Haaga-Helia
ammattikorkeakoulu Oy

Tiedon laadun hälytysjärjestelmä Primukseen

Jani Kurki-Suonio

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2020

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

15.5.2020

Tekijä(t)
Jani Kurki-Suonio

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko

Tiedon laadun hälytysjärjestelmä Primukseen

Sivu- ja liitesivumäärä
39 + 6

Opinnäytetyön otsikko englanniksi
Information quality alert system for Primus school administration

Toisen asteen oppilaitos käyttää koulutuksen tietojen tallennukseen Primus-oppilashallintojärjestelmää. Yksi keskeisimmistä ongelmista on, että sinne syötetty tieto ei ole aina ajantasaista. Tiedon puutteet tekevästä raportoinnista virheellistä, aiheuttavat vääriä tulkintoja tai estävät muutoin oppilaitoksen toimintoja. Nämä puutteet heijastuvat oppilaitoksen toimintaan mm. laatunäkökulmasta ja voivat vaikuttaa oppilaitoksen rahoitukseen.

Ammatillisen toisen asteen koulutuksen uudistus v. 2018 toi tuloksellisuuden merkittäväksi osaksi oppilaitoksen rahoitusta. Tämä asettaa uusia vaatimuksia tiedon laadulle, kun mm. valmistuneiden opiskelijoiden määrä vaikuttaa aikaisempaa merkitsevämmällä osuudella oppilaitoksen tulorahoitukseen. Samana vuonna käyttöön otettiin KOSKI-palvelu, kansallinen opintosuoritusten ja opiskeluoikeuksien valtakunnallinen tietovaranto. Aiemmin siirtyneiden opiskelijoiden myöhemmin huomattu virheellinen tieto oppilaitoksen omassa Primus-oppilashallintojärjestelmässä on vaikeammin korjattavissa sen jälkeen, kun tieto on siirtynyt KOSKI-palveluun. Tämän vuoksi virheet ovat aikaisempaa kriittisempiä ja haitallisempia oppilaitoksen toiminnalle kuin ennen KOSKI-palvelua.

Opinnäytetyön tarkoituksena on luoda oppilaitokselle tiedon laadun hälytysjärjestelmä, joka auttaa oppilaitosta ylläpitämään oikea-aikaisesti ja täsmällisesti Primus- oppilashallintojärjestelmänsä tiedon laatua. Hälytysjärjestelmä huolehtii viestimällä asianosaisille Wilma-viestien avulla, mikäli havaitaan puutteita Primus-oppilashallintojärjestelmässä. Kun opiskelua koskevat tiedot ovat oppilaitoksen laatukäsikirjaan kuvattujen prosessien mukaisesti kirjattu oppilashallintojärjestelmään, on raportointi ja muu tiedon hyödyntäminen täsmällistä ja tarkkaa.

Asiasanat
Tiedon laatu, prosessien ohjaus, Primus-oppilashallintojärjestelmä

Sisällys

1 Johdanto

Tiedon laadun ongelmiin oppilaitoksen tärkeimmässä tietojärjestelmässä, oppilashallinto-ohjelmassa, pätee samankaltaisuus tietokoneohjelmien ohjelmistovirheiden, bugien, kanssa. Mitä myöhemmin virhe havaitaan, sitä suurempi sen haitallinen vaikutus on ja sitä kalliimmaksi sen korjaaminen tulee.

Miksi oppilashallinnon tiedon laatu on ongelma? Suurin osa oppilashallinnon tiedon laadun ongelmista vaikuttaa joko suoraan tai välillisesti oppilaitoksen rahoitukseen. Ammatillisen koulutuksen rahoitus tulee nykyään melko monimutkaisia ja byrokraattisia mekanismeja pitkin oppilaitokselle. Tämän lisäksi laadulliset ongelmat tiedossa vääristävät tiedoista muodostettavia raportteja ja voivat pahimmillaan johtaa vääriin strategisiin päätöksiin. Talous- ja strategisen puolen lisäksi osa virheistä voi aiheuttaa laadullisia ongelmia opetukseen, joista seuraa haitallisia vaikutuksia oppilaitoksen maineelle. Näilläkin voi olla taloudellisia vaikutuksia ainakin pitkällä tähtäimellä.

Tämän opinnäytetyön tarkoituksena on parantaa oppilaitoksen kykyä reagoida oppilashallintojärjestelmässä oleviin tiedon laadullisiin ongelmiin tai tietopuutteisiin. Opinnäytetyön toimeksiantaja on toisen asteen ammatillista koulutusta antava koulutuskonserni Careeria. Careeria on pääosin Itä-Uudellamaalla ja pääkaupunkiseudulla toimiva ammatillista koulutusta nuorille ja aikuisille tarjoava monialainen oppilaitos. Tutkintovalikoima käsittää yli 80 eri alan ammatillista tutkintoa. Konsernilla on vuotuisesti n. 11 000 opiskelijaa ja sen liikevaihto on n. 35 miljoonaa euroa. Työntekijöitä on yli 400. Careeria syntyi v. 2019 alussa, kun kaksi itä-uusimaalaista oppilaitosta, Edupoli ja PointCollege, yhdistivät toimintansa. Careerialla on tällä hetkellä 10 eri toimipistettä Itä-Uudellamaalla ja pääkaupunkiseudulla (Careeria 2020). Syntyneen organisaation suuri opiskelijamäärä ja opiskelijatyövuosimäärä asettavat haasteita opiskelijahallinnon tiedon laadulle, johon Careeria tarvitsee uusia ratkaisuja.

Careerian muodostaneista oppilaitoksista yhdellä, Edupolilla, on pitkät perinteet omasta sovelluskehityksestä. Edupoli oli yksi ensimmäisistä, ellei todennäköisesti ensimmäinen oppilaitos Itä-Uudellamaalla, jolla oli omia Internet-palvelimia jo 1990-luvun puolivälissä. Edupoli jopa tarjosi Internet- ja palvelinkapasiteettia ulkopuolisille asiakkaille

Internet palveluntarjoajan muodossa aina vuoden 2003 loppuun saakka. Tämän toiminnan yhtenä osana Edupolille syntyi omaa sovelluskehitystä ja verkko-oppimisympäristö eStudio, joka oli verrattavissa vastaaviin oppimisympäristöihin, esimerkiksi Moodleen. Edupolissa on kehitetty ja ylläpidetty useita sisäisiä sovelluksia liittyen Edupolin toimintaan. Niiden käyttöä on jatkettu edelleen Careeriassa. Oli siis luonnollista, että Careeria voisi käyttää omaan toimintaansa räätälöityä sovellusta oppilashallinnon tiedonlaadun parantamiseen.

1.1 Opinnäytetyön tavoitteet

Työn tavoitteena on kehittää Careerian toimeksiannon perusteella oppilaitoksen käyttöön järjestelmä, joka valvoo ja viestii oppilashallinto-ohjelman tietosisällisistä virheistä tai puutteista asianosaisille. Tähän hyödynnetään oppilashallinnon ohjelmassa etukäteen määriteltyjä sääntöjä. Tiedon laadun hälytysjärjestelmä Primus-oppilashallintojärjestelmää varten tähtää tiedon laadun parantamiseen virheisiin puuttumisen kautta ja niistä raportoimalla asianosaisille. Hälytysjärjestelmällä on tarkoitus kehittää oppilashallinnon tiedon laatua siten, että järjestelmään syötettäisiin tieto oikea-aikaisesti ja oikeassa muodossa, oikeiden henkilöiden toimesta. Hälytysjärjestelmän tarkoitus on valvoa Primuksen oppilastietokannan tietokenttien sisältöä *PrimusQuery*-kyselyiden avulla. Havaitut virheet ja puutteet kirjataan järjestelmän omaan tietokantaan ja tiedon ylläpitäjälle, kulloisellekin vastuuhenkilölle raportoidaan päivittäin.

Opinnäytetyön tekijä on toiminut sovelluskehitystehtävissä Careeriassa yli kymmenen vuoden ajan. Tätä kautta on syntynyt myös laaja käsitys oppilaitoksen päivittäisestä toiminnasta mm. näyttötutkintojen vastaanottajan toimen kautta.

1.2 Opinnäytetyön rajoitteet

Opinnäytetyön aikataulun puitteissa toteutettiin toimeksiantajalle versio, johon on toteutettu ohjelman keskeiset toiminnot. Opinnäytetyön yhteydessä toteutettu avoimen lähdekoodin versio ei sisällä kaikkia toimeksiantajan tarvitsemia räätälöintejä, kuten käyttäjätunnistusta hyödyntäen toimeksiantajan käyttäjätietokantaa, vaan ne toteutetaan erikseen, opinnäytetyön ulkopuolella. Käyttöliittymän kieli on myös tässä vaiheessa

oletus, eli englanti ja myös kielen osalta lokalisointi suomen kielelle jää pois opinnäytetyöstä.

2 Oppilashallinnon tiedonhallinta

Pelkkä tietojärjestelmä ei takaa laadukasta tietoa, tärkein yksittäinen seikka tiedon laadun takaamiseksi on ihmisten tekemä työ, tiedon syöttäminen järjestelmään (Systeemityö 2/2009).

2.1 Tiedon laadun ongelmien havainnointia ja keinoja

Mitä tiedon laadulla tavoitellaan? Yksinkertainen vastaus on, että tieto on hyödyllistä käyttäjälleen vasta silloin, kun se on riittävän laadukasta (Eppler 2009, 1). Vaikka tiedon laatu on käsitteenä itsessään vaikeasti määriteltävä, Eppler esittää kolme eri kategoriaa, joihin tiedon laadulliset ongelmat voidaan jakaa:

1. Tiedon sisältöongelmat
2. Tiedon ajankohtaisuus
3. Tiedon muotoon liittyvät ongelmat

Ensimmäinen, kategorioista ilmeisin on sisältöön liittyvät ongelmat. Ne ovat tilanteita, joissa tiedon tuottaja ei ole tuottanut täsmällistä tietoa, tai tarvittava tieto yksinkertaisesti puuttuu. Toisena kategoriana ovat tiedot, joiden oikeellisuus on riippuvainen ajankohdasta. Ajatellaan esim. valmistumispäivämäärää. Kun henkilö opiskelee, kentän tulee olla tyhjä, mutta heti valmistumisen jälkeen kenttä on virheellinen, kunnes oikea päivämäärä on siihen syötetty. Viimeinen, eli tiedon muotoon liittyvät ongelmat kuvaavat tilannetta, jossa tieto on syötetty muodossa, joka on sinänsä oikein ja soveltuu johonkin tiettyyn tilanteeseen, mutta sen tulkinta jossain toisessa tilanteessa antaa eri tuloksen. Tällainen tilanne voisi syntyä esimerkiksi tilanteissa, jos tietokenttiä käytetään eri tarkoituksiin eri tilanteissa. Toinen esimerkki ovat esimerkiksi lukuarvoja sisältävät kentät, joista puuttuu yksikkö, jolloin niiden tulkinnassa voi tulla virheitä (Eppler 2009, 23).

Tiedon laadullisten ongelmien havainnointi ei välttämättä ole yksinkertaista. Ongelmien tunnistamiseen on Thomas C. Redman (Fisher, Zwass, Madnik, Wang, Pierce 2005, 46)

nimeää erilaisia keinoja, joita seuraavassa tarkastellaan Primus-oppilashallintojärjestelmän näkökulmasta:

- Tiedon jäljitys koko tiedonsyöttö ketjun ajan. Tätä toimintoa kutsutaan yleisesti tietojärjestelmissä auditointilokiksi. Vaatii, että tietojärjestelmä pitää sisällään muutosloki-toiminnon, joka pitää kirjaa muutoksista, niiden tekijästä ja ajankohdasta. Idea on, että tarkastelemalla auditointilokia, voimme tehdä havaintoja, missä vaiheessa virheet ilmaantuvat tietojärjestelmään. Primuksessa on sisäänrakennettuna kattava auditointijärjestelmä, joka pitää kirjaa kuka, mitä ja milloin tietoja on muutettu. Tämän ratkaisulla voidaan puuttua yksittäisiin virheisiin, mutta varsinaisia trendejä tai tilastoja tämä ei mahdollista. Auditointi voi kuitenkin auttaa rajaamaan virheiden syntymekanismeja.
- Syötettyjen arvojen vertailu annettuihin raja-arvoihin. Tässä metodissa rakennetaan erityisiä liiketoimintasääntöjä, joiden avulla virheet voidaan estää jo syöttövaiheessa. Primuksessa on olemassa mekanismi, joka mahdollistaa tiettyyn pisteeseen asti sääntöjen rakentamisen. Tällaisia sääntöjä ovat, että jotta tiettyyn kenttään voi syöttää jotain, myös jossakin toisessa kentässä tulee olla arvo. Kuten muutkin vaihtoehdot, tämä on käyttökelpoinen ja tämän avulla on mahdollista päästä eroon joistakin virheistä. Teknisesti ratkaisu ei kuitenkaan kykene estämään kaiken tyyppisiä virheitä. Sääntöjä ei voida laatia liian tiukoiksi, koska ne estäisivät tietojen syötön tietyissä tilanteissa ja tekisivät ohjelman käytöstä käyttökelvottoman. Tyypillinen tilanne on, että tiedon ensimmäisessä syöttötilanteessa ei yksinkertaisesti ole olemassa kaikkia tarvittavia tietoja. Luonnollisesti näiden liiketoimintasääntöjen laatiminen ja niiden ylläpitäminen vaativat myös ylimääräistä työtä.
- Tietojen tarkastelu sen asiantuntijoiden toimesta. Tietojärjestelmään syötettyä tietoa täytyy tarkkailla sellaisten henkilöiden toimesta, jotka tuntevat tiedon parhaiten. Käytännössä tämä toteutuu useimmiten yleisestikin erilaisten tietojärjestelmien osalta, mutta on toki tarpeellista tunnistaa tämän ilmiön hyödyllisyys ja muodostaa siihen johdonmukainen prosessi. Kyseessä on kuitenkin henkilöstöresursseja vaativa toimintamalli, joten hyödyntämis- ja toteutusmahdollisuuksia suhteessa kustannuksiin joudutaan tarkastelemaan tarkemmin, kuin automatisoiduissa ratkaisuissa.

