SAVONIA
ammattikorkeakoulu

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

RAPORTOINTITYÖKALUN KEHITYS

TEKIJÄ/T Tuomas Kauhanen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Tuomas Kauhanen	
Työn nimi Raportointityökalun kehitys	
Päiväys 11.5.2022	Sivumäärä/Liitteet 29
Toimeksiantaja/Yhteistyökumppani(t) Istekki Oy	

Tiivistelmä

Istekin tietoliikennepalveluilla oli käytössään laite- ja valvontajärjestelmätietojen tarkastamiseen kehitetty työkalu, jonka toiminnallisuuksissa on ollut puutteita. Puutteet ovat johtuneet Istekin toiminnanohjausjärjestelmän vaihtumisen jälkeisistä muutoksista aiemman työkalun toimintaperiaatteisiin. Niiden vuoksi laite- ja valvontajärjestelmätietojen tarkastukset eivät ole tuottaneet luotettavia tuloksia. Työkalussa käytetty koodipohja oli myös syytä päivittää, jotta käytetyt ohjelmointikirjastot ja -kieli olisivat ylläpidettyjä. Tavoitteena oli kehittää kokonaan uusi työkalu, jota olisi helpompi ylläpitää jatkossa ja lisätä siihen mahdollisia muita laite- tai järjestelmätarkastuksia tarpeen mukaan. Tavoitteena oli myös, että tietojen tarkastus tulisi tehtyä luotettavasti ja sillä saataisiin eri järjestelmien välisistä tiedoista yhteneväisiä.

Tässä opinnäytetyössä kehitettiin Istekin tietoliikennepalveluiden käyttöön laite- ja valvontajärjestelmien tarkastamiseen tarkoitettu työkalu. Työkalu tarkastaa suoraa API-rajapintaa käyttäen kaikki tuotannonohjausjärjestelmään tuotantoon merkityt verkkolaitteet ja tekee määritellyt tarkastukset haetuille laitteille. Tarkastukset määriteltiin suunnitteluvaiheessa tärkeimpiin laitetietoihin, joita asiantuntijat käyttävät päivittäisessä työssään sekä kriittisimpiin järjestelmiin, joilla on merkitystä työn sujumisen kannalta. Työkalu toteutettiin Python-ohjelmointikielellä, sillä se on tehokas ja käytössä myös muissa toimeksiantajan oman kehitystyön työkaluissa. Python on myös nopea oppia, joten se oli perusteltu valinta toimeksiantajalle, jolle päivittäinen työ ei sisällä varsinaisesti ohjelmointitehtäviä.

Tarkastusten jälkeen työkalu muodostaa saaduista tuloksista tarkastusraportin, joka tallennetaan tiedostoon. Tarkastusraportille tallennettiin yksittäisen laitteen tarkastuskohteiden lisäksi kooste kaikkien tarkastusten tuloksista. Näitä tuloksia voidaan käyttää jatkokehityksenä mahdollisesti kehitettävän käyttöliittymän tarpeisiin monipuolisesti. Tällä hetkellä toimeksiantajan tavoitteena oli saada muodostettua tarkastuksista luotettava raportti, jota voitaisiin tarpeen mukaan jatkossa jalostaa. Työkalun avulla laite- ja valvontajärjestelmätietoihin saadaan pidemmällä aikavälillä yhteneväisyyttä ja luotettavuutta.

Avainsanat

Istekki, tietoliikenne, raportointityökalu, Python

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Tuomas Kauhanen	
Title of Thesis Development of the Reporting Tool	
Date 11 May 2022	Pages/Appendices 29
Client Organisation /Partners Istekki Oy	

Abstract

The aim of this thesis was to develop a new reporting tool with reliable data for the client organization, Istekki Oy. The reporting tool is used to get all the necessary device data from network devices and other related systems. The need for development of the new reporting tool was due to the change in the client organization's service management system, which caused problems considering the reliability of the data gathered by the current reporting tool. The target was that it would be easier to update and add more features to the reporting tool in the future. Additionally, it was necessary to update the programming language and the libraries used in the development process to match the current needs.

In this thesis, a replacing reporting tool was created for the client organization. The tool uses service management system's application programming interface to fetch all network devices that match the criteria. After that it does all the necessary checks for the devices. The checks were designed so that the most important data for employees and for the critical systems would be checked. Those are the most important features that are needed daily. The tool was developed with Python, which is an effective programming language and is used in other programs by the networking team of the client organization. Python is also very fast to learn which is a good feature when daily work does not usually involve programming tasks.

After the checks have been made the reporting tool forms and saves the report as a file. In addition of single device results, the reporting tool also forms a summary of the results. These results can be used in several ways in further development, for example to use in a graphical user interface which will be developed in the future. The current aim of the client organization was to create a reliable report after the checks. That report could be then developed further to match the needs. By using this new reporting tool, reliable and consistent data from the device and monitoring systems will be available for the networking team of the client organization.

Keywords
Istekki, networking, reporting tool, Python

SISÄLTÖ

KÄYTETYT LYHENTEET 5
1 JOHDANTO 6
2 TYÖKALUUN LIITTYVÄT JÄRJESTELMÄT 7
2.1 Efecte-toiminnanohjausjärjestelmä 7
2.1.1 Palvelunhallinta 7
2.1.2 Laskutus 7
2.1.3 Konfiguraationhallintatietokanta 7
2.2 Valvontajärjestelmät 8
2.3 Varmuuskopiointijärjestelmä 9
2.4 Langattoman verkon ohjaimet 10
3 SUUNNITTELU 11
3.1 Nykyinen työkalu ja haasteet 11
3.2 Järjestelmistä tarkastettavat tiedot 11
3.3 Ohjelmointiympäristö 14
3.4 Haettujen tietojen hyödyntäminen 16
4 TOTEUTUS 18
4.1 Kehitysympäristön käyttöönotto 18
4.2 Toiminnallisuudet 18
4.2.1 Tietojen haku Efectestä 19
4.2.2 Tietojen tarkastukset haetuilla laitetiedoilla 20
4.2.3 Virheenkäsittely 22
4.3 Raportin koostaminen 22
5 YHTEENVETO 24
6 LÄHDELUETTELO 29

KÄYTETYT LYHENTEET

API = Application Programming Interface, ohjelmointirajapinta tietojen siirtämiseen eri järjestelmien välillä

BGP = Border Gateway Protocol, reititysprotokolla autonomisten järjestelmien välillä

CMDB = Konfiguraationhallintatietokanta (Configuration Management Database), yrityksen laite- ja ohjelmistokanta

DNS = Domain Name System, Internetin nimipalvelujärjestelmä, muuntaa verkkotunnukset IP-osoitteiksi

EQL = Efecte toiminnanohjausjärjestelmän kyselykieli (Efecte Query Language)

FQDN = Fully Qualified Domain Name, täydellinen piirinimi

ICMP = Internet Control Message Protocol, hyödynnetään virheviestien ja tilatietojen lähettämisessä laitteilta

ITSM = Information Technology Service Management, IT-alan palveluhallinta

IP = Internet Protocol, verkkoprotokolla tietoliikennepakettien toimittamiseen

ISIS = Intermediate System to Intermediate System, reititysprotokolla tehokkaaseen verkon sisäiseen liikenteeseen

JSON = JavaScript Object Notation, kevyt standardoitu tekstimuotoinen tiedostoformaatti

MAC = Media Access Control, laitteen verkkosovittimen yksilöivä tunnus eli MAC-osoite

OSPF = Open Shortest Path First, reititysprotokolla optimaalisen reitin löytämiseen

SNMP = Simple Network Management Protocol, protokolla, jolla voidaan kysyä laitteen tilaa tai laite voi hälyttää

VS Code = Visual Studio Code, avoimeen lähdekoodiin perustuva ohjelmoijille tarkoitettu tekstieditori

WLC = Wireless LAN Controller, langattoman verkon ohjain

1 JOHDANTO

Istekin tietoliikennepalveluilla oli käytössään työkalu, joka tarkasti verkkolaitteiden laitetiedoista olivatko tiedot täytettyinä ja ilmoitti mahdollisista puutteista sähköpostiviestillä. Toimeksiantajan toiminnanohjausjärjestelmän muutoksen jälkeen työkalua pyrittiin päivittämään samaa koodipohjaa hyödyntäen, mutta työkalun toiminnassa oli havaittu ongelmia ja käytetty ohjelmointikieli sekä kirjastot olivat jo ilman ylläpitoa. Yrityksen käytössä on useita järjestelmiä, joiden käyttöön toiminnanohjausjärjestelmässä olevan laite- ja ohjelmistokannan (CMDB) dataa tulisi saada hyödynnettyä, mutta tällä hetkellä järjestelmien tiedot ovat poikkeavia tietokannan tietoihin nähden.

Tämän opinnäytetyön aiheena oli korvata puutteellisesti toimiva raportointityökalu. Työssä tehtiin Istekin tietoliikennepalveluiden käyttöön työkalu, jonka avulla laite- sekä valvontajärjestelmätiedot saatiin tarkastettua automaattisesti. Työkalu hyödyntää konfiguraatiohallintatietokannan datan tarkastamisessa suoraa API-rajapintaa, jonka avulla saadaan siirrettyä tietoja eri järjestelmien välillä, mikä puolestaan varmistaa ajantasaisen tiedon kyseiselle tarkastushetkelle. Myös laskutustietojen tarkastaminen laitteilta työkalun avulla oli asiakkaan toiveena ja toteutettiin työkalulla. Tarkastusten perusteella voidaan yksittäisten laitteiden jokaiselle tarkastuskohteelle muodostaa tarkastusraportille yksilöllinen tulos, jota voidaan tulkita joko suoraan tiedostoon tallennetusta raportista tai jatkokehityksenä tehtävästä käyttöliittymästä.

Työkalun merkitys ryhmän toiminnalle on merkittävä, mutta tulee todennäköisesti näkyviin vasta viiveellä. Työkalun tarkoituksena oli auttaa asiantuntijaa pitämään CMDB:n laitetiedot ajan tasalla ja täytettyinä. Tarkoituksena oli myös yhteneväinen tieto eri järjestelmien välillä ja työkalun laajennettavuus jatkossa. Laitetietojen täyttäminen on tärkeää, jotta asiantuntijoilla olisi mahdollisuus esimerkiksi tietää missä kyseinen laite sijaitsee, mikä sen IP-osoite tai isäntänimi on. Suuremmassa mittakaavassa eroavaisuudet tiedoissa voisivat näkyä myös pidentyneenä palveluvasteena häiriötilanteissa, mikä vaikuttaisi merkittävästi asiakaskokemukseen. Laskutustietojen täyttäminen ajantasaisesti varmistaa puolestaan, että myydyt palvelut ja tuotteet näkyvät myös rahavirrassa yritykselle heti tuotantoon ottamisen jälkeen. Toteutetun työkalun avulla voitiin reagoida nopeammin muun muassa edellä mainittuihin epäkohtiin.

