VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jonna Jaakkola

VERSIONHALLINTA PEREHDYTYKSEN TYÖVÄLINEENÄ

Case: Oiwa Solutions Oy, Opintopolku 2.0

Liiketalous
2020

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Jonna Jaakkola
Opinnäytetyön nimi	Versionhallinta perehdytyksen työvälineenä Case: Oiwa Solutions Oy, Opintopolku 2.0
Vuosi	2020
Kieli	suomi
Sivumäärä	49 +1 liitettä
Ohjaaja	Raija Tuomaala

Tämän opinnäyteytyön toimeksiantajana toimii Oiwa Solutions Oy. Opinnäytetyön tavoitteena oli päivittää toimeksiantoyritykselle aiemmin luotua opintopolkua vastaamaan enemmän toimeksiantajan tarpeita. Tämän opinnäytetyön käytännön osuudessa käydään läpi opintopolkuun tehtyjä muutoksia perehtyjän kannalta.

Työn teoriaosuudessa kerrotaan versionhallinnasta sen kehityksen ja toimintojen kautta, sekä kerrotaan myös versionhallintajärjestelmistä. Aiheet ja käsitteet on tuotu esiin ja kuvattu aiemmin luodun kirjallisuuden, sekä verkkomateriaalin kautta.

Tämän opinnäytetyön lopputuloksena toimeksiantaja sai päivitetyn opintopolun. Sen avulla perehtyjän on vaivatonta siirtyä toimeksiantoyrityksen asiakasprojekteihin. Tämä opinnäytetyö tukee toimeksiantajan onboarding-prosessia, mutta se on helposti hyödynnettävissä tämän opinnäytetyön muodossa myös muille aloille.

Avainsanat Business Intelligence, Git, Onboarding, Versionhallinta

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Jonna Jaakkola
Title	Version Control as a Tool in Onboarding Case: Oiwa Solutions Oy, A Learning Path 2.0
Year	2020
Language	Finnish
Pages	49 + 1 Appendices
Name of Supervisor	Raija Tuomaala

This thesis was made for Oiwa Solutions Oy. The aim of this thesis was to update a Learning Path that was previously made for the company. The aim of the changes was to create similar tasks to Learning Path with the tasks of Oiwa Solutions Oy. In the practical section of this thesis all the changes made to the original Learning Path from the view of the developer trainee are described.

The theoretical study of the thesis tells about version control through its development and functions, as well as about different version control systems. The topics and concepts are introduced based on previously created literature, as well as online material.

As the result of this thesis, Oiwa Solutions Oy received an updated Learning Path version, which makes it easy for the developer trainee to move on to the client projects of the Oiwa Solutions Oy. This thesis supports the onboarding process at Oiwa Solutions Oy, while being useful in the form of this thesis for other areas as well.

Keywords Business Intelligence, Git, Onboarding, Version Control

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

1 JOHDANTO

Tässä opinnäytetyössä päivitetään Riku Ojasen toimeksiantajalle luomaa opintopolkua ja luoda siihen testivaiheita, joiden avulla toimeksiantaja voi seurata uuden työntekijän etenemistä opintopolulla. Opinnäytetyössä opintopolkua tarkastellaan versionhallinnan näkökulmasta, sillä toimeksiantaja on vaihtamassa versionhallintajärjestelmänsä Gitiksi.

Opinnäytetyön **toimeksiantajana** toimii Oiwa Solutions Oy. Oiwa Solutions Oy on tiedolla johtamisen ratkaisuihin erikoistunut IT-alan yritys. Opinnäytetyössä luotavat dokumentit ja materiaalit jäävät toimeksiantajalle, eikä työssä esitellä tarkkoja määritelmiä opintopolusta.

Opintopolku on kattava perehdytyskokonaisuus, jossa käydään pala kerrallaan läpi tietovarastointi- ja Business Intelligence -ratkaisun vaiheita. Opintopolussa perehtyjän on tutustuttava jokaiseen aiheeseen kunnolla ja luotava toimiva kokonaisuus ennen kuin pystyy etenemään seuraavaan aiheeseen opintopolulla. Opintopolun merkitys toimeksiantajalle on suuri, sillä työtehtävät toimeksiantoyrityksessä eivät juurikaan vastaa ammattikorkeakouluissa käsiteltäviä aiheita. Opintopolku mahdollistaa perehtymisen ja aiheisiin tutustumisen tekemisen kautta. Opintopolussa on paljon kehitettävää, mutta se luotiin muotoon, jota on helppo kehittää. (Ojanen 2019)

Business Intelligence on tiedon systemaattista hankintaa, varastointia ja analysointia. Business Intelligence sisältää työvälineitä, joiden avulla raakadatasta, eli jalostamattomasta datasta, jota voidaan transformoida ja käsitellä, saadaan yritystoimintaa hyödyttävää tietoa. Business Intelligencen työvälineet mahdollistavat suurten tietomäärien tehokkaan tulkitsemisen. Business Intelligence helpottaa päätöksentekoa yrityksissä, sillä raportoinnin ja analytiikan avulla saadaan ymmärrettävää tietoa yritysten liiketoiminnasta ennen nykyhetkeä, nykyhetkessä ja tulevaisuudessa. (Kaario & Peltola 2008, 61; Ite wiki 2020; Hovi 2020; Tieteen Termipankki 2020)

Versionhallinta tapahtuu järjestelmässä, joka tallentaa tiedostojen muutokset yhdeksi tiedostoksi tai useiksi tiedostoiksi. Kun tiedostoa muutetaan versionhallinnassa, siitä syntyy uusi versio, mutta myös vanhemmat versiot pysyvät tallessa. Tämä mahdollistaa tiedoston palauttamisen aiempaan versioon. Versionhallinnalla pystytään myös seuraamaan kuka on tehnyt muutoksen ja mitä on muutettu. Versionhallintaan soveltuvat lähestulkoon kaikki tietokoneen tiedostotyypit. **(**Chacon & Straub 2014; Sinc 2011)

Perehdytys tarkoittaa perinteisen määritelmän mukaan työnantajan luomaa koulutuskokonaisuutta uuden työntekijän työsuhteen alussa. Työnantajalla on vastuu varmistua siitä, että työntekijällä on riittävä osaaminen työtehtäväänsä. Englannin kielessä perehdytyksestä käytetään termejä orientation ja onboarding. Termit eroavat toisistaan tavoitteissaan. Orientation tähtää työssä vaadittavien perusvalmiuksien hankkimiseen, kun onboarding sen sijaan tähtää tuottavaan tekemiseen ja työntekijän sitoutumiseen. (Siitonen 2020; Luoto 2012)

Onboarding-prosessi vaatii työnantajalta jo ennen työsuhteen alkamista yhteydenpitoa ja huolehtimista. Huolellisesti suunnitellun perehdytysohjelman luominen on työläs toteuttaa, eikä vastaa välttämättä uuden työntekijän työtehtäviä, mutta laajemmin ymmärrettyyn perehtymiseen kuuluvat yhteinen tavoitteiden asettaminen ja palautteen saaminen työtehtävien haltuunotosta sekä niiden toteuttamisesta. Onnistuneen onboarding-prosessin jälkeen työntekijällä on selkeä kuva työnantajan tavoitteista ja työntekijällä on halu ja oikea tunnelma työskennellä työnantajalle. Onnistunut onboarding-prosessi palvelee myös työnantajaa, sillä se nopeuttaa työntekijän pääsyä tuottavaan tekemiseen. (Luoto 2012; Maurer 2020)

Tämä opinnäytetyö on empiirinen eli havainnollinen tutkimus, jossa hyödynnetään laadullista tutkimusta. Laadullisen tutkimuksen keinoin pystytään kuvaamaan versionhallinnan merkitystä ja mahdollisuuksia uuden työntekijän perehdytysprosessissa. Teoriataustan sisältö määräytyi keskeisten tutkimuskysymysten perusteella versionhallintaan ja tapoihin, jotka helpottavat uuden työntekijän pääsyä tehokkaaseen työskentelyyn. Lähestymistapa- ja toteutusosiossa kuvataan ja perustellaan opintopolkuun tehtyjä muutoksia

versionhallinnan kannalta sekä esitellään opintopolkuun lisättyjä testivaiheita toimintatutkimuksen näkökulmasta. Johtopäätökset ja pohdinta -luvussa kuvataan tutkimuksen tuloksia, työn merkittävyyttä, mahdollisia jatkokehitysideoita ja kehittämishanketta kokonaisuudessaan.

2 TARVEANALYYSI JA KUVAUS NYKYTILASTA

Opinnäytetyön tavoitteena on luoda toimeksiantajalle entistä parempi ja ajantasaisempi opintopolku, joka mahdollistaa uuden työntekijän ja harjoittelijan tuottavaan työhön pääsemisen. Opintopolkua päivitetään toimeksiantajan toiveiden ja tarpeiden mukaisesti, sekä siihen lisätään testivaiheita, joiden avulla voidaan seurata suorittavan työntekijän etenemistä opintopolulla. Opinnäytetyön tutkimus on rajattu versionhallintajärjestelmä Git:n käyttämiseen työntekijän ja harjoittelijan perehdytyksessä asiakasprojekteihin.

Lähtökohta tälle tutkimukselle oli tarpeiden määritteleminen. Toimeksiantajaa haastattelemalla, opintopolun läpikäymisellä ja asiakasprojektissa mukana olemisella selkenivät tämän kehittämishankkeen tavoitteet ja keskeisimmät ongelmat. Kun keskeisimmät ongelmat selkenivät, pystyi opinnäytetyön aiheen rajaamaan versionhallinnan merkitykseen asiakasprojekteissa ja siten myös työntekijöiden perehdytyksessä.

Opinnäytetyön käytännön osuudessa kuvataan opintopolkuun tehtyjä muutoksia. Muutosten painopiste on versionhallintajärjestelmä Git:iin liittyvissä tehtävissä, sillä toimeksiantaja tulee käyttämään jatkossa syntyvissä asiakasprojekteissa Git-järjestelmää Azure DevOps:n ympäristössä, eikä alkuperäisessä opintopolussa ole toimeksiantajan mukaan riittävästi kuvattu versionhallintaa. Jotta Git:n käytöstä todellisessa asiakasprojektissa saadaan riittävän realistinen kokonaiskuva ja ohjeistus, muutetaan opintopolkua toimeksiantajan toiveiden mukaiseksi. Vaikka opintopolku on jo kattava kokonaisuus Business Intelligenceä, on toimeksiantajalla tarve kehittää opintopolkua, sekä sen toteutuksen testaamista. Opintopolku on toimeksiantajan mukaan liian helppo toteuttaa, eikä siten vastaa todellista asiakasprojektia tarpeeksi. Tavoitteena on luoda asiakasprojektia vastaava yhdenmukainen perehdytyskokonaisuus. Yhdenmukainen perehdytysprosessi mahdollistaa tulevaisuudessa toisista työntekijöistä riippumattoman työskentelyn, sillä jokainen projekti toteutetaan samalla tavalla niissä puitteissa, joissa ne on mahdollista toteuttaa. Kun opintopolun tehtävät vastaavat asiakasprojektia, pääsee

myös perehtyjä heti opintopolun suoritettuaan kiinni tuottavaan työntekoon. (Salonen 2020)

Alkuperäisessä opintopolussa perehtyjä loi itse operatiivisen lähdejärjestelmän taulurakenteen ja dataa siihen. Kun data luodaan itse, siitä ei tule riittävän monimutkaista vastaamaan asiakkaan raakadataa. Tietovarastoinnista toimeksiantajalla heräsi tarve kuvata Extract Transform and Load (ETL) -prosessissa käytettävän Temp-scheman lisäksi myös Working-kanta, sillä sen asiakasprojekteissa käytetään molempia tapoja, eikä alkuperäisessä opintopolussa ole kuvattu lainkaan working-kannan käyttöä. Alkuperäisessä opintopolussa on materiaalia sekä tabulaarisesta, että multidimensionaalisesta kuutiosta, mutta multidimensionaaliseen liittyvästä MDX -kielestä ei opintopolussa ole tietoa tai esimerkkejä, vaikka sitäkin joudutaan tietyissä asiakasprojekteissa hyödyntämään. Raportoinnin painopiste toimeksiantajalla on siirtynyt Power BI -raportteihin opintopolun SSRS-raporttien sijaan ja päivitetyssä opintopolussa Power BI:n osuutta halutaankin korostaa luomalla selkeästi haastavampia ja realistisempia esimerkkejä. Jokainen opintopolun vaihe tulisi jatkossa tallentaa versionhallintaan. Se auttaa perehtyjää pitämään materiaalinsa tallessa ja samalla toimeksiantajan on mahdollista seurata perehtyjän etenemistä. Etenemistä seurataan myös testauspatteristolla, jolla varmistetaan perehtyjän riittävä tutustuminen aiheisiin opintopolun varrella.

Opintopolkuun tulisi lisätä:

- Git-versionhallintajärjestelmä
- Realistiset lähdejärjestelmät
- Temp-scheman ja Working-kannan erot
- OLAP ja MDX
- Power BI -tehtäviä
- Testauspatteristo

Perehdytyksen kannalta olennaisin tutkimuskysymys on, mitä opintopolun pitäisi sisältää, jotta se vastaisi asiakasprojektia mahdollisimman realistisesti? Miten onboarding-prosessin onnistuminen taataan? Miten perehtymistä voidaan seurata ja

testata? Versionhallinnasta tärkeintä toimeksiantajalle olisi tietää, mikä Git on ja kuinka siitä voidaan hyötyä yrityksessä ja perehdytyksessä.

