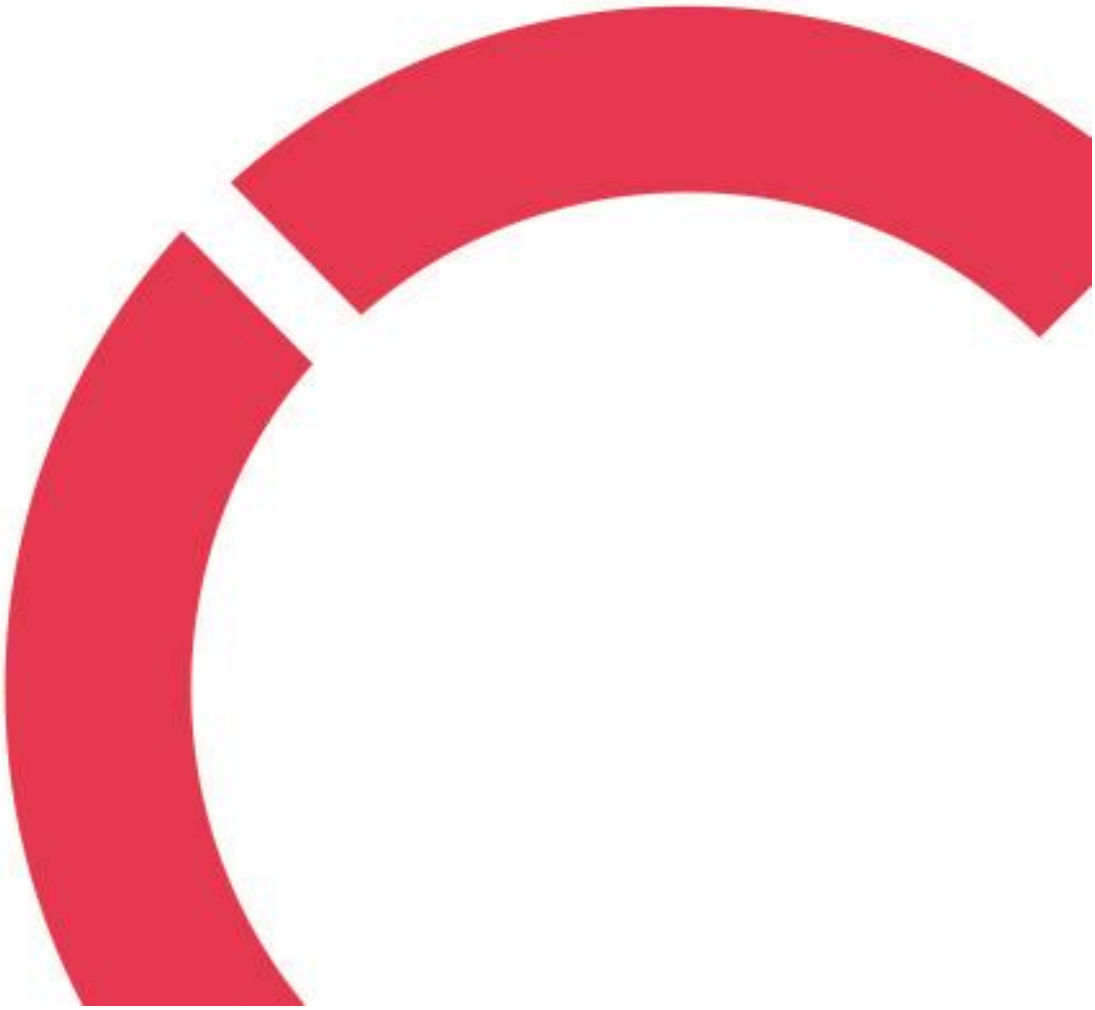
Mika Kontio

OHJELMISTOSUUNNITTELIJAN 9 VIIKKOA

Päiväkirjamuotoinen opinnäytetyö

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutus
Joulukuu 2022



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Joulukuu 2022	**Tekijä/tekijät** Mika Kontio
Koulutus Tieto- ja viestintätekniikka		☐ AMK ☐ YAMK
Työn nimi Ohjelmistosuunnittelijan 9 viikkoa		
Työn ohjaaja Jari Isohanni		**Sivumäärä** 33
Työelämäohjaaja Pasi Harju		

Opinnäytetyössä seurattiin päiväkirjamuodossa ohjelmistotyöntekijän työskentelyä Livion Oy IT-alan yrityksessä yhdeksän viikon ajanjakson verran. Työn tavoitteena on syventää kirjoittajan tuntemusta tämän työskentelytavoista ja antaa näkemys siitä, miten voisi parantaa työntekoa. Työstä ilmenee ohjelmistoalan työntekijän monipuolinen arki haasteineen ja onnistumisineen, ja mitä IT-alalla työskentelyssä voi odottaa työtehtäviltä ja työyhteisöltä. Opinnäytetyön tekijä analysoi tehtyä työtä ja tapahtumia ja reflektoi suoriutumistaan viikkoanalyyseissä.

Asiasanat
Ohjelmointi, ohjelmistosuunnittelu, päiväkirja, reflektio.

ABSTRACT

Centria University of Applied Sciences	**Date** December 2022	**Author** Mika Kontio
Degree programme Information Technology		
Name of thesis Software Developer's 9 weeks		
Centria supervisor Jari Isohanni		**Pages** 33
Instructor representing commissioning institution or company Pasi Harju		

In this study, the work of a software engineer was watched in Livion Oy IT-company in a diary format for a period of nine weeks. The goal of this study is to deepen the writer's self-knowledge of their working habits and give insight into how to improve their working. In the study it is found how varied ordinary workdays of a software engineer can be with their challenges and successes and what can be expected from IT-job's working tasks and environment. The writer analyses done work and events, and reflects on their performance.

Key words
Programming, software design, diary, reflection.

KÄSITTEIDEN MÄÄRITTELY

GCS

(Google Cloud Storage) Googlen pilvipalvelu tiedostojen tallentamiseen ja jakamiseen.

UUID

(Univeral Unique Identifier) Universaalisti ainutlaatuinen tunniste.

1 JOHDANTO

Tämä työ on opinnäytetyö päiväkirjan muodossa ja kirjoitan sen Livion Oy IT-alan yrityksessä, jossa olen töissä ohjelmistosuunnittelijana. Päiväkirjamuotoinen opinnäytetyö koostuu päivittäisistä raporteista, joissa asetetaan kullekin päivälle tavoite ja selvennetään, mitä päivän aikana tapahtui ja mihin tavoitteisiin päästiin. Lisäksi kullekin viikolle kirjoitetaan viikkoraportti, joissa analysoidaan tapahtunutta koko viikon ajalta.

Aikaväli opinnäytetyölle on 1.4.2022 - 30.11.2022. Valitsin ajankohdan, koska tehtäväni edellisessä projektissa päättyi 31.3. ja siirryin seuraavaan projektiin. Koin ajankohdan täydelliseksi aloittaa kirjoittamaan opinnäytetyötä päiväkirjana, koska uskon sen helpottavan siirtymistäni seuraavaan projektiin ja auttavan kunkin sen hetkisen tehtävän käsittelyä, sillä päiväkirjan kirjoittaminen pakottaa jäsentämään ajatuksia.

Livion Oy on Kokkolassa sijaitseva IT-alan yritys, joka keskittyy modernien verkko-ohjelmistoratkaisujen kehitykseen, joilla on tarkoitus automatisoida sellaisia tehtäviä, joita ihmisen on vaikeaa, vähäpätöistä tai tarpeetonta suorittaa. Livionilla on yrityksenä yli kymmenen vuoden tausta, ja alussa kehitettiin pieniä verkko-ohjelmia täyttämään palvelualan yritysten tarpeita. (Linkedin.)

Työskentelyyn Livion Oy:llä tarvitaan laajaa osaamista modernien verkkosivujen kehittämisestä. Avainkäsitteitä ja vaadittuja osaamisen tai tuntemuksen alueita ovat HTML, CSS, JS, TS, NodeJS, React, Git, Cloud, Kubernetes, Docker, IOT, ja monet näihin käsitteisiin pohjautuvat teknologiat. Pohjimmillaan työntekijältä tarvitaan taitoa ratkaista loogisia ongelmia ja hyvää kommunikoinnin taitoa niin kollegoiden kuin asiakkaiden kanssa.

2 LÄHTÖTILANTEEN KUVAUS

Livionilla on toimistotilaa Kokkolassa ja Oulussa, joista pääsääntöisesti Kokkolassa valmistetaan ohjelmistoja ja Oulussa laitteita. Kokkolan toimistossa tilaa on noin viidelletoista työntekijälle, mikä on sopiva nykyiseen työntekijöiden määrään nähden. Monilla IT-alan yrityksillä on työntekijöillä mahdollisuutena tehdä töitä kotoa, sillä työskentely ja siihen liittyvät tehtävät voidaan suorittaa internetin välityksellä, kuten työn tuloksien jakaminen ja tapaamisten pitäminen toisten työntekijöiden ja asiakkaiden kanssa. Toimistotilalle on tarvetta etenkin silloin, kun kehitetään laitteita, testataan niille suunniteltua ohjelmistoa ja tarvitaan säilytystilaa kyseisille laitteille, mutta myös esimerkiksi silloin, kun halutaan järjestää tapaamisia tai kokouksia kasvotusten, tai kotioloissa on vaikeuksia saada työrauhaa, tai halutaan pystyä nopeasti konsultoimaan vertaisia eri asioissa. (Linkedin.)

