

Tristan Lumme

## 24 Pesulan seurantaohjelmisto

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

11.5.2017

Tekijä(t) Otsikko	Tristan Lumme 24 Pesulan seurantaohjelmisto
Sivumäärä Aika	49 sivua + 3 liitettä 11.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Lehtori Jaana Wuorila-Stenberg Osakas Nico Nevala
<p>Insinööri työ tehtiin 24 Pesula Oy:n projektista, joka suoritettiin Helsingissä. Projektin tarkoituksena oli luoda sovellus listaamaan yrityksen toimipisteiden koneet ja niiden käyttökerrat automatisoidusti tietokantaan.</p> <p>24 Pesula Oy on vuonna 1999 perustettu yritys ja Suomen ainoa itsepalvelupesulaketju. Aikaisemmin koneiden käytönseuranta hoidettiin siten, että käyttödata haettiin paikan päältä lukemalla yksittäisten koneiden lokitiedot. Yrityksen koneissa käytettiin myös Atmelin modifioituja piirikortteja, joihin oli liitetty MC66 GSM -modeemi. Piirikorttien avulla saatiin lähetettyä käyttötiedot reaaliaikaisesti palvelimelle.</p> <p>Paikan päältä kerätyt, koneiden käyttökerrat eivät sisältäneet aikatietoja, mutta palvelimen vastaanottamista soittotiedoista saatiin myös aika selville. Palvelin ei tosin osannut listata käyttötietoja tietokantaan. Vastaanotettu tieto esiintyi soittonumerona, jonka palvelin tallensi XML-tiedostoon. Laitetiedon hakeminen taulukosta soittonumeron avulla oli ihmiselle hyvin työlästä, kun käyttökertoja alkoi olla tuhansia.</p> <p>Tässä opinnäytetyössä esitetään 24 Pesula Oy:n historia ja sen konsepti. Lisäksi käydään tarkkaan läpi 24 Pesulan seurantaohjelmiston kehitysprosessia suunnittelu- ja kehitysvaiheista Alfa-versioihin asti ja tutustutaan ohjelmiston käyttöön. Ohjelmointipuolesta käydään läpi ainoastaan päänäköymän luonnissa käytettyä C#-koodia, koska koko ohjelmiston koodin esittäminen ei olisi kannattavaa sen laajuuden ja muutosherkkyyden vuoksi.</p> <p>Projektissa onnistuttiin hyvin ja 24 Pesulan seurantaohjelmisto sai pelkistetyn listausominaisuuden lisäksi kehittyneempiä lisäominaisuuksia ja hyödyllisiä parannuksia. Lisäominaisuudet, kuten tuloksarsinta ja trendinäköymä, helpottivat tulosten tarkastelua huomattavasti.</p>	
Avainsanat	24 Pesula, C# ohjelmointi, Ohjelmisto

Author(s) Title	Tristan Lumme The Monitoring Software of 24 Laundromat
Number of Pages Date	49 pages + 3 appendices 11 May 2017
Degree	Bachelor of Engineering
Degree Programme	Automation technology
Specialisation option	
Instructor(s)	Jaana Wuorila-Stenberg, Senior Lecturer Nico Nevala, Co-Owner
<p>24 Laundromat Ltd had a project in Helsinki that aimed to create a piece of software that would automatically read the result data of each machine in company locations and list the result into a database.</p> <p>24 Laundromat Ltd was formed in 1999 and it is the only self-service laundry chain in Finland. Result tracking was earlier performed by reading the list of uses directly from every machine in every post. The company also used modified circuit cards by Atmel that were equipped with a MC66 GSM modem. The circuit cards allowed every use to be sent to the server in real time.</p> <p>The list of uses that was read directly from the machines did not contain time information but the call records received by the server were marked with date and time. The received data was shown as a call number that was saved in a XML file. The information of a machine had to be found by using this call number and then compared with a list. The job was laborious for a person to do, as the amount of uses reached thousands.</p> <p>This thesis will focus on the history and concept of 24 Laundromat Ltd. The development process of the Monitoring Software is later on reviewed carefully. The review will start from the early steps of the design and end to the Latest Alfa versions. As concerns the programming, only some parts of the C# code associated with the software main view, will be reviewed because the code is very extensive and sensitive for changes.</p> <p>The project succeeded and the Monitoring Software of 24 Laundromat got more advanced features and useful improvements. Features like eliminating results and a trend view made it much easier to consider the results.</p>	
Keywords	24 Laundromat, C# Programming, Software

# Sisällys

## Lyhenteet

1	Johdanto	1
2	24 Pesula Oy	2
2.1	Alkutaival	2
2.2	Nykytilanne	3
2.3	Tulevaisuuden näkymä	4
3	Seurantaohjelmisto	5
3.1	Ohjelmiston tarkoitus	5
3.2	Suunnittelu- ja luontiprosessi	6
3.2.1	Kokeiluvaihe	7
3.2.2	Päänäkymän luonti C#-ohjelmointikielellä	9
3.2.3	Tulosten listaus	16
3.3	Versiovaiheet	22
3.3.1	Kehitysversiot	23
3.3.2	Alfa-vaihe	26
3.3.3	Jatkosuunnitelmat	33
3.4	24 Pesulan seurantaohjelmiston käyttö	35
3.4.1	Aloitustoimenpiteet	35
3.4.2	Tiedosto-valikko	37
3.4.3	Trendi	40
3.4.4	Tuloskarsinta	43
3.4.5	Lisätietoja	46
4	Yhteenveto ja mietteet	47
	Lähteet	48

## Liitteet

Liite 1. XML-lokikansion valinta

Liite 2. Konelistauksen CSV-tiedoston ulkoasu

Liite 2. Pällekkäisyys päiväkohtaisessa trendissä

## Lyhenteet

APL	A Programming Language. Kanadalainen, vuonna 1957 kehitetty, notaatioon perustuva sääntökieli, jonka peruseriaate on ilmaista matemaattiset toiminnot eri symboleilla ja niiden yhdistelmillä.
Atmel	Kalifornialainen yritys, joka on maailmanlaajuisesti johtavassa asemassa tuottaen mikro-ohjaimia, kapasitiivisia kosketusratkaisuja, logiikoita, radio- taajuus- ja muita nykypäivän komponentteja kehittämään teollista yritys- maailmaa sekä IoT:ta.
C#	Microsoftin kehittämä ohjelmointikieli, joka muistuttaa C, C++ ja Java kie- liä. Se on helppokäyttöisempi kuin C++ ja täyttää monia Javan puutteita.
CLR	Common Language Runtime. .NET Frameworkin tarjoama run-time- ympäristö, joka sisältää kehitysprosesseja helpottavia ominaisuuksia ja koodin suoritusta.
CSV	Comma-separated values. Tiedostotyyppi, johon voi tallentaa dataa tau- lukkomuotoon. CSV-tiedosto sisältää tekstiosia, jotka on eroteltu pilkuilla tai puolipisteillä.
MC66	Nelitaajuuksinen GSM/GPRS-modeemi teollisuuskäyttöön.
MySQL	Ruotsalaisryityksen kehittämä relaatiotietokantaohjelmisto, joka tarjoaa kehitykseen ja hallintaan tarkoitettuja työkaluja.
SQL	Structured Query Language. SQL on standardi ohjelmointikieli tallentami- seen, manipulointiin ja tietojen hakemiseen relaatiotietokannasta.
XAMPP	Kehittäjille suunnattu ympäristö, jolla on helppo luoda paikallisia Apache- verkkopalvelimia ja yhdistää niihin MariaDB:ta, PHP:ta, Perlia ja MySQL:aa.
XML	Extensible Markup Language. Tiedostotyyppi, jossa on tarkasti määritetty tiedon rakenne ja merkitys.

## 1 Johdanto

24 Pesula on Suomen ainoa itsepalvelupesulaketju. Se perustettiin vuonna 1999. Yritys toimi pitkään täysin perheen sisäisin voimin, mutta sitä mukaan, kun yritys on kasvanut, on myös lisäävun tarve kasvanut eri työtehtävissä.

Isoksi haasteeksi koitui vähitellen yrityksen kyky seurata omaa kehitystä, koska toimipaikkojen lisääntyminen, sekä asiakasmäärän runsastuminen on kasvattanut yksittäisten koneiden käyttömääriä rajusti. Tämä käyttömäärien nousu johti siihen, että oli kannattavaa aikaa kehittää automatisoitua seurantaohjelmistoa.

Seurantaohjelmiston toimintaperiaate kartoitettiin siten, että se taulukoi koneiden käyttömäärät samalla periaatteella, kuin se oli tehty aikaisemmin manuaalisesti. Nyt prosessi onnistuu nopeammin ja nykyaikaisesti tietokoneen avulla, jolloin ihminen voi keskittyä valmiiksi listattujen tulosten jälkikäsittelyyn.

Ohjelmiston suunnittelu, toteutus ja kehitys on toteutettu Metropolia Ammattikorkeakoulun toisen työharjoittelujakson ja insinööriyön aikana. Työn suunnitteluvaihe ja ensimmäiset kokeelliset toteutukset suoritettiin ensimmäisen kuukauden aikana, minkä jälkeen ohjelmaa testattiin ensimmäistä kertaa oikeassa työympäristössä. Toisena kuukautena korjattiin ohjelmiston vikoja ja suunniteltiin tulevia ominaisuuksia. Kun työharjoittelujakso päättyi joulukuun lopulla, jatkettiin työsuhdetta toistaiseksi. Harjoittelujakson jälkeen ohjelmisto on kokenut isoja muutoksia ja saanut paljon hyvää palautetta.

Koska yritys kasvaa kovaa vauhtia, on saatava kustannuksellisesti mahdollisimman paljon hyötyä irti. Automatisoimalla yksinkertaisia, mutta aikaa vieviä työtehtäviä, saadaan työvoimaa kohdistettua hyödyllisemmin.

Tässä opinnäytetyössä perehdytään 24 Pesula Oy:n historiaan ja sen toimintaan. Lisäksi käydään tarkkaan läpi Helsingissä luodun 24 Pesulan seurantaohjelmiston kehitysprosessia suunnittelu- ja kehitysvaiheista Alfa-versioihin asti ja tutustutaan ohjelman käyttöön.

## 2 24 Pesula Oy

### 2.1 Alkutaival

24 Pesulan ensimmäinen konsepti syntyi saunan lauteilla, jossa kokkolalaiset Nevalan veljekset miettivät, mikä toimiala rahoittaisi heidän autourheiluharrastustaan. Veljekset etsivät alaa, jossa automaation osuus oli mahdollisimman pientä ja johon heidän osaamisensa saattoi poikia uudenlaista bisnestä. Pohdiskelu johti lopulta pesula-alalle. [1, s. 16.]

Veljekset tutkivat paljon eri markkinoita, toimialoja ja kysyivät näkemyksiä eri tahoilta. Pesula-alan väeltä kysyttiin neuvoja, miten toiminnan kuuluisi hoitua, koska kokemusta ei ollut. Kun tietoa oli tarpeeksi ja idea vaikutti kannattavalta, he ottivat lainaa yrityksen perustamista varten. Nevalan veljeksiä asui eri puolilla Suomea ja koska jokaisella oli oma palkkatyönsä, oli selvää, että yrittäjyys pidettiin sivutoimisena. [1, s. 16.]

Ideana oli luoda systeemi, joka helpottaa ihmisiä arkisissa pyykinpesuhaasteissa. Esimerkiksi matot ja isot kodintekstiilit ovat usein haasteellisia artikkeleita pestäväksi kotikonstein. Monissa taloyhtiöissä on oma pesutupa, mutta niissä ei aina pysty tai saa pestä isoja ja painavia pyykkejä, kuten mattoja. Tästä syystä ihmiset ovat vieneet haastavampia pestäviä suoraan palvelemaan pesulaan. Pesulat eivät kuitenkaan ole kaikille ensisijainen ratkaisu suurehkon hintansa takia. [1, s. 16.]

Ratkaisuksi tuli välimuotoinen kompromissi, joka sisältää vastaavia palveluita, kuin perinteisessä pesulassa, mutta vaatii asiakkaalta itsepalvelua [1, s. 16].

Marraskuussa 1999 perustettiin avoin yhtiö, joka oli nimeltään Pesukatti. Pesukatista tuli Suomen ensimmäinen täysin automatisoitu itsepalvelupesula. Ajatuksena oli, että vaikka tekniikan mahdollisuudet ovat rajattomat, asiakas ei saa tuntea itseään hölmöksi tekniikan kanssa. [1, s. 16 -17.]

Automatisoinnin ansiosta työntekijöitä ei tarvita, kuin ajoittaisia huoltotoimenpiteitä varten. Nevalan perhe on näin kyennyt panostamaan enemmän yrityksen liiketoiminnan suunnitteluun (kolmen vuoden jaksoissa), rakentamiseen ja palvelupisteiden lisäämiseen. [1, s. 16 - 17.]

## 2.2 Nykytilanne

Pesukatti vaihtoi nimensä 24 Pesulaksi vuonna 2012 tammikuussa ja on edelleen Suomen ainoa itsepalvelupesulaketju. Automaattisuus on iso valtti, mutta myös paikalla olevaa henkilökuntaa käytetään väliaikaisesti, jos sille koetaan tarvetta. Esimerkiksi uuden laitteen käytön alkumetreille on hyvä saada asiantunteva henkilö opastamaan. Näin myös kokeillaan, onko laite mahdollista pitää asiakkaiden käytössä ilman henkilökunnan opastusta. 24 Pesulalla on oma tuotanto- ja tuotekehitysosasto, jossa suunnitellaan ja valmistetaan suurin osa pesulakoneista ja hallintajärjestelmistä. [2.]

24 Pesula on edelleen kasvuyritys, mutta se on saanut positiivista noteerausta Tekesiltä. Kehityksen kulku nousee kovaa vauhtia ja tahti kiristyy. Arvoltaan yritys on perheyri-tytys, jonka organisaatorakenne koostuu tällä hetkellä viidestä sisäisestä organisaatio-osasta. Nämä osat ovat: hallinto, tuotekehitys, tuotanto, markkinointi ja ylläpito. Oman henkilöstön lisäksi käytetään myös alihankkijoita. [2.]

24 Pesulalla on tällä hetkellä seitsemän henkilöä, jotka ovat täysipäiväisesti töissä, ja määrää tullaan lisäämään vuoden 2017 aikana. Yrityksellä on yhdeksän omistajaa, joista osa tekee puolipäivittäisesti töitä, jotta yrityksen liiketoimintaa saadaan kehitettyä. [2.]

Pesuloita yrityksellä on 16: Helsingissä olevat toimipisteet ovat Etu-Töölö, Konala, Val- lila ja Viikki. Tampereelta löytyvät Epilä, Tullintori ja Turtola. Muita toimipisteitä ovat Hämeenlinna, Kokkola, Pirkkala, Raisio, Turku ja Ylöjärvi. Myöhemmin mukaan tuli Porin pesula, joka avattiin 24.2.2017 ja Jyväskylän pesula, joka avattiin 13.4.2017. Uu- sin pesula avataan Seinäjoelle 8.5.2017. Asiakaskunta koostuu pääosin 30 - 65- vuotiaista, naispuolisista kuluttajista. Asiakkaiden pesemät tuotteet ovat yleensä matto- ja ja isoja kodintekstiilejä. [2.]

Yrityksen kilpailuetu on digitaalisuus, skaalautuvuus ja uudenlaisen pesulamarkkinan rakentaminen. [2.]



## 2.3 Tulevaisuuden näkymä

Nykyinen markkina-alue 24 Pesulalla on koko Suomen kuluttajakunnat. Liikevaihto on noussut kahdessa vuodessa 170 %, mutta saman kasvuvauhdin ylläpitäminen edellyttää, että toimintaa on laajennettava Suomesta ulkomaisille markkinoille. Tavoitteena on, että 24 Pesula on kansainvälinen yritys vuoteen 2022 mennessä. Ensisijaisena suunnitelmana on laajentaa Iso-Britannian markkinoille vuoden 2017 aikana, jossa 24 Pesula toimii nimellä 24 Laundromat Ltd. [3.]

Strategiana on käyttää samaa konseptia, kuin Suomessakin, mutta paikalliset kuluttajatrendit sekä käyttäytymistavat on otettava huomioon [3].

24 Pesula esitteli Konalassa, Kauppakeskus Ristikossa 25.10.2016 maailman ensimmäisen itsepalvelukäyttöön suunnitellun tasopesukoneen, joka on tarkoitettu kovapohjaisille matoille. Kone on patentoitu ja pesee maton käyttökuivaksi noin 20 minuutissa maton koosta riippuen. [3.] Ensimmäisen tuotekehitysversion käyttökokemuksen perusteella kehitetään tuotantoversio, joka tullaan lisäämään palvelukokonaisuuteen jokaisessa kaupungissa, jossa 24 Pesula palvelee.

Myös laitteiden seuraamiseen ja kommunikointiin suunnitellaan uudempaa ratkaisua tulevaisuudelle. Uudistus sisältää muutoksia niin hardware-puoleen, kuin ohjelmistollisesti. Palvelimeen ja laitteiden kanssa kommunikointiin liittyvistä uudistuksista on lisää tietoa luvussa 3.3.3.

### 3 Seurantaohjelmisto

#### 3.1 Ohjelmiston tarkoitus

Yrityksen menestyksen ylläpitämistä helpottaa se, että pystytään tehokkaasti ja yksityiskohtaisesti seuraamaan sen kehitystä. Tähän asti 24 Pesulan koneiden käyttötietoja on voitu seurata kahdella tavalla. Ensimmäinen tapa on mennä paikan päälle ja lukea pesukoneesta käyttökertojen lukumäärä. Tällöin jokainen kone on käytävä yksitellen läpi, jotta koko toimipisteen käyttödata saadaan selville. Toinen, kehittyneempi tapa on järjestelmä, joka perustuu GSM-laitteilla välitettyyn soittonumeroinformaatioon. Tämä informaatio välitetään palvelimena toimivaan Windows-tietokoneeseen, joka vastaanottaa puhelun ja tallentaa soittajan puhelinnumeron XML-tiedostona Windowsin resursienhallintaan. Jokaisella pesukoneella on konekohtainen soittnumero. XML-tiedostosta selviää myös tallennushetkellä määräytynyt käyttöjärjestelmän aikatieto. Tarkka aika on jälkimmäisen menetelmän etu, koska itse pesukoneet eivät tallenna aikatietoja omaan lokiinsa. [4.]

Palvelimelle tallennettavassa menetelmässä on kuitenkin yksi haitta. Koska tieto esiin-tyy vain puhelinnumeronä yksittäisessä XML-tiedostossa, tämä numero on ensin luet-tava tiedostosta ja sitten verrattava erillisellä listalla oleviin numeroihin. Jos numero on sama kuin listalla, saadaan tietää, mille koneelle kyseinen puhelinnumero kuuluu. Tä-mä operaatio on erittäin työläs, jos vertailun tekee ihminen. Vuoden 2016 aikana 24 Pesula on rekisteröinyt tuhansia käyttöjä puolen vuoden aikana, ja tämä määrä tulee vain nousemaan sitä mukaa, kun toimipaikkoja, konetyyppejä ja asiakkaita tulee lisää. Tilanne vaatii muutosta tapaan, miten tuloksia seurataan, ja koska kyseessä on auto-matisointiin perustuva yritys, on korkea aika, että myös tulosten seuranta automatisoi-daan. [4.]

Toimeksiantajalla oli vain yksi pyyntö. Toimeksiantaja halusi, että yritykselle luodaan ohjelmisto, joka lukee pesukoneiden lähettämät puhelinnumerot ja vertailee niitä tieto-kannan tyyppiseen listaan. Tämä vertailtava lista sisältää kaikkien 24 Pesula Oy:n ko-neiden tunnistamiseen tarvittavat tiedot. Ohjelmiston on tarkoitus löytää täsmäävä kone listalla olevien puhelinnumeroiden perusteella. Täsmäävä kone tulostetaan ohjelmiston listanäkymän riville, josta käy ilmi löydetyn koneen tyyppi, paikka, palvelunumero, soit-tonumero ja koneen käyttöaika sekunnin tarkkuudella.