2 TYÖKALUUN LIITTYVÄT JÄRJESTELMÄT

2.1 Efecte-toiminnanohjausjärjestelmä

Efecte on pilvipohjainen IT-palvelunhallintajärjestelmä. Järjestelmän avulla esimerkiksi asiakkailta tulevat häiriöilmoitukset ja palvelupyynnöt saadaan käsiteltyä hallitusti. IT-ympäristöissä on erittäin kriittistä pitää laite- ja järjestelmätiedot ajan tasalla, mikä onnistuu myös kyseisen järjestelmän kautta (CMDB). Tähän liittyen järjestelmään tallennetulle verkkolaitteen laitekortille voidaan esimerkiksi yksilöivien laitetietojen lisäksi tallentaa myös laitteiden laskutustietoja. Yksilöiviä laitetietoja ovat esimerkiksi yksilöllinen laitenumero, sarjanumero, IP-osoite, isäntänimi ja laitteen verkkosovittimen MAC-osoite. Jokaisella laitekortilla on vähintään yksi yksilöivä laitetieto. Toiminnanohjausjärjestelmän avulla pystytään siis hallitsemaan laajasti IT-ympäristön toimintaa ja se onkin merkittävä osa toimeksiantajan työntekijöiden päivittäistä arkea. Seuraavissa kappaleissa käsitellään tarkemmin järjestelmän sisältämiä osa-alueita. (Efecte a)

2.1.1 Palvelunhallinta

IT-alan palveluhallintajärjestelmää kutsutaan ITSM-järjestelmäksi. Efecten palvelunhallinnan avulla esimerkiksi asiakkaiden ja yrityksen sisäiset palvelupyynnöt, tiketit sekä häiriöilmoitukset saadaan keskitetysti tallennettua ja vietyä eteenpäin asianosaisten ryhmien käsiteltäväksi. ITSM-järjestelmä pitää sisällään kaiken tarvittavan liittyen IT-palveluiden suunnitteluun, rakentamiseen, toteuttamiseen ja tukemiseen. Näin saadaan työstä tehokasta, hallittua ja töiden organisointi helpottuu. ITSM tekee työstä myös läpinäkyvää, sillä kaikki järjestelmään tallennettu tieto on saatavilla ja käytettävissä esimerkiksi raporttien luontia varten. Palvelunhallintajärjestelmä on käytännössä koko toiminnan keskiössä tahdittaen tekemistä päivittäin. (Efecte b)

2.1.2 Laskutus

Toiminnanohjausjärjestelmään kuuluu olennaisena osana laitteiden laskutusmahdollisuus. Efecte tuottaa toimeksiantajalle laskutusdatan esimerkiksi tuotannossa olevien laitteiden laitekortilla olevien laskutustietojen mukaisesti. Järjestelmän kautta voidaan laskuttaa myös erillislaskutettavia tuotteita tai muita palveluita. Laskutuskohde valitaan asiakkuuden mukaisesti, josta laskut kohdentuvat oikealle taholle laskutusjaksoittain automaattisesti. Järjestelmässä itsessään ei ole tarkastusta, joka vertaisi, onko tuotannossa oleville laitteille määritettynä laskutuskohde, mikä voi aiheuttaa puutteita laskutettavissa laitteissa suhteessa tuotannossa oleviin laitteisiin. Verkkolaitteiden osalta ylläpitolaskutuksen tietojen päivittämisestä laitekorteille huolehtivat yleisesti ottaen laitteen asentavat asiantuntijat. Inhimillisten virheiden määrä voi olla kuitenkin merkittävä yritykselle taloudellisesti, mikäli puutteita ei pystytä säännöllisesti tarkastamaan ja niihin reagoimaan.

2.1.3 Konfiguraationhallintatietokanta

Toimeksiantajan käytössä olevaan toiminnanohjausjärjestelmään, Efecteen, kuuluu olennaisena osana laite- ja ohjelmistokannan ylläpito CMDB:lla. Esimerkiksi verkkolaitteista voidaan tallentaa omille laitekorteilleen muun muassa laitteiden sarjanumerot, isäntänimet, IP-osoitteet, laitteiden sijainnit, status sekä laskutustiedot asiakkuuden mukaisesti. (Efecte c)

Järjestelmän avulla laitetiedot ovat hallitusti samassa paikassa ja esimerkiksi elinkaarisuunnittelu on mahdollista yksilöivien tietojen avulla. Näin laitekannan käyttöikä on suunnitelmallista ja tilanteen mukaan harkittua, mikä puolestaan voi vähentää laitekannan vanhentumisesta ja kunnosta johtuvia häiriöitä. CMDB:n avulla on mahdollista hallita myös erilaisia riippuvuuksia tietojen välillä, esimerkiksi sitä, mikä näyttö on asennettuna tiettyyn työasemaan tai mikä häiriöilmoitus liittyy mihinkin laitteeseen. (Efecte c)

Useat CMDB:een tallennetut tiedot ovat hyödynnettävissä myös muiden järjestelmien käyttöön. Käytännössä tämä tapahtuu hyödyntämällä Efectessä olevaa rajapintaa (Efecte Web API), jonka avulla järjestelmästä voidaan tuoda tai sieltä voidaan hakea tarvittavat tiedot. Rajapinta mahdollistaa lähestulkoon kaiken Efectessä olevan tiedon hakemisen ohjelmallisesti. Järjestelmästä haetuista tiedoista esimerkiksi verkkolaitteiden isäntänimeä ja IP-osoitetta voidaan hyödyntää muun muassa valvontajärjestelmien laitteiden yksilöintiin. Efectestä voidaan hakea rajapintaa hyödyntämällä esimerkiksi listaus verkkolaitteista, jota voidaan puolestaan verrata valvontaan lisättyihin laitteisiin. Laitteiden sijaintitietojen oikeellisuus puolestaan korostuu erityisesti häiriötilanteissa, kun asiantuntijan tulisi päästä kohteeseen korjaamaan kyseinen vika, ottamaan yhteyttä kohteessa olevaan henkilökuntaan tai vaihtamaan laite kokonaan uuteen.

Opinnäytteen aiheena olevan työkalun keskeinen ajatus on, että CMDB:n tietoja hyödyntämällä saadaan yhtenevää tietoa eri järjestelmien välille. Käytännössä tällä tarkoitetaan sitä, että muissa järjestelmissä olisi samat tiedot Efecten kanssa, eikä ristiriitoja ilmenisi päivittäisessä työssä. Muiden järjestelmien keräämät tarkemmat tiedot laitteista on mahdollista tuoda vastavuoroisesti Efecteen, joita voisivat olla esimerkiksi laitteen ohjelmistoversio tai saatavuuslukemat. Tämä lisää datan luotettavuutta ja kehittää toimintaa pidemmällä aikavälillä, kun laitteiden tiedot ovat kohdallaan eri järjestelmissä. Erityisen tärkeitä osa-alueita, joita työkalun avulla voidaan tarkastaa ovat DNS-tietueet (Domain Name System, nimipalvelun tietue), IP-osoite, laitteen status sekä laskutustiedot. Näiden tietojen avulla voidaan tarkastaa yhteneväisyyksiä muihin järjestelmiin ja ilmoittaa mahdollisista puutteista. Tämä vähentää myös inhimillisten virheiden ja muistin varassa olevien asioiden määrää.

2.2 Valvontajärjestelmät

Toimeksiantajalla on käytössään avoimeen lähdekoodiin pohjautuva verkonvalvontajärjestelmä LibreNMS. Valvonnassa olevilta laitteilta pystytään keräämään järjestelmän avulla huomattava määrä erilaista dataa. Järjestelmän avulla voidaan esimerkiksi valvoa laitteiden saatavuutta verkonvalvontapalvelimilta ja kerätä laitteilta erinäistä dataa. Järjestelmän keräämät tiedot voivat pitää sisällään esimerkiksi

- CPU ja linjakorttien käyttöasteita
- reititysprotokollien statustietoja
- sensoridataa, kuten laitteiden lämpötiloja
- laitteiden levyjen käyttöasteita
- kaistanleveyttä

- verkkoviiveitä.

ICMP on protokolla, jonka avulla saadaan lähetettyä erilaisia virheviestejä tai tilatietoja laitteilta. SNMP on puolestaan protokolla, jonka avulla voidaan kysyä laitteen tilaa tai laite voi hälyttää. Järjestelmässä hyödynnetään ICMP-protokollaa verkkolaitteiden valvontakyselyssä ja SNMP-protokollaa muiden tietojen hakemisessa. Tyypillisesti tärkeimmät tiedot verkkoyhteyksien valvonnan suhteen ovat verkkolaitteille konfiguroitujen OSPF-, ISIS- ja BGP- naapuruuksien tiedot, joiden avulla saadaan muun muassa tiedoksi kyseiseltä reititysprotokollalta, onko kyseinen yhteysväli ylhäällä. Tietojen pohjalta järjestelmään on mahdollista luoda erinäisiä hälytyssääntöjä omien mieltymysten mukaisesti. Järjestelmään on myös mahdollista tehdä dynaamisesti muodostettuja laiteryhmiä, joihin asetetaan laitteita automaattisesti tiettyjen ryhmittelysääntöjen mukaan. Ryhmittelysääntöjä voidaan tehdä tarpeen mukaan, mikä helpottaa laitetietojen analysoinnissa ja hälytysten ryhmittelyssä esimerkiksi kiireellisyyden mukaan. (LibreNMS) (LibreNMS Docs)

Tässäkin tapauksessa korostuu tietojen oikeellisuus, jotta laitteet osuisivat oikeisiin hälytyssääntöihin ja data olisi luotettavaa. Kehitetyn työkalun avulla järjestelmässä olevaa dataa pystytään vertaamaan Efectessä olevaan dataan ja ilmoittamaan mahdollisista epäkohdista. Järjestelmien väliset tarkastukset ovat hyödyllisiä datan luotettavuuden kannalta. Jokaisessa järjestelmässä tulisi olla yhteneväiset tiedot, jotta esimerkiksi valvontajärjestelmän hälytyssäännöt kohdistuvat laitteille oikein. Näin pystytään kohdistamaan resursseja oikein esimerkiksi tilanteissa, joissa laitteet eivät vastaa enää verkosta.

Toimeksiantajan käytössä on myös keskitetty lokienhallintajärjestelmä Graylog, johon kerätään laitteilta saapuvaa syslog-tyyppistä lokitietoa. Lokitiedot pitävät sisällään eri järjestelmäviestejä, joihin voidaan tarvittaessa reagoida esimerkiksi viestin riskikertoimen mukaan. Järjestelmä on itsenäinen eikä liity muihin esiteltyihin järjestelmiin millään tavalla. Järjestelmän käyttöliittymässä pystytään esittämään laitteilta kerättyjä lokitietoja halutulla tavalla, muun muassa omien näkymien ja räätälöityjen hakuehtojen perusteella. Lokitiedoista voidaan ajaa myös hälytyssääntöjä sopivien hakuehtojen perusteella. Järjestelmästä voidaan tehdä hakuja esimerkiksi laitteen nimen tai IP-osoitteen mukaisesti, joten myös siksi olisi suotavaa, että tiedot olisivat täytettyinä oikein.

2.3 Varmuuskopiointijärjestelmä

Toimeksiantajan eräänä verkkolaitteiden konfiguraatioiden varmuuskopiointijärjestelmänä on Oxidized. Se on avoimeen lähdekoodiin perustuva järjestelmä, josta on mahdollista hakea nopealla aikataululla verkkolaitteen konfiguraatio. Järjestelmä mahdollistaa konfiguraation tuomisen uuteen verkkolaitteeseen rikkoutuneen tilalle samaa pohjakonfiguraatiota hyödyntäen tai sitä mukaillen. Järjestelmän avulla voidaan ottaa yhteys verkkolaitteelle ja tallentaa sen koko konfiguraatio joko tiedostoon tai Git-versionhallintaan tietyin väliajoin.

Toimeksiantajalla kyseinen järjestelmä hyödyntää LibreNMS-järjestelmän tarjoamaa rajapintaa ja pyytää sen kautta listan valvontaan lisätyistä laitteista ja laiteryhmistä laitteen lisäämisen ja järjestelmän käynnistymisen yhteydessä. Käytännössä laitteen varmuuskopioinnin käynnistyminen vaatii laitteelle oikeat yhteysasetukset Oxidized-järjestelmään kyseisen laiteryhmän osalta, isäntänimen sekä laitteen lisäämisen LibreNMS-järjestelmään. Laitteen löytyessä Oxidizedin laitelistalta järjestelmä hakee kysei-

sen laiteryhmän yhteysasetuksia hyödyntäen laitteen konfiguraation 86400 sekunnin (vuorokausi) välein ja tallentaa sen. Jokaisen laitteen uusimman konfiguraation pystyy hakemaan käyttöliittymän avulla laitteen IP-osoitteella. Tästä järjestelmästä olisi hyvä tarkastaa työkalun automaatiolla löytyykö esimerkiksi kaikille verkonvalvontaan lisätyille laitteille varmuuskopio sekä käydä järjestelmästä ne laitteet läpi, jotka eivät enää vastaa verkosta.

