaatioiden ei tarvitse sisältää samoja kirjastoja tai edes samaa

ohjelmointikieltä. (Haddad 2015) Modulaarisuus mahdollistaa uusien ominaisuuksien kehittämisen helpommin eri tiimeissä, sillä muutokset tapahtuvat vain yhdessä microservicen applikaatiossa.

Ilmon kehitystä aloittaessa en ollut vielä kuullutkaan microserviceistä ja sen arkkitehtuurin tuomista hyödyistä. Monoliittinen rakenne kuitenkin tyydyttää järjestelmän tarpeet hyvin toistaiseksi, mutta tulevaisuudessa tietokannan ja lähdekoodin kasvaessa tilanne voi olla toinen. Järjestelmän sisältäessä 54 data-oliota ja niitä käyttävien muiden luokkien määrä kasvattaa kehitysympäristön kokoa merkittävästi. Kokonaiskuvan saaminen järjestelmästä sen laajuuden takia voi olla haastavaa uudelle ohjelmoijalle, mutta ollessani ainut kehittäjä minulla on täysi tuntemus siitä.

Monoliittista järjestelmää on helpompi kehittää yksin, testata ja asentaa palvelimelle. Kehityksestä jokaista microservicen osaa täytyy ajaa erikseen taustalla, jos niitä tarvitaan. Ilmon ensimmäinen versio tehdään loppuun monoliittisella arkkitehtuurilla, mutta JHipsterillä on myös luoda microservice-tyyppisiä projekteja, joten tulevaisuudessa migraatio uuteen arkkitehtuuriin on myös mahdollista.

3.4 Seurantaviikko 4

Maanantai 06.03.2017

Tämän viikon tarkoituksena on tehdä näyttelytilauksen back-endin ohjelmalogiikka valmiiksi. Back-endille saapuva näyttelytilaus-olio koostuu neljästä muusta oliosta: tilaajasta, näyttelyn perustiedoista, räätälöidystä paketista sekä itse tilauksesta. Springin REST ottaa olion vastaan, jonka se pilkotaan osiksi ja osien omat servicet käsittelevät ne. Jos prosessi menee läpi ilman virheitä tai ongelmia, restcontroller palauttaa front-endiin URL-osoitteen tilauksen maksamista varten. Järjestyksessä oliota käsittelevät servicet tekevät seuraavat toimenpiteet:

1. Tilauksen tekijälle luodaan käyttäjä järjestelmään "näyttelyomistaja"-roolilla.
2. Valitut näyttelyominaisuuksien hinta lasketaan
3. Luodaan uusi tapahtuma ja generoidaan luokat valituista ominaisuuksista ja näyttelytyypistä
4. Käsitellään tilauksen olio
5. Maksuintegraatio

Päivän aikana ohjelmoin tilaajan käyttäjäluontiin liittyvät toiminnallisuudet. En kuitenkaan voinut käyttää olemassa olevaa käyttäjäluonti-metodia, koska tilaus-olion sisältämä asiakas-olio kattaa enemmän tietoja kuin järjestelmän rekisteröityvä ilmoittaja-tason käyttäjä. Tämän tason käyttäjä tulee aktivoida tunnus sähköpostilla, kun taas näyttely tilaaja-tason käyttäjä aktivoidaan onnistuneen maksutapahtuman seurauksesta. Ongelmalliseksi tilaajan käyttäjäluonnissa osoittautui salasanaan liittyvät haaste. Front-endillä tilauksessa ei kysytä tulevan käyttäjän salasanaa, vaan se generoidaan back-endissä. Tällä hetkellä tilauksen maksamisen onnistuessa tilaajlle lähetetään generoitu salasana sähköpostilla tunnusten aktivoituessa. On kuitenkin tietoturvallisesti vaarallista lähettää salasanoja ns. "plain textillä", joten tähän täytyy palata vielä jatkossa. Tilaajan voisi ensimmäisellä kirjautumiskerralla ohjata vaihtamaan salasana heti.

Tiistai 07.03.2017

Päivän tarkoituksena on jatkaa tilauksen käsittelyä back-endissä. Vuorossa on tilaajan räätälöimän paketin kokonaishinnan laskeminen ja pakettiolion tallentaminen tietokantaan. Paketin hinnan laskeminen tapahtuu hakemalla tietokannasta hinnat ominaisuuksille, joita tilaaja on valinnut.

Puolen päivän jälkeen minulla on tarkoituksena esitellä järjestelmä läpi graafikon kanssa "join.me"-palvelun kautta. Graafikko tulee tekemään demon perusteella tarjouksen sivuston ulkoasun suunnitellusta ja toteutuksesta.

Tilauksen hinnan uuden tietotyypin (BigDecimal) takia hinnan laskeminen ei ollut yhtä suoraviivaista kuin se olisi laskettu double-tietotyypillä. BigDecimalia käyttäessä jokainen olio on muuttumaton luonnin jälkeen, ainoastaan luomalla uuden instanssin oliosta voidaan tämän tietotyypin arvoja laskea yhteen. BigDecimal-luokan "add" metodi ajaa tämän asian. Alla olevassa lähdekoodissa BigDecimal-tietotyypin määrä alustetaan hakemalla paketin pohjahinta "featurePricingService"-nimisestä palvelusta, jonka jälkeen tähän hintaan lisätään "kotisivun" hinta, mikäli tilaaja on sen ominaisuuden valinnut.

Iltapäivän graafikolle suunnattu demotilaisuus sujui kohtalaisen hyvin, uskon että hän sai loppujenlopuksi riittävän kuvan järjestelmän rakenteesta. Oma ongelmani oli edetä hitaasti ja selittää tarkasti järjestelmän eri osa-alueita, järjestelmän ollessa minulle erittäin tuttu ja ollessani tottunut jo sen eri osien navigointiin. Keskustelussa kävi ilmi, että myyntisivut luodaankin ehkä erikseen Ilmo-järjestelmän ulkopuolelle. Itse kannatan ideaa, sillä

myyntisivujen sisällöntuotantoa voisi tehdä irrallisena muusta Ilmosta. Myöskään tekemäni työ ei mene hukkaa, sillä palaavan näyttelytilaaja-asiakkaan täytyy kuitenkin tehdä uusi tilaus Ilmon kautta. Haasteena tässä implementaatiossa on luoda integraatio myyntisivujen ja Ilmon välillä.

Keskiviikko 08.03.2017

Eilen ohjelmoimisen jääneessä normaalia vähemmälle järjestelmän esittelyn seurauksena, jatkan tänään tilauksen käsittelyn back-end osaa. Päivän tarkoitus on luoda näyttely-olio asiakkaan tilauksesta ja liittää se itse tilaukseen. Näyttelyn luominen on jo järjestelmän vanha ominaisuus, mutta sitä tulee uuden tilauksen yhdistämisen takia laajentaa. Vanhassa versiossa näyttelyjärjestäjä pystyy luoda näyttelyn kaikilla mahdollisilla ominaisuuksilla, mutta tämä muuttuu nyt maksupakettien mukaiseksi. Adminin tulee kuitenkin jatkossa pystyä luomaan näyttely manuaalisesti front-endin käyttöliittymän kautta ilman tilauksen tekoa tai maksua.

Onnistuin tekemään tarpeelliset laajennukset ja refaktoroinnin näyttelynluonnista back-endiin ilman ongelmia. Vanha versio käytti "createEvent"-nimistä metodia joka parametreissa vastaan perustiedot näyttelystä sekä käyttäjätunnuksen. Ero perustiedoissa on se, että tilaajaa ei syötä kaikkia tietoja tilaukseen heti, vaan vasta myöhemmin näyttelyhallinnan käyttöliittymän kautta. Tein tämän metodin "createEventForCustomer", joka syöttää uuden version (puutteelliset) perustiedot näyttely-olioon ja sen jälkeen kutsuu olemassa olevaa "createEvent"-metodia.

Näyttelyluonnin haasteena on se, että päivämääriä jolloin näyttely pidetään ja näyttelyn tyyppiä, ei voida muuttaa tilauksen jälkeen. Syy näille estoille johtuu, että moni näyttelyn rakenteen olioista ovat riippuvainen näistä arvoista. Näyttelyn tulisi siis pystyä poistamaan ja luomaan uudestaan sen tilauksen rajoitteissa.

Torstai 09.03.2017

Torstaina tarkoituksena on luoda itse maksutilaus-olio, johon tämän viikon muu kehittäminen on johtanut. Tämän olion luonnin ohjelmoimisessa ei tulisi olla aikaa vievää. Olion tallentuessa tietokantaan onnistuneesti sisältäen paketin, hintatiedot, näyttelyn ja käyttäjän, siirryn aloittamaan maksuun siirtymistä tilauksesta. Maksupyyntö-olion

luomiseen voin hyödyntää osaa näyttelyyn ilmoittautumisessa käytetyistä metodeista, mutta ainakin tuotelistan luominen täytyy tehdä uudelleen.

Päivän aikana sain kaiken suunnitellun tehtyä. Tilaus tallentui tietokantaan onnistuneesti ja oikeilla arvoilla. Refaktoroin maksuintegraatiota käsittelevää "PaymentService"-nimistä luokkaa atomisemmaksi. HTTP-post pyynnön palauttava metodi ottaa nyt parametrina vastaan pyynnön kehon ja osoitteen. Pyynnön keho voi olla, joko näyttelyyn ilmoittautuminen, tai näyttelyn tilaaminen ja se luodaan omissa metodeissaan.

Maksun luomisen onnistuessa Maksukaistan rajapintaan, tallensin URL-osoitteen tokenin tilausriville tietokantaan ja määritin front-endin ohjaamaan Maksukaistan osoitteeseen, jossa käyttäjä maksaa tilauksen. Tilauksen jälkeen käyttäjä ohjataan takaisin järjestelmän tilauksen vahvistuksen laskeutumissivulle.

Perjantai 10.03.2017

Kuten näyttelyihin ilmoittamisen maksussa, myös näyttelyn tilauksen maksusta palautuu tieto Ilmoon. Perjantain tehtävänä on luoda toiminnallisuudet maksutapahtumasta suoriutumisesta kertoville eri palautuskoodeille. Onnistuneen maksun seurauksena asiakkaalle lähetetään kuitti tilauksesta, sekä kirjautumistiedot erillisenä sähköpostina. Epäonnistuneessa tilauksessa näyttely, tilaus ja käyttäjä poistetaan tietokannasta. Käytän olemassa olevaa, automaattisesti ajettavaa metodia näyttelytilausten hakemiseen ja niiden käsittelyyn.

Sain tehtyä toiminnallisuudet onnistuneelle, epäonnistuneelle ja keskeytyneelle tilaukselle. Toisin kuin ilmoittautumisen, ei keskenjääneeseen maksutapahtumaan pääse enää käsiksi jos siitä on kerran poistunut. Päivän työt olivat lähinnä työlästä rutiinimaista ohjelmointia, mitään uutta en varsinaisesti oppinut.

3.4.1 Viikkoraportti

Viikon merkittävin haaste oli laajentaa olemassa olevia toiminnallisuuksia ilman että niitä ennestään käyttävät metodit eivät vaikutu tai rikkoudu. Tapahtuman voi luoda joko Ilmon admin-tason käyttäjä tai tilauksella ja maksutapahtuman kautta tapahtumajärjestäjä. Ero näiden kahden välillä on näyttelyjärjestäjä suorittaa maksutapahtuman prosessin ja

hänelle luodaan käyttäjä, kun taas järjestelmän ylläpitäjä voi luoda tapahtuman kenelle tahansa käyttäjälle ilman että siitä syntyy tilausta (esim. demotilaisuuksissa). Oleellista on välttää lähdekodin kopioitumista (eng. ”code duplication”), vaikka nämä molemmat prosessit ovat sisällöltään lähes yhtenevät tapahtuma-olion ja sen alaluokkien luomisessa. Ratkaisussani molemmat käyttötapaukset johtavat samaan polkuun, kunnes kaikki muu tarpeellinen on käsitelty. Tässä tapauksessa maksutapahtuman onnistuminen ja näyttelyjärjestäjälle käyttäjän luominen.