### 3.2 Suunnittelu- ja luontiprosessi

Suunnittelun ensimmäiset kokeiluvaiheet pyörivät SQL-tietokantaidean ympärillä. Tiedetään, että kun asiakas käyttää pesukonetta, niin kone soittaa palvelimelle, joka tallentaa soittonumeron XML-tiedostoon. Palvelimena toimiva Windows-tietokone oli tarkoitus saada toimimaan SQL-palvelimena, joka sisältää tietokannan kaikista yrityksen konetyypeistä ja niiden tiedoista. SQL-ideaa kokeiltiin XAMPP Control Panelin kautta toimivalla MySQL-ympäristöllä. Ongelmia kuitenkin alkoi syntyä, kun kahta erillistä järjestelmää yritettiin yhdistää.

24 Pesulan konetyyppien lähettimenä toimii toistaiseksi vielä modifioitu Atmelin piirikortti, jossa on kiinni MC66-GSM-modeemi. Kun konetyyppiä käytetään, se antaa modeemille käskyn soittaa ennalta määrättyyn numeroon. Palvelimena toimiva tietokone ottaa puhelun vastaan ja tallentaa soittajan numeron XML-tiedostona suoraan Windowsin resurssienhallintaan.

Suunnittelun alkuvaiheessa ajateltiin, että soittotieto voidaan kerätä suoraan vastaanottamalla koneiden soitot. Palvelin osasi kuitenkin jo tallentaa soittonumerot resurssienhallintaan, joten tätä prosessia ei kannattanut lähteä ideoimaan uudestaan. Käyttötiedot oli vain saatava resurssienhallinnasta SQL-tietokantaan, mutta automaattista tapaa tiedostojen siirtämiselle SQL:ään ei tuntunut löytyvän. Ongelmaa yritettiin ratkaista käyttämällä Microsoftin CLR-tekniikkaa, mutta sekään ei tuottanut tulosta. CLR ei ollut yhteensopiva MySQL:n kanssa ja ainoa järkevä tapa saada XML-tiedostosta sisältö tietokantaan, oli tehdä se manuaalisesti. Tämä ei kuitenkaan ollut se suunta johon tähdättiin.

Ratkaisun löytäminen oli pitkä prosessi ja sisälsi paljon Googlen hakukoneen käyttöä. Ratkaisu ei kuitenkaan löytynyt Internetistä, vaan ulkopuoliselta henkilöltä joka ei ollut aikaisemmin kuullut 24 Pesula-yrityksestä. Päädyimme jättämään SQL-tietokannan kokonaan ja siirryimme rakentamaan alusta asti omaa Windows-sovellusta, joka mahdollisti samat listausominaisuudet kuin SQL-tietokanta, mutta antoi paljon helpomman lähestymistavan järjestelmän ja sen ominaisuuksien suunnitteluun.

Ensimmäinen ehdotus ohjelmointikieleksi oli APL, joka on Suomessa melko tuntemattomaksi jäänyt ohjelmointikieli. Metropolia Ammattikorkeakoulun oppitunneilta oli kuitenkin jäänyt C# parhaiten muistiin. Tämä antoi heti matalamman kynnyksen siirtyä

sovelluksen suunnitteluun pienten kertausopintojen jälkeen. 24 Pesulan Seurantaohjelmistoa alettiin toteuttaa Microsoftin Visual Studio-työkalulla.

Käyttöliittymän ensimmäisiä rakenteita kehitettiin marraskuun aikana vuonna 2016. Päänäkymään luotiin DataGridView-taulukko, johon oli tarkoitus saada automaattisesti listautumaan valmis lista lokikansion tietojen perusteella. DataGridView on Windowsin Visual Studio-ohjelmiston valmis taulukointiominaisuus, joka nopeuttaa ohjelmoijan työtä niin, että taulukkorakennetta ja sen ominaisuuksia ei tarvitse koodata erikseen.

Lokikansion tietojen saamiseksi ohjelman piti saada tieto, missä kansio sijaitsee. Kansiota ei voitu määrittää ennalta, koska lokikansio ei välttämättä aina sijaitse samassa polussa. Lisättiin ominaisuus, jonka avulla käyttäjä itse valitsee kohdekansion. Pelkkä kohdekansion löytyminen ei kuitenkaan vielä riittänyt tietojen listaamiseen. Suunniteltiin puolipisteillä eritelty CSV-taulukko, johon oli listattu jokainen laitetyyppi tietoineen, joka yrityksestä löytyi. Nyt käyttäjä pystyi valitsemaan lokikansion lisäksi myös tämän tiedoston, jonka avulla ohjelmisto osasi vertailla löytynyttä dataa. Ohjelmisto ei ollut täysin automaattinen, vaan käyttäjän piti käynnistää prosessi erillisellä, ohjelmaan lisätyllä painikkeella, jotta tiedot listautuivat DataGridView-taulukkoon. Ohjelmointiprosessista on lisää tietoa luvussa 3.2.2, jossa käydään läpi 24 Pesulan seurantaohjelmiston päänäkymän koodia ja kerrotaan koodin eri osien toimintaperiaatteista.

Tarkoitus ei ole opettaa ohjelmoimaan C# ohjelmointikielellä. Kielen oppimista varten kannattaa tutustua Ghodrat Moghadampourin 2009 kirjoittamaan teokseen, C#-ohjelmointi. Lisäksi Stackoverflow-verkkosivusto tarjoaa paljon hyödyllistä tietoa, saman ohjelmointikielen parissa kamppailevien henkilöiden kysymyksien ja niihin saatujen vastausten muodossa.

### 3.2.1 Kokeiluvaihe

24 Pesulan seurantaohjelmiston ensimmäisestä versiosta käytettiin nimitystä Kehitysversio. Versioiden numerointi on tärkeää, jotta tiedetään mitä muutoksia kukin versio sisältää. Ohjelmisto käyttää semanttista versiointia. [5.] Alfa-nimitystä käytettiin vasta, kun ohjelmistoa voitiin käyttää sillä varoituksella, että pieniä ongelmia saattoi ilmetä. Ohjelmiston virheettömästä toiminnasta ei alkuvaiheessa ollut vielä takuita, joten päätettiin käyttää Kehitysversio -nimitystä, kunnes päätoiminnan virheettömyys voitiin varmistaa. Kehitysversio-nimitystä käytettiin kaikissa version 0.1-päivityksissä.

Ensimmäisessä kehitysversiossa ei ollut vielä versionumerointia. Idea oli vain saada itse päällistaus toimimaan. Yritykseltä sai tiedot, miltä lokikansiossa oleva XML-tiedosto näyttää sisältä ja tämän perusteella ensimmäisiä kokeiluja varten luotiin kolme XML-tiedostoa. XML-tiedosto sisälsi aina yhden yrityksen laitteen soittonumeron, sekä mieltävaltaisen päivämäärän ja kellonajan.

Listauksen koodaukselliset vihjeet ja esimerkit löytyivät pääasiassa Stackoverflow-verkkosivustolta, jonne ihmiset kirjoittivat ongelmiaan, joita olivat kohdanneet omissa ohjelmointiyrityksissään, ja saivat näihin vastauksia muilta käyttäjiltä. Osa hyödynneistä tiedoista oli peräisin myös Microsoftin Visual Studio-ohjelmistoa opastavasta verkkosivusta.

Kun listaus saatiin onnistuneesti toteutettua omaa testilokikansiota hyödyntäen, oli tarkoitus matkustaa Tampereelle kokeilemaan listauksen toimivuutta virallisten lokitiedostojen kanssa. Matkaa ei kuitenkaan vielä kyetty järjestämään, joten ohjelmiston toimivuutta hiottiin entisestään ja siihen lisättiin valmiuksia tulevaisuuden ominaisuuksia ajatellen. Ohjelmiston pääikkuna koristeltiin 24 Pesulan logoilla ja valikkoon lisättiin ohjekirja aloitustoimenpiteitä varten. Tälle ensimmäiselle viralliselle luomukselle annettiin versionumeroksi 0.1.0 (Kehitysversio). Jo seuraavina kahtena päivänä ohjelmistoa kehitettiin eteenpäin ja ennen Tampereen reissua saatiin valmiiksi versio 0.1.2, johon oli suunniteltu päällistauksen lisäksi apuohjelma, jolla saatiin karsittua valmis lista ajan mukaan.

Tampereen ohjelmistokokeilu osoittautui erittäin tärkeäksi kokemukseksi, joka lopulta näytti tien ohjelmiston tulevaisuudelle. Ohjelmisto ei toiminut paikan päällä virheettömästi. Kun listausprosessi käynnistettiin, nousi tietokoneen prosessorikäyttö sataan prosenttiin eikä listaus näyttänyt etenevän. Listauksesta oli vain noin sadasosa käytöstä listattuna, kun ohjelmisto ei kyennyt enää etenemään. Koodia ei ollut mahdollista muokata paikan päällä, mutta listauksen ongelmaksi selvisivät sellaiset soittonumerot, joita ei ollut CSV-tilukossa olemassa. Tästä viasta on tarkempi selitys luvussa 3.3.1.

Ongelmaa lähdettiin korjaamaan välittömästi Helsinkiin päästyä, kun asiat olivat vielä tuoreessa muistissa. Listausprosessi muokattiin ohittamaan numerot, joita ei ollut olemassa. Lisäksi havaittiin luontitiedoissa ongelmia aina, kun lokikansion tietoja kopioitiin tai siirrettiin kansiota toiseen. Tällöin luontiajat muokkaantuivat aina uusimmaksi ajaksi, mikä väärästi listaustietoja. Tämä ongelma korjattiin niin, että päivämäärä otettiin

jatkossa tiedoston nimestä eikä luontitiedoista. Korjaus oli mahdollinen, koska lokikansion tiedostot oli alun perin määritetty tallentuvan niin, että tiedoston nimeksi määräytyi tiedoston tallennushetkenä ollut päivämäärä ja kellon aika.

Versiosta 0.1.3 tuli tärkeä korjauspäivitys, joka lähetettiin palvelinkoneelle Team-Viewer-ohjelmiston avulla 19.12.2016. Tämän jälkeen listaus onnistui virheettömästi, ja näin virstanpylväs oli viimein saavutettu. Työharjoittelujakso alkoi lähettää loppuaan, mutta ohjelmiston viime hetken menestys vakuutti 24 Pesula Oy:n osakkaan, Nico Nevalan niin, että työsuhde sai jatkaa toistaiseksi. Ohjelmiston numerointi ”hyppäsi” versioon 0.2.0 (Alfa), jossa oli työnantajan yllätykseksi toteutettu pylväsdiagramminäkymä. Uuden näkymän lisäksi valmiille listoille annettiin tallennusmahdollisuus PDF- tai CSV-muotoon ja ulkoasun ilmettä tehostettiin lisäämällä sovellukselle 24 Pesulan logolla varustettu ikoni käynnistyskuvakkeelle. Tarkemmat versiotiedot ja -luettelo löytyy luvusta 3.3.

### 3.2.2 Päänäkymän luonti C#-ohjelmointikielellä

Nyt tarkastellaan ohjelmiston päänäkymässä käytettyä koodia ja syvennyttään tarkemmin siihen, miten koodin eri osat toimivat.

Kuten aikaisemmin mainittiin, C#-kieleen oli helppo siirtyä, koska se oli jäänyt parhaiten muistiin Metropolia Ammattikorkeakoulun oppitunneilta. C# osoittautui hyväksi vaihtoehdoksi myös siksi, että se kuuluu .NET-kehitysympäristöön. .NET-sovelluskehityksen hyvä puoli on, että se palvelee tilanteissa, joissa tietokoneet ja laitteet ovat yhteydessä toisiinsa Internetin kautta. Lisäksi sovelluskehityksen tarjoamat, valmiiksi rakennetut osat helpottavat työskentelyä ja laaja toiminnallisuuksien kirjasto on vapaata lähdekoodia. [6, s. 12.]

24 Pesulan seurantaohjelmiston Alfa-vaiheet eivät kuitenkaan käytä vielä suoraan .NET:in tarjoamaa ominaisuutta linkittää ohjelmiston toiminnot Internetin välityksellä yrityksen koneisiin. Koska linkki pesukoneiden ja palvelimen välillä on jo olemassa, on helpompi keskittyä tiedon keruuseen Windowsin resurssienhallinnasta tässä vaiheessa ohjelmoinnin opiskelua.

Ohjelmiston päänäköymän listaus vaatii, että ohjelma tietää, mistä kansiota lokitiedostot löytyvät. Sivulla 11 on esitetty kuva 1 juuri tästä tilanteesta. Koodin riviltä 55 eteenpäin on *avaaKansioToolStripMenuItem\_Click*-niminen metodi, joka kuuluu *public partial class*-luokkaan [6, s. 33 – 35]. *avaaKansioToolStripMenuItem\_Click* suorittaa metodin *DialogResult*. Rivillä 57 luodaan *DialogResult*-olio ja avataan käyttäjälle dialogi, jonka kautta voidaan valita mikä tahansa kansio. Tämän jälkeen koodi tarkistaa, valitsiko käyttäjä kansion. Jos kansio on valittu, tulee rivin 58 ehtolauseesta tosi, minkä jälkeen kaikki valitusta kansiota löytyneet tiedostot tallennetaan muistiin ja tiedostojen lukumäärä tulostetaan konsoliin. Konsolin näyttämät tiedot on tarkoitettu ohjelmiston toiminnon seuraamista ja ongelmatilanteiden etsimistä varten.

Ohjelmisto varmistaa, että kaikilla löydettyillä tiedostoilla on normaalit käyttöoikeudet ja ilmoittaa käyttäjälle, että tiedostopolku on kytketty. Lopuksi se merkitsee Tiedostovalikon painikkeen eteen, että kyseinen toiminto on toteutettu. Ohjelmisto suorittaa nämä toimenpiteet siitä huolimatta, vaikka tiedostoille ei syystä tai toisesta ole mahdollista antaa tavallisia käyttöoikeuksia. Tämä johtuu siitä, että nämä toimenpiteet eivät ole vertailulauseen takana, kuten kuvasta 1 voidaan päätellä.

Rivillä 76 luodaan *sf*-niminen olio. Olion jälkeen tarvitaan ehto, jolla varmistetaan, ettei tiedostokansio ole tyhjä, koska muuten *sf* antaa virheen. Virhe johtuu siitä, että *sf* olettaa saavansa arvon, mutta jos kansio on tyhjä, ei arvoa ole mahdollista saada. Tällä toiminnolla tallennetaan löydetty tiedostolista metodin ulkopuolelle *savePath*-nimellä, mutta toiminto on parhaillaan turha, koska tiedostoja kutsutaan uudestaan myöhemässä vaiheessa. *savePath* jätetään kuitenkin koodiin siltä varalta, jos sitä tulevaisuudessa tarvitaan. Vertailulause palvelee kuitenkin *MessageBox*-metodia, jonka tarkoitus on kertoa käyttäjälle eteen ponnahtavalla ikkunalla, jos tiedostot on löydetty ja mikä on niiden lukumäärä, tai jos koko tiedostopolku on tyhjä.

```

54 0 references
    string[] savePath { get; set; }
55 1 reference
    public void avaaKansioToolStripMenuItem_Click(object sender, EventArgs e) //Avaa XML kansio, Nappi.
    {
56         DialogResult result = folderBrowserDialog1.ShowDialog();
57         if (result == DialogResult.OK)
58         {
59             string[] files = Directory.GetFiles(folderBrowserDialog1.SelectedPath);
60
61             Console.WriteLine("Tiedostojen lukumäärä on {0}.", files.Length); //laskee tiedostojen lukumäärän. en vielä tiedä miksi.
62
63             var directory = new DirectoryInfo(folderBrowserDialog1.SelectedPath) //varmistetaan että oikeudet tiedostoihin ovat normaalit.
64             { Attributes = FileAttributes.Normal };
65             foreach (var info in directory.GetFilesSystemInfos("*.xml", SearchOption.AllDirectories))
66             {
67                 info.Attributes = FileAttributes.Normal;
68             }
69
70             label1.Text = "Tiedostopolku kytketty"; //informaatio ohjelmiston oikeaan laitaan.
71             avaaKansioToolStripMenuItem.Checked = true;
72             bool XMLauki = avaaKansioToolStripMenuItem.Checked;
73
74             SaveFileDialog sf = new SaveFileDialog();
75
76             if (files.Length != 0)
77             {
78                 sf.FileName = files[0];
79                 string savePath = Path.GetDirectoryName(sf.FileName);
80                 MessageBox.Show("Tiedostot löydetty: " + files.Length.ToString(), "Viesti"); //pop uppi mikä ärsyttävästi ilmoittaa että kyllä se tiedosto ny o siel.
81             }
82             else
83             {
84                 label1.Text = "Tiedostopolku tyhjä";
85             }
86         }
87     }
88 }
89
90
91
92
93
94

```

Kuva 1. Koodi, jolla annetaan käyttäjän valita kohdekansio, jossa lokitiedostot sijaitsevat.

Luvussa 3.2.3 käydään läpi, miten tulosten listaus varsinaisesti toimii, mutta ennen sitä ohjelmisto joutuu tekemään pari toimenpidettä ennen kuin varsinainen listauksen suoritus on mahdollista. Nämä toimenpiteet ovat tiedoston nimen ja sisällön selvitys ja kone-listauksen haku.

Sivun 13 kuvassa 2 näkyy koodi, jolla selvitetään tiedostojen nimet. Luvussa 3.3.1 kerrotaan, miten ohjelmiston versiossa 0.1.3 korjattiin aikaongelmat, jotka ilmenivät, kun tiedostoja siirrettiin kansioista toiseen. Tämän takia tiedoston aikatieto selvitetään tiedoston nimestä Windowsin antamien luontitietojen sijaan.

Kuvasta 2 nähdään heti ensimmäisenä *if*-lauseen jälkeen, miten kaikki tiedostot haetaan kansioista muuttujaan *xmlpolku*, jolloin edellä mainittua *savePathin* tallentamista metodin ulkopuolelle ei enää tarvita. Tiedostot tallennetaan listaksi, jotta niitä päästään selaamaan *foreach*-toistolauseella koodin rivillä 228. Tämä toistolause toistuu niin monta kertaa, kun se löytää tietoja tarkasteltavasta kohteesta. Näin ei tarvitse rajoittaa esimerkiksi *for*-toistolauseeseen, jonka enimmäiskierron määrä on aina se, mikä sille on asetettu. Tietysti tämän enimmäisarvon voi asettaa käyttämällä *Count*-metodia, joka selvittää listan tallennettujen tiedostojen lukumäärän, mutta miksi täyttää ylimääräisiä metodeja, kun on olemassa valmis lause, jota käyttää? [6, s. 114 – 115.]



Edellä mainittu *foreach*-lause sisältää *string*-tyyppisen muuttujan *dir*, koska tarkoitus on käsitellä merkkijonoja. Seuraavaksi *foreach*-lauseen sisälle on merkattu listan nimi *xmlpolku*, josta tieto poimitaan yksitellen ja käsiteltävät tiedot tallennetaan vuoronperään muuttujaan *dir*.

Toistolauseen sisällä tarkistetaan, että kyseinen tiedosto on tyypiltään XML. Tarkistus suoritetaan siten, että ensin selvitetään tiedoston nimessä olevien merkkien lukumäärä ja tallennetaan se muuttujaan *kokPit*. Jos esimerkiksi *dir* saa arvon 24-11-2016\_14-25-28.xml, tulee *kokPit*-arvoksi 23. Koska tiedoston tyyppi on oltava XML, tiedetään että, nimen perässä tulee aina olemaan merkintänä joko .xml tai .XML. Näin voidaan olettaa, että nimen perässä oleva tyyppimerkintä on aina neljän merkin pituinen. Nämä neljä merkkiä vähennetään muuttujasta *kokPit* ja tulos tallennetaan uuteen muuttujaan, jonka nimi on *lopPit*. Kun tiedetään *lopPit* ja *kokPit*, voidaan näiden muuttujien arvoja hyödyntää koodin rivillä 232 olevassa lauseessa, joka paljastaa tiedoston nimen perässä olevat neljä viimeistä merkkiä. Nämä neljä viimeistä merkkiä tarkistetaan rivin 235 ehtolauseessa, jossa on lisäksi oletettu, että tiedoston nimen kokonaispituus on 23.