2.4 Langattoman verkon ohjaimet

Langattomia tukiasemia voidaan keskitetysti hallita langattoman verkon ohjainten (WLC) avulla. Ohjaimet keräävät tukiasemilta saapuvaa dataa ja sen perusteella voidaan seurata esimerkiksi laitteissa kiinni olevien päätelaitteiden määrää, analysoida verkon käyttöä muun muassa langattoman verkon nimen (SSID) tai käytetyn radiotaajuuden perusteella sekä tietenkin sitä, onko tukiasema pysynyt ylipäätään verkossa. Ohjainten avulla tehdään myös tukiasemien provisiointia eli määritellään laite käyttökuntoon.

Ohjaimet tallentavat tukiasemilta myös rekisteröidyn DNS-nimen, hallintaverkon IP-osoitteen sekä sarjanumeron. Myös muita tietoja ohjaimille rekisteröityneistä tukiasemista on mahdollista hakea. Efectestä voidaan puolestaan ladata lista käytössä olevista tukiasemista ja tarkastaa ovatko ne lisättyinä valvontaan. Lisäksi voidaan tehdä tarkastuksia, joissa verrataan WLC-ohjaimilta, LibreNMS:stä sekä Efectestä kerättyjä tietoja ristiin. Näiden tietojen avulla pystytään tekemään kattavia tarkastuksia työkalun avulla ja välttämään tulevaisuudessa ilmaantuvia ristiriitoja tiedoissa.

3 SUUNNITTELU

Kehitettävän työkalun suunnittelu lähti liikkeelle toimeksiantajan tarpeiden perusteella. Koska kyseessä oli jo toiminnassa ollut, mutta jatkokehitystä tarvitseva ja käytännössä kokonaan koodipohjan uudistusta kaipaava työkalu, oli tietyt reunaehdot havaittavissa jo alkuvaiheen määrittelyssä. Näitä reunaehtoja suunnittelussa olivat pyrkimys työkalun helppoon ylläpidettävyyteen, työkalun laajennettavuus tarpeen mukaan sekä tietenkin ongelman ratkaisu eli Efecte-tiedon käyttäminen laitetietojen tarkastuksessa eri järjestelmien välillä sekä puutteiden havaitseminen laitetiedoissa. Tämän lisäksi suunnittelua ohjasi vahvasti tarve työkalun kehittämiseen myös tulevaisuudessa ja sen huomioiminen uutta koodipohjaa rakennettaessa. Seuraavissa kappaleissa käydään tarkemmin läpi työkalulle asetettuja vaatimuksia, koodauskielen ja tarvittavien ohjelmointikirjastojen valintaa ja sitä, miten tietoja voidaan yhdistellä työkalun käyttöön.

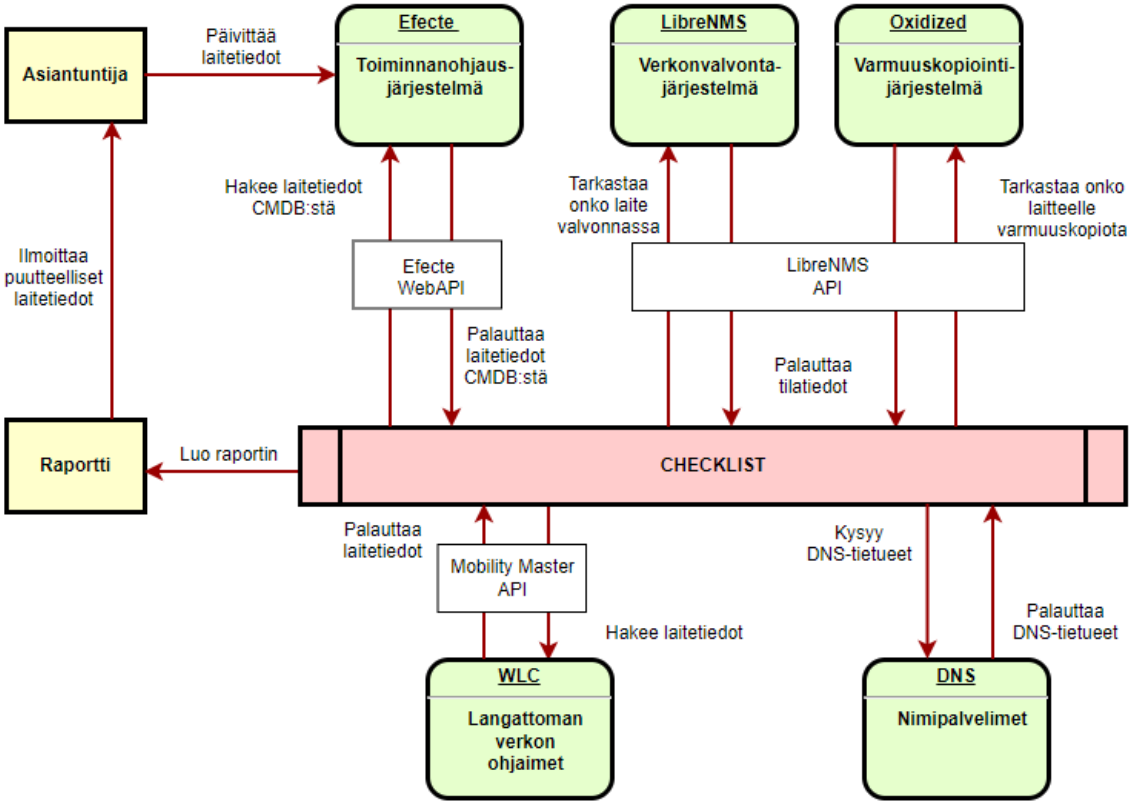
3.1 Nykyinen työkalu ja haasteet

Nykyinen työkalu on toteutettu Istekin tietoliikennepalveluiden toimesta useampi vuosi sitten. Toiminnanohjausjärjestelmän muutoksen seurauksena työkalua pyrittiin muiden työtehtävien lomassa kehittämään uutta järjestelmää hyödyntäen, mutta toiminnassa on edelleen havaittu ongelmia ja on päädytty kehittämään kokonaan uusi ratkaisu. Tietojen haku vanhassa työkalussa oli toteutettu niin, että laitetiedot haettiin erilliseen tietokantaan Efectestä ja vanha raportointityökalu haki tiedot puolestaan tästä tietokannasta. Toiminnanohjausjärjestelmässä on kuitenkin olemassa kätevä rajapinta (Efecte Web API) tietojen hakemiseen ja viemiseen, mikä haluttiin ottaa uudella työkalulla käyttöön suoraan ilman välissä olevaa tietokantaa. Tähän ajatusmalliin pohjautui koko uuden työkalun kehittäminen sekä jatkokehitys tämän jälkeen. Suunnittelua ohjasi myös pyrkimys siihen, että tiedot olisivat eri järjestelmien välillä yhtenäisiä

Työkaluun liittyvät järjestelmät ovat melko laajoja, eri valmistajien ja eri tehtäviä tekeviä kokonaisuuksia. Tämän takia on hankalaa löytää markkinoilta kyseiseen tehtävään soveltuvaa valmista työkalua, joka pystyisi vastaamaan kaikkiin tarpeisiin. Työkalun tarkoitus on pystyä hakemaan tietoa Efectestä ja sen jälkeen raportoimaan tarkastusten puutteet tarkastusraportille. Myöhemmässä vaiheessa voidaan viedä automaatiota eteenpäin niin, että kyseiset laitetiedot päivitettäisiin soveltuvin osin käytössä oleviin järjestelmiin. Työkalun tavoitteena on tuottaa yhteneväistä tietoa eri järjestelmien välillä sekä helpottaa asiantuntijan työtä eri järjestelmistä tietoa haettaessa. Vaikka omalla kehityksellä tehty työkalu vaatii työaikaa ja panostusta, saadaan tässä tapauksessa kehitettyä tarkalleen omiin tarpeisiin sopiva väline. Toimeksiantajan toimintaympäristö on myös jatkuvasti muuttuva ja työkalun avulla on pystyttävä tekemään monipuolisia tarkastuksia eri ehtojen perusteella. Omalla kehityksellä voidaan varmistua siitä, että työkalua on mahdollista kehittää eteenpäin ja siihen voidaan liittää mahdollisesti myös muita tuotantoon tulevia järjestelmiä.

3.2 Järjestelmistä tarkastettavat tiedot

Luvussa 2 käsiteltiin työkaluun liittyviä järjestelmiä. Seuraavassa kuvassa (kuva 1) on esitettynä näiden järjestelmien yhteys kehitettävään työkaluun ja siihen mitä tietoja järjestelmistä halutaan tarkastaa. Kutakin kokonaisuutta avataan tarkemmin käytäessä kaaviota läpi järjestelmä kerrallaan.



Kuva 1. Tietovirtakaavio

Ensimmäinen vaihe kaaviossa on tietojen päivittäminen Efecten laitekortille. Tämän tehtävän suorittaa yleensä laiteasennuksen yhteydessä kyseisen laitteen asentava tietoliikenneasiantuntija. Asiantuntijan on päivitettävä laitekortille seuraavat tiedot:

- laitenimi (verkkolaitteiden yrityksen sisäisen nimeämispolitiikan mukaisesti)
- sarjanumero
- laitteen sijainti
- laitteen status
- laskutustiedot
- isäntänimi
- aliverkko ja IP-osoite
- MAC-osoite
- asentajan nimi.

Kehitettävä raportointityökalu ottaa Efecte WebAPI:n kautta yhteyden CMDB:een ja palauttaa haetun laitekortin tiedot työkalun käyttöön. Efecte WebAPI:in tehtävät kyselyt toteutetaan Efecten omalla kyselykielellä Efecte EQL (Efecte Query Language), josta laitetiedot saadaan kätevästi ohjelman käyt-

töön. Kyselyssä voidaan hyödyntää erilaisia suodattimia tarpeen mukaan ja hakea erilaisia tietoja järjestelmästä. Laitetiedot kerätään puolestaan erilliselle laitelistalle, jota käytetään tarkastusten pohjana laiteryhmittäin.

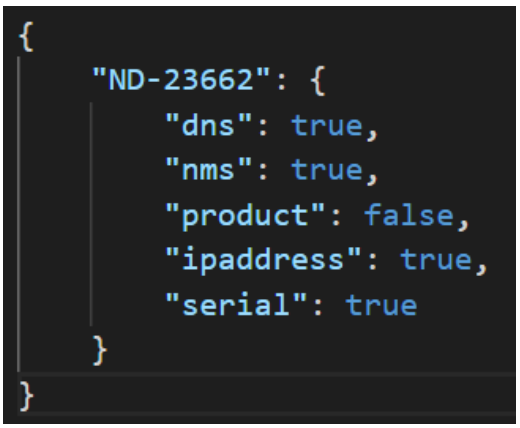
LibreNMS API tarjoaa raportointityökalulle kätevän rajapinnan, jonka avulla voidaan tarkastaa, onko verkkolaite lisättynä valvottavien laitteiden joukkoon. Järjestelmä palauttaa kyselystä statuksen laitteen tilan mukaisesti. Tilaa voidaan hyödyntää työkalussa merkitsemällä palautuneen arvon perusteella tarkastusraportille sen mukainen lopputulos. LibreNMS-järjestelmä rekisteröi sinne lisätyiltä laitteilta myös niiden laitetietoja kuten sarjanumeroita. Esimerkiksi tätä tietoa voidaan kehitettävässä työkalussa verrata Efectestä haettuun tietoon ja raportoida tarvittaessa poikkeamista tarkastusraportille.