Viikon tavoitteisiin pääsin hyvin ja maksuintegraatio on toistaiseksi käsitelty, ainoastaan näyttelytilauksen olioon ja toiminnallisuuteen tulee muutoksia. Havahduin ettei näyttelytilauksen yhteydessä kannattaisi vielä luoda itse näyttelyä tai käyttäjää vastaaviin tauluihinsa. Jos nämä tilauksen alaoliot luotaisiin vasta onnistuneen maksutapahtuman seurauksena ei näyttelytilausta tai käyttäjätiliä tarvitsisi luoda ollenkaan ja näin väliaikainen salasanakin voitaisiin generoida juuri ennen sähköpostin lähettämistä. Tämä vaihtoehtoinen rakenne nopeuttaisi tilauksen käsittelyä ostohetkellä ja vähentäisi tarpeetonta tietokannan rivien poistoa.

Ilmon tarkoitus oli alunperin pystyä taipumaan erilaisiin mahdollisiin muuttujiin, jopa sen keskeisten ominaisuuksien rakenteessa. Eri maiden koiranäyttelyt voivat sisältää erilaisia sääntöjä näyttelyyn ilmoittautumisessa, tulosten syöttämisessä tai muissa rajoitteissa. Järjestelmän tekninen suunnittelu ja toteutus on ollut raskaampaa, sillä mahdollisia tulevia käyttötapauksia ei voi vielä tietää. Tämä on johtanut modulaariseen ohjelmistosuunniteluun, jossa jokainen käyttötapaus on jaettu pieniin paloihin sisältäen vain tarvitun toiminnallisuuden. Tämän suunnittelun tyypillisesti vaatii jatkuvaa viikoittaista ponnistelua ja välitöntä refaktorointi uusien puutteiden ilmaantuessa.

Haastattelemamme graafikko ei tuntenut front-endin käyttämiä teknologioita (Node, AngularJS). Teknisten tietotaitojen puutteen vuoksi jalkauttaminen kehitysympäristöön ja itse järjestelmään olisi vienyt useita työtunteja. Kehitysympäristön asentaminen vaatii paikallisen tietokannan luomista, back-endin ja front-endin ajoa sekä näiden tarpeellisten kirjastojen ja viitekehyksien asennuksen.

Tämä on yksi syistä jonka takia järjestelmää on ollut kätevää kehittää yksin. Versiohallinnassa ilmeneviä ”merge”-konflikteja ei tule lainkaan, sillä ainoana kehittäjänä kukaan muu ei tee muutoksia lähdekoodiin. Myöskään tilanteita, joissa erimielisyydet rakenteesta, ohjelmalogiikasta tai arkkitehtuurista johtaisivat kehityksen hidastumiseen ei synny. Voin itse päättää olioiden ja niiden attribuuttien nimistä ja minulla on automaattinen selvyys järjestelmän jokaisen osan toiminnasta. Tästä johtuen minun ei tarvitse lukea

eikä ymmärtää toisen kehittäjän lähdekoodia, vaan ainoastaan omaa vanhempaa tuotostani. En joudu myöskään odotella toisen toiminnallisuudet valmistumista, jotta pääsen toteuttamaan omaani.

Ainona teknisenä kehittäjä keskisuuren projektin parissa työskentelemisen edut vaikuttavat lähinnä konfliktien määrän ja työn jouhevuuteen. Oppimisprosessi on täysin itseni vastuulla ja sisältää usein artikkeleiden, dokumentaation tai kirjojen lukua. En saa mahdollisuutta oppia seniorin tuella tai toisen juniori tason ohjelmoijan kanssa. Pariohjelmoinnin kautta rakenne -ja logiikkavirheet vähenisivät useamman silmäparin tarkastellessa lähdekoodia, puhumattakaan nopeammin toteutuvista toiminnallisuuksista useamman ohjelmoijan työskennellessä saman projektin parissa. Ainoastaan oma periaatteeni ja viitseliäisyyteni vaikuttavat hyvän lähdekoodin tuottoon, kun tiimityöskentelyn ”code review” -prosessin auttaisi selvittämään vastaako koodi tarpeita. (Pink 2008) Kun kukaan ei katselmoi lähdekoodiani, on helppo lipsua hyvien ohjelmointikäytäntöjen ulkopuolelle. Lipsuminen on lähinnä oman kuopan kaivamista, sillä omat tehdyt virheet ja laiskuudet tulee varmasti kohtaamaan myöhemmin.

3.5 Seurantaviikko 5

Maanantai 13.03.2017

Viime viikonloppuna sain idean parantaa näyttelyiden tilauksen logiikkaa. Tällä hetkellä näyttelyn tilaus tallennettaan ennen maksutapahtumaa, jolloin järjestelmä luo asiakas- ja näyttely-olion tietokantaan. Ongelmaksi tässä ratkaisussa syntyy epäonnistuneet sekä keskenjääneet tilaukset. On täysin turhaa lisätä ja poistaa tietokannasta tilauksia, jotka eivät toteudukaan. Uudessa ratkaisussa laajennan NäyttelyTilaus-olion kattamaan kaikki tallentamiseen tarvittavat tiedot siten että käyttäjän ja näyttelyn luominen suoritetaan vasta onnistuneen maksutapahtuman seurauksena.

Rakenne on jälleen järkevämpi kuin ennen, lisäsin attribuutit NäyttelyTilaus-olioon sekä muutin käyttäjä- ja näyttelyluonnin tapahtumaan vasta maksutiedon saapuessa back-endiin. Koska tunnusten luominen suoritetaan vasta tilauksen jälkeen, ei generoitua salasanaa tarvitse tallentaa mihinkään ennen sen lähettämistä asiakkaalle sähköpostitse. Muutoksen seurauksena itse näyttelytaulusta tietokantaa lukittavia poistotoimenpiteitä ei tule muuten kuin poikkeustapauksissa. Taulu on rakenteen hierarkiassa korkealla, joka kasvattaa merkittävästi sen tietokannan lukemisen määriä.

Tiistai 14.03.2017

Tänään luon pohjan onnistuneen näyttelytilauksesta lähtevälle sähköpostilla. Sähköpostien luonti tyylillisesti on erittäin aikaa vievää työtä ja en keskity siihen tänään. Vaivalloiseksi HTML-sähköpostin rakenteen luomisessa on se, että CSS-tiedostoa ei voida käyttää, vaan tyyli pitää manuaalisesti lisätä jokaiseen elementtiin. Työ HTML-muotoisen sähköpostin luomiseen on erittäin mekaanista ja aikaa vievää.

Tämän lisäksi konfiguroin Spring Securityä sallimaan näyttelytilauksen rajapinnan päätepisteille pääsyn ilman kirjautumista. Konfigurointi tapahtuu manuaalisesti määrittelemällä jokainen päätepisteen osoite ja yhdistämiseen vaadittava käyttäjän rooli. Front-endin puolella AngularJS:n ’routeen’ täytyy lisätä samantapainen oikeus sallimaan todentamaton liikenne.

Tilauksesta muodostuva sähköposti lähtee kaikilla muilla tiedoilla paitsi maksun ja arvolisäveron erittelyllä. Toteutan nämä puuttuvat toiminnallisuudet myöhemmin. Testausta hidasti, se että ensimmäisen näyttelytilauksen pystyy tehdä vain ainoastaan kerran yhdellä sähköpostilla. Etuna tässä oli kuitenkin se, että pystyin samalla testaamaan maksamattoman tilauksen automaattista poistamista suorittavaa metodia.

Toisesta tilauksesta ja siitä eteenpäin näyttelyn tilaaminen tapahtuu kirjautuneen ’näyttelyomistaja’-tasoisen käyttäjän kautta. Spring Securityssä piti sallia useita rajapinnan päätepisteitä sallimaan kirjautumattoman käyttäjän tietoliikenne. Näitä päätepisteitä olivat mm. hintojen hakeminen, tilauksen lähettäminen back-endiin sekä maksun jälkeinen tilauksen vahvistamisen sivu.

Keskiviikko 15.03.2017

Näyttelytilauksen tekeminen on toistaiseksi valmis, ainoastaan koodi kaipaa hieman refaktorointia vastaamaan omia standardejani. Olen alkanut käyttämään for-looppien sijasta enemmän Javan 8. version tarjoamia streameja ja lamban-lausekkeen funktionaalisempaa lähestymistapaa. Päivän aikana on tarkoitukseni myös luoda toinen automaattisesti suoritettava metodi, joka seuraa ja tarkistaa näyttelytilauksien tilaa.

Näyttelytilauksen ja ilmoittautumisen maksutapahtumien tilaa tarkistavat metodit ovat nyt erotettu erilleen. Spring Framework:illä on työkalu, joka mahdollistaa metodin

käynnistyvän automaattisesti tietyn väliajoin. Ilmon tapauksessa maksun tilat tarkistetaan 15 minuutin välein, mutta tämä tullaan muuttamaan tiheämmäksi tuotantoon siirtyessä.

Uusi metodi hakee ensin kaikki näyttelytilaukset tietokannasta, joita ei ole vielä maksettu. Tämän jälkeen metodi luo kutsun jokaisesta tilauksesta Maksukaistan rajapintaan tarkoituksena hakea sen maksun tila. Jos maksu on maksettu, tilauksesta lähetetään sähköposti asiakkaalle, ja näyttely- ja käyttäjä luodaan tietokantaan. Jos maksu on yli tunnin vanha, sitä vastaava tilaus poistetaan tietokannasta.

Torstai 16.03.2017

Aloitan järjestelmän erilaisten ”lukkojen” ja estojen luomisen. Näyttelyomistajan rakentaessa näyttelyä on oleellista, että se tapahtuu järjestelmällisesti oikealla tavalla. Tietyt ominaisuudet aukeavat käyttöön vasta kun siitä riippuvat ominaisuudet on käsitelty loppuun. Esimerkiksi aikataulua tapahtumapäivälle ei voida luoda ennen kuin päivään osallistuvat rodut on syötetty järjestelmään tuomareineen. On tärkeää, että tapahtumaa järjestäjä käyttäjä saa tiedon miksi jokin toiminto on toistaiseksi estetty ja mitä hänen tulee tehdä sen avaamiseksi. Osa estoista on jo ohjelmoitu tietokantaan boolean (true/false) tietotyyppinä. Laajempien ominaisuuksien avautuminen vaatii ohjelmakoodin läpikäymään useampia listoja etsien mahdollisia virheitä. Päivän tarkoituksena on kartoittaa käyttötapauksia joissa estoja tarvitaan.

Sain käytyä suurimman osan toiminnallisuuksista läpi selvittäen tarvetta estoille. Tähän asti tarvittavien estojen kokonaismäärä on ainakin kahdenkymmenen luokkaa ja luvun voi odottaa kasvavan. Aloitin estojen ohjelmoimisen samassa järjestyksessä kuin kuvittelen tapahtumaa järjestäjän käyttäjän etenisi. Näistä sain kuitenkin vain muutaman yksinkertaisemman tehtyä ja ilman että käyttäjälle palautuu ilmoitusta toiminnon onnistumisesta tai epäonnistumisesta. Päivän aikana ymmärrykseni käyttäjää rajoittavista toiminnallisuuksista ja niiden tärkeydestä kasvoi.

3.5.1 Viikkoraportti

Viikon teemaksi nousi selvästi tietoturva ja järjestelmän eheys. Useat tämän seurantaviikon aikana tekemistäni ratkaisusta olivat vahvasti yhteydessä näihin osa-alueisiin lukuun ottamatta sähköpostin lähetystä. Tutustuin tarkemmin järjestelmän tietoturvaan ja tallennettavien tietojen validointiin kuluneen viikon aikana. Käyttäjän

järjestelmään tallentava data tapahtuu lomakkeiden kautta. Validoinnin tarkoitus on auttaa käyttäjiä täyttämään lomakkeita oikein, jotta järjestelmä pystyisi prosessoimaan sen onnistuneesti. (Mozilla Foundation 2017)

Opin viikon aikana ymmärtämään syvällisemmin tietoturvaan liittyvistä riskeistä kuten SQL-injektiosta tai käyttäjän istuntoon kohdistuvasta hyökkäyksestä. 2016 oli merkittävä vuosi tietoturvan kannalta. Yhdysvaltojen presidentin vaaleihin liittyi syytöksiä hakkeroinnista, Yahoo:lla tapahtunut tietomurto vuoti 1,5 miljardia salasanaa ja kiristysohjelmien (eng ”ransomware”) tartuttamien tietokoneiden määrä kasvoi. (F-Secure 2017.) On entistä tärkeämpää, että nykyaikainen tietojärjestelmä ottaa huomioon tietoturvatarpeet.