Jos 24 Pesula Oy päättää jatkossa käyttää eripituisia tiedostonimiä, on 23 merkin oletus kannattavampaa ottaa ehtolauseesta pois ja tehdä muokkauksia tuleviin toimenpiteisiin koodissa. Tällä hetkellä se toimii hyvänä lisävarmistuksena, jotta lokikansioon mahdollisesti joutuvat, väärät tiedostot saadaan tehokkaasti karsittua pois ja näin tiedoston luontiaika on mahdollista hakea tiedoston nimen perusteella.

Jos tiedoston nimi on oikean pituinen ja sen tyyppi on XML, selvitetään seuraavaksi tiedoston luontiaika käyttäen hyödyksi sen nimeä. Tässä toimenpiteessä eritellään omat muuttujat päivälle, kuukaudelle, vuodelle, tunnille, minuutille ja sekunnille. Muuttujien lähteenä hyödynnetään edellä mainittua muuttujaa *dir*, josta tällä kertaa halutaan tietää vain tiedoston nimi ilman tiedostotyyppiä. Pelkkä tiedoston nimi saadaan komenolla *Path.GetFileName(dir)*. Saatu tieto tallennetaan *string*-tyyppiseen muuttujaan *Tiedostonimi*.

Koodin riviltä 245 alkaa päivämäärän päiväarvon haku tiedoston nimestä. Koska päivämäärä on merkattu ensimmäisenä tiedoston nimeen, tulee lauseen ensimmäiseksi arvoksi 0. Jokainen kohta päivämäärässä vuosilukua lukuun ottamatta on aina esitetty kahdella merkillä, joten tällä oletuksella voidaan koodissakin käyttää suoraan vakioarvoisia numeroita. Esimerkiksi päiväarvo alkaa tekstirivin kohdasta 0, ja se on kahden

merkin pituinen. Kuukausi alkaa kohdasta 3, ja se on myös kahden merkin pituinen. Näin jatketaan eteenpäin, kunnes kaikki päivämäärän osat on selvitetty ja ne yhdistetään *string*-tyyppiseen muuttujaan *aikaYhteensä*. Ohjelma liittää selvitettyt tiedot niin, että se lisää jokaisen tiedon väliin joko pisteen, välilyönnin tai kaksoispisteen. Näin lopullinen aikatieto tulee oikeaan muotoon, kun käytetään *DateTime*-metodia koodin myöhemmissä osissa.

```

221 if (avaaKansioToolStripMenuItem.Checked == true)
222 {
223     StreamWriter csvWhole = new StreamWriter(new FileStream(@"\Raporttitiedot\Kokonaisraportti.csv", FileMode.Create, FileAccess.ReadWrite), Encoding.UTF8);
224     csvWhole.WriteLine("Paikka:Kone;Maksupalvelu;Soltto nro:Aika");
225
226     string[] xmlpolku = Directory.GetFiles(folderBrowserDialog1.SelectedPath);
227
228     foreach (string dir in xmlpolku) //haakee jokaisen tiedoston kansioista
229     {
230         int kokPit = dir.Length; int lopPit = kokPit - 4;
231         var tiedostomuoto = dir != null ?
232             dir.Substring(lopPit, dir.Length - lopPit) : null;
233
234         if (Path.GetFileName(dir).Length == 23 && (tiedostomuoto == ".XML" || tiedostomuoto == ".xml"))
235         {
236             //Console.WriteLine(dir); //näyttää mitä löyty
237
238             //jaetaan tiedoston nimi osiin, niin että poimitaan siitä päivämäärän ja kellonajan osat.
239             string Tiedostonimi = Path.GetFileName(dir);
240
241             var tPäivä = Tiedostonimi != null ?
242                 Tiedostonimi.Substring(0, Tiedostonimi.Length >= 0 ? 2 : Tiedostonimi.Length) :
243                 null;
244             var tKuukausi = Tiedostonimi != null ?
245                 Tiedostonimi.Substring(3, Tiedostonimi.Length >= 3 ? 2 : Tiedostonimi.Length) :
246                 null;
247             var tVuosi = Tiedostonimi != null ?
248                 Tiedostonimi.Substring(6, Tiedostonimi.Length >= 6 ? 4 : Tiedostonimi.Length) :
249                 null;
250             var tTunnit = Tiedostonimi != null ?
251                 Tiedostonimi.Substring(11, Tiedostonimi.Length >= 11 ? 2 : Tiedostonimi.Length) :
252                 null;
253             var tMinuutit = Tiedostonimi != null ?
254                 Tiedostonimi.Substring(14, Tiedostonimi.Length >= 14 ? 2 : Tiedostonimi.Length) :
255                 null;
256             var tSekunnit = Tiedostonimi != null ?
257                 Tiedostonimi.Substring(17, Tiedostonimi.Length >= 17 ? 2 : Tiedostonimi.Length) :
258                 null;
259
260             string aikaYhteensä = tPäivä + "." + tKuukausi + "." + tVuosi + " " + tTunnit + ":" + tMinuutit + ":" + tSekunnit;
261             DateTime created = Convert.ToDateTime(aikaYhteensä);
262
263
264
265
266

```

Kuva 2. Koodi, jolla selvitetään tiedostojen nimet.

Kuten aikaisemmin mainittiin, jos tiedoston nimen pituus muuttuu, pitää myös ehtolauseesta poistaa vastaava ehto, jolla oletetaan tiedoston nimen pituuden olevan vakio. Jos näin käy, vaarantuu myös äsken esitetty menetelmä hakea päivämäärä ja kellonai- ka tiedoston nimestä ja siten myös tämä menetelmä joudutaan muokkaamaan. Pa- hemmassa tapauksessa tiedoston aikatiedon selvitykselle joudutaan keksimään tapa, joka poikkeaa täysin esitetystä. Ei ole kuitenkaan mitään syytä tiedossa, miksi näin tulisi käymään, mutta mahdolliset tilannemuutokset, tulee kuitenkin aina pitää mielessä ohjelmistoa luodessa. Yksi tilannemuutos saattaa kuitenkin olla mahdollinen, kun koko yrityksen koneiden seurantamenetelmä uudistetaan [7].

Viimeisenä ennen tulosten listausta on haettava kaikki konelistauksen tiedot, joita käy- tetään listausprosessissa soittonumeroiden vertailussa. Tämä tilanne on esitetty sivulla 14 olevassa kuvassa 3, josta myös selviää rivinumeroiden perusteella, että kyseinen koodin osa on kirjoitettu ennen luontitietojen selvittämistä. Ohjelmiston vanhemmissa versioissa konelistaus haettiin vasta luontitietojen selvittämisen jälkeen, mutta tämä

pakotti kirjoittamaan koodin siten, että konelistaus haettiin jokaisen soittonumeron selvittämistä varten uudestaan. Ylimääräinen varmistelu loi runsaasti raskasta laskentatietoa, koska selvitettävää tiedostoa oli tuhansia ja jokaista selvitettävää numeroa kohden ohjelman piti käydä läpi yli sata soittonumeroa sisältävä taulukko täsmäävän soittonumeron löytämiseksi.

```

197 //Luodaan lista 24pesulan konesta
198 // Get the file's text.
199 String values = File.ReadAllText(@"24Koneet.csv", Encoding.GetEncoding("iso-8859-1"));
200
201 // Split into lines.
202 values = values.Replace('\n', '\r');
203 string[] lines = values.Split(new char[] { '\r' }, StringSplitOptions.RemoveEmptyEntries);
204
205 // See how many rows and columns there are.
206 int num_rows = lines.Length;
207 int num_cols = lines[0].Split(';').Length;
208
209 string[,] csvresult = new string[num_rows, num_cols];
210
211 // Load the array.
212 for (int r = 0; r < num_rows; r++)
213 {
214     string[] line_r = lines[r].Split(';');
215     for (int c = 0; c < num_cols; c++)
216     {
217         csvresult[r, c] = line_r[c];
218     }
219 }
220

```

Kuva 3. Koodi, jolla haetaan konelistauksen tiedot.

Kaikki 24 Pesulan konetyypit on kirjoitettu *24Koneet.csv*-nimiseen CSV-tiedostoon. CSV-tiedoston ulkoasu on esitetty liitteessä 2, kun se on avattu ohjelmalla Notepad++. Tämä tiedosto on alun perin generoitu Microsoft Excelin generaattorilla. Soittonumerot on muokattu siten, että jokaisen nollan sijaan käytetään Suomen suuntanumeroa +358, kuten ne on tallennettu myös lokikansion XML-tiedostoihin. Näin soittonumeroiden vertaileminen saadaan mahdolliseksi ilman ohjelmistollisia lisätoimenpiteitä.

Koodin rivillä 199 kirjoitetaan koko CSV-tiedoston sisältö muuttujaan *values*, joka on tyyppiltään *string*. Koska CSV-tiedostoa voidaan lukea kuten tekstitiedostoa, käytetään koodissa *File.ReadAllText*-metodia. Metodien sulkujen sisällä määritetään ensin tiedoston sijainti ja sen nimi lainausmerkkien sisälle. Koska tiedosto on ohjelmiston juuripolussa, käytetään ennen tiedoston nimeä merkintää `.\` [8.] Tämän jälkeen koodataan tiedoston teksti tukemaan standardia, ISO 8859-1. ISO 8859-1 on latinalaisten aakkosten standardin ensimmäinen osa, joka sisältää kaikkien tavallisten ASCII-merkkien lisäksi useita erikoismerkkejä. ISO 8859-1 standardia käytetään oletuksena HTML-kielen versiossa 4.01 [9]. Tämä standardi mahdollistaa muun muassa Å-, Ä- ja Ö-kirjainten hyödyntämisen, joita Pohjoismaissa käytetään.

Kun koko taulukko on tallennettu muuttujaan *values*, pitää merkkijono jakaa riveihin. Tämä aloitetaan määrittämällä heksadesimaaleihin perustuva ohjausmerkkijono. Jakaminen suoritetaan *Replace*-metodilla rivin 202 mukaisesti. Kun tieto on tallennettu *values*-muuttujaan, määräytyy ohjausmerkkijono kymmenen desimaalin mukaisesti, jota kutsutaan *Line Feediksi*. *Line Feed* -merkki on `\n` ja sillä kerrotaan tavallisesti tietokoneelle uudesta rivistä. *Line Feed* vaihdetaan *Replace*-metodilla `\r` merkkiin, joka tarkoittaa *Carriage Returnia*. *Carriage Return* on 13 desimaalia sisältävä ohjausmerkkijono, jolla viitataan suoraan rivinvaihtoon. [10.] Tämä saa aikaan sen, että jokaisen rivin väliin jää tyhjä rivi. Tämä tyhjä rivi poistetaan koodin rivillä 203, jossa luodaan *lines*-niminen lista. Tämä lista sisältää kaikki konetyypin tiedot yhdellä rivillä, mutta jokainen sarake täytyy vielä erotella riviltä.

Kuten aikaisemmin opittiin, CSV-tiedostossa jokainen sarake on erotettu puolipisteellä, joten sarakkeet voidaan kertoa myös tietokoneelle tätä tietoa hyödyntäen. Aloitetaan selvittämällä rivien ja sarakkeiden lukumäärät kokonaislukumuuttujilla (int) *num\_rows* ja *num\_cols* [6, s. 26 – 27]. *Length*-metodi kertoo merkkien lukumäärän määrätyssä instanssissa, joka tässä tapauksessa on edellä mainittu *lines*-taulukko. *lines.Length* kertoo rivien lukumäärän. Sarakkeiden lukumäärän selvittämiseksi tarkastellaan ainoastaan ylintä *num\_rows* riviä, joka määritetään koodin rivillä 207, *lines*-nimen perässä, hakasulkeiden sisällä olevalla nollalla. *Split*-metodilla jaetaan rajattu merkkijono alimerkkijonoihin, jotka tässä tapauksessa erotellaan puolipisteellä, kuten voidaan huomata kuvasta 3 [11].

Nyt tiedetään rivien ja sarakkeiden lukumäärä. Luodaan uusi olio nimeltä *csvresult* koodin rivillä 209. Olion tyyppi on *string* ja siihen on esilistattu tyhjät paikat niin, että jokaisella rivillä on aina rakennettavan taulukon rivin ja -sarakkeen numero. Rivi ja sarake on erotettu pilkulla. Viimeinen tehtävä on listata tiedot oikeille riveille ja tähän käytetään kahta *for*-toistolauseetta. Ensimmäinen *for*-toistolause käsittelee rivejä. Ensimmäisen toistolauseen alkuarvona käytetään muuttujaa *r* (row), joka on alustettu nollaksi. Toistolause kasvattaa *r*-muuttujan arvoa yhdellä joka kierroksella, kunnes se ei ole enää pienempi kuin *num\_rows*-muuttujaan tallennettu lukumäärä. Jokaisella kierroksella luodaan yksi valmis rivi, jonka sarakkeet ovat eritelty *line\_r*-muuttujaan. Sarakkeiden erottelemiseksi käytetään *Split*-metodia, kuten aikaisemmin esitettiin, mutta tällä kertaa sitä käytetään erottamaan jokainen sana puolipisteen kohdalta.

Ajatellaan esimerkiksi niin, että ollaan ensimmäisen toistolauseen ensimmäisellä kierroksella, jolloin *r*-muuttujan arvo on 0. Kuten liitteestä 2 huomataan, rivi 0 näyttää seuraavalta: *Paikka;Kone;Maksupalvelu;Soitto nro*. Tämä edellä esitetty rivi tallentuu muuttujaan *line\_r* siten, että muuttujan rivillä 0 on arvona *Paikka*, rivillä 1: *Kone*, rivillä 2: *Maksupalvelu* ja rivillä 3: *Soitto nro*.

Kun *line\_r* on tallentunut, siirrytään seuraavaan toistolauseeseen, jossa käsitellään sarakkeita. Sisemmän toistolauseen alkuarvona käytetään muuttujaa *c* (Column), joka on alustettu nollassi. Idea on täysin sama, kuin edellisen toistolauseen tilanteessa, mutta nyt sisempi toistolause jää toistamaan itseään, kunnes *c*-muuttujan arvo ei ole enää yhtä suuri kuin muuttujan *num\_cols*-arvo. Tämän toistolauseen sisällä, koodin rivillä 217 tallennetaan tietoja muuttujaan *csvresult*. Hakasulkeiden sisällä on ensin *r* ja toisena *c*. Merkit viittaavat siihen, että arvo, joka tallennetaan *csvresultiin*, tulee aina sille riville, joka kullakin toistolauseiden kierroksella vastaa muuttujien *r* ja *c* arvoja. Ensimmäisellä kierroksella kummankin muuttujan arvo on 0, ja koska tieto haetaan *line\_r*-muuttujan kohdasta, joka on määritetty muuttujalla *c*, tulee *csvresultin* listan ensimmäiseen kohtaan arvoksi: *Paikka*. Tämän jälkeen jatketaan edelleen sisemmän toistolauseen suorittamista, jolloin *c:n* arvoksi tulee 1. Uuden *c*-muuttujan arvon myötä tallennetaan *csvresultin* listan seuraavalle riville *Kone*. Kun kaikki neljä arvoa on tallennettu, poistutaan sisemmän toistolauseen ulkopuolelle.

Sisemmän toistolauseen ulkopuolella on odottamassa ensimmäisenä esitetty toistolause, jonka ehto toistolauseesta poistumiselle ei ole vielä täyttynyt. Ensimmäinen toistolause määrittää *r*-muuttujan arvoksi 1 ja tallentaa *line\_r*-muuttujaan uuden rivin, jonka jälkeen palataan takaisin sisempään toistolauseeseen. Tätä jatkuu niin kauan, kunnes *r*-muuttujan arvo ei ole enää pienempi kuin *num\_rows*, jolloin myös viimeinen rivi on tallennettu *csvresultiin*. Toistolauseissa käytetään pienempi kuin -merkkiä sen takia, koska aikaisemmin esitetty *Length*-metodi kertoo lukumäärän, mutta listat alkavat aina arvosta 0, jolloin korkein arvo jää yhden arvon pienemmäksi kuin *Length*-metodin antama arvo.

### 3.2.3 Tulosten listaus

Nyt ohjelmisto on hakenut ennalta määritetystä kohteesta konelistauksen tiedot ja tallentanut ne muistiin, jossa jokainen tieto on nopeasti haettavissa. Lisäksi ohjelmisto

tietää, mistä kohdekansioista löytyvät lokitiedostot, ja niistä kaikista on selvitetty aikatie-  
to tiedoston nimen perusteella. Näin päästään viimein suorittamaan itse listaus.

Listaus aloitetaan *StreamReader*-metodilla, jolla jokainen lokikansion XML-tiedosto luetaan. *StreamReader* lukee tiedostot käyttämällä samaa *dir*-muuttujaa, joka esiteltiin sivulla 12. Tilanteen rakennetta selkeyttääksemme, muistutetaan että itse listausprosessi on saman *foreach*-toistolauseen sisällä, kuin lokitiedostojen nimien selvitys. Toisin sanoen, jokainen XML-tiedosto prosessoidaan yksitellen. Kun ensimmäisestä tiedostosta on selvitetty aikatie- to tiedoston nimestä, jatkaa se listausprosessiin, jossa kyseisen tiedoston sisältö luetaan.

Kuva 4 sivulla 18 esittää koko listausprosessin koodin, joka suoritetaan edellä mainittujen toimenpiteiden jälkeen. *StreamReaderilla* luodaan olio nimeltä *reader*. Koodin rivillä 279 oliosta luetaan rivi, joka tallennetaan muuttujaan *l*. *for*-toistolause on jäänne ajatuksesta, jolla voi valita tekstitiedostosta tietyn rivin siten, että jokainen kierros toistolauseessa suorittaa *reader.ReadLine()*-metodin uudestaan. Kun metodi suoritetaan samaan tiedostoon uudemman kerran, lukee se automaattisesti seuraavan rivin [6, s. 352–355]. *for*-toistolauseen voisi ottaa kokonaan pois koska tällä hetkellä tiedostosta on tarkoitus lukea ainoastaan ylin rivi, mutta jos tiedoston sisällön rakenne tulee tulevaisuuden uudistusten yhteydessä muuttumaan, voi toistolauseen jättämisestä olla hyötyä (kts. luku 3.3.3). 24 Pesulan Seurantaohjelmisto voisi tulevaisuudessa tarjota käyttäjälle erillisen asetusikkunan, jossa käyttäjä voi muun muassa määrittää, miten lokitiedostoja luetaan.

Seuraavaksi esitetään asia, joka tämän projektin alkutaipaleilla opittiin kantapään kautta. Kun ohjelmoidaan useita rivejä koodia, ei aina ole mahdollista tietää, onnistuuko koodin kääntäminen täysin mutkitta. Tämä projekti opetti sen, että koodi saattaa kään-  
tyä ensimmäisillä yrityksillä hyvin, mutta kun käsitellään tuhansia tiedostoja, jotka ovat lähes aina erilaisia toisiinsa verrattuna, kasvaa todennäköisyys ennalta arvaamatto-  
maan virheeseen. Yksi esimerkki tällaisesta virheestä on esitetty sivuilla 8 ja luvussa 3.3.1, joissa kerrotaan ohjelmiston virheellisestä koodin käännöstä Tampereen ohjel-  
mistokokeilussa. Onneksi C#-kielessä on olemassa poikkeusten käsittely.