- Asiakaspalautteen hyödyntäminen. Asiakaspalautteen keruu on yleinen ilmiö, mutta sen käyttö virheiden havainnointiin vaatii oman säännönmukainen prosessin. Tämänkin toteuttamismahdollisuuksia rajaa sen työ- ja resurssivaltaisuus.
- Asiakaspalautteen kerääminen haastatteluin. Yleiskuvaa tietojärjestelmän tietojen laadusta on toki mahdollista kerätä myös kohdennetuin haastatteluin. Tässä on pelkkään asiakaspalautteeseen hyödyntämiseen verrattuna ainakin se etu, että kysymykset voidaan laatia siten, että niiden vastauksia pystytään jalostamaan helpommin virheiden havainnointiin, kuin asiakkaalta saadun palautteen tapauksissa.

2.2 Oppilaitoksen tiedonlaadun valvonnan historiaa

Oppilaitoksen tiedon laatuvirheet tulevat yleensä ilmi siinä vaiheessa, kun tietoaineistosta koostetaan raportteja. Kun raportin tuloksia verrataan muihin lähteisiin eroavaisuudet lopputuloksessa yleensä paljastavat, että aineisto ei voi olla kaikilta osin kunnossa.

Careerian edeltäjäoppilaitos Edupoli havaitsi v. 2013 uuden oppilashallintojärjestelmän StudentaPlussan käyttöönoton jälkeen tiedoissa suuria määriä virheitä. Virheitä pyrittiin korjaamaan koostetulla virhelistauksella v. 2013 lomakauden aikana. Seuraava virhelistaus tehtiin kesälomakaudelle 2014, jonka jälkeen aloitettiin säännölliset tarkistuslistat. Tämän johdosta oppilaitos aloitti myös kokeilun tiedonlaadun parantamiseksi. Ratkaisuna kokeiltiin bonusjärjestelmää syksystä 2014 vuoden 2015 loppuun. Kouluttajia palkittiin oikeista kirjauksista palkanlisillä. Tietoja tarkastettiin Excel-taulukoiden avulla oppilashallinnon pääkäyttäjän toimesta. Bonusjärjestelmä ei kuitenkaan merkittävästi lisännyt tietojen oikeellisuutta, koska vain noin puolet tiedoista oli ohjeiden mukaisesti kirjattu.

Tarkistuslistat jaettiin oppilaitoksen Intranetin kautta. Niihin virheet oli kirjattu keskitetysti ja niitä ylläpidettiin säännöllisesti. Tarkistuslistojen virhemääristä oli Intranetissä myös kokonaismääräinen kooste ja myöhemmin tiimikohtainen kooste. Erilaisia listoja oli toistakymmentä ja niillä tarkistettavia asioita n. 70, esimerkiksi tutkinnon osien arvi-

ointitiedot, tilastointiin liittyvät tiedot, sopimukset, rahoitustiedot, oppisopimustiedot ja oppilaitoksen sisäiset tarkenteet.

Kun v. 2018 siirryttiin käyttämään Primusta, siirrettiin Excel-taulukoissa olleet tietokantakyselyt Primuksen valmiisiin hakuihin. Näin päästiin Excel-taulukoista ja niiden manuaalisesta päivityksestä eroon. Nyt vastaavat virheraportit tuovat Primuksessa näkyville opiskelijat, joiden tiedoissa on virheitä ja luettelojärjestyksellä saadaan esille kunkin tiimin korjausta tarvitsevien opiskelijoiden tiedot. Tarkistuslistojen määrä ja sitä myötä tarkistettavien asioiden määrä ja aihe on lisääntynyt. Tämän on mahdollistanut tarkistuslistojen helppo päivitettävyys. Tänä päivänä Opetushallituksen KOSKI-palvelu ilmoittaa monista ennen manuaalisesti tarkistetuista virheistä. Tosin Opetushallitus on rakentanut monia uusia palveluita, joten tiedonlaadun tarve on itsessään moninkertaistunut. (Careerian Primus-pääkäyttäjä 29.4.2020)

2.3 Oppilashallinto-ohjelmat Careeriassa

Careeria-oppilaitos ja sen edeltäjät ovat käyttäneet oppilashallinto-ohjelmistoja 1980-luvulta saakka. Ensimmäinen oppilashallinto-ohjelma oli MS-DOS-pohjainen Ohra, jossa painopiste oli lähinnä opiskelijoiden yhteystiedoissa sekä tiedoissa, mitä tutkintoa tai kurssia he opiskelivat. Sittemmin vuosikymmenten vaihtuessa oppilashallinto-ohjelmistot ovat seuranneet toisiaan ja oppilashallinto-ohjelmien toiminnot ovat monipuolistuneet kattamaan hyvin laajasti kaikki oppilaitoksen toiminnot. 1990-luvulla Windows-pohjaiset OK2000 ja sitä seurannut Studenta sekä StudentaPlus perustuivat Microsoft SQL -tietokantaan. Tämä mahdollisti oppilaitokselle omien raporttien tuottamisen oppilashallinnossa olevan tiedon perusteella.

Miten ja mitä toimintoja kulloisestakin oppilashallinto-ohjelmistosta on otettu käyttöön, on riippunut monesta seikasta. Tärkeimpänä vaikuttimena on ollut osaaminen ja koulutus, uuden ohjelmiston käyttöönotto on aina iso ja raskas prosessi, joka on osin koettu ylimääräisenä työnä muun, varsinaisen työn rinnalla. Toisaalta ulkoiset paineet esimerkiksi viranomaistahoilta ovat vaikuttaneet siihen, mitä osia oppilashallinnon toiminnoista on kulloinkin otettu käyttöön. Toimintoja, jotka on koettu vähemmän tär-

keiksi, tai joiden hyödyllisyys ei ole ollut riittävän ilmeistä, ovat voineet jäädä hyödyntämättä.

Oppilashallinto-ohjelmistojen uusien toimintojen ja tietokenttien myötä työmäärä on kasvanut. Ne ovat tuoneet uusia tiedon syöttämiseen liittyviä velvoitteita varsinkin opettajille ja kouluttajille. Nämä on saatettu nähdä turhana byrokratiana, jota ei ole koettu hyödyntävän varsinaista opetusta ja olevan pois varsinaisesta opetustyöstä. Hyöty uusista toiminnoista ja tietokentistä näkyykin usein oikeastaan vasta raportoinnin kautta.

Tiedon syötön käytännöt ovat vaihdelleet suuresti oppilaitoksen eri osastojen välillä. Toisaalla jokin tehtävä on saattanut olla assistentin tehtävä ja toisaalla se on kuulunut koulutuksen vastuuopettajalle. Sinänsä oppilaitosmaailmassa tämä ei ole ollut poikkeuksellista, vaan pikemminkin sääntö kuin poikkeus. Marja Siniranta kuvaa hyvin samankaltaisesti tilannetta omassa opinnäytetyössään vuodelta 2009, jossa hän kuvaa oman oppilaitoksensa oppilashallinnon prosesseja ja niiden jakautumista eri käyttäjien ja roolien välillä (Siniranta M. 2009, 66).

Vuonna 2018 Edupoli otti käyttöön Primuksen, johon konvertointiin tietokanta StudentaPlussasta. Edupolin Primukseen yhdistettiin Amiston Primus sekä Itä-Uudenmaan oppisopimuskeskuksen SopimusPro. Nykyinen Primus-oppilashallintojärjestelmä on otettu käyttöön v. 2019, kun Careeria Oy syntyi kahden oppilaitoksen (Edupoli ja PointCollege) yhdistäessä toimintansa. Careerian Primus-oppilashallintojärjestelmä koostuu Careeriaan yhdistettyjen oppilaitosten aikaisemmista oppilashallinnon tietokannoista. Vaikka oppilaitoksilla oli tässä vaiheessa sama Primus-ohjelmisto käytössään, oli yhdistämisprosessi haastava. Oppilaitoksissa Primusta oli käytetty vuosien saatossa eri tavoin ja samoja kenttiä oli hyödynnetty eri lailla eri oppilaitoksissa. (Careerian Primus-pääkäyttäjä 29.4.2020)

2.4 KOSKI-palvelu ja tiedon laadun merkitys rahoitukseen

Tiedon laadun merkitys ammatillisen koulutuksen hallinnon näkökulmasta on kasvanut merkittävästi viime aikoina. Tämän kasvun tärkein yksittäinen tekijä on ollut Suomessa

vuonna 2018 käyttöönotettu KOSKI-palvelu. Kaikki ammatilliset oppilaitokset on velvoitettu siirtämään säännöllisesti tiedot opiskelijoiden opiskelun etenemisestä, eroamisista ja valmistumisista KOSKI-palveluun.

KOSKI-palvelu on asettanut tiedolle uuden vaatimustason. Jotta KOSKI-tiedonsiirto toimisi, tietosisältö tulee olla tietyssä muodossa. Oppilashallinnolta vaaditaan, että tietyt kentät tulee olla käytössä ja tiedot tulee olla KOSKI-määritysten mukaisessa muodossa. Kaikki pakolliset kentät tulee olla täytetty ja tietojen kirjaukset tulee olla aikataulullisesti kirjattu loogisesti oikein. Aikaisemmin tietoja on ollut mahdollista korjata jälkikäteen, eikä korjausajankohdalla ole ollut merkitystä. Myös opiskelijoiden etuudet ovat nykyään riippuvaisia KOSKI-palveluun kirjatuista tiedoista, koska viranomaiset, esimerkiksi KELA, hakevat tietonsa sieltä säännöllisesti. Opiskelija näkee myös itse tietonsa opintopolkupalvelusta. Korkeakouluhaku ja yhteishaku ottavat myös tiedot KOSKI-palvelusta.

Ammatillisen koulutuksen uudistuksen myötä rahoitus on muuttunut entistä riippuvaisemmaksi koulutuksen tuloksellisuudesta. Tämän vuoksi on entistäkin tärkeämpää, että oppilaitoksen prosessit ja toiminnot on yhtenäistetty ja standardoitu vastaamaan tarpeita. Käytännön päivittäisessä työssä tämä pitäisi näkyä erityisesti oppilashallinnon tiedon laadun huomioimisessa.

2.5 Tiedon laadun ongelmat ja niiden vaikutus oppilaitoksen toimintaan

Careerian pääkäyttäjät ovat selvittäneet runsaan joukon erilaisia tiedon laadun ongelmia, jotka aiheuttavat erilaisia hankaluuksia oppilaitoksen toiminnalle ja luoneet niihin perustuen PrimusQuery-kyselyt. Suuri osa kyselyistä on julkaistu oppilashallintoa päivittäin käyttäville Primus-käyttäjille valmiiden hakujen näkymän kautta. Näin he voivat oman viikottaisen toimintansa osana tarkistaa halutessaan omien tietueidensa virheellisyyksiä näiden kyselyiden tuottamilla tarkistuslistoilla. Suurin osa Primuksen käyttäjistä ei kuitenkaan käytä Primusta suoraan Primuksen kautta, vaan sen web-pohjaisen käyttöliittymän, Wilman, kautta. Suurimman osan tästä käyttäjäryhmästä muodostavat tunti- ja vastuuopettajat, joita Careeriassa on n. 300. Tälle ryhmälle on ominaista, että heidän Wilma-käyttönsä on satunnaisempaa, eikä rutiineja synny samassa määrin.

2.6 Kohti isompia oppilaitosyksiköitä

Toisen asteen oppilaitoksissa kehitys on viime aikoina vienyt kohti yhä isompia oppilaitoskokonaisuuksia. Itä-Uudellamaalla, josta Careeria tulee, oli 2010-luvun alussa neljä suomenkielistä toisen asteen oppilaitosta. Nämä olivat toisen asteen aikuiskoulutusta tarjoava Edupoli, nuorille, alle 18-vuotiaille tarkoitettu Amisto, sekä Porvoon terveydenalan oppilaitos ja Porvoon kauppaoppilaitos.

Toisen asteen oppilaitosten toiminta on järjestämisluvanvaraista toimintaa, jota ohjaa Suomen lainsäädäntö (Finlex 2017). Opetusministeriö ylläpitää tätä lainsäädäntöä ja käytännön toimia toteuttaa ja valvoo suomessa Opetushallitus. Toisen asteen opetus on ollut säästötoimenpiteiden kohteena koko 2010-luvun (Opetus- ja kulttuuriministeriö 5.3.2017). Suomen hallitusohjelmissa yhtenä säästökeinona ja valtiovallan tahtotilana on ollut oppilaitosten yhdistäminen isommiksi kokonaisuuksiksi. Tämän politiikan pyrkimyksenä on ollut luoda entistä vahvempia koulutuksen järjestäjiä, joilla on riittävät resurssit suoriutua niille asetetuista vastuista ja tavoitteista (Opetus- ja kulttuuriministeriö 2014, 6). Tällä ns. ”suuruuden matematiikalla” on tavoiteltu synergiaetuja esimerkiksi yhdistämällä opintolinjoja, opettajaresursseja, toimitiloja ja varsinkin tietojärjestelmiä. Asiaa on käsitelty erilaisissa poliittisissa elimissä Itä-Uudellamaalla ja asiat ovat lopulta edenneet siten, että kaikki suomenkieliset toisen asteen oppilaitokset yhdistyivät lopulta yhdeksi v. 2019 alusta EdupoliPointCollege Oy:ksi, toukokuussa 2019 tapahtuneen nimenvaihdoksen jälkeen Careeria Oy:ksi.

Oppilaitosyhdistymiset alkoivat v. 2012, kun Porvoon kauppaoppilaitos ja Porvoon terveydenhoito-oppilaitos yhdistyivät Point College Oy:ksi. Yhdistymiset jatkuivat, kun nuorisoaste Amisto ja aikuiskoulutus Edupoli yhdistyivät Edupoli nimen alle v. 2018. Yhdistymiset mahdollisti toisen asteen koulutuksen reformi, kun suomen lainsäädännössä nuoriso- ja aikuiskoulutuksen lainsäädäntö yhdistettiin eikä erillisiä oppilaitoksia nuoriso- ja aikuiskoulutukseen enää tarvittu (Ammatillisen koulutuksen Reformi). Nämä kaksi oppilaitosta olivat jo ennestään osa samaa kuntayhtymää ja yhdistäneet mm. tietotekniikan, kiinteistö-, talous- ja siivouspalvelut jo paljon aiemmin.

Opetus- ja kulttuuriministeriö toteutti v. 2018 alusta Suomessa toisen asteen koulutuksen reformin. Tämä oli myös yksi oppilaitosten yhdistymisen mahdollistava tekijä. Ennen v. 2018 toisen asteen nuoriso- ja aikuiskoulutuksella oli oma lainsäädäntönsä Suomessa, mikä korvattiin uudella yhteisellä lainsäädännöllä v. 2018 alusta, reformin tultua voimaan. Reformissa uudistettiin toisen asteen koulutus mm. virtaviivaistamalla oppilaitosten prosesseja (Opetus- ja kulttuuriministeriö 2018).