Aloitin työskentelyn Livion Oy:llä vuonna 2014, ja kaikki työkokemukseni on kertynyt yrityksen alaisuudessa. Aloittaessani yrityksen toimintamalli oli vielä asiakasprojektipohjainen, eli ohjelmistoja suunniteltiin ja valmistettiin tilauksesta yrityksen ulkoisiin tarpeisiin. Viime vuosina yrityksen suunta on muuttunut sisäänpäin, kun on haluttu keskittyä omaan tuotteeseen, jonka kehittäminen ja myynti jatkuvat edelleen. Projekti, johon keskitytään, on nimeltään Connect2. Projektia on tehty useita vuosia, ja se sisältää useiden kymmenien koodikantojen ja päätelaitteiden kehitystyön. Projekti vaatii kokonaisuudessa osaamista verkko-ohjelmistojen kehittämisessä, verkkoinfrastruktuurin ylläpidossa sekä päätelaitteiden suunnittelussa ja valmistamisessa. Tarkemmin vaaditaan yleisesti osaamista tietokoneiden, eri käyttöjärjestelmien, koodieditorin, komentoliittymän, NodeJS:n, JavaScriptin sekä useiden kolmansien osapuolien kehittämistä työkaluista.

Toimin Connect2-projektissa ohjelmistosuunnittelijana ja ohjelmistokehittäjänä. Pystyn vaikuttamaan siihen, millaiseksi ohjelmiston yksittäiset osa-alueet kehittyvät esittämällä ideoita ja ehdotuksia siihen, miten asioita voitaisiin tehdä tai miltä lopputulokset voisivat näyttää. Kehittäjänä olen vastuussa koodin tuottamisesta, sen laadusta ja toiminnallisuuden varmistamisesta. Vastuuseen kuuluvat myös kollegoiden tuottaman koodin laadun ja toiminnallisuuden katselmointi ja edistäminen. Luultavasti en tule työskentelemään päätelaitteiden tai niiden sisältämän koodin kanssa, vaan käyttöliittymien ja päätelaitteiden käyttämien rajapintojen kanssa. Työskentelen projektissa monen saman roolin omaavan työntekijän kanssa. Olemme pieni kehitystiimi, jonka tarkoitus on tukea jäseniään ja parantaa jäsenten tehokkuutta työnteossa avustamalla tehtävien aloituksessa ja suunnittelussa.

Alkuun keskityn projektin koodikantojen connect2-apps ja connect2-backend sisältämien ominaisuuksien kehittämiseen ja parantamiseen. Connect2-apps on koodikanta, joka sisältää usean käyttöliittymän lähdekoodin ja connect2-backend taas sisältää lähdekoodin usealle ohjelmalle, jotka toimivat rajapintoina connect2-appsin sisältämille käyttöliittymille ja datan käsittelijöinä.

3 PÄIVÄKIRJARAPORTOINTI

3.1 Viikko 1

Perjantai

Ensimmäisen työpäiväni tässä projektissa käytin tärkeimpien ohjelmistojen lähdekoodin tutkimiseen, jotka sisältyvät Connect2-apps- ja Connect2-backend-monorepoihin. Monorepo on säiliö projekteille, ja sen tarkoitus on helpottaa ohjelmistojen versionhallintaa ja ylläpitoa. Monorepojen käytön etuja ovat jaettu koodi, koodipakettien versioiden yhtenäisyys, näkyvyys muille koodaajille, yhtenäiset työkalut sekä jokainen monorepon projektin rakentuminen samalla prosessilla. Livion-yrityksen tapauksessa Connect2-appsia käytetään käyttöliittymäprojektien, ja Connect2-backendiä taustapalveluiden, kuten rajapintojen ja näkymättömien järjestelmäprojektien ylläpitoon, sillä ne käyttävät paljon samaa koodia ja jakavat näennäisen ohjelmistotyypin. (Monorepo Tools.)

Etuna Livion-yrityksen monorepoarkkitehtuurissa on, että projektien asentaminen kehittämistä varten on nopeaa ja helppoa, sillä tarpeenmukainen koodiriippuvaisuuksien asentaminen tapahtuu yhdellä komennolla, ja riippuvaisuuksia, joita komento ei asenna, on vähän, kuten tietokanta-ajurit. Asensin monorepojen projektit ja tutkin niitä tämän päivän ajan.

Viikkoanalyysi

Sisällytetty ensi viikon analyysiin.

3.2 Viikko 2

Maanantai

Connect2-apps ja Connect2-backend on suunniteltu ja valmistettu GCS-pilvipalvelussa ajettaviksi ohjelmiksi. Asiakkaan tarpeiden vuoksi näistä palveluista on saatava paikallisella tietokoneella ajettavia ohjelmia. Tätä tehtävää suorittavat minun lisäkseni kollegani, joten minulle päätyy vain osa tarpeellisten muutosten aiheuttamasta työstä.

Ensimmäinen tehtäväni on korvata nykyinen tiedostojen käsittelyjärjestelmä. Se tallentaa tiedostoja, kuten dokumentteja ja kuvia, GCS-pilvipalveluun, luo ja jakaa julkisen linkin näihin tiedostoihin, ja lähettää tiedoston, kun linkkiä käytetään (Google Cloud Storage). Korvaavan järjestelmän kuuluu hoitaa samat asiat, mutta käyttäen paikallisen tietokoneen kiintoasemaa tiedostojen tallentamisen sijaintina. Kutsun vanhempaa GCS:tä käyttävää moduulia GCS-moduuliksi ja uutta onpremise-moduuliksi.

Jaoin onpremise-moduulin rakentamisen osa-alueisiin:

- Rakenna uusi moduuli, jota käytetään määritettyjen ehtojen täyttyessä
- Rakenna funktio, joka tallentaa tiedoston paikalliselle kiintoasemalle haluttuun polkuun
- Rakenna funktio, jolla voi poistaa tiedostoja
- Lisää tiedostojen tallentamiseen ominaisuus, jolla voidaan korvata saman polun tiedosto
- Rakenna funktio, joka tallentaa tietokantaan tiedostojen metatiedot eli lisätietoja, jotka eivät liity tiedoston sisältöön.
- Rakenna rajapinta tiedostojen noutamiseen

Tiistai

Olen työskennellyt Connect2-backend-projektissa vuosia sitten, joten minulla on rajallisesti kokemusta ja tietoa taustalla olevasta järjestelmästä. Päätin aloittaa onpremise-moduulin kirjoittamisen laatimalla kirjallisen suunnitelman sen sijaan, että ryhdyn välittömästi koodaamaan. Toivon tämän auttavan minua huomaamaan asioita, joita on parempi selvittää ennen kuin ryhdyn koodaamaan, kuten onko taustajärjestelmässä valmiina sellaisia osia, joita voin hyödyntää, jotta en kirjoita jo olemassa olevaa koodia uudelleen.

Tutkin GCS-moduulia ja sitä, miten sitä on käytetty tiedostojen käsittelyssä, minkä pohjalta kirjoitin alustavan suunnitelman, joka sisältää maanantaina mainitsemani moduulin rakentamisen osa-alueiden kohdat. Suunnitelman laatiminen johti minut kysymyksiin:

- Millä perusteella ohjelman käynnistyessä valitaan, mitä moduulia käytetään?
- Suorittaako GCS-moduuli tiedostojen tallentamisen samaan polkuun päällekirjoittamalla edellisen tiedoston
- Ovatko GCS-pilvipalvelun tiedostot julkisia, eli voiko kuka tahansa käsitellä niitä, jos tietää siihen viittaavan verkko-osoitteen
- Miten koodaan rajapinnan, joka lähettää tiedostoja?

Keskiviikko

Tänään selvitin edellisen päivän aikana kehittämiäni kysymyksiä ja aloitin koodaamaan onpremise-moduulia. Kollegat ovat olleet korvaamaton avun lähde, sillä suurimpaan osaan kysymyksistä löytyy lähes välitön vastaus. GCS-moduulin toiminnasta minulle selvisi, että tiedostot päällekirjoitetaan, mikäli niitä ollaan tallentamassa samaan polkuun. Uuden järjestelmän täytyy toimia samalla tavalla. Lisäksi selvisi, että pilvipalvelun tiedostot ovat julkisia, mutta kollegoiden ja oman mielipiteeni mukaisesti onpremise-moduulin kuuluisi sisältää tiedostojen pääsynvalvontaa. Tulevaisuuden kehitysmahdollisuutena GCS-moduulin kuuluisi myös kyetä hallitsemaan tiedostoihin käsiksi pääsyä.

Peruste, jolla tiedostojen tallentamiseen käytetty moduuli valitaan, riippuu ympäristömuuttujasta, ON_PREMISE_STORAGE, joka luetaan, kun Connect2-backend käynnistyy. Muuttujan arvo on joko tosi tai epätosi. Mikäli arvo on tosi, käytetään onpremise-moduulia, muussa tapauksessa GCS-moduulia.

Sain päivän aikana valmiiksi ympäristömuuttujasta riippuvan keinon valita käytettävä moduuli ohjelman käynnistyksen yhteydessä ja tiedostojen tallentamisen ja korvaamisen samassa polussa.

Torstai

Koodausprosessi voidaan jakaa karkeasti kahteen vaiheeseen: kirjoitus ja uudelleenkirjoitus. Ensimmäisessä vaiheessa koodista tehdään toimiva, eli se suorittaa onnistuneesti tehtävää, johon se on suunniteltu. Toinen vaihe, jota usein laiminlyödään ajan säästämisen vuoksi, uudelleenkirjoitus, on tärkeämpi osa koodausprosessia kuin ensimmäinen kirjoitusvaihe, sillä uudelleenkirjoittamisen vaiheessa koodista tehdään järkevä ja ymmärrettävämpi, ja siitä etsitään ja poistetaan bugeja ja tarpeetonta koodin toistumista.