Poikkeusten käsittelyn avulla ohjelmisto ei kaadu kesken ajon tai anna käyttäjälle vir-  
hesanomaa, jota voi olla vaikea ymmärtää, jos koodissa on tapahtunut jokin poikkeus. Tämä turvatoiminto aloitetaan *try*-lohkolla, jonka sisälle koko listausprosessi kirjoite-

taan. Poikkeusta käsittelevä osa tulee heti *try*-lohkon jälkeen ja se merkitään sanalla *catch*. Jos poikkeus tapahtuu, *try*-lohkon sisällä oleva koodin kääntö keskeytyy ja suoriutus siirtyy *catch*-lohkoon. [6, s. 370 – 371.] *catch*-lohkon sisälle on kirjoitettu toiminto, joka vaihtaa DataGridView-taulukon neljännen sarakkeen arvoksi arvon *null countj*-muuttujan määrittelevällä rivillä. Rivin arvon nollaus on turhaksi jäänyt poikkeustoiminto vanhasta listausmenetelmästä, jossa listaus suoritettiin erillisenä toimintona vasta tiedoston aikatieiden selvittämisen jälkeen. *Null*-arvo on käytännössä sama, kuin tyhjä, jota käytetään yleisesti *string*-tyypeissä. Esimerkiksi numeerisille tyypeille vastaava arvo on 0. [6, s. 143–144.]

```

274 using (StreamReader reader = new StreamReader(dir)) //valitsee tiedostosta rivin...
275 {
276     string l = null;
277     for (int i = 0; i < 1; i++)
278     {
279         l = reader.ReadLine();
280         if (l == null) break;
281     }
282
283     try
284     {
285
286
287         var puhnro = l != null ?
288             l.Substring(8, l.Length >= 8 ? (l.Count() - 18) : l.Length) :
289             null; // ...ja ottaa täältä riviltä puhelinnumeron
290
291         //Listataan tulokset
292         int rivi = 0;
293         int sarake = 0;
294         foreach (var koneet in csvresult)
295         {
296             if (puhnro == koneet)
297             {
298                 this.dataGridView1.Rows.Add();
299                 //Console.WriteLine(puhnro);
300
301                 this.dataGridView1.Rows[countj].Cells[0].Value = csvresult[rivi, 0];
302                 this.dataGridView1.Rows[countj].Cells[1].Value = csvresult[rivi, 1];
303                 this.dataGridView1.Rows[countj].Cells[2].Value = csvresult[rivi, 2];
304                 this.dataGridView1.Rows[countj].Cells[3].Value = puhnro;
305                 this.dataGridView1.Rows[countj].Cells[4].Value = created;
306                 countj = countj + 1;
307
308                 //Tallentaa datagridin csv tiedostoksi
309                 csvwhole.WriteLine(csvresult[rivi, 0] + ";" + csvresult[rivi, 1] + ";" + csvresult[rivi, 2] + ";" + puhnro + ";" + created);
310                 label5.Text = "";
311
312                 break;
313             }
314             if (sarake == 3)
315             {
316                 rivi = rivi + 1;
317                 sarake = -1;
318             }
319             sarake = sarake + 1;
320         }
321     }
322     catch (Exception)
323     {
324         this.dataGridView1.Rows[countj].Cells[4].Value = null;
325     }
326 }
327
328
329

```

Kuva 4. Koodi, jolla päällistaus suoritetaan.

Koodin riviltä 287 eteenpäin päästään viimein seuraamaan, miten ensimmäinen lokitieto tallennetaan DataGridView-taulukkuun, jolloin se näkyy käyttäjälle visuaalisesti. Ohjelmisto tietää nyt rivin, jossa soittonumero sijaitsee, mutta *l*-muuttujaan tallentunut rivi näyttää seuraavanlaiselta: +CLIP: "+3584610009998",145,,,0. Tarkoitus on poimia koko riviltä ainoastaan lainausmerkkien sisällä oleva tieto. Toiminnon voisi normaalisti suorittaa toistolauseella, jossa jokainen merkki käydään läpi ja lainausmerkkien avulla määritetään soittonumeron sijainti, mutta koska lainausmerkkejä käytetään C#-kielessä ilmaisemaan *string*-tyyppistä tekstiä koodissa, on lainausmerkkiä nykyisellä tietotasolla vaikea soveltaa kiinnepisteenä. Ratkaisuksi käytetään samaa ideaa, kuin sivuilla 11–14

esitettyssä menetelmässä, jossa poimitaan päivämäärän ja kellonajan osat tiedoston nimestä. Muuttujan nimi on *puhnro*, jonka tyyppi on *var*. *Var* on toiminnaltaan samanlainen, kuin *string*-tyyppi, mutta se on implisiittinen, tarkoittaen että *var*-tyyppiä voidaan käyttää esimerkiksi tilanteissa, joissa lopputuloksen tyyppi ei olion luonnin yhteydessä ole vielä tiedossa [12]. *puhnro*-muuttujaan tallennetaan muuttuja *l*, ja ehtona on, ettei muuttuja *l* saa antaa arvoa: *null*. Koodin seuraavalla rivillä käytetään *Substring*-metodia, jolla saadaan hajotettua koko *l*-muuttuja osiin ja sen merkkejä päästään tarkastelemaan numeerisia menetelmiä ja laskusääntöjä hyödyntäen [13].

Tiedetään, että jokainen XML-tiedosto on soittonumeroa lukuun ottamatta täysin identtinen. Näin voimme olettaa, että ennen soittonumeron +-merkkiä, rivin alku sisältää yhteensä kahdeksan merkkiä, kun välilyönti lasketaan mukaan. Täten sulkumerkkien sisälle ensimmäiseksi sijoitetaan arvoksi 8, joka on haluttu aloituskohta. Seuraavaksi selvitetään *Length*-metodilla *l*-muuttujan merkkien kokonaislukumäärä, jonka on tarkoitus olla enemmän tai yhtä paljon, kuin edellä määrittämämme aloituskohta. Jos tämä ehto ei pidä paikkaansa, ilmenee poikkeus ja koodin kääntäjä hyppää suoraan *catch*-lohkoon. Lopuksi selvitetään soittonumeron viimeisen numeron sijainti. Kuten aloituskohtaa selvitetäessä, voimme olettaa, että myös soittonumeron jälkeen merkkejä on aina kymmenen. Koska saamme *Length*-metodilla selville *l*-muuttujan merkkien lukumäärän, voimme tästä lukumäärästä vähentää soittonumeron aloituskohdan arvon ja soittonumeron jälkeen esiintyvien merkkien summan, joka on 18.

Listaus aloitetaan luomalla kaksi laskuria, joiden alkuarvo on 0. Ensimmäinen laskuri laskee rivien lukumäärää ja toinen sarakkeiden lukumäärää. Laskurikäytäntö osoittautui hyväksi ratkaisuksi, kun käytetään *foreach*-toistolauseetta, koska arvoaan muuttavia muuttujia ei toistolauseessa ole. Koodin rivillä 294 käytetään kyseistä toistolauseetta kutsumaan kaikki *scvresult*-muuttujaan tallennetut tiedot. Tiedot tallentuvat yksitellen muuttujaan: *koneet*, jota verrataan ehtolauseessa tiettyyn soittonumeroon, joka on tallentunut muuttujaan *puhnro*. Ideana on, että *foreach*-toistolauseeseen tullaan sisälle aina yhden soittonumeron kanssa, jota verrataan *csvresultin* sisältämien konelistaustietojen kanssa. Toistolause lukee jokaisen tiedon, mutta ainoastaan soittonumerot voivat vastata toisiaan, jolloin ehtolauseen arvoksi tulee tosi.

Ensimmäisellä kierroksella *koneet*-muuttuja saa arvoksi: Paikka. Tämä ei oletettavasti pidä paikkaansa ensimmäisessä ehtolauseessa, joten koodin kääntö hyppää ehtolauseen yli, jolloin vastaan tulee toinen ehtolause. Toisen ehtolauseen ideana on kysyä,



onko *sarake*-muuttujan arvo 3. Ensimmäisellä kierroksella tämäkään ei pidä paikkaansa, joten koodin kääntö hyppää ehtolauseen yli ja kasvattaa *sarake*-muuttujaa yhdellä. Neljännellä kierroksella *koneet*-muuttujan arvoksi tulee: Soitto nro, ja sarakkeen arvo on 3. Yksikään tämän rivin tiedoista ei vastannut soittonumeroehdon kanssa, mutta neljännellä kierroksella *sarake*-muuttujan arvo pitää jälkimmäisessä ehtolauseessa paikkaansa, jolloin *rivi*-muuttujaa kasvatetaan yhdellä ja *sarake*-muuttuja saa negatiivisen arvon: -1. Negatiivinen arvo annetaan sen takia, koska ehtolauseen jälkeen *sarake*-muuttujaan lisätään joka tapauksessa luku yksi, jolloin arvoksi saadaan jälleen 0 ja laskuri toimii oikein.

Jos *puhnro*-muuttuja useiden kierrosten jälkeen vastaa viimein *koneet*-muuttujaa, *foreach*-toistolauseen ensimmäisessä ehtolauseessa, kirjoitetaan ensimmäinen rivi *DataGridView*-näkyymään seuraavasti. *DataGridView*-näkymä tyhjennetään aina ensimmäisenä Päivitä-painiketta painettaessa, joten koko taulukko on alussa tyhjä. Uusi tyhjä rivi luodaan aina sitä mukaa, kun sitä tarvitaan. Tämä tapahtuu koodin rivillä 298 *Rows.Add*-metodilla. Rivin luonnin jälkeen edellä esitetyn laskurimenetelmän kertomat tiedot voidaan ottaa hyötykäyttöön. Luokka kutsutaan aina ensin tavallisten sääntöjen mukaisesti, jonka jälkeen kerrotaan rivi, jolle tieto tallennetaan. Rivin arvon kertoo *countj*-muuttuja, joka alustetaan nolaksi aina Päivitä-painiketta painettaessa ja siihen lisätään luku yksi vain, jos soittonumerot vastaavat toisiaan. Näin lasketaan luodut rivit.

Ensimmäisellä kierroksella *countj*-muuttujan arvo on 0. Kun tiedetään rivi, kerrotaan *sarake* pelkällä numerolla. Tähän ei tarvita erillistä muuttujaa, koska sarakkeet ovat kaikki ennalta tiedossa ja voidaan siksi merkitä suoraan oletetulla numerolla. Viimeisenä kirjoitetaan *Value*-muuttujan sisältö, jolla viitataan kyseisen rivin sarakkeen sisältöön. Tämän sisällöksi määritetään *scvresult*-listan kohta, jonka rivitieto vastaa edellä esitettyä laskurimenetelmän tulosta sekä saraketietoa, joka tässä tapauksessa on sama, kuin *DataGridView*-näkymän *sarake*. Samaa ideaa käytetään jokaiseen sarakkeeseen siten, että ainoastaan sarakenumero määrää oikeat tiedot oikeaan paikkaan rivillä. Soittonumero ja aikatieto kirjoitetaan suoraan kuvan 4 mukaisesti muuttujilla. [14.]

Kun ensimmäinen rivi on kirjoitettu, annetaan *countj*-muuttujalle uusi arvo. Ajoitus on tässä kohdassa tärkeää. Kuten aikaisemmin opimme, listaus alkaa arvolla 0. Ohjelmalle on kerrottava, että olemme rivillä 0, jolloin arvo muutetaan vasta listaan tallentamisen jälkeen. Samaa muuttujaa voidaan hyödyntää merkitsemällä 24 Pesulan seuranta-

ohjelmiston päänäköymän yläpalkkiin numeron, joka ilmoittaa käyttäjälle tallennettujen rivien lukumäärän.

Luvussa 3.4.1 näytetään, että ohjelmisto tallentaa kokonaisraportti.csv-tiedoston juuripolkuun. Tämä toimenpide suoritetaan juuri tässä vaiheessa koodin rivillä 311. Sivulla 13 olevasta kuvasta 2 nähdään, että koodin rivillä 223 luodaan *csvwhole*-niminen olio *StreamWriter*-luokalla. Tiedoston tallennuskansioksi määritetään ohjelmiston juuripolussa oleva Raporttitiedostot-kansio, ja siihen tallennetaan valmiiksi ensimmäinen rivi, joka sisältää sarakkeiden otsikot. Kun palataan takaisin sivulla 18 olevaan kuvaan 4, huomaamme että koodin rivillä 311 käytetään samaa *WriteLine*-metodia uuden rivin lisäämiseksi, kuin sivun 11 kuvassa 1. Sulkumerkkien sisälle on kirjoitettu jokainen samalle riville haluttu tieto. Tiedon lähteenä käytetään samoja muuttujia, kuin *DataGridView*-näköymän sarakkeita tallennettaessa. Jokaisen tiedon väliin lisätään erikseen puolipisteet säilyttämään CSV-tiedostolle vaadittu sisällöllinen ulkoasu. CSV-tiedostosta on lisää tietoa lyhenteissä ja liitteessä 2.

CSV-tiedoston tallennustoiminnon jälkeen määritetään *label5.Text* tyhjäksi, joka on Microsoft Visual Studiosissa käytettävä tekstiominaisuus. Tämä teksti on asetettu keskele *DataGridView*-näköymää ja sen arvoksi on Päivitä-painiketta painettaessa määritetty teksti: Ei tuloksia! Tämä teksti ei tule päivityksen alussa tai sen aikana näkyviin, koska päänäköymän ulkoasu päivitetään vasta koko päivitysprosessin jälkeen. Jos tuloksia ei löydy, Ei tuloksia! -teksti tulee näkyviin, mutta jos yksikin tulos löytyy, määritetään tämä teksti tyhjäksi. [15.]

Ohjelmiston vanhemmissa versioissa käytettiin tekstin tyhjentämisen jälkeen *rivilaskuri*-muuttujaa, jonka tarkoitus oli laskea käyttäjälle rivien lukumäärä. Koska *countj*-muuttuja laskee täysin saman, voitiin *rivilaskuri*-muuttuja poistaa. Ehtolauseen lopuksi suoritetaan *break*-lause, jolla hypätään ulos *foreach*-toistolauseesta [6, s. 116]. Toistolauseetta on turha jatkaa, koska haluttu tieto etsityn soittonumeron mukaan on tallennettu ja näin päästään välittömästi hakemaan seuraava mahdollinen lokitiedosto vertailuun (Kuva 2). Kun uusia lokitietoja ei enää ole, suljetaan: *scvwhole*.

### 3.3 Versiovaiheet

Tässä osiossa käydään läpi tarkemmin 24 Pesulan seurantaohjelmiston versioita syventymällä versiolokin tietoihin. Ohjelmisto on kokenut kokeiluvaiheen jälkeen kaksi vaihetta: Kehitysversiot ja Alfa-vaihe. Luvussa 3.3.2 esiintyy kuvia, joiden soitt numerot on osittain sensuroitu luottamuksellisista syistä.

Ohjelmiston vaiheista on pidetty tarkkaa kirjaa siitä asti kun ensimmäinen kehitysversio julkaistiin yrityksen sisällä. Ensimmäistä kehitysversiota edeltävät kokeilut perustuivat työharjoittelujakson aikana tapahtuneeseen oppimisvaiheeseen, jossa C#-ohjelmointikielen opiskelu oli pääasiallisena aiheena. Ohjelmiston valmiiksi viemiselle ei annettu odotteita, vaan asiaa lähdettiin tutkimaan siltä kannalta, mikä oli mahdollista toteuttaa.

Pelkän käyttöliittymän suunnittelu ja toteutus vaati kuukauden työn vuoden 2016 marraskuussa, jolloin lähinnä tutustuttiin Microsoft Visual Studio -ohjelmiston ympäristöön ja kokeiltiin erilaisia toimintoja ja niiden käyttäytymistä.

Tie on ollut pitkä ja tunteja on kertynyt lukematon määrä, mutta lopulta se mitä lähdettiin hakemaan, saavutettiin ja ohjelmisto on parantunut versio versiolta siinä missä taidotkin.

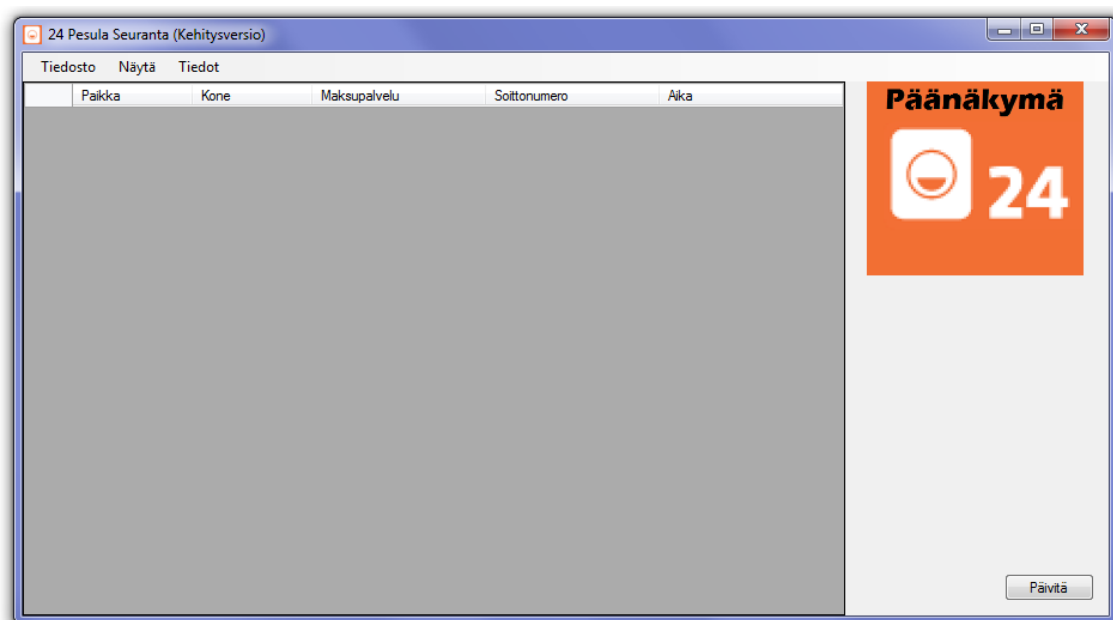
Seuraavalla sivulla on esitetty 24.4.2017 päivitetty taulukko, josta ilmenee 24 Pesulan seurantaohjelmiston versioiden nimitys, numero ja julkaisuajanjakso. Tätä taulukkoa kannattaa silmäillä lukuja 3.3.1 ja 3.3.2 luettaessa. Versiomuutosten yksityiskohdat esitetään näissä luvuissa Alfa-versioon 0.2.5 asti.

Taulukko 1. 24 Pesulan seurantaohjelmiston versioloki.

Version nimitys	Versionumero	Julkaisupäivä
Kehitysversio	0.1.0	13.12.2016
Kehitysversio	0.1.1	14.12.2016
Kehitysversio	0.1.2	15.12.2016
Kehitysversio	0.1.3	19.12.2016
Alfa	0.2.0	23.12.2016
Alfa	0.2.1	03.01.2017
Alfa	0.2.2	10.01.2017
Alfa	0.2.3	17.01.2017
Alfa	0.2.4	07.02.2017
Alfa	0.2.5	06.03.2017
Alfa	0.2.6	14.03.2017
Alfa	0.2.7	06.04.2017
Alfa	0.2.8	21.04.2017

### 3.3.1 Kehitysversiot

24 Pesulan seurantaohjelmiston versio 0.1.0 oli ensimmäinen yrityksen sisällä julkaistu lopputulos, joka saavutettiin marraskuussa 2016 tehdyn oppimisvaiheen seurauksena. Käyttöliittymän päänäkökulma oli huolellisesti suunniteltu selkeyttä ja helppoutta vaalien, mikä palveli suuresti myöhemmässä vaiheessa. Tulevat versiot ovatkin lähinnä kokeineet vain tyylillisiä jatkomuokkauksia. Version kaikki tarvittavat ominaisuudet löytyivät ohjelmiston yläreunalla sijaitsevasta työkalupalkista (Kuva 5). Ohjelmiston toimivuus vaati sen, että käyttäjä valitsi tiedostopolusta XML -lojikansion sekä CSV-tiedoston, joiden perusteella ohjelma osasi listata tiedot päänäkökulman DataGridView-tilaan. Tähän versioon lisättiin myös tulostinominaisuus ja karsintamahdollisuus aloitus- ja lopetuspäivämäärien mukaan, mutta näitä ominaisuuksia ei vielä kehitetty valmiiksi asti. Seuraavalla sivulla on kuva 5 24 Pesulan seurantaohjelmiston version 0.1.0 päänäkökulmasta.