2.7 Oppilaitosten prosessit

Aikaisemmin, kun oppilaitokset olivat kooltaan suhteellisen pieniä, monissa alueen oppilaitoksissa oli kirjavia, toisistaan poikkeavia käytäntöjä oppilaitoksen toimintojen toteuttamiseksi. Osa toiminnoista koettiin liian hankaliksi toteuttaa oppilashallinto-ohjelmistojen kautta tai oppilashallinnon-ohjelmasta puuttuivat toiminnot sellaisten toteuttamiseksi. Tyypillisinä ratkaisuina olivat erilaiset Excel-taulukot, joita käytettiin opetuksen hallintoon liittyvien tietojen ylläpitoon. Myös paperiset dokumentit tietyn kouluttajan kansioissa olivat osa tiedonhallintaa.

Kun oppilaitokset yhdistyivät, tuli keskeiseksi asiaksi kuvata ja dokumentoida erilaiset prosessit, sekä selvittää, millä tavalla prosesseihin liittyviä tietoja ylläpidetään ja kuka niitä ylläpitää eri oppilaitoksissa. Marja Siniranta korostaakin roolien merkitystä oppilaitoksen tiedonhallinnan kannalta. Käyttäjien on osattava viedä asiat oikeisiin paikkoihin oppilashallintajärjestelmässä ja ymmärrettävä eri tietokenttien merkitys samalla tavalla (Siniranta, M. 2009, 47).

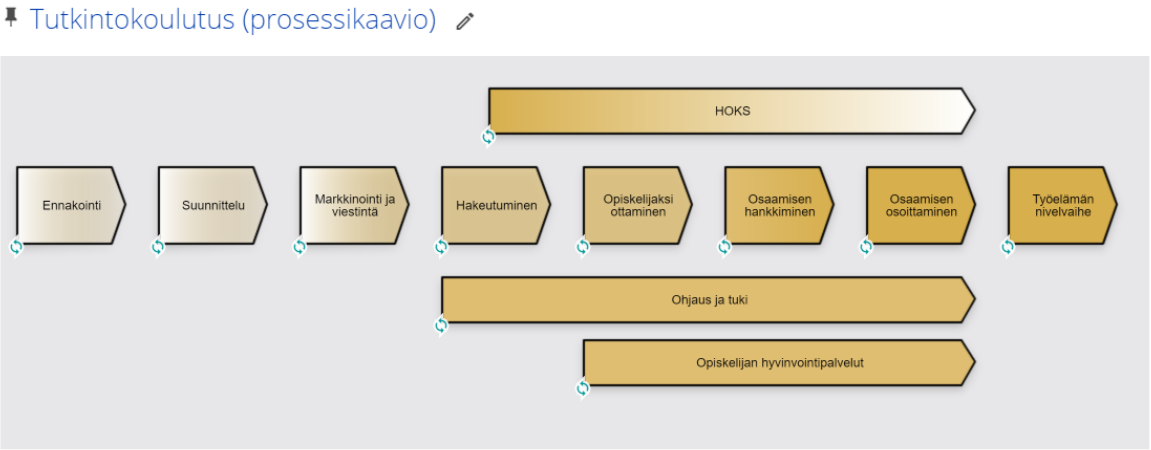
2.8 Parasta Digitukea -hanke

Toisen asteen oppilaitoksilla on syntynyt laaja tahtotila toimintojen ja niihin liittyvien prosessien yhtenäistämiseksi. Standardoimalla ja luomalla yhteisiä käytäntöjä oppilaitosten valmiudet keskinäiseen yhteistyöhön paranevat. Tavoitteena on myös jakaa ja synnyttää ns. hyviä käytäntöjä oppilaitosten kesken. Careeria sekä joukko muita ammatillisia oppilaitoksia osallistuivat v. 2018 – 2019 toteutettuun, Opetus- ja kulttuuriministeriön rahoittamaan Parasta Digitukea -hankkeeseen. Hanke pyrki toimimaan itsearviointivälineenä, jonka avulla hankkeeseen osallistuvat oppilaitokset voisivat mitata kyvykkyyksiään yhteentoimivuuden kehittämiseksi. Hankkeen lopullisena tavoitteena oli ko-

konaisvaltaisen kypsyystasomallin luominen koulutuksen järjestäjien kehittämisen välineeksi. Perinteisesti on ajateltu, että yhteensopivuudella tarkoitetaan nimenomaan tietojärjestelmien välistä yhteensopivuutta. Tämä hanke kohdistui kuitenkin kaikkien neljän toiminnan osa-alueen kehittämiseen: johtaminen ja ihmiset, toiminta ja laatu, hallinta sekä tekniikka (Parasta Digitukea kehityshanke 18.12.2019).

Keskeisenä toimenpiteenä hankkeessa oli osallistuneiden oppilaitosten toiminnan tietomallinnus, jonka käytännön tuloksena syntyi oppilaitokselle hankitun ARC – kokonaisarkkitehtuurin mallintamistyökalun (Arter 2020) avulla päivitettävä dokumentaatio oppilaitoksen toiminnoista jaoteltuna prosesseihin, niihin kytkeytyviin resursseihin ja sidosryhmiin.

Tiedon laadun kannalta Parasta Digitukea -hankkeen ansiosta oppilaitoksen prosessit dokumentoidaan ja eri henkilöiden roolit oppilashallinto-ohjelman käytön näkökulmasta selkeytyvät. Tämän lisäksi määritellään, missä vaiheessa mikäkin tieto tulee esimerkiksi kirjata oppilashallinto-ohjelmaan. Oppilaitoksen prosesseja on toki kuvattu aikaisemminkin, mutta ARC-ohjelman avulla dokumentaation hyödynnettävyys nousee uudelle tasolle. Ohjelma kuvaa kokonaisvaltaisesti oppilaitoksen toimintaa ja siinä kytkeytyvät pelkkien prosessien lisäksi tietojärjestelmät, niiden tieto, käyttäjät rooleineen ja näistä syntyvät kyvykkyydet yhteen. Dokumentaatio hyödyntää oppilashallinnon tiedonlaadun kehittämistä. Esimerkiksi tutkintokoulutuksen prosessikaavion (kuva 1) avulla oppilaitoksen henkilökunta pystyy pureutumaan eri näkökulmista prosesseihin ja niihin kytkeytyviin tietojärjestelmiin eri rooleissa. Dokumentaation avulla on mahdollista saada parempi kokonaiskuva, miten eri toimenpiteet missäkin vaiheessa vaikuttavat ja miksi tiettyjen tietojen kirjaaminen eri vaiheissa oikein on tärkeää.



Kuva 1. Esimerkkinä Careerian tutkintokoulutuksen tietomallinnus. (Careerian prosessikaaviot 2019)

2.9 Primus-oppilashallintojärjestelmän tietokantaa hyödyntävät sovellukset

Vaikka Primus tänä päivänä kattaa toisen asteen oppilaitoksen toiminnot varsin kattavasti, erityisesti tiedon raportointiin oppilaitoksissa on herännyt tarpeita, joita Primus ei suoraan kata. Puhutaan esimerkiksi vastuukouluttajan työpöydästä, jossa olisi yhdellä silmäyksellä nähtävissä tilannenäkymiä omien koulutusten ja niihin liittyvien opiskelijoiden opintojen tilasta.

Nykyään tiedolla johtaminen on keskeinen välinen oppilaitoksen johtamisstrategioissa (Opetushallitus 2020). Tarvetta varten tarvitaan usein tietovarastoratkaisuja, joissa Primuksesta tuotettu tieto olisi yhdistetty esimerkiksi taloustietoihin, palautekyselyihin ja muihin oppilaitoksen toimintaa koskeviin tietoihin. Näitä tarpeita varten Primus-oppilashallintojärjestelmän ympärille on syntynyt kolmansien osapuolien toteuttamaa ohjelmistoteollisuutta, joka hyödyntää oppilashallintosovelluksella tuotettua tietoa eri tavoin.

2.9.1 School Day Helsinki Oy

School Day DashBoard on School Day Helsinki Oy -nimisen ohjelmistofirman toteuttama Primus-oppilashallintojärjestelmän dataa hyödyntävä data-analytiikkasovellus. School Day Dashboard -sovelluksen pääasiallinen käyttötarkoitus on oppilaitoksen rahoituksen perustana olevien lukujen tarkastelu. Tärkeimpänä mittarina ovat opiskeli-

javuodet, suoritetut tutkinnot ja suoritettujen tutkintojen osien määrät. Näiden lisäksi School Day Dashboardista on mahdollista nähdä myös vuositavoite opiskelijavuosien osalta, joka mahdollistaa sen, että tavoitteen toteutumista voidaan seurata reaaliajassa kustannuspaikoittain (Visma 2020).

Oppilaitoksella on luonnollisesti suuri tarve seurata toimintansa tunnuslukuja. Aikaisemmin seurantaa on tehty hyödyntäen Primuksessa olevaa PQ-laskentafunktiota, josta saatuja raportteja hyödynnettiin Excel-taulukkolaskentaohjelmassa. Nämä olivat kuitenkin työläitä ja virhealttiita menetelmiä, joista on siis mahdollista päästä eroon School Dayn Dashboard -sovelluksen avulla. School Day Dashboard mahdollista erillisistä raporteista luopumisen, kun ajantasainen tieto näkyy reaaliajassa (Visma 2020).

2.9.2 TCD Consulting and Research Oy

Konsultointi- ja ohjelmistoyritys TCD Consulting and Research Oy on tehnyt 2014 vuodesta eteenpäin tietovarastointiprojekteja koulutuksen järjestäjille. Puhelin ja sähköpostikeskustelussa yrityksen edustaja, Arto Pärnänen kuvailee heidän toteuttamiaan ratkaisuja seuraavasti; Ratkaisussa ammatillisessa koulutuksessa käytössä olevasta Primuksesta tiedot on siirretty PrimusQueryllä SQL-tietokantaan. Toteutuksessa siirto on tapahtunut eräajona, yleensä kerran päivässä ajastettuna. Tarkoitus siis on, että tietokannan käyttäjä voi käyttää SQL-kyselyitä ja työkaluja halutessaan jatkojalostaa Primuksesta saatavaa dataa. Tämän päälle TCD Consulting and Research Oy on rakentanut toiminnallisuuden Primus-ilmoituksia varten. Ilmoitukset on toteutettu PrimusQuery-hakuina, ja hakukyselyt määritellään hakutiedostoihin. Haut ja tulosten käsittely on tehty *Microsoftin Integration Service*:n avulla ja nämä on ajastettu *SQL-Agentilla* suoritettavaksi. Ilmoitukset voidaan lähettää vastaanottajalle joko sähköpostilla, Wilma-viestinä tai SMS-viestinä. Ilmoituksia on käytetty niin opintohallinnon tueksi (esimerkiksi hälytykset siirrettävistä hakijoista), tai viestejä voidaan lähettää myös opiskelijalle - esimerkiksi tekstiviesti-ilmoitus, jos opiskelija on valittu koulutukseen.

Käydyn puhelinkeskustelun ja sähköpostiviestinnän perusteella TCD Consulting and Research Oy:n ratkaisu muistuttaa paljolti samaa kuin Careerialle toteutettu tiedon laadun hälytysjärjestelmä. Tästä voi mielestäni päätellä, että tiedon laadun ongelmakenttä

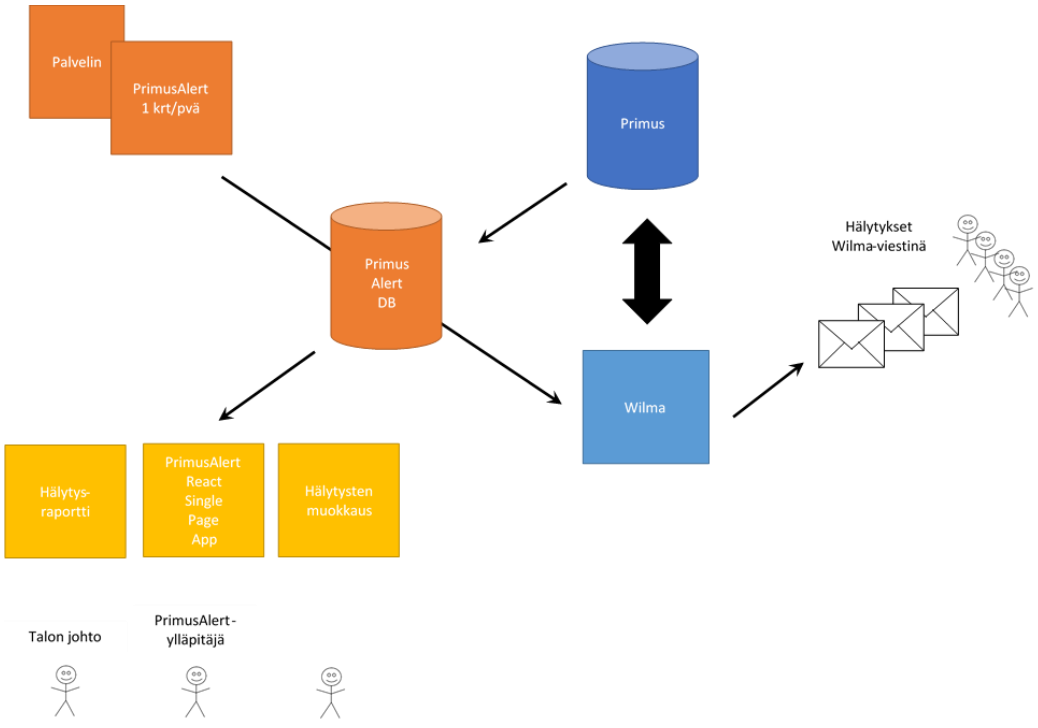
ja ratkaisukeinot ovat hyvin samankaltaisia ammatillisessa koulutuksessa. (Arto Pärnänen 2020)

3 Vaatimusten mukainen Primus-oppilashallintojärjestelmän tiedon laadun hälytysjärjestelmä

Tämän opinnäytetyön tavoitteena toteutettiin toimeksiantaja Careerian vaatimusten mukainen ohjelmisto, jonka avulla Careeria pystyy parantamaan oppilashallintonsa tiedonlaatua sekä säästämään työresursseja turhan manuaalityön vähetessä (kuva 2).