GCS-moduulissa on toistuvaa koodia eri funktioissa, joissa rakennettiin polku tiedostoon eri tavoin tiedoston kirjoitus- luku- ja poistotapauksissa. Kirjoitin onpremise-moduuliin funktion, joka rakentaa tiedoston polun aina samalla tavalla, minkä vuoksi moduuli on hieman lyhyempi, ymmärrettävämpi ja helpommin laajennettavissa, mikäli sille joskus on tarvetta.

Yhtenä ongelmana onpremise-moduulissa oli se, että kun tiedostoja tallennetaan, luetaan tai poistetaan, ohjelma kaatuu, mikäli tiedoston polkurakenteesta puuttuu välikansio. Tämän ratkaistakseni tein erillisen funktion, joka ensin tarkistaa löytyykö tiedosto annetusta polusta, ja jos ei löydy, tapaus käsitellään ilman ohjelman kaatumista.

Onpremise-moduulilla pystyy nyt tallentaman, lukemaan ja poistamaan tiedostoja, ja polun olemassaolon voi varmistaa. Lisäksi se miten tiedostopolku rakennetaan, tapahtuu yhdellä tavalla, eikä eri tavoin kunkin operaatiotyypin mukaan.

Perjantai

Käytän tietokantaa tallentamaan tiedostoihin liittyvää metatietoa. Suunnitelma on tallentaa tiedoston nimi, formaatti, sijaintipolku, sekä UUID. Kun tiedostoa noudetaan mitä tahansa kautta, sen metatiedot ensin noudetaan tietokannasta UUID:n avulla, koska UUID on lähes varmasti ainutlaatuinen (Telecommunication Standardization Sector of ITU, 2014), eikä voi viitata eri tiedoston metatietoihin. Metatiedosta luetaan tiedoston polku, tiedosto luetaan kyseisestä polusta ja lähetetään taholle, joka pyytää tiedostoa. Taustalla olevan järjestelmän vuoksi tämän tietokantarakenteen valmistaminen oli nopea ja helppo tehdä.

Ryhdyin rakentamaan rajapintaa, jonka kautta tiedostoja pystytään noutamaan verkkopyynnöllä https://{DOMAIN}/api/file/{UUID}, jossa {DOMAIN} on rajapinnan verkko-osoite ja {UUID} on kuhunkin tiedostoon viittaava ainutlaatuinen tunniste, joka on luotu tiedoston tallentamisen yhteydessä. Rajapinnan rakentaminen oli yksinkertaista, sillä tiedoston polun hakeminen tapahtuu yhdellä funktiolla, ja sen lukeminen ja tiedoston lähettäminen toisella. Taustalla olevan järjestelmän ansioista sain lisättyä yksinkertaisen mutta tehokkaan tavan suojata tiedostoja ulkopuolisilta tahoilta; järjestelmään täytyy olla kirjautunut sisään, jotta tiedostoja voi noutaa.

Viikkoanalyysi

Ensimmäinen viikko laajassa projektissa, joka on tekijälle lähestulkoon uusi, on haastavaa henkisellä tasolla. Päällimmäisenä tuntemuksena on epäonnistumisen pelko, sillä aluksi ei ole mitään tietoa siitä millainen projekti on. Mitä suurempi projekti on, sitä vaikeampaa ja enemmän aikaa vievää on tutustua kaikkiin sen sisältämiin ominaisuuksiin ja hienovaraisiin yksityiskohtiin. Tärkeässä roolissa tässä tilanteessa ovat oma taitotaso, dokumentaatio ja kollegat. Oma taitotaso lisää itsevarmuutta ja vähentää tuntemattoman pelkoa, dokumentaatio sisältää tiivistettyä tietoa projektista ja kollegat, joilla on projektista mahdollisesti vuosien kokemus pystyvät oikein muotoillulla kysymyksellä auttamaan etenemään ongelmakohdissa nopeasti. Tapauksessani korkeassa asemassa ovat kollegat, joista osa on ollut projektissa sen alusta alkaen, ovat tästä syystä ainutlaatuinen tiedonlähde, kuin harvoissa vuosia vanhassa projektissa on enää jäljellä.

Tämän viikon tavoitteena on ollut rakentaa tiedostojen hallintajärjestelmä korvaamaan aiempi systeemi asiakkaan tarpeiden vuoksi. Kirjoitin onpremise-moduulin, joka tallentaa, lukee ja poistaa tiedostoja paikallisella kiintolevyllä, tallentaa tiedostojen metatietoa tietokantaan, pystyy lähettämään tiedostoja niitä pyytävälle ja saavuttaa korkeamman turvallisuustason kuin GCS-moduuli. Tavoitteen saavuttamista helpotti huomattavasti kollegoiden apu, kun tutustuin taustalla olevaan järjestelmään ja ennalta valmistettuun koodiin. Oli hyvä idea aloittaa uuden koodin kehittäminen ensin luomalla kirjallinen suunnitelma, jossa ilmeni suurin osa uuden koodin tarpeista, koska se helpotti muotoilemaan kysymyksiä siten että toisten oli helppo ymmärtää mitä tarvitsen. Koska olen jatkanut kirjallisen suunnitelman täydentämistä sitä mukaa, kun etenen tehtäväni kanssa, se on muuttumassa suunnitelmasta dokumentaatioksi, jota kollegani ja tulevat tekijät voivat hyödyntää. En olettanut tällaista seurausta sille, että kirjoitin suunnitelman, mikä oli positiivinen yllätys.

3.3 Viikko 3

Maanantai

Kollegojen kanssa keskustelun pohjalta päätimme lisätä toisen tason turvallisuutta onpremise-moduuliin, tarkemmin siihen kuka käyttäjä saa noutaa mitä tiedostoja. Taustajärjestelmässä on tietokantoihin liittyvä systeemi, joka tallentaa turvattuihin dokumentteihin listan tageja, joista käyttäjällä täytyy olla johonkin käyttöoikeus, jotta dokumenttia voi käsitellä. Onpremise-moduulin tiedostojen käsittely vaatii toimiakseen tietokantaan tallennettuja tiedoston metatietoja, joten lisäämällä tagit metatietoihin tiedostojen käsittely voidaan estää sellaisilta käyttäjiltä, joilla ei ole käyttölupaa niihin. Lisäsin kuhunkin sitä vaativaan funktioon vaihtoehtoisen parametrin tageille, jotta tiedostoja tallennettaessa metatietoihin lisätään tagi, ja tiedostoja luettaessa käyttäjän tagit annetaan verrattaviksi metatietojen tageihin. GCS-moduuli sivuuttaa tämän parametrin, eikä parametri vaikuta sen toimintaan, mutta jos tulevaisuudessa halutaan kehittää sama toiminnallisuus GCS-moduuliin, se on mahdollista käyttäen samaa parametria.

Tiistai

Tämän päivän käytin onpremise-moduulin testaamiseen ja bugien korjaamiseen. Suoritin testausta käyttämällä Connect2-appsissa niitä käyttöliittymiä, joita käyttäjät siinä käyttäisivät, tallentamalla ja avaamalla logoja, dokumentteja, tai muita tiedostoja, joita Connect2-apps:lla on tarkoitus käsitellä. Tällainen testaus on manuaalista testaamista, joka saattaa viedä paljon aikaa. Monella tapaa olisi parempi vaihtoehto luoda automaattiset testit tekemään testejä aina kun jokin osa testattavaa koodia muuttuu.

GCS-moduulia varten ei ole automaattisia testejä, joten minulla ei ollut hyvää referenssiä automaattisten testien luomiseen onpremise-moduulia varten. Tämän lisäksi on seuraava ongelma: Jos GCS-moduulilla ja onpremise-moduulilla on omat automaattiset testit, vain toista voi testata testien ajon aikana, koska käytettävä moduuli valitaan dynaamisesti ympäristömuuttujien perusteella testien käynnistyessä. Tämän voisi ratkaista käynnistämällä erilliset automaattiset testit onpremise-moduulia varten, mutta se monimutkaistaisi niiden ajamista. Päädyimme kollegan kanssa johtopäätökseen, ettei onpremise-moduulille tehdä testejä aiemman perusteella ja onpremise-moduuli on pieni ja sitä ei todennäköisesti muuteta tulevaisuudessa.

Keskiviikko

Testaus jatkui vielä tänään ja totesin että onpremise-moduuli on valmis vertaisarviointiin. Livion-yrityksessä käytetään Bitbucket-nimistä verkkopalvelua, joka käyttää git:iä. Git on versionhallintaohjelma, jolla seurataan kansiorakenteiden ja tiedostojen sisällön muuttumista. Vertausta voidaan tehdä mihin tahansa aiempaan tiedostojen versioon nähden, sillä git tallentaa seurattujen tiedostojen historian (Git). BitBucketin käyttöliittymällä voin lähettää haluamilleni kollegoille koodin arviointipyynnön, minkä jälkeen kollegat tarkastelevat koodin, kommentoivat sitä, tekevät korjaus- tai muutosehdotuksia ja hyväksytät koodin mahdollisten muutosten jälkeen. Tämän jälkeen koodi on valmis liitettäväksi kehityshaaraan, josta se myöhemmin liitetään master-haaraan.