3 VERSIONHALLINTA

Versionhallinta on kehittynyt viiden vuosikymmenen aikana merkittävästi ja kehittyy koko ajan. Versionhallinnan kehitys voidaan tällä hetkellä jakaa kolmeen sukupolveen. Versionhallintajärjestelmät ovat samalla tavalla kehittyneet ja usein nykyään ne jaetaankin kahteen osa-alueeseen: keskitettyihin ja hajautettuihin. Keskitetyissä ja hajautetuissa versionhallintajärjestelmissä on useita samoja toimintoja, mutta jokaisella järjestelmällä on myös omia kehittyneempiä toimintojaan.

3.1 Versionhallinnan kehitys

Versionhallinnan kehitys lasketaan alkaneeksi vuodesta 1972. Silloin perustettiin ensimmäinen versionhallintajärjestelmä Source Code Control System (SCCS). SCCS kehitettiin Unix -käyttöjärjestelmälle ja se toimi ainoastaan lähdekooditiedostoille. Tässä vaiheessa olisi teknologisesti ollut mahdollista tallettaa vain tiedostojen muutokset, mutta kulttuurillisesti siihen ei oltu valmiita ja tästä syystä jokainen tiedoston versio talletettiin kokonaisuudessaan. 1980-luvulla kehitettiin ensimmäinen järjestelmäriippumaton versionhallintajärjestelmä Revision Control System (RCS), mutta edelleenkin versionhallinta toimi vain tekstitiedostoille ja vain yhden henkilön käytössä. Tätä kehitystä kuvataan versionhallinnan ensimmäiseksi aalloksi. (LinkedIn Corporation 2020; Carstensen 2016)

Versionhallinnan toisen aallon katsotaan alkaneen vuodesta 1986, jolloin keskitetty versionhallinta kehitettiin. Concurrent Versions System (CVS) oli ensimmäinen järjestelmä, jossa oli keskitetty repository, eli tietokanta, jossa tiedostoja ja niiden kaikkia versioita säilötään (Ernst 2018; Brun 2012). Keskitetty repository mahdollisti usean henkilön samanaikaisen versionhallinnan käytön. CVS perustui kehityksestään huolimatta silti vielä yksittäisten tiedostojen muutosten jäljittämiseen, koko hakemistopuun muutosten jäljittämisen sijaan. CVS oli ensimmäinen versionhallintajärjestelmä, joka hyödynsi tiedostoja muokatessa niiden lukitsemista, mutta siitä huolimatta se oli epäluotettava aina Subversionin käyttöönottoon saakka. Vuonna 2000 kehitettiin muitakin kuin tekstitiedostoja

tukeva ja koko hakemiston muutoksia jäljittävä keskitetty versionhallintajärjestelmä Subversion (SVN). Vuonna 2004 Microsoft lanseerasi Team Foundation Serverin (TFS), joka vastaa toiminnallisuudessaan pitkälti SVN:n toimintoja. TFS:n suosioon vaikutti merkittävästi se, että monilla yrityksillä oli jo entuudestaan Microsoftin lisenssejä ja siten oli helpompaa ottaa käyttöön TFS. (LinkedIn Corporation 2020; Carstensen 2016, Rouse 2011; Perforce Software 2020a; Perforce Software 2020b)

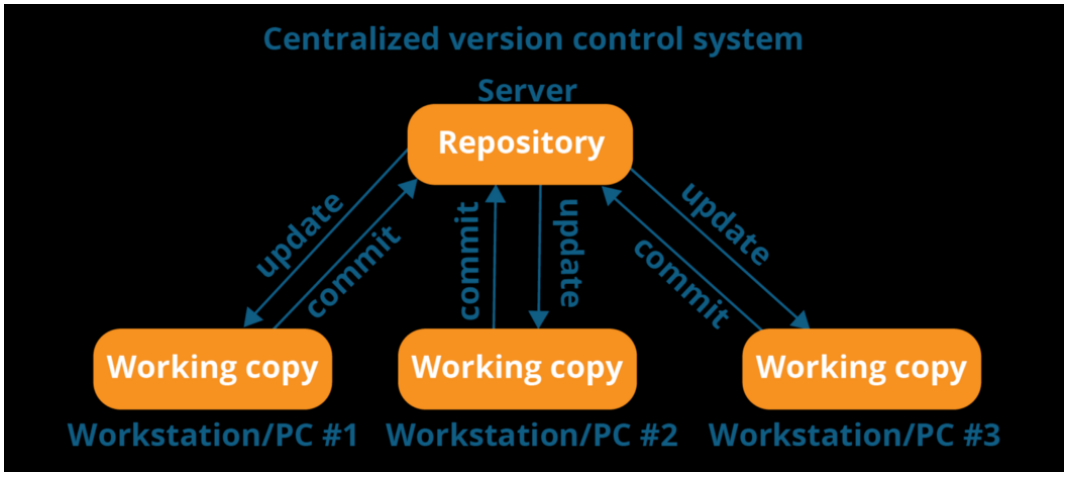
Versionhallinnan kolmannen aallon perustana on hajautettu versionhallintajärjestelmä. Hajautetun versionhallinnan kuvataan saaneensa alkunsa vuonna 2005 BitKeeper-yrityksen emoyhtiön päätöksestä tehdä tytäryhtiönsä tuotteesta Community Edition maksullinen. Community Edition oli alun perin Linus Torvaldsin luomassa Linux-käyttöjärjestelmässä käytössä ilmaiseksi. Kun BitKeeper päätti kaupallistaa tuotteensa, Torvalds loi Community Editionin tilalle versionhallintajärjestelmä Git:n varmistaakseen, että kaikki Linux-käyttöjärjestelmän ympärillä on ilmaista. Vuonna 2005 luotiin myös toinen kilpaileva versionhallintajärjestelmä Mercurial perustuen samaan BitKeeperin päätökseen kaupallistaa Community Edition. Vuonna 2020 hajautetut versionhallintajärjestelmät ovat edelleen käytössä. Tosin ne ovat kehittyneet vastaamaan enemmän käyttäjien tarpeita. (LinkedIn Corporation 2020; Larabel 2016; The Linux Foundation 2020)

Versionhallintajärjestelmien kehityksestä on tehty ennustuksia, mutta ennustusten toteutumisen näkee vasta tulevaisuudessa. On ennustettu, että versionhallintajärjestelmistä tulee käyttäjäystävällisempiä ja reaaliaikaisia. Toisaalta on ennustettu myös, että Git syrjäyttää muut versionhallintajärjestelmät, sillä yhä uesampi yritys vaihtaa jo nyt versionhallintajärjestelmänsä Git:iin. Ennusteissa on pohdittu myös tekoälyn hyödyntämistä versionhallinnassa. (Ahmad 2020; Anstett 2020; Marroccos 2020)

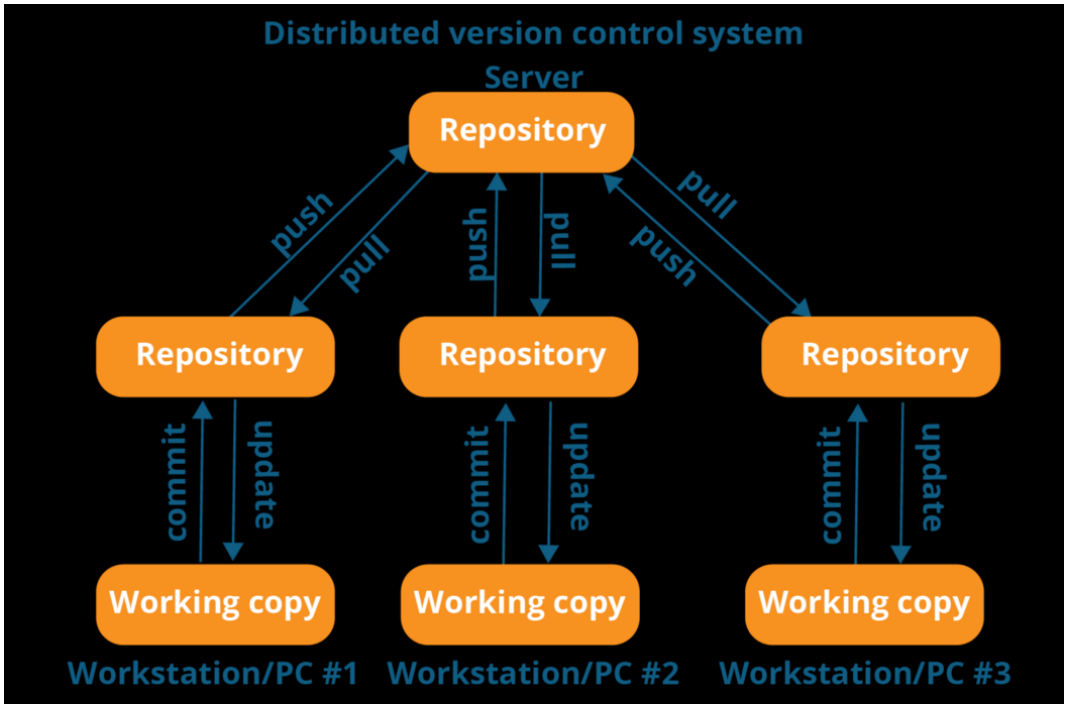
3.2 Versionhallintajärjestelmät

Versionhallintajärjestelmät ovat ohjelmistoja, jotka auttavat työskentelemään yhdessä ja arkistoimaan työn kaikki vaiheet. Versionhallintajärjestelmän

päätavoitteina ovat samanaikainen työskentely, jokaisen version ja muutoksen arkistointi ja ristiriitojen syntymisen estäminen. Versionhallintajärjestelmän avulla voi palata tiedoston aiempaan versioon, jos rikkoo tai kadottaa tiedoston tai osia siitä, verrata ajan saatossa tehtyjä muutoksia, sekä nähdä kuka muutoksia on tehnyt ja minkä takia. Tämä helpottaa selvittämään mahdollisen virheen jälkeen sen alkuperää. (Sinc 2011; Atlassian 2020)



Kuva 1 Keskitetyn versionhallinnan toimintaperiaate (Lubański 2019)



Kuva 2 Hajautetun versionhallintajärjestelmän toimintaperiaate (Lubański 2019)

Versionhallintajärjestelmät jaetaan nykyisten määrittelyjen mukaan keskitettyyn ja hajautettuun versionhallintaan. Keskitetyssä versionhallinnassa on ideana, että tiedostosta on olemassa repositoryssa yksi keskitetty kopio, josta kehittäjät ottavan niin sanotun työkopion. Työkopiota voidaan muokata versionhallinnan ulkopuolella, jonka jälkeen tehdyt muutokset voidaan tuoda takaisin repositoryssa olevaan kopioon. Kun muutos on tuotu takaisin, muut käyttäjät näkevät päivitetyn version alkuperäisestä kopiosta. Kuvassa 1 kuvataan tätä toimintaperiaatetta, kun sen sijaan kuvassa 2 kuvataan hajautetun versionhallinnan toimintaperiaatetta. Hajautetussa versionhallinnassa kehittäjä kloonaa koko repositoryn omalle kovalevylleen pull -toiminnolla, jonka jälkeen voi tehdä tälle kloonille muutoksia. Klooniin tehdyt muutokset voidaan työntää push-toiminnolla takaisin alkuperäiseen repositoryyn, jonka jälkeen muutokset ovat kaikkien käytettävissä. (Atlassian 2020; Perforce Software 2020c; Lubański 2019; Chacon & Straub 2014)

3.2.1 Azure DevOps Server

Azure DevOps Server, aiemmin Team Foundation Server (TFS) ja Visual Studio Team System (VSTS), on Microsoftin vuonna 2005 tuotteistama versionhallintajärjestelmä. Se on keskitetty versionhallintajärjestelmä, joka on räätälöity toimimaan Visual Studio - ja Eclipse-sovelluksissa kaikilla alustoilla. (Cool 2019; Mar 2020; Microsoft Docs 2020h)

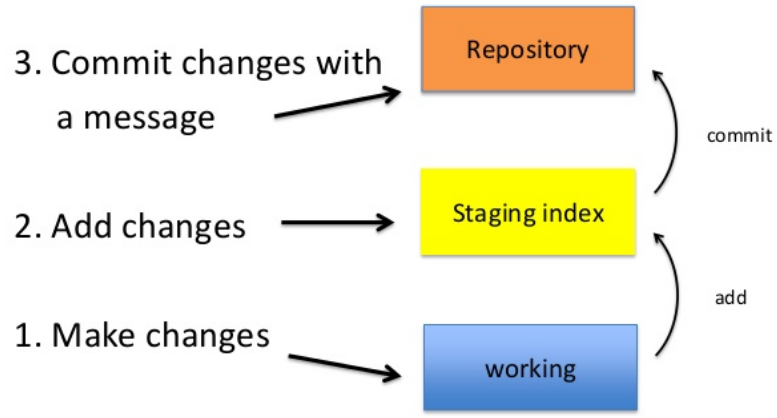
Azure DevOps Serverin arkkitehtuuri toimii asiakas-, sovellus- ja datatasolla. Asiakastaso kommunikoi sovellustason kanssa, kun luodaan tai hallitaan tiedostoja. Asiakastasolla Azure DevOps Serverillä ei ole sovellusta, vaan silloin hyödynnetään esimerkiksi Visual Studiota. Sovellustaso sisältää ASP.NET-verkkopalvelun, joka on asiakastason käytettävissä. ASP.NET-verkkopalvelu on ilmainen viitekehys avoimella lähdekoodilla verkkosovellusten ja -palveluiden kehittämiseen (Microsoft 2020a). Azure DevOps Server ei tue suoraa pääsyä asiakastasolta datatasolle, joten se toteutaan verkkopalvelun kautta. Datatasolla tietovarastot keskustelevat sovellustason palveluiden kanssa. (Microsoft Corporation 2015)