Itse sovelluksen lisäksi tarkoitus on, että Careeria saa projektin myötä itselleen kokemusta uusista teknologioista ja menetelmistä. Näitä ovat Microsoftin uusin kehitysympäristö .NET Core, React-käyttöliittymän kehityskirjasto ja Azure-pilvipalveluiden hyödyntäminen omien palvelimien sijaan sekä avoimeen lähdekoodiin perustuvan sovelluskehityksen toteuttaminen.

Toisaalta ohjelmisto toteutettiin siten, että se on mahdollisimman yleiskäyttöinen ja hyödynnettävissä vähintään Primus-oppilashallintoa käyttävissä ammatillisen koulutuksen oppilaitoksissa. Avoimen lähdekoodin ohjelmistona se tulee olemaan kaikkien halukkaiden saatavilla ja hyödynnettävissä joko sellaisenaan tai osana jotakin toista hanketta.



Kuva 2. Primus-hälytysjärjestelmän toteutusperiaate.

3.1 Primus-oppilashallintojärjestelmä

Primus-oppilashallintojärjestelmän on alun perin peruskouluille toteuttanut Starsoft Oy. Yrityskaupan myötä Primus on siirtynyt osaksi norjalaislähtöisen ohjelmistotalo Visman tuotteeksi. Primus on yksi pitkäikäisimmistä oppilashallintojärjestelmistä, jonka kehityksen juuret ovat jo 1980-luvulta (Vaasalaisia.Info 16.8.2019).

Primus on tietokantasovellus, joka poikkeaa kuitenkin olennaisesti tyypillisistä tietokantasovelluksista. Primuksessa on oma, sovelluskohtainen tietokantaratkaisu. Suurin osa nykyisistä, käytössä olevista tietokantasovelluksista (DB-Engines, huhtikuu 2020) pohjautuu *SQL-tietokantakyselykielellä* toimiviin yleiskäyttöisiin *tietokantamoottoreihin*. Tällaisia tietokantoja ovat esim. Oracle, MySQL, Microsoft SQL ja PostgresSQL. Kuten näiden nimestä voi päätellä, niillä on yhteys *SQL-kieleen*. Siinä missä SQL-kieli on geneerinen, relaatiotietokannan luontiin, kyselyihin, muokkaukseen ja ylläpitoon tarkoitettu tietokantakyselykieli, PrimusQuery, Primuksen tietokannan kyselykieli toimii vain Primus-oppilashallinnon tiedon kanssa. Koska PrimusQueryä voidaan käyttää vain Primus-sovelluksen kanssa, voidaan perustellusti sanoa, että PrimusQuery on toteutustavaltaan *kovakoodattu* tai toisaalta, että se on räätälöity Primus-sovellusta varten.
Geneerisestä SQL-kyselykielestä poiketen PrimusQuery-kyselyissä otetaan suoraan kantaa itse Primus-oppilashallinnon sisältöön eikä PrimusQueryä voisi käyttää muun tyyppisiin tietokantoihin.

Ohessa esimerkkejä Primus-kyselystä, jolla haetaan vuoden 2020 Careerian opiskelijat.

Primus-käyttöliittymässä muodostetaan kyselytyökalulla kysely
((Arv. valmistumispäivä>=1.1.2020 JA Valmistumispäivä="" JA Eropäivä="") TAI (Valmistumispäivä>=1.1.2020 TAI Eropäivä>=1.1.2020)) JA Opiskelijalaji->Selite<>"*virtuaali*" JA Opiskelijalaji->Selite<>"Ei aloittanut"
…joka kääntyy PrimusQuery-kyselyksi
((K202>=%D1.1.2020% AND K203="" AND K310="") OR (K203>=%D1.1.2020% OR K310>=%D1.1.2020%)) AND K171^K2<>"*virtuaali*" AND K171^K2<>"Ei aloittanut"

Kenttiin osoitetaan tietyillä kiinteillä indeksinumeroilla, eli kyselykielessä on sisäänrakennettu oppilashallintojärjestelmän tietosisällön rakenne. Seuraavaksi voidaan verrata tätä kyselyä SQL-kieleen pohjautuvaan relaatiotietokantaan.

Sama mukailtuna SQL-lauseiksi kuvitteellisesta oppilashallintotietokannasta
USE DATABASE OpiskelijaHallinto; SELECT * FROM Opiskelijat WHERE ((ArvioituValmistumisPvm >= 1.1.2020 AND ValmistumisPvm IS NULL) OR (ValmistumisPvm >= 1.1.2020 OR EroPvm >= 1.1.2020)) AND OpiskelijaLajit.Name != ”*virtuaali*” AND OpiskelijaLajit.Name != ”Ei aloittanut” AND INNER JOIN OpiskelijaLajit ON Opiskelijat.OpiskelijaLajiID =OpiskelijaLajit.OpiskelijaLajiID

Tästä huomataan, että PrimusQuery muistuttaa loogiselta rakenteeltaan SQL-kyselyä ja siinä on samat boolean-operaattorit, eli AND ja OR, joilla rajataan hakutuloksia. Kenttänimien lisäksi SQL-lauseen tietokanta-relaatiot luodaan JOIN komennoilla. Tämän lisäksi Primuksen tietokanta sisältää ns. monirivisiä kenttiä, joille ei ole vastaavaa suoraa vastinparia SQL-relaatiotietokannoissa.

3.2 Wilma-käyttöliittymä

Wilma on vaihtoehtoinen käyttöliittymä Primukseen. Wilma tarjoaa web-käyttöliittymän, jota voidaan käyttää web-selaimella tai mobiililaitteella. Wilmaa hyödyntäen on myös tehty puhelimella tai tabletilla toimiva mobiili-applikaatio. Wilman tärkeimmät käyttäjäryhmät ovat opettajat, huoltajat, opiskelijat ja koulutuksen sidosryhmät. Wilman etuja on sen helppokäyttöisyys verrattuna Primuksen omaan käyttöliittymään. Wilman käyttöliittymä voidaan räätälöidä siten, että kullakin käyttäjäryhmällä on vain ne näkymät, joita he tarvitsevat.

4 Sovelluksen kehityskulku

Kehitystyö aloitettiin aloituspalaverilla 29.1.2020 toimeksiantajan edustajien kanssa. Aloituspalaverissa käytiin läpi ja sovittiin minimiversion ominaisuudet. Kävin läpi ja kerroin lyhyesti ehdottamani teknologiavalinnat perusteluineen sovelluksen toteuttamiseksi.

Toimeksiantajan tahtotila oli, että ohjelmisto olisi mahdollisimman pian käytettävissä. Riittäisi, että ohjelman perustoiminnot ovat olemassa ja toimeksiantaja antoi tämän suhteen melko vapaat kädet toteutukseen.

4.1 Hälytyksiä varten valitut säännöt

Palaverissa (liite 2) valittiin yhdeksän tapausta, joista kehitettäisiin tiedonlaatua koskevat hälytykset tätä projektia varten. Näiden tapausten perusteella lähetettäisiin Wilma-hälytysviestit muistutuksena päivittäin. Seuraavassa taulukossa kuvailtuna tapaukset, sekä niiden pohjalla olevien tietovirheiden vaikutus oppilaitoksen toimintaan:

1. Huoltajan tiedot puuttuvat alle 18-vuotiaalta oppilaalta
   Vaikutus: Osa Careerian opiskelijoista on alaikäisiä, joten heihin on saatava yhteys mm. erilaisten toimenpiteiden lupien pyytämistä varten.
   Kohde: Vastuuopettaja

2. Toisen rivin huoltajan tiedot
   Vaikutus: Kuten edellä, eli myös toisen huoltajan tiedot saattavat puuttua.
   Kohde: Vastuuopettaja

3. HOKS-tapahtumatyyppi "Ensikertainen hyväksyntä" puuttuu
   Vaikutus: Laki ammatillisesta koulutuksesta edellyttää, että jokaiselle oppilaalle tehdään HOKS (Henkilökohtainen osaamisen kehittämisen suunnitelma). HOKS on rahoituksen peruste ja Opetushallituksella on valtakunnallinen eHOKS-järjestelmä, johon HOKSin tiedot tulee siirtyä. Ensikertainen hyväksyntä on tehtävä, jotta HOKS-tiedot siirtyvät Pri-

muksesta eHOKSiin. Opetushallitus tarkkailee, että opiskelijoiden HOKSit on luotu ja se voi jopa periä takaisin maksettuja rahoitusosuuksia, jos HOKSeja ei tehty. Kun tieto siirtyy eHOKSiin, se antaa signaalin Opetushallituksen ARVO-järjestelmälle, joka lähettää palautekyselyn opiskelijalle. Opiskelijalle on 30 päivää aikaa vastata ARVO-kyselyyn. Myös tämä ARVO-kysely on osa oppilaitoksen rahoituksenperustetta ja siksi tärkeä.

Kohde: Vastuuopettaja

4. Opiskelijan väliaikainen keskeytys on päättymässä kahden viikon kuluessa

   Vaikutus: Muistutus vastuukouluttajalle opiskelijasta, joka palaa jatkamaan opintoja. Vastuukouluttajan on valmisteltava koulutuksen jatkamista.

   Kohde: Vastuuopettaja

5. Opiskelijan arvioitu päättymispäivä on tulossa neljän viikon sisällä

   Vaikutus: Muistutus, että vastuuopettaja tarkistaa ajoissa, onko opiskelijan opinnot edenneet suunnitellusti vai tarvitseeko opiskelijan opiskeluaikaa jatkaa ja HOKS päivittää. Tällä tavoitellaan sitä, että mm. todistusta tehtäessä se tulostuu oikean muotoisena, kaikki suoritukset kirjattuna.

   Kohde: Vastuuopettaja

6. Opiskelija valmistumassa, tarkista YTO:t (Yhteiset tutkinnon osat)

   Vaikutus: On tarpeen varmistaa hyvissä ajoin ennen opintojen päättymistä, että opiskelija on suorittanut kaikki yhteiset tutkinnon osat, muutoin valmistuminen ei ole mahdollista. Kaksoistutkintojen opiskelija suorittaa sekä lukio- että ammatillista koulutusta samanaikaisesti ja tällöin opinnot eivät näy suoraan Careerian Primuksesta. Ne, mitä lukiossa on suoritettu, tulee kirjattua oikein oikeisiin kohtiin ammatillisen koulutuksen todistukselle. Yleissivistävät aineet ovat olleet pakollisia toisen asteen perustutkinnoissa v. 2018 ammatillisen koulutuksen reformin jälkeen.

   Kohde: Yhteisten tutkinnon osien hoksaajat

7. Samalle ajalle tehty useita koulutussopimuksia
   Vaikutus: Virhe tuottaa kaksoislaskutusta tai sen, että jokin sopimus jää laskuttamatta kokonaan – rahoitus jää saamatta.
   Kohde: Vastuuopettaja

8. Opiskelijavuosi-taulukon (OV-taulukko) oikeellisuusvirheitä
   Vaikutus: Opiskelijavuositaulukossa kerrotaan päivämäärämuodossa opiskelijaoikeuden tila (läsnä, loma, väliaikaisesti keskeytynyt, katsotaan eronneeksi, valmistunut), rahoitus, osa-aikaisuusprosentti ja koulutuslaji. Jos jossain näistä tapahtuu muutos, korjataan taulukon rivitietoja. Taulukko on kronologisesti etenevä.
   Kohde: Opintosihteeri/assistentti

9. Opiskeluaika ohitse, muttei eronnut/valmistunut
   Vaikutus: Opiskelijan tila KOSKI-palvelussa on edelleen läsnä. Opiskeluoikeus tulee päättyä joko valmistumiseen tai eronneeksi katsomiseen.
   Kohde: Vastuukouluttaja

4.2 Pääkäyttäjän haastattelu 29.4.2020, mistä ongelmien arvellaan johtuvan?

Osana opinnäytetyötä yhtä pääkäyttäjää haastateltiin hänen näkemyksiään seikoista, jotka aiheuttavat edellä kuvattuja virheitä.

Mitkä ovat kokemuksenne perusteella olleet keskeisimpiä syitä virheiden syntymiseen?

> Oppilaitoksilla on ollut erilaisia tapoja kirjata tietoja, tiedot ovat voineet olla eri järjestelmissä tai tiettyjä tietoja ei ole ollut tarpeen kirjata. Oppilaitoksilla on ollut myös eri tyyppisiä koulutuksia, joten tarkkoja, tyypitteleviä kirjauksia ei ole tarvinnut tehdä. KOSKI-palvelu on vielä aika uusi, vuodesta 2018 käytössä, joten ei vielä täysin ymmärretä, kuinka kriittinen se on virheiden osalta. KOSKI-palvelu myös tarkentaa validiointiaan jatkuvasti. Kaikissa Caree-

riaan liittyneissä oppilaitoksissa ei ole ollut olemassa prosessia virheiden tarkistukseen tai virheiden tarkistusta ei ole tehty johdonmukaisesti.

Mitä virheiden havaitsemiseksi ja korjaamiseksi on tehty?

Osa ongelmista on saatu kuriin teknisin ratkaisuin esimerkiksi pakollisilla kentillä. Tietokenttiin voidaan luoda myös erityisiä tarkastussääntöjä, joilla on mahdollisuus asettaa tallentamisen esto, jos määrättyjä kenttiä ei ole täytetty. Tarkastussäännöt toimivat osaan virheistä ja laadullisista ongelmista, muttei kuitenkaan kaikkiin. Pääkäyttäjien luomat tarkastuslistat ovat myös Wilmassa näkyvissä opettajille ja kouluttajille. Tähän asti on muistuteltu manuaalisesti Intranetissä sekä Careerian viikko- ja kuukausi-infoissa sekä vierailtu klusterien palavereissa.

Jo ennen palaveria oli muodostunut käsitys ohjelman rakenteesta. Ohjelmisto tulisi koostumaan kahdesta eri osasta. Ensimmäiseksi valmistuisi palvelinosa eli ohjelmakoodi, joka suoritetaan määräajoin tausta-ajona palvelimella ja joka tuottaa itse hälytykset. Kun palvelinosa on valmis, voidaan todeta, että tuotteen vähimmäisvalmiusaste (minumum viable product) on toteutunut. Toinen, käyttöliittymäosa, tarvitaan, jotta ohjelman käyttäjät voivat ylläpitää hälytyksiä sekä seurata ja tuottaa raportteja hälytyksistä.