Oxidized järjestelmästä voidaan työkalun avulla tarkastaa, löytyykö laitekortille tallennetulle IP-osoitteelle kyseisen laitteen konfiguraatiosta varmuuskopiota. Järjestelmän tarkastus tehdään LibreNMS-järjestelmän rajapintaa hyödyntäen, johon Oxidized on liitettynä. Myös varmuuskopioinnin onnistuminen voitaisiin tarkastaa samassa yhteydessä. Tämä tarkoittaa sitä, että varmistetaan järjestelmän saaneen tallennettua laitteen uusin konfiguraatio lähiaikoina, jolloin voidaan luottaa konfiguraation ajantasaisuuteen. Näin ei pääse aiheutumaan tilannetta, jossa varmuuskopio on olemassa, mutta se on esimerkiksi puoli vuotta vanha, ja sen aikana ajantasainen konfiguraatio olisi ehtinyt muuttumaan useamman kerran.

Jokaisella laitteella tulisi olla Efectessä merkittynä laitenimi, joka tulisi löytyä myös nimipalvelimilta. Raportointityökalun avulla tarkastetaan, löytyykö kyseinen laitenimi DNS-tietueista ja palautetaan tieto työkalulle raportoitavaksi, mikäli näin ei ole. Nimipalvelimilla laitteiden nimet tallennetaan FQDN-niminä, jolloin tässä tapauksessa Efectestä saadun laitenimen perään liitetään vielä domain-nimi. Esimerkiksi ”devicename.example.fi”. Työkalu ottaa tämän huomioon tehdessään kyselyä palvelimelle lisäämällä domain-nimen erilliseltä listalta ja käy ne järjestyksessä läpi, kunnes osuma löytyy.

Langattoman verkon ohjainten avulla langattomilta tukiasemilta voidaan hakea tarvittavia tietoja. Tietoja voivat olla esimerkiksi laitenimi, sarjanumero, langattoman verkon profiili ja tieto siitä, kuinka kauan laite on ollut verkossa. Ohjaimilta saadaan myös merkittävä määrä muuta tietoa sinne rekisteröidyiltä tukiasemilta, mitä voidaan hyödyntää tarkastuksissa. DNS-tietueista tehdään tässä vaiheessa myös tarkastus, että tukiasema on rekisteröity oikealla nimellä nimipalvelimelle. Tämän lisäksi työkalun avulla on hyvä tarkastaa, että tukiasemilla on Efectessä sama laitenimi kuin millä ne ovat rekisteröityneenä langattoman verkon ohjaimille.

Jokainen epäonnistunut vaihe raportoidaan laitekohtaisesti raportointityökalussa suoritettavan ohjelman lopputuloksena syntyvälle JSON-muotoiselle raportille. JSON on standardimuotoinen, hyvin kevyt ja helppo tiedostoformaatti, johon tallennettu data on loogisessa järjestyksessä ja ihmisen luettavissa (Kuva 2). Tallennettua dataa saadaan myös muokattua haluttuun esitysmuotoon tarpeen mukaan. (Python.org a)



Kuva 2. Kuvaleike raportointityökalusta, esimerkki JSON-tiedoston rakenteesta

Työkalu suunniteltiin sillä periaatteella, että laitetiedoista saataisiin karsittua pois inhimillisten virheiden aiheuttamat puutteet ja epäselvyydet sekä niihin pystyttäisiin reagoimaan nykyistä aiemmin. Työkalun päätoiminnallisuuksien määrittämisen jälkeen valittiin ohjelmalle sopiva ohjelmointiympäristö.

3.3 Ohjelmointiympäristö

Työkalun ohjelmointikieleksi valikoitui jo varhaisessa vaiheessa Python 3. Python on varsinaisesti skriptauskieli, jonka pääasiallinen tarkoitus on yksinkertaisesti sanottuna saada jokin tehtävä suoritetuksi. Muita skriptauskieliä ovat muun muassa Bash ja Perl, jotka eroavat kumpikin Pythonista muun muassa syntaksin ja saatavilla olevien ominaisuuksien puolesta. Omaan silmään Python on selkeästi yksinkertaisempi ja tarjoaa enemmän ominaisuuksia suorittaa monipuolisiakin tehtäviä tehokkaasti. (Gift & Jones, 2009, s. 1)

Pythonin valinta tämän työkalun toteutukseen oli selkeä myös toimeksiantajan puolelta. Työntekijöiltä löytyy jo valmiiksi osaamista sekä projekteja, joissa Pythonia on käytetty ja havaittu sopivaksi ryhmän tarpeisiin. Python on myös suosittu ohjelmointikieli automaatioiden rakentamisessa ja verkkoympäristöjen hallintatyökalujen toteuttamisessa. Sen avulla voidaan toteuttaa useita verkkoteknisiä ratkaisuita, joita toimeksiantajalla onkin jo toteutuksessa. Näiden syiden vuoksi on mielekästä käyttää samaa ohjelmointikieltä ja pyrkiä yhtenäistämään projekteja myös jatkossa. Tämä helpottaa koodin ylläpitoa sekä parantaa tehokkuutta, kun toimintatavat ovat selkeät koko ryhmän sisällä.

Ohjelmointikielenä Python tarjoaa tehokkaan ympäristön erityisesti nopeaan kehitystyöhön. Python sisältää valmiina olevia toiminnallisuuksia ja saatavilla on tarpeen mukaan avoimen lähdekoodin kirjastoja. Python on alustariippumaton eikä vaadi toimiakseen merkittäviä asennustöitä, se sisältyy esimerkiksi automaattisesti useimpiin Unix-pohjaisiin järjestelmiin ja muihinkin se on tarjolla asennuspakettina. (Gift & Jones, 2009, ss. 1, 5)

Pythonilla kirjoitettu koodi mielletään usein aloittelijankin silmään helppolukuiseksi ja helpoksi oppia. Tämä tietenkin helpottaa myös jatkokehitystä, kun valmiista koodista saadaan jo nopealla perehtymisellä visio sen toiminnallisuuksista. Tämä on edesauttanut Pythonin suosiota ohjelmoijien keskuudessa opittavuuden ja tehokkaan kehittämisprosessin ansiosta. Pythonin ehdoton vahvuus on myös vahva kehittäjien yhteisö, jotka kehittävät toiminnallisuuksia edelleen ja toimivat tukena erilaisilla fooru-

meilla. Yhteisön avulla kehittäjien keskuudessa saadaan ylläpidettyä niin koodipohjaa kuin ohjelmoijien tietoisuutta parhaista tavoista tehdä jokin tehtävä Python-ohjelmointikieltä hyödyntäen. (Gift & Jones, 2009, ss. 1-2, 5)

Python on myös erittäin joustava ohjelmointikieli ja se on suunniteltu käytettäväksi useassa eri käyttötarkoituksessa. Joku voi haluta toteuttaa sen avulla erilaisia toiminnallisia verkkosivuja, toinen teolliseen käyttöön soveltuvia robotteja ja joku puolestaan voi haluta käyttää sitä pelin luomiseen. Python-kehittäjien yhteisön ylläpitämä ohjelmointikirjasto (Python Package Index, PyPi) tarjoaakin monipuolisia avoimen lähdekoodin lisäosia kehitysprojekteihin vaihtelevien tarpeiden mukaan. PyPi:n verkkosivuilta voi hakea erilaisia projekteja tiettyjen hakusanojen, kuten projektin nimen, käyttöjärjestelmän, ohjelmointikielen sekä halutun ohjelmistokehyksen mukaan. Näin voidaan löytää omaan projektiin mahdollisimman sopivat lisäosat ja säästää samalla huomattava määrä aikaa kehitystyössä, kun kehittäjä voi tehdä vain latauksen valmiista kirjastosta ja sisällyttää sen omaan projektiinsa. (Python Packaging Authority) (Python Package Index)

Python pitää usein valmiina sisällään ”pip” -komentorivityökalun, jolla kyseisiä lisäosia voidaan asentaa. Mikäli työkalu puuttuu, se voidaan asentaa komentoriviltä komennolla ”ensurepip” tai vaihtoehtoisesti ”get-pip.py”. Lataukset onnistuvat tämän jälkeen käyttöjärjestelmän mukaan seuraavasti:

```
Unix/MacOS:  python -m pip install JokuPaketti          # viimeisin versio
             python -m pip install JokuPaketti==1.0.4   # tietty versio
             python -m pip install 'JokuPaketti>=1.0.4' # minimi versio
Windows:     Muuten sama komento, mutta ”python” –> ”py”
```

Yllä olevissa esimerkeissä ”JokuPaketti” kohtaan syötetään haluttu lisäosan nimi. Tämän jälkeen työkalu hakee lisäosan PyPi:stä ja asentaa sen työasemalle. Nyt haluttu lisäosa on valmis lisättäväksi omaan projektiin. (Pip documentation v22.0.4)

Yksi Pythonin eduista on myös sen sisältämä dictionary-tietorakenne. Käytännössä se koostuu avain-arvopareista, joiden hyödyntäminen suurissa tietojoukoissa on tehokasta ja loogista. Dictionary luodaan yleensä kaarisulkujen avulla ja sisällön muodostaminen voidaan tehdä muutamallakin eri tavalla. Ensimmäinen ja selkein keino on lisätä arvot listauksena pilkulla eroteltuna.

Tässä tilanteessa tietorakenne luodaan esimerkiksi näin:

{2010: ’nelli’, 0909: ’ada’, 0107: ’johanna’} tai {’nelli’: 2010, ’ada’: 0909, ’johanna’: 0107}.

- Toinen tapa (luettelon ymmärrys, eng.”list comprehension”):

  {}, {x: x ** 2 for x in range(10)}.

- Kolmas tapa (tyypitetty):

  tyhjä dictionary → dict(),

  dictionary, joka sisältää arvoja → dict([('nelli', 2010), ('ada', 0909), ('johanna', 0107)]),

  dictionary, joka sisältää arvoja → dict(nelli=2010, ada=0909, johanna=0107).

Kuten havaitaan, dictionaryyn voidaan syöttää tietoja usealla eri menetelmällä, mutta silti sen palauttama sisältö on aina samanmuotoista ja sisältää määritellyt avain-arvoparit. Tietorakenteen luomisen jälkeen sen sisältö voidaan yksinkertaisesti tulostaa avain-arvopareina tai tiedoista voidaan hakea tehokkaasti tarpeen mukaan eri asioita perustuen avain-arvopariin. (Python.org b)

Tässä opinnäytetyössä käytettiin laitetietojen käsittelyssä dictionary-tietorakennetta ja hyödynnettiin sen sisältämiä operaatioita, jolloin laajasta laitemassasta saatiin haettua tehokkaasti laitteiden tiedot raportoinnin tueksi. Myös raportti saatiin luotua samaisen tietorakenteen avulla selkeäksi ja toimivaksi kokonaisuudeksi.

3.4 Haettujen tietojen hyödyntäminen

Kuvassa 1 (s.12) kerrotaan työkalun ottavan yhteyden ensin toiminnanohjausjärjestelmään ja hakevan laitteen tiedot työkalun käytettäväksi raportin luomista varten. Käytännössä työkalu tekee ensin EfecteWebAPI:n kautta kyselyn CMDB:hen. Kyselyssä määritetään hakuehdoiksi laitetyyppi (verkkolaite) ja status (tuotannossa), joka palauttaa listan hakuehdot täyttävistä laitteista. Laitteet ovat yksilöitynä listalle laitenumeroittain. Tarvittaessa listoja voidaan tehdä eri tarpeisiin eri hakuehdoilla, mikä helpottaa työkalun jatkokehitystä. Esimerkiksi langattoman verkon laitteet haetaan työkalussa omalle listalleen.