Azure DevOps Serverin heikkoutena pidetään sitä, ettei se sovellu kaikkien ohjelmointikielten käyttöön. Samanaikaisten muutosten aiheuttamat ristiriidat ovat myös Azure DevOps Serverin heikkous, sillä niiden ratkaiseminen on hankalaa. Azure DevOps Serverin käyttöliittymä on todettu myös hankalaksi käyttää. (TrustRadius 2013-2020)

Azure DevOps Serverin vahvuudet sen sijaan ovat virheiden jäljittäminen, integaartio ja projektinhallinta. Azure DevOps Serverillä muutosten ja samalla virheiden jäljittäminen onnistuu helposti. Azure DevOps Server on integroitavissa saumattomasti esimerkiksi Microsoftin tuotteiden kanssa, jotka ovat jo entuudestaan yleisesti yritysten käytössä. Koska Azure DevOps Server tukee ketterän kehityksen projektinhallinnan viitekehyksiä kuten Scrumia, myös projektinhallinta on helppoa. (TrustRadius 2013-2020)

3.2.2 GitHub

GitHub on Chris Wanstrathin, P. J. Hyettin, Tom Preston-Wernerin ja Scott Chaconin vuonna 2007 perustama verkkopalvelu. GitHub toimii Linus Torvaldsin luomasta Git-versionhallintajärjestelmästä graafisena käyttöliittymänä. Git on hajautettu versionhallintajärjestelmä. GitHub:iin on lisätty työkaluja, jotka mahdollistavat helpomman versionhallinnan ohjelmistokehityksessä. Helpotusta tuo myös onlinessa toimiminen, joka mahdollistaa työskentelyn missä tahansa paikassa. (Mar 2020; Heise Medien 2020; Kumar 2019)



Kuva 3 Git toimii kolmella tasolla (Dolan 2016)

Koska GitHub toimii Git-versionhallintajärjestelmän päällä, sen arkkitehtuurina toimii Git:n arkkitehtuuri. Versionhallintajärjestelmillä on kolme ydintoimintoa, jotka Linus Torvalds on alunperin kehittänyt Git-versionhallintajärjestelmään. Ne ovat sisällön tallentaminen, sisällön muutosten seuraaminen ja sisällön jakaminen. Git toimii Azure DevOps Serverin tavoin kolmella kerroksella. Git-versionhallintajärjestelmän kerrokset ovat working directory, staging area ja local repository. Kuvassa 3 kuvataan, mitä jokaisella tasolla tapahtuu: Working-tasolla tehdään muutoksia tiedostoon paikallisesti, Staging-tasolla muutokset lisätään versionhallintajärjestelmään ja Repository-tasolla muutokset liitetään viestin kera paikalliseen repositoryyn. (Hay 2018; Intellipaat 2011-2020)

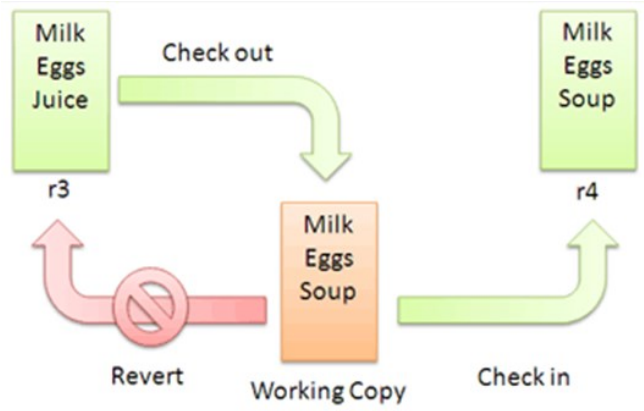
GitHub:n heikkouksena pidetään Git-versionhallinnan monimutkaisten rakenteiden ja toimintojen oppimista, hinnoittelua ja turvallisuusriskejä. Vaikka suuri osa GiHub:sta on täysin ilmainen, siinäkin on osia, jotka kerryttävät maksuja erityisesti silloin, kun kehittäjiä on useampia. Kuten muita verkkosovelluksia kohtaan, myös GitHub:ia kohtaan on ollut tietoturvarikkomuksia, mikä olisi hyvä ottaa huomioon, kun GitHub:iin lisätään arkaluonteisia tiedostoja. (Intellipaat 2011-2020; Glancy 2020)

GitHub:n hyviä puolia sen sijaan mainitaan olevan helpompi yhteistyö, rekrytointi ja integraatio. GitHub mahdollistaa yhteistyön, vaikka fyysinen sijainti olisi eri ja

pääsy yrityksen sisäiseen verkkoon olisi estynyt, sillä se toimii verkossa. Nykyään GitHub toimii myös rekrytointivälineenä, sillä GitHub:ssa oman osaamisensa voi julkaista omaan profiiliinsa kaikkien nähtäville. GitHub integroituu yleisimpien alustojen kanssa, mikä mahdollistaa monipuolisen työskentelyn. GitHub:n etuna mainitaan myös laaja dokumentaatio lähes jokaisesta Git:n toiminnosta, mikä helpottaa ongelmatilanteiden selvittämistä. (Glancy 2020; Apiumhub 2020)

3.3 Yleiset toiminnot versionhallinnassa

Versionhallinnassa toimiakseen ja ymmärtääkseen versionhallinnan merkitystä, olisi hyvä ymmärtää edes yleisimmät versionhallinnan toiminnot. Eri versionhallintajärjestelmillä on omat käsitteensä jokaiselle toiminnolle. Tässä osiossa esitellään yleisimmät käsitteet eri versionhallintajärjestelmistä, sekä niiden merkitykset. Kuvassa 4 mallinnetaan versionhallinnassa tehtävää tiedostomuutosta tyypillisimpiä toimintoja hyödyntäen.



Kuva 4 Tiedoston muokkaaminen versionhallinnassa. (Azad 2007)

Tyypillisesti versionhallinnassa tiedosto lisätään Add-toiminnolla repositoryyn. Kun tiedosto on lisätty repositoryyn, se todennäköisesti ladataan Check Out -toiminnolla paikalliseksi kopioksi muokkausta varten. Kuvassa 4 repositoryyn lisättyä tiedostoa kuvaa vihreä laatikko, jossa tiedoston sisältöä kuvaa Milk, Eggs ja Juice. Paikallista kopiota kuvassa 4 edustaa oranssi laatikko, jonka sisältöä kuvaa Milk, Eggs ja Soup. Muutoksen jälkeen tiedosto siirretään takaisin Check In -

toiminnolla repositoryyn ja samalla kirjataan Checkin Message, jossa kuvataan tehtyjä muutoksia, tai päädytään säilyttämään alkuperäinen versio, jolloin Revert -toiminnolla palautetaan tiedosto alkuperäiseen muotoonsa. Kuvassa 4 r3 ja r4 ovat tiedoston revisioita, eli tiedoston versioita, joiden tarkoitus on korvata edeltäjänsä (Wildbit 2007-2019). Repositoryyn Check In -toiminnolla tuotua tiedostoa kuvaa vihreä laatikko, jonka sisältönä on Milk, Eggs ja Soup. Kuvan 4 tapauksessa Checkin Message kirjattaisiin Check In -nuolen kohdalla.

Versionhallinnalle tyypillistä on, että useat henkilöt muokkaavat tiedostoja ja onkin todennäköistä, että seuraavana päivänä toinen henkilö haluaa muokata tiedostoa. Niimpä tämä toinen henkilö päivittää Update-toiminnolla oman näkymänsä repositorysta ja saa käyttöönsä viimeisimmät versiot tiedostoista, joissa edellisenä päivänä tehdyt muutokset ovat voimassa. Jos muutoksissa on virheitä tai halutaan tarkastella, ketkä tiedostoa ovat muokanneet tai mitä tiedostossa on muutettu, Changelog- tai History-toiminnoilla päästään näihin tietoihin käsiksi. (Azad 2007; Microsoft Docs 2020d)

3.3.1 Map/ Clone

Map-toiminnolla luodaan olemassa olevasta repositorysta klooni tai kopio paikalliselle koneelle tai muulle tuetulle protokollalle. Klooni sisältää täydellisen kopion kaikista versionhallinnan repositoryssa olevista tiedostoista ja kansioista, sekä niiden jokaisesta versiosta. (Atlassian Bitbucket 2020b, GitHub Docs 2020)

3.3.2 Add

Add-toiminnossa tiedosto tuodaan ensimmäistä kertaa repositoryyn. Tiedoston jäljittäminen versionhallinnassa alkaa tästä hetkestä. Jotta tiedostoa voidaan muokata versionhallinnassa, se täytyy tuoda ensin Add-toiminnolla repositoryyn. (Azad 2007; GitHub 2020)

3.3.3 Head

Head vastaa Check Out -toiminnolla ladattua viimeisintä versiota tiedostosta. Toisin sanoen sen avulla nähdään Branch:ssä tiedoston versio, johon on viimeksi tehty muutoksia. Jokaisessa Branch:ssä on oma Head. (Azad 2007; Gregory 2011)

3.3.4 Check Out

Check Out -toiminnolla ladataan tiedosto repositorysta paikalliseksi kopioksi joko manuaalisesti tai automaattisesti riippuen versionhallintajärjestelmästä. Paikalliseen kopioon voidaan tehdä muutoksia, jotka tuodaan takaisin repositoryyn joko Check In -toiminnolla tai Revert-toiminnolla. Check Out -toiminnolla voidaan ladata myös useampia tiedostoja kerrallaan. (Azad 2007; Microsoft Docs 2020b, MadCap Flare 2020)

3.3.5 Check In/ Push

Check In -toiminnolla siirretään tiedosto muutosten jälkeen takaisin repositoryyn. Jos tiedosto on muuttunut, se saa uuden versionumeron. Kun tiedosto on siirretty Check In -toiminnolla, seuraava tiedoston muokkaaja voi jälleen Check Out -toiminnolla ladata tiedoston muokattavaksi. Jos Check In -toiminnolla yritetään tuoda tiedostoa, joka viittaa johonkin toiseen tiedostoon, joka ei ole versionhallinnassa, ilmenee virhe. Tosin, virhettä ei ilmene, jos nämä tiedostot ovat samassa Check In -listassa. Check In -lista on listaus kaikista Check Out -toiminnolla ladatuista tiedostoista, joihin on tehty muutoksia. Check In -listalta hyväksytään tai hylätään kaikki muutokset, joita versionhallinnan repositoryyn halutaan tuoda Check In -toiminnolla. (Azad 2007; Microsoft Docs 2020c)

3.3.6 Checkin Message/Commit

Checkin Message nimensä mukaisesti kertoo käyttäjältä seuraavalle Check In -toiminnossa tehtyjä muutoksia. Tyypillisesti viestin sisältö on kuvaus koodimuutoksesta: ”Muutos Mergeen”, ”Bodyn päivitys”, mutta muiden on vaikea tulkita näistä, mikä muutoksen tarkoitus on todella ollut. On mahdollista, että muutoksen tekijäkään ei muista tekemäänsä muutosta ajan kuluessa. Senpä takia

Checkin Messageen tulisi kuvata mahdollisimman tarkasti muutoksen sisältöä ja sijaintia. (Azad 2007; Dott 2020)

3.3.7 Changelog/ History

Changelog on tiedosto, joka näyttää tiedostoille tehdyt muutokset kronologisessa järjestyksessä siitä päivästä lähtien, kun tiedostot on lisätty versionhallintaan. Changelog sisältää yleensä tiedoston version, päivämäärän ja listan tiedostoon lisätyistä, kehitetyistä ja poistetuista piirteistä. Changelog helpottaa muutosten jäljittämistä erityisesti useiden henkilöiden projekteissa. (Azad 2007; Lacan 2017; Gaël 2020)

3.3.8 Update/ Sync/Get latest/Pull

Update-toiminnolla synkronoidaan omat tiedostot versionhallinnan repositoryssa olevien tiedostojen kanssa. Repositoryssa olevista tiedostoista tuodaan tällä toiminnolla jokaisen tiedoston viimeisin versio. Update ei synkronoi uusia luotuja tiedostoja, ellei niitä ole ehditty tuomaan repositoryyn Check In -toiminnolla, eikä siten myöskään synkronoi tiedostoja, jotka ovat Check Out -toiminnolla ladattu pois repositorysta. (Azad 2007; Microsoft Docs 2020d)

3.3.9 Revert/Rollback

Revert-toiminnolla voi hylätä kaikki tehdyt muutokset ja palata tiedoston viimeisimpään repositoryn versioon. Revert ei kuitenkaan poista muutosjoukkoja tai dataa. Revert-toiminto näkyy Changelog:ssa samanlaisena muutoskena, kuin muut toiminnot ja myös Revert-toiminnosta voi palata tiedoston edeltävään versioon tarvittaessa. (Azad 2007; Microsoft Docs 2020e)

3.3.10 Rename

Kun tiedosto nimetään uudelleen Rename-toiminnolla, siitä syntyy kaksi kopiota. Toinen kopioista on vanhalla nimellä ja toinen uudella. Kun nimeäminen tuodaan Check In -toiminnolla repositoryyn, versionhallintajärjestelmä integroi molemmat kopiot Changelog:n ylläpitämiseksi. Changelogissa muutos näkyy vanhalla nimellä

olevasta kopiosta Delete-toimintona ja uudella nimellä olevasta kopiosta Add-toimintona, vaikka tiedoston sisältö ei olisi muuttunut. (Microsoft Docs 2020a)

Tiedoston nimeämisessä tulisi olla selkeä logiikka. Tiedoston nimen tulisi alusta alkaen olla kuvaava, sillä uudelleen nimeämistä olisi hyvä välttää. Uudelleen nimeämisessä ongelmana on se, että sen seurauksena tiedostojen väliset riippuvuudet voivat hajota. (National Instruments Corporation 2020)

3.4 Kehittyneemmät toiminnot versionhallinnassa

Versionhallinnassa voi ilmetä yleisistä toiminnoista poikkeavia monimutkaisempia toimintoja. Esimerkiksi ristiriitojen ymmärtämiseksi, ratkaisemiseksi ja välttämiseksi olisi hyvä tutustua kehittyneempiinkin toimintoihin. Tässä osiossa esitellään osa versionhallintajärjestelmissä esiintyvistä kehittyneemmistä toiminnoista.