Git:ssä on käsite haaroista, joista yksi tunnetaan master-, tai nykyisin main-haarana. Haaroja käytetään erottelemaan ohjelmiston koodia kunkin käyttöympäristön tarpeiden perusteella. Esimerkiksi main-haaraa useimmiten käytetään sellaisen ohjelmistoversion ylläpitoa varten, joka on yrityksen tuote (Git). Livionilla oleellisimmat haarat ovat master-, development- ja ominaisuushaarat. Ominaisuushaaroja luodaan ja käytetään uusien ominaisuuksien kehittämiseen ja ne periytetään development-haarasta. Ominaisuushaaroista uusi tai muutettu koodi yhdistetään takaisin development-haaraan, joka taas yhdistetään master-haaraan, kun uusien tai muuttuneiden ominaisuuksien julkaiseminen nähdään ajankohtaiseksi tai tarpeelliseksi.

Palautteen saatuani suoritin muutamia korjauksia, joiden jälkeen liitin ominaisuushaaran development-haaraan, ja tämä tehtävä oli valmis.

Torstai

Connect2-backend- ja Connect2-apps-käyttäjillä on rooli, joka kuvaa käyttäjän merkitystä, ja sitä, mitä tämä pystyy tekemään järjestelmällä. Yksi tunnetuin rooli, jota suurimmassa osaa roolipohjaisia järjestelmiä käytetään, on admin-rooli, joka antaa käyttäjälle laajat käyttöoikeudet järjestelmään.

Livion-yrityksen Connect2-backend alkuperäisessä käyttöympäristössä roolit ja niihin liittyvät säännöt tallennetaan tietokantaan. Tämän vuoksi rooleja pystyy lisäämään, muokkaamaan ja poistamaan ilman muutoksia lähdekoodiin. Seuraava tehtäväni on tuottaa koodi, joka synkronisoi etänä oleviin tietokantojen roolit alkuperäisen käyttöympäristön tietokannan kanssa, käytännössä kopioi alkuperäiset roolit kohdetietokantaan.

Keskusteluissa kollegan kanssa päätimme, että synkronisointi tapahtuu säännöllisesti puolen tunnin välein, mitä varten löytyy aiemmin käytetty toimintapa, joka on Connect2-backendissä nimetty agenteiksi.

Agentilla tarkoitetaan erillistä koodikokonaisuutta, joka säännöllisin aikavälein käynnistetään ja joka suorittaa määritetyn tehtävän eristettynä vaikutuksen alaisesta kohdejärjestelmästä.

Agentin nimeksi tulee connect-onpremise-db-roles-sync-agent, mutta kutsun tässä tekstissä sitä sync-agentiksi. Agentti suorittaa tehtävänsä seuraavalla prosessilla: se ottaa yhteyden etärajapintaan, joka on yhteydessä roolien lähde tietokantaan, noutaa kaikki roolit rajapinnasta, ottaa yhteyden paikalliseen kohde tietokantaan, poistaa roolit sieltä ja tallentaa noudetut roolit sinne.

Viikkoanalyysi

Sain valmiiksi onpremise-moduulin, joka käsittelee tiedostoja kirjoittaen ja lukien niitä paikalliselta kiintolevyltä GCS-pilvipalvelun sijaan. Onpremise-moduulin tärkeänä erona GCS-moduulin on turvallisuustason huomattava nousu, sillä onpremise-moduulissa tiedostojen luku on rajoitettu käyttäjille, joilla on käyttöluvat tiedostojen metatietoihin, jotka on tallennettu tietokantaan. Tämän mahdollisti se, että onpremise-moduuli tallentaa tiedostojen metatiedot ja käyttöoikeuksiin liittyvät tagit tietokantaan ja tiedostojen noutamista varten rakennettiin rajapinta, joka noudon yhteydessä tarkistaa käyttäjän käyttäjäoikeudet.

Vertaispalaute on osoittautunut tärkeäksi koodin laadun suhteen. Kullakin koodaajalla on omia näkemyksiä ja lähestymistapoja ongelmiin ja erilaista osaamista ja kokemusta, minkä vuoksi on todennäköistä, että he huomaavat toisen koodista sellaisia kohtia, jotka voivat aiheuttaa ongelmia, tai tekemällä asioita eri tavalla lopputulos olisi parempi tyylin, tehokkuuden tai molempien kannalta. Versiohallintaan valmistettuja ohjelmia hyödyntämällä vertaisarviointi on nopeaa ja tehokasta, eikä se ole ainoa syy miksi niitä on suositeltavaa käyttää.

3.4 Viikko 4

Tiistai

Tämä päivä, kuten edellinen torstai, kului suurimmalta osalta taustalla olevan järjestelmän tutkimiseen ja siihen, miten agentin saa valmistettua Connect2-backendissä hyödyntäen valmiiksi asennettuja ja rakennettuja koodipaketteja. Useimmat paketit ovat kolmannen osapuolen valmistamia, kuten tietokantaan käyttöpaketti, ja muutama on kollegoideni tekemiä paketteja, joilla otetaan muun muassa yhteyttä Connect2-backend-rajapintaan.

Roolien noutamista varten tehtiin palvelutili, joka on yleinen käsite sellaiselle käyttäjätilille, joka on luotu järjestelmien automatisaatioita varten, eikä sillä ole ihmistä käyttäjänä (Service accounts). Tämä palvelutili luotiin sitä varten, että etänä olevaan rajapintaan voidaan kirjautua sisään, ja palvelutilillä on oikeudet noutaa kaikki roolit, jotka löytyvät rajapinnan tietokannasta.

Keskiviikko

Rakentaessani agenttia huomasin, ettei palvelutili saanut noudettua kaikkia rooleja, sillä roolien noutamista varten tehty koodi ei sisältänyt ominaisuutta, jolla kaikki roolit voitaisiin noutaa, sillä tietokannasta rooleja haettaessa ei voitu välttää suodattamista mikä rajasi tuloksia. Roolien noutamisen koodia muutettiin siten, että palvelutilille tehtiin poikkeustapaus, joka sallii palvelutilin noutaa kaikki roolit.

Torstai

Vanhojen roolien poiston ja noudettujen roolien tallentamisen välissä ohjelma kaatui, minkä seurauksesta kohdekanta jäi rooleista tyhjäksi. Tämän seurauksena järjestelmä saattaa kaatua ja aiheuttaa käyttökatkoksia, sillä roolit puuttuvat, joten päätin että roolien tallentamiseen täytyy olla jokin toinen keino. Tämän miettiminen vei paljon aikaa, ja lopulta kysyin apua kollegalta, mikä jälkikäteen ajatellen olisi pitänyt tehdä paljon aikaisemmin. Ongelma ratkaistiin transaktiolla; tietokantojen operaatioita suoritetaan jonossa, jossa kaikkien operaatioiden on onnistuttava, muussa tapauksessa mikään operaation aiheuttama muutos ei astu voimaan tietokannassa (What is a Transaction?). Uuden suunnitelman mukaisesti kohdetietokannan roolien muutokset tapahtuvat seuraavaa kaavaa noudattaen: poista kohdetietokannasta roolit, joita ei löydy noudettujen roolien joukosta, tallenna roolit, joita ei löydy kohdetietokannasta, sekä päällekirjoita roolit, jotka löytyvät sekä noudetuista että kohdetietokannan rooleista. Tällä

tavalla missään välissä tietokantaoperaatioita kaikki roolit eivät katoa, jos roolien synkronisointi epäonnistuu syystä riippumatta. Loppupäivä kului koodin testaukseen ja lähetin sen vertaisarviointia varten.

Perjantai

Ryhdyin tutkimaan seuraavan tehtävän vaatimuksia. Connect2-appsin eräässä käyttöliittymässä käyttäjä valitsee yhden tai useamman avaimen, joka varataan asiakkaalle, jonka tiedot täytetään lomakkeeseen varauksen tekemiseksi. Avaimia etsitään hakusanalla, joka voi viitata muun muassa avaimen nimeen, sijaintiin tai rakennukseen, jossa avainta voidaan käyttää. Kun käyttäjä täyttää asiakkaan tietojen lomakkeen ja tämän jälkeen muuttaa avaimien hakusanaa, lomake katoaa avaimien lataamisen ajaksi, minkä jälkeen se palaa tyhjänä, ja lomake täytyy täyttää uudelleen. Tehtäväni on muuttaa koodia siten, että asiakastieto lomake ei tyhjene avaimia ladattaessa.

Avaimien varaamisen koodi on monimutkaista itseni, sekä kollegoiden mielestä, joilta kysyin neuvoja tehtävän suorittamiseen, minkä vuoksi on vaikeaa selvittää mitä koodille olisi kannattavinta tehdä ongelman korjaamiseksi.

Tein roolien synkronointikoodiin vielä joitain koodin tyylimuutoksia paremman luettavuuden vuoksi. Vertaisarvioinnissa ei ilmennyt mitään korjattavaa, joten liitin ominaisuushaaran development-haaraan, ja tehtävä oli valmis.

Viikkoanalyysi

Tällä viikolla suorittamani tehtävän vaikeustaso oli hieman korkeampi, koska taustalla olevasta järjestelmästä kuului selvittää enemmän asioita ja tutustuin uusiin käsitteisiin. Uusia asioita minulle olivat kuinka käyttää tämän projektin tietokanta-ajurin transaktiotoimintoa ja miten palvelutilejä luodaan ja käytetään. Rajapinnan muokkauksen yhteydessä tutustuin rajapintamallin automaattiseen generointiin, agentteihin ja miten niitä luodaan Livion-yrityksen ohjelmistoympäristössä. Rajapintamalli on tiedosto, joka sisältää tiedon siitä mitä parametreja ja millaista dataa rajapinnasta saadaan. Sen tarkoituksena on helpottaa kehittäjien ohjelmistokehitystä antamalla kehittäjän tekstinkäsittelytyökalulle näytettävää tietoa, jottei kehittäjän tarvitse navigoida eri projektien välillä tutkimassa millaista parametrit ja data ovat muodoltansa.