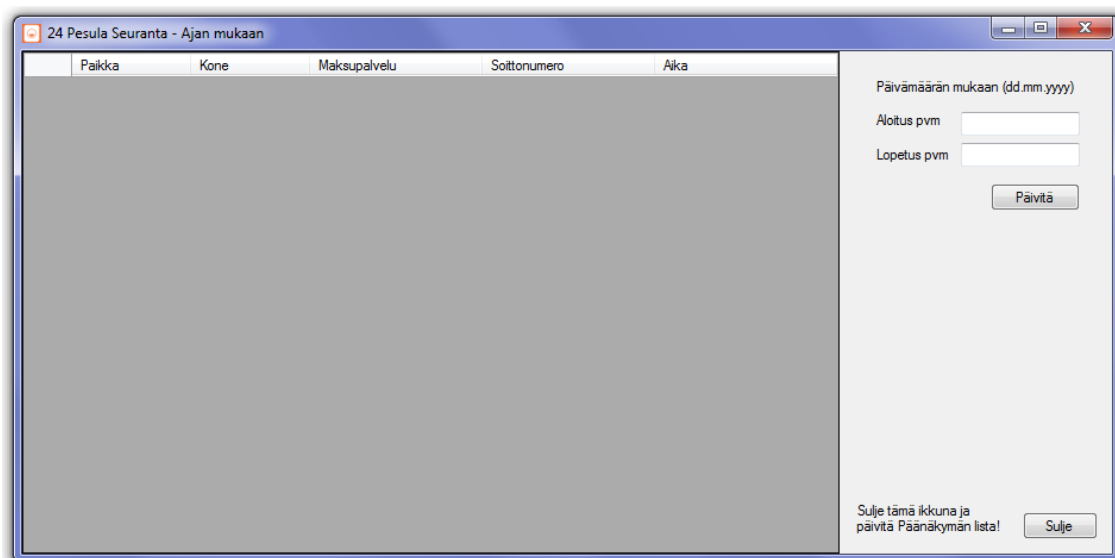


Kuva 5. 24 Pesula Seuranta. Ohjelmistoversio 0.1.0 (Kehitysversio), päänäkymä.

Alkuvaiheissa ohjelmointi muodostui mielekkääksi, kun tuloksia syntyi onnistumisien ja taidon kehittymisen myötä. Tämä johti siihen, että seuraavaa versiota julkaistiin välittömästi sitä mukaa kun parannettavia kohtia ja uusia ideoita saatiin toimimaan. Versio 0.1.1 julkaistiin jo heti seuraavana päivänä 14.12.2016. Siihen lisättiin vain painike Aikaikkunan sulkemiseksi ja lisättiin tieto, milloin lista oli viimeksi päivitetty.

Versio 0.1.2 julkaistiin 15.12.2016 samalla, kun huomattiin, että testilokikansiossa olevien XML-tiedostojen sisällöt oli luotu virheellisesti ja tämä saattoi estää ohjelmiston toiminnan varsinaisessa kokeiluvaiheessa. Tähän versioon saatiin toimimaan ominaisuus, jolla listaa karsitaan aloitus- ja lopetuspäivämäärien avulla. Ongelmaksi jäi kuitenkin se, että käyttäjän piti kirjoittaa päivämäärät niille varattuihin tekstikenttiin. Tekstikentät suurensivat riskiä, että päivämäärän saattoi kirjoittaa väärin tai väärässä muodossa, ja tämä yleensä johti ohjelmiston kaatumiseen. Toinen uudistus oli se, että ohjelmisto tallentaa oman CSV-tiedoston ohjelmiston juuripolkuun, josta listattua tietoa voitiin hyödyntää ohjelmiston omia jatkokäsittelyitä ajatellen.

Seuraavalla sivulla on kuva 6 24 Pesulan seurantaohjelmiston version 0.1.2 Ajan mukaan -näköymän ulkoasusta. Kuvassa näkyy varoitus, joka kehottaa käyttäjää sulkemaan kyseisen ikkunan, koska päänäkymää ei ole vielä päivitetty. Tämän ominaisuuden tarkoitus oli varmistaa, että käyttäjä karsii aina uusinta listaa.



Kuva 6. 24 Pesula Seuranta. Ohjelmistoversio 0.1.2 (Kehitysversio), Ajan mukaan.

Kuten luvussa 3.2.1 kerrottiin, Tampereen ohjelmistokokeilu osoittautui erittäin tärkeäksi kokemukseksi, joka lopulta näytti oikean reitin ohjelmiston tulevaisuudelle. Versio 0.1.2 ei toiminut, koska se oletti, että kaikki lokikansion numerot löytyisivät verrattavasta CSV-tiedostosta. Tämä ei kuitenkaan pitänyt paikkaansa, koska lokikansio sisälsi myös yrityksen sisäisiä soittonumeroita, joiden avulla pesukoneita etäkäynnistettiin ja tämän takia kansio sisälsi myös numeroita, joita ei esiintynyt yrityksen laiteluettelon tiedoissa.

19.12.2016 julkaistu versio 0.1.3 oli tärkeä korjauspäivitys, joka pelasti tilanteen ohittamalla kaikki listausprosessissa eteen sattuneet numerot, joita ei ollut CSV-luettelossa. Samalla korjattiin aikatietojen selvitys niin, että ohjelmisto luki ne suoraan tiedoston nimestä eikä luontitiedoista. Tämä toi etuna sen, ettei Windows voinut enää sotkea aikatietoja, jos tiedostoja kopioitiin tai siirrettiin. Aikatietojen sotkeentuminen oli havaittavissa ainoastaan, kun tiedostoja kopioitiin tai siirrettiin kansioista toiseen Windows 10-käyttöjärjestelmässä.

Ohjelmiston versiossa 0.1.3 muutettiin myös tapaa, miten käyttäjä valitsee Ajan mukaan -ikkunalle päivämäärät. Tekstilaatikoiden sijaan tilalle tehtiin kolme numerolaatikkoa, joista ensimmäinen osoitti päivän, toinen kuukauden ja kolmas vuoden. Tämä poisti lähes kokonaan mahdollisuuden kirjoittaa päivämäärämuoto väärin.

Versiossa 0.1.3 oli myös kehittyneempi ohjekirja, joka esitti käyttäjälle aloitustoimenpiteiden lisäksi tietoa Ajan mukaan -toiminnon käytölle ja esitti tulevan Trendinäkymän olevan kehitysvaiheessa.

### 3.3.2 Alfa-vaihe

Kuten luvussa 3.2.1 lopussa kerrottiin, ohjelmiston versionumerointi hyppäsi suoraan versiosta 0.1.3 versioon 0.2.0. Versionumeroinnissa päätettiin käyttää semanttista numerointimenetelmää, jossa kaikki 0.1:lla alkavat ovat kehitysversioita ja 0.2:lla alkavat Alfa versioita. Semanttisessa versioinnissa ensimmäinen numero viittaa merkittävään päivitykseen. Toinen numero viittaa väliversioon ja viimeinen numero on korjaus- tai kehityspäivitys. [5.] 24 Pesulan seurantaohjelmiston tapauksessa ensimmäinen numero on pysynyt samana alusta alkaen.

Versionumerointi loi selkeyttä versiotarkastelua tehdessä, mutta semanttiseen versiointiin olisi voinut alkuvaiheessa syventyä tarkemmin, jotta siirtyminen Alfa-vaiheeseen olisi nostanut ensimmäistä numeroa toisen numeron sijaan. Alfa-nimitystä alettiin käyttää kun listausominaisuus saatiin onnistuneesti toimimaan ja siten päätettiin nostaa väliversioon viittaavaa numeroa. Ei ole vielä päätetty, kumpaa numeroa ohjelmiston mahdollinen Beeta-vaihe tulee nostamaan, mutta uuden tiedon valossa ensimmäistä numeroa saatetaan nostaa, jos ohjelmiston kehitystä päätetään jatkaa. Ohjelmiston kehityksen tulevaisuudesta on lisätietoa luvussa 3.3.3.

24 Pesulan seurantaohjelmiston versio 0.2.0 julkaistiin 23.12.2016. Suurin uudistus oli trendiominaisuus, jota pystyi karsimaan aloitus- ja lopetuspäivämäärien avulla Ajan mukaan -ikkunan tapaan. Ajan mukaan -ikkunan nimi muutettiin Tuloksarsinta-nimeksi. Nimi uudistettiin, koska nyt listaa voitiin karsia päivämäärien lisäksi myös konetyyppien ja toimipaikkojen mukaan. 0.2.0:aan lisättiin myös PDF- ja CSV-tallennusominaisuus tuloksarsintänäkymään, mikä korvasi vanhan idean tulostusominaisuuden käyttämiselle. Oli järkevämpää luoda tallennusominaisuus, jolla listan pystyi tallentamaan suoraan tietokoneeseen ja käyttää sitten erillistä PDF -lukuohjelmistoa, jossa on tulostusominaisuus jo olemassa. Ohjelmiston ulkoasua, painikkeita ja ikkunoiden käyttäytymisominaisuuksia muokattiin luomaan siistimpää ulkoasua ja helpottamaan vertailuominaisuuksia.

Versio 0.2.0 sisälsi uuden tavan taulukoida tuloksia pylväsdiagrammeihin, mutta tämä sisälsi heikkouden. Koska ohjelmointitaito oli tässä vaiheessa vielä aloittelijatasolla, haasteeksi muodostui keksiä menetelmä, miten trendinäköymän tulokset saa esitettyä jokainen omana pylväsväriä toimipisteen ja ajan mukaan. Silloisena ratkaisuna kirjoitettiin jokainen toimipaikka ja konetyyppi erikseen koodiin, jotta tulokset tulivat eriteltyiksi omalle trendipalkille. Heikkoutena oli se, että nyt ohjelmisto vaati uuden päivityksen aina kun uusi laitetyyppi tai toimipiste tuli yritykseen lisää.

Seuraavaksi esitetään luettelo toimipisteistä sekä konetyypeistä, joita versio 0.2.0 tuki.

Toimipisteet:

- Hämeenlinna
- Kokkola
- Konala
- Pirkkala
- Raisio
- Tullintori
- Turku
- Turtola
- Vallila
- Viikki
- Ylöjärvi.



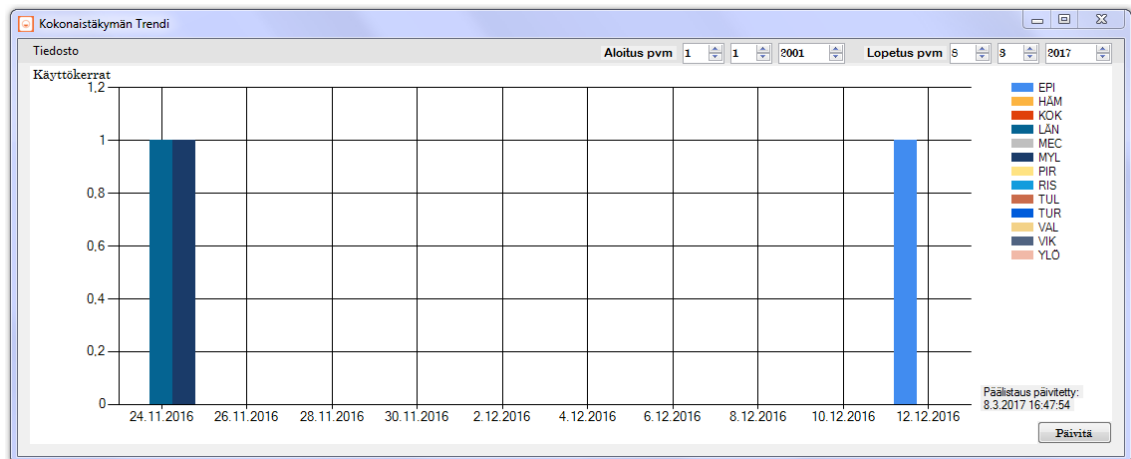
Konetyypit:

- pyykkikone
- iso pyykkikone
- jätti-pyykkikone
- mattokone
- tasopesukone.
- kuivausrumpu
- kuivauskaapisto.

Ohjelmiston Alfa-vaihe sai ensimmäisen korjauspäivityksen 3.1.2017. Tämä 0.2.1-versio sisälsi vain kirjoitusvirheen korjausta sekä ratkaisun ongelmaan, joka keskeytti ohjelman virheettömän toiminnan, jos syötettyä päivämäärää ei ollut kalenterissa.

Versio 0.2.2 julkaistiin yrityksen sisällä 10.1.2017, kun 24 Pesula avasi uuden toimipisteen Helsingin Etu-Töölöön, Mechelininkadulle. Ohjelmistonkin oli tuettava uutta toimipistettä. Uuden toimipisteen tuen lisäksi aikaongelmien selvittely jatkui edelleen. Tällä kertaa ohjelmisto kehitettiin huomauttamaan, jos käyttäjä asetti uudemman aloituspäivämäärän, lopetuspäivämäärään nähden. Muuten ohjelmisto sai jälleen ulkoasullisia havainnollistamisparannuksia.

Sivulla 29 on kuva 7 version 0.2.2 Trendinäköymästä, joka otettiin ensimmäisenä käyttöön versiossa 0.2.0. Kuvasta 7 huomataan, miten ohjelma näyttää intervallit 0,2 käyttökerran väleinä, mikä ei käytännössä ole mahdollista. Pylväät näyttävät kuitenkin tulokset oikein, joten intervaleille ei vielä tehty korjausta tässä versiossa. Koska 24 Pesulan virallinen data on luottamuksellista, näkyy sivun 29 kuvissa ainoastaan testikansion XML-tiedostojen mukaan generoitu data.



Kuva 7. 24 Pesula Seuranta. Ohjelmistoversio 0.2.2 (Alfa), Kokonaistäkymän Trendi.

Kuvassa 8 esitetään sama data version 0.2.2 Tuloksarsintaikkunassa. Ainoa ero versioon 0.2.0 nähden on se, että Paikat-näkymään lisättiin paikka: MEC, joka viittaa Mechelininkatuun. Version 0.2.2 tuloksarsintanäkymän ainoa puute oli, että se näytti kaikki yrityksessä olevat toimipaikat ja koneityypit yhtä aikaa, mikä teki karsintaominaisuuksien käytöstä työläämpää. Käyttäjän piti tarkistaa päänäkymän listalta tai arvata, mitä paikkoja ja koneita oli mahdollista lähteä karsimaan.

Paikka	Kone	Maksupalvelu	Soittnumero	Aika
EPI	P1	06060070137	+358465008	12.12.2016 13:29:50
LÄN	T1	06060071008	+35850400	24.11.2016 14:25:28
MYL	I1	06060071083	+35850317	24.11.2016 14:27:50

Kuva 8. 24 Pesula Seuranta. Ohjelmistoversio 0.2.2 (Alfa), Tuloksarsinta.

Versio 0.2.3 julkaistiin yrityksen sisällä 17.1.2017. Se oli loistava esimerkki siitä, miten käy kun ohjelmoinnissa aletaan kulkea myös sivuteillä. Tämä versio esitteli ensimmäisen aloitusruudun (Splash Screen), jolla ohjelmisto käynnistyi. Aloitusruudun idea on indikoida käyttäjälle, että ohjelmisto on käynnistymässä ja lataa kaikki tarvittavat komponentit toimiakseen. Tässä versiossa käynnistysruudun ainoa funktio oli esittää käyttäjälle, minkä ohjelmistoversion hän juuri käynnisti. [15.]

Versio 0.2.3 sisälsi ison helpottavan uudistuksen. Käyttäjän ei enää tarvinnut hakea erillisestä kansioista CSV-tiedostoa, josta ohjelmisto vertailee kaikki yrityksen koneet. CSV-tiedoston hakuominaisuuden korvasi Konelistausnäkyvä, jonka tehtävänä oli avata kaikki yrityksen konetyypit näyttävä lista. Käytännössä kyseessä on sama CSV-tiedosto, jonka käyttäjä valitsi aikaisemmissa versioissa aina erikseen, mutta tällä kertaa se oli siirretty ohjelmiston juuripolkuun, josta sen tiedot sai kätevästi ilman käyttäjän toimenpiteitä. Konelistaus määritettiin myös laskemaan ja esittämään käyttäjälle kaikki yrityksen konetyypit sekä näiden yhteenlasketun summan.

Myös trendinäkyvä sai uuden ominaisuuden, mikä mahdollisti toimipisteiden karsimisen. Tämä karsintaominaisuus toi runsaasti selkeyttä pylväsdiagrammien erottelemiseen. Myöhemmin huomattiin, että kyseisen version trendinäkyvän valintalaatikot eivät pysyneet oikeassa paikassa, jos ikkunan kokoa muutettiin. Tämä ongelma korjattiin seuraavassa päivityksessä matemaattisella kaavalla.

Valintalaatikon paikkaa ikkunan leveysuunnassa korjataan kaavalla:

$$kerroinW = \left( \left( \frac{formWidth}{1126.0} \right) - 1 \right) * 33$$

Valintalaatikon leveyskoordinaatin lopputulos on *kerroinW*. Tulos on kuva-alkioina. *formWidth* on koodin selvittämä arvo ikkunan todellisesta leveysarvosta. Arvo 1126.0 on ikkunan alkuperäinen leveys, joka kirjattiin ylös kun valintalaatikot asetettiin haluttuun paikkaan ikkunassa. Jakolaskun tuloksesta vähennetään luku yksi, jotta arvoksi saadaan nolla aina kun ikkuna on vakioleveydessä. Näin määritetään, kumpaan suuntaan keskipisteestä valintalaatikon on tarkoitus lähteä siirtymään, jos ikkunan kokoa muutetaan. Koko tulos kerrotaan arvolla 33, jolla määritetään, kuinka paljon valintalaatikko siirtyy suunnassaan. Tämä arvo selvitettiin kokeilemalla. Liian iso arvo siirsi valintalaatikkoa liian pitkälle ikkunan leveyteen nähden ja liian pieni arvo jättäti. Korkeutta

kontrolloivassa kaavassa siirtymän voimakkuutta määräävä arvo on 14, koska ikkunan alkuperäinen korkeus on 456,0.

Versio 0.2.4 julkaistiin pienen viiveen jälkeen 7.2.2017. Oli selvä, että ominaisuuksien kehittämiseen käytettiin yhä enemmän aikaa samalla kun vaatimustaso koveni. Tämä versio sisälsi lukumäärällisesti eniten uudistuksia edeltäjiinsä nähden. Suurin uudistus oli päästä eroon ennalta määritetyistä toimipisteistä ja konetyypeistä. Ratkaisuksi kehitettiin tallennuspistetyyppinen menetelmä, jossa luotiin 20 tyhjää paikkaa toimipisteille ja konetyypeille.