Jokainen listalle lisätty verkkolaite käydään tämän jälkeen silmukassa läpi. Tässä vaiheessa laitteista tarkastetaan puutteelliset CMDB tiedot. Työkalu ei ota kantaa siihen, mitä laitetietoihin on täytetty, vaan keskittyy siihen, että kentissä on jokin tieto täytettynä. Tyhjät kentät laitetiedoista palautuvat listalle ”None”-tyyppisinä, jotka voidaan raportoida JSON-muotoiselle tarkastusraportille. Raportilla jokainen laite yksilöidään ensin laitenumeron perusteella. Tämän jälkeen jokaisen laitteen yhteyteen luodaan yksilöivä nimi jokaiselle tarkastuskohteelle ja tarkastuksen tulokseksi joko ”true/false” (kts. s.13, kuva 2). CMDB:ssä olevien puutteiden läpikäynnin jälkeen työkalu ottaa yhteyden DNS nimipalvelimille ja tarkastaa löytyykö kyseiselle laitteelle merkintää sekä raportoi mahdollisesta puutteesta.

Seuraavaksi työkalu käyttää verkkolaitteen isäntänimeä ja ottaa LibreNMS API:in yhteyden tarkastaakseen löytyykö laite valvonnasta. Valvontajärjestelmä LibreNMS käyttää DNS-nimeä tarkastaakseen vastaako järjestelmään lisätty laite verkosta, minkä vuoksi on tärkeää, että laitteelle löytyy oikea merkintä. Mikäli laite löytyy, saadaan LibreNMS-järjestelmästä haettua myös muita tietoja käsiteltävänä olevasta laitteesta. Tässä opinnäytetyössä verrattiin myös, onko CMDB:stä laitekortilta haettu sarjanumero sama kuin valvontajärjestelmän laitteelta saama sarjanumero. Molemmista tilanteista raportoidaan tarvittaessa poikkeamat tarkastusraportille.

Kuvassa 1 voidaan todeta työkalun ottavan yhteyden myös Oxidized-varmuuskopiointijärjestelmään LibreNMS API:n kautta. Kyseinen rajapinta tarjoaa yksinkertaisen liittymän varmuuskopiointijärjestelmään, josta voidaan suorittaa tilakysely käsittelyssä olevalle laitteelle. Saatu tilatieto palautetaan tarkastusraportille.

Langattoman verkon laitteisiin saadaan otettua yhteys työkalulla laitevalmistajan API-rajapintaa käyttäen. Työkalu hakee yksittäisen laitteen tiedot listalle, joiden avulla voidaan tehdä vertailu Efectestä saatuihin tietoihin. Laitetietoja käytetään myös nimipalvelimelle suoritettavan tarkastuksen yhteydessä sekä sarjanumeron vertailuun Efecte-järjestelmään merkityn tiedon pohjalta. Käytännössä työkalulla on käytössään kaikki yksittäiselle laitteelle haetut tiedot rajapinnasta ja niitä voidaan hyödyntää monipuolisesti myös työkalun jatkokehityksessä.

4 TOTEUTUS

4.1 Kehitysympäristön käyttöönotto

Kehitysympäristön tekstieditoriksi otettiin käyttöön Visual Studio Code. Se tarjosi tähän tarkoitukseen riittävän kattavat työkalut eikä vaatinut erillistä lisenssiä työasemalle, kuten esimerkiksi Microsoftin saman tuoteperheen Visual Studio. VS Code on kevyempi ja suppeammilla ominaisuuksilla varustettu ohjelmoijille tarkoitettu tekstieditori, mutta se on myös laajennettavissa lisäosien avulla tarpeen mukaan. Tämän työkalun kehitystä varten asennettiin VS Codeen seuraavat lisäosat:

- Python, koodikielen tunnistamiseen ja ajamiseen liittyen

- Git Graph, versionhallinnan näkymä suoraan tekstieditorissa

- Remote SSH, etäyhteyden saamiseksi testipalvelimeen editorin kautta.

Lisäksi työasemalle asennettiin erillisenä ohjelmana Git, joka tarjoaa erinomaiset työkalut versionhallintaa varten niin komentorivillä kuin suoraan tekstieditorin lisäosaa hyödyntämällä. Git:iin on mahdollista asentaa myös kolmannen osapuolen tarjoama käyttöliittymä työpöytäohjelmana. Ohjelman viimeisimmän version ilman käyttöliittymää tai sen kanssa sai ladattua suoraan Git:n verkkosivuilta, josta asennus tapahtui suoraviivaisesti seuraamalla opastusta. Tässä työssä erilliselle käyttöliittymälle ei nähty tarvetta, joten latasin komentoriviä hyödyntävän Git:n.

Seuraavaksi Visual Studio Coden kautta otettiin etäyhteys testipalvelimen kansioon, johon kehitettävän työkalun tiedostot tullaan sijoittamaan. Yhteys avattiin asennettua Remote SSH lisäosaa hyödyntämällä. Testipalvelimella on tehokasta tehdä kehitystyötä, sillä siellä voidaan asentaa tarvittavia ohjelmistokirjastoja sekä tehdä kyselyitä toiminnanohjausjärjestelmään ja muihin työkaluun liittyviin järjestelmiin. Tämän lisäksi palvelimelta on avattuna tarvittavat kohdejärjestelmän portit ja liikenne eri järjestelmien välillä onnistuu olemassa olevien palomuuriavausten perusteella. Tämä edesauttaa sitä, että kehitysaikana saadaan testattua toiminnallisuuksia nopealla aikataululla ja kehitystyö saadaan nopeasti liikkeelle.

4.2 Toiminnallisuudet

Seuraavissa kappaleissa käydään läpi ohjelmakoodissa olevia toiminnallisuuksia, sekä vaiheittain sitä, miten eri toiminnallisuudet käsittelevät järjestelmistä haettuja tietoja keskenään. Toiminnallisuuksien suunnittelun yhteydessä voitiin havaita järjestelmiin toteutettavien kyselyiden olevan itsenäisiä kokonaisuuksia, minkä vuoksi jokaiseen järjestelmään toteutettu kysely on erillinen funktio, jotka käydään läpi laitekohtaisessa järjestyksessä. Kerättyjen tietojen avulla tarkastusraportille voidaan muodostaa laitekohtaiset tiedot kyselyiden tuloksista ja nostaa eroavaisuudet selkeästi esille. Eri järjestelmissä olevat laitetiedot on tärkeää pitää yhteneväisinä keskenään, mikä voi joskus olla haastavaa laajan ympäristön ylläpidossa. Tarkastusraportin hyöty tulee esille etenkin suuria laitemassoja tarkasteltaessa, mikä parhaassa tapauksessa voi säästää merkittävän määrän asiantuntijan työaikaa tiettyä ristiriitaa selvittäessä.

4.2.1 Tietojen haku Efectestä

Ensimmäisenä työkalu ottaa yhteyden EfecteWebAPI:in. Yhteydenottoa varten työkaluun on määritetty yksilölliset API-tunnukset erilliseen ”config.cfg”-tiedostoon, josta työkalu lukee tarvittavat kirjautumistiedot Pythonin ”configparser”-luokkakirjastoa hyödyntämällä. Kyseinen lisäosa selkeyttää varsinaisessa ohjelmakoodissa tarvittavaa sisältöä ja erillisen kirjautumistietoja ylläpitävän tiedoston käyttäminen helpottaa jatkokehitystä esimerkiksi silloin, kun samoja kirjautumistietoja käytetään useassa eri ohjelmakoodissa. Näin kehittäjän tarvitsee tehdä mahdolliset muutokset ainoastaan yhteen tiedostoon.

Yhteyden saatuaan työkalu hakee EQL:a käyttäen Efectestä kaikki verkkolaitteet, joiden statukseksi on määritetty ”Tuotannossa”. Käytännössä funktio lähettää GET-pyynnön mukana EQL-syntaksin mukaisen pyynnön, joka palauttaa listan ehtoihin sopivista laitteista laitenumeroittain. Kyseistä listaa käydään seuraavassa vaiheessa silmukalla läpi laitenumeroittain ja haetaan yksittäisen laitteen tiedot.

Yksittäisen laitteen tiedot haetaan CMDB:stä niin, että ne pitävät sisällään kaiken laitekorttiin kuuluvan tiedon XML-muodossa. Näin laitteilta saadaan tarvittaessa hyödynnettyä tässä vaiheessa kerättyä tietoa tehokkaasti myös myöhemmin, eikä erillisiä hakuja tarvita. Työkaluun on tulossa tulevaisuudessa myös jatkokehitystä, mikä on hyvä ottaa huomioon funktioita suunnitellessa jo alkuvaiheessa. Nyt tietojen hakeminen laitetiedoista ei ole tiettyihin kenttiin sidonnainen, vaan ohjelmassa voidaan hyödyntää kaikkea laitekortilta saatavaa tietoa tarpeen mukaan, mikä helpottaa erilaisten tarkastusten tekemistä.

Tässä vaiheessa haetut tiedot halutaan muuttaa helpommin käsiteltävään muotoon, joka tehdään käyttämällä Pythonin dictionary-muotoista tietorakennetta (kuva 3). Dictionaryn etuna on se, että myöhemmässä vaiheessa tietoja voidaan käsitellä avain-arvopareina, jolloin käsittelystä tulee yksinkertaisempaa ja tarkastuksissa voidaan viitata suoraan avaimeen, jolla on tietty arvo. Dictionaryn avaimiksi talletetaan laitelistalta Efecte ID, laitteen nimi, laitteen malli, laitteen status, sarjanumero, IP-osoite, MAC-osoite sekä laskutuskohde. Jokaiselle avaimelle muodostuu Efectestä haetun tiedon perusteella jokin arvo, jota voidaan hyödyntää seuraavissa toiminnallisuuksien vaiheissa.

```
# Laitetaan dictiin datat xml aineistosta
data = {
    "Efecte_ID": efecte_id,
    "Name": name,
    "Model": model,
    "Status": status,
    "Serial": serial,
    "IP_address": ip_address,
    "MAC_address": mac_address,
    "Product": product,
}
```

Kuva 3. Kuvaleike raportointityökalusta, dictionary tietorakenne

4.2.2 Tietojen tarkastukset haetuilla laitetiedoilla

Tarkastukset aloitetaan käynnistämällä tarkastusfunktio, jonka jälkeen tarkastuksista voidaan luoda tarkastusraportti. Jokaisen tarkastuskohteen yhteydessä käynnistetään yksilöllinen funktio, joka tekee kyseiselle kohteelle tarvittavat tarkastukset omassa funktiossaan ja palauttaa arvon tarkastusraportille. Tällä toimintatavalla saadaan mahdollisessa jatkokehityksessä hyödynnettyä olemassa olevia funktiota ja selkeytettyä samalla myös koodin toimintaa.

Efecte-tarkastukset suoritetaan hyödyntämällä laitenumeroa. Efecte-tiedoista tarkastetaan laitteen nimi, malli, sarjanumero, IP-osoite, MAC-osoite ja laskutustiedot. Funktiot toimivat kaikki samalla periaatteella hakien ensin yksilölliset laitetiedot laitenumeron perusteella. Tiedot palautuvat dictionary-tietorakenteena, jolloin käytettävissä olevaa avain-arvoparia voidaan hyödyntää. Käytännössä tämä tarkoittaa sitä, että muuttujaan voidaan tallentaa avaimen palauttama arvo. Mikäli arvo on tyhjä, voidaan tässä tapauksessa todeta, että tietoa ei ole täytetty asianmukaisesti.