Projektissa datan validointi suoritetaan front- ja back-endissä. Front-end käyttää AngularJS:n omaa validointi mekanismia ja back-endin Spring Frameworkin tarjoamaa ”Bean Validation”-rajapintaa. Molemmissa ratkaisuissa ovat omat hyvät ja huonot puolensa, mutta yhdessä ne muodostavat parhaimman ratkaisun järjestelmän kannalta. Front-endin lomakkeen validointi antaa käyttäjälle välitöntä palautetta sen täyttämisen onnistumisesta, mutta ei tietoturvan kannalta ole paras ratkaisu helpon kierrettävyyden takia. Back-endin validointi on ehdottomasti turvallisempaa, mutta ei pääse samaan käyttäjäystävällisyyteen kuin front-endin validointi. Päätökseni käyttää molempia ratkaisuja on työläämpi toteuttaa, sillä validointien ehdot täytyy ohjelmoida molempiin osa-alueisiin erikseen. Tämä lisää työmäärää ja testauksen tarvetta, mutta parantaa samalla järjestelmän vakautta ja tarjoaa käyttäjälle paremman kokemuksen. (Smashing Magazine, 2009)

Järjestelmän käyttöön ottava kohderyhmä ei alkuselvityksemme mukaan ole tietoteknisiltä taidoilta edistyneintä luokkaa, joten riski tahalliselle väärinkäytölle on pieni. Tämä ei kuitenkaan tarkoita tietoturvallisuuden laiminlyönnistä. Järjestelmän tallentaessa käyttäjien henkilötietoja tulee sen noudattaa henkilötietolakia 7.32 §, jossa rekisterinpitäjää vaaditaan toteuttamaan tarpeelliset tekniset toimenpiteet henkilötietojen suojaamiseksi asiattomalta pääsyltä. (Henkilötietolaki 22.4.1999/523)

Spring Securityn rajapinta toimii järjestelmän tietoturva- ja autentikaatioratkaisuna mahdollistaen käyttäjäroolien mukaisen ”kulunvalvonnan”. Käyttäjän pyytäessä järjestelmältä dataa tai muita resursseja Spring Security varmentaa onko käyttäjä todennettu ja onko hänellä riittävästi valtuuksia. Spring Securityn voidaan jakaa kahteen pääalueeseen jotka ovat todentaminen (eng. ” authentication”) ja valtuuttaminen (”authorization”). Näistä ensimmäisen tarkoitus on todentaa onko pyynnön tekevä käyttäjä

kuka hän väittää olevansa. Valtuuttamista tarvittaessa todentaminen on jo tapahtunut järjestelmässä ja jäljellä on prosessi selvittää omaako todennettu käyttäjä oikeuksia suorittaa toiminnallisuutta tai päästä käsiksi resursseihin. Järjestelmään kirjautuessa käyttäjätunnus ja salasana tarkistetaan tietokannasta ja käyttäjä on näin todennettu. Todentamisen jälkeen järjestelmä hakee tietokannasta käyttäjätilin sisältämät roolit, joiden perusteella eri sivunäkymiin ja resursseihin tarjotaan tai evätään pääsy. (Spring Security Reference, 2015)

JHipsterin yksi tietoturvaratkaisu ja Ilmossa käytössä oleva vaihtoehto on Spring Securityn käyttö JWT:llä eli JSON Web Tokenilla. Tämä ei ole oletuksena Spring Securityn tarjoama ratkaisu vaan ”Java JWT: Json Web Token for Java and Android”-projektista tehty integraatio. (JHipster: Securing your application 2017) JWT on kolmiosainen salakirjoitettu objekti joka koostuu ylätunnisteesta, tietosisällöstä ja allekirjoituksessa. Tässä ratkaisussa käyttäjän tilaa ei ikinä tallenneta palvelimen muistiin vaan jokainen rajapintaan saapuva pyyntö pitää sisällään JWT:n. Yksinkertaisuudessaan järjestelmän palvelinpuoli tietää salauksen purkamiseen käytettävän salausavaimen ja sillä tarkistaa JWT:n pätevyyden määrittäen pääsyn suojattuihin resursseihin. (Auth0 2017).

Spring Securityn käyttö on ollut yleisesti helppoa ja vaivatonta. Suurin osa tarkistuksista tapahtuu yleensä itse kirjaston lähdekoodin puolella, joten logiikan kirjoittaminen on minimaalista. Yksi konfigurointitiedosto pitää sisällään määritykset kaikista rajapinnan osoitteista ja vaadittavista käyttäjärooleista. Esimerkiksi määrittämällä ”api/events/**” ja ”permitAll”-oikeuden kaikki rajapintapyynnöt ”/api/events”-osoitteeseen, sekä sen aliosoitteisiin sallitaan myös todentamattomalle käyttäjälle. Tämän lisäksi käyttämällä annotaatio-ilmaisuja ohjainluokissa voidaan metodiin määrittää vaadittu käyttäjärooli sen suorittamiselle. Esimerkiksi kirjoittamalla ”@Secured(”ROLE_ADMIN”) metodin yläpuolelle aiheuttaa Spring Securityn tekemään valtuuttamisen käyttäjälle, joka metodin haluaa suorittaa, sallien vain ”admin”-tasoisen käyttäjän suorituksen. (Spring Security Reference, 2015)

3.6 Seurantaviikko 6

Maanantai 20.03.2017

Maanantain tarkoitus on mahdollistaa alikäyttäjien lisäämisen tapahtumaan. Tämä tapahtuu näyttelyjärjestäjän hallintapaneelista. Uuden alikäyttäjän lisääminen vaatii

sähköpostin syöttämistä, jolloin uusi käyttäjätili luodaan. Näyttelyjärjestäjän ja alikäyttäjän ero on siinä ettei alikäyttäjä pysty hallitsemaan muita saman näyttelyn alikäyttäjiä eikä ostaa omaa näyttelyä. Aloitan lisäämällä uuden roolin "ROLE_SUBOWNER" tietokantaan ja ohjelmoimalla metodin tämän tason käyttäjän luomiseksi.

Pystyin käyttäjäluontiin käyttämään jo olemassa olevaa metodia. Ohjelmoin myös poikkeustapaukset tilanteille, joissa käyttäjätili on jo olemassa ilman aliomistajan oikeutta (lisätään uusi oikeus käyttäjätilille) tai käyttäjä on olemassa aliomistajan oikeuksilla. Loin uuden sivun hallintapaneeliin, jossa näyttelyjärjestäjälle listataan alikäyttäjät ja jossa niitä voidaan lisätä. Sivulta puuttuu toistaiseksi toiminnallisuus poistamaan alikäyttäjä näyttelystä.

Tiistai 21.03.2017

Maanantai iltapäivällä kävi ilmi, että näyttelyyn ilmoittautumisesta lähtevä maksukuitti ja numerolappu tuleekin lähettää PDF-liitteenä sähköpostissa, eikä HTML-viestinä. Asia on positiivinen, sillä HTML-pohjaisten sähköpostiviestien luominen on erittäin työlästä. Yleisimmät sähköpostipalvelut poistavat HTML-viestien <script>-elementit tietoturvasyistä. Tämä tarkoittaa, että jokaisen elementin tyyli pitäisi kirjoittaa aina uudestaan sen kohdatessa. PDF-tiedoston luontiin on useita kirjastoja, joista valitsen Apachen PDFBoxin. Tarkoitus on tänään tutustua tähän kirjastoon ja luoda pdf-tiedosto ilmoittautumisen kuitista.

Sain työn nopeasti alkuun sillä ilmoittautumista käsittelevä metodi tuottaa jo olion sisältäen kaikki kuittiin tarvittavat tiedot. PDF alkoi muodostumaan myös pikkuhiljaa. PDFBox kirjaston dokumentin luominen tapahtuu yksinkertaisesti määrittelemällä tekstin fontin ja koon jälkeen sen aloituskoordinaatit. Onnistuin myös lisäämään yrityksen logon ja useita kuitin ulkonäköä ehostavia muotoja generoituvaan kuittiin. Kuitti sisältää kaksi listaa, joista ensimmäinen sisältää ilmoittautuneet koirat ja toinen ilmoittautuneet junior handlerit. Päivän haastavin asia oli määrittää, missä tämän toisen listan ensimmäinen jäsen näkyy sen koordinaattien ollessa riippuvaisia ensimmäisen listan pituudesta. Päivän päätteeksi olin tyytyväinen kuitin ulkonäköön sekä rakenteeseen ja opin perusteet PDF:n luomisesta Javalla.

Keskiviikko 22.03.2017

PDF-dokumentteja täytyy luoda vielä kaksi lisää, yksi numerolappuun ja toinen näyttelytilauksen kuittiin. Tarkoituksena on tänään aloittaa numerolapun tekemistä, joka vaatii enemmän graafista suunnittelua kuin eilinen ilmoittautumisen kuitti. Numerolapun pitää sisältää myös QR- ja viivakoodi. Työn pitäisi sujua nopeammin, sillä voin hyödyntää ilmoittautumislapun luomisessa käytettyä lähdekoodia. En kuitenkaan varmaankaan saa numerolappua tänään täysin valmiiksi, koska työpaikalleni on tulossa haastateltavaksi graafikko aamupäivällä ja minun odotetaan esittelevän järjestelmää hänelle.

Sain PDF-dokumentin hyvälle mallille ja melkein valmiiksi. QR-koodin generoimiseen käytin projektin ulkoista kirjastoa, jonka asennuksen jälkeen sen käyttäminen oli varsin yksinkertaista. QR-koodi generoituu .png tiedostoksi liitettäväksi dokumenttiin. Haasteeksi nousi näiden tiedostojen tallennuksen sijainti sekä tietojen säilytettävyys. Jokaisesta ilmoittautumisesta järjestelmä luo näyttelyn alkuun mennessä kuvineen ainakin neljä eri tiedostoa jotka täytyy tallentaa palvelimelle. En ehtinyt tähän ongelmaan pureutua syvemmin, sillä demotilaisuus graafikon kanssa venähti pitkälle iltapäivään. Oma esitykseni sujui mielestäni paremmin kuin edellisellä kerralla. Onnistuin kertomaan järjestelmän eri osa-alueista ja sivustorakenteesta selkeämmin ja perusteellisemmin.

Torstai 23.03.2017

Tänään tarkoituksena on saada numerolapun PDF-tiedosto valmiiksi ja aloittaa näyttelytilauksen kuitin luomista. Luin illalla PDFBox kirjastosta lisää ja huomasin käyttäväni dokumentin mittayksiköissä Yhdysvaltalaista yksikköjärjestelmää jossa yhden koordinaatti pisteen koko on 1/72 tuumaa. Havainnon seuraksena en aio muuttaa jo tehtyä ilmoittautumisen kuittia. Ohjelmoin kuitenkin metodin numeronlapun luomisen lähdekoodiin, joka muuntaa tuumat millimetreiksi. Muuntaminen auttaa hahmottamaan muotojen ja viivojen piirtämistä paremmin PDFBox-kirjastoa käytettäessä.

Sain numerolapun valmiiksi, mutta sain selville kriittisen tiedon sen rakenteesta. Numerolapun rakenteessa yhden irti leikattavan osan täytyy olla standardikokoinen, joka minun toteutuksessani uupuu. Muuttaminen vaatii lisätyötä, sillä muita osia täytyy pienentää jotta saan tämän tärkeän osan oikean kokoiseksi. Lähdekoodini tämän dokumentin luomisessa oli jo selvästi laadukkaampaa kuin ilmoittautumisen kuitin luomisessa. Lähes staattinen dokumentin sisällön rakenne myös helpotti elementtien

asettelua. Päivän lopussa selvitin myös kuinka kansioita luodaan palvelimelle Javalla ja tein alustavan hahmotelman.

Perjantai 24.03.2017

Torstaina päätin että teenkin vaihtelun vuoksi perjantaina töitä kotoa käsin, kun siihen kerta on minulla mahdollisuus. Perjantain tarkoitus on ohjelmoida kansioiden luominen PDF-dokumenttien tallennusta varten sekä ohjelmoida näyttelytilauksen pohjalta kuitti asiakkaalle lähetettäväksi. Jokaiselle näyttelylle luodaan oma kansio palvelimelle ja jokainen ilmoittautumistapahtuman tiedostot tallennetaan omaan kansioon sen näyttelyn sisälle. Lähdekoodin kannata kansioiden luominen on varsin suoraviivaista, kansio luodaan jos sitä ei ole. Tämä täytyy kuitenkin testata huolella sillä kaikki yhden näyttelyn numerolaput luodaan kerralla.