4.3 Käytetyt teknologiat

C#-ohjelmointikieli

C# on Microsoftin kehittämä, olioparadigmaa noudattava, staattisesti tyypitetty ohjelmointikieli. Se on saanut vaikutteita C, C++ ja Java-ohjelmointikielissä ja muistuttaa varsinkin syntaksiltaan näitä ohjelmointikieliä (Microsoft .NET / C# Guide 26.2.2020). Erityisesti C#:n muistuttaa keskeisiltä ominaisuuksiltaan Java-ohjelmointikieltä. Toisin, kuin C tai C++ kielistä - C#:sta löytyvät Javan tapaan mm. automaattinen roskien keruu sekä muistin ylivuodot estävä, Javasta tuttu virtuaalikoneen käyttö ohjelman suoritusympäristönä.

4.3.1 .NET Core

Tiedon laadun hälytysjärjestelmä Primukseen -ohjelman teknologiavalinnaksi valittiin Microsoftin uusin suoritusympäristö .NET Core. Tärkeimpiä valintakriteereitä oli toteutuksen pitkäikäisyys uuden suoritusympäristön ansiosta ja sen mahdollistava hyvä suorituskyky. Tämän lisäksi vahvana valintakriteerinä oli Careerian aikaisempi, pitkäaikainen kokemus Microsoft kehitysympäristöistä, Microsoft .NET:stä ja Classic ASP:sta.

.NET Core on Microsoftin julkaisema, avoimen lähdekoodin suorituspalvelu, joka on saatavilla useille laiteympäristöille. NET Core perustuu aikaisempaan, Microsoftin .NET suorituspalveluun. Se on kuitenkin kehitetty kokonaan alusta alkaen uudestaan. Tällä on pyritty pääsemään eroon monista, vanhoista ohjelmakoodiin liittyneistä ongelmakohdista, kuten *anti-patterneista* ja *code-smell* ongelmista – eli ohjelmakoodiin liittyvistä sinänsä toimivista, mutta arkkitehtuurisesti huonoista toteutuksista (Erik Englund, 21.4.2019). Microsoft .NET Core onkin osoittautunut huomattavasti suorituskykyisemmäksi, kuin aikaisemmat .NET versiot (Scientific Programmer ,18.8.2019). Tällä hetkellä .NET Core on saatavilla Windowsin lisäksi Linuxille ja Macintoshin Os X:lle.

Microsoftin .NET on Microsoftin kehitysympäristö, ohjelmointiympäristö ja ekosysteemi, josta Microsoft julkaisi ensimmäisen kaupallisen, version v. 2000. Kaupallisena ja suljettuna tuotteena Microsoftin .NET ympäristö oli suojattu mm. ohjelmistopatenteilla ja mallisuojilla, jotka tekivät siitä avointa lähdekoodia ja avointa kehitystä suosiville yrityksille ja organisaatioille vähemmän houkuttelevan vaihtoehdon. Esimerkiksi yliopistoissa ja korkeakouluissa vastaavin ominaisuuksin, mutta avoimeen lähdekoodiin perustunut Java sai vahvan jalansijan, .NET:in jäädessä marginaaliseksi varsinkin opetuksessa. Microsoftin .NET yleistyi kuitenkin nopeasti yritysmaailmassa, varsinkin siksi, että suuri osa Microsoftin Windowsiin kehitetyistä sovelluksista kehitettiin jatkossa sillä.

Uudesta .NET Coresta puhuttaessa on huomioitava, että .NET:in rinnalla on ollut todellisuudessa lähes koko .NET:in olemassaolon ajan tarjolla myös avoimeen lähdekoodiin perustuva vaihtoehto. Sen nimi on Mono. Scientific Programmer verkkojulkaisun (Scientific Programmer 18.8.2019) artikkelissa kuvataan eroavaisuuksia Monon ja .NET Coren välillä seuraavasti; Yksi keskeinen rajoite Microsoft .NET ympäristössä on sen

saatavuus vain Microsoftin omille käyttöjärjestelmille, eli Microsoftin Windows-versiolle. Microsoft .NET:ille on kuitenkin ollut olemassa avoimeen lähdekoodiin perustuva, toisaalla kehitetty vaihtoehto nimeltä Mono. Kuten nyt saatavilla oleva Microsoftin .NET Core, Mono tuki muitakin ympäristöjä kuin Windowsia, eli Mac OSX:ää, sekä Linuxia. Se ei tosin ominaisuuksilta ollut kaikilta osin yhdenveroinen .NET:in kanssa, eikä täysin yhteensopiva. Koska Microsoft .NET on suljettu järjestelmä ilman ulkopuolisten pääsyä sen lähdekoodiin tai dokumentaatioon, kehitetyt ominaisuudet tulivat aina hieman jälkijunassa Mono:n ominaisuuksiksi, eikä yhteensopivuudesta ollut aina takeita. Lisäksi halukkuutta sitoutua käyttämään Mono:a saattoi rajoittaa Microsoftin .NET:iin liittyvät ohjelmistopatentit. Monet alan asiantuntijat varoittelivat, että Microsoft olisi voinut haastaa oikeuteen Mono:n käyttäjät ja vaatia korvauksia ohjelmistopatenttiensa rikkomisesta. Microsoft antoi kyllä lupauksen, ettei se hyödynnä tätä mahdollisuutta, mutta tilanteen epäselvyys oli kuitenkin syy monille yrityksille ja organisaatiolle jättää hyödyntämättä Monoa. Sittemmin asiat ovat muuttuneet ja eri vaiheiden kautta Mono:a kehittävä Xamarin yritys on nykyään osa Microsoftia. Mono soveltuu edelleen .NET Core:sta huolimatta tiettyihin sovelluskehitysalueisiin, ehkä tunnetuimpana peliohjelmointi Unity-pelinkehitysympäristön avulla. Tämä siitä syystä, että .NET Core ei tue vielä kaikkia .NET:in sovelluskehitysalueita, eikä ole varmaa, tuleeko tukemaankaan (Scientific Programmer 18.8.2019).

4.3.2 React – webkäyttöliittymä-kirjasto

React on Facebook-yrityksen kehittämä komponenttikirjasto, jonka avulla voidaan toteuttaa interaktiivisia käyttöliittymiä. Pääasiallisin käyttökohde ovat web-sivuilla toimivat sovellukset, vaikka Reactia voidaan hyödyntää myös esim. Android tai iPhone-sovelluksissa.

Reactin ominaisuuksilla, kuten deklaratiivisilla näkymillä on mahdollista luoda ohjelmakoodia, joka on johdonmukaista ja josta virheiden havaitseminen on helppoa. React on komponentti-pohjainen ja sen ansioista sillä on mahdollista luoda monimutkaisia käyttöliittymiä (Reactjs.org 2020).

4.4 Avoin lähdekoodi

Käytetyissä teknologiavalinnoissa painottuu ensisijaisesti opinnäytetyön tekijän oma tietotaito ja toisaalta yhteensopivuus Careerian nykyisten teknisten ratkaisujen kanssa. Ohjelmiston pitkä elinkaari on myös yksi vaatimus, samoin kuin monikäyttöisyys ja tarvittaessa alustariippumattomuus. Tämä varmistetaan siis käyttämällä kehitysympäristönä Microsoftin uusinta, avoimen lähdekoodin alaisena julkaistua .NET Core -ohjelmistokomponenttikirjastoa (Framework). Myös itse produkti jaetaan avoimena lähdekoodina *MIT-lisenssin* alaisena.

Lähdekoodi löytyy

Palvelinosio:	https://github.com/jkurkisuonio/PQ-TiedonLaatuService
Käyttöliittymä:	https://github.com/jkurkisuonio/PQTiedonlaadun-UI

4.5 Azure-pilvipalvelut

Azure on yhteisnimitys Microsoftin pilvipalvelu-alustalle. Azure huolehtii sovellusten ylläpidosta vikasietoisin ratkaisuin, eli yhden palvelimen hajoaminen ei keskeytä palvelua – toisin kuin tyypillisessä omaan palvelimeen perustuvassa ratkaisussa. Azure antaa myös käyttäjälleen tietoa käyttöasteesta, sekä skaalautuu palvelun käytön mukaan. Azuren hinnoittelu perustuu käyttöön, joten alkuun pääsee ilman alkuinvestointeja (Get started guide for Azure developers, 18.11.2019).

4.6 Tiedon laadun hälytysjärjestelmä Primukseen v0.1

Versiossa 0.1 pyrittiin selvittämään keskeisten teknisten ratkaisujen toimivuus, jotta voidaan varmistua, että sovellus voidaan ylipäätänsä toteuttaa. Ensimmäisessä versiossa ei ole käyttöliittymää lainkaan, vaan tarvittavat asetukset muokataan suoraan tietokannasta, tai ne on *kovakoodattu* itse ohjelmaan.

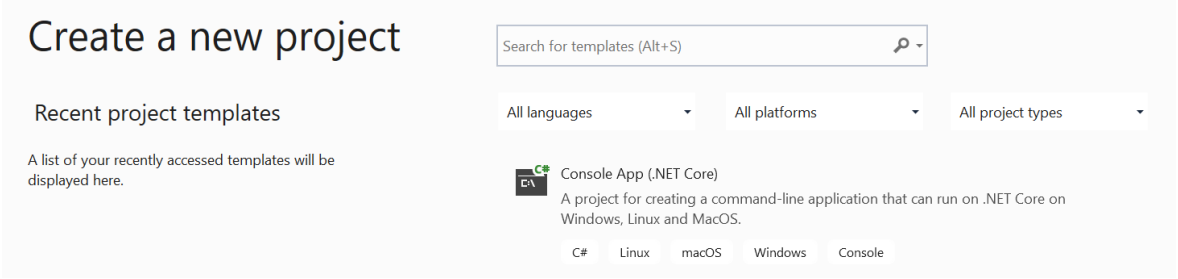
4.6.1 Versio 0.1 vaatimukset

- PrimusQuery-kyselyn suoritus ohjelmakoodista ja vastauksena saadun XML-muotoisen hälytystapahtuman *serialisointi* C#:n ymmärtämään oliomuotoon.

- Hälytysviestin muodostaminen serialisoidusta oliosta.
- Primus antaa vastauksensa suoraan konsolille, joka voidaan ohjata tiedostoon tai sitten lukea se kutsuvaan ohjelmaan. Vastauksesta tulee ilmetä: hälytyksen tyyppi, oppilas, jota hälytys koskee, vastaanottaja. Näiden lisäksi saadaan toki automaattisesti tietoja, kuten hälytyksen ajankohta, joka on hälytysohjelman suoritusajankohta.
    - PrimusQueryssä pystytään itse määrittelemään vastauksen muodon. Muoto on periaatteessa vapaa, koska muoto määritellään PrimusQuery-kyselyssä. Päädyin käyttämään XML muotoa, koska sen muodostaminen on helppoa ja sen serialisoinniksi C#-luokiksi löytyy valmis, helppokäyttöinen ratkaisu, XDocument-kirjasto.
- Yhteyden luonti Wilmaan Wilman JSON-rajapintaa hyödyntäen
- Wilma-viestin lähettäminen hälytyksen vastaanottajille

4.7 Palvelinosan kehittäminen

Koska Microsoftin aikaisempi sovelluskehitysympäristö .NET oli ollut minulle tuttu jo vuosien ajan, oli siirtyminen .NET Core:en etukäteen ajateltuna melko helppoa. Kehitystyökaluna on Microsoftin Visual Studio 2019. Siinä on valmiina tuki .NET Core:lle. Sinänsä .NET Core:han ei ole sidottu Visual Studioon, mutta Microsoftin pitkäikäisenä tuotteena se on kehitysympäristönä eittämättä helppokäyttöisin. .NET Core:sta löytyy valmis template .NET Core konsoli-sovellukselle (kuva 3).



Kuva 3. Uuden projektin luonti Visual Studiossa.

Windowsiin sidotun .NET ympäristön merkittävä vahvuus on ollut tarjolla olleet valmiit ratkaisut erilaisiin sovelluksen tarvitsemiin perus-infrastruktuuriratkaisuihin. Tällaisia ovat esimerkiksi käyttäjätunnistus ja sovelluksen asetusten tallentaminen keskitet-

tyyn konfiguraatiotiedostoon. Ensimmäinen perehdyttävä ongelma olikin selvittää, miten konfiguraatiotiedosto oli toteutettu uudessa .NET Core:ssa. .NET Core:ssa konfiguraatio on toteutettu uudella ConfigurationBuilder-luokalla, joka jättää kehittäjälle huomattavan vapaat kädet käytännön toteutukselle. Löysin valmiin esimerkin, jossa Configuration-builder luokalla luotiin *JSON*-formaattinen tiedosto, johon tarvittavat asetukset tallennetaan. Tarkoitus on toteuttaa ohjelmakoodissa hyviä käytänteitä, joihin kuuluu, että ympäristöspesifistiset asetukset tallennetaan erilliseen konfiguraatiotiedostoon, eikä kovakoodata ohjelmakoodiin. Tällaisia asetuksia ovat komentojonotiedoston sijainti, jota käytetään Primus-kyselyn käynnistykseen, Wilma-palvelimen osoite, käyttäjätunnus, yrityskohtainen API-avain ja salasana Wilmaan. Näiden lisäksi tarvitaan Microsoft SQL -palvelimen osoite, jonne perustetaan tietokanta Wilma-hälytyksiä varten. Aluksi testivaiheessa käytän paikallisesti asennettua Microsoft SQL Express -palvelinohjelmistoa. Tiedot tallennetaan oletusarvoisesti appsettings.json-nimiseen tiedostoon, jossa asetukset oli tallennettuna yksinkertaisin JSON-muotoisin avainparein:

Esimerkki appsettings.json tiedostosta:

```
{
 "application": {
  "SourceCmdFilename": "primuskysely.source",
  "DestinationCmdFileName": "primuskysely.cmd",
  "wilmaCompanySpesificKey": "xxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "wilmaUrl": "https://wilma.careeria.fi/",
  "wilmaUsername": "xxxxxxx ",
  "wilmaPasswd": "xxxxxxxxx"
 },
  "Logging": {
   "LogLevel": {
    "Default": "Information",
    "Microsoft": "Warning",
    "Microsoft.Hosting.Lifetime": "Information"
   }
  },
  "AllowedHosts": "*",
 "ConnectionStrings": {
  "PrimusAlertContext": "Serv-
er=localhost\\SQLEXPRESS;;Database=PrimusAlert;Trusted_Connection=True;Mu
ltipleActiveResultSets=true"
 }
 }
```

Ensimmäisessä versiossa ei vielä ollut tietokantayhteyttä, vaan tarkoitus oli saada toimimaan yhteydet sovelluksen eri rajapintojen, eli Primuksen ja Wilman välillä.