3.4.1 Branch

Branch:n avulla voi luoda yksityiseen käyttöön kopion alkuperäisestä tiedostosta tai kansiosta esimerkiksi testaamista varten. Branch-toimintoon liittyy verbi branching, joka tarkoittaa suomeksi haarautumista (Redfox Languages Oy 2014-2020). Haarautuminen mahdollistaa rinnakkaisen kehitystyön samalla erottaen keskeneräiset versiot testatuista ja hyväksytyistä versioista. (Azad 2007; Schiestl 2020)

3.4.2 Diff/Change/Delta

Diff-toiminnon avulla voidaan etsiä eroavaisuuksia kahden tiedoston tai tiedostoversion välillä. Diff-toiminto ei näytä koko tiedostoa, jota on muutettu, vaan se tuo ainoastaan muutetut tai toisistaan eroavat kohdat. Kun Diff-toimintoa oppii tulkitsemaan, pystyy ymmärtämään projektin tai tiedoston kehitystä. (Azad 2007; Günther 2014)

3.4.3 Merge

Merge-toiminnolla liitetään muutokset toisesta tiedostosta toiseen, jotta saadaan esimerkiksi haarautuneet muutokset repositoryssa olevaan tiedostoon ajantasalle. Merge-toimintoa hyödynnyteään erityisesti versionhallinnassa, jossa kehittäjiä on enemmän, kuin yksi, sillä merge yhdistää kaikki muutossarjat yhteen tiedostoon. Tällä mahdollistetaan päällekkäisyyksien välttäminen. (Azad 2007; Atlassian BitBucket 2020c; TechoPedia 2020)

3.4.4 Conflict & Resolve

Eri versionhallintajärjestelmistä riippuen Conflict ilmenee eri tilanteissa. Esimerkiksi Conflict voi ilmetä, kun samanaikaisesti yritetään Check In -toiminnolla tuoda kahden tai useamman eri kehittäjän ristiriidassa olevaa muutosta tiedostoon. Samanaikaisesti voidaan tuoda vain yhden kehittäjän muutos. Versionhallintajärjestelmä ei osaa päättää, mitkä muutokset pitäisi tuoda, joten Resolve-toiminnolla ristiriita ratkaistaan manuaalisesti tai automaattisesti riippuen siitä, osaako versionhallintajärjestelmä ratkaista ristiriidan. Jos versionhallintajärjestelmä ei osaa ratkaista ristiriitaa, se ratkaistaan manuaalisesti järjestelmästä saatujen tietojen perusteella. (Azad 2007; Ernst 2018; JetBrains 2000-2020; Microsoft Docs 2020f)

3.4.5 Locking

Locking-toiminnolla lukitaan tiedosto vain omaan käyttöön kunnes lukitus päätetään taas purkaa. Osa versionhallintajärjestelmistä hyödyntää tätä toimintoa, jotta vältyttäisiin conflict:lta. Esimerkiksi keskitetyssä versionhallintajärjesetelmässä tiedosto lukitaan toisilta käyttäjiltä, kun joku lataa siitä Check Out -toiminnolla itselleen kopion, jota muokkaa. Kun tiedostoon on tehty muutos ja se palautetaan takaisin Check In -toiminnolla, lukitus avautuu ja tiedosto on näin kaikkien käytettävissä. (Azad 2007; Microsoft Docs 2020g)

3.4.6 Stash/ Shelve

Stash-toimintoa käytetään, kun halutaan pitää tallessa Check Out -toiminnolla ladattuun kopioon tehdyt muutokset, mutta ei haluta korvata niillä alkuperäisen tiedoston tietoja. Tehdyt muutokset tallennetaan muualle versionhallintajärjestelmään myöhemmin hyödynnettäväksi ja branch:iin jää käytettäväksi Head. Jotta Stash-toiminto siirtää muutokset muualle ja alkuperäinen tiedostoversio jää voimaan, tulee Stash-komennon tapahtua ennen Check In -toimintoa. (Chacon & Straub 2014; TFS Tutorial 2020)

4 LÄHESTYMISTAPA JA TOTEUTUS

Tämän projektin lähestymistapa ja toteutus poikkesivat varsinaisesta tietovarastointi- ja Business Intelligence -projektista, sillä tavoitteena oli muokata toimeksiantajan toiveiden mukaisesti sille aiemmin luotua opintopolkua. Tosin opintopolun muutosten kartoittamisessa ja toteutuksessa täytyi huomioida varsinaisen tietovarastointi- ja Business Intelligence -projektin toimintatavat. Muutosten kartoittamista tässä projektissa helpotti asiakasprojektissa mukana oleminen, alkuperäisen opintopolun läpi käyminen työharjoittelussa ja toimeksiantajan kanssa käyty haastattelu ja keskustelut. Opintopolun muutosten valmistuttua, toimeksiantajan työntekijät tarkastelivat muutoksia ja antoivat muutoksia koskevaa palautetta.

4.1 Työvälineet

Tässä projektissa käytettiin toimeksiantajan ennalta määriteltyjä Microsoftin työvälineitä, joten tätä projektia varten ei tarvinnut tutustua eri tuoteperheiden tietovarastointi- ja Business Intelligence -työvälineisiin. Vaikka työvälineet tietokantojen hallintaa, ETL-prosessia, datan analysointia ja datan raportointia koskien olivat tuttuja jo työharjoittelusta, joutui tässä projektissa tutustumaan versionhallintajärjestelmä Git:n toiminnallisuuksiin Azure DevOpsin ympäristössä.

Alkuperäisessä opintopolussa opetellaan käyttämään tietovarastointi- ja business intelligence työvälineitä lukuunottamatta versionhallintaa, mikä oli yksi tärkeimmistä syistä tämän projektin toimeksiannolle. Versionhallinta on päivittäisessä käytössä asiakasprojekteissa, joten onnistuneen onboarding-prosessin tavoitteena oli opettaa opintopolun toteuttaja hyödyntämään versionhallintaa eri tilanteissa. Versionhallinnan merkitystä kasvatettiin projektin aikana siten, että jokainen vaihe opintopolussa on julkaistu myös versionhallintaan.

Toimeksiantaja oli ennen tämän projektin alkamista päättänyt ottaa käyttöön versionhallintajärjestelmä Git:n Azure DevOps:n ympäristössä. Tästä syystä tässä projektissa luotiinkin opintomateriaalia nimenomaan tähän ennalta määrättyyn

ympäristöön. Azure DevOps on Microsoftin tuotteistama versionhallintajärjestelmä. (Microsoft Docs 2020h)

Tietokantojen hallitsemiseen toimeksiantoyrityksessä hyödynnetään Microsoftin SQL Server Management Studiota, SSMS. SSMS:ään opintopolun aikana luoduista tietokannoista tehdään versionhallintaan tietokantaprojektit, jonka jälkeen niitä voidaan hyödyntää mahdollisten virheiden sattuessa. (Microsoft Docs 2020i)

Tietovarastoinnin olennaisena osana toteutettava ETL-prosessi toteutetaan opintopolussa Microsoftin SQL Server Data Tools:n, SSDT, avulla. SSDT toimii liitoksissa Microsoftin ohjelmointikehitysympäristön Visual Studion kanssa. ETL-prosessissa SSDT kommunikoi Visual Studion päällä toimivan SQL Server Integration Services:n, SSIS, kanssa. SSIS-työkalulla luodaan paketit, jotka siirtävät ja muokkaavat dataa tietovarastoon ja sen sisällä. SSIS-työkalulla luodut paketit talletetaan opintopolun aikana versionhallintaan, jossa niitä on myöhemmin helppo muokata. (Keary 2020)

SSDT:n kanssa liitoksissa Visual Studion päällä toimii myös SQL Server Analysis Services, SSAS, jolla toteutetaan opintopolun tabulaarinen kuutio. SSAS mahdollistaa myös multidimensionaalisten kuutioiden luomisen analysointityökaluillaan, mutta opintopolussa toteutetaan ainoastaan tabulaarinen kuutio. Opintopolun kuutio luodaan heti versionhallintaan ja siten onkin helposti muokattavissa niin, että eri versiot pysyvät tallessa. (Guru99 2020a)

Raportointi toteutetaan opintopolussa sekä staattisella että dynaamisella raportointityövälineellä. Staattisessa raportoinnissa työvälineenä toimii niin ikään Visual Studion päällä ja SSDT:n kanssa liitoksissa oleva Microsoftin SQL Server Reporting Services, SSRS. Dynaamisessa raportoinnissa sen sijaan hyödynnetään Microsoftin Power BI -työkalua. Power BI -työkalulla datasta saadaan luotua selkeitä ja dynaamisia visualisointeja asiakkaiden tarpeisiin. Kaikki opintopolun raportit luodaan heti versionhallintaan, sillä samoja raportteja saattaa muokata useampi henkilö ja versionhallinnassa estetään mahdollisten konfliktien syntyminen sekä raportit ovat aina helposti löydettävissä. (Guru99 2020b; Microsoft 2020b)

4.2 Opintopolun muutokset

Alkuperäisessä opintopolussa oli hyvä ja selkeä rakenne, mikä vastasi pitkälti asiakasprojektin rakennetta. Toimeksiantajalla oli silti herännyt tarve kehittää opintopolkua vastaamaan entistä enemmän asiakasprojektia, jotta uuden työntekijän olisi vaivatonta siirtyä asiakasprojektiin perehdytyksen jälkeen. Tavoitteena oli myös taata vaivaton siirtyminen asiakasprojektista toiseen, kun kaikki työskentelevät samalla tavalla. (Salonen 2020)

Toimeksiantajan tavoitteet vastaavat pitkälti englanninkielistä termiä onboarding, jossa työntekijää perehdytetään jo ennen työsuhteen alkamista yrityksen toimintatapoihin ja pyritään saamaan työntekijä kiinnostumaan työnteosta yrityksessä. Onboarding-prosessin toteuttaminen on ollut yritykselle hankalaa ja aikaa vievää. Opintopolun muutosten toteuttamisessa pidin onboarding-termin mielessäni, jotta työntekijöiden olisi jatkossa mahdollista toteuttaa opintopolkua myös lähes itsenäisesti samalla tuntematta turhautumista. Näin helpotettiin toimeksiantajan taakkaa kouluttaa uutta työntekijää, mutta samalla uusi työntekijä pääsee siirtymään tuottavaan työhön vaivattomasti. (Luoto 2012; Maurer 2020)

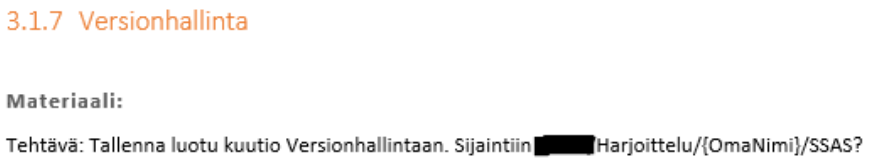
4.2.1 Asennusohjeet

Opintopolusta aiemmin erillään ollut sovellusten asentamiseen liittynyt ohje lisättiin opintopolun alkuun selkeyttämään kokonaisuutta. Asennusohjeita joutui myös päivittämään, sillä sovellukset olivat päivittyneet sen jälkeen, kun alkuperäistä opintopolkua varten ohjeet oli tehty. Todennäköisesti asennusohjetta joudutaan päivittämään jatkossakin, sillä sovellukset päivittyvät ja uusia työvälineitä kehitetään säännöllisen epäsäännöllisesti.

4.2.2 Versionhallinta

Versionhallinnan merkitys työvälineenä toimeksiantoyrityksessä on suuri, joten sen merkitystä haluttiin lisätä myös perehdytyksessä. Alkuperäisessä opintopolussa versionhallinnasta oli kuvan 5 mukainen tehtävä, jossa kehotettiin julkaisemaan tabulaarinen kuutio toimeksiantajan versionhallintaan sijaintiin:

Harjoittelu/{OmaNimi}/SSAS. Tehtävälle ei kuitenkaan ollut materiaalia, mikä teki erityisen hankalaksi itsenäisen työskentelyn.



Kuva 5 Alkuperäisen opintopolun materiaali ja tehtävä versionhallinnasta.

Päivitetyssä opintopolussa jokainen opintopolun vaihe lisätään versionhallintaan, joten versionhallinnasta lisättiin heti opintopolun alkuun materiaalia ja tehtäviä koskien versionhallintajärjestelmä Git:n toimintoja Azure DevOps:n ympäristössä. Kuvassa 6 on opintopolun alussa olevat materiaalit ja tehtävät versionhallintaa koskien. Tehtävänä on perehtyä Git:n yleisimpiin toimintoihin ja selittää termi repository, sillä näiden ymmärtäminen on opintopolulla etenemisen kannalta erittäin tärkeää. Ensimmäinen toiminnallinen tehtävä versionhallintaan liittyen on luoda paikallinen repository ja lisätä Opintopolku-tiedosto sinne, jotta siihen päivittäin tekemät muutokset eivät katoaisi.