3.5 Viikko 5

Maanantai

Edellisen viikon ratkaisuni lomakkeen tyhjenemiselle avaimen valinnan yhteydessä ei toiminut oletetusti jälkitarkastelussa, vaan se aiheutti uuden sivuvaikutuksen, jossa valitut avaimet hävisivät, minkä vuoksi muutos päätettiin ottaa takaisin, ja ryhdyin kehittämään toista ratkaisua.

Tiistai

Ongelmalle ei päivän mittaan löytynyt sopivaa ratkaisua, ja tehtiin päätös, että lomake rakennetaan myöhemmällä ajankohdalla uudestaan, koska sille on suunniteltu uutta rakennetta.

Keskiviikko

Connect2-appsissa tagin alle voidaan lisätä rakennuksia, joihin voidaan lisätä avaimia. Rakennuksia lisätään näkymän kautta, jossa kaikki aiemmin luodut rakennukset näkyvät. Kun uusi rakennus lisätään, näkymän kuuluisi hetken kuluttua päivittyä ja näyttää lisätty rakennus listalla, mutta päivittyminen ei toimi, vaan käyttäjän täytyy virkistää sivu manuaalisesti saadakseen ajantasainen tieto näkymään. Rakennuksien tiedot tallennetaan välimuistiin noutamisen yhteydessä, minkä jälkeen rakennuksia uudelleen noudettaessa niitä etsitään ensin välimuistista, jotta vältytään tekemästä rakennuksien noutoja verkon yli. Tähän tarkastukseen liittyvä systeemi ei ymmärrä tietojen muuttuneen, vaikka se itse on tehnyt muutoksen ja sen kuuluisi noutaa uusien rakennuksien tiedot.

Kollegoiden konsultoinnin jälkeen päädyttiin ratkaisuun, joka se oli alun perin; navigoidaan luodun rakennuksen yksityiskohtanäkymään. Yksityiskohtanäkymässä rakennusta voi muokata. Rakennuksen muokkauksessa näkymässä on enemmän säädettävää tietoja kuin luontinäkymässä, joten on luonnollista, että käyttäjä ohjataan rakennustietojen sivulle automaattisesti rakennuksen luonnin jälkeen. Kun käyttäjä navigoi takaisin rakennuslistanäkymään, näkymä noutaa rakennuksien tiedon verkon yli, koska näkymän vaihtumisen yhteydessä edellisen näkymän välimuisti poistetaan.

Muutos oli pieni ja yksinkertainen implementoida, mistä syystä se arvioitiin nopeasti ja sain sen liitettyä development-haaraan.

Torstai

Avaimen saadakseen asiakkaan täytyy tehdä sopimus sen käyttöönotosta. Sopimukseen merkitään muun muassa asiakkaan ja avaimien tiedot, mutta myös avaimiin kohdistuneet tapahtumat sopimuksen aikana. Sopimusta voidaan muokata sen luomisen jälkeen, missä ilmeni ongelma: avaimiin merkityt tapahtumat sopimuksessa häviävät sopimuksen muokkauksen jälkeen, mikä vääristää sopimuksen tilan. Tähän syynä on, ettei sopimuksen avaimien tapahtumia haeta, kun sopimusta muokataan, minkä seurauksena tapahtumia ei lähetetä päivityksen yhteydessä tallennettavaksi.

Ongelmaan on kaksi korjausmahdollisuutta: hakea muokattavan sopimuksen avaimien tapahtumat, ja lähettää ne tallennuksen yhteydessä, tai palvelimen puolella hakea tietokannasta sopimuksen avaimien tapahtumat ja liittää ne muokattuun tietoon. Aiemmassa korjauksessa on vaikeutena, että muutoksia täytyy tehdä sekä käyttöliittymään että palvelimeen.

Perjantai

Ongelman korjaamiseen mietittiin eri mahdollisuuksia kollegoiden kanssa. Tapahtumien noudon lisäys vaatisi muutoksia sekä palvelimeen että käyttöliittymään. Lisäksi samaa sopimuksen tiedon noutamisen kyselyä käytetään muuallakin kuin muokkauksen yhteydessä, ja tarpeettoman tiedon noutamista pyritään välttämään. Parempana vaihtoehtona pidettiin muutosta vain palvelimeen, sillä muutokset rajoittuvat sopimuksen muokkauksen logiikkaan.

Tein muutoksen jossa muokkaamattoman sopimuksen avaimien tapahtumat liitetään tallennettavan sopimuksen avaimien tapahtumiin, minkä seurauksena sopimuksen tila muuttui oletetusti, ja lähetin muutoksen arvioitavaksi. Loppupäivä kului ongelman tutkimiseen mikä mahdollisesti liittyy tämän päivän aikana korjattuun ongelmaan.

Viikkoanalyysi

Joskus ongelmiin ei löydy sopivaa ratkaisua eri syitten takia. Tämän viikon tapauksessa koodi, jota yritin muuttaa, sisälsi sivuvaikutuksia, joista haluttiin eroon, mutta näitä sivuvaikutuksia korjattaessa saatiin aikaan uusia sivuvaikutuksia, jotka olivat yhtä ongelmallisia kuin alkuperäinen ongelma. Toisinaan ongelmat ovat pieniä ja yksinkertaisia ratkaista, mutta niille sopivan ratkaisun löytäminen voi viedä paljon aikaa, sillä mahdollisuuksia on monia, ja niistä voidaan valita vain yksi.

3.6 Viikko 6

Maanantai

Avaimen luovutuksen yhteydessä luodaan avaimen luovutuksesta sopimus, joka sisältää henkilön tietoja, jolle avain luovutetaan. Sopimuksen luomisesta lähetetään vahvistus sähköpostitse asiakkaalle, jolle avain luovutetaan sopimuksen mukaisesti. Joskus sähköpostien mukana tulleet kuvat eivät aukea, ja tehtäväni on tutkia mistä ongelma johtuu. Todennäköisin syy ongelmaan löytynee lähiaikoina muutetusta tiedostojen käsittelyjärjestelmästä, jonka kehittämisessä olen ollut pääosassa.

Aloitin tutkimalla mitä tiedostoja käsittelevä palvelin tekee, kun kuvatiedostoja yritetään noutaa. Tässä ilmeni, ettei kyselyn tekijällä ollut oikeuksia lukea tiedostoja, kun kysely tuli muualta kuin omista applikaatiostamme, tässä tapauksessa käyttäjän sähköpostipalvelun verkkosivulta. Kuvat eivät olleet julkisia; niitä ei voi noutaa palvelimelta ilman riittäviä käyttöoikeuksia.

Tiistai

Ryhdyin kehittämään erilaisia ratkaisuvaihtoehtoja siihen, miten kuvia saadaan näkymään myös sellaisille käyttäjille, joiden on tarkoitus nähdä ne, mutta vähemmillä käyttöoikeuksilla. Yksi ratkaisu olisi muuttaa kuvien linkkiä siten että niihin lisätään kertakäyttöinen koodi, joka sallii tiedoston lukemisen tiedostoja käsittelevällä palvelimella. Tällaista tapaa käytetään jo muidenkin tyyppisten tiedostojen noutamiseen ulkoisista applikaatioista. Toinen ratkaisu kollegan kanssa pohtimisen jälkeen oli, että lisätään tiedostoihin merkintä, joka ilmoittaa onko tiedosto julkinen vai ei, mikä oli mielestäni nerokasta. Esitin ehdotuksen muille kollegoille ja se sai huomattavaa kannatusta, minkä vuoksi aloitin ratkaisun kehittämisen.

Aikaisemmin tiedostoja pystyi lukemaan kuka tahansa mistä tahansa, kun vain tiesi linkin tiedostoon. Järjestelmän muuttumisen myötä tiedostoja voi lukea, jos lukijalla on riittävät käyttöoikeudet. Järjestelmää kehitettäessä olisi pitänyt huomioida paremmin käyttäjien erilaisia tarpeita. Käyttäjiä on tähän tapaukseen liittyen kahdentyyppistä, ulkoisia ja sisäisiä. Vain sisäisillä käyttäjillä oli kyky lukea tiedostoja uudistetun tiedostojen käsittely järjestelmän kautta ja ulkoisten käyttäjien tarpeet olivat unohtuneet.

Keskiviikko

Lisäsin tiedostojen julkisen flagin käytön koodiin, jotta järjestelmän ulkoiset käyttäjät voivat lukea tiedostoja, jos heillä on linkki tiedostoon. Tiedostot, jotka on luotu ennen muutosta vaativat kyseisen merkinnän, jotta ne toimivat oletetusti. Puuttuvat merkinnät lisätään järjestelmän päivityksen jälkeen skriptillä, joka tutkii kaikki tallennetut tiedostot ja lisää merkinnän tarpeen mukaan. Tein verkkoreittiin muutoksen, mitä kautta tiedostot haetaan; mikäli tiedosto on julkinen, sen voi noutaa kuka hyvänsä ilman enempiä käyttölupia.