Oli hyvä idea kehittää versioon 0.1.2 ominaisuus, joka loi päänäkömän listauksen yhteydessä kokonaisraportti.csv tiedoston 24 Pesulan seurantaohjelmiston juuripolkuun. Tätä ominaisuutta käytettiin hyödyksi kaikissa tarvittavissa toimenpiteissä, joissa listatulle tiedolle haluttiin jatkokäsittelyä ja näin ohjelmiston käyttö nopeutui, koska ohjelman ei tarvinnut hakea tietoja enää uudestaan lokikansion ja konelistauksen perusteella. Konelistauksen ansiosta trendinäköymässä voitiin hyödyntää CSV-tiedostoa selvittämään nopeasti, mitkä toimipisteet ovat käytössä, ja tämän tiedon perusteella ne tallennettiin automaattisesti tyhjiin tallennuspaikkoihin. Konetyyppien maksimaalinen lukumäärä oli periaatteessa rajaton, mutta jos tallennuspaikat loppuivat kesken, ei ohjelmisto kyennyt hyödyntämään ylimenevää tietoa kaikissa tarvittavissa toimenpiteissä. Tallennuspaikkaidea toi enemmän vapautta toimipistepäivityksiin, mutta ennemmin tai myöhemmin senkin tilan tiedettiin tulevan tiensä päähän. Jos tallennuspaikat loppuvat, niitä joutuu kirjoittamaan lisää suoraan koodiin, ellei kyseistä menetelmää onnistuta korvaamaan paremmalla menetelmällä.

Tuloskarsintaikkuna koki myös muutoksen. Automaattisesti mukautuvien paikkojen ja koneiden tulon myötä tuloskarsinnan valintalaatikoitakaan ei voinut enää pitää samantaisina. Tilalle kehitettiin valintalaatikkolista, johon tiedot listattiin aakkosjärjestyksessä sitä mukaa kun toimipisteitä ja konetyyppejä ilmaantui.

Konelistausnäkömään lisättiin editointiominaisuus. Editointiominaisuuden tarkoituksena oli antaa käyttäjälle mahdollisuus lisätä ja poistaa koneita ja tallentaa muutokset CSV-tiedostoon. On olemassa CSV-tiedoston muokkaukseen soveltuvia ohjelmistoja joilla voi tehdä muokkauksia suoraan tiedostoon. Notepad++ osoittautui hyväksi vaihtoehdoksi muokkausten tekemiseen, mutta konelistauksen selkeämpi tyyli vähensi mahdol-

lisiä muokkauksesta aiheutuvia virheiden määriä ja helpotti, jos käyttäjä ei osannut avata CSV-tiedostoa muokkaukseen soveltuvalla ohjelmalla.

Muita version 0.2.4 uudistuksia:

- CSV-tiedostojen tallentamisessa ilmentyneet enokoodausongelmat korjattiin.
- Konelistauksen laskemat laitetyypit järjestettiin aakkosjärjestykseen
- Lisättiin varoitus tallennusyrityksiin ja ylittyneisiin tallennuspaikkoihin.
- Ohjelmiston käyttöä selkeytettiin erilaisilla muokkauksilla.
- Päänäkymän sulje-painikkeen, tulostarkinnan koneiden ja paikkojen nimitykset muokattiin havainnollisuuden parantamiseksi.
- Trendi-ikkunan minimikorkeutta nostettiin.
- Ulkoasuun tehtiin parannuksia.

Alfa-vaiheen lopullista ulkoasua edustava versio 0.2.5 julkaistiin 6.3.2017. Sillä saavutettiin täydennystilauksen tavoitteet tulevaisuuden visioita lukuun ottamatta. Tämä versio mahdollisti trendinäkymien valinnan siten, että kokonaisnäkymän lisäksi käyttäjä sai valita myös kuukausittaisen tai päiväkohtaisen trendinäkymän. Kuukausittainen näkymä poisti tarpeen asettaa alku- ja loppupäivää valintalaatikkoon ja päiväkohtainen näkymä näytti trendiviivoilla valitun päivän käytöt kellonaikoina. Toinen uudistus oli trendinäkymän zoomaaminen X-akselin suuntaisesti käyttäen hiiren rullaa.

Suurin uudistus oli ominaisuus luoda raportti karsittujen tietojen perusteella, mikä oli yrityksen pitkäaikainen haave tavasta saada tiedot raportoitua. Tämä ominaisuus toimi tulostarkintaikkunan kautta, jonne lisättiin painikkeet yhteenvetotrendien näyttämiseksi ja Luo raportti -painike, jonka kautta käyttäjä sai tallentaa tiedoston halutulla nimellä haluttuun tallennuspaikkaan. Raporttiominaisuus loi A4-kokoisen PDF-asiakirjan. Tämä asiakirja sisälsi tulostarkintanäkymän esittämät kustannuspaikat, löydettyjen konetyyppien käyttökerrat, yhteenlasketut käyttökerrat näyttävän trendin ja suosituimman kellonajan tietyltä ajalta näyttävän trendin.

Muita uudistuksia ja parannuksia toi muun muassa ominaisuus, joka aukaisi uuden ikkunan vain kerran eikä tehnyt enää duplikaattia, jos saman ikkunan aukaisi vanhan ollessa jo auki. Tulokarsintaikkunan avaamiselle luotiin poikkeava ominaisuus, joka edelleen mahdollisti useiden tulokarsintaikkunoiden aukaisun, mutta tällä kertaa ikkunat numeroituivat käyttäjän avaamien tulokarsintaikkunoiden määrän mukaan. Ohjelmiston päivämäärävalinnat siirtyivät tukemaan kalenterin mukaista formaattia sen sijaan, että käyttäjä olisi voinut valita jokaiselle kuukaudelle esimerkiksi 31. päivän oman mielensä mukaan. Päivämääräkorjauksen myötä ohjelmiston päivämääriin liittyvät ongelmat saatiin vihdoinkin täysin eliminoidua.

Tiedoston tallennusominaisuuden lisääntyvän käyttötarpeen myötä huomattiin, että ylikirjoitusprosessi ei aikaisemmin ollut korvannut jo olemassa olevaa tiedostoa. Tämä ongelma saatiin tässä versiossa korjattua lisäämällä koodiin ehtojen mukaisia toimenpiteitä ja poistettiin turhia .pdf- tai .csv-päätteiden lisäyksiä. Valittavasta tiedostosta kopioitui joka tapauksessa koko nimi päätteiden kanssa, kun tiedoston olemassaolo selvitetiin. Ennen päätelisyysten poistamista korvattava tiedosto jäi kansioon ja sen viereen tallentui toinen vastaavan niminen tiedosto, jonka nimen perässä oli kaksi päätettä peräkkään. Ylikirjoitusongelman korjauksen yhteydessä lisättiin myös hyödyllisiä varoituksia käyttäjälle ja tehtiin ulkoasumuutoksia.

Seuraava ohjelmistoversio oli kehitteillä, kun projekti sai lisää tuulta siipensä alle. 24 Pesula alkoi kartoittaa alihankkijoita liittymään projektiin mukaan. Kartoituksen tarkoituksena oli selvittää, miten vastaavaa ohjelmistoa saadaan skaalattua ulkopuoliselle palvelimelle. Kartoituksessa selvisi, että ohjelmistopohjissa oli osittain päällekkäisyyksiä, joten 24 Pesulan seurantaohjelmiston lisäominaisuuksien kehitystyö päätettiin laittaa hetkeksi tauolle. Ohjelmisto sai kuitenkin lisää korjauksia käytössä havaittuihin ongelmiin, kuten Ohje -näytön ulkoasun uudistuksia. Samalla suunniteltiin virkistyspainikkeet mahdollistamaan viimeisimmän datan karsimisen ilman, että koko ikkunaa tarvitsi avata uudestaan.

### 3.3.3 Jatkosuunnitelmat

24 Pesulalta saadun tiedon mukaan seurantaohjelmistoa aiotaan hyvällä todennäköisyydellä käyttää jatkossakin tulevan palvelimen rinnalla. Haasteita tuovat muun muassa se, että MC66-modeemin käyttö aiotaan lopettaa ja uuden palvelimen myötä käynnistystiedot eivät tule enää olemaan nykyisessä muodossa. Tulevan tietoliikennejärjes-

telmän tyyppi on vielä suunnitteluvaiheessa. Tämä tarkoittaa sitä että 24 Pesula Seuranta ohjelmiston on mukauduttava tulevaisuuden mukaan ja sen listausmenetelmää joudutaan uudistamaan.

On kaksi mahdollista tapaa, joita lähteä ideoimaan, kun suunnitellaan uudistuksen myötä muutettavaa menetelmää listata käynnistystiedot. Ensimmäinen tapa on se, että ohjelmistoon liitetään .NET:in mahdollistama ominaisuus, jolla haetaan käynnistystiedot suoraan uudelta palvelimelta. Toisen tavan idea on, että jos uudistuneen palvelimen tiedot on mahdollista synkronoida Windowsin resurssienhallintaan, tarvitsee ohjelmistoon tehdä ainoastaan muutoksia tapaan, jolla tiedot luetaan tiedostoista nykyisen perusidean mukaisesti.

Ohjelmistolle on jatkokehityssuunnitelma, jolla on tarkoitus antaa käyttäjälle mahdollisuus käynnistää listausprosessi automaattiseen tilaan. Tällöin projektin alkuperäinen tavoite luoda sovellus listaamaan yrityksen toimipisteiden koneet ja niiden käyttökerrat automatisoidusti, tulee vihdoinkin saavutetuksi.

Tämän opinnäytetyön kirjoittamisen aikana 24 Pesulan Seurantaohjelmisto on aktiivisesti käytössä. Kaikkia sen ominaisuuksia testataan tarkasti ja mahdollisia puutteita kirjataan ylös. Jos ongelmia tai parannuskohtia ilmenee, korjataan ne koodiin. Tarkoitus on saada ohjelmasta mahdollisimman stabiili, havainnollistava ja helppokäyttöinen, jotta sillä on mahdollisimman iso rooli tulevaisuuden seurantamenetelmien joukossa.

Kun tarvittavat korjaukset ja uudistukset on lisätty koodiin ja tulevaisuuden tiedonkeruumenetelmät on tiedossa, siirtyy ohjelmisto versioinnissa Beeta-vaiheeseen. Beeta-versiot tulevat tukemaan vanhan tiedonkeruumenetelmän lisäksi uutta menetelmää niin, että ohjelmisto osaa automaattisesti valita oikean menetelmän tietotyypin mukaan. Näin vanhoja käyttötietoja voidaan edelleen lukea uuden menetelmän rinnalla. Lisäksi on tarkoitus päästä eroon tyhjästä tallennuspaikoista, joita tällä hetkellä käytetään löydetyille toimipisteille. Beeta-versioissa aletaan myös ensimmäistä kertaa kehittää ominaisuutta, jolla tietyn laitetyypin voi etäkäynnistää tarvittaessa. Tämä ominaisuus tulee toimimaan vaihtoehtoisena tapana nykyisen etäkäynnistysmenetelmän kanssa. Etäkäynnistys onnistuu henkilökunnan puolelta, matkapuhelinta käyttämällä. Etäkäynnistystä tarvitaan esimerkiksi silloin, jos asiakkaalla on ilmennyt ongelmia pyykinpesun yhteydessä ja kone tarvitsee uuden käynnistuksen ilman, että asiakas joutuu maksamaan tästä lisää.

### 3.4 24 Pesulan seurantaohjelmiston käyttö

Tässä luvussa käydään läpi, miten 24 Pesulan Seurantaohjelmisto on suunniteltu käytettäväksi. Kaikki käyttöön liittyvät ohjeet löytyvät myös ohjelmiston päänäköymän Tiedot-valikosta, johon tämän opinnäytetyön käyttöosion tiedot perustuvat. Ohjeet tuodaan esille kuvien ja esimerkkien avulla ja niissä paljastetaan yksityiskohtia, joita ei ohjelmiston Alfa-vaiheen ohjeikkunassa ole tuotu esille.

Tämän luvun kuvat ovat pääasiassa ohjelmiston Alfa-versiosta 0.2.6, joka edustaa Alfa vaiheen viimeisintä ulkoasua ennen siirtymistä Beetaan. Kaikki 0.2.6 jälkeen julkaistut versiot ovat korjauspäivityksiä ja niitä kehitetään tämän opinnäytetyön kirjoittamisen aikana. Uusien päivitysten tuomat ohjelmistomuutokset on otettu huomioon lukua 3.4 kirjoitettaessa. Alla on esitetty taulukko 2 ohjelmiston päänäköymän valikkorakenteesta. Taulukkoa kannattaa silmäillä ohjetta lukiessa. Vihreällä taustalla korostettu otsikko tarkoittaa pudotusvalikkopainiketta, jonka alle on lueteltu kaikki ne painikkeet, jotka esiintyvät otsikkoa vastaavan valikon ollessa auki. Sinisellä merkattu alue tarkoittaa sitä, että Tiedosto-valikon Tallenna-painike avaa alivalikon, joka sisältää esitetyt painikkeet.

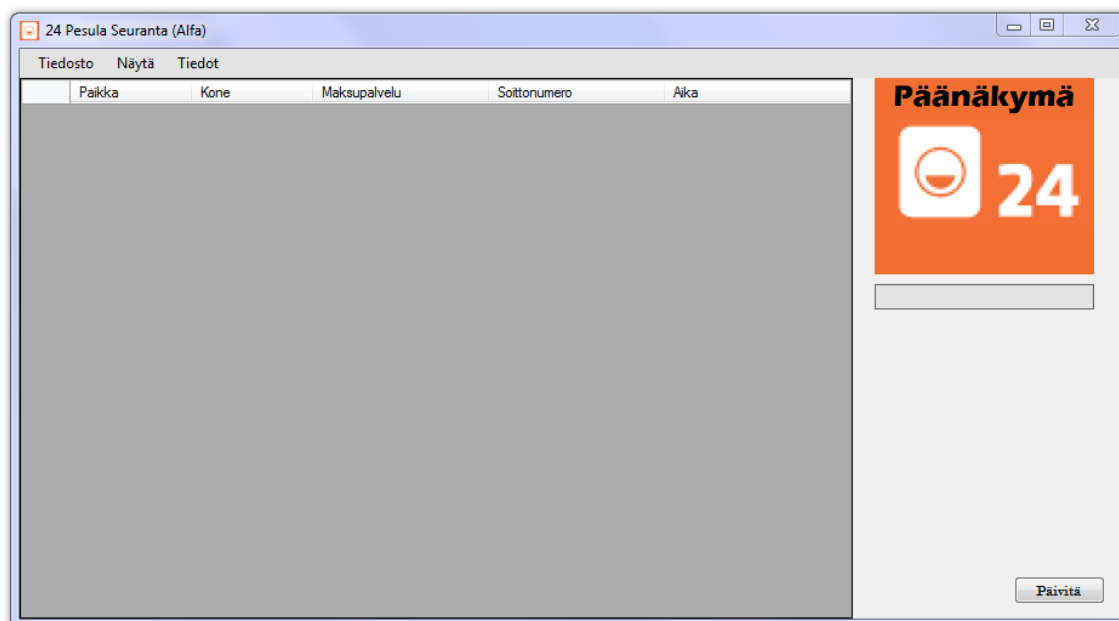
Taulukko 2. Päänäköymän valikkorakenne.

Päänäköymä		
Tiedosto	Näytä	Tiedot
Avaa XML-kansio	Trendi	Versio
Konelistaus	Tuloskarsinta	Ohje
Tallenna		
Tallenna CSV		
Tallenna PDF		
Sulje ohjelma		

#### 3.4.1 Aloitustoimenpiteet

24 Pesula Seurantaohjelmisto käynnistyy aloitusruudulla, josta ilmenee ohjelmiston nimi ja versio. Käynnistysruudun hävitessä eteen avautuu ohjelmiston päänäköymä. Päänäköymän tarkoituksena on mahdollistaa ohjelmiston perustoiminnoille niiden edellyttämät toimenpiteet. Sivulla 36 on kuva 9 Päänäköymän ulkoasusta.





Kuva 9. 24 Pesula Seuranta. Ohjelmistoversio 0.2.6 (Alfa), Päänäkymä.

Päänäkymän vasemmassa ylänurkassa on painike, jossa lukee Tiedosto. Ohjelmiston ensimmäiset toimenpiteet käynnistyksen jälkeen tehdään aina tämän valikon kautta. Tiedosto-painikkeesta avautuvasta pudotusvalikosta on lisää tietoa luvussa 3.4.2.

Kun valikko on avattu, kannattaa varmistaa, että konelistaus on kunnossa valitsemalla kohta Konelistaus. Konelistauksen kautta voidaan varmistaa ja tarvittaessa muokata listausta niin, että kaikki tarvittavat koneet, paikat ja niiden tiedot ovat listattuna ja näin ohjelmiston käytettävissä. Jos tarvittavat tiedot ovat kunnossa, voidaan konelistaus huoletta sulkea ja palata takaisin ohjelmiston päänäkymään.

On suositeltavaa, että ohjelmistoa käytettäessä käytössä on aina viimeisin listaus yrityksen käytöistä. Ohjelmisto pitää sulkemisesta huolimatta tallessa listauksen, jota siinä oli viimeksi käytetty, ja tätä on myös mahdollista käsitellä ohjelmiston karsinta- ja trendityökalulla. Jotta päästään käsittelemään tuoreinta listaa, on Tiedosto-valikosta valittava kohta Avaa XML -kansio. Tämä avaa eteen valintaikkunan, josta käyttäjä valitsee ohjelmiston tietoon kansion (Loki), jonne palvelin tallentaa kaikki yrityksen viimeisimmät käytöt XML-tiedostoina. Kun kansio on valittu, käyttäjä saa ilmoituksen löytyneiden tiedostojen lukumäärästä. Lisäksi päänäkymän oikeaan reunaan ilmestyy indikaatio, että tiedostopolku on kytketty.

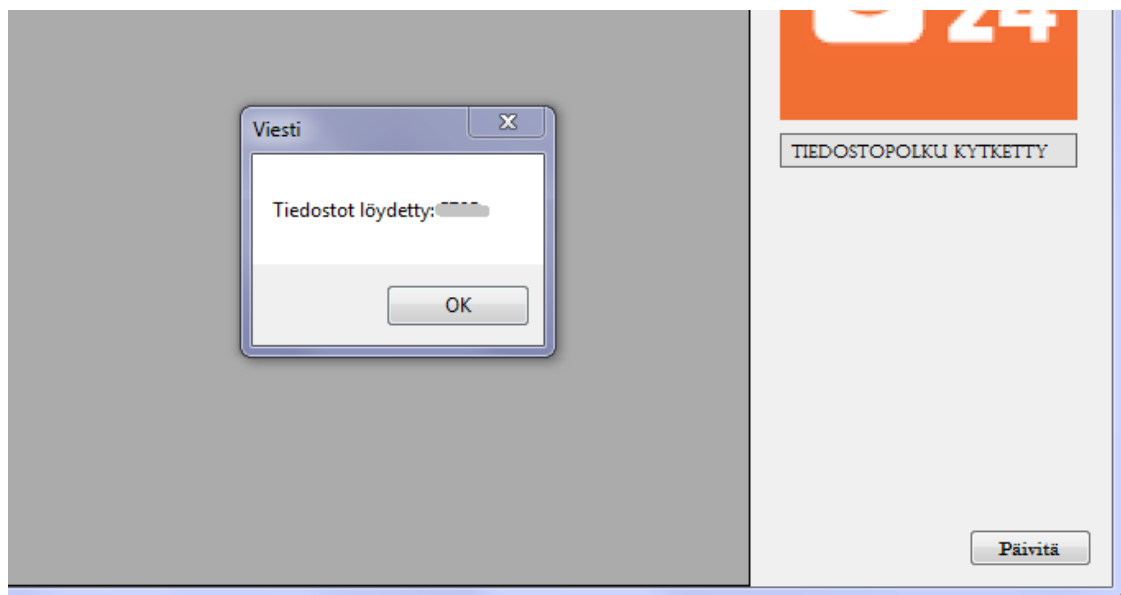
Lokikansion XML-tiedostosta ohjelmisto saa selville asiakkaan käyttämän konetyypin soittonumeron ja tiedoston luontiajan. Konelistaus määrittää jokaiselle soittonumerolle paikkatiedon, konetyypin ja maksupalvelunumeron. Kun lokikansio on kytketty ohjelmiston tietoon, voidaan painaa päänäkömään oikeassa alareunassa olevaa Päivitä-painiketta. Ohjelmisto tallentaa ensin kaikki konelistauksesta löytyneiden koneiden tiedot muistipaikkaan, jonka jälkeen se listaa koneiden tiedot soittonumeroiden perusteella oikeille riveille listausnäkyssä. Rivin loppuun lisätään XML-tiedostojen luontiajat, joilla esitetään kunkin koneen käytön aloitusajankohta. Tämä toimenpide näkyy käyttäjälle ainoastaan päivityksen ilmaisuna, jolloin Windowsin työkalurivi ilmoittaa listan päivityksestä. Prosessin lopuksi ohjelmisto tallentaa juuripolkuun Kokonaisraportti.csv-tiedoston, joka sisältää koko listauksen tiedot ja viimeisimmän muokkausajan. Windowsin työkalurivi ilmoittaa päivityksen valmistumisesta.