Kuvassa 4 voidaan havaita aiemmassa kappaleessa esitelty toimintatapa laskutustuotteen tarkastamisen osalta. Muuttujaan ”product” määritetään CMDB:stä haetuista laitteen tiedoista avaimelle ”Product” määritellyn arvon. Funktio palauttaa tarkastusraportille arvon ”True”, mikäli muuttujan arvo on muuta kuin ”None” eli tyhjä.

```
# Hakee laitteen efecte-ID:n ja tarkastaa laitekortilta onko laskutustietoja
def checkProduct(efecte_id):
    try:
        device = device_data.get(efecte_id)
        # print(device)
        product = device.get("Product")

        if product != None:
            # Laite on laskutuksessa
            return True
        else:
            # Laite ei ole laskutuksessa
            return False
    except Exception as e:
        print(str(e))
        return False
```

Kuva 4. Kuvaleike raportointityökalusta, esimerkki avain-arvoparin hyödyntämisestä

Nimipalvelusta laitteen tarkastaminen tapahtuu ottamalla ensin yhteys nimipalvelimiin, joiden osoitteet ovat määriteltyinä erillisessä config-tiedostossa. Tässä tarkastuksessa on otettava huomioon, että CMDB:sta haettu laitenimi ei sisällä domain-tietoa. Siksi haetun laitenimen perään lisätään ensin tarvittava domain-tieto, jonka jälkeen kysely voidaan suorittaa FQDN-nimeä hyödyntäen. Funktio käyttää apunaan Pythonin ”dns”-kirjastoa, jonka avulla tiedon hakeminen palvelimilta helpottuu ja kyselyitä saadaan suoritettua tehokkaasti. Funktiossa otetaan ensin yhteys ensimmäiselle palvelimelle, josta palautunut tieto käsitellään sopivaan formaattiin ja sama tehdään sen jälkeen myös toisella palvelimella. Käytännössä molemmilta palvelimilta pitäisi palautua sama tieto, joten näitä kahta verrataan keskenään ja palautetaan tarkastusraportille tarkastuksen lopputulos. Mikäli tietoa ei saada haettua, palautuu funktiosta tarkastusraportille virhe.

Työkalu tarkastaa onko käsittelyssä oleva verkkolaite lisättynä LibreNMS-valvontajärjestelmään käyttäen tarkastuksessa apunaan CMDB:stä haettua laitenimeä. Funktiossa otetaan ensin yhteys LibreNMS:n rajapintaan, johon voidaan suorittaa statuskysely laitenimellä. Rajapinta vastaa kertomalla onko laitteen status järjestelmässä ”ok”. Tämä kertoo työkalulle, onko laite lisättynä järjestelmään, jonka perusteella tarkastusraportille voidaan muodostaa uusi tarkastuskohde palautuneen arvon perusteella. Käytännössä jokaisen tuotannossa olevan verkkolaitteen tulisi olla heti asennuksen jälkeen lisättynä verkonvalvontaan, minkä vuoksi tarkastaminen ja puutteiden ilmaantuessa verkkolaitteen lisääminen järjestelmään on erittäin tärkeää. Näin ei pääse aiheutumaan tilannetta, jossa verkkolaite ei vastaa enää häiriötilanteen vuoksi verkosta, mutta kukaan ei tiedä asiasta, sillä laite ei ole valvonnassa.

Seuraavaksi työkalu tarkastaa varmuuskopioinnin tilan. Tarkastaminen tapahtuu hyödyntämällä laitenimeä, kuten valvontajärjestelmän tarkastuksessakin. Varmuuskopioinnit pidetään Oxidized-järjestelmässä, johon LibreNMS-järjestelmästä voidaan rajapintaa käyttäen tehdä kyselyitä. Varmuuskopioinnin tilan tarkastavassa funktiossa otetaankin ensin yhteys LibreNMS API:in, kuten äskeisessäkin tarkastuksessa. Sen kautta voidaan tehdä erillinen kysely Oxidizediin ja palauttaa kyselyn tuloksena saatu statustieto tarkastusraportille. Tässä kyselyssä statustieto palautuu joko ”true” tai ”false” -arvona, jota voidaan hyödyntää suoraan muuttujasta.

Yhtenä osa-alueena työkalussa on erilaisten laiteryhmien määrittäminen tarkastukseen. Tässä opinnäytetyössä työkalun tarkastukseen määritettiin laiteryhmiksi laitteista verkkokytkimet sekä langattoman verkon tukiasemat. Tukiasemien osalta työkalu ottaa ensin yhteyden EfecteWebAPI:in, josta haetaan tarkoituksenmukaisella kyselyllä langattoman verkon laitteet erilliselle listalle. Seuraavaksi listan laitteet käydään yksitellen läpi ja haetaan jokaisen laitteen tiedot laitenimen perusteella Aruba Mobility Master API:ista. Tämän jälkeen Efectestä saatuja tietoja verrataan Aruba Mobility Masterista saatuihin tietoihin sekä tehdään tarkastukset nimipalveluun, valvontajärjestelmään sekä verrataan laitteen sarjanumeroita järjestelmien välillä.

Langattoman verkon laitteiden sarjanumeron vertailun lisäksi työkalun avulla voidaan tehdä myös muita eri järjestelmien välisiä tarkastuksia. Tässä opinnäytetyössä tehtiin valvontajärjestelmästä haetun laitteen sarjanumeron ja CMDB:stä palautuneen sarjanumeron välinen vertailu verkkokytkimille. Funktio hakee ensin laitetiedot käsittelyssä olevan laitenumeron perusteella ja palauttaa Efecten tiedoista erillisiin muuttujiin sarjanumeron sekä laitenimen. Palautuneen laitenimen avulla voidaan ottaa yhteys LibreNMS-valvontajärjestelmään, josta palautuneista laitetiedoista voidaan hakea laitteen sarjanumero. Seuraavaksi palautuneita sarjanumeroita voidaan verrata funktiossa keskenään ja palauttaa tarkastusraportille jälleen vertailun lopputulos. Mahdollisissa ristiriitatilanteissa sarjanumeroiden välillä voitaisiin valvontajärjestelmästä haettua tietoa pitää luotettavampana, sillä se on haettu suoraan laitteelta. CMDB:ssä oleva tieto on usein manuaalisesti kirjoitettu, joskin useimmissa tapauksissa ensimmäistä konfiguraatiota tehdessä muun muassa sarjanumero on kopioitu suoraan laitteelta. Tässä on kuitenkin aina inhimillinen riski siihen, että kopiointi ei ole onnistunut asianmukaisesti, jonka vuoksi sarjanumeroiden vertailu järjestelmien välillä on tarpeellista.

4.2.3 Virheenkäsittely

Virheenkäsittely on tärkeää, jotta mahdolliset virhetilanteet koodin suorittamisessa saadaan kiinni ja ohjelman suorittaminen ei niiden seurauksena keskeydy. Virheenkäsittelyn tarkoituksena on siis se, että ohjelma pystyy jatkamaan suorittamistaan, vaikka tarkastuksissa tulisikin virheitä. Nämä virheet voidaan esimerkiksi kirjata tarkastusraportin erilliseen osioon ja niihin voidaan sitä kautta myös reagoida.

Tarkastusfunktioihin on sisällytetty ”try-catch” -blokki. Se tarkoittaa sitä, että mikäli jokin virhe ilmenee ohjelman suorittamisessa, siirrytään suoraan Pythonin ohjelmakoodin except-lohkoon. Työkalussa tämä on toteutettu niin, että except-lohkosta palautuu tarkastusraportille arvoksi ”false”. Tämä poistaa tilanteet, joissa esimerkiksi järjestelmästä haettu tieto on jossain muussa muodossa kuin oletettu formaatti. Virheenkäsittelyn avulla voidaan yrittää kyseistä operaatiota, mutta sen epäonnistuessa palautetaan tapahtunut virhe. Tämä on kätevä toimintatapa, kun halutaan varmistaa, että ohjelma pystyy jatkamaan tehtävän suorittamista virheestä huolimatta.

Virheenkäsittely rajapintakyselyiden yhteydessä tapahtuu HTTP-pyynnön yhteydessä, jolloin sille asetetaan aika, jonka sisällä kyseisen pyynnön on suoriuduttava. Tämä on tärkeää siksi, että mikäli API-rajapinnassa ilmenee jokin häiriötilanne, voi käydä niin, että kysely jää odottamaan lopputulosta loputtomasti. Timeout-ajan määrittämisen avulla pyyntö saadaan lopetettua mahdollisessa virhetilanteessa ja palautettua kyseisestä tilanteesta virhe. Kyseinen virhekäsittely hoidetaan ohjelmassa jokaisen HTTP-pyynnön kutsussa.

Näiden lisäksi dictionary-tietorakenteesta tietoja haettaessa palautuu kyselystä virheviesti, mikäli haettua avainta ei löydy. Kyselyyn voidaan määrittää tarpeen mukaan myös haluttu oletusarvo, mikäli kysely päättyy virheeseen. Näin saadaan vaikutettua palautusarvoon ja tarvittaessa haettua virhetilanteet sen mukaan.

4.3 Raportin koostaminen

Kuten luvussa 4.2.2 havaittiin, jokaisen tarkastuksen lopputuloksena palautuu tarkastusraportin kohteelle lopputuloksen mukaan joko ”true” tai ”false” -arvo. Raportti tallennetaan JSON-tiedostona, joten sitä on myös helppo silmäillä loogisen rakenteen vuoksi. Tallentunut tiedosto on näin myös esimerkiksi käyttöliittymän luettavissa jatkokehitystä ajatellen. Raportti itsessään koostetaan niin, että jokaiselle käsittelyn alla olevalle verkkolaitteelle muodostuu yksilöllinen otsikko, mikä puolestaan muodostuu CMDB:n laitenumeron perusteella. Jokainen tarkastuskohta lajitellaan tarkastusraporttiin oman otsikkonsa alle, joista laajimmat eritellään vielä teeman mukaisen otsikon alle luettavuuden helpottamiseksi. Tässä vaiheessa nähtiin tarpeelliseksi muodostaa erillisiksi teemoikseen Efectestä tehtävät tarkastukset sekä järjestelmien väliset tarkastukset. Näihin teemoihin nähtiin mahdollisuudet tehdä myös muita tarkastuksia, joita voidaan jatkossa kehittää. Siitä syystä on hyvä jo tässä vaiheessa miettiä jatkokehityksen tarpeita. Kuvassa 5 esitetään tarkastusraportin muodostaminen ohjelmakoodissa ja ajatus siitä, kuinka tarkastukset eri tarkastuskohteille on tehty.

```
for key, value in device_data.items():
    results[key] = {
        "efecte": {
            "hostname": checkName(key),
            "model": checkModel(key),
            "serial": checkSerial(key),
            "ip-address": checkIP(key),
            "mac-address": checkMac(key),
            "product": checkProduct(key)
        },
        "dns": checkDNS(value.get("Name")),
        "nms": checkNMS(value.get("Name")),
        "oxidized": checkBackup(value.get("Name")),
        "cross-check": {
            "nms-serial-efecte": compareSerial(key)
        }
    }
```

Kuva 5. Kuvaleike raportointityökalusta, tarkastusraportin muodostaminen ohjelmakoodissa

Jokaisen käsittelyssä olevan laitteen suoritettua kaikki tarkastukset saadaan raportille näyttävä kooste eri tarkastusten tuloksista. Tämän jälkeen tarkastusraportti käydään läpi vielä tarkastuskohteittain ja tehdään näistä tuloksista kooste. Koosteella otetaan huomioon kaikki ”false”-arvon saaneet tarkastuskohdat ja siitä nähdään suoraan tarkastuksista löytyneet virheet. Lisäksi koosteeseen sisällytetään tieto siitä, kuinka monta laitetta tarkastuksessa on käyty yhteensä läpi. Näitä tietoja voidaan tarvittaessa hyödyntää esimerkiksi käyttöliittymässä selkeinä mittareina laitetietojen sen hetkisestä tilanteesta.