Kansioiden polkuja luodessa tajusin, ettei tuotantopalvelimen kansiorakenne vastaa tietenkään oman kehitysympäristöni kansiopolkua. Tästä syntyi myös tarve tallennussijainnin päättämiselle esimerkiksi järjestelmän asentamisesta uudelle palvelimelle. Konfiguroin polkujen sijainnit määritystiedostoihin, joista kerron enemmän tämän viikon analysoimisessa. Ensimmäisen tason kansio luodaan onnistuneen näyttelytilauksen seurauksena, jonka jälkeen toisen tason kansiot aina onnistuneen ilmoittautumisen jälkeen. Tiedostomäärän kasvaessa on tärkeää seurata palvelimen kiintolevy tilaa ja tarpeen tullessa poistaa menneiden näyttelyiden dokumenttia tai siirtää toiselle palvelimme. Tänään opin konfiguroimaan applikaatiota paremmin eri tarpeisiin sekä luomaan dynaamisesti kansioita palvelimelle.

3.6.1 Viikkoraportti

Koko viikko kului maanantaita lukuun ottamatta PDF-tiedostojen luomisen parissa. Niiden luominen oli työlästä ja pikkutarkkaa hommaa, kuten Java ohjelmointi yleensä onkin. Selvittelin erilaisia vaihtoehtoja PDF-generaatio kirjastoille, joista maksullinen ”iText” oli yleisin. iTextillä oli selvästi monipuolisin dokumentaatio ja käyttäjien luomia esimerkkejä löytyi paljon. Maksullisen lisenssin takia päädyin valitsemaan ja opettelemaan ilmaista Apachen PDFBox-nimistä avoimen lähdekoodin pdf-dokumentin luomisen tarkoitettua kirjastoa.

Yksi vaikeimmista haasteistani työpaikallani on antaa aika-arviointa ominaisuuksien ja toiminnallisuuksien työmäärästä. Yleensä toiminnallisuudet pitävät sisällään

tiedostamattomia käyttötapauksia. jotka voidaan vasta havaita toteutushetkellä. Viikon tehtävänä oli luoda PDF-tiedostoja kuiteista, joita asiakas tuottaa eri toiminnoilla. Tehtävän voi pilkkoa useaksi palaseksi: kirjaston valitseminen, kuitin ulkoasun suunnitteleminen, kuitin sisältämien tietojen muotoilu Java-olioiksi ja olion sisältämien tietojen liittäminen kuittiin.

Tehtävän määrittelyn ulkopuolelta kuitenkin unohtui kuitin välitysmekanismi eli sähköposti, jota varten PDF-tiedostot tulee tallentaa kovalevylle muistiin. Kaikki ilmoittautumisten numerolaput lähetetään samanaikaisesti, joten niiden tulee löytyä jonkun tiedostopolun päästä ja mieluiten loogisesti. Jo nyt voidaan huomata taskin koko on kasvanut alkuperäistä suuremmaksi ja aikatauluun voidaan odottaa venymistä. Kyseisessä tapauksessa oli vielä huomioitava se, että kehitysympäristöni tiedostopolku eroaa tuotantopalvelimen polusta. Tulevaisuuden ja mahdollisten uusien kehittäjien kannalta ratkaisin ongelman sitomalla polun osoitteen konfiguraatio-tiedostoihin. Tiedostopolun osoite vaihtuu sen mukaan aktiiviseksi suoritetaanko ohjelmaa palvelimella vai kehitysympäristössä.

Suurin osa järjestelmään toteuttamistani toiminnallisuuksista tai kokonaisuuksista ovat olleet jo käsitetasolla minulle täysin uusia. Antamani aikatauluarvioit ovat jäävät usein liian lyhyiksi ollessani vielä tietämätön uuden teknologian käytöstä tai määrittelyn ollessa puutteellista. Hyvän määrittelyn tärkeys on korostunut tälläkin viikolla, sillä yhden dokumentin rakennetta täytyy vielä muuttaa merkittävämmin ja yhden kokonaan uuden dokumentin luominen täytyy tehdä.

Aikataulusta viivästyminen on yleistä ohjelmistokehityksessä. Muuttuvien taskien, työtehtävien ja uusien asioiden saralla ajallisten kestojen arvioiminen voi olla vaikeaa. Kehitystyö on luonteeltaan insinöörimäistä, samoja asioita harvoin tehdään uudestaan ja uudestaan. Tämän takia tarkkojen aikamääreiden antaminen on lähes mahdotonta vaan arvioita pitäisi ajatella suuntaa antavaksi. On tietenkin mahdollista arvioida kestääkö jonkin ominaisuuden luominen päiviä tai viikkoja. Suurimmat viivästykset tulevat joko määrittelyn puutteesta tai vuorovaikutuksesta muun koodin kanssa. Jos uuden ominaisuuden implementoiminen tarkoittaa vanhan ominaisuuden laajentamista, työhön tulee sisältymään vanhan koodin uudelleenkirjoittamista sekä testausta.

Kireät aikataulut saattavat aiheuttaa myös oikoteiden tekemistä. Arkkitehtuurillisesti virheellinen ratkaisu voi kehityksen kannalta olla nopein, kun tarvittavaan resurssiin mennäänkin olemassa olevan logiikan sijaan suoraan oikotietä pitkin. Tämä lopulta johtaa

esimerkiksi back-endin service-tasolla siihen, että jokaiseen serviceen injektoidaan jokainen muu service. Huonosti tehdyt kokonaisuudet kasvattavat järjestlemän teknistä velkaa aiheuttaen tulevaisuuden ominaisuuksien kehityksen hidastumista. (Flower 2013) Kapselointi eli yhteenkuuluvien osien niputtaminen yhteen vaurioituu tässä. Ilmon tapauksessa kapselointi tapahtuu kolmitasoarkkitehtuurin muodossa, jossa yhteen domain-tason olioon kuuluu yksi service ja yksi controller back- ja front-endissä.

3.7 Seurantaviikko 7

Maanantai 03.04.2017

Aloitan tänään työni poikkeuksellisesti myöhemmin sillä seurantaviikolla 6. maintsemani graafikko tulee toimistolle tänään Tampereelta. Päivän agendana on käydä hänen kanssa järjestelmäni front-endiä tarkemmin läpi ja selvittää parannustarpeita. Tulemme myös miettimään yhdessä sivustolle visuaalista yleisilmettä ja helppokäyttöisempää käyttöliittymää. Tavoitteena on käydä kaikki järjestelmän sivut läpi tarkemmin, selvittää tarpeet sekä suunnitella kuinka teemme töitä jatkossa yhdessä.

Kävimme ja suunnittelimme järjestelmää ulkoasua yli puolet työpäivästä. Koen että teimme merkittävää edistystä ja sain hyviä vinkkejä käyttöliittymäsuunnittelua varten. Korjattavaa sivuilta löytyi ainakin vähän jokaiselta, mutta en vielä tärkeämmiltä töiltäni niitä ehdin vielä työstämään. Oli myös vaihtelua päästä ensimmäistä kertaa projektin aikana työskentelemään ja keskustelemaan teknisistä asioista toisen työntekijän kanssa. Saimme sovittua myös jatkotyöskentely tavat ja käytännöt.

Tiistai 04.04.2017

Eilisen käydyn pitkän palaverin seuraksena yksi parannuskohde on kehittää ”virheilmoitus-logia” (eng. ”error log”) eteenpäin näyttelyjärjestäjälle. Tällä hetkellä näyttelyluomisen puutteista ja vaiheista kertova logi näkyy ainoastaan hallintapaneelin pääsivulla, mutta se halutaan lisätä useimmille hallintapaneelin alasivuille. Alasivuilla login halutaan pitävän sisällään ainoastana sille sivulle littyviä virheilmoituksia. Päivän tehtävänä on kasvattaa erilaisten virheilmoitusten määrää, refaktoroida niitä käsittelevä metodi useammaksi pienemmäksi palaseksi ja toteuttaa sivukohtainen pyyntö logille.

Päivän aikana sain hajoitettua virheilmoitus-login luovan metodin useammaksi pienemmäksi metodiksi, joista kukin testaa vain omalle kohdealueelleen kuuluvia ominaisuuksia. Muutin myös virhelogin jäsenten koostumusta vastaamaan muotoa, jossa ne voidaan helposti kääntää back-endissä käyttäjän omalle kiellelle. Sivu pyytää lähettämällä parametrin back-endille, joka palauttaa yhden tai useamman kohdealueen login riippuen pyynnöstä. Virheilmoitus-logia oli erittäin mieluista ohjelmoida, sillä pääsin vihdoin testaamaan erilaisia harvinaisia käyttötapauksia ja onnistuin tekemään sen lähdekoodista helposti jatkokehitettävää.

Keskiviikko 04.04.2017

Keskiviikkona parannuskohteena on kehätoimitsijoiden käyttäjätunnusten luonnin parantaminen ja niiden tuominen näyttelyjärjestäjän hallintapaneliin. Tällä hetkellä käyttäjätunnukset luodaan samalla kun näyttelyn kehät voidaan generoida. Käyttäjätunnuksen muoto on ennen pilottia vielä ”demo-1-3”, jossa ensimmäinen numeron on tapahtumapäivän tunnus ja toinen kehän numero. Ongelmaksi muodostuu se että näyttelyjärjestäjä voi muuttaa kehän numeroa mielensä mukaan ja näin ollen kehän numero ei enää vastaakkaan käyttäjätunnuksen numeroa. Myös toiminnallisuus jolla kehätoimitsijoiden käyttäjätunnukset saisi näkyviin näyttelyn hallintapaneeliin puuttuu vielä täysin. Tarkoituksena on irrottaa käyttäjätunnusten luominen kehien generoimisen yhteydessä ja toteuttaa se vasta kun näyttelyjärjesteäjä on ”hyväksynyt” aikataulun lopulliseksi.

Näyttelyjärjestäjä voi uuden toiminnallisuuden myötä lisätä käyttäjätunnukset kun jokaiselle kehälle on annettu numero. Näillä käyttäjätunnuksilla ei ole aitoa sähköpostia, joten ohjelmoin salasanan tallentumaan käyttäjä-taulun ”recovery_key”-kolumniin, jota käytetään salasanan palauttamisessa. Tällä pienellä ”hackillä” käyttäjätunnukset voidaan tuoda salasanoineen näyttelyjärjestäjän hallintapaneelista avattavaan ikkunaan.

Torstai 05.04.2017

Työpäivä jää tänään lyhyeksi ollessa sairas. Tavoitteena kuitenkin saada näyttelyjärjestäjän hallintapaneelin ilmoittautumisen seurannan alasivun front-end osat tehtyä. Alasivuilla näyttelyjärjestäjä voi seurata rotukohtaisia ja tuomarikohtaisia ilmoittautumismääriä. Varsinaisia toiminnallisuuksia sivuilla ei ole datan listaamisen lisäksi

ja olen jo aikaisemmin ohjelmoinut back-endillä vastaavan toiminnallisuuden. Data tulee back-endiltä JSON-muodossa kuten kaikilla muillakin sivuilla.

Rotukohtaisen ilmoittautumismäärän JSON-objekti koostui useista sisäkkäisistä listoista (tapahtumapäivä, rotu, kilpailuluokat, ilmoittautuneet koirat), joita looppasin HTML-taulujen sisälle. Tauluista pystyy myös etsimään yksittäisen rodun ilmoittautumismääriä. Tuomareiden ilmoittautumismäärien data oli selvästi yksinkertaisempi sisältäen vain yhden sisäkkäisen listan. Olin tyytyväinen että sain nopeasti iltapäivän aikaan hoidettua nämä kaksi roikkumaan jäänyttä tehtävää.

3.7.1 Viikkoraportti

Viikon työt etenivät hyvään tahtiin, sain tärkeitä ominaisuuksia valmiiksi ja järjestelmä on jälleen lähempänä tuotantoon siirtymistä. Uusi tavoitteemme on saada siitä myyntikuntoinen versio touko-kesäkuun vaihteessa ja uskon pääseväni tähän tavoitteeseen. Graafikkon vierailu toimistolla toi uutta energiaa ja motivaatiota työntekoon. Viikon teemaksi nousi selvästi käyttöliittymän parantaminen ja käyttäjäystävällisyyteen panostaminen. Ajoittain on hankalaa katsoa luomaani järjestelmää ”uusilla silmillä”, sillä elementtien sijoittelu ja eri nappien toiminnot ovat minulle täysin tuttuja. Käyttöliittymän ollessa yksi järjestelmän tärkeimmistä kulmakivistä ja on oleellista että ennen julkaisua sitä pitäisi testata uusilla käyttäjillä.