Edeltä voidaan päätellä miten tärkeää on, että konelistauksen tiedot on oikein määritetty. Jos yksikin soittonumero on väärin, jäävät sitä vastaavan koneen käyttötiedot kokonaan huomiotta. Samoin, jos soittonumeroon liittyvät lisätiedot ovat väärin tai sisältävät virheitä, kopioituvat ne päälistaukseen. Päälistauksesta virheet saattavat mahdollisesti jatkaa raportointivaiheeseen asti, jos niitä ei siihen mennessä ole huomattu.

Onnistunutta päälistausta voidaan tarkastella suoraan päänäkömästä sarakkeiden ryhmittelyjä muokkaamalla, mikä onnistuu niiden otsikoita klikkaamalla. Vaihtoehtoisesti käyttäjä voi siirtyä kehittyneempiin tarkasteluvaihtoehtoihin seuraavien lukujen ohjeiden mukaisesti.

### 3.4.2 Tiedosto-valikko

Tiedosto-valikossa ylimpänä löytyy avauspainike XML-polulle. Tämän ominaisuuden käyttäminen on välttämätöntä, jos halutaan tarkastella yrityksen viimeisimpiä käynnistystietoja. Liitteessä 1 on kuva kansioden selausnäkömästä. Liitteen kuvassa määritetään ohjelmistolle lokikansion sijainti. Kuvan 10 esimerkissä on ilmoitettu lokikansion sisältämien tiedostojen lukumäärä. Lukumäärä on jouduttu sensuroimaan luottamuksellisista syistä.



Kuva 10. 24 Pesula Seuranta. Ohjelmistoversio 0.2.6 (Alfa), Indikaatio.

Kaikkia indikaation viestissä ilmoitettuja tiedostoja ei yleensä tule listaukseen mukaan, koska kansio saattaa sisältää tiedostoja, jotka eivät sisällä tarvittavia tietoja tai ovat tiedostotyyplitään erilaisia. Lisäksi on mahdollista, että tiedostosta löytynyt soittonumero ei vastaa mitään konelistauksen numeroa, jolloin tiedoston määrittämää konetta ei huomioida listauksessa.

Tiedosto-valikossa toisena toimintona on konelistaus. Konelistaus näyttää kaikki 24 Pesula Oy:n koneet, niiden paikat, maksupalvelu- sekä soittonumerot, jotka on listattu ohjelmiston juuripolussa sijaitsevaan CSV-tiedostoon. Jos listauksessa on puutteita tai virheitä, on listaa mahdollista muokata kirjoittamalla suoraan rivin sarakkeelle. Tietoja voidaan myös poistaa, jos konetyyppi on virheellinen tai poistettu käytöstä. Kursori on muistettava poistaa ruudulta muokkauksen jälkeen, jotta kyseisen ruudun muokkaus huomioidaan tallennettaessa. Tallennuspainike löytyy konelistauksen vasemmassa ylänurkassa olevasta Tiedosto-valikosta. Tallentamisen jälkeen konelistaus käynnistyy uudelleen ja ilmoittaa tallennuksen onnistumisesta. Jos muutokset tallentuivat, näkyvät nämä konelistauksessa välittömästi edellä mainitun prosessin jälkeen.

Juuripolun CSV-tiedostoa on mahdollista muokata myös erillisellä sovelluksella, mutta tällöin on oltava tarkkana, että tiedoston ulkoasu on oikein. Jokainen sarake erotellaan puolipisteellä ja soittonumeron tulee olla muotoa +358123456789. Liitteessä 2 on esitetty konelistauksen käyttämän CSV-tiedoston ulkoasu.

Kuvassa 11 on näkymä konelistauksesta. Konelistauksen sarakejärjestys on sama, kuin kaikissa muissakin listoissa, mikä helpottaa listan muokkaamista. Päänäkymässä ja tulostarkintanäkymässä lisätään konelistauksen mukaisen sarakejärjestyksen perään aikatieta, josta ilmenee, milloin kutakin lokista löytynyttä konetta on käytetty. Alla olevasta kuvasta on huomattavissa kohta, jossa koneella R1 ei ole soittonumeroa. Tätä konetta ei ole mahdollista käyttää missään ohjelmiston toimenpiteissä ennen kuin soittonumero on päivitetty konelistaukseen.

Konelistauksen oikeassa reunassa on ilmoitettu jokaisen konetyypin lukumäärä toimipisteestä riippumatta. Lisäksi kaikkien koneiden yhteenlaskettu lukumäärä on ilmoitettu listassa alimpana.

Tiedosto					Koneet yhteensä
	Paikka	Kone	Maksupalvelu	Soittonumero	
▶	KOK	M1	06060071014	+358503420000	I1: 11
	KOK	M2	06060061882	+358504360000	J1: 13
	KOK	M3	06060071026	+358505730000	M1: 12
	KOK	J1	06060070149	+3584650010000	M2: 9
	KOK	P0	06060062085	+358504360000	M3: 1
	KOK	P1	06060071010	+358469220000	P0: 1
	KOK	P2	06060071068	+358504670000	P1: 13
	KOK	R1	06060070128		P2: 11
	KOK	T1	06060061425	+358504360000	P3: 3
	EPI	M1	06060061871	+358505230000	R1: 12
	EPI	M2	06060062084	+358505230000	R2: 10
	EPI	J1	06060061902	+358505230000	T1: 13
	EPI	P1	06060070137	+3584650000000	TPK: 1
	EPI	P2	06060070138	+3584650000000	Yhteensä: 110
	EPI	P3	06060070139	+3584650000000	
	EPI	R1	06060070142	+3584650010000	
	EPI	R2	06060070126	+3584650000000	
	EPI	T1	06060061835	+358505230000	
	YLÖ	M1	06060070121	+358469210000	
	YLÖ	M2	06060071015	+358504630000	
	YLÖ	J1	06060070140	+3584650010000	
	YLÖ	I1	06060071070		
	YLÖ	P1	06060070141	+3584650010000	
	YLÖ	P2	06060071022	+358504650000	

Kuva 11. 24 Pesula Seuranta. Ohjelmistoversio 0.2.6 (Alfa), Konelistaus.

Tiedosto-valikon loppupuolelta löytyy Tallenna-painike. Päänäkymän tallennuspainike tallentaa päälistan esittämän listausnäkökuvan kokonaisuudessaan. Käyttäjä voi valita tallennusformaatiksi joko PDF- tai CSV-muodon. PDF-muoto on helppo avata ja tarkastella erillisellä PDF-tiedoston lukuun soveltuvalla sovelluksella, kun taas CSV-tiedoston voi avata tekstitiedostona ja sitä voi muokata myöhemmässä vaiheessa.

Sulje ohjelma -painikkeella ohjelman voi sulkea kokonaisuudessaan vaikka muita ohjelmiston omia ikkunoita on auki.

### 3.4.3 Trendi

Päänäkymän Näytä-valikko tarjoaa käyttäjälle edistyneempiä työkaluja yrityksen käytön seuraamiselle. Ensimmäisenä valikosta löytyy Trendi, jota painamalla eteen avautuu uusi ikkuna. Tämä ikkuna näyttää ensimmäisen avauksen jälkeen kaikki päälistauksen esittämät käytöt graafisena pylväsdiagramminäkymänä. Yrityksen yhteenlasketut, toimipistekohtaiset käyttötiedot on esitetty Y-akselilla ja jokaista käyttötietoa vastaava aika on esitetty X-akselilla. Toiminto hyödyntää CSV-listaa, joka päänäkökuvaa päivitettäessä on tallentunut juuripolkuun. Jos päänäkökuvaa ei ole päivitetty ennen Trendi-ikkunan aukaisua, näkyy tästä varoitus ikkunan oikeassa alareunassa. Trendi-ominaisuuksia on tästä huolimatta mahdollista käyttää, mutta käytettävästä listasta saattaa olla olemassa päivitetyn versio, mikä kannattaa ottaa huomioon. Alla on esitetty taulukko 3 selkeyttämään trendinäkökuvan valikkorakennetta. Trendinäkökuvan valinta ilmenee ohjelmistossa tyhjänä laatikkona, kun trendinäkökuvaa ei ole valittu.

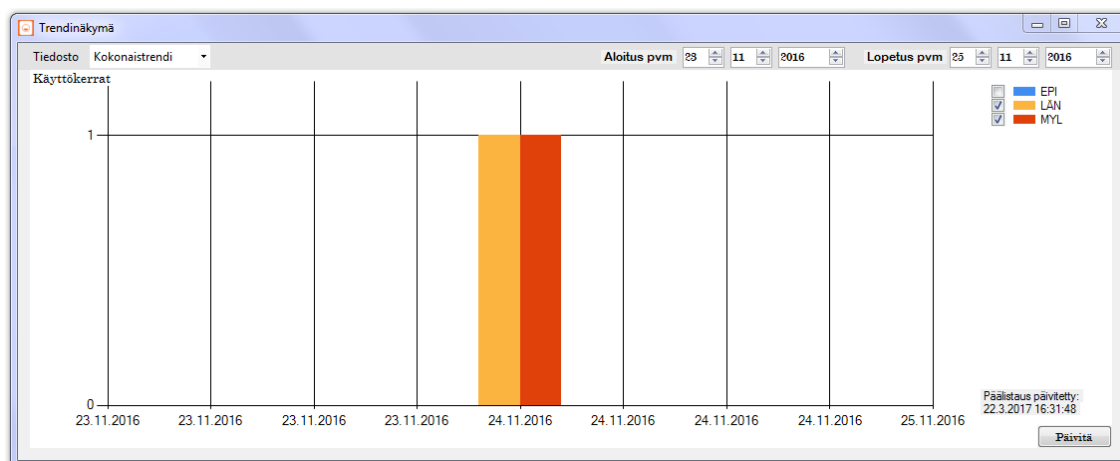
Taulukko 3. Trendinäkökuvan valikkorakenne.

Trendi	
Tiedosto	Trendinäkökuvan valinta
Virkistä	Kokonaistrendi
Sulje	Kuukausitrendi
	Päivätrendi

Trendinäkökuvan oikeassa laidassa näkyy 24 Pesulan toimipisteet. Kuvasta 12 on havaittavissa, että toimipisteitä on paljon vähemmän, kuin sivulla 29 esiintyvässä kuvassa 7, jossa esitettiin trendinäkökuvan ohjelmiston versiosta 0.2.2. Tämä johtuu siitä, että uu-

dempi ohjelmisto karsii päivitetyn päänäkömään mukaan kaikki ne toimipisteet, joita ei esiinny listauksessa sen sijaan, että kaikki yrityksen toimipisteet esiintyisivät yhtä aikaa. Koska alla olevan kuvan tilanteessa on ohjelmiston tiedossa kolme käyttöä sisältävä testilokikansio ja käytöt ovat jokainen eri toimipisteeltä, näyttää ohjelmisto ainoastaan nämä kolme karsittavaa toimipistettä ja näin käyttäjän havainnointia on helpotettu huomattavasti. Toimipisteiden vieressä on valintalaatikot, joita klikkaamalla voidaan määrittää, mitkä toimipisteet näkyvät ja mitkä eivät. Jos valintalaatikossa on merkki, näkyy kyseisen toimipisteen käyttötieto trendi-ikkunassa, yhdellä tai useammalla ajan hetkellä.

Ikkunan oikeassa yläreunassa on omat numerolaatikot aloitus- ja lopetuspäivämäärille. Näiden laatikoiden avulla käyttäjä voi tarkastella trendiä haluamaltaan aikaväliltä. Oletuksena trendinäkömää käyttää aloitus- ja lopetuspäivämääräarvoina 1.1.2001. Jos päänäkömään tallentama CSV-tiedosto löytyy juuripolusta, määrytyy lopetuspäivämääräksi tiedostosta löytyneen, viimeisimmän käytön päivämäärä. Tiedoston viimeisin muokausaika näkyy Päivitä -painikkeen vieressä kellon ajan tarkkuudella.



Kuva 12. 24 Pesula Seuranta. Ohjelmistoversio 0.2.6 (Alfa), Trendinäkömää.

Ikkunan vasemmassa yläreunassa Tiedosto-painikkeen vieressä on pudotusvalikko trendinäkömälle, jota halutaan tarkastella. Pudotusvalikko antaa käyttäjälle kolme vaihtoehtoa, joista käyttäjän on valittava trendinäkömää ennen kuin trendiominaisuuksia voidaan käyttää. Trendinäkömää ohjeistaa käyttäjää pudotusvalikon viereltä, mitä on tehtävä. Ohjeistus vaihtaa värin punaiseksi, jos käyttäjä yrittää päivittää ikkunan ennen trendinäkömään valitsemista.

Kokonaistrendi näyttää koko listauksen tiedot samalla tavalla, kuin trendinäkymän aukaisun alussakin. Näkymää voi karsia vapaasti päivän tarkkuudella aloituspäivämäärän ja lopetuspäivämäärän mukaan. Lisäksi toimipisteiden karsinta on käytettävissä.

Kuukausitrendi näyttää yrityksen käyttötiedot yhden kuukauden ajalta. Päivämäärien valintalaatikoista on poissa käytöstä vasemmalta oikealle katsottuna neljä ensimmäistä numerolaatikkoa. Näin jäljelle jää enää kaksi aktiivista laatikkoa, joiden avulla käyttäjä voi valita halutun kuukauden ja vuoden. Näin haluttujen toimipisteiden kuukausittaisia käyttötietoja voi tarkastella nopeammin sen sijaan, että määrittää aloitus- ja lopetuspäivämäärät kokonaistrendinäkymän tapaan.

Päivätrendi näyttää valitun päivän näkymän kellon aikoina käyttäen pylväsdiagrammin sijaan trendiviivaa. Päivämäärien valintalaatikoista on poissa käytöstä vasemmalta oikealle katsottuna kolme ensimmäistä numerolaatikkoa. Näin haluttujen toimipisteiden käyttötietoja voidaan tarkastella tietyltä päivältä trendiviivaa tarkastelemalla. Trendiviiva nousee aina yhden tunnin intervalleissa, jonka aikana käyttötieto kirjataan. Päivän ensimmäinen käyttötieto nostaa toimipisteen trendiviivaa aina 10 minuuttia ennen alkavaa tuntia arvoon 1. Jos saman tunnin aikana on tapahtunut esimerkiksi kolme käyttöä samassa toimipisteessä, nousee sen trendiviiva arvoon 3. Jos valitulta päivältä löytyy tuloksia, muuttuu löydetyn toimipisteen valintalaatikon taustaväri vaalean vihreäksi. Tämä parantaa selkeyttä ja havainnoi käyttäjää, jos tulos sisältää useamman kuin yhden toimipisteen. Katso esimerkki liitteestä 3, jossa kahden eri toimipisteen tulokset sattuvat päivätrendinäkymässä päällekkäin.

Jokaista trendinäkymää on mahdollista zoomata X-akselin suuntaisesti käyttämällä hiiren rullaa. Näin tuloksia pääsee seuraamaan nopeasti halutulta etäisyydeltä ilman, että päivämäärälaatikoihin tarvitsee koskea. Tämä ominaisuus on hyödyllinen myös silloin, jos tuloksia on niin paljon, etteivät ne mahdu visuaalisesti samalle aikavälille jolloin ohjelmisto käyttää tulosten esittämiseen pidempää intervallia.

Trendinäkymän Tiedosto-valikosta löytyy painikkeet Virkistä ja Sulje. Näistä painikkeista on tietoa luvussa 3.4.5.

### 3.4.4 Tulokarsinta

Päänäkymän Näytä-valikko tarjoaa käyttäjälle edistyneempiä työkaluja yrityksen käyttöjen seuraamiselle. Näytä-valikon toisena toimintona löytyy Tulokarsinta. Komento avaa eteen uuden ikkunan, jossa käyttäjä voi karsia listaa pienemmäksi käyttämällä joko aikaväliä, konetyyppejä, toimipistettä tai kaikkia näitä yhteensä. Alla on esitetty taulukko 4 selkeyttämään tulokarsintanäkymän valikkorakennetta.

Taulukko 4. Tulokarsintanäkymän valikkorakenne.

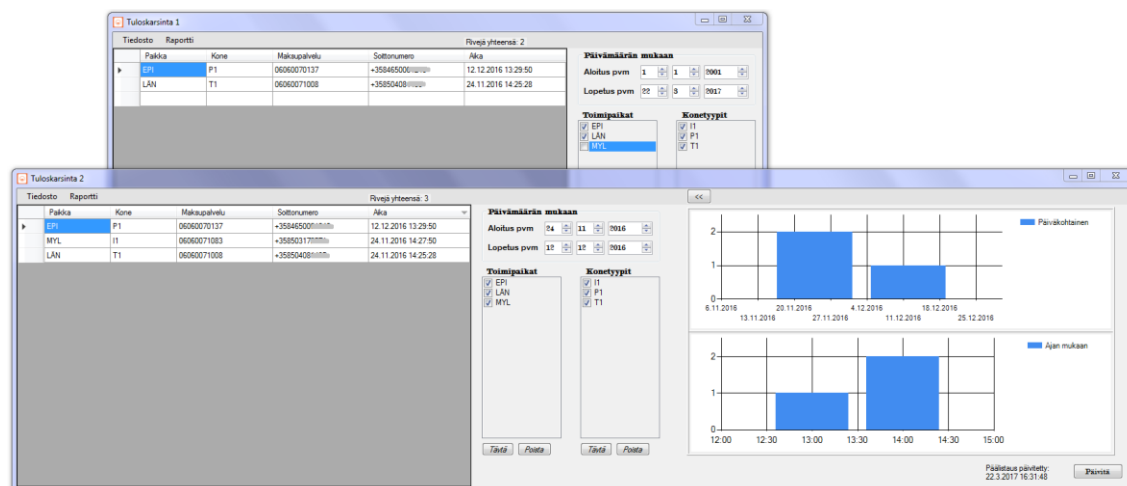
Tulokarsinta	
Tiedosto	Raportti
Tallenna	Näytä yhteenvetotrendi
Tallenna CSV	Luo raportti
Tallenna PDF	Avaa raportti, kun se on tallennettu
Virkistä	
Sulje	

Jos päänäkymän lista on päivitetty, näkyy viimeisin muokauspäivämäärä Tulokarsinta-ikkunan oikeassa alareunassa, Päivitä-painikkeen vieressä. Tieto on ilmoitettu kellon ajan tarkkuudella. Tämä tieto näkyy myös silloin, jos päänäkymän lista ei syystä tai toisesta ole tuottanut tuloksia tai tulosten päivittämisessä on tapahtunut virhe. Ohjelmisto lukee viimeisimmän muokauspäivämäärän juuripolkuun tallentuneesta CSV-tiedostosta trendinäkymän tavoin. Tästä tiedostosta on lisää tietoa aikaisemmassa luvussa 3.4.1.