2 Versionhallintajärjestelmä Git

MATERIAALI:

https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

https://www.atlassian.com/git/tutorials/what-is-version-control

https://docs.microsoft.com/en-us/azure/devops/repos/git/command-prompt?view=azure-devops

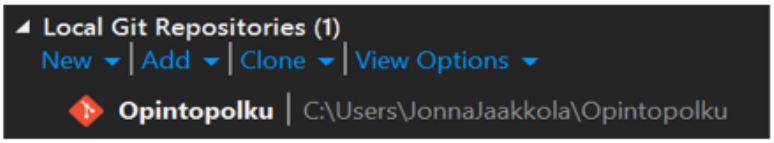
https://techterms.com/definition/repository

Versionhallinta on kehittäjien jokapäiväisessä käytössä ja siten onkin hyvä ymmärtää sen merkitys ja osata edes perus toiminnot.

TEHTÄVÄ: Tutustu ja selitä käsite repository (repo).

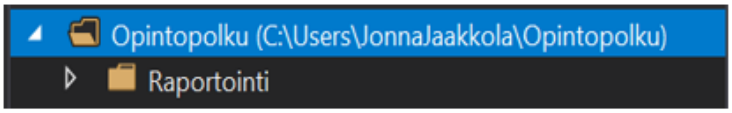
TEHTÄVÄ: Tutustu ja kuvaile seuraavat Git:n toiminnot: Add, Clone, Commit, Push, Pull, Fetch, Branch ja Stash.

TEHTÄVÄ: Luo Visual Studiossa Team Explorer ikkunassa paikallinen Git repo. Nimeä repo Opintopoluksi.



Kuva 1 Paikallinen repo luotu Visual Studiossa.

TEHTÄVÄ: Lisää Opintopolun alle kansio. Nimeä kansio Raportoinniksi. Lisää tämä Harjoitteluohjelma-toteutus tiedosto luomasi kansion alle. Jatkossa kaikki muutokset tähän tiedostoon tehdään versionhallinnassa.



Kuva 2 Solution Explorer-välilehdellä näkyy Opintopolun alle luotu raportointi kansio.

Kuva 6 Opintopolun alussa olevat materiaalit ja tehtävät koskien versionhallintaa.

Toinen toiminnallinen tehtävä versionhallintaa koskien on lisätä opintopolun aikana luoduista tietokannoista tietokantaprojektit versionhallintaan. Tämä toimintatapa ei ole ollut toimeksiantoyrityksessä yleisesti käytössä, mutta sitä toivottiin lisättäväksi opintopolkuun. Edellä mainittujen toiminnallisuuksien lisäksi versionhallintaan luodaan kaikki opintopolun raportit, tabulaarinen kuutio ja ETL-

prosessi, jotka olivat jo entuudestaan tuttuja toimintatapoja toimeksiantoyrityksessä lukuunottamatta Power BI -raporttien tallettamista. Siitä huolimatta, että versionhallinnan hyödyntäminen oli tuttua näissä toiminnoissa, ei alkuperäisessä opintopolussa versionhallintaan talletettu, kuin tabulaarinen kuutio.

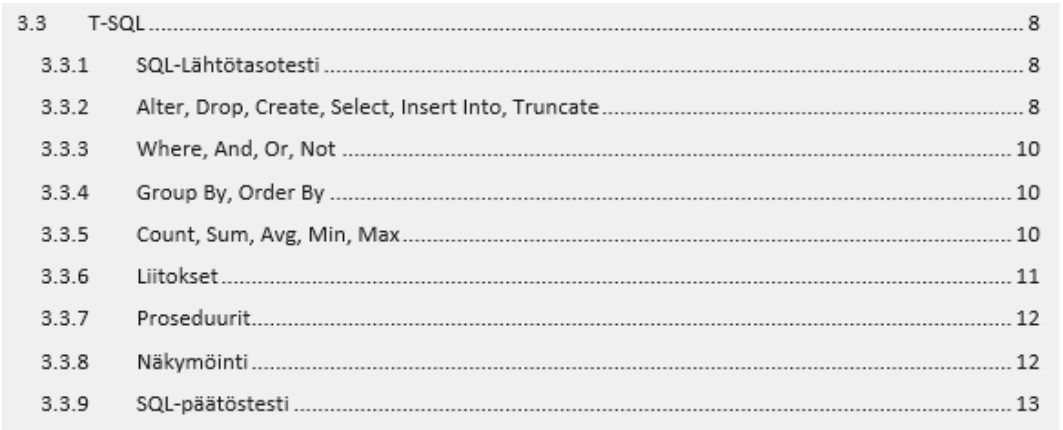
Perehtyjien lisäksi versionhallinnasta tarvittiin opintomateriaalia myös yrityksen työntekijöille, sillä toimeksiantoyritys oli päättänyt vaihtaa versionhallintajärjestelmänsä TFS:stä Git:iin, joka ei ollut kaikille työntekijöille entuudestaan tuttu. Opintopolkuun tehdyillä muutoksilla pyrittiin palvelemaan tästä syystä myös yrityksen työntekijöitä. Esimerkiksi työntekijöille uutena toimintatapana ja mahdollisuutena on versionhallinnan hyödyntäminen Power BI -raporttien tallettamisessa versionhallintaan.

4.2.3 Lähdejärjestelmät ja T-SQL

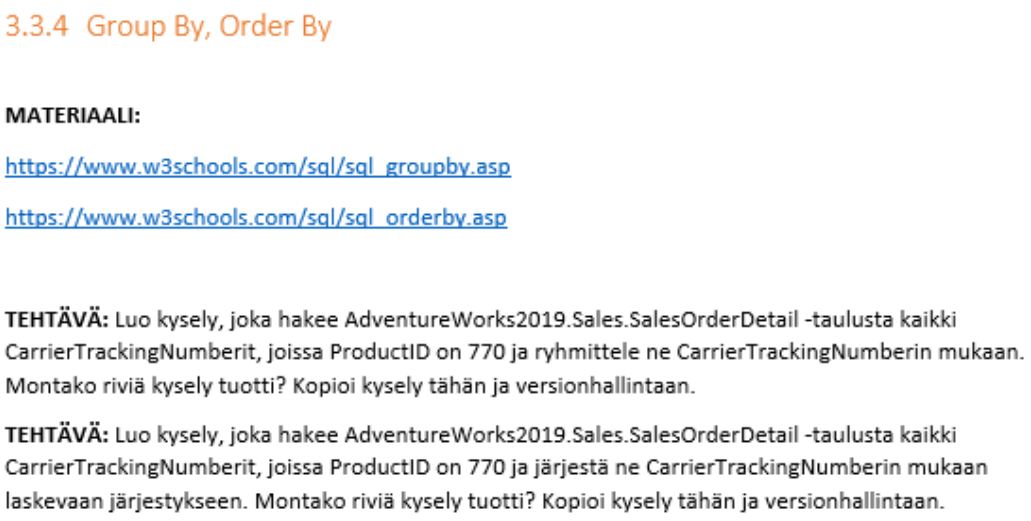
Alkuperäisessä opintopolussa lähdejärjestelmä luotiin itse ja samalla opittiin hyödyntämään T-SQL -kieltä erilaisissa tilanteissa. Transact-SQL, eli T-SQL, on Microsoftin ja Sybasen SQL -kielestä laajentama kokonaisuus, jossa on SQL -kielen perusominaisuuksiin lisätty muun muassa tapahtumien hallintaa, virhekäsittelyä ja rivien käsittelyä (Rouse 2019). Keskusteluissa toimeksiantajan edustajan kanssa kävi ilmi, että luodessa itse lähdejärjestelmää, siitä ei saa tarpeeksi haastavaa vastaamaan asiakasprojektia, joissa joku lähdejärjestelmä on aina olemassa. Usein toimeksiantajan asiakkaiden lähdejärjestelmät ovat monimutkaisia ja nissä on paljon dataa, joten myös opintopolun lähdejärjestelmissä haluttiin huomioida nämä ominaisuudet. Lähdejärjestelmän valinnassa päädyttiin hyödyntämään Microsoftin luomaa AdventureWorks2019 -mallitietokantaa. AdventureWorks2019 toimii sekä SQL Server:llä että Azure Database:ssa ja se on ladattavissa Microsoftin internetsivustolta backup-tiedostona omalle tietokoneelle (Microsoft 2020c). Koska tietokannan voi jokainen ladata omalle tietokoneelleen, on opintopolun itsenäinen tekeminen riippumaton toimeksiantajasta siltä osin. AdventureWorks2019 -tietokanta pitää sisällään useita tauluja ja dataa on paljon, mikä vastaa toimeksiantajan asiakkaiden lähdejärjestelmiä. Tosin AdventureWorks2019 -tietokanta on toimeksiantajan asiakkaiden lähdejärjestelmiä

selkeämpi ja yhdenmukaisempi, sekä siihen on luotu tarvittavat relaatiot jo valmiiksi. Näistä piirteistä huolimatta AdventureWorks2019 -tietokanta toimii lähdejärjestelmänä opintopolussa paremmin kuin itse luotu operatiivinen tietokanta, sillä datan tulisi olla myös entuudestaan tuntematonta ja sitä pitäisi olla paljon, jotta se vastaisi todellista asiakasprojektia mahdollisimman paljon.

Koska T-SQL -kielen merkitys pieneni lähdejärjestelmän muutoksessa, siitä luotiin oma opintokokonaisuutensa opintopolkuun. T-SQL -kielen osaaminen ja käyttäminen ovat toimeksiantoyrityksessä työntekijän kannalta merkittävä osa työntekoa, sillä jokaisessa asiakasprojektissa käytetään jollakin tapaa T-SQL -kieltä. T-SQL -osuus opintopolussa sisältää materiaalia ja tehtäviä T-SQL -kieleen liittyviin käskyihin, joihin toimeksiantoyrityksen asiakasprojekteissa törmätään. Kuvassa 7 on opintopolun T-SQL -osion sisältö otsikoittain. Opintopolussa jokaisen otsikon alla on materiaalia otsikon aiheista ja tehtäviä, joissa luodaan AdventureWorks2019 -tietokantaan kyseiseen aiheeseen liittyviä kyselyitä. Esimerkiksi kuvassa 8 on materiaalia ja tehtäviä aiheista Group By ja Order By. Tehtävinä on luoda T-SQL -kyselyt ja kertoa kuinka monta riviä kysely tuotti. Lisäksi tehtävissä kopioidaan luodut kyselyt versionhallintaan, jotta ne ovat myöhemminkin käytettävissä.



Kuva 7 T-SQL -osion sisältö otsikoittain



3.3.4 Group By, Order By

MATERIAALI:

https://www.w3schools.com/sql/sql_groupby.asp

https://www.w3schools.com/sql/sql_orderby.asp

TEHTÄVÄ: Luo kysely, joka hakee AdventureWorks2019.Sales.SalesOrderDetail -taulusta kaikki CarrierTrackingNumberit, joissa ProductID on 770 ja ryhmittele ne CarrierTrackingNumberin mukaan. Montako riviä kysely tuotti? Kopioi kysely tähän ja versionhallintaan.

TEHTÄVÄ: Luo kysely, joka hakee AdventureWorks2019.Sales.SalesOrderDetail -taulusta kaikki CarrierTrackingNumberit, joissa ProductID on 770 ja järjestä ne CarrierTrackingNumberin mukaan laskevaan järjestykseen. Montako riviä kysely tuotti? Kopioi kysely tähän ja versionhallintaan.

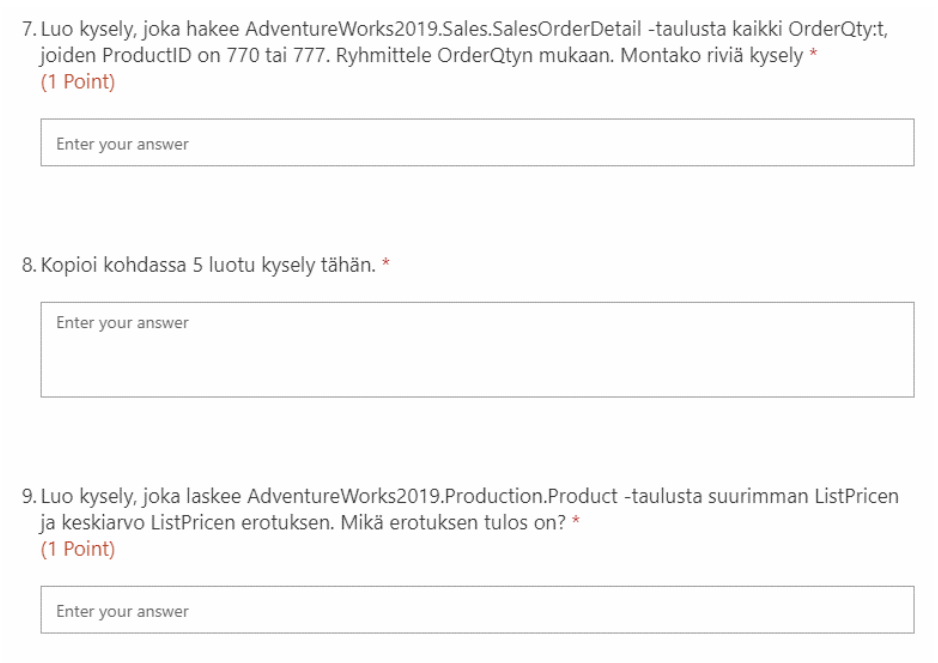
Kuva 8 T-SQL -osion Group By, Order By materiaalit ja tehtävät.

4.2.4 Testauspatteristo

Kuten kuvasta 7 huomataan, opintopolkuun lisättiin myös testivaiheita. Testivaiheilla haluttiin pystyä tarkastelemaan opintopolun toteuttajan etenemistä opintopolulla. Opintopolun testaus painotettiin nimenomaan T-SQL -kielen osaamiseen, sillä se nähtiin tärkeimmäksi osa-alueeksi kaikista. T-SQL -kielen testaaminen tapahtuu T-SQL -osiota ennen lähtötasotestinä ja T-SQL -osion jälkeen päätöstestinä. Testit toteutettiin Microsoftin Forms -työkalulla, sillä testi on helposti jaettavissa ja muokattavissa, sekä tulosten tarkastelu käy vaivattomasti.