4.8 Tiedon laadun hälytysjärjestelmä Primukseen v0.2

Seuraavassa vaiheessa pitää mukaan ottaa tietokanta, johon hälytykset tallennettaisiin. Tallennetuista hälytyksistä voitaisiin seurata, kuinka kauan hälytykset ovat voimassa, eli kun sama hälytys toistuisi päivittäin, tietokantaan tallentuisi jälki tästä joka kerta kyseisen hälytyksen kohdalle. Näiden merkintöjen perusteella on mahdollista laskea ja raportoida virheiden korjaukseen mennyt aika. Tietokantaan tallennetaan myös tieto, onko jokin hälytys käytössä vai ei. Tämän avulla hälytyskäyttöliittymän käyttäjä voi kytkeä päälle ja pois hälytyksiä tarpeen mukaan.

4.8.1 Versio 0.2 vaatimukset

Saatujen hälytysten tallentaminen tietokantaan. Tietokanta tarvitaan, jotta voidaan seurata, kuinka pitkään jokin yksittäinen hälytys on ollut ajankohtainen. Tietokanta kerää siis tiedon, koska tietty tapaus havaittiin ensimmäisen kerran, sekä viimeisimmän kerran. Näiden päivämäärien erotus voidaan laskea ja on siis seurattavissa.

Ennen tietokantaa oli luotu vain yksi hälytys ja sen perusteella lähetetty Wilma-viesti (liite 3) oli kovakoodattu suoraan ohjelmakoodiin. Jotta Wilma-viestejä voisi olla useampia ja jotta niitä voitaisiin muokata, hälytysten laukaisemat Wilma-viestit pitää tallentaa tietokantaan. Viestit pitäisi myös tallentaa mallipohja-muodossa, jotta viestissä voi esiintyä merkittyjä sanoja, jotka korvataan viestejä lähetettäessä aina kulloiseenkin viestiin sopivalla merkkijonoilla. Päädyttiin seuraaviin avainsanoihin, jotka merkitään viestipohjaan %-merkkien välisinä:

%DateNow% =	nykyinen päivämäärä
%ReceiverEmail% =	Wilma hälytyksen vastaanottaja, esim. opettajan nimi
%AlertTypeName% =	Wilma hälytyksen tyypin nimi, esim. ”alle 18 vuotiaan opiskelijan tiedot puuttuvat.”
%StudentName% =	Opiskelijan nimi
%WilmaStudentUrl% =	Suora linkki Wilmaan kyseisen opiskelijan tietoihin.

Wilma-viestien tallennus tietokantaan vaikutti myös siten, että käyttöliittymäsovellusta tehtiin samanaikaisesti.

Tietokanta suunniteltiin HAAGA-HELIAssa tietokannatkurssilla opetettujen metodien avulla hyödyntäen Exceliä tietokantataulujen suunnitteluun. Sovelluksen tietokantaan tuli kolme taulua. AlertType-nimiseen tauluun tallennetaan tiedot hälytyksestä. Siellä tärkein on QueryName-kenttä, jonka perusteella Primuksesta suoritetaan saman niminen PrimusQuery.

4.9 Tietokannan toteutus - koodi ensin vai tietokanta ensin?

.NET Core, sisältää tietokantojen hyödyntämiseksi Entity Framework -nimisen sovelluskehikko-osan. Sen avulla voidaan luoda, muokata ja ylläpitää sen tukemia (mm. Microsoft SQL server) SQL-tietokantoja siten, että tietokantaobjekteja käsitellään ohjelmakoodissa .NET objekteina. Tätä menetelmää kutsutaan yleisesti Object Relation Mapping (ORM) –metodiksi. (Entity Framework Tutorial)

Lähestymistapoja tietokannan luomiseen on olemassa kaksi, tietokanta ensin tai koodi ensin (database first/code first). Tietokanta ensin tavassa luodaan sovellukselle tarvittava tietokanta tietokannan ylläpitotyökalulla. Microsoft SQL Serverin tapauksessa käytetään SQL Server Enterprise Manageria tai vaihtoehtoisesti käytetään jo valmista tietokantaa. tämän jälkeen .NET Coren Entity Frameworkin mukana tulevilla apuohjelmilla generoidaan tietokantaa mallintava C# olio-luokat tietokantakyselyä varten. (Entity Framework Tutorial) Tämä lähestymistapa on käyttökelpoinen varsinkin silloin, kun jollekin olemassa olevalle tietokannalle halutaan luoda esimerkiksi käyttöliittymä.

Koska tässä projektissa ei ollut olemassa valmista tietokantaa tälle projektille soveltui koodi ensin -lähestymistapa. Koodi ensin -lähestymistapa toimii tietysti päinvastoin kuin tietokanta ensin. Luodaan siis ensin C#-olioluokat, jotka edustavat normaalisti yksi-yhteen tietokantaan tulevia tietokantatauluja. Ohessa esimerkkinä tämän sovelluksen AlertType taulun C#-luokka:

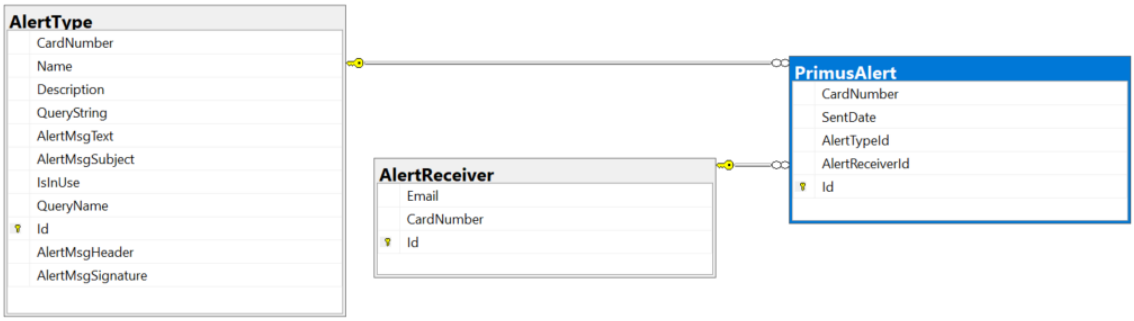
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace PQ_TiedonLaatuService.Models.Database
{
   public class AlertType
   {
      [Key]
      [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
      public int Id { get; set; }
      public string CardNumber { get; set; }
      public string Name { get; set; }
      public string Description { get; set; }
      public string QueryString { get; set; }
      public string QueryName { get; set; }
      public string AlertMsgHeader { get; set; }
      public string AlertMsgText { get; set; }
      public string AlertMsgSignature { get; set; }
      public string AlertMsgSubject { get; set; }
      public bool IsInUse { get; set; }
      public virtual ICollection<PrimusAlert> PrimusAlerts { get; set; }
   }
}
```

AlertType-luokassa tietokantataulun yksittäiset kentät ovat luokan muuttujia. Sen lisäksi luokka sisältää määreitä, attribuutteja kuten [Key] ja [DatabaseGenerated(DatabaseGenratedOption.Identity)], jotka kertovat, mikä kenttä on kyseisen taulun avainkenttä.

Entity-datamodel luo abstraktio-tason tietokannan ja sovelluksen väliin. Tämän ansiosta ohjelmoija ei esimerkiksi kirjoita SQL-lauseita tietokannan muokkaamiseksi, vaan Entity Framework toteuttaa sen. Toinen piirre on, että tietueet ovat staattisesti tyypitettyjä tietokannasta ohjelmakoodiin ja päinvastoin.

Kun tietokannan luokat on luotu, .NET Core:n Entity Frameworkin komentotyökaluilla käynnistetään prosessi, joka luo tietokantaan vastaavat taulut. Jos tietokantaan teh-

dään myöhemmin muutoksia, löytyy erikseen migraatio-prosessi, jossa tietokantaan tehdyistä muokkauksista luodaan versiohallinta (kuva 4).



Kuva 4. Tietokannan lopullinen ER-kaavio

4.10 Tiedon laadun hälytysjärjestelmä Primukseen v0.3

Seuraavaan versioon jouduttiin muuttaman tietokannan rakennetta jakamalla Wilma-viestiä varten varattu kenttä useampaan osaan. Käytännön testeissä huomattiin, että hälytykset kohdistuivat usein samaan opettajaan. Samalla opettajalla saattoi olla esimerkiksi koko luokallinen opiskelijoita, joilla kaikilla oli sama virhe heidän tiedoissaan. Näin jokaisesta syntyi kustakin yksi hälytys. Tästä syntyi suuri määrä hälytyksiä ja sitä myöten Wilma-viestejä. Jos näin olisi toimittu, Wilma-hälytykset olisivat pian täyttäneet opettajan Wilma-postilaatikon. Järjestelmää oli siis muutettava siten, että tietyn hälytystyypin hälytykset per vastaanottaja kerätään samaan viestiin ja yksittäisten viestien sijaan lähetetään yksi koontiviesti. Tällöin viestin alku ja loppu ovat samat, mutta keskiosassa on luettelona opiskelijat linkkeineen, joita kyseinen hälytys koskee.

4.10.1 Versio 0.3 vaatimukset

Informaatiotulvan välttämiseksi Wilma-viestejä tulisi lähettää vain yksi päivässä per vastaanottaja, silloin kun kyseessä on saman tyyppinen virhe. Saman tyyppiset hälytykset tulee siis kerätä samaan viestiin.

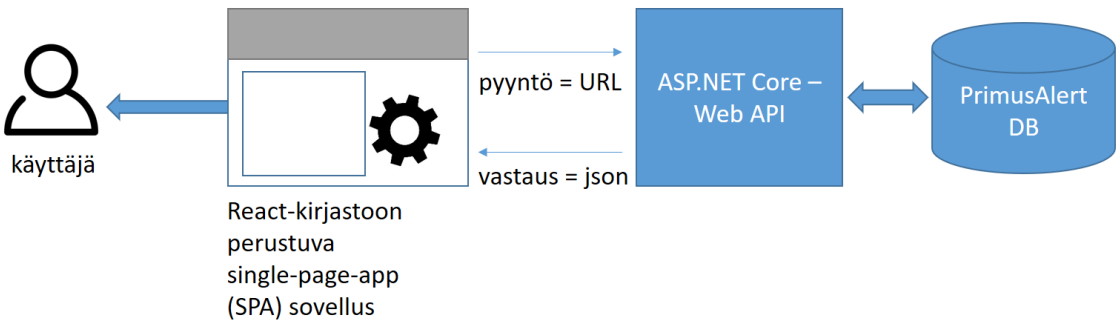
4.11 Käyttöliittymän luominen

Tietokannan luonnin yhteydessä todettiin, että käyttöliittymä on luotava valmiiksi ennen ohjelman käyttöönottoa. Ilman käyttöliittymää Wilma-viestien muotoilu olisi

enemmän kuin hankalaa (liite 4). Käyttöliittymä olisi siis luotava, jotta projektia voisi jatkaa.

Käyttöliittymää varten luotiin uusi Visual Studio 2019 –projekti ”PQTiedonlaadun-UI”. Visual Studiosta löytyy valmiiksi projektityyppi, jossa on yhdistettynä MVC-mallia toteuttava backend-osa käyttöliittymälle, sekä React Javascript -komponettikirjastolla toteutettava *SPA*, eli single page app-sovellus. Käyttöliittymäsovelluksen arkkitehtuuri näkyy kuvassa 2.

Single Page App -sovelluksessa käyttöliittymä toimii käyttäjän selaimessa Javascriptillä toteutettuna React-sovelluksena (kuva 5). Tämä poikkeaa siis tavallisista web-sivuista, joissa sivut ja käyttäjän näkymät muodostetaan palvelimella, josta ne lähetetään jokaisen käyttäjän tekemän muutoksen jälkeen uutena näkymänä. React-sovelluksessa käyttäjän selain lähettää ainoastaan tarvittaessa GET, PUT- ja POST-pyyntöjä palvelimelle. Pyynnöt käsitellään palvelimella ja vastaukset lähetetään takaisin *JSON*-formaattisina tietueina. Tämä toimintatapa tarjoaa monia etuja verrattuna perinteiseen tapaan, jossa kaikki käyttöliittymän toiminnot ja logiikka ovat palvelimella. Koska käyttöliittymä luodaan kokonaan käyttäjän selaimessa, tämä mahdollistaa perinteisiin web-sivuihin nähden dynaamisemman toteutuksen ja sovellus vastaa pyyntöihin nopeammin, kun erillisiä pyyntöjä palvelimelle ei tarvitse tehdä joka toimenpiteen vuoksi. Pyynnöt ovat myös tiedonsiirto kooltaan huomattavan pieniä, joten sovellus toimii esimerkiksi mobiilikäytössä nopeammin, kuin perinteinen sovellus.



Kuva 5. React-sovelluksen toteutusperiaate (mukaillen Smith J.P 13.9.2019)

Käyttöliittymä koostuu tällä hetkellä kolmesta sivusta. Home on oletussivu, jolla sovellus aukeaa. jossa on sovelluksen periaatekaavio. Varsinaiset toiminnot ovat kahdella

muulla sivulla. AlertType, josta kuvakaappaus (liite 5, kuva 1) sivulla voidaan luoda uusia hälytystyyppejä, sekä muokata vanhoja. Alerts sivu (liite 5, kuva 2) on raporttinäkymä, joka näyttää kaikki toteutuneet hälytystapahtumat. Raporttinäkymässä näkyy päivämäärä kenttä ”First Date”, joka kertoo, milloin hälytystapahtuma havaittiin ensimmäisen kerran ja ”Last Date” – milloin hälytystapahtuma on havaittu viimeisen kerran. Tämä näkymä näyttää myös siis hälytykset, joiden aikaansaama virhe- tai muu tilanne on jo korjattu. Tällöin LastDate-kentässä on jo mennyt päivämäärä. Virheet, jotka ovat vielä akuutteja erottaa päivämäärästä, joka vastaa menossa olevaa päivämäärää. Tällöin päivämäärä kertoo, että myös tänään kyseinen virhetilanne on todettu ja tästä on lähetetty Wilma-viesti. Days-kenttä kertoo ensimmäisen ja viimeisen päivämäärän välisen päiväeron lukuna. Tämän tiedon perusteella nähdään, kuinka kauan virheiden korjaus kestää päivinä. Raporttinäkymä perustuu Reactin valmiiseen taulukko-komponenttiin. Sen ansiosta tietoja voidaan järjestää minkä tahansa kentän mukaan tai suodattaa jonkin sarakkeen perusteella kirjoittamalla sarakkeen nimen alla olevaan laatikkoon sana, jonka alun perusteella kentässä olevat tiedot suodatetaan. *PQTiedonlaadun-UI* käyttöliittymäprojektiin liitettiin projektina myös *PQTiedonlaatuService*-palvelinosa, jonka avulla saadaan luotua tietokantayhteys PrimusAlert-hälytystietokantaan.

4.12 Sovelluksen autentikointi ja autorisointi

Koska käyttöliittymäsovellus näyttää ja käsittelee sensitiivistä tietoa, sen AlertType ja Alerts-sivut määriteltiin sivuiksi, joihin vaaditaan kirjautuminen ja käyttäjän tunnistaminen (autentikointi) sekä käyttöoikeudet (autorisointi). Microsoft .NET Core:ssa on olemassa valmis mekanismi tätä varten. Riittää, että metodeihin, tai ohjelmaluokkiin, jotka halutaan käyttäjätunnistuksen taakse, liitetään attribuutti [authorize]. Kun käyttäjä pyytää sivun tai palvelun, joka määritelty autentikoitavaksi, tarkistaa .NET Core, onko käyttäjä kirjautunut ja jos on, onko hänellä riittävät oikeudet käyttää ko. palvelua. Jos käyttäjä ei ole kirjautunut, hänet lähetetään kirjautumissivulle.

5 Johtopäätökset ja pohdinta

Opinnäytetyön lopputuloksena syntyi Careerialle käyttökelpoinen järjestelmä oppilashallinnon virheiden tunnistamiseen ja niistä raportointiin. Lisäksi järjestelmän lähdekoodi julkaistiin avoimen lähdekoodin MIT-lisenssin alaisena, jolloin sitä voidaan hyödyntää vapaasti alkuperäislähde mainiten. Näin siitä voi olla hyötyä myös muille Primusta käyttäville oppilaitoksille. Linkki lähdekoodiin ja lyhyt seloste ohjelmasta julkaistiin Wisman Primus-InSchool tukisivuston keskustelufoorumilla 14.5.2020 osoitteessa: https://community.visma.com/t5/InSchool-Ammatilliset/Primus-oppilashallinnon-tiedon-laadun-halytysjarjestelma/m-p/298253#M2925

Sen lisäksi, että ohjelmaa voivat hyödyntää ja edelleen kehittää muut Primusta käyttävät tahot, ohjelman lähdekoodin julkaisu oli hyödyllistä myös muulla tavalla. Julkaisuun varautuminen vaikutti siihen, että sovelluskehitys ja sen dokumentaatio tulee tehtyä paremmin ja kattavammin, kuin jos ohjelma olisi tehty pelkästään toimeksiantaja Careerian näkökulmasta.

Ohjelmaa ei saatu tuotantokäyttöön opinnäytetyön valmistumiseen mennessä ja näin Azure-pilvipalvelun hyödyntäminen ja siitä kokemusten saaminen jäi opinnäytetyön ulkopuolelle. Demo versio saatiin kuitenkin toimintaan ja toimeksiantajan antaman lausunnon (liite 5) mukaan etäpalaverissa 23.4.2020 esitelty versio vastasi sinänsä annettuja määrittelyjä hyvin ja tarjosi ainekset jatkokehitykselle.

5.1 Jatkokehitys

Ohjelma on tällaisenaan käyttökelpoinen ja toimeksiantaja ottaa sen käyttöön, mutta vaatii asennusta varten muutoksia. Tärkein muutos on käyttöliittymän autentikointi, sillä oppilaitoksilla – Careeria mukaan lukien – on käytännössä aina olemassa käyttäjätietokanta, jota halutaan käyttää käyttäjien tunnistamiseen.

Ohjelman kehityksen aikana on tullut ainakin seuraavia tarpeita:

- Käyttöliittymän kieliversiointi. Ainakin suomenkielinen versio olisi suotava.
- Raportointi

- Intranet raportointi. Käyttöliittymäsovelluksessa voisi olla oma API, jonka avulla Intranet sovelluksesta voisi hakea kulloisenkin käyttäjän kesken olevat tapaukset näkyviin omassa, henkilökohtaisessa Intranet-näkymässä
- Raportti, joka näyttäisi parhaillaan ”auki” olevat virheet
- Raportin vientimahdollisuus jatkokäsittelyä varten, esim. .csv formaatissa, jolloin siirrettävissä esim. Excel-taulukkolaskentaohjelmaan

Lähteet

Ammatillisen koulutuksen Reformi. AMKE. Luettavissa: https://www.amke.fi/media/julkaisuja/ammatillisen-koulutuksen-reformi-2017.pdf. Luettu 29.4.2020.

Arter 2020. ARC-Ohjelmisto. Luettavissa: https://www.arter.fi/ohjelmistot/arc-ohjelmisto/. Luettu 2.5.2020.

ASP.NET documentation. Dot.Net Core. Luettavissa: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1. Luettu 15.5.2020.

Careeria 2020. Tarina Careerian takana. Luettavissa: https://careeria.fi/careeria. Luettu: 17.4.2020.

Careerian Primus pääkäyttäjä, 29.4.2020. Järjestelmäassistentti. Careeria. Haastattelu. Porvoo.

Englund, E. 21.4.2019. Anti-Patterns and Code Smells. Luettavissa: https://itnext.io/anti-patterns-and-code-smells-46ba1bbdef6d. Luettu 7.5.2020.

Entity Framework Tutorial. Creating a Model for an Existing Database in Entity Framework Core. Luettavissa: https://www.entityframeworktutorial.net/efcore/create-model-for-existing-database-in-ef-core.aspx. Luettu 15.5.2020.

Entity Framework Tutorial. What is Entity Framework? Luettavissa: https://www.entityframeworktutorial.net/what-is-entityframework.aspx. Luettu 15.5.2020.