Käyttäjän ohjaaminen näyttelyä luodessa on erittäin kriittistä, toimintoja on paljon ja useimmat niistä täytyy suorittaa tietyssä järjestyksessä. Esimieheni mukaan järjestelmää käyttävä kohderyhmä ei ole tietoteknisiltä taidoiltaan edistyksellinen. Tämän myötä käyttöliittymän selkeys ja yksinkertaisuuden tärkeys korostuvat. (Paunovic 2012) Tällä viikolla luomani virheilmoitus-logi ja toimintojen estäminen ovat vain yksi osa käyttäjän johdattamista. Tapahtumajärjestäjän hallintapaneelin pääsivulle tulen ohjelmoimaan useiden verkkokauppojen tilaussivuilla näkyvän tilauksen edistymistä kuvaavan janan. Jana kertoo käyttäjälle näyttelyn rakentamisen vaiheet ja mitä käyttäjän tulee tehdä päästäkseen siirtymään seuraavaan vaiheeseen. Muita sivuille suunnitteilla olevia käyttäjän johdattamiseen liittyviä ominaisuuksia ovat pienet ohjelaatikot sekä ”usein kysytyt kysymykset”-osio. Tekemäni virhelogi koostuu viesteistä kuten ”lauantaille ei ole lisätty rotuja” tai ”näyttelyn sijainti puuttuu”, antaen käyttäjälle tietoa siitä mitä hänen tulisi tehdä. Estot ja ”lukot” ovat vielä kehitysasteella. Ne toimivat mutta eivät anna palautetta tai ilmoita miksi jotain toimintoa ei suoritettu. Oppimiseni kannalta olisi hyödyllistä perehtyä eri teoksiin käyttöliittymäsuunnittelusta.

Front-endin näkymien rakentamiseen hyödynnän laajasti käytettyä Twitterin Bootstrap nimistä viitekehystä. Kyseinen viitekehitys on GitHub-verkkosivun eniten tähtiä saanut viitekehys ja toisella sijalla eniten tähditetyistä ohjelmistovarastoista (GitHub 2017). Bootstrapin etuja on nopea kehitettävyys sekä helposti implementoitava responsiivinen rakenne erikokoisille näyttöresoluutioille. Se kuului valmiiksi JHipsterin teknologiapinoon ja entuudestaan tuttuna, minun oli helppo ottaa se käyttöön. Puutteellisten front-end taitojeni takia sivusto näyttää hyvin laatikkomaiselta ja Bootstrapillä tehtyjen esimerkkien kaltaiselta. Tätä viitekehystä on kuitenkin helppo räätälöidä paremman näköiseksi sen jälkeen kun yleisilme on suunniteltu.

Viikon aikana jouduin selvittämään yhdessä graafikon kanssa kuinka kehitämme järjestelmää yhdessä. Front-endin ja back-endin yhdistyessä yhdeksi paketiksi projektin rakentuessa tarkoittaa, että front-endiä kehittääkseen myös back-endin tulee olla käynnistettynä samalla tietokoneella. Tämä kasvattaa kehitysympäristön asennuksessa ja suorittamisessa tarvittavien kirjastojen ja kehittäjäpakettien määrää. Päädyimme ratkaisuun, jossa graafikko ensin piirtää rautalankamallit käyttäen Bootstrapin ”grid”-ominaisuutta. Niiden ollessa valmiita, implementoin muutokset sivustolle ja graafikko alkaa luoda tyyliä ja sivuston identiteettiä oman kopioon rakenteesta.

3.8 Seurantaviikko 8

Maanantai 10.4.2017

Aloitan viikon tekemällä hallintapaneelin Ilmon työntekijälle. Hallintapaneelin tarkoitus on tuoda yritykseni työntekijälle mahdollisuuden tarkastella tilauksia näyttelyistä. En tässä vaiheessa kehitä ylimääräisiä toimintoja kriittisten ominaisuuksien lisäksi, sillä tämän osa-alueen prioriteetti ei ole korkealla. Vaihtoehtona tämän käyttöliittymän sijaan olisi tehdä hakuja tietokantaan manuaalisesti, mikä ei kuitenkaan ole käyttäjäystävällisin tai nopein tapa ratkaista tämä ongelma. Hallintapaneliin tulee kuitenkin sisällyttää yksi erittäin kriittinen ominaisuus, mahdollisuus syöttää näyttelytilaukseen Maksukaistan asiakaskohtaiset rajapinta- ja salausavaimet. Näitä avaimia käytetään autentikoimaan ilmoittautumisen maksut ja ohjaamaan ne näyttelyjärjestäjän tilille.

Loin työntekijän hallintapaneelin, mutta toistaiseksi siihen pääsee käsiksi vain ”admin”-tasoinen käyttäjä. Ohjelmoin hallintapaneelille oman sivun, joka tekee pyynnön back-endin hakien tietokannasta listan näyttelytilauksista. Näyttelytilaukset listataan

järjestykseen aloittaen tuoreimmasta. Listasta on mahdollisuus siirtyä kyseisen tapahtuman hallintapaneeliin (sama näkymä kuin näyttelyjärjestäjällä), näyttelyn kotisivulle sekä tarkastella näyttelytilausta tarkemmin. Tein jokaisen listan jäsenen kohdalle napin joka avaa uuteen ikkunaan lomakkeen Maksukaistan avainten syöttämistä varten.

Jatkossa on tärkeää myös erottaa rooleja eritasoisille työntekijöille, siten ettei jokaisella hallintaoikeuksia omaavalla käyttäjällä ole pääsyä kriittisiin ominaisuuksiin vaan ne jäävät adminille. Pääsin maanantaina hyvin päivän tavoitteisiin ja tein havainnon eritasoisista työpaikan sisäisistä järjestelmän käyttäjärooleista.

Tiistai 11.04.2017

Aamupäivän tarkoitus on kehittää työntekijän hallintapaneelia pidemmälle luomalla sivu yhdelle näyttelytilaukselle. Tämä sivu näyttää kaikki tiedot tilauksesta, asiakkaasta sekä tuo viime viikolla tekemäni näyttelyn virhelogin myös työntekijän katseltavaksi. Sivulla ei varsinaisesti ole muita toiminnallisuuksia toistaiseksi staattisen datan esittämisen lisäksi. Päivemmällä pidän myös demon esimiehilleni, jossa näytän heille mitä olen edellisen demotilaisuuden jälkeen saanut aikaan. Iltapäivällä parantelen työntekijän hallintapaneelin sivuja jos aikaa jää.

Sain näyttelytilauksen sivun ohjelmoitua front-endiin ennen demon pitämistä. Olen ohjelmoinut back-endin lähettämään dataa DTO (eng. ”Data Transfer Object”) muodossa vähentäen datan määrää. Ilmon tapauksessa objektin aliluokat koostuvat ainoastaan niiden pääavaimista. Jos alaluokkia tarvitaan, ne voidaan hakea erillisellä haulla. Näyttelytilauksen haku on kuitenkin poikkeuksellinen, sillä front-endissä halutaan esittää kaikki sen aliluokkien tiedon, joten ohjelmoin back-endin restcontrollerin palauttamaan koko tilauksen sen DTO:n sijaan.

Päivän demotilaisuus sujui mielestäni hyvin, näytin esimiehilleni uuden työntekijän hallintapaneelin sekä viimeviikkoisen kehätoimitsijoiden käyttäjätunnusten toiminnallisuudet. Päädyimme siirtämään tunnuslukujen luomisen ”Tulosjärjestelmä”:n alle, sillä kehätoimitsijoiden tunnusluvut eivät varsinaisesti kuulu kehien ja niiden aikataulujen luomiseen.

Keskiviikko 12.04.2017

Keskiviikon tarkoitus on kehittää kehätoimitsijoiden seurannan yleisnäkymää. Näyttelyjärjestäjällä on mahdollisuus seurata näyttelypäivinä kuinka arvostelu kehäkohtaisesti etenee ja tarvittaessa siirtyä itse suorittamaan arvosteluja esimerkiksi häiriötilanteissa. Kyseinen sivu on jo luotu aikaisemmin ja tällä hetkellä se pyytää back-endiltä tapahtumapäivän kehiä, arvosteltujen koirien määrää ja rodun koko koiramäärää. Suunnittelen myös aamupäivä mitä toimintoja tälle sivulle olisi hyvä kehittää.

Onnistuin kehittämään kehien yleisseurantaa kohtalaisesti eteenpäin. Toistaiseksi tapahtumajärjestäjällä on ainoastaan mahdollisuus siirtyä kerrallaan yhden rodun arvostelunäkymään, sillä olen aikaisemmin ohjelmoinut järjestelmän hakemaan kehän yleisnäkymän kehätoimitsijan käyttäjätunnuksen mukaan. Tässä näkymässä front-endillä jokaisella kehällä on oma Boostrapin ”panel”-elementti jonka sisään looppaan kehän rodut. Ohjelmoin luupin muuttamaan rodun rivin taustan värin vihreäksi, mikäli rotu on arvosteltu onnistuneesti loppuun sekä keltaiseksi jos rotu on arvostelussa tällä hetkellä. Sivu vaatii vielä jonkin verran kehitettävää, jonka suoritan kun muut korkea prioriteettiset tehtävät on tehty.

Torstai 13.04.2017

Toimiston ollessa tyhjillään tänään päätin tehdä töitä etänä kotoa tänään pääsiäisloman alkaessa huomenna. Mitään suurempaa toiminnallisuutta en asettanut päivän tehtäväksi vaan tarkistan kaikki parin viime viikon aikaiset työt ja parantelen omaa lähdekoodiani. Tämän jälkeen rakennan projektin ja asennan sen pyörimään palvelimelle, jossa suoritan testaan näyttelyn tilauksen, työntekijän hallintapaneelin sekä näyttelyjärjestäjän kehäseurannan osa-alueet.

Kirjoittamani lähdekoodini ei vaatinut paljoa korjailtavaa, poistin muutaman ylimääräisen ja turhan tietokantahaun. Kaikki toiminnallisuudet toimivat hyvin myös tuotantoa vastaavassa versiossa palvelimella. Olin myös unohtanut testata kuinka PDF-dokumentit tallentuvat palvelimelle käyttäen ”production”-ympäristöprofiilia. Järjestelmä loi ensimmäisen tason kansion onnistuneesti, mutta ei onnistunut luomaan toisen ja kolmannen tason kansioita. En vielä onnistunut selvittämään bugin syytä iltapäivän aikana, mutta en usko sen olevan haastava ongelma ratkaistavaksi. Koen etänä työskentelyn olevan hyvä vaihtoehtona tällaisille työpäiville, joiden aikana en kehitä mitään uutta ja näin ollen tarvitse esimieheni välitöntä palautetta.

3.8.1 Viikkoraportti

Viikko piti sisällään paljon käyttöliittymän suunnittelua ja toteutusta, joka ei ole vahvinta osaamisen aluettani. Käytin aikaa lukiessani eri lähteiden käsityksiä käyttöliittymän parhaista käytännöistä. Eri yritykset ovat luoneet käytäntöjä ja ohjeita käyttöliittymän suunnittelulle. Näitä ovat esimerkiksi Googlen ”Material Design” ja Applen ”iOS Human Interface Guidelines”. Käyttöliittymän on järjestelmän osa, jonka kanssa käyttäjä on suoraan tekemisissä ja sen tarkoitus on varmistaa toimintojen sulavuus käyttäjälle keskittyen heidän tarpeisiinsa. (Kemppainen 2014)

Front-endin viitekehyksen, Bootstrapin, ansiosta yksinkertaisten käyttöliittymien luominen on ollut helppoa. Laatikkomaisten elementtien kanssa on suoraviivaista suunnitella ja toteuttaa alkeellisia versioita etenkin datan esittämiselle. Edistyneemmät toiminnot vaativat jo parempaa JavaScriptin taitamista. Ennen projektin alkamista taitoni front-endin kehityksessä olivat todella alkeelliset. Kuukausien kuluessa opin AngularJS:n ekosysteemin hyvin ja käyttöliittymän kehitys alkoi sujua paremmin. Suurin ongelmani oli kuitenkin et etten osannut muokata elementtejä ns. ”lennosta”. Jokaisen tiedon tallennus tai muokkaus operaation jälkeen määritin ohjelma suoritti sivulatauksen, jotta uusi data saataisiin näkyvin. Tämä ei ikinä olisi toiminut tuotannossa käyttöliittymän tahmaisuuden ja sivulatauksen aiheuttaman kuorman takia. Käyttöliittymän kehittäminen saavutti uuden ulottuvuuden kun sain vihdoin ymmärsin kuinka AngularJS:llä voidaan lähettää ”eventtejä” ”kuuntelijoille”. Näiden tapahtumien avulla voidaan esimerkiksi ponnahdusikkunasta suoritetun tiedon tallentamisen jälkeen lähettää tallennettu datan takaisin sivulle ilman sen lataamista.