Raportin muodostamisen jälkeen se tallennetaan erillisessä funktiossa JSON-tiedostoksi. Tiedostoon kirjoittaminen onnistuu ”json”-kirjastoa hyödyntämällä, jossa ”dump”-funktiolle annetaan parametreinä tiedostoon vietävä tieto, tiedostotyyppi sekä raportin ulkoasuun vaikuttavan sisennyksen määritys. Tiedostoon vietävänä tietona tässä tapauksessa on tarkastuskohteista muodostettu raportti sekä kooste tarkastusten tuloksista, jotka molemmat sisällytetään samaan JSON-tiedostoon. Tiedot ovat myöhemmässä vaiheessa luettavissa tiedostosta ja käytettävissä esimerkiksi käyttöliittymän kautta. Tässä vaiheessa työkalun tärkein tehtävä oli muodostaa laitetietojen puutteista luotettava raportti, jota voidaan jatkojalostaa jatkokehityksessä, joten käyttöliittymään ei keskitytty tämän opinnäytetyön kontekstissa. Raportin avulla saadaan kuitenkin laitteittain kattava kooste eri tietojen tilanteesta järjestelmissä. Opinnäytetyön tarkoituksena oli saada kehitettyä uusi työkalu, jota voitaisiin tarpeen mukaan myös laajentaa tulevaisuudessa, johon kyseinen käyttöliittymän kehitys liittyy olennaisesti.

Raportin hyödyntäminen tulevaisuudessa kehitettävässä käyttöliittymässä on mahdollistettu loogisella ja ohjelmallisesti luettavissa olevalla tiedostoformaatilla. Tässä opinnäytetyössä keskityttiin siihen, että tarkastusten lopputuloksena tallentuva tiedosto on hyödynnettävissä jatkokehityksen tarpeisiin, kun tarpeet ovat tarkentuneet. Tarkastusraportin avulla pystytään saamaan kattavaa tietoa eri tarkastusten tuottamista tuloksista ja käyttämään tietoja apuna tarvittavissa muutoksissa. Automaattisesti tuotetun raportin hyöty on siinä, että se pystyy käymään merkittävän laitemassan tarkastukset läpi verrattain lyhyessä ajassa suhteessa ihmisen tekemiin tarkastuksiin. Lopputuloksena tuotetun raportin avulla saadaankin vähennettyä manuaalisia tehtäviä ja keskityttyä enemmän tuottavaan työhön.

5 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli kehittää Istekin tietoliikennepalveluiden käyttöön uusi raportointityökalu, jonka avulla laitetiedot saadaan tarkastettua luotettavasti yrityksen toiminnanohjausjärjestelmästä. Myös tarkastusten suorittaminen muihin järjestelmiin toiminnanohjausjärjestelmästä haettujen tietojen perusteella olivat työn tavoitteena. Tavoitteena oli saada myös yhtenevää tietoa järjestelmien välille raportointityökalun tekemien tarkastusten perusteella sekä varmistaa laitteiden päätyminen laskutukseen. Olemassa oleva työkalu haluttiin samassa yhteydessä korvata uudella koodipohjalla, sillä sen käyttämälle ohjelmointikielelle ei ollut saatavilla enää tukea ja työkalun kehitys nähtiin haastavaksi jatkossa.

Raportointityökalun suunnittelussa otettiin huomioon asetetut reunaehdot ja tiimin sisäiset tarpeet muun muassa ohjelmointikielen valintaa ajatellen. Ohjelmointikielen valinnassa ohjasi tiimin aiemmat kehitystyöt sekä sitä kautta tiimin osaaminen Python-kieleen. Myös ohjelmointikielen tehokkuus työympäristössä, jossa työkalujen ohjelmointi ei ole päivittäistä on tärkeää ottaa huomioon. Näin aikaa ei mene tehtävien tekemiseen, jotka ohjelmointikielessä itsessään on jo huomioitu esimerkiksi lisänä asennettavan ohjelmointikirjaston sisällä. Tässä työkalussa pystyttiin tehokkaasti hyödyntämään valmiita lisäosia ja keskittymään tarvittavien tarkastusten tekemiseen sekä raportin muodostamiseen verkkolaitteittain.

Suunnitteluvaiheessa oli tärkeää pitää huolta siitä, että kokonaisuus säilyy eheänä ja toiminnanohjausjärjestelmästä haettavat tiedot olisivat niitä, joita asiantuntija tarvitsee arkipäiväisissä tehtävissään. Näin työkalun avulla voitaisiin maksimoida saatu hyöty, kun yleisimmät puutteelliset tiedot jäisivät tarkastusraportille. Tarkastusraportin avulla tietoja voitaisiin havaita tehokkaammin ja niihin pystyttäisiin myös reagoimaan. Jatkossa puutteellisia tietoja voisi ajatella olevan entistä vähemmän ja tarkastuksen seurauksena ne tulisivat ilmi aiempaa nopeammin ja automaattisesti. Suuremmassa mittakaavassa voidaan ajatella raportointityökalun parantavan työn tehokkuutta ja tietojen luotettavuutta sekä tuovan tuotannossa olevat laitteet ajantasaisemmin laskutukseen.

Mielestäni suunnitteluvaiheessa työkalun tärkeimmät ominaisuudet saatiin kartoitettua hyvin ja niiden pohjalta pystyttiin suunnittelemaan toiminnallisuuksia tietojen hakuun ja tarkastamiseen eri järjestelmien välillä. Toimeksiantajan käytössä on useita eri järjestelmiä, mutta työkalun kehityksessä haluttiin keskittyä niistä tärkeimpiin ja saada työkalu toimintaan ensin. Myöhemmässä vaiheessa voidaan ajatella, että työkalua voitaisiin laajentaa myös muista järjestelmistä tehtäviin tarkastuksiin. Tässä vaiheessa läpikäydyillä tarkastuksilla saadaan kuitenkin jo merkittävä osa järjestelmistä katettua, joten laajennukselle ei nähty vielä tarvetta. Toimintaympäristö laajenee tulevaisuudessa hyvin suurella todennäköisyydellä, joten työkalun on hyvä olla tähän muutokseen sopeutuva. Mielestäni suunnitteluvaiheessa tähän asiaan pystyttiin kiinnittämään jo hyvin huomiota, joten en usko, että työkalun laajentaminen tulevaisuudessa on ongelmallista.

Suunnitteluvaiheessa ei ollut vielä tarkentunut missä lopullinen raportti halutaan esittää. Kävimme toimeksiantajan kanssa ajatusten vaihtoa siitä, esitetäänkö raportti käyttöliittymässä vai onko se vain kooste tuloksista, jotka voidaan lukea suoraan tiedostosta. Joka tapauksessa JSON-muodossa oleva raportti oli tahtotila alusta alkaen ja toteutuksen edetessä se päädyttiin tallentamaan sellaisenaan

hakemistoon. Tietoliikennepalveluille on tulossa myös oman kehitystyön tuloksena syntyvä erillinen hallintatyökalu, jonka sisään voitaisiin jatkossa rakentaa raportille oma käyttöliittymä. Tämänkin vuoksi tällä hetkellä raportin tallentaminen tiedostoon riittää toimeksiantajan tarpeisiin erittäin hyvin. Käyttöliittymän osalta tarpeet tarkentuvat ajan kuluessa, jolloin siihen voidaan panostaa enemmän ja tuoda opinnäytetyön aiheena muodostuneelta raportilta tarvittavia tietoja esille.

Toteutusvaiheessa haasteita aiheutti alkuun Python-ohjelmointikieleen syventyminen. Olin aiemmin käyttänyt muutamassa projektissa kyseistä ohjelmointikieltä, mutta syvempi perehtyminen oli jäänyt vielä vaiheeseen. Työn edetessä huomasin, että syntaksi alkoi käydä yhä tutummaksi ja kehitti osaamistani huomattavasti. Huomasin myös, että useat kommentit Pythonin nopeasta oppimisesta ja selkeydestä sekä tehokkuudesta pitivät hyvin paikkansa. Tämän kehitystyön jälkeen minulle heräsi motivaatio opetella lisää ja harjoitella entistä haastavampia projekteja, jotta ammatillinen kasvu ei jäisi tähän. Työkalun tarkastusfunktiot saatiin toteutettua mielestäni erittäin tehokkaasti Pythonilla sekä sen tarjoamien ohjelmointikirjastojen avulla.

Toteutusvaiheessa eräänä haasteena tuli eteen myös tietorakenteiden käsittelyt. Raportointityökalussa käytettäväksi tietorakenteeksi valikoitui Pythonin dictionary, joka ei itselleni ollut alkuun millään lailla tuttu. Etenkin raportin koosteen muodostamisessa joutui miettimään useaan kertaan, kuinka tietorakenteen avain-arvoparit toimivat erilaisten hakujen yhteydessä. Raportti muodostettiin niin, että sen sisällä oli yksittäisten tarkastusten lisäksi erilliset koostetut tarkastukset Efecte- ja järjestelmien välisille tarkastuksille. Nämä koostetut tarkastukset muodostivat raporttiin vielä sisäkkäisen dictionaryn, josta tietojen hakeminen koitui yllättävän haastavaksi.

Virheidenkäsittelyyn pyrittiin kiinnittämään jo varhaisessa vaiheessa huomiota, sillä suuren laitemassan läpikäynti ohjelman avulla vaatii sen, että ohjelma pystyisi suorittamaan tehtävät loppuun saakka. Jokaisessa funktiossa pyrittiin ottamaan huomioon yksilölliset tarpeet ja kehitystyössä testattiin erilaisia tilanteita sekä haettiin laitekannaksi erilaisia laitteita. Näiden pohjalta pystyttiin kartoittamaan puutteet ja kiinnittämään asioihin paremmin huomiota ohjelmassa.

Työkalun kehittämisen pääpaino oli siinä, että toiminnallisuudet saataisiin kuntoon mahdollisimman kattavasti. Tavoitteena oli, että työkalu pystyisi suorittamaan tarvittavat tarkastukset monipuolisesti ja funktioiden hienosäätö tehtäisiin vasta myöhemmässä vaiheessa. Tämä osoittautui hyväksi toimintatavaksi, sillä aikaa työkalun kehittämiseen oli muiden töiden ohella huomattavan vähän. Haastavinta työkalun tekemisessä oli se, että alkuun toiminnallisuudet hahmottuivat melko hitaasti ja meni pitkiäkin aikoja ennen perehtymistä seuraavaan asiaan. Siksi oli myös haastavaa muistaa, millainen ajatus tiettyjen funktioiden käsittelyyn oli ollut edellisen kerran kehitystä tehtäessä.

Loppua kohti työkalun kehittämiseen löytyi kuitenkin enemmän aikaa ja sen vuoksi funktiot rakentuivat suhteellisen hyvään tahtiin ja toiminnallisuudet saatiin käyttöön vaatimusten mukaisesti. Kehitystyön aikana oli useita kertoja, jolloin täytyi miettiä ominaisuuksien laajuutta ja välillä niitä oli myös hyvä rajata tämän työn ulkopuolelle. Työn pohjalta on kuitenkin mielestäni hyvä lähteä tekemään työkalun kehitystä eteenpäin tulevien tarpeiden perusteella.

Työkalun kehityksen pohjalta oli loppuvaiheessa tiedostettavissa jo jatkokehityskohteita. Työkalu on tarkoitus ottaa käyttöön Istekin tietoliikennepalveluiden kehittämän verkon hallintaohjelman lisäosana, johon voidaan tulevaisuudessa kehittää käyttöliittymä tarkastusraportin tietojen lukemista varten. Tämä on suurin ja tärkein jatkokehityksen kohde, joka mielestäni on hyvä ottaa kehityksen alle heti, kun uusi järjestelmä saadaan käyttöön. Järjestelmän avulla saadaan raportin lukemisesta miellyttävämpää ja sen avulla pystytään mahdollisesti hakemaan helpommin haluttuja asioita.