Eppler Martin J. 2006. Managing Information Quality

Finlex 2017. Laki ammatillisesta koulutuksesta. Luettavissa: https://www.finlex.fi/fi/laki/alkup/2017/20170531. Luettu 29.4.2020.

Finlex 2017. Laki valtakunnallisista opinto- ja tutkintorekistereistä. Luettavissa: http://finlex.fi/fi/laki/ajantasa/2017/20170884. Luettu 29.4.2020.

Fisher, Zwass, Madnik, Wang, Pierce 2005. Information Quality

KOSKI. Opetushallitus. Luettavissa: https://confluence.csc.fi/display/OPHPALV/Koski. Luettu: 29.4.2020.

KOSKI-tietovaranto. Luettavissa: https://www.oph.fi/fi/palvelut/koski-tietovaranto. Luettu 11.4.2020.

Microsoft Azure Developer Guide, 18.11.2019. Get started guide for Azure developers. Luettavissa: https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer-guide. Luettu: 7.5.2020.

Opetus- ja kulttuuriministeriö 2017. Toisen asteen koulutuksen uudistaminen. Luettavissa: http://www.pilkahdus.fi/sites/default/files/tammilehto220814.pdf. Luettu: 26.4.2020

Opetus- ja kulttuuriministeriö 5.3.2017. Hallituksen vastaus välikysymykseen ammatillisen koulutuksen leikkausten aiheuttamista seurauksista ja koulutuksen eriarvoistamisesta. Luettavissa: https://minedu.fi/artikkeli/-/asset_publisher/hallituksen-vastaus-valikysymykseen-ammatillisen-koulutuksen-leikkausten-aiheuttamista-seurauksista-ja-koulutuksen-eriarvoistumisesta. Luettu 29.4.2020.

Opetushallitus. 2020. Tiedolla johtaminen. Luettavissa: https://www.oph.fi/fi/kehittaminen/tiedolla-johtaminen. Luettu 15.5.2020.

Opetushallitus. Organisaation ID (OID)-koodin luominen. Luettavissa: https://www.oph.fi/sites/default/files/documents/oid-ohje03122019.pdf. Luettu 11.4.2020.

Opetushallitus. Tutkintojen perusteet. Luettavissa: https://www.oph.fi/fi/koulutus-ja-tutkinnot/tutkintojen-perusteet. Luettu 11.4.2020.

ReactJs.org 2020. React – A JavaScript Library for building interfaces. Luettavissa: https://reactjs.org/. Luettu 30.4.2020.

Siniranta, M. 2009. OLAP-tekniikan hyödyntäminen oppilashallintojärjestelmässä. Luettavissa: http://urn.fi/URN:NBN:fi:amk-201003064787 . Luettu: 28.4.2020.

Smith, J. P. 13.9.2019. Entity Framework Core in Action. Luettavissa: https://www.thereformedprogrammer.net/how-to-write-good-testable-asp-net-core-web-api-code-quickly/. Luettu 9.5.2020.

Systeemityö 2/2009. Tietojärjestelmän laadun ratkaisevat tietosisällön oikeellisuus ja merkitys. Luettavissa: http://www.sytyke.org/lehtiarkisto/kirj/st20092/ST092-15A.pdf. Luettu: 17.4.2020.

Visma 2020. Dashboard avasi silmät ja vapautti Exceleistä. Luettavissa: https://www.visma.fi/blog/dashboard-avasi-silmat-ja-vapautti-exceleista/. Luettu: 25.4.2020.

Liitteet

Liite 1. Sanasto

Liite 2. Muistiot palavereista

Liite 3. Esimerkki Wilma-viesti

Liite 4. Kuvakaappaukset sovelluksesta

Liite 5. Toimeksiantajan arvio demo-versiosta

Liite 1. Sanasto

Ammatillisen koulutuksen rahoituksen perusteet	Ammatillisen koulutuksen rahoituksessa on asteittain siirrytty suoritus- ja laatu- sekä vaikuttavuusperusteisuuteen.
Avoin lähdekoodi	Avoin lähdekoodi tarkoittaa ohjelmistoa, jonka lähdekoodi on saatavilla ja vapaasti hyödynnettävissä.
Deserialisointi	Päinvastainen operaatio kuin serialisointi, ks. serialisointi.
HOKS	Henkilökohtaisen osaamisen kehittämissuunnitelma. Laaditaan jokaiselle ammatillisen koulutuksen opiskelijalle ja päivitetään koko opiskelun ajan.
KOSKI-palvelu	KOSKI-tietovarannossa ylläpidetään keskitetysti kaikkien suomalaisten opintosuorituksia arvosanoineen. KOSKI on opetusministeriön palvelu, joka aloitti toimintansa v. 2018.
Kovakoodaus	Tietokoneohjelman ympäristöön liittyvät asetukset on kirjoitettu suoraan ohjelmakoodiin, eikä niitä voi muuttaa kääntämättä ohjelmaa uudelleen. Tämä tekee ratkaisusta jäykän, mutta käytetään usein ajan säästämiseksi ohjelmiston kehityksen alkuvaiheessa.
MVC	Model – View – Controller on ohjelmistokehitysarkkitehtuuri, jossa ohjelman tietomalli (model), näkymä (view) ja ohjauslogiikka (controller) on erotettu toisistaan.
Ohra	Careerian edeltävällä Edupoli-oppilaitoksella käytössä ollut oppilashallinto-ohjelmisto
OID	Organisaatioiden yksilöintiin käytetty ISO-standardoitu yksilöintikoodi (OID-yksilöintitunnukset).
OK2000	Careeriaa edeltävällä oppilaitoksella ollut oppilashallinto-ohjelmisto.
Primus-oppilashallintojärjestelmä tai Primus	Eräs, melko yleisesti myös ammatillisten, toisen asteen oppilaitosten, käyttämä tietojärjestelmä oppilaitoksen toiminnan ohjausta varten.

Reformi	Viitataan ammatillisen koulutuksen reformiin. Opetusministeriön käynnistämä, v. 2018 loppuun saatettu ammatillisen koulutuksen uudistus (Amisreformi).
Serialisointi	Ohjelmoinnissa käytetty termi, jolla tarkoitetaan ohjelmassa olevan datan muuttamista merkkijonoksi ja takaisin.
SPA, Single-Page-App	Nimitys web-sivu sovellusarkkitehtuurille, jossa koko web-sivustolla toteutettava sovellus on yhdellä web-sivulla ladattava Javascriptillä toteutettu sovellus.
SQL Integration Service	Microsoftin SQL Serverissä oleva työkalu tietokannan tietojen siirtoon.
SQL Server Agent	Työkalu, jolla voidaan ajastaa Microsoft SQL Serverissä tietokantoihin liittyviä ylläpitotoimenpiteitä.
SQL-kieli	Standardoitu komentokieli, jonka avulla luodaan, muokataan ja luetaan relaatiotietokantoja ja niissä olevaa dataa.
Studenta	Careeriaa edeltävällä Edupolilla v. 2013 asti käytössä ollut oppilashallinto-ohjelmisto.
StudentaPlus	Careeriaa edeltävällä Edupolilla v. 2018 saakka käytössä ollut oppilashallinto-ohjelmisto.
Tietokantamoottori	Palvelinsovellus, joka ylläpitää tietokantaa palvelimella sovellusten käytettäväksi, esimerkiksi Microsoft SQL Server, MySQL tai Oracle. Myös Primus-sovelluksen takana on oma Primus-tietokantamoottori.
Yhteensopivuus	Opetushallituksen 1.1.2019 käynnistämä kaksivuotinen hanke ammatillisten oppilaitosten toimintamallien yhtenäistämiseksi. Hankkeen tavoitteena on, että oppilaitokset voisivat helpommin vaihtaa keskenään tietoa järjestelmiensä kautta. Tällä pyritään oppilaitosten välisen yhteistyön lisäämiseen ja syventämiseen.
YTO	Yhteiset tutkinnon osat tulivat kaikkiin toisen asteen ammatillisiin tutkintoihin pakollisiksi osiksi v. 2018 Reformin yhteydessä (ks. Reformi). Yhteisiä tutkinnon osia on kolme: viestintä- ja vuorovaikutusosaaminen, matemaattis-luonnontieteellinen osaaminen ja yhteiskunta- ja työelämäosaaminen.

Liite 2. Palaverimuistiot

Projektin kickoff palaveri 29.1.2020

Paikka: Careerian toimipiste, Porvoo

Läsnä: Careerian Primus/Wilma-pääkäyttäjät ja edustajat, opinnäytetyön toimeksiantaja, palvelupäällikkö Päivi Koponen

Aihe: Käydään läpi toimeksiantajan, Careerian vaatimukset.

Aloituskokouksessa Käydään läpi toimeksianto, vaatimusmäärittely ja ohjelman kuvaus. Pääkäyttäjät esittelevät Excel-taulukkoa, johon he ovat keränneet ongelmatilanteet, joihin halutaan luoda hälytykset. Sovitaan minimitoteutuksen sisällöstä.

Projektin esittely 23.4.2020

Paikka: Teams-etäkokous

Läsnä: Careerian edustajat (kolme hlöä), allekirjoittanut

Aihe: Tutustuminen tuotantoon otettavaan versioon, jatkoaikataulusta sopiminen

Kävin läpi sovelluksen keskeiset toiminnot ja samalla esittelin tuotoksen ryhmän uudelle jäsenelle, Primus-pääkäyttäjien esimiehelle.

Esittelin käyttöliittymän, kerroin teknologiavalinnoista (React, .NET Core) ja syistä niiden valintaan. Toimeksiantajan edustajat olivat tyytyväisi näkemäänsä. Kysymyksiä herätti, voiko viestiä muokata, voisiko järjestelmästä saada raporttia esimerkiksi Microsoftin PowerBI järjestelmään. Esitin, että järjestelmään voisi lisätä toiminnon, jolla raportin saisi Excelin ymmärtämään .csv formaattiin, jolloin raporttia voisi jälkikäsitellä Excelissä tai vaikkapa PowerBI:ssä. Tätä pidettiin hyödyllisenä jatkoa ajatellen. Esitettiin myös kysymys, miten havaitaan, jos järjestelmä ei toimi. Keskusteltiin, että tälle on saatava ylläpitorutiini, jossa valvotaan säännöllisesti, että järjestelmä toimii. Havaitsin, että tähän käyttökelpoinen ratkaisu on, että järjestelmään tarvitaan myös tietokantaan merkintä ja näkymä käyttöliittymään, milloin Wilma-hälytysten eräajo on ajettu. Samalla järjestelmä näyttäisi virheilmoituksena, mikäli jokin päivittäinen ajo on jäänyt pois välistä.

Päätökset: Päätettiin, että sovellus voidaan ottaa tuotantokäyttöön tällaisenaan ja loput toiveet jätetään seuraaviin versioihin. Päätettiin aikataulusta seuraavaa:

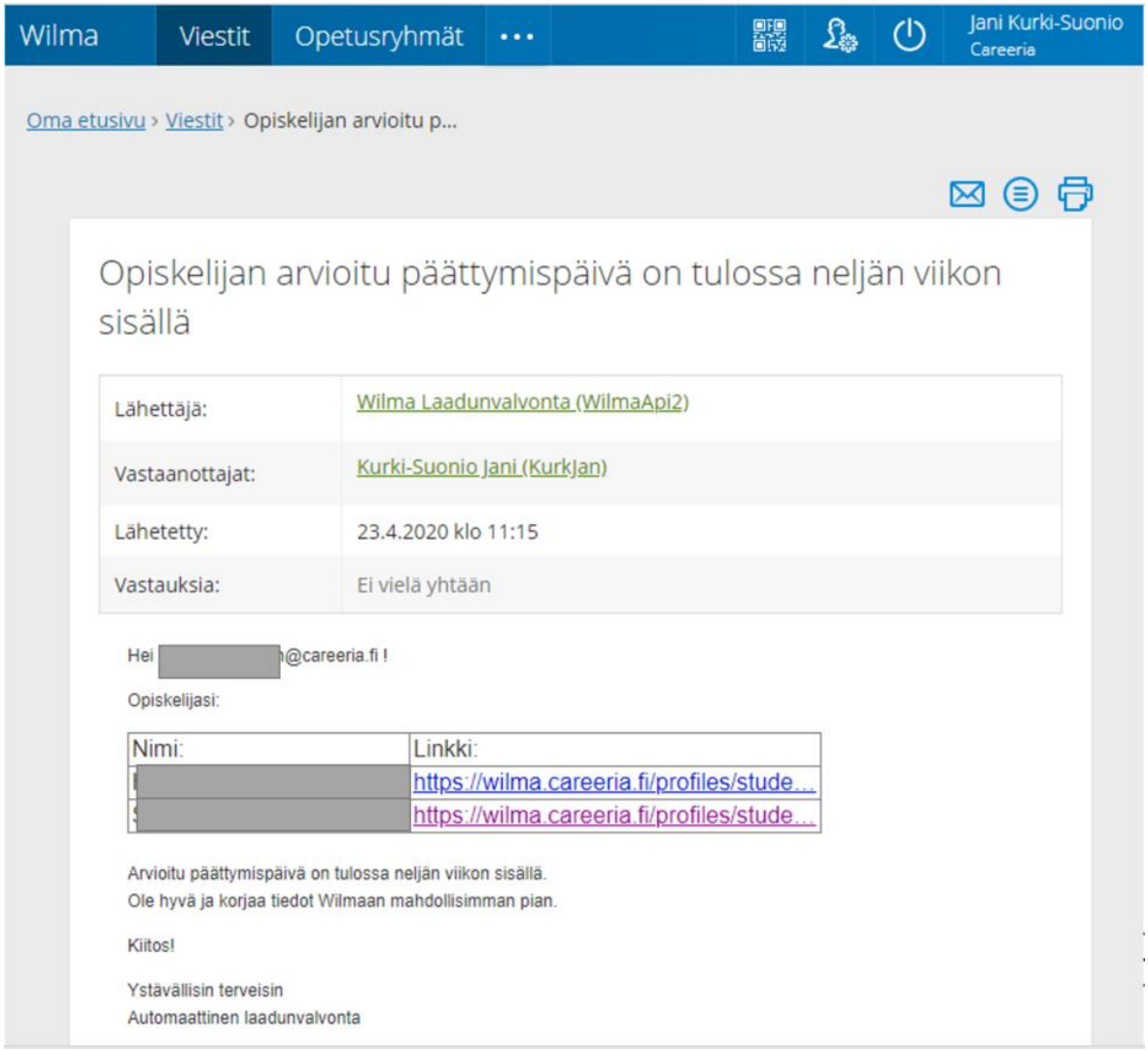
23.4.2020 Toimitan Careerian edustajille sähköpostitse Wilma-muistutusviestien pohjan, johon toimeksiantajan edustajat miettivät sopivan viestimuodon.4.5 – 5.5.2020 Viestipohjat viimeistellään ja järjestelmä asennetaan Careerian palvelimille.

7.5.2020 Järjestelmä esitellään Careerian henkilökunnalle CareeriaLive!- Careerian henkilökunnan sisäisessä tiedotustilaisuudessa.

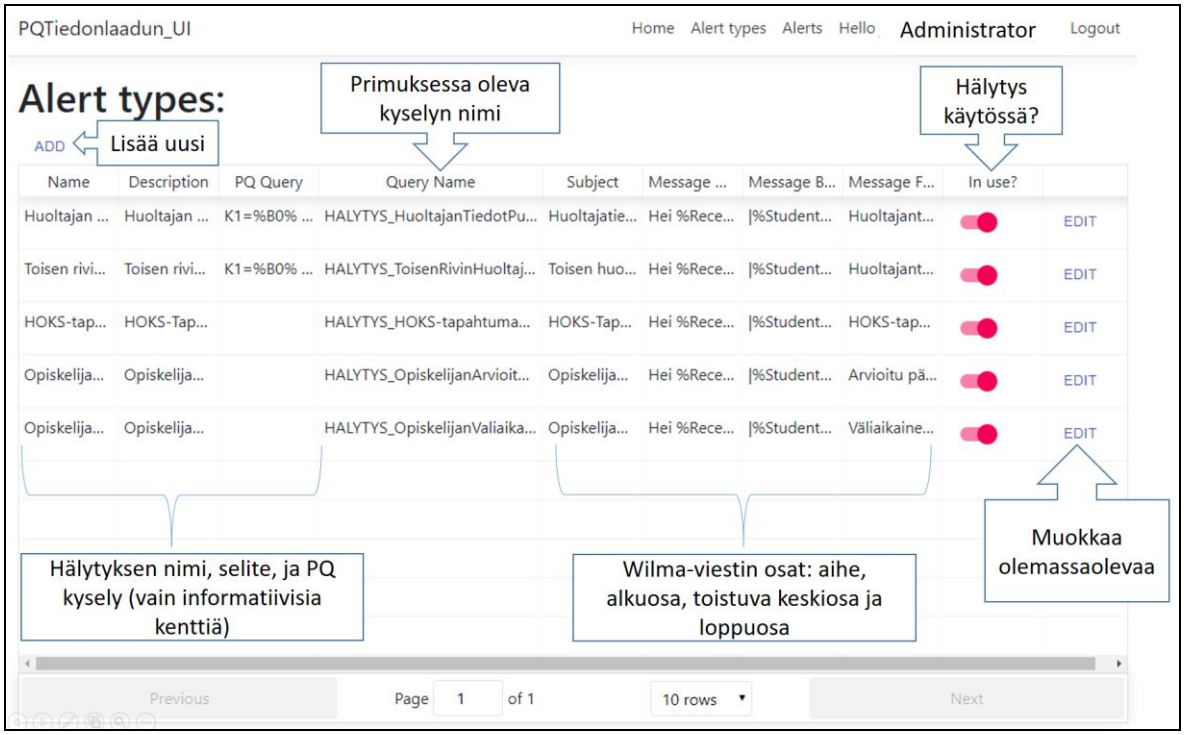
18.5.2020 Järjestelmä otetaan käyttöön

19.5.2020 Järjestelmän ensimmäiset Wilma-viestit tulevat käyttäjille

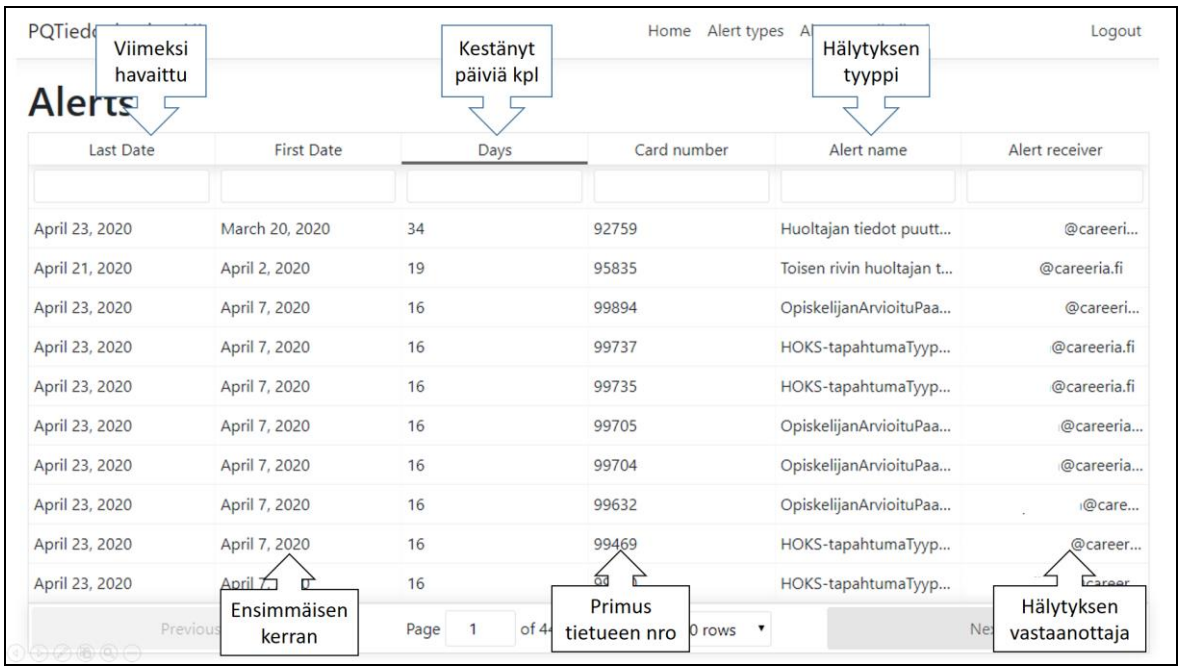
Liite 3. Esimerkki Wilma-viesti:



Liite 4. Kuvakaappaukset käyttöliittymäsovelluksesta



Kuva 1. Hälytys-tyyppien näkymä.



Kuva 2. Toteutuneiden hälytys-ilmentymien näkymä.

Liite 5. Toimeksiantajan lausunto Demo-versiosta

Tiedonlaadun hälytysjärjestelmä

Käytössämme on Primus-Wilma-Kurre-opintohallintojärjestelmä, joka sellaisenaan ei tuota erilaisista ei-toivotuista tilanteista hälytyksiä tai muistutuksia käyttäjilleen. Opettajilla on melko tyypillisesti eri opintojen vaiheissa olevia opiskelijoita vastuullaan. Yksilöllinen opiskelijoiden eteneminen tuo haasteita opettajan arkeen, ja opiskelijan prosessin etenemiseen sidotut dokumentointi- ja tapahtumavaatimukset tuovat työn organisointiin ja aikataulutukseen haasteita. Jani Kurki-Suonion opinnäytetyö varmistaa laadunhallintaamme hälytyksin oikea-aikaisesti ja kohdennettuna niille henkilöille, joiden vastuulla ko. prosessin vaihe on.

Jani esitteli tiedonlaadun hälytysjärjestelmän demoversion etäpalaverissa 23.4., palaverissa totesimme, että järjestelmä vastaa annettua tehtävää ja määrittelyjä hyvin ja siinä on aineksia jatkokehitykselle. Opinnäytetyöhön valitut hälytykset ovat prosessin kannalta tärkeitä prioriteetiltaan. Järjestelmä tukee hyvin henkilökohtaistamisprosessin laadunhallinnan kehittämistä, joka on yksi kehittämisen painopisteistämme. Tiedonlaadun hälytysjärjestelmälle oli ja on selkeä tarve, johon Janin tekemä järjestelmä tuo ratkaisun. Organisaation eri tasoilla tiedonlaadun ajantasaisuuden ja oikeellisuuden varmistamisen tukeminen nähdään prosessia sujuvoittavana ja varmistavana työvälineenä, jolle on ollut selkeä tarve.

11.5.2020 Porvoo

Päivi Koponen
laatupäällikkö
Careeria Oy