Käytettävyys on yksi järjestelmän kulmakivi ja viikon aikana toteutin käyttöliittymiä yritykseni työntekijälle sekä kehätoimitsija-tasoiselle käyttäjälle. Etenkin jälkimmäinen, asiakkaan käyttämä käyttöliittymän suunnittelu, on erityisen huolella suunniteltava kokonaisuus. Kehätoimitsijan käyttöliittymään päästään käsiksi muun sivuston takaa URL-osoitteesta, mutta sen pääasiallinen näyttöpääte on tabletti. Käyttöliittymä sisältää paljon toiminnallisuuksia ruudunkokoon suhteutettuna ja ne tulee asetella loogisesti. ”Selkeästi esitetyt tiedot ja looginen käyttöjärjestys vähentävät käyttäjän epävarmuutta, nopeuttavat päätöksentekoa ja ohjelmiston käyttöä sekä minimoivat mahdolliset virheet.” (Kemppainen 2014.) Suunnittelussa pyrin asettumaan järjestelmää käyttävän käyttäjän rooliin ja miettiä mitä haluaisin näyttöpäätteellä nähdä ja missä sen olisi hyvä sijaita.

Sivustolla on kaksi navigaatio elementtiä käytössä tällä hetkellä. Ensimmäinen on perinteinen sivuston yläreunaan kiinnitetty ylänavigaatio ja toinen on näyttelyjärjestäjän hallintapaneelissa siirtymiseen käytettävä sivupalkki. Olen tehnyt kaikkien lomakkeiden rakenteesta ja painikkeiden sijainneista yhteneviä keskenään. Sivuston laajuinen yhteinen ulkonäkö auttaa merkittävästi käyttäjää navigoimaan sekä käyttämään palvelua. Usability.gov-verkkosivusto listaakin konsistenssin yhdeksi käyttöliittymäsuunnittelun parhaaksi käytännöksi. Käyttäjän oppiessa tekemään jotain sivulla, hänen tulisi pystyä siirtämään ja käyttämään kyseistä taitoa myös muualla sivustolla. (Usability.gov 2017)

Työntekijän käyttöliittymä ei ole prioriteetissa läheskään yhtä korkealla. Julkaisun riittää, että sieltä löytyvät kriittisimmät toiminnallisuudet järjestelmän toimivuuden kannalta. Ratkaisuni pitää sisällään kaksi eri sivua, toiselle listataan kaikki näyttelytilaukset sekä muutama perusominaisuus ja toinen sivu keskittyy yhden näyttelytilauksen käsittelyyn. Käyttöliittymä tulee jatkokehityksessä laajentumaan todennäköisesti. Organisaation mahdollisen kasvun kautta ei ole turvallista, että jokainen yrityksen järjestelmää käyttävällä henkilöllä on pääsy järjestelmän kriittisiin toiminnallisuuksiin. Käyttöliittymä tullaan todennäköisesti toteuttamaan silloin uusien roolien perusteella, jolloin esimerkiksi myyntiin keskittyvät työntekijät näkevät käyttöliittymästä vain heille tarkoitettuja resursseja.

3.9 Seurantaviikko 9

Tiistai 18.04.2017

Päivän tavoitteena on jatkaa kehätoimitsijan käyttöliittymän parantelemista sekä siihen liittyvien toiminnallisuuksien toteutusta. Näyttelyjärjestäjälle generoidaan automaattisesti kehäkohtaiset käyttäjätunnukset, kun hän on saanut aikataulun tehtyä hyväksytysti. Näyttelypäivänä nämä tunnukset jaetaan kehätoimitsijoille paikan päällä, joiden kautta he voivat kirjautua järjestelmään ja aloittaa tulosten syöttämisen ilmoittautuneille koirille. On tärkeää kuitenkin tietää, kuka missäkin kehässä on tuloksia syöttämässä joten kehätoimitsijan kirjautuessa ensimmäistä kertaa tunnuksille ”ui-modal”-ponnahdusikkuna pyytää häntä syöttämään nimensä. Voin ratkaisussa käyttää "user"-luokkaan liittyvää jo olemassa olevaa "userinfo"-aliluokkaa joka sisältää käyttäjän tarkempia tietoja.

Päivän aikana tämän tehtävän käyttötapaukset selkenivät tarkemmin saadessani tietoon, että kehien toimitsijat saattavat vaihtua päivän aikana, etenkin isoissa näyttelyissä. Tämän seurauksena muutin käyttäjän syöttämän nimen näkymään kokojaan sivuston yläreunassa ja sitä painamalla nimeä pystyi muuttamaan. Olin valmiiksi ohjelmoinut

jokaisen uuden arvostelun hakemaan aina kyseisen kehän toimisijan nimen tietokannasta joten suurempia muutoksia ei tässä tarvittu.

Keskiviikko 19.04.2017

Jatkan tänään kehätoimitsijan arvosteluihin liittyviä toiminnallisuuksia. Kirjoitin arvosteluihin liittyvän logiikan talven alla ja muistin virkistämiseksi käyn sen läpi tänään huomattuani ainakin yhden puutteen. Arvostellessa on tärkeää edetä johdonmukaisessa järjestyksessä, sillä eri "kilpailuista" parhaimmat siirtyvät aina ylemmän tason kilpailuihin. Esimerkiksi rodun parhaan koiran valintaa varten tarvitaan että kaikki urokset ja naaraat on arvosteltu. Arvostelun syöttäessä järjestelmä tarkistaa onko kaikki kyseiseen kilpailuun osallistuvat koirat jo arvosteltu. Jos kaikki kyseiset koirat on arvosteltu siltä kilpailun tasolta, avaa järjestelmä seuraavan kilpailun arvostelumahdollisuuden. Ongelmakohtana huomasin että jos johonkin kilpailuun ei ole yhtään osallistujaa niin järjestelmä ei tunnista sitä vielä ja näin ollen seuraavan tason kilpailuun ei voi siirtyä ollenkaan. Haaste vaatii logiikan korjausta back-endissä ja front-endin puolelle tarvitaan tyhjän kilpailun kohdalle ainakin ilmoitus, että kyseisessä kilpailussa ei ollut yhtään osallistujaa. Tärkeintä on kuitenkin ettei arvosteleminen estyisi missään tilanteessa logiikkavirheen vuoksi, joten kilpailujen lukotus tulee olla enemmän löysää kuin erittäin tiukkaa.

Sain ongelman korjattua ja arvostelun pitäisi nyt olla sulavampaa. Tein ehdot tarkistamaan osallistuvien koirien määrän ja merkitsemään kilpailun käydyksi, jos koirien kokonaismäärä on sama kuin arvosteltujen koirien määrä. Muuten arvosteluita käsittelevä service-luokka on ehkä back-endin heikointa koodia sisältämä osuus. Talvella ohjelmoin kyseisen palikan kiireen alla, joka näkyy saman koodipätkien toistuvuudessa sekä optimoinnin vähäisyydessä. Huonon metodien nimeämisen seurauksena oli aikaa vievää selvittää mitä eri toiminnallisuudet tekivätkään. Dokumentoitu lähdekoodi helpotti logiikan ymmärtämistä. Tämä kyseinen järjestelmän osa tarvitsee joskus laajemman uudelleenkirjoituksen, mutta tällä hetkellä siihen ei ole aikaa.

Torstai 20.04.2017

Jätän arvosteluihin liittyvät hommat hetkeksi taka-alalle, sillä sain graafikoltamme rautalankamalleja eri sivuille. Lähes kaikki sivut vaativat enemmän tai vähemmän muutosta. Työtäni muuttaa näitä sivuja helpottaa Bootstrapin ominaisuuden, ”gridin”, käyttäminen. Kyseinen front-endin viitekehyksen yksi ominaisuus jakaa ruudun

leveyssuunnassa 12 eri kolumniin. Nämä kolumnit määrittävät elementtien leveyden responsiiviseksi erikoisille näytöille. Esimerkiksi HTML-taulukko voidaan asettaa viemään tietokoneen näytöstä puolet sen koko leveydestä, mutta matkapuhelimen näytöllä sama taulukko vie koko sen leveyden. Rautalankamalleihin on merkitty eri elementtien tarvitsemat kolumnien määrät ja näin voin helposti niitä muuttaa sivustolla. Ensisilmäyksen perusteella ainakin etusivun listaan näyttelyistä tarvitaan enemmän dataa kuin mitä siihen tällä hetkellä tulee, joten pelkkää front-end työskentelyä ei ole tänään luvassa.

Sain päivän aikana ohjelmoitua etusivun rakenteen uusiksi sisäänkirjautumista lukuun ottamatta. Olemassa oleva sisäänkirjautuminen tapahtuu ponnahdusikkunan kautta, mutta se halutaan myös mahdollistaa suoraan etusivulla sijaitsevan lomakkeen kautta. Haastavinta etusivun tekemisessä on eri roolien näkymät. Kirjautumaton käyttäjä näkee kaiken saman kuin kirjautunut käyttäjä, muttei pysty siirtymään tietyille sivuille. Tapahtumajärjestäjän kirjautuessa etusivun rakenne hänelle on taas aivan erilainen. En keksinyt parempaa ratkaisua eri näkymille kuin piilottaa niitä roolien mukaan. Ratkaisu on heikko sillä muiden roolien näkymät saa esille pienellä selaimen kehittäjäkonsolin näpräämisellä ja koodin toistuvuutta on paljon.

Perjantai 21.04.2017

Tänään jatkan front-endin näkymien muuttamisen rautalankamallien kaltaisiksi. Tehtävänä on myös laajentaa back-endin näyttelyiden listausta tekevää metodia. Näyttelyt halutaan listata omalle sivulleen ja uuden suunnitelman myötä myös etusivulle. Tällä hetkellä metodi tarkistaa näyttelykohtaisesti onko käyttäjällä järjestelmään lisättyjä koiria joita hän voisi näyttelyyn ilmoittaa. Etusivun näyttelyiden listausta pääsee kuitenkin katselemaan kirjautumaton käyttäjä jolloin metodi tuottaa virheen, sillä olemattoman käyttäjän koiria ei ole. Refaktoroinnin lisäksi toteutan front-endiin näyttelyitä listaavan sivun sekä alan muuttamaan näyttelyjärjestäjän hallintapaneelia rautalankamallin mittojen mukaiseksi.

Springin Securityn rajapintaa käyttäen voin saada selville onko metodia suorittava käyttäjä kirjautunut, hänen käyttäjätunnuksensa sekä hänen eri roolinsa. Muutin näyttelyitä listaavan toiminnallisuuden hakemaan mahdolliset ilmoitettavat koirat näyttelyihin jos käyttäjä on kirjautunut sisään ja hän omistaa tavallisen käyttäjän roolin. Ratkaisu oli yksinkertainen eikä vaatinut ollenkaan muutoksia front-endiin. Näyttelyitä listaavalle sivuille sain HTML-taulun kopioitua etusivulta. Lisäsin taulukkoon kolumneja vastaamaan tälle sivulle asetettuja tarpeita. Iltapäivä kului yrittäessäni keksiä hyvää ja selkeää

ratkaisua näyttämään näyttelykohtaisesti mahdollisia ilmoitettavia koiria kuitenkin tuloksetta.

3.9.1 Viikkoraportti

Viikko sujui rauhallisesti työpaikalla. Tehtävät pitivät sisällään ainoastaan vanhojen tehtyjen ominaisuuksien laajentamista sekä edellisenä viikkona aloitettua rautalankamallien pohjalta tehtävää sivujen rakenteen luomista. Töiden parissa oli erityisten mieluisaa se ettei tietokantaan tullut muutoksia. Tietokantaa päivittämistä hallitsee ”Liquibase”-niminen kirjasto. Kirjasto mahdollistaa tietokantamuutosten generoimisen esimerkiksi Javan domain-tason olioiden muuttuessa. Nämä muutokset otetaan tietokannassa käyttöön kun back-endin suoritetaan seuraavan kerran. Tietokantataulujen määrän ollessa suuri sen käynnistyminen datan importoimisen lisäksi vie aikaa.