Torstai

Testasin eilen kirjoittamiani muutoksia julkisen merkinnän eri tiloissa, ja kaikki näytti toimivan oletetusti. Koska tällä muutostyöllä oli kiire, se liitettiin välittömästi tuotantoympäristöön minimaalisten testauksien jälkeen. Vielä täytyi tarkastaa kaikki tiedostot sen varalta, tarvitseeko niihin lisätä positiivinen julkinen merkintä. Kirjoitin skriptin, joka päättelee merkinnän tarpeellisuuden tiedoston reitin mukaan ja lisää merkinnän tietokannan dokumentille. Tämä oli mahdollista koska julkiset tiedostot ovat tietyssä kansiossa reittinsä mukaan.

Perjantai

Aloitin seuraavan tehtävän selvittelyä. Kyseessä on vanhan moduulin päivittäminen nykyaikaan ja uuden ominaisuuden lisääminen siihen. Koodin toimintaan liittymättömiä muutoksen kohteita ovat moduulin tiedostojen tyypin muuttaminen .js-tiedostoista .ts-tiedostoiksi, sekä tarpeelliset koodimuutokset tiedostotyypin muutosten takia, että moduuli edelleen toimisi. Varsinaisia toiminnallisuuden muutoksia ovat uuden toiminnon lisäys, ulkoisen rajapinnan käyttötavan uudistaminen, sekä virhetilanteista syntyvien viestien selkeyttäminen. Tämän päivän ajan tutkin moduulin sisältöä ja suunnittelin missä järjestyksessä tekisin työn vaiheet.

Viikkoanalyysi

Kun tiedostojen lukemiseen lisättiin uusi turvallisuuden kerros, en kiinnittänyt suunnittelussa huomiota jälkikäteen ajatellen itsestään selvään tarpeeseen, että joidenkin tiedostojen täytyy olla julkisia, sillä niitä täytyy voida jakaa sähköposteissa ja viesteissä. Ongelma ei aiheuttanut suurempaa haittaa kuin joidenkin kuvatiedostojen puuttumista eri viesteistä, mikä lähinnä ihmetyttää asiakkaita. On kuitenkin mahdollista, että asiakkaat alkavat epäilemään viestien aitoutta joidenkin kuvien puuttuessa, kuten logo viestin ylälaidassa, jonka olemassaoloon on totuttu.

3.7 Viikko 7

Maanantai

Tuote nimeltään RoomRobot on IOT-laite, jonka päätarkoitus on toimia etäverkkoasemana heikoillakin kuuluvuusalueilla. Laitteessa on verkkoyhteystilaus, jossa asiakkaille on 20 mbps:n verkkoyhteys. Uutena ominaisuutena tarjoamaan asiakkaille vaihtoehtoisesti 50 mbps:n verkkoyhteyttä. Muutoksia verkkoyhteys tilauksiin suorittaa kollegani asiakkaiden pyynnöstä. Asiakkaiden määrä on ajan myötä kasvanut, mikä työllistää kollegaani hiljalleen enemmän. Tavoitteenani on muuttaa olemassa olevaa työkalua, mikä nopeuttaa muutostöiden tekemistä laitteiden verkkoyhteys tilauksiin. Mahdollisiin verkkoyhteys tilauksen muutoksiin on aiemmin kuulunut vain verkkoyhteyden avaaminen ja sulkeminen. Uusi ominaisuus on verkkoyhteyden nopeuden muuttaminen kahden vaihtoehdon välillä ja lisäksi tavoitteisiini kuuluu parantaa olemassa olevaa koodia.

Ryhdyin tutkimaan koodikantaa ja ulkoista rajapintaa, jota käytämme verkkoyhteyden muutoksien tekemiseen.

Tiistai

Ensimmäisenä muutoksena päätin muuttaa koodikannan tiedostojen tyypit .js-tiedostoista .ts-tiedostoiksi. Tiedoston tyypin muuttaminen on suoraviivaista, tiedoston nimeä muutetaan siten että pääte muutetaan yhdestä toiseen, tässä tapauksessa .js-päätteestä .ts-päätteeksi. Tämän seurauksena tiedostojen sisältämän koodin toiminta ei muutu mitenkään, mutta se antaa kehittäjälle laajemman valikoiman työkaluja koodin kehittämiseen. On kuitenkin joitain koodimuutoksia, joita kehittäjän täytyy tehdä tiedostopäätteen muuttamisen jälkeen, kuten tyyppien lisäys muuttujiin, sekä koodivirheiden korjaaminen, jotka havaitaan .ts-tiedostojen koneellisen tulkitsijan takia.

Keskiviikko

Löysin huomattavan määrän tarpeita pienille muutoksille, jotka yksinkertaistavat moduulin toimintaa. Koodi ei tarvitse muutoksia, jotka nopeuttaisivat sen toimintaa, sillä sitä käytetään suhteellisen harvoin ja sen tehtävä on pieni. Sen sijaan keskityin koodin luettavuuteen, mikä helpottaa moduulin kehittämistä tulevaisuudessa. Oli kuitenkin yksi suorituskykyyn liittyvä muutos, jonka halusin tehdä. Ulkoisesta rajapinnasta noudetaan kaikki verkkoyhteystilaukset, ja niistä paikallisesti etsitään yksi, jota halutaan muuttaa. Järkevämpää on noutaa rajapinnasta vain kyseinen tilaus, koska tuhansien muiden hakeminen

aiheuttaa tarpeettomia kuluja verkon käytön, toiminnon keston, sekä koodia suorittavan laitteen muistin käytön vuoksi.

Torstai

Tänään jatkoin muutosten tekemistä, ja aloitin testaamaan tekemiäni muutoksia. Löysin pieniä helposti korjattavia ongelmia, ja kehitystyö sujui hyvin. Rajapinta, jota käytämme muuttamaan laitteiden verkkoyhteyksiä toimi oletetusti; pystyin avaamaan ja sulkemaan verkkoyhteyksiä, sekä vaihtamaan verkkoyhteyden nopeuden. Poikkeuksellisesti, kun yritin vaihtaa nopeuden takaisin yhden muutoksen jälkeen, rajapinta palautti virheviestin, eivätkä muutokset astuneet voimaan. Samassa yhteydessä huomasin, että numerosarja, jota käytetään verkkoyhteys-tilauksen noutamiseen, oli muuttunut virheelliseksi ensimmäisen onnistuneen muutoksen jälkeen. Usean verkkoyhteys tilauksen numerosarjan perään oli ilmestynyt ylimääräinen välilyönti, joka esti tilauksen noutamisen rajapinnasta numerosarjan avulla. Yritin selvittää näitä ongelmia, mutta tuloksetta yksin ja kollegoiden kanssa. Päivän päätteeksi otimme yhtyettä rajapinnan kehittäjiin sähköpostitse.

Perjantai

Sain aamupäivän aikana yhteyden sähköpostitse ulkoisen rajapinnan tukihenkilöön, joka pyrki löytämään ongelmaan ratkaisua prosessin, sekä laatimani ongelman kuvauksen pohjalta. Sain päivän aikana erilaisia ehdotuksia siitä, miten rajapintaa voidaan käyttää verkkoyhteyden nopeuden muuttamiseen. Epäkohtana suurimmassa osassa ratkaisuja oli, etteivät ne noudattaneet virallista dokumentaatioita rajapinnan käytöstä. Päivä kului eri ratkaisuja kokeilemalla, joista yksikään ei toiminut oletetulla tavalla.

Viikkoanalyysi

Monimutkaisten käsitteiden selittäminen ei ole helppoa, sillä siinä vaaditaan kaksi taitavaa osapuolta, selittäjä ja ymmärtäjä. Molempien osapuolien on oltava riittävän korkealla ymmärryksen tasolla aiheesta, jotta tiedon jakaminen onnistuu. Tällä viikolla minulle kävi selkeästi ilmi, ettei minun ja ulkoisen rajapinnan tukihenkilön välinen kommunikaatio tuottanut tulosta, sillä pyrittiin saamaan rikkinäinen toiminto toimimaan, eikä itse ongelman korjausta saatu vireille.

3.8 Viikko 8

Maanantai

Edellisen viikon muutostyöt ovat jääneet takasijalle siksi aikaa, kunnes tukihenkilö, johon olin yhteydessä saa ratkaistua ongelmat, joiden vuoksi emme voi käyttää rajapintaa luotettavasti. Kuvasin ongelman vielä kertaalleen sähköpostitse tukihenkilölle, ja pyysin kollegaani myös olemaan yhteydessä rajapinnan tukitiimiin, koska hän oli ennenkin ollut tekemisissä heidän kanssaan.

Tänään tavoitteena oli korjata ongelma puhelinnumeroihin liittyen. Käyttäjien on pystyttävä lisäämään puhelinnumeroita järjestelmään haluamassaan muodossa ja useimmat käyttäjät jättävät niihin välilyöntejä, jotta puhelinnumero olisi helpompi lukea. Puhelinnumeroa käytetään lähettämään viestejä käyttäjälle eri asioihin liittyen, kuten sopimuksien allekirjoittamista tai PIN-koodin saamista varten. Palvelin lähettää viestejä ulkoisen viestintäpalvelun rajapinnan kautta, jossa ongelma ilmeni. Kyseinen rajapinta ei osaa käsitellä puhelinnumeroita, joissa on välilyöntejä. Päätin korjata ongelman poistamalla välilyönnit numeroista, kun ne lähetetään viestintärajapinnalle, jotta numerot säilyisivät käyttäjien haluamassa muodossa välilyönteineen.