Lähtötasotestillä opintopolun toteuttaja ja toimeksiantaja saavat hyvän kuvan toteuttajan osaamisesta T-SQL -kielessä. Lähtötasotesti sisältää 14 tehtävää, joissa pitää luoda AdventureWorks2019 -tietokantaan liittyen erilaisia kyselyitä ja antaa vastaus tai luotu kysely sen mukaan, mitä pyydetään. Vastaukseksi pyydetään joko itse kyselyä, kyselyn tulosta tai sitä, kuinka monta riviä kysely tuotti. Kyselyt vaikeutuvat, mitä isompaan tehtävänumeroon lähtötasotestissä mennään. Kuvassa 9 on esimerkit lähtötasotestin keskivaikeista tehtävistä. Kuvassa 9 nähdään myös, miten vastausta voidaan pyytää eri tavoilla. Lähtötasotestistä pisteitä saa, kun antaa oikean vastauksen parittomissa tehtävissä. Parillisissa tehtävissä pisteitä ei saada,

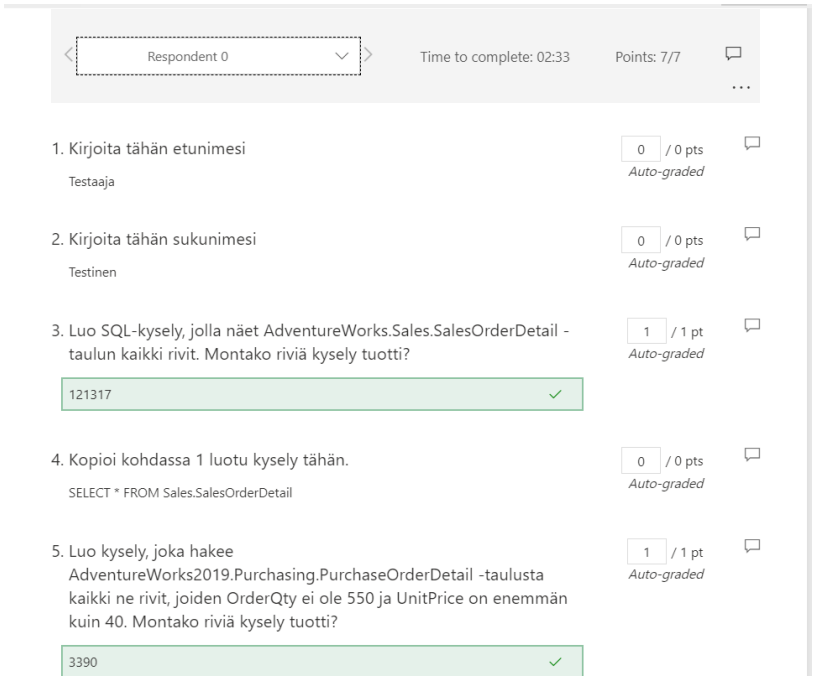
vaan niissä kyselyt kerätään talteen myöhempää tarkastelua varten. Jos parittomassa tehtävässä on vastannut väärin, pystyy toteuttaja selvittämään toimeksiantajan kanssa sen, missä on mennyt pieleen.



Kuva 9 T-SQL -lähtötasotestin keskivaikeita tehtäviä.

Päätöstesti sen sijaan tehdään vasta, kun T-SQL -osuus opintopolussa on suoritettu. Päätöstesti on kopio lähtötasotestistä, sillä sen avulla halutaan tarkastella, onko T-SQL -osuuden avulla kehitytty T-SQL -kielen käyttämisessä. Kuvassa 10 on testin toteuttajan nähtäville tuleva sivu, josta toteuttaja näkee, miten suoriutui testistä. Kuvassa 11 sen sijaan on toimeksiantajan näkymä samoista vastauksista. Toimeksiantaja voi tarkastella kuka testin on tehnyt, kauanko testiin meni aikaa ja millaisia vastauksia toteuttaja on antanut. Toimeksiantaja pystyy tarkastelemaan myös vastausten keskiarvoa ja useamman toteuttajan vastauksia kysymyskohtaisesti.

Kuva 10 Opintopolun T-SQL -päätöstestissä toteuttaja näkee oman tuloksensa, kun on lähettänyt testin vastaukset.



Kuva 11 Toimeksiantajan näkymä T-SQL -päätöstestin vastauksista.

4.2.5 Temp-schema / Working-kanta

Alkuperäisessä opintopolussa ETL-prosessissa oli materiaalia ja tehtäviä ainoastaan Temp-schemasta, mikä ei silti täysin vastannut toimeksiantajan toimintatapoja. Usealla toimeksiantajan asiakkaalla hyödynnetään Temp-scheman sijaan erillistä Working-tietokantaa ETL-prosessissa. Se, ettei working-tietokannasta ollut missään perehtyjien materiaaleissa mainintaa tästä toisesta tavasta, loi haastetta toimeksiantajalle. Sen takia päivitettyyn opintopolkuun lisättiin materiaaliin maininta siitä, että joissain asiakasprojekteissa voidaan käyttää Temp-scheman tilalla Working-tietokantaa.

4.2.6 Multidimensionaalinen kuutio

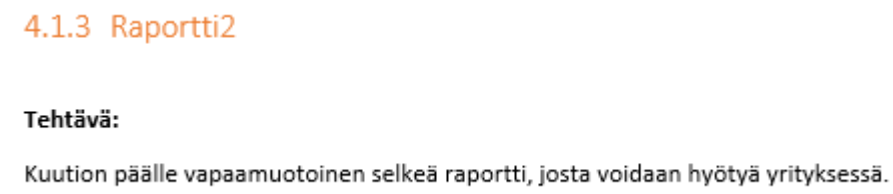
Toimeksiantoyrityksessä töitä tehdään sekä tabulaarisen kuution että multidimensionaalisen kuution parissa. Toimeksiantajan asiakasprojekteissa painopiste on selvästi enemmän tabulaarisen kuution puolella, mutta myös joissain asiakasprojekteissa käytetään multidimensionaalista kuutiota. Alkuperäisessä opintopolussa multidimensionaalisesta kuutiosta oli pieni tehtävä, jossa piti selvittää tabulaarisen kuution ja multidimensionaalisen kuution erot. Toimeksiantajan toiveesta opintopolkuun lisättiin materiaalia ja tehtävä multidimensionaalisen kuutioon liittyvästä MDX-kielestä. Tehtävässä pitää vertailla DAX-kielen ja MDX-kielen eroja.

4.2.7 Raportointi

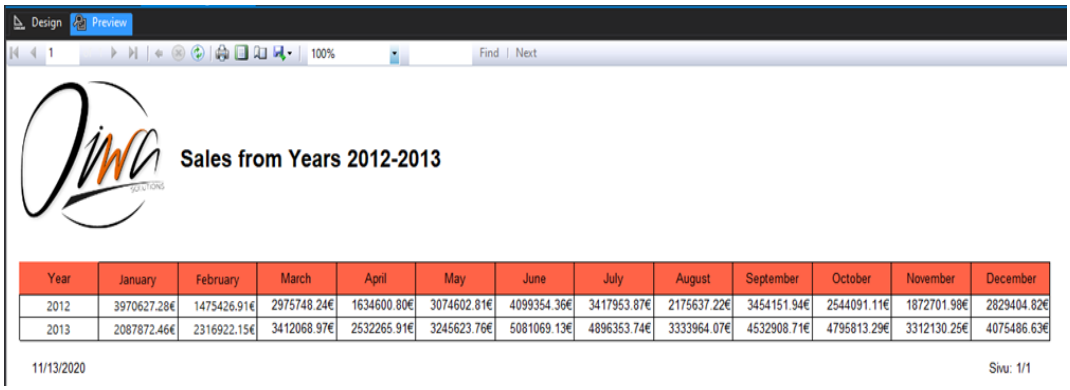
Aiemmin raportoinnin painopiste on ollut enemmän staattisissa SSRS-työkalulla tehdyissä raporteissa, mutta nykyään paino on siirtynyt enemmän dynaamisten Power BI -työkalulla tehtyjen raporttien puolelle. Tästä syystä opintopolkuun lisättiin selkeämmät tehtävät raportointiin liittyen. Alkuperäisessä opintopolussa ei ollut mitään kuvia eikä ohjeita siitä, mitä raporteille pitäisi tulla.

Kuvassa 12 on SSRS-työkalulla tehtävään raporttiin ohje alkuperäisestä opintopolusta. Ohje on ensikertalaiselle liian suppea. Henkilölle, joka ei ole koskaan ennen edes nähnyt SSRS -työkalulla tehtyä raporttia, on lähes mahdoton tehtävä luoda hyödyllinen raportti ilman esimerkkiä. Niimpä päivitettyyn

opintopolkuun lisättiin myös SSRS-työkalulla tehtävistä raporteista esimerkit, vaikka painopiste onkin siirtynyt enemmän dynaamiseen raportointiin. Kuvassa 13 on päivitetyn opintopolun esimerkkiraportti. Raportti on osa tehtävänantoa, jossa neuvotaan luomaan kuvan mukainen raportti AdventureWorks2019-tietokannan datasta. Raportti on pelkistetty kokonaisuus siitä, mitä toimeksiantoyrityksessä tehdään, mutta silti siinä pitää osata hyödyntää erilaisia muotoiluja ja toimintoja SSRS-työkalusta. Kuvan 13 raportin lisäksi päivitettyyn opintopolkuun lisättiin esimerkki SSRS-raportista, joka luodaan opintopolussa aiemmin luodun kuution datasta.



Kuva 12 Alkuperäisen opintopolun tehtävä SSRS-työkalulla luotavista raporteista.



Year	January	February	March	April	May	June	July	August	September	October	November	December
2012	3970627.28€	1475426.91€	2975748.24€	1634600.80€	3074602.81€	4099354.36€	3417953.87€	2175637.22€	3454151.94€	2544091.11€	1872701.98€	2829404.82€
2013	2087872.46€	2316922.15€	3412068.97€	2532265.91€	3245623.76€	5081069.13€	4896353.74€	3333964.07€	4532908.71€	4795813.29€	3312130.25€	4075486.63€

Kuva 13 Päivitetyn opintopolun SSRS-työkalulla luotu raportti.

Toimeksiantaja toivoi, että raportoinnissa painotettaisiin dynaamista Power BI -raportointia. Alkuperäisessä opintopolussa ei ollut esimerkkejä SSRS-raporttien tavoin Power BI -raporteista, vaan kuvan 14 mukainen tehtävänanto. Tästä syystä opintopolkuun lisättiin kolme selkeää esimerkkiraporttia. Raporteista kaksi on tehty

tietokannan datasta ja yksi kuution datasta. Kuvassa 15 on toinen tietokannan päälle luoduista raporteista. Raportti on osa tehtävänantoa, jossa kehotetaan luomaan kuvan 15 mukainen raportti AdventureWorks2019-tietokannan datasta. Raportissa on hyödynnetty tiettyjä muotoiluja ja visualisointeja, joita myös toimeksiantajan asiakasprojekteissa hyödynnettäisiin.

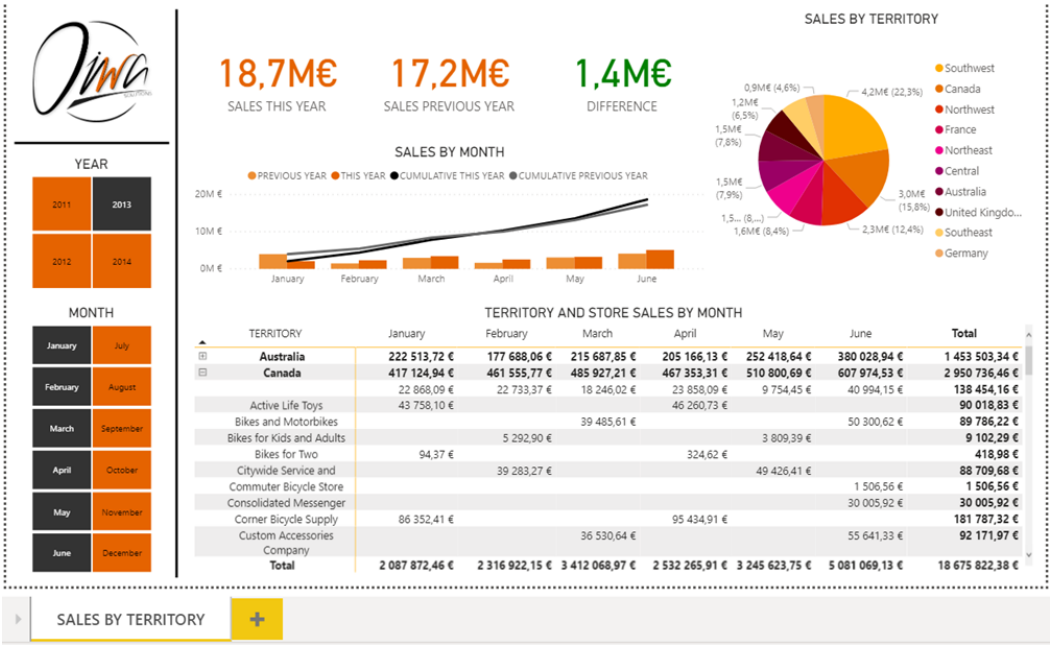
4.2.1 Kuution päälle

Tehtävä:

Mallinnus ja mittarit kuutiosta -> Datan visualisointi

Tehtävä: Tee monipuolinen mutta selkeä raportti käyttäen erilaisia elementtejä

Kuva 14 Alkuperäisen opintopolun tehtävä koskien Power BI -raportointia.