Kehittämässäni järjestelmässä ei ole paljoa vaativaa logiikkaa sisältäviä toiminnallisuuksia. Keskiviikkona käsittelemäni koirien arvostelemiseen liittyvä logiikka on tällä hetkellä järjestelmän vaativinta. Merkittävä osa työstäni on yksinkertaisemman logiikan kirjoittamisen jälkeen datan käsittelemistä front-endissä. Kehitystyöhöni on muodostunut projektin aikana selkeä kaava, jonka mukaan kehitän uusia ominaisuuksia. Aloitan aina suunnittelemalla ja määrittelemällä ominaisuuden sisältämät asiat ja tarpeet. Tämän jälkeen käytän JHipsteriä ominaisuuden tiedostojen luomisessa. Tiedostojen generoitumisen jälkeen kirjoitan toiminnallisuutta vaativan logiikan back-endiin ja testaan eri käyttötapaukset lähettämällä kutsuja järjestelmän rajapintaan komentorivin kautta. Viimeisenä vuorossa on ominaisuuden front-endissä tarvittavien osien toteutus.

3.10 Seurantaviikko 10

Maanantai 24.04.2017

Jäin kuumeen takia kotiini tekemään töitä etänä. Halusin saada rautalankamallit tehdyksi ennen keskiviikkoa esimiehieni kanssa pidettävää demotilaisuutta. Rautalankamallien määritteet täytyy vielä implementoida ilmoittaja-tason käyttäjän ”omat koirat”-sivulle sekä kokonaisuudessaan näyttelyjärjestäjän hallintapaneeliin. Nämä alueet ovat kuitenkin nopeammin toteuttavissa, sillä kyseisten sivujen ulkoasu on lähes vastaava rautalankamallien kanssa. Päivän tavoitteena on toteuttaa nämä sivut.

Sain muutettua kaikki loput sivut vastaamaan graafikon tekemiä rautalankamalleja. Ilman tyylejä ne näyttävät kuitenkin hyvin yksinkertaisilta ja irrallisilta. Muutoksissa jouduin vaihtamaan joitain HTML-elementtejä toisiin elementteihin, joka kasvatti työmäärää kohtalaisesti. Olen front-endin kehittämisessä käyttänyt paljon Bootstrapin sisältämää "panel"-komponenttia joka on yksinkertaisesti otsikollinen laatikko pyöristetyillä kulmilla. Kyseisellä komponentilla olen saanut nopeasti sijoitettua eri dataa sisältäviä osia sivuille. Työpäivä jäi muuten lyhyemmäksi sairauden takia.

Tiistai 25.04.2017

Aloitan tiistain jatkamalla etätyöskentelyä kotoa käsin. Eilen iltapäivällä saapuneessa sähköpostissa esimieheltäni pyydettiin, että etusivun ja tapahtumasivun tapahtumataulukkoa voisi järjestää sen kolumnien perusteella niitä painamalla. Esimerkiksi painamalla näyttelyn nimeä ensimmäisen kerran näyttelyt siirtyisivät aakkosjärjestykseen ja toiselle painalluksella käänteiseen järjestykseen. Olen toteuttanut samanlaisen toiminnallisuuden tapahtuman omien kotisivujen "aikataulut"-sivulle, jossa rotuja pystyy järjestämään aakkosjärjestykseen. Tehtävä vaatii kolumnien sisältämän tekstin muuttamisen painettavaksi hyperlinkiksi joka muuttaa taulukon näyttelyitä looppaavan AngularJS:n direktiivin ehtoja. Päivän tarkoituksena on mahdollistaa taulukoiden uudelleen järjestäminen kolumnien perusteella sekä tarkistaa vielä kaikki järjestelmän sivut joille tein rakennemuutoksia rautalankamallien seurauksena.

AngularJS käyttää listan objektien looppaamiseen "ng-repeat"-nimistä direktiiviä, jonka ottaa vastaan vaihtoehtoisena parametrina "orderBy"-filtterin määrittämään järjestystä. Painamalla taulukon kolumnia kyseinen suodatin saa lausekkeeksi kaksi arvoa: arvon jolla taulukko suodatetaan sekä totuusarvomuuttujan. Totuusmuuttuja määrittää suodatetaanko kolumni normaalissa vai käänteisessä järjestyksessä. Käytin aikaa tekemällä taulukon suodatusominaisuudet huolellisesti jokaiseen kolumniin ja lisäsin nuolenkärki-ikonin havainnollistamaan suodatettua kolumnia. Taulukko mielestäni hyvin ja se toimii kuten pitää. En löytänyt sivustolta korjattavaa joten se on valmis esiteltäväksi huomisessa demotilaisuudessa.

Keskiviikko 26.04.2017

Tänään käyn toimistolla vain demotilaisuuden ja sen jälkeisen palaverin verran. Demossa käydään läpi kaikki uudet toiminnallisuudet kahden viikon ajalta sekä katselmoidaan miltä rautalankamalli sivuilla näyttää. Ennen tilaisuutta syötän järjestelmään testidataa havainnollistamisen helpottamiseksi. Demon jälkeen kirjaan tarvittavat muutokset ja uudet asiat ylös ja lähden kotiin lepäämään.

Demo eteni hyvin ja sain esiteltyä kaikki mitä olin tähän asti tehnyt. Tilaisuudesta ilmeni että jotkin rautalangat tarvitsevat vielä muutoksia ja yksi uusi ominaisuus tulee kehittää järjestelmään. Näyttelyjärjestäjän hallintapaneeliin halutaan "todo"-listan tyylinen tehtäväluettelo johon asiakas itse voi kirjoittaa muistiin asioita. Saimme listan toteuttamiseen vaadittavat tarpeet hyvin määriteltyä mock-up kuvien lisäksi yhdessä esimiesteni kanssa. Tilaisuudessa totesimme myös tämänhetkisen ilmoittautumisen numerolapun koirien järjestysnumeron generoimisen logiikan olevan viallinen. Kyseisten numeroiden generoiminen on loogisesti hyvin monimutkainen ja useita sisäkkäisiä for-silmukoita sisältävä lähdekoodin osa. Toteutin demotilaisuuden jälkeen tehtäväluettelon tietokantataulun ja siihen liittyvät back-endin toiminnallisuudet.

Torstai 27.04.2017

Sairasloma

Perjantai 28.04.2017

Sairasloma

3.10.1 Viikkoraportti

Viimeinen seurantaviikko jäi vajaaksi sairauden yllättäessä. Yksi riskeistä kehittää järjestelmää yksin on se, että esimerkiksi sairaustapauksessa työ seisahtuu lähes kokonaan. Tällä viikolla sain järjestelmän sisältämään loput graafikon suunnittelemista ulkoasun rakenteista. Sivujen rakenteet antavat kehitystä helpottavia rajoitteita elementtien koolle, niiden ansiosta suunnittelutyöhön käyttämäni aika vähenee jatkossa.

Lähitulevaisuudessa alan implementoimaan järjestelmään graafikon luomia CSS-tyylitiedostoja, jossa piilee omat haasteensa minulle. Kyseinen ohjelmistokehitykseen liittyvä osa-alue on osaamiseni heikoimpia puolia ja olen keskittynyt siihen vähiten.

Kokonaisuudessaan järjestelmä lähenee julkaisukelpoista versiota, ainoastaan halutut uudet ominaisuudet siirtävät suunniteltua aikataulua yhä eteenpäin. Kehotin työpaikallani priorisoinnin kannattavuutta keskittyen ensin välttämättömiin ja kriittisiin ominaisuuksiin, joilla saadaan julkaisukelpoinen palvelu. Aikataulu on muutenkin tällä hetkellä kiireinen eikä esimerkiksi testaussuunnitelmaa ole toteutettu. Ketterään kehitykseen kuuluu termi ”minimal viable product” eli MVP tarkoittaa julkaistavaa tuotetta pienimmällä mahdollisella toteutuksella. (Haapajoki 2015) Ilmon tapauksessa järjestelmän ominaisuuksien määrä on jo ylittänyt MVP:n sisältämät rajoitteet.

Etätyön tekeminen on mahdollista työpaikallani ja olen aina satunnaisesti tehnytkin etäpäiviä. Mielestäni etätyöpäivät sopivat työtehtäville, joissa määrittely on valmis ja en tarvitse ominaisuuksien ja logiikan kanssa apua. Järjestelmän kehitys ei ole riippuvainen muista kehittäjistä kuin itsestäni joten voisin tehdä mahdollisesti enemmänkin etätyötä. Ohjelmistokehitystiimissä etätyöskentelyyn järjestämiseen pitäisi kuitenkin ottaa huomioon työkaverit ja heidän tarpeensa. Olen kuitenkin huomannut että toimistolla työskennellessäni olen hieman tehokkaampi ja keskittyneempi. Kotona ärsykkeiden ja muiden houkutusten määrä on suurempi ja itseni kannalta haluan erotella kodin ja työpaikan erikseen työpäivän jälkeen. Etätyöpäivinä olen yhteydessä toimistolle sähköpostilla tai Slackin kautta.

Ohjelmistokehityksen alalla etätyö tekeminen on usein mahdollista käytännössä. Kannettavalla tietokoneella työaseman voi pystyttää kotiin tai kahvillaan ja nykyteknologian myötä kommunikoimiseen on käytössä useita erilaisia työkaluja ja ohjelmia fyysisen läsnäolon sijaan. Yhdysvalloissa tehdyn tutkimuksen mukaan 23% osallistuneista tekee etätöitä kerran tai kahdesti kuukaudessa, 22% 3-5 työpäivää ja 24% yli 10 työpäivää. (Jones 2015). Tutkimukseen osallistui 1011 18-50-vuotiaista aikuista. Etätyön tekemisen mahdollisuus on täysin riippuvainen yrityksestä ja sen käytänteistä. Helmikuussa 2017 teknologiajätti IBM ilmoitti lopettavan työntekijöiden etätöiden mahdollisuuden kokonaan. (Kessler 2017) Tiedotus aiheutti kohun mediassa uuden linjauksen ehdottomuuden takia, nykyisten työntekijöiden täytyy siirtyä toimistolle töihin työsuhteen päättymisen uhalla.

4 Pohdinnat ja päätelmät

Lähtötilanteessani kuvasin osaamiseni tason olevan ”taitavan suoriutujan”-luokkaa ja nyt seurantaviikon jälkeen koen olevani kokonaisuudessaan samalla tasolla. Full-stack ohjelmoijan työnkuva on laaja ja näin sen eri osa-alueilla kehittymisen on hitaampaa verrattuna siihen jos keskittyisi ainoastaan yhteen kokonaisuuteen. Varsinainen työnkuvani on kuitenkin projektin aikana laajentunut kattamaan osa-alueita full-stack kehittäjän roolin ulkopuolelta. Olen työaikojeni ulkopuolella harjoitellut Java-pohjaisten applikaatioiden asentamista palvelimelle. Tuotantoon siirtyessä tulen kuitenkin hoitamaan järjestelmän palvelinasennuksen, tietokannan pystyttämisen sekä kaiken muun konfiguroinnin aina virtualhostista määrittymisestä SSL-sertifikaatin asentamiseen. Koen kuitenkin näistä kaikista olevan hyötyä tulevaisuudessa esimerkiksi DevOps-tyylisissä työtehtävissä.

Suurin kehittymisen osa-alueeni oli yllätykseksi projektinhallinta ja erityisesti määrittely ohjelmointitaitojen sijaan. Määrittelyssä selvitän nykyisin nopeammin eri käyttötapauksia ja poikkeustilanteita, jotka voivat aiheuttaa häiriöitä järjestelmään. Sanonta ”mittaa kahdesti, leikkaa kerran” kuvaa hyvin nykyistä ajattelutapaani ohjelmistokehityksessä. Huonosti suunniteltu ja löysästi määritellystä ominaisuudesta voi helposti tulla vakavampi ongelma projektin edetessä tai jatkokehityksessä. Itse kohtasin seurantaviikkojen aikana muutaman tällaisen tilanteen. Vanhaan lähdekoodiin palaaminen voi tarkoittaa, että sen sisältämä logiikka täytyy käydä läpi. Refaktorointi ja uudelleen testaus on työlästä ja pidentää aikataulua aina. Omasta kehittymisestä konkreettisesti itselleni kertoo se, että oma vanhempi tuotos näyttää epäloogiselta tai muuten huonolta koodilta.