Tiistai

Kaikki tiedostot, joita järjestelmämme käsittelee tallentuvat GCS-moduulin avulla GCS-pilvipalveluun. Käyttäjät pystyvät noutamaan tiedostoja suoraan tästä GCS-pilvipalvelusta, mikä on turvallisuusriski, sillä kuka tahansa, joka tietää linkin tiedostoihin voi noutaa niitä. Monet tiedostot sisältävät asiakkaiden henkilökohtaisia tietoja, kuten nimiä, puhelinnumeroita, osoitteita, ym., jotka eivät saa päätyä asiaan kuulumattomien henkilöiden käsiin. Nyt tehtäväni on muokata moduulia siten, että se ei anna käyttäjille suoraa linkkiä pilvipalvelun tiedostoon, vaan itse generoidun linkin omaan järjestelmäämme, joka hakee tiedostoja pilvestä, ja lähettää ne palvelimen kautta käyttäjälle. Tarkoitus on pitää tiedoston lähde salattuna ja estää tiedostojen noutaminen henkilöiltä, joilla ei ole järjestelmän valvomaa käyttöoikeutta niihin. Aiemmin kehittämäni onpremise-moduuli, joka pohjautuu GCS-moduuliin, toimii tällä tavalla, siksi turvallisuusmuutosten teettäminen vaikuttaa helpolta tehtävältä.

Ensisijaisesti paransin GCS-moduulin luettavuutta tekemällä koodiin tyylimuutoksia. Seuraavaksi selvitin mitä muutoksia moduuli vaatii, jotta linkit päätyvät vain järjestelmän käytettäviksi, ja miten lähetämme tiedostot niitä pyytäville, joilla on tarpeelliset käyttöoikeudet. Nämä ongelmat ovat jo ratkaistuja onpremise-moduulissa ja sen käyttökohteissa.

Keskiviikko

Koodissa tiedostojen tallentaminen pilveen ei muuttunut mitenkään, mutta uutta tietoa tallennetaan omaan tietokantaamme, ja käyttäjälle palautetaan eri tietoja tiedoston saamiseksi. Muutokset noudattavat onpremise-moduulin tapaa käsitellä tietoja; tiedosto kirjoitetaan haluttuun kohteeseen, ja tiedoston linkki, lukijan tarvitsemat oikeudet, tiedoston tyyppi ja reitti tallennetaan tietokantaan. Tietokannan dokumentille luodaan ainutlaatuinen linkki, jolla tiedosto pystytään noutamaan, ja tämä linkki palautetaan käyttäjälle. Linkin ominaisuutena on, että vain sen omistaja pystyy noutamaan tiedoston sen avulla, ellei tiedosto ole merkitty julkiseksi tiedostoksi, jonka lukemiseen ei tarvita käyttöoikeuksia. Tiedoston tallentamisen lisäksi muutin tiedostojen poistoa, jossa uutena tarpeena on poistaa tiedostoon viittaava dokumentti tietokannasta.

Perjantai

Eilen sain viimeisteltyä tarpeelliset metodit tiedostojen käsittelyyn: tallentaminen, lukeminen ja poistaminen. Käyttäjällä on jo ennalta keinot nähdä, tallentaa, ja poistaa tiedostoja, mutta ei keinoa avata niitä. Tiedostojen avaamiseen tarvitaan uusi verkkoreitti palvelimelle, jonka kautta muut toiminnot ovat käytettävissä. Reitti on muotoa https://{domain}.fi/file/{uuid}, jossa domain on palvelimen osoite, /file/ on reitti tiedostojen noutamiseen, ja uuid viittaa noudettavaan tiedostoon ainutlaatuisella tunnisteella. Reitti tarkastaa kyseylä tekevän käyttäjän tiedoista onko tällä oikeuksia tiedoston lukemiseen, minkä jälkeen tiedoston data ohjataan käyttäjälle. Vaikeinta reitin valmistamisessa oli virhetilanteiden käsittely, sillä mahdollisia virheitä oli monta, käyttäjä ei ole kirjautunut järjestelmään, käyttäjällä on puutteelliset lukuoikeudet, tiedostoa dataa ei löydy pilvestä, tiedoston tietoja ei löydy tietokannasta, sekä muut määrittämättömät virheet.

Viikkoanalyysi

3.9 Viikko 9

Maanantai

Aamupäivästä testasin edellisellä viikolla tekemiäni muutoksia tiedostojen lukemiseen ja käsittelyyn pilvessä. Todettuani muutosten toimivan, lähetin koodin kollegoiden tutkittavaksi ja jäin odottamaan palautetta.

Iltapäivällä aloitin uuden tehtävän, jossa täytyi selvittää ongelma avaimien luovutussopimuksiin liittyen. Sopimuksiin liitettyjen avaimien tapahtumatiedot häviävät sopimuksista, kun sopimusta päivitetään. Tämä aiheuttaa ongelmia käyttäjille, koska he eivät näe mitä avaimille on viimeksi tehty, kunnes avaimen tila muuttuu seuraavan kerran sopimuksen päivittämisen jälkeen. Tutkin ensimmäisenä käyttöliittymää, jonka kautta sopimuksia voidaan päivittää, ja siitä huomasin, että avaimiin liittyviä tapahtumia ei lähetetä palvelimelle, kun päivitys tehdään. Päivittäminen on rakennettu siten, että vanhaa tietoa ylitse kirjoitetaan uudella tiedolla. Avaimien tiedot lähetetään sopimuksen päivityksen yhteydessä, mutta tiedoista puuttuvat avaimien viimeisimmät tapahtumat, minkä takia ne katoavat tallennuksen tapahtuessa.

Tiistai

Tänään kehitin kaksi eri ratkaisua siihen, miten sopimuksien avaimien käyttötiedot eivät häviä, kun sopimusta päivitetään. Ensimmäinen ratkaisu on sopimuksen päivittämisprosessin alussa noutaa sopimuksen avaimien perustietojen lisäksi niiden käyttötiedot, ja liittää päivitettävien avaimien tietoihin. Ratkaisu tarvitsee paljon uutta koodia, sillä huomioon pitää ottaa tilanteet, kun avaimia halutaan lisätä tai poistaa. Toinen ratkaisu oli tehdä muutoksia palvelimen logiikkaan, missä sopimuksien avaimien tietoihin liitetään sopimuksen aikaisemmasta tilasta avaimien käyttötiedot. Ratkaisun tekeminen palvelimella osoittautui paremmaksi vaihtoehdoksi, koska huomioon ei tarvitse ottaa avaimien lisäystä tai poistoa, sillä ne tapahtuvat käyttöliittymässä.

Keskiviikko

Tänään suoritin eilen suunnitellun ratkaisun implementoinnin, sekä tarkastin muuttuneen koodin toiminnallisuuden; pysyvätkö avaimien käyttötiedot ennallaan sopimuksien muokkaamisen jälkeen. Ratkaisu oli yksinkertainen, testaus ei vienyt kauan aikaa, ja tulos oli positiivinen, eli ratkaisu toimi. Ratkaisu julkaistiin ja siirryin seuraavaan tehtävään.

Tavanomaisten avaimien säilömisen lisäksi avainlaitteissa voidaan säilöä ohjelmoitavia avaimia. Ohjelmoitavien avaimien etuna on, että yhtä avainta voidaan käyttää missä tahansa lukossa, jos avain on ohjelmoitu avaamaan avattava lukko. Avainlaatikoihin on asennettavissa moduuli, joka kykenee suorittamaan avaimien ohjelmointia, kun asiakas tarvitsee avainta. Järjestelmän käyttöä varten tarvitaan erillinen käyttäjä rajapintaan, jonka kautta ohjelmoitavia avaimia voidaan käsitellä. Tarpeena on luoda kysely, joka noutaa rajapinnasta käyttäjän oikeudet eri toimenpiteisiin ohjelmoitavien avainten järjestelmässä, ja sieventää oikeudet määritellyiksi nimikkeiksi. Kollegani auttoi minut alulle tehtävässä näyttämällä, miten rajapintaa käytetään.

Torstai

Tänään tavoitteena oli tehdä rajapintaan kysely, joka hakee käyttäjän oikeudet ohjelmoitavien avaimien järjestelmään, sekä kehittää logiikka, joka sieventää käyttöoikeudet nimikkeiksi, joista käyttäjä näkee silmäyksellä mitä ohjelmoitavilla avaimilla voi tehdä. Käyttöoikeudet ovat lupamaskiksi nimitetty bittijono, jossa kukin bitti merkitsee eri käyttöoikeutta. Esimerkiksi ensimmäinen bitti antaa oikeuden lukea avaimien tietoja, toinen bitti antaa oikeuden muokata niitä, jne. Yksinkertaisin määritellyistä nimikkeistä tarvitsee neljän tietyn bitin lupamaskista olevan päällä. Kehitin erilaisia ratkaisuja, miten nimikkeiden vaatimat bitit voidaan tarkistaa lupamaskista, ja niistä yksinkertaisin oli käyttää bittioperaattoreita vertaamalla lupamaskia toiseen maskiin, joka sisältää vaaditut oikeudet. Ratkaisun suunnittelussa meni lähes koko työpäivä, ja sen koodaamiseen alle viisi minuuttia.