Kuva 15 Päivitetyn opintopolun Power BI -raportti AdventureWorks2019-tietokannan datasta.

5 JOHTOPÄÄTÖKSET JA POHDINTA

Tämän projektin lopputuloksena syntyi toivotulla tavalla päivitetty ja perehdytystä paremmin tukeva opintopolku. Opintopolun muutokset on toteutettu täysin toimeksiantajan toiveiden mukaisesti. Toiveissa oli lisätä versionhallinnan merkitystä opintopolussa ja ylipäänsä tukea lisää uuden työntekijän tuottavaan työhön pääsemistä eli onboarding-prosessia. Toimeksiantajan toiveet toteutettiin, mutta itselleni heräsi ajatuksia kehittää vielä tiettyjä kohtia, jotka jäävät seuraavien kehittäjien tehtäväksi.

Kokonaisuudessaan projektin toteutus sujui hyvin. Aloitin projektin toteutuksen heti suoritettuani työharjoitteluni toimeksiantoyrityksessä. Projektin etenemistä edisti tutustuminen alkuperäiseen opintopolkuun työharjoittelussa ja toimeksiantajan asiakasprojektissa toteuttajana toimiminen. Varsinainen toteutus tapahtui Covid19-viruksen aiheuttamassa poikkeustilassa itsenäisesti kotona, mutta apua sai aina tarvittaessa toimeksiantajalta Microsoft Teams:n välityksellä, mikä mahdollisti tehokkaan toteutuksen. Esittelin opintopolkuun tehdyt muutokset koko toimeksiantajan työporukalle ja sain palautetta niin uudelta työntekijältä, kuin talossa pitkään töitä tehneiltä. Palaute oli yhtä poikkeusta lukuunottamatta hyvää. Negatiivinen palaute koski tehtävää, jossa yhtä ETL-prosessissa käytettävää työkalua ei ollut tarpeeksi selkeästi kuvattu tehtävänannossa. Heti palautteen kuultuani lisäsin tehtävästä kuvallisen selityksen kyseisen työkalun tehtävästä.

Koska kaikki esille nousseet epäkohdat on korjattu, palvelee opintopolku tässä hetkessä juuri niin hyvin, kuin se tässä tilanteessa voi palvella niin toimeksiantajaa, kuin myös uusia työntekijöitä. Opintopolun muutosten merkitys on erittäin suuri toimeksiantajalle, sillä ilman tehtyjä muutoksia perehtyjät eivät pääsisi yhtä nopeasti tehokkaaseen työntekoon kiinni. Tehokas ja kattava perehdyttäminen säästää toimeksiantajalta eri resursseja, joista selkeästi merkittävimpinä rahaa ja aikaa. Opintopolkuun tehdyillä muutoksilla saavutettiin hyvin pitkälti saumaton siirtyminen asiakasprojekteihin, mikä on myös perehtyjälle mukavempi.

Tähän opinnäytetyöhön löytyi paljon materiaalia ja useimmat materiaalit tukivat toisiaan. Versionhallinta toimi tässä opinnäytetyössä suuressa roolissa ja siitä löytyi

kirjallisuuden lisäksi paljon materiaalia verkosta. Muihin opinnäytetyössä nousseisiin aiheisiin olisi ollut materiaali painetuissa kirjoissa, mutta Covid19-viruksen takia kirjastot suljettiin, eikä painettua materiaalia ollut saatavissa. Tästä syystä materiaali täytyi etsiä internetistä. Siitä huolimatta luotan lähteiden oikeellisuuteen ja luotettavuuteen, koska lähteet tukevat toisiaan. Jos on käytetty vain yhtä lähdettä, olen pitänyt lähdettä riittävän luotettavana kirjailijan tai artikkelin kirjoittajan taustan vuoksi.

Suurimmaksi haasteeksi tässä opinnäytetyössä koitui rajallinen aika. Varsinaisesta opinnäytetyön toteuttamisesta aikaa söi toimeksiantajan asiakasprojektissa mukana oleminen. Tosin se oli kriittinen osa opintopolun toteuttamista, sillä ilman asiakasprojektissa mukana olemista en olisi ymmärtänyt tiettyjen vaiheiden merkitystä ja muutosten tärkeyttä. Asiakasprojektista asiakkaalta saatu hyvä palaute lisäsi myös itsellä halua luoda selkeä ja kattava opintopolku, jotta seuraavat uudet työntekijät voisivat kokea yhtä palkitsevan tunteen.

Toiseksi suureksi haasteeksi koitui aiheen rajaaminen niin, että näkökulma tukisi toimeksiantajan lisäksi muita aiheesta kiinnostuneita. Tähän teemaan liittyy vahvasti myös se, mitä opinnäytetyössä voidaan näyttää, ettei paljasteta liikesalaisuuksia. Koen, että pystyin luomaan selkeän ja selkokielisen opinnäytetyön aiheesta, joka koskettaa jollain tavalla jokaista, joka joko hakee uuteen työpaikkaan töihin tai toimii perehdyttäjänä. Aihe toimii myös erinomaisesti eri aloille, sillä versionhallintaa voidaan hyödyntää kaikkialla, missä tiedostoja ja niiden versioita halutaan tallentaa. Tässä toteutuksessa painopiste on nimenomaan Business Intelligence:ssä, koska toimeksiantaja toimii sillä alalla.

Tämän opinnäytetyöprojektin avulla pystyn vastaamaan tutkimuskysymyksiin, jotka nousi esille toimeksiantajan tarpeiden määrittelyssä. Tutkimuskysymyksistä kolme liittyi perehtymiseen. Ensimmäisessä pohdin, mitä opintopolkuun tulisi lisätä, jotta se vastaisi mahdollisimman paljon asiakasprojektia. Toteuttaessani oikeaa asiakasprojektia toimeksiantajalle ymmärsin, kuinka tärkeää opintopolkuun on lisätä realistisempia lähdejärjestelmiä, sillä lähdejärjestelmän data määrittelee lopullisen tuotteen eli raportin sisältöä suuresti. Opintopolkuun lisätyt pienemmät

muutokset koskien esimerkiksi MDX-kieltä ovat myös tärkeitä lisäyksiä, sillä joissain asiakasprojekteissa voi kyseisiin aiheisiin törmätä. Toinen tutkimuskysymys koski onboarding-prosessia. Onboarding-prosessin onnistuminen taataan opintopolussa laadukkailla ja asianmukaisilla materiaaleilla ja tehtävillä, sekä testauspatteristolla. Opintopolun suorittaminen mahdollistaa toimeksiantoyrityksen tapojen omaksumisen jo hyvissä ajoin selkeiden ohjeiden avulla, mikä on yksi onboarding-prosessin keskeisimmistä piirteistä. Kolmanteen tutkimuskysymyseen vastaus on testauspatteristo. Kysymyksenä oli, kuinka perehtymistä voidaan seurata ja testata. Nyt opintopolkuun luotu testauspatteristo helpottaa toimeksiantajaa seuraamaan perehtyjän etenemistä opintopolulla, mutta testauspatteristoon olisi hyvä jatkossa lisätä tehtäviä myös esimerkiksi raportointiin liittyen. Viimeinen tutkimuskysymys koski versionhallintaa ja siinä haluttiin tietää, mikä Git on ja kuinka siitä voitaisiin hyötyä yrityksessä. Versionhallinta oli jo entuudestaan käytössä toimeksiantoyrityksessä, mutta nyt haluttiin tietää nimenomaan Git:in hyödyistä. Opintopolkuun lisätyt muutokset eivät ole pelkästään perehtyjiä varten, vaan myös toimeksiantoyrityksen omille työntekijöille. Opintopolun alkuun on lisätty materiaalia pelkästään Git:in toiminnoista, mutta opintopolun edetessä näitä toimintoja hyödynnetään eri tavoin. Opintopolussa on säilytetty vanhoja tapoja, joissa versionhallintaa on hyödynnetty. Tämän lisäksi siihen on lisätty tapoja, joista jatkossa voitaisiin hyötyä toimeksiantoyrityksessä. Yhtenä esimerkkinä uusista toimintatavoista toimii Power BI -raporttien tallettaminen versionhallintaan. Toimeksiantoyrityksen näkökulmasta vastaten Git on TFS:n korvaajana merkittävä työkalu asiakasprojekteissa ja sitä voidaan jatkossa hyödyntää totuttua laajemmin.

Minulle nousi esiin myös jatkokehitysideoita tätä opinnäytetyötä tehdessä. Toimeksiantajan puolelta kehitysideana olisi esimerkiksi luoda oma opintopolkunsa ohjelmistokehittäjille, sillä kasvavassa yrityksessä tavoitteena on palkata myös business intelligence:n osaajien lisäksi ohjelmistokehittäjiä. Tosin myös nyt luotu opintopolku tarvitsee päivittää varmasti muutaman vuoden sisällä, sillä työssä käytettävät työvälineet kehittyvät jatkuvasti, mikä samalla aiheuttaa muutoksia myös toimintatavoissa. Opintopolussa olisin halunnut luoda testivaiheen myös opintopolun loppupuolelle, jossa luotiin esimerkiksi raportteja, mutta

valitettavasti aika ei siihen riittänyt. Siinäkin on siis seuraavalle kehittäjälle idea kehittää testauspatteristoa eteenpäin.

Pelkästään tähän opinnäytetyöhön liittyvä jatkokehitysidea olisi luoda eri alan näkökulmasta vastaava kokonaisuus. Eri aloilla on eri tavat ja versionhallinta taipuu todella moneen eri näkökulmaan erittäin hyvin. Myös eri versionhallintajärjestelmät antavat variaation mahdollisuuden.

Henkilökohtaisesti tällä opinnäytetyöllä on ollut valtava merkitys ammatillisessa kehittymisessä. Teoriaan tutustuminen avarsi omaa ymmärrystäni versionhallinnan merkityksestä erityisesti kehitystyössä. Lisäksi toteutin asiakasprojektia itsenäisesti, mikä auttoi luomaan enemmän asiakasprojektia vastaavan opintopolun toimeksiantajalle. Opin toimimaan asiakkaiden parissa, sekä tekemään ryhmätyötä toisen toteuttajan kanssa. Mikä tärkeintä, onnistuin täyttämään asiakasprojektissa asiakkaan toiveet ja opintopolun muutoksissa onnistuin täyttämään myös toimeksiantajan toiveet. Tämän opinnäytetyön jälkeen koen olevani paljon itsevarmempi työtehtävissäni.

LÄHTEET

Ahmad, S. 2020. The Future of GIT (2020). Viitattu 9.10.2020. https://towardsdatascience.com/the-future-of-git-2020-8e33b2f8746d

Anstett, T. 2020. How Will Git Develop in the Future? Viitattu 9.10.2020. https://www.k15t.com/blog/2019/02/how-will-git-develop-in-the-future

Apiumhub. 2020. Benefits of Using Github. Viitattu 14.10.2020. https://apiumhub.com/tech-blog-barcelona/using-github/

Atlassian. 2020. What is Version Control: centralized vs. DVCS. Viitattu 9.10.2020. https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs

Atlassian Bitbucket. 2020a. What is Version Control. Viitattu 2.10.2020. https://www.atlassian.com/git/tutorials/what-is-version-control

Atlassian Bitbucket 2020b. Tutorials – Git Clone. Viitattu 8.10.2020. https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone

Atlassian Bitbucket 2020c. Tutorials – Git Merge. Viitattu 8.10.2020. https://www.atlassian.com/git/tutorials/using-branches/git-merge

Azad, K. 2007. A Visual Guide to Version Control. Viitattu 6.10.2020. https://betterexplained.com/articles/a-visual-guide-to-version-control/

Brun, J. 2012. Version Control – Git Basics. Viitattu 7.10.2020. https://nceas.github.io/oss-lessons/version-control/1-git-basics.html

Carstensen, W. 2016. A Brief History of Version Control. Viitattu 9.10.2020. https://www.red-gate.com/blog/database-devops/history-of-version-control

Chacon, S; Straub B. 2014. Pro Git. 2nd Edition. Apress.