Ohjelmointitaidot ovat myös kasvaneet kaikilla saroilla päiväkirjamuotoisen opinnäytetyön aikana. Suurin murros tapahtui back-endin puolla sisäistäessäni Javan 8:n version tuoman funktionaalisen lähestymistavan. Version sisältämien rajapintojen ansiosta lähdekoodini tuntuu ja näyttää paljon ammattimaisemmalta. Streamit ovat nopeuttaneet listojen käsittelyä ja lambda-lausekkeet ovat vähentäneet ”turhien” objektien luomista. Yleisesti nykyisin tuottamani koodi on paljon suoraviivaisempaa ja loogisempaa verrattuna opinnäytetyön alkuun. Järjestelmä on nyt myös arkkitehtuurillisesti paremmin tehty kuin seurantaviikon alussa. Olen siirtänyt logiikkaa pois controller-luokista service-luokiin ja näin toteuttaen selvemmin kolmitasoarkkitehtuuria määritteitä. Isommille ominaisuuksien osille olen luonut kokonaan omia service-luokkia rakenteen selkeyttämiseksi. Oppiminen näkyy myös siinä, että refaktoroinnin ja logiikan uudelleenkirjoituksen jälkeen Java-tiedostojen rivimäärät ovat yleensä vähentyneet huomattavasti.

Merkittävimpiä oppimiani konsepteja seurantaviikkojen aikana ovat olleet maksuintegraation luominen ja PDF-dokumenttien generointi. Maksamisen toteuttaminen ohjelmoimisella on kuulostanut aina korkean luokan konseptilta, mutta päästessäni toteuttamaan sitä koko prosessin kulku aukeni minulle. Vaikka järjestelmän maksuintegraatio toteutettiin kolmannen palveluntarjoajan kautta sain selvyyden miten esimerkiksi verkkokaupat toteuttavat ostosten maksamisen Ominaisuuden tekeminen avasi maksamisen ulkopuolelta myös muita konsepteja. Opin kuinka eri järjestelmät voivat kommunikoida toistensa kanssa suojatusti. Kaikki viestit salakirjoitetaan ennen toisen järjestelmän rajapintaan lähettämistä, jossa ne sitten avataan salausavaimella.

JavaScriptiä ohjelmoidessa front-endissä saan tuloksia näkyviin yleensä nopeasti. Syntaksi eli kielen muodon tullessa tutummaksi virheiden määrä on vähentynyt ja minun tarvitsee entistä harvemmin tarkistaa onko selaimen kehitystyökalun konsolissa virheilmoituksia. Joudun kuitenkin käyttämään aikaa ongelmien selvittämiseen ja niiden ratkaisujen hakemiseen internetistä. Kirjastojen lisääminen projektin front-endiin tuntuu edelleen haastavalta ja joudun usein palauttamaan versiohallinnasta vanhemman version lisäämisen epäonnistuessa. Suhtautumiseni front-endin kehitykseen on positiivisempaa kuin projektin alussa joka näkyy kehityksen ollessa mielekkäämpää tietyillä osa-alueilla. Viimeisellä seurantaviikolla toteuttamani taulukon uudelleenjärjestämisen ominaisuutta oli mielenkiintoista työstää ja opin siitä paljon uutta. Osaamisen karttuessa korjaus- ja kehitystarpeita alkaa huomaamaan enemmän.

Olen pyrkinyt organisoimaan työn tekemistäni paremmin yrittäen saaden siitä johdonmukaisempaa. Seurantaviikoilta on helposti havaittavissa etteivät päivien työt välttämättä liity ollenkaan edellisenä päivänä tekemiini työtehtäviin, vaan saatan palata keskenjääneen ominaisuuden työstämiseen parinkin viikon kuluttua. Johdonmukaisuus kehitystyössä helpottaisi projektinhallintaa. Kokonaisuudet tulisi tehdä huolellisesti kokonaan loppuun ennen seuraavaan siirtymistä. Suurimman haasteen omassa työntekemisessäni olen huomannut liittyvän työtehtävien kiinnostavuuteen. Järjestelmässä on paljon ns. ”rutiiniohjelmointia” eli ohjelmointia, joka toistuu eri paikoissa samanlaisena eikä varsinaisesti vaadi logiikan luomista. Kyseinen ohjelmointi on aikaa vievää ja pitkäveteistä, mutta se on tehtävä kuitenkin. On helppo harhautua kehittämään mielekkäämpää toiminnallisuutta, jolloin edellinen työtehtävä jää kesken.

Päiväkirjamuotoisen opinnäytetyön aikana huomasin aamulla asetettujen tavoitteiden motivoivan tekemistä koko päivän ajan. Kirjoittaessa työpäivän aikana tehtävistä tehtävistä suuntasi minua suunnittelemaan ja määrittelemään enemmän päivän aikana toteutettavia työtehtäviä. Ennen seurantaviikkojen alkua lähdin tavallisesti jatkamaan

edellisenä päivänä keskenjäänyttä ominaisuutta ja tein sitä sinä päivänä niin pitkälle kuin sain. Tavoitteiden asettaminen tehtäville auttaa muodostamaan niiden valmiin määritelmän paremmin (eng. ”definition of done”) ja työpäivät tuntuvat vähemmän kaoottiselta. Valmiin määritelmän ansiosta seurantaviikkojen aikana tehdyt järjestelmän osat valmistuivat useammin kokonaan valmiiksi verrattaessa aikaan niitä ennen. Täysin valmiiksi tehdyt osat vähentävät teknistä velkaa ja antavat tarkemman kuvan koko järjestelmän valmiusasteesta.

Jatkossa tulen suunnittelemaan päivieni sisällöt tarkemmin ja pyrin asettamaan niille tavoitteita. Suunnittelemalla ja määrittelemällä tarkemmin voin välttää koodin turhaa uudelleenkirjoitusta. Ohjelmoinnin kannalta Javan 9. versio on tulossa pian ja näin tuomassa jälleen uutta opeteltavaa. Törmäsin opinnäytetyön aikana ”Kotlin”-ohjelmointikieleen, joka on hyvin Javan tapainen, mutta sisältää mielestäni parempia ratkaisuja sekä kirjoittamisen kannalta paremman syntaksin. En tule muuttamaan Ilmoa käyttämään Kotlinia vaan aion opetella sen tulevaisuutta varten. Front-endin puolelta aion jatkossa korjata puutteellisen JavaScript osaamiseni. Tällä hetkellä osaan käyttää vain AngularJS-viitekehystä, mutta en oikeastaan perinteistä Javascriptiä peruskäsitteiden ulkopuolelta.

Kehittymistä löytyy vielä niin ohjelmistokehityksen kuin projektihallinnankin saralla. Työkuvani laajentuessa uusia asioita on paljon opeteltavana vanhojen teemojen ulkopuolella. Oppimistapani on muuttunut projektin ja opinnäytetyön kirjoittamisen aikana kauemmaksi yritys-erehdys-oppimisesta. Koen olevani pisteessä, jossa nykyisessä toimenkuvassani tarvitut taidot ovat siinä pisteessä että työntekeminen on selvästi sujuvampaa ja luontevampaa. Pystyn omaksumaan osaamisalueeseeni liittyvät konseptit nopeammin pienemmällä työmäärällä. Koko järjestelmän kehitys yksin on johtanut siihen että olen oppinut laajan kirjon eri teknologioita ja käytäntöjä niin ohjelmointikielissä, kehityksen viitekehityksissä kuin ohjelmistokehityksessä yleensä. Projektin luonne ja mahdollisuuteni suunnitella sen toteutus itse on edesauttanut uusien konseptien harjoitteluja, mutta myös vanhojen taitojen syvempää hiomista. Ainoastaan varmuus siitä, ovatko ratkaisuni tehty oikein, on työympäristöni takia vaikea saavuttaa. Oman työn itsekriittisyys on korostunut projektin aikana. Oikean rakenteen luominen ja hyvien käytäntöjen noudattaminen vähentävät teknistä velkaa ja edesauttavat omaa tai tulevan työntekijän järjestelmän jatkokehitystä.

5 Lähteet

Andrew, O. 20.03.2017. How Good Web Design Influences Consumer Behavior. ITBusinessEdge. http://www.smallbusinesscomputing.com/emarketing/how-good-web-design-influences-consumer-behavior.html. Luettu: 25.03.2017

AngularJS 2017. AngularJS. Luettavissa: https://angularjs.org/. Luettu: 07.04.2017

Auth0 2017. Introduction to JSON Web Tokens. Luettavissa: https://jwt.io/introduction/. Luettu: 18.03.2017

Bambora 2017. Bambora Api Docs. Luettavissa: https://payform.bambora.com/docs/web_payments/?page=testing. Luettu: 22.2.2017

Cox, J, P 10.12.2015. Luettavissa: https://motherboard.vice.com/en_us/article/why-you-dont-roll-your-own-crypto. Luettu: 15.02.2017

Finlex 1999. Henkilötietolaki 22.4.1999. Luettavissa: http://www.finlex.fi/fi/laki/ajantasa/1999/19990523#L7. Luettu: 18.03.2017

Flower, M. 01.10.2003. TechicalDebt. Luettavissa: https://martinfowler.com/bliki/TechnicalDebt.html. Luettu: 10.04.2017

GitHub 2017. Most Starred Repositories. Luettavissa: https://github.com/search?q=stars:%3E1&s=stars&type=Repositories Luettu: 8.4.2017

Haapajoki, S. 22.02.2015. Minimal Viable Product – Mikä on MVP ja miten se tehdään? Klopal Oy. Luettavissa: https://www.klopal.fi/2015/02/22/minimum-viable-product-mika-on-mvp-ja-miten-se-tehdaan/. Luettu: 10.05.2017

Haddad, E. 08.08.2015. SERVICE-ORIENTED ARCHITECTURE: SCALING THE UBER ENGINEERING CODEBASE AS WE GROW. Uber Engineering. Luettavissa: https://eng.uber.com/soa/. Luettu: 26.03.2017

JHipster 2017. Securing your application. Luettavissa: https://jhipster.github.io/security/. Luettu: 18.03.2017

Jovanovic, J. 07.07.2009. Web form Validation: Best Practices and Tutorials. Smashing Magazine. Luettavissa: https://www.smashingmagazine.com/2009/07/web-form-validation-best-practices-and-tutorials/. Luettu: 01.05.2017

Kemppainen, M. 26.6.2014. Käyttöliittymäsuunnittelua käytännössä. Provianet Oy. Luettavissa: https://www.provianet.fi/kayttoliittymasuunnittelua-kaytannossa/**.** Luettu: 08.05.2017

Kessler, S. 21.03.2017. IBM, remote-work pioneer, is calling thousands of employees back to the office. Quartz Media LCC. Luettavissa: https://qz.com/924167/ibm-remote-work-pioneer-is-calling-thousands-of-employees-back-to-the-office/. Luettu: 30.04.2017

Michael, M. 16.02.2017. THE STATE OF CYBER SECURITY 2017. F-Secure. Luettavissa: https://business.f-secure.com/the-state-of-cyber-security-2017. Luettu: 01.05.2017

Mozilla Foundation 2017. Form data validation. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation. Luettu: 01.05.2017

Paunovic, G. 04.2012. THE IMPORTANCE OF USER INTERFACE DESIGN. Art Version Company. Luettavissa: https://artversion.com/blog/importance-of-user-interface-design/. Luettu: 05.05.2017

Pink, G. 29.8.2008. The Benefits of Code Review. Microsoft. Luettavissa: http://blogs.microsoft.co.il/gilf/2008/08/29/the-benefits-of-code-review/. Luettu 26.03.2017

Pivotal 2015. Spring Security Reference. Luettavissa: https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/. Luettu: 18.3.2017

Richardson C. 2017. Pattern: Monolithic Architecture. Microservices.io. Luettavissa: http://microservices.io/patterns/monolithic.html. Luettu: 07.04.2017

Toniut, L. 25.08.2015. Analysing the SQL Server Numeric Data Types. Vertabelo sp. z.o.o. Luettavissa: http://www.vertabelo.com/blog/technical-articles/the-right-data-for-the-job-analysing-the-sql-server-numeric-data-types. Luettu: 26.03.2017

Usability.gov 2017. User Interface Design Basics. Luettavissa: https://www.usability.gov/what-and-why/user-interface-design.html. Luettu: 01.05.2017