Käyttöliittymän lisäksi ohjelman sisälle voitaisiin kehittää funktiot käsittelemään laitelistat asiakkuuksittain. Toimeksiantajan asiakkuuksilla on erilaisia tarpeita, jolloin näihin tarpeisiin pystyttäisiin työkalun tarkastusten avulla reagoimaan tehokkaammin erillisten asiakaskohtaisten laitelistauksien perusteella. Asiakkuuksille voitaisiin tämän jälkeen muodostaa omat yksilölliset tarkastusraportit ja niitä hyödyntää yksilöllisemmin. Myös tietyt laiteryhmät voitaisiin erotella omikseen saatujen tietojen perusteella. Laiteryhmiä voisivat olla esimerkiksi verkkolaitteet, joille on määriteltynä saatavuustaso tai paikkakuntakohtaiset laitetiedot tietyn laitetyypin osalta. Tämän avulla saadaan raportista suurempi hyöty, kun tiedot olisivat suuresta massasta lajiteltuina pienempiin osa-alueisiin tarpeen mukaan.

Yksi kehitettävä osio kehitetyssä työkalussa voisi olla myös eri järjestelmien välisten tarkastusten laajentaminen. Työkalun avulla saadaan tarkastettua laitteiden sarjanumeroiden vastaavuutta eri järjestelmissä. Laitetiedoista on käytössä kuitenkin useita muitakin eri tietoja, joiden avulla voidaan tehdä tarkastuksia tietojen yhteneväisyyteen. Efectessä olevan tiedon pitäisi aina olla oikein, jotta siitä saadaan kaikki hyöty irti. Työkalun ottaessa yhteyden eri järjestelmiin, voidaan laitteelta hakea suoraan luotettavat tiedot. Tietoja voisivat olla esimerkiksi laitteen erilaiset tilatiedot valvontajärjestelmässä tai tarkastus laitteille, jotka vastaavat verkosta, mutta eivät ole lisättyinä valvontaan.

Merkittävä jatkokehityksen askel voisi olla myös laitetietojen automaattinen päivitys soveltuvin osin. Työkalussa on rajapintoihin tarvittavat yhteydet ja sen avulla voitaisiin ainakin Efecte-tietoja päivittää myös automaattisesti puutteellisten tietojen ilmaantuessa. Laitetietojen päivittämisessä tulee kuitenkin olla tarkkana, että tiedot eivät missään vaiheessa mene sekaisin ja aiheuta jatkossa ongelmia. Tämän ominaisuuden avulla työkalusta saataisiin asiantuntijoiden arkea vielä entisestään helpottava työkalu ja saataisiin poistettua yksinkertaisia poikkeamia laitetiedoissa. Tämä on ominaisuutena joka tapauksessa sellainen, joka tulee työkalun kautta käyttöön jossain vaiheessa tietyssä laajuudessa.

Tietoliikennepalveluiden asiantuntijat tekevät paljon yhteistyötä myös muiden ryhmien kanssa. Asiantuntijat saattavat saada esimerkiksi lähitukiryhmän asiantuntijoilta kyselyitä mihin langattomaan tukiasemaan jokin työasema on yhdistynyt. Kyseessä on rutiininomainen toimenpide, joka voitaisiin työkalua hyödyntäen toteuttaa jatkossa niin, että kehitettävään verkonhallintajärjestelmään luotaisiin haku toimenpidettä varten. Tämä helpottaisi molempien ryhmien työtä kyseisissä tilanteissa ja nopeuttaisi myös toimintaa, kun tiedon saisi ilman yhteydenottoa asiantuntijaan. Haku voitaisiin kehittää esimerkiksi niin, että asiantuntija syöttäisi työaseman numeron hakukenttään, jonka avulla haettaisiin tarvittava tieto langattoman verkon ohjaimilta ja palautettaisiin käyttöliittymään.

Työkalun toiminnallisuutta testatessa huomasin, että tiettyjen toiminnallisuuksien osalta kyselyiden kestoissa oli merkittävää viivettä. Tämä oli tärkeä huomio tässä vaiheessa, sillä isompaa laitemassaa

ajettaessa olisi hyvä optimoida jokainen vaihe mahdollisimman tehokkaaksi. Tämän avulla kokonaissuoritusaika säilyy kohtuullisena ja ohjelman suorittaminen on mielekästä kaikin puolin. Eräs havaittu ongelmakohta oli langattoman verkon rajapintakyselyn toteuttaminen, jossa yksittäisen laitteen osalta tehtiin tarkastuksissa kaksi kertaa sama kysely laitetietokantaan eri tarkastusfunktiossa. Tähän oli järkevää tehdä muutos niin, että tarkastusten alussa tehdään vain yksi kysely, jossa haetut tiedot tallennetaan globaalille listalle, jota voidaan hyödyntää jatkokyselyissä tehokkaammin. Kyseiset tiedot ovat ajantasaisia siitä huolimatta, että niitä käytetään myöhemmässä vaiheessa tarkastuksia uudelleen. Näin ei tarvita jokaisessa vaiheessa tehdä samaa kyselyä uudelleen, jolloin ohjelman suorittaminen nopeutuu ja liitännäisjärjestelmien kuormatkaan eivät tarpeettomasti nouse kyselyn seurauksena.

Toinen kehitettävä asia liittyy ohjelmakoodin yleiseen suorittamiseen tietyin osin. Huomasin testausvaiheessa, että jokaiselle laitteelle toiminnanohjausjärjestelmästä haettavat tiedot odottavat edellisen hakutoiminnan suorittamista loppuun. Tämä hidastaa ohjelman suorittamista kokonaisuutena merkittävästi, kun seuraavaan hakuun siirrytään vasta edellisen jälkeen. Toimintaa saadaan näppärästi tehostettua useamman ajon yhtäaikaisella käynnistämisellä. Käytännössä tämä voitaisiin toteuttaa säkeistämällä, jolloin esimerkiksi laitelista voitaisiin jakaa tietyn kokoisiin osiin ja nämä osiot voitaisiin laittaa yhtäaikaisesti ajoon. Esimerkiksi 120 laitteen laitelistasta voitaisiin tehdä 30 laitteen kokoisia osioita, joita muodostuisi neljä kappaletta. Nämä osiot voitaisiin laittaa yhtä aikaa suoritettavaksi ja näin nopeuttaa merkittävästi haun suoritusaikaa.

Työkalun tekeminen oli mielestäni innostavaa. Pidin siitä, että työkalusta on tulevaisuudessa apua kollegoideni työssä ja, että se tulee oikeaan tarpeeseen. Yhteistyö toimeksiantajan kanssa meni myös todella sujuvasti ja asioita saatiin edistettyä hyvässä tahdissa aina tarpeen mukaan. Myös apua ongelmatilanteisiin sai erittäin hyvin ja siitä suuri kiitos myös opinnäytetyön ohjaajalle. Opinnäytetyön tekemisessä suurimpana oppina itselle oli syvällisempi perehtyminen Python-ohjelmointikieleen. Olin aiemmin tehnyt muutamia henkilökohtaisia vapaa-ajan projekteja kieltä hyödyntäen, mutta tässä mittakaavassa projekti oli ensimmäinen. Aina uuteen ohjelmointikieleen tutustuttaessa tulee vastaan tilanteita, jolloin joutuu miettimään miksi asiat eivät toimi. Hiljalleen harjoittelemalla aloin päästä syntaksiin kiinni ja sain funktioita toteutettua.

Opettavaisia tilanteita tuli kuitenkin eteen muun muassa tietorakenteiden käsittelyiden kanssa. Työkalu muodosti tarkastuskohteista sisäkkäisiä dictionaryjä, joiden läpikäynti koosteraporttia varten vaati syvempää porautumista tietorakenteeseen. Tämä kohta vaati yllättävän paljon työstämistä ja aikaa, mutta oli samalla todella opettavaista, kun asiaa sai lopulta etenemään ja oikeaa tietoa haettua. Huomasin, että pala kerrallaan etenemällä ja asioita testaamalla konsoliin oli mielekästä nähdä mitä funktio saa aikaan. Tämän tavan avulla pystyin näkemään käytännössä mikä asia vaatii korjaamista ja sain edistettyä logiikkaa tietojen haussa oikeaan suuntaan.

Oli myös mielenkiintoista toteuttaa yhteyksien ottamista eri järjestelmiin rajapintojen kautta. En ollut aiemmin tehnyt tätäkään vastaavassa mittakaavassa, joten se opetti merkittävästi uusia asioita toimintatavoista ja toteutustekniikoista. Uskon, että samoja periaatteita hyödyntämällä pystyn suorittamaan myös muissa tulevaisuuden projekteissa samanlaisia tehtäviä ja iskostamaan asioita entistä paremmin selkärankaan. Oli myös tärkeää päästä näkemään, mitä kaikkea tietoa rajapintojen avulla

voidaan järjestelmistä hakea. Tämä helpottaa jatkokehitystä työkalun kanssa, kun tietää paremmin, mitkä asiat ovat mahdollisia saada tietoon rajapintaa hyödyntäen.

Seuraavaksi työkalua voidaan ruveta tulevissa vaiheissa kehittämään tarpeen mukaan ja lisäämään siihen uusia ominaisuuksia. Toimeksiantajan toimintaympäristössä tapahtuu usein muutoksia, ja työkalun toteutuksessa ei ole pyritty siihen, että se pystyy vastaamaan jokaiseen tarpeeseen heti. Se on kuitenkin jatkokehitettävissä ja muokattavissa tarpeiden mukaan, mikä on tärkeä asia tulevaisuudessa. Jatkokehitystarpeiden mukaan työkaluun on mahdollista lisätä uusia ominaisuuksia ketterästi uusia tarkastusfunktioita kehittämällä. Uskon, että kehitetystä työkalusta tulee merkittävä osa laitetietojen ajantasaisuuden varmistamiseen ja se helpottaa myös työntekijöiden tekemistä jatkossa.

6 LÄHDELUETTELO

Efecte a. (ei pvm). *Pilvipohjaista palvelunhallintaa*. Haettu 15. 10. 2021 osoitteesta Efecte: https://www.efecte.com/fi/home

Efecte b. (ei pvm). *What is Service Management*. Haettu 15. 10. 2021 osoitteesta Efecte: https://www.efecte.com/what-is-it-service-management

Efecte c. (ei pvm). *Service Configuration Management*. Haettu 15. 10. 2021 osoitteesta Efecte: https://www.efecte.com/platform/service-configuration-management

Gift, N.;& Jones, J. M. (2009). *Python for Linux and Unix System Administration.* Sebastopol, California: O'Reilly Media, Inc.

LibreNMS Docs. (ei pvm). *LibreNMS Features*. Haettu 15. 10 2021 osoitteesta LibreNMS Docs: https://docs.librenms.org/Support/Features/

LibreNMS. (ei pvm). *LibreNMS*. Haettu 15. 10. 2021 osoitteesta LibreNMS: https://www.librenms.org

Pip documentation v22.0.4. (ei pvm). *User Guide*. Haettu 18. 3. 2022 osoitteesta Pip documentation v22.0.4: https://pip.pypa.io/en/stable/user_guide/

Python Package Index. (ei pvm). *The Python Package Index*. Haettu 18. 3. 2022 osoitteesta Python Package Index: https://pypi.org/

Python Packaging Authority. (ei pvm). *Python Packaging User Guide*. Haettu 18. 3. 2022 osoitteesta Python Packaging Authority: https://packaging.python.org/en/latest/overview/

Python.org a. (ei pvm). *JSON encoder and decoder*. Haettu 17. 2. 2022 osoitteesta Python.org: https://docs.python.org/3/library/json.html

Python.org b. (ei pvm). *Built-in Types*. Haettu 13. 4. 2022 osoitteesta Python.org: https://docs.python.org/3/library/stdtypes.html#typesmapping