Cool, J. 2019. Now available: Azure DevOps Server 2019. Viitattu 12.10.2020. https://azure.microsoft.com/en-us/blog/now-available-azure-devops-server-2019/

Dolan, M. 2016. Introduction to Git and GitHub. Slideshow page 19. Viitattu 14.10.2020. https://www.slideshare.net/bcbbslides/introduction-to-git-and-github-72514916

Dott, E. 2020. Commit Messages for Source Control Management Systems. Viitattu 8.10.2020. https://dev.to/elmardott/commit-messages-for-source-control-management-systems-56oj

Ernst, M. 2018. Version Control Concepts and Best Practices. Viitattu 7.10.2020. https://homes.cs.washington.edu/~mernst/advice/version-control.html

Gaël, T. 2020. A Beginner's Guide to Git - What is a Changelog and How to Generate it. Version 9.1.0. Viitattu 8.10.2020. https://www.freecodecamp.org/news/a-beginners-guide-to-git-what-is-a-changelog-and-how-to-generate-it/

GitHub Docs. 2020. Cloning a Repository. Viitattu 8.10.2020. https://docs.github.com/en/free-pro-team@latest/github/creating-cloning-and-archiving-repositories/cloning-a-repository

GitHub. 2020. Git Guides – Git add. Viitattu 8.10.2020. https://github.com/git-guides/git-add

Glancy, J. 2020. The Advantages and Disadvantages of Using GitHub. Viitattu 14.10.2020. https://www.codeclouds.com/blog/advantages-disadvantages-using-github/

Gregory, G. 2011. Trunk vs. HEAD in Version Control Systems. Viitattu 8.10.2020. https://garygregory.wordpress.com/2011/02/03/trunk-vs-head-in-version-control-systems/

Guru99. 2020a. SSAS Tutorial: What is, Architecture, SSAS Cube & Types. Viitattu 26.11.2020. https://www.guru99.com/ssas-tutorial.html

Guru99. 2020b. SQL Server Reporting Services (SSRS) Tutorial for Beginners. Viitattu 26.11.2020. https://www.guru99.com/ssrs-tutorial.html

Günther, T. 2014. Understanding Version Control with Diffs. Viitattu 8.10.2020. https://www.sitepoint.com/understanding-version-control-diffs/

Hay, W. 2018. The Architecture and History of Git: A Distributed Version Control System. Viitattu 14.10.2020. https://medium.com/@willhayjr/the-architecture-and-history-of-git-a-distributed-version-control-system-62b17dd37742

Heise Medien. 2020. GitHub populärer als SourceForge und Google Code. Viitattu 13.10.2020. https://www.heise.de/developer/meldung/GitHub-populaerer-als-SourceForge-und-Google-Code-1255416.html

Hovi, A. 2020. Data-alan termien selitykset ja kuvaukset. Viitattu 6.10.2020. https://www.arihovi.com/materiaalit/datapedia-data-alan-termit-avattuna/

Intellipaat. 2011-2020. What is Git. Viitattu 14.10.2020. https://intellipaat.com/blog/what-is-git/

Ite Wiki. 2020. Digitalisoinnin opas. Viitattu 5.10.2020. https://www.itewiki.fi/opas/bi-business-intelligence-ja-raportointi/

JetBrains. 2000-2020. Resolve Conflicts. Viitattu 8.10.2020. https://www.jetbrains.com/help/idea/resolving-conflicts.html

Kaario, K; Peltola, T. 2008. Tiedonhallinta - Avain tietotyön tuottavuuteen. 1. painos. Porvoo. WSOYpro/Docendo-tuotteet.

Keary, T. 2020. What is Microsoft SSIS? Viitattu 26.11.2020. https://www.comparitech.com/net-admin/what-is-microsoft-ssis/

Kumar, V. 2019. How does GitHub work? Answer. Viitattu 13.10.2020. https://www.quora.com/How-does-GitHub-work

Lacan, O. 2017. Keep a Changelog. Version 1.0.0. Viitattu 8.10.2020. https://keepachangelog.com/en/1.0.0/

Larabel, M. 2016. BitKeeper Version Control Released As Open-Source. Viitattu 9.10.2020. https://www.phoronix.com/scan.php?page=news_item&px=BitKeeper-Open-Source

LinkedIn Corporation. 2020. The History of Version Control. Viitattu 9.10.2020. https://www.lynda.com/ALMTFS-tutorials/history-version-control/106788/115979-4.html

Lubański, M. 2019. Centralized vs Distributed Version Control Systems. Viitattu 9.10.2020. https://medium.com/faun/centralized-vs-distributed-version-control-systems-a135091299f0

Luoto, L. 2012. Ajattele perehdytys uudelleen. Viitattu 30.9.2020. https://www.psycon.fi/blogi/ajattele-perehdytys-uudelleen

MadCap Flare. 2020. Checking Out Source Control Files—Team Foundation Server. Viitattu 8.10.2020. https://help.madcapsoftware.com/flare2020/Content/Flare/Source-Control/TFS/Process/Checking-Out-Source-Control-Files-TFS.htm#HowtoCheckOutFilesFromSourceControlManually

Mar, W. 2020. Microsoft TFS vs. Git and GitHub. Viitattu 12.10.2020. https://wilsonmar.github.io/tfs-vs-github/

Marrocos, M. 2020. Azure DevOps or GitHub. Viitattu 9.10.2020. https://medium.com/devops-cloud-it-career/azure-devops-or-github-c83fe1eced4d

Maurer, R. 2020. New Employee Onboarding Guide. Viitattu 6.10.2020. https://www.shrm.org/resourcesandtools/hr-topics/talent-acquisition/pages/new-employee-onboarding-guide.aspx

Microsoft. 2020a. ASP.NET. Viitattu 13.10.2020. https://dotnet.microsoft.com/apps/aspnet

Microsoft. 2020b. Mikä Power BI on? Viitattu 26.11.2020. https://docs.microsoft.com/fi-fi/power-bi/fundamentals/power-bi-overview

Microsoft. 2020c. AdventureWorks sample databases. Viitattu 3.12.2020. https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms

Microsoft Corporation. 2015. Team Foundation Server (TFS) Architecture. Viitattu 13.10.2020. https://social.technet.microsoft.com/wiki/contents/articles/16664.team-foundation-server-tfs-architecture.aspx

Microsoft Docs 2020a. How to: Rename Elements in the Version Control System. Viitattu 7.10.2020. https://docs.microsoft.com/en-us/dynamicsax-2012/developer/how-to-rename-elements-in-the-version-control-system

Microsoft Docs 2020b. How To: Check Elements Out of the Version Control System. Viitattu 7.10.2020. https://docs.microsoft.com/en-us/dynamicsax-2012/developer/how-to-check-elements-out-of-the-version-control-system

Microsoft Docs 2020c. How to: Check Elements into the Version Control System. Viitattu 7.10.2020. https://docs.microsoft.com/en-us/dynamicsax-2012/developer/how-to-check-elements-into-the-version-control-system

Microsoft Docs 2020d. How to: Synchronize Your Local Repository with the Version Control System. Viitattu 8.10.2020. https://docs.microsoft.com/en-us/dynamicsax-2012/developer/how-to-synchronize-your-local-repository-with-the-version-control-system

Microsoft Docs 2020e. Roll back changesets. Viitattu 8.10.2020. https://docs.microsoft.com/en-us/azure/devops/repos/tfvc/roll-back-changesets?view=azure-devops

Microsoft Docs 2020f. Resolve Team Foundation Version Control conflicts. Viitattu 8.10.2020. https://docs.microsoft.com/en-us/azure/devops/repos/tfvc/resolve-team-foundation-version-control-conflicts?view=azure-devops

Microsoft Docs 2020g. Understand Lock Types. Viitattu 8.10.2020. https://docs.microsoft.com/en-us/azure/devops/repos/tfvc/understand-lock-types?view=azure-devops

Microsoft Docs 2020h. Adopting Team Explorer Everywhere. Viitattu 12.10.2020. https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2013/gg413285(v=vs.120)?redirectedfrom=MSDN

Microsoft Docs 2020i. Download SQL Server Management Studio (SSMS). Viitattu 26.11.2020. https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

National Instruments Corporation. 2020. Renaming Files in Source Control. Viitattu 7.10.2020. http://zone.ni.com/reference/en-XX/help/371361R-01/lvhowto/renaming_files_control/

Ojanen, R. 2019. Tietovarastointi- ja Business Intelligence -ratkaisut Case: Oiwa Solutions Oy – opintopolku. Opinnäytetyö, alempi AMK. Vaasan Ammattikorkeakoulu, Tietojenkäsittely, liiketalouden koulutusohjelma. Bokfix Oy.

Perforce Software. 2020a. What is SVN (Subversion). Viitattu 9.10.2020. https://www.perforce.com/blog/vcs/what-svn

Perforce Software. 2020b. What is Team Foundation Server. Viitattu 9.10.2020. https://www.perforce.com/blog/vcs/what-team-foundation-server

Perforce Software. 2020c. What is Version Control. Viitattu 9.10.2020. https://www.perforce.com/blog/vcs/what-is-version-control

Redfox Languages Oy. 2014-2020. Branching englannista suomeksi. Viitattu 8.10.2020. https://redfoxsanakirja.fi/fi/sanakirja/-/s/eng/fin/branching

Rouse, M. 2011. Concurrent Versions System (CVS). Viitattu 9.10.2020. https://whatis.techtarget.com/definition/Concurrent-Versions-System-CVS

Rouse, M. 2019. T-SQL (Transact-SQL). Viitattu 3.12.2020. https://searchsqlserver.techtarget.com/definition/T-SQL

Salonen, T. 2020. Architect, BI & Data. Oiwa Solutions Oy. Haastattelu 25.9.2020.

Schiestl, B. 2020. Branching Definition – What is a Branch. Viitattu 8.10.2020. https://www.perforce.com/blog/vcs/branching-definition-what-branch

Siitonen, E. 2020. Perehdytys kuuluu kaikille. Viitattu 30.9.2020. https://www.tehy.fi/fi/blogi/perehdytys-kuuluu-kaikille

Sinc, E. 2011. Version Control by Example. Viitattu 2.10.2020. https://ericsink.com/vcbe/html/intro.html

TechoPedia. 2020. Merge. Viitattu 8.10.2020. https://www.techopedia.com/definition/1217/merge

TFS Tutorial. 2020. Source Control Shelving and Unshelving. Viitattu 8.10.2020. http://www.tfstutorial.com/shelving-and-unshelving

The Linux Foundation. 2020. 10 Years of Git: An Interview with Git Creator Linus Torvalds. Viitattu 9.10.2020. https://www.linuxfoundation.org/blog/2015/04/10-years-of-git-an-interview-with-git-creator-linus-torvalds/

Tieteen Termipankki. 2020. Raakadata. Viitattu 6.10.2020. https://tieteentermipankki.fi/wiki/Nimitys:raakadata

TrustRadius. 2013-2020. Azure DevOps Server (formerly TMS) Reviews. Viitattu 14.10.2020. https://www.trustradius.com/products/azure-devops-server/reviews?qs=pros-and-cons

Wildbit. 2007-2019. An Introduction to Version Control. Viitattu 8.12.2020. http://guides.beanstalkapp.com/version-control/intro-to-version-control.html

LIITE 1

OPINTOPOLUN MUUTOSTEN KARTOITUS

Toimeksiantaja: Oiwa Solutions Oy

Haastateltava: Teemu Salonen, Arkkitehti, BI & Data

Haastattelu pv: 25.9.2020

Kysymykset:

1. Opintopolun nykytilanne? (Kohderyhmä, ympäristö, testaus)
2. Miksi opintopolkua halutaan päivittää?
3. Millä keinoilla varmistetaan, että pystyn päivittämään opintopolkua oikein?
4. Mitä opintopolkuun tulisi lisätä tai mitä siitä tulisi poistaa?
5. Mihin ympäristöön/ympäristöihin opintopolku toteutetaan?
6. Miten voin testata opintopolun toimivuutta?
7. Milloin opintopolun tulisi olla valmis?

Vastaukset:

1. Opintopolussa jo osin vanhentunutta asiaa, joka pitäisi päivittää. Opintopolussa on myös jotkin osuudet hieman suppeasti kuvattu ja niihin tarvitaan lisää laajuutta, esim. versionhallinta, joka on kuitenkin kehitystyössä merkittävässä roolissa.
2. Opintopolkua halutaan päivittää, jotta se palvelisi uutta työntekijää ja harjoittelijaa mahdollisimman hyvin ja kattavasti. Varsinaisiin asiakasprojekteihin siirtyminen olisi näin ollen helppoa ja vaivatonta. Myös asiakasprojektien välillä siirtyminen olisi helppoa ja vaivatonta, jos kaikki projektit olisi tehty samalla logiikalla.
   Tarve myös kehittää ja hioa omat prosessit vastaamaan kaikilta osin opintopolkua.
3. Riittävällä ohjauksella varmistetaan, että opintopolku vastaa Oiwan laatustandardeja ja yhteisiä käytäntöjä. Ohjaavan henkilön varattava kalenterista aikaa ohjaukselle.
4. Tässä on useampiakin kehityskohteita:

- Operatiivisena järjestelmänä voitaisiin käyttää esim. Microsoftin AdventureWorks-tietokantaa. Siellä dataa on laajasti ja se on enemmän aidon asiakasympäristön näköinen. Operatiivisena järjestelmänä voisi olla osana myös erilliset tiedostot, esim. CSV tai XLSX. Tämä kuitenkin rajattava suhkot tiukasti, että saadaan järkevässä aikataulussa toteutettua. Valitaan lähdejärjestelmistä 5-10 taulua, joilla opintopolun alku toteutetaan.
- Temp-scheman lisäksi selkeästi Working-kannan käyttö.
- Versionhallinta laajemmin. Esitelläänkö jopa molemmat TFS ja Git. Nykyään kyllä käytössä enemmänkin pelkkä Git.
- Ohjeessa on paljon ilmaan jääviä kysymyksiä. Pitäisikö Opintopolun yhteydessä olla joku testauspatteristo? Tuleeko opintopolkua käyvän henkilön muuten perehdyttyä tarpeeksi aiheeseen ja sen jälkeen olevaan kysymykseen?
- Kuutioissa OLAP (multidimensionaalinen) kuutio saa jäädä pienemmälle osuudelle, kuin Tabulaarinen. Nykyään tehdään lähes poikkeuksetta tabulaarinen kuutio asiakkaan tarpeisiin. Tähän liittyen on DAX-kuvattu, mutta MDX:ää ei. Pieni kappale myös MDX:stä.
- Raportoinnissa myös vähemmän painoa SSRS ja lisätään PowerBI-osuutta. Tehdään selkeästi jo vähän haastavampi PowerBI-raportti.

5. Opintopolku toteutetaan joko lokaalisti omalle koneelle tai Azure ympäristöön. Omalle koneelle toteutettaessa ei tarvitse huolehtia oikeuksista ja nettiyhteyksistä. Opintopolun seuraaminen olisi näin ollen mahdollista missä puitteissa tahansa.
6. Opintopolku auditoidaan kokeneen (senior/arkkitehti) ja kokemattoman (harjoittelija/uusi työntekijä) henkilön toimesta. Kokenut henkilö huomaa selkeät asiavirheet ja epäjohdonmukaisuudet, mutta kokematon henkilö huomaa liian suppean tai puuttellisen ohjeistuksen.
7. Tämän vuoden loppuun mennessä.