Perjantai

Koodia testannut kollegani huomasi, että vaikka rajapinnan käyttäjällä on kaikki käyttöluvat, meidän palvelimellamme käyttäjän luvat ovat puutteelliset. Sain selvitettyä, ettei palvelimemme käyttämä NodeJS tue yli 32-bittisiä numeroita, eli kolmaskymmeneskolmas bitti ja siitä suuremmat niin sanotusti vuotavat ylitse ja tekevät satunnaisen oloisesti muutoksia bittimaskin muihin bitteihin. Tilanteen ratkaisuun käytin erityistä numeroprimitiiviä, joka lisättiin NodeJS-ympäristöön vuonna 2020, BigIntiä. BigInt pystyy käsittelemään äärettömän suuria lukuja tyypillisien numero-operaattorien kanssa, kuten lisäys-, vähennys-, kerto-, jako- ja jakojäännösoperaattorit, sekä joitain bittioperaattoreita. Sen haittana on suorituskykyvaje verrattuna tavalliseen numeroprimitiiviin, sekä se, etteivät jotkin numero operaattorit toimi sen kanssa. Tähän tarpeeseen BigInt-primitiivi on kuitenkin riittävä, sillä kyseessä ovat rajallisen suuret bittimaskit, ja niille suoritetaan vain kaksi operaatiota. Ratkaisun käyttöönotto vaati muutoksia rajapintakyselyn vastauksen vastaanottoon, jossa tekstinä tullut bittimaski muutetaan numeroksi. Räätälöin vastaanottimen siten että bittimaski muutetaan BigIntiksi tavallisen numeron sijaan.

Viikkoanalyysi

Alkuviikon ongelman pohdinta ja ratkaisujen kehittäminen kesti kauan koodin kirjoittamiseen nähden, vaikka lopullinen ratkaisu oli pieni ja yksinkertainen. Ongelmaan löytyi mainitsemieni kahden ratkaisun lisäksi monta muuta ratkaisua, joita kutakin kehitin parhaaksi mahdolliseksi ratkaisuksi ja niistä valitsin yhden. Viikon jälkimmäisestä tehtävästä ilmeni yllättävä ongelma, joka pohjautuu ulkoisen rajapinnan ja sitä hyödyntävän palvelimen eroavaisuuksiin. Tässä tapauksessa rajapinta pystyy käsittelemään suurempia numeroita kuin sen hyödyntäjä. En ollut ennen kohdannut vastaavaa ongelmaa, siksi oli opeteltava BigInt-primitiivi ja sen käyttö.

Ohjelmointi ei ole koskaan pelkästään koodin tuottamista, vaan se on loogisten ongelmien tutkintaa, joihin on aina useita ratkaisuja. Harvoin yksi ratkaisu on parempi kuin toinen. Sen sijaan kussakin ratkaisussa on hyviä ja huonoja puolia. Ratkaisuista täytyy valita se, joka sopii parhaiten koodin tarpeisiin huomioon ottaen ratkaisujen edut ja puutteet.

4 JOHTOPÄÄTÖKSET

Osuuteni Connect2-projektissa on ollut kehittää uusia ja uudistaa vanhentuneita ominaisuuksia, sekä korjata virheitä. Projektiin liittyessäni koin vaikeaksi suorittaa kutakin näistä tehtävistä, koska en tiennyt paljoa projektin rakenteesta, ja siksi oli vaikeaa löytää koodista niitä paikkoja joihin lisäyksen tai muutoksen tekeminen oli tarpeellista. Tämä on helpottunut ajan myötä, kun olen tehnyt töitä useamman projektin osa-alueen parissa. Kaikkein tärkeimmäksi eduksi laajassa projektissa työskennellessäni huomasin kollegat, ja sitä myötä kommunikointitaidon, tarkalleen ottaen taito ymmärtää monimutkaisia käsitteitä ja selityksiä käsitteistä, sekä pystyä tuottamaan yksinkertainen selitys monimutkaisesta asiasta. Monesti tehtävän täyttäminen on ollut nopeampaa ja helpompaa kun on voinut turvautua kollegoihin ja heidän osaamiseensa ja taitoihinsa, tai pelkästään siihen, että on joku, joka kuuntelee mitä ongelmia kehitystyössä on kohdattu.

Ensimmäinen tehtäväni projektissa oli valmistaa järjestelmä, joka tallentaa tiedostoja paikalliselle asemalle. Ominaisuuden kehittämistä varten kirjoitin alustavan suunnitelman, johon listasin mitä järjestelmän kuuluu pystyä tehdä, ja kehittämisen yhteydessä laajensin listaa siten että siitä ilmeni, miten tavoitteita on saavutettu ja voidaan saavuttaa. Työskentelyni tapoja ja tottumisia tutkiessani huomasin, että tällaisten suunnitelmien kirjoittaminen jää pois lähes aina, sillä se ei ole tullut osaksi työskentelyni rutiiniani. Lisäksi yritän saada kaikki tehtävääni liittyvät asiat tehtyä yhdellä kertaa, mikä on johtanut tärkeiden ajatusten kadottamiseen usean päivän mittaisten tehtävien aikana. Olisi järkevää saada suunnitelman kirjoittaminen osaksi työskentelyrutiiniani ja kirjoittaa mitä työllä tavoitellaan, miten tavoitteeseen päästään, sekä töiden aikana ilmenevät ideat. Tämä organisoisi ajatuksiani, tärkeät tai muuten hyvät ajatukset olisivat edelleen tallessa ja työn tekö pysyisi johdonmukaisempana. Myöhemmin tämä suunnitelma muuttuu dokumentaatioksi, koska se sisältää dokumentaatiolle kaiken olennaisen, toiminnan, toiminnan oletetun tuloksen ja jatkokehitysideat.

Ohjelmoinnin alalla joka päivä opitaan jotain uutta. Ala muuttuu jatkuvasti, uusia teknologioita kehitetään, joka projekti on erilainen, tehtävillä on uusia ennen kohtaamattomia vaatimuksia ja uusia ennennäkemättömiä ongelmia ilmenee vanhassa koodissa ja uutta koodia kehittäessä. Kun työntekijä aloittaa hänelle uudessa projektissa työskentelyn, alussa on vaikeaa löytää ne osat koodia, jonne halutaan lisätä tai muokata jotain. Projektikohtainen oppiminen on tärkeää kehitystyön kannalta, sillä mitä paremmin tuntee projektin, sitä nopeampi sitä on kehittää, etenkin jos jotain aiemmin kehitettyä täytyy muuttaa. Koin itsevarmuutta projektin rakenteen tuntemuksesta puolen vuoden jälkeen, mistä eteenpäin työn teko on tuntunut tehokkaimmalta. Olen ollut kauan tietoinen Connect2-projektissa käytettävän

NodeJS-ympäristön numeroiden kolmenkymmenkahden bitin rajoitteesta, mutta rajoitus ei ole aiemmin tullut vastaan missään projektissa työskennellessä. Tässä projektissa rajoitus on konkreettisesti ollut esteenä, joten siihen täytyi keksiä ratkaisu, joka oli käyttää BigInt-primitiiviä. BigInt-primitiivillä on merkittäviä performanssi- ja käytettävyysrajoitteita, minkä vuoksi se ei sovi kaikkiin tilanteisiin, mutta tämän projektin tarpeeseen se sopi täydellisesti.

Merkittäväksi haasteeksi olen aina nähnyt työskentelyn toisten kanssa. Etenkin silloin kun on tarpeellista selittää vaikeita käsitteitä tai koodin toimintaa, haastavinta on ymmärtää ja saada toinen ymmärtämään kaikki tarpeellinen. Ymmärrettävyyden vaikeus korostuu etenkin silloin kun vastapuolella on henkilö, jolla on vähän tai ei ollenkaan osaamista ohjelmoinnin teknisestä puolesta, kuten suurin osa asiakkaista. Tässä projektissa esille otettava esimerkki on tapaus, jossa vastapuoli ei koskaan ymmärtänyt esillä olevaa ongelmaa täysin. RoomRobotin kehitystyöhön liittyen tarpeellinen rajapinta sisälsi vian, jonka vuoksi rajapinta ei toiminut oikein. Uskon ettei rajapinnan teknisen tuen työntekijä, joka päätyi vastaamaan tapauksesta, koskaan ymmärtänyt täysin ongelmaa, vaikka selitin sen moneen kertaan eri tavoin sähköpostitse. Tätä tekstiä kirjoittaessani ongelma on ratkennut, mutta se vaati yhteydenottoa sekä omani että rajapinnan yrityksen johtoportaisiin, ja kokonaisuudessaan yli kuusi kuukautta aikaa.

Koen kehittymisen tarvetta itseni ilmaisussa, jotta työnteko toisten kanssa olisi helpompaa ja vähemmän raskasta. Uskon että minun tarvitsee käyttää enemmän aikaa valmistautumisessa esittämään käsitteitä, jotta osaisin selittää ne selkeästi. Monesti valmistautumiseen ei kuitenkaan ole aikaa, sillä keskustelut voivat alkaa yllättäen videopuhelun muodossa heti kun ongelmia otetaan esille.

LÄHTEET

Linkedin. Livion Oy. Saatavissa:
https://www.linkedin.com/company/livion-oy Viitattu 12.06.2022.

Monorepo Tools. Saatavissa:
https://monorepo.tools Viitattu 12.06.2022

Google Cloud Storage. Saatavissa:
https://cloud.google.com/storage#section-1 Viitattu: 23.06.2022

Telecommunication Standardization Sector of ITU. 2014. ITU-T Recommendations. Saatavissa:
https://handle.itu.int/11.1002/1000/11746 Viitattu: 9.05.2022

Git. Saatavissa:
https://git-scm.com Viitattu: 27.5.2022

What is a Transaction? Microsoft. Saatavissa:
https://learn.microsoft.com/en-gb/windows/win32/ktm/what-is-a-transaction?redirectedfrom=MSDN
Viitattu: 7.10.2022

Service accounts. Microsoft. Saatavissa:
https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-service-accounts
Viitattu 7.10.2022