Tulokarsintaikkuna on mahdollista avata päänäkymästä niin monta kertaa, kuin on tarve. Tämä mahdollistaa useiden karsittujen tulosten vertailemisen samaan aikaan. Sivulla 44 olevassa kuvassa 13 on avattu kaksi tulokarsintaikkunaa. Kuten kuvasta huomataan, kummallakin ikkunalla on oma numero ikkunan otsikon jälkeen, jonka avulla navigointi ikkunoiden välillä on havainnollisempaa. Ikkunoiden numerointi noudattaa nousevaa järjestystä aina korkeimmasta ikkunanumerosta ylöspäin. Jos esimerkiksi kuvan 13 ikkunoiden viereen avataan uusi tulokarsintaikkuna, saa se numeron kolme. Jos äsken mainitun toiminnon jälkeen Tulokarsinta 2 poistetaan ja käyttäjä päättää avata uuden tulokarsintaikkunan, saa uusi ikkuna arvon 4. Näin vältetään mahdollinen



sekaannus, jossa ikkunoiden 1 ja 3 väliin luotu, uusi ikkuna saattaa luoda harhaan johtavan mielikuvan vanhasta, poistetusta ikkunasta.



Kuva 13. 24 Pesula Seuranta. Ohjelmistoversio 0.2.6 (Alfa), Tuloskarsinta 1 ja 2.

Yllä olevassa kuvassa näkyvät tuloskarsintaikkunat ovat erinäköiset. Kun tuloskarsintaikkunan avaa päänäköymästä, avautuu se ilman yhteenvetotrendiä, kuten yllä olevan kuvan Tuloskarsinta 1. Tuloskarsinta 1 -ikkunan oikeassa ylänurkassa näkyy päivämäärälaatikot aloitus- sekä lopetuspäivämäärille, joita on mahdollista asettaa haluaman tarkasteluajanjakson mukaisesti. Oletusarvona laatikoissa lukee 1.1.2001. Jos ohjelma löytää juuripolusta päänäköymän tallentaman CSV-tiedoston, määräytyy lopetuspäivämäärän arvoksi CSV-tiedoston viimeisin muokkauspäivä kellon ajan tarkkuudella. Ohjelma saattaa kaatua, jos se ei löydä kyseistä tiedostoa tarvittaessa.

Päivämäärälaatikoiden alapuolella on toimipaikkavalinta ja konetyyppivalinta. Jos Päänäköymän lista on tuottanut tuloksia, näkyy tuloskarsintaikkunassa kaikki toimipaikat ja konetyypit, jotka löytyivät päänäköymän listalta. Valintalaatikot on automaattisesti merkitty aktiivisiksi. Jos valintalaatikon merkki poistetaan, jättää listaus poistetun toimipaikan tai konetyypin huomiotta. Kun tuloskarsintänäköymän lista päivitetään, tulostuu listaan ainoastaan ne toimipaikat ja konetyypit, jotka jäivät valituksi. Kummassakin valintalistassa on oltava vähintään yksi toimipaikka ja konetyyppi valittuna, jotta tuloskarsinnan lista antaa tuloksen.

Paikka- ja konevalintojen alla on kummassakin omat painikkeet, joissa lukee Täytä ja Poista. Täytä-painiketta painamalla painikkeen yllä olevan valintalistan kaikki valinta-

laatikot pakotetaan aktiivisiksi ja Poista-painikkeella toiminto on päinvastainen. Näin nopeutetaan asetusten asettelua. Toiminnon tarpeellisuus tulee esimerkiksi tilanteessa, jossa kaikki valintalaatikot ovat aktiivisia ja käyttäjän täytyy asettaa kumpikin valintalista niin että, vain yksi valintalaatikko on aktiivinen.

Kun käyttäjä on määrittänyt tarvittavat asetukset, painetaan oikeassa alareunassa olevaa painiketta, jossa lukee Päivitä. Windowsin työkalurivi ilmoittaa listan päivityksestä sekä päivityksen valmistumisesta, minkä jälkeen lista on nähtävissä tuloksarsintaikkunan DataGridView-näkymässä. Tämän jälkeen käyttäjä voi tarvittaessa suorittaa seuraavia tuloksarsintatoiminnon lisäominaisuuksia:

- Tuloksarsintaikkunan Tiedosto-valikosta löytyy Tallenna-painike, jolla listatun näkymän voi tallentaa kokonaisuudessaan PDF- tai CSV-tiedostona myöhempää tarkastelua varten. Päänäkymän tallennusominaisuuteen nähden tuloksarsinnasta on hyötyä, jos halutaan koko listauksen sijaan valita tallennettavat rivit.
- Tuloksarsintaikkunan Raportti-valikosta löytyy ensimmäisenä painike Näytä yhteenvetotrendi. Tämä painike levittää tuloksarsintaikkunaa oikealle paljastaen kaksi trendiä. Ylemmässä trendissä näkyy yhteenlasketut käytöt päiväkohtaisesti ja alemmassa suosituimmat kellonajat yhteenlaskettujen käyttötietojen perusteella (kts. Kuva 13).
- Tuloksarsintaikkunan Raportti-valikosta toisena on painike Luo raportti. Tämä painike avaa eteen dialogin, josta käyttäjä valitsee tallennuspaikan raportille. Raportti tallentuu PDF-muodossa ja sisältää tuloksarsinnan esittämät kustannuspaikat, konetyyppien käyttökerrat ja yhteenvetotrendit. Jokaisesta raportista ilmaantuu tulosten lisäksi raportin tarkka luontiaika sekä seurantaohjelmiston versio.

Yllä olevassa listassa mainittu yhteenvetotrendi sisältää ominaisuuden, jolla trendit on mahdollista siirtää oikeasta reunasta vasemmalle DataGridView-näkymän päälle. Tämä onnistuu trendien yläpuolella olevasta painikkeesta, joka sisältää kaksi peräkkäistä pienempi kuin -merkkiä (<<). Painikkeen merkit kääntyvät toisinpäin osoittaen, että trendit voi palauttaa takaisin ikkunan oikeaan reunaan tätä painiketta painamalla. Kun trendit ovat vasemmalla, saadaan ikkunan koko pienennettyä esimerkiksi tilanpuutteen takia tai helpottamaan tulosten vartailua.

Tuloksarsintaikkunan Tiedosto-valikosta löytyy painikkeet Virkistä ja Sulje. Näistä painikkeista on tietoa seuraavassa luvussa 3.4.5.

### 3.4.5 Lisätietoja

Trendinäköymän ja tulostokarsinnan tiedostovalikosta löytyy Virkistä-painike. Tämä toiminto on kätevä ominaisuus, jos päänäköymän lista on päivitetty uudestaan trendinäköymän tai tulostokarsinnan ollessa auki. Tällöin auki oleva ikkuna saadaan virkistettyä käyttämään viimeisintä listatietoa ja sen asetukset palautuvat oletusarvoihin eikä koko ikkunaa tarvitse tämän takia sulkea ja avata uudestaan päänäköymän Näytä-valikosta.

Tiedosto-valikosta löytyy myös Sulje-painike edellä mainituille ikkunoille sekä konelistauskelle. Toisin kuin päänäköymässä, tämä painike ei sulje koko ohjelmistoa, vaan ainoastaan sen ikkunan, jonka Tiedosto-valikosta Sulje-painiketta painetaan.

Päänäköymän Tiedot-valikosta pääsee tarvittaessa tarkistamaan ohjelmiston versiohistorian, josta selviää yksinkertaisena listana ohjelmiston versiovaiheiden kehitys. Listasta on yksityiskohtainen selvitys tämän opinnäytetyön luvusta 3.3. Tiedot-valikosta on mahdollista lukea ohjeet, joihin myös tämä luku (3.4) perustuu. Ohje on esitetty pelkkänä tekstinä, mutta ohjeen osat on lokeroitu välilehtiin, jolloin tarvittava tieto on nopeasti löydettävissä.

Jokaisen listauksen yllä on merkattu kerättyjen rivien lukumäärä, kun päivitys on suoritettu. Päänäköymässä tämä luku on todennäköisesti pienempi, kuin lokikansion tiedostojen lukumäärä. Syy johtuu siitä, ettei kaikki lokikansion sisältämä tieto vastaa konelistauksen tietoja. Tästä saa tarkemman käsityksen luvun 3.4.1 kohdassa, jossa esitetään päivitysprosessin toimenpiteet.

Ohjelmisto on kehitetty helpottamaan tietynlaisten XML-tiedostojen lukemista eikä sitä saa käyttää kaupallisessa tarkoituksessa. Ohjelmistossa saattaa esiintyä ominaisuuksia, jotka ovat vielä kehitysvaiheessa tai poissa käytöstä. Myös odottamattomat ja en-tuudestaan tuntemattomat ohjelmistovirheet voivat olla mahdollisia.

## 4 Yhteenveto ja mietteet

24 Pesula -yrityksen tulevat muutokset ohjelmistoratkaisujen ja hardware-vaihdosten puolella ratkaisevat paljon, tullaanko seurantaohjelmistoa vielä jatkamaan. Jos ohjelmisto jatkaa Beeta-vaiheeseen, joudutaan sen toimintamuotoa mahdollisesti uusimaan. Näillä näkymin ohjelmiston Beeta-vaihe tullaan näkemään vain, jos ohjelmistolle löytyy paikka vielä tulevaisuuden muutosten jälkeen.

Suurin voitto koko projektissa on ollut ohjelmointitaitojen oppiminen. Teoreettiset asiat, kuten matematiikka eivät olleet ikinä vahvimmasta päästä. Ajatus minkään ohjelmointikielen oppimisesta oli todella kaukainen, ellei jopa mahdoton. Monista yrityksistä huolimatta, lähtien liikkeelle Salon seudun ammattiopiston alkeisteoriaopinnoista, oli vain vahvistus sille, ettei ohjelmointi tulisi välttämättä ikinä luonnistumaan. Tämä projekti kuitenkin onnistui heittämään kaikki vanhat luulot syrjään. Osoittautui, että jos muita vaihtoehtoisia työmuotoja ei löydy ja aikaa annetaan tarpeeksi, niin mikä tahansa näyttää olevan mahdollista. Mikä voisi olla sen hienompi idea opinnäytetyöksi, kuin osoittaa onnistuneensa jossakin, jonka koki mahdottomaksi?

Tämä opinnäytetyö mahdollisti myös vanhojen aloittelijavirheiden korjaamisen ohjelmiston koodissa. Virheet ilmaantuivat tiedonkeruun aikana. Muun muassa listausprosessi koki suuria muutoksia opinnäytetyön kirjoittamisen yhteydessä. Oli lähes järkyttävää nähdä, miten päänäköymän listausprosessi sisälsi useita kymmeniä rivejä turhaa, laskentatehoa vieviä toimenpiteitä, joiden tarkoitus oli vain varmistaa, että turhat soittot numerot jäävät pois lopullisesta listausprosessista. Myös itse listausprosessi vaatii oman osuutensa prosessitehosta. Kun listausmenetelmä uudistui, parani myös koko ohjelmiston hyötysuhde merkittävästi.

Hyötysuhdeparannus otettiin käyttöön 24 Pesulan seurantaohjelmiston versiossa 0.2.8, josta ei edellisen version lisäksi ole enää suoranaisesti mainintaa tässä opinnäytetyössä.

## Lähteet

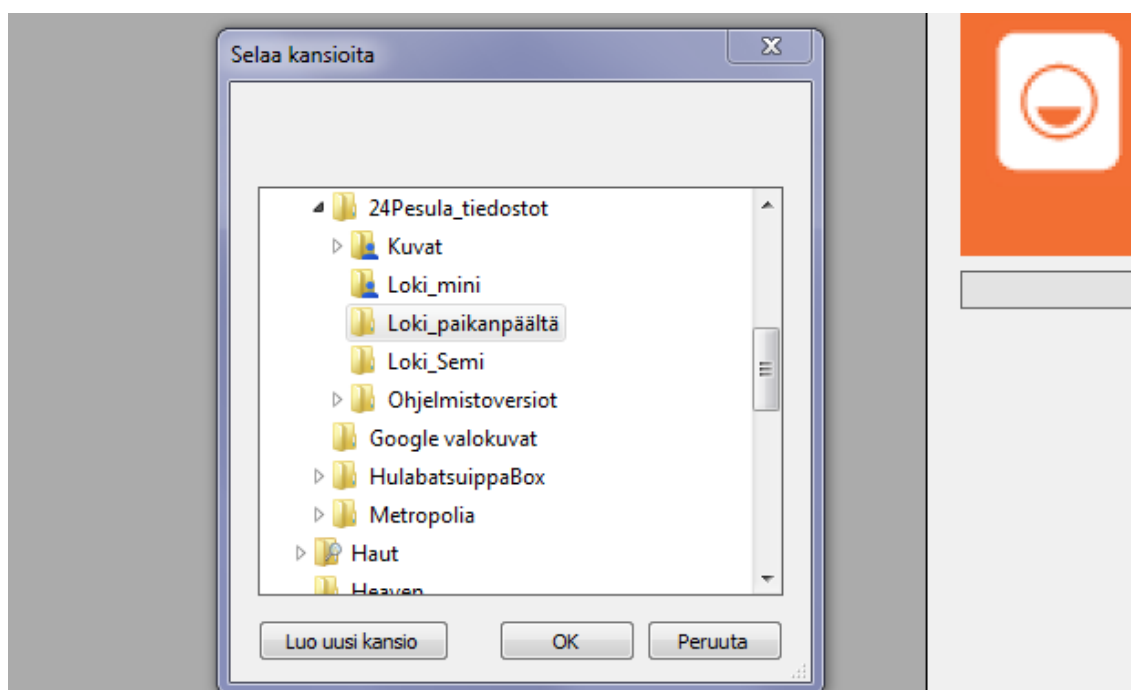
- 1 Nevala. 10.3.2003. Nevalan veljekset halusivat rahoittaa autoiluaan ja päätyivät pesulaan. Kauppalehti. Luettu 15.2.2017.
- 2 Nevala, Nico. 2017. Sähköpostiviesti. Luettu 13.1.2017.
- 3 Nevala, Asko. 2016. 24 Pesula Britannian markkinoille! Yrityksen sisäinen dokumentti. Luettu 7.4.2017.
- 4 Lumme, Tristan. 2016. Työharjoitteluraportti. Luettu 7.4.2017.
- 5 Semantic Versioning 2.0.0. Verkkodokumentti. Preston-Werner, Tom. <<http://semver.org/>>. Luettu 18.4.2017.
- 6 Moghadampour, Ghodrat. 2009. C# -ohjelmointi. Jyväskylä, WSOYpro Oy. Luettu 12.4.2017.
- 7 Nevala, Asko. 24 Pesula IOT. Sähköposti. Luettu 22.3.2017.
- 8 Microsoft. File.ReadAllText Method (String). <[https://msdn.microsoft.com/en-us/library/ms143368\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/ms143368(v=vs.110).aspx)>. Luettu 7.4.2017.
- 9 HTML ISO-8859-1 Reference. Verkkodokumentti . W3Schools. <[https://www.w3schools.com/charsets/ref\\_html\\_8859.asp](https://www.w3schools.com/charsets/ref_html_8859.asp)>. Luettu 7.4.2017.
- 10 2.4.4.4 Character literals. Verkkodokumentti. Microsoft. <[https://msdn.microsoft.com/en-us/library/aa691087\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa691087(v=vs.71).aspx)>. Luettu 10.4.2017.
- 11 String.Split Method. Verkkodokumentti . Microsoft. <[https://msdn.microsoft.com/en-us/library/system.string.split\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.string.split(v=vs.110).aspx)>. Luettu 10.4.2017.
- 12 Var (C# Reference). Verkkodokumentti . Microsoft. <<https://msdn.microsoft.com/fi-fi/library/bb383973.aspx>>. Luettu 11.4.2017.
- 13 String.Substring Method (Int32, Int32). Verkkodokumentti. Microsoft. <[https://msdn.microsoft.com/en-us/library/aka44szs\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/aka44szs(v=vs.110).aspx)>. Luettu 11.4.2017.

- 14 DataGridView.Rows Property. Verkkodokumentti. Microsoft.  
<[https://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview.rows\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview.rows(v=vs.110).aspx)>. Luettu 12.4.2017.
- 15 Splash Screens the Right Way. Verkkodokumentti . Big Nerd Ranch.  
<https://www.bignerdranch.com/blog/splash-screens-the-right-way/>>. Luettu 12.4.2017.

## XML-lokikansion valinta

24 Pesulan seurantaohjelmiston päänäköymän tiedostovalikosta valitaan Avaa XML -kansio. Tämä avaa alla näkyvän dialogin, josta käyttäjä voi määrittää lokikansion sijainnin. Tämän kansion avulla ohjelmisto löytää listauksessa tarvittavat käyttötiedot.

Alla olevassa tilanteessa ohjelmistolle valitaan kansio, joka on kopioitu yrityksen palvelimelta ohjelmiston testikäyttöä varten.



## Konelistauksen CSV-tiedoston ulkoasu

Alla on esitetty konelistauksen käyttämän CSV-tiedoston ulkoasu. Ylintä riviä ei hyödynnetä ohjelmiston käytössä, koska sarakkeiden otsikot ovat jo valmiiksi määritetty ohjelmiston DataGridView-näkymään. CSV-tiedoston otsikot ovat hyödyllisiä, kun listaa tutkitaan käyttämällä muita ohjelmia.

CSV-tiedoston muokkaamisessa on huomioitava, että puolipisteitä on joka rivillä sama määrä, oli rivin sarakkeella tietoa tai ei. Kuvasta nähdään, että yksi soittnumero puuttuu, jolloin se on voitu jättää kirjoittamatta ja rivin päättää puolipiste. Mutta jos esimerkiksi maksupalvelun numeroa ei tiedetä ja soittnumero tiedetään, on konetyypin ja soittnumeron väliin kirjoitettava kaksi puolipistettä peräkkäin, jotta näiden kahden tiedon väliin jää yksi tyhjä ruutu.

Soittnumerot on esitettävä tarkalleen alla näkyvän kuvan mukaisesti, jotta ne ovat identtiset XML-tiedostojen sisältämien soittnumeroiden kanssa. Soittnumerot on jouduttu osittain sensuroimaan luottamuksellisista syistä.

1	Paikka;Kone;Maksupalvelu;Soitto nro
2	KOK;M1;06060071014;+35850342
3	KOK;M2;06060061882;+35850436
4	KOK;M3;06060071026;+35850573
5	KOK;J1;06060070149;+358465001
6	KOK;P0;06060062085;+35850436
7	KOK;P1;06060071010;+35846922
8	KOK;P2;06060071068;+35850467
9	KOK;R1;06060070128;
10	KOK;T1;06060061425;+35850436
11	EPI;M1;06060061871;+35850523
12	EPI;M2;06060062084;+35850523
13	EPI;J1;06060061902;+35850523
14	EPI;P1;06060070137;+358465000
15	EPI;P2;06060070138;+358465000
16	EPI;P3;06060070139;+358465000
17	EPI;R1;06060070142;+358465001
18	EPI;R2;06060070126;+358465000
19	EPI;T1;06060061835;+35850523
20	YLÖ;M1;06060070121;+35846921
21	YLÖ;M2;06060071015;+35850463



## Päällekkäisyys päiväkohtaisessa trendissä

Alla on esitetty tilanne, jossa trendinäkymässä on valittu käytettäväksi päivänäkymää ja päivämääräksi on asetettu 24.11.2016.

Ikkunan oikeassa laidassa on havaittavissa, että kaikki kolme toimipistettä on valittu, mutta ainoastaan toimipisteiden LÄN ja MYL valintalaatikoiden taustaväri on vaalean vihreä. Tämä tarkoittaa sitä, että päivämäärältä 24.11.2016 löytyy vain näiden kahden toimipisteen tulokset. Trendinäkymässä on kuitenkin havaittavissa ainoastaan punaista väriä edustavan MYL-toimipisteen käyttötieto.

Koska toimipisteellä LÄN on samalta päivältä ja saman tunnin ajalta yhtä paljon käyttöä, kuin toimipisteellä MYL, on se jäänyt jälkimmäisenä piirretyn viivan alle. Tilanteen voi tarkistaa poistamalla valintalaatikoista merkki kohdasta MYL ja päivittää ikkuna sen jälkeen uudestaan. Tällöin trendinäkymässä on huomattavissa samanlainen trendiviiva, mutta sen väri on keltainen.

