Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2021

Aki Rönnkvist

TUOTANNONHALLINTA-JÄRJESTELMÄN KEHITYS ELEKTRONIIKKA-VALMISTAJALLE

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2021 | 36 sivua

Aki Rönnkvist

TUOTANNONHALLINTAJÄRJESTELMÄN KEHITYS ELEKTRONIIKKAVALMISTAJALLE

Elektroniikkalaitteiden valmistajan on halutessaan CE-merkinnän laadittava laitetta koskevat tekniset asiakirjat, joiden perusteella voidaan arvioida vastaako laite kaikkia sitä koskevia vaatimuksia. Tämä opinnäytetyö on tehty yhteistyössä Pietiko Oy:n kanssa, jossa CE-merkinnän vaatimat tiedot on aina kirjattu Excel-tiedostoon. Työn tarkoitus oli löytää uusi tuotannonhallintajärjestelmä, joka korvaisi Excelissä olevan tuotantotietokannan, paperiset koontaohjeet ja hankalaksi käyneen varastokirjanpidon.

Työssä tutustuttiin tarkemmin nykyisiin tuotannon käytäntöihin ja selvitettiin niiden puutteet. Nykyisen järjestelmän puutteiden ja ongelmien selvittyä tutkittiin suosituimpia tuotannonhallinta-järjestelmiä ja niiden hyviä ja huonoja puolia. Valmiista järjestelmistä selvisi, että ne on suunnattu enemmän isoille yrityksille ja konserneille, ja sen myötä ne ovat hyvin kalliita Pietikon kaltaiselle pienelle yritykselle. Täysin uuden järjestelmän kehitystä verrattaessa valmiiden järjestelmien käyttöönottoon ja konfigurointiin selvisi, että työmäärä saattaisi olla hyvinkin samaa luokkaa, mutta ylläpito tulisi huomattavasti kalliimmaksi.

Pohdinnan jälkeen tultiin siihen tulokseen, että paras valinta yritykselle on täysin uuden järjestelmän kehitys. Näin tulevaisuudessa säästetään rahaa ja saadaan järjestelmästä täysin yritykselle räätälöity. Uuden järjestelmän pohjaksi valittiin Django-sovelluskehys, sillä se takaa työkalut kevyen järjestelmän kehitykseen yrityksen sisäiseen käyttöön.

Uuteen järjestelmään kehitettiin tilauksenhallinta, varastokirjanpito, tuotantotietokanta ja digitaaliset koontaohjeet. Suurin osa vanhan järjestelmän puutteista pystyttiin korjaamaan ja tuotannon työvaiheita optimoimaan. Uuden järjestelmän pitäisi tulevaisuudessa säästää aikaa ja manuaalista työtä yrityksen tuotannon prosessissa.

ASIASANAT:

Elektroniikkateollisuus, kehitysalustat, järjestelmäsuunnittelu, ohjelmistokehitys, sovelluskehykset

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2021 | 36 pages

Aki Rönnkvist

DEVELOPMENT OF MANUFACTURING CONTROL SYSTEM FOR ELECTRONICS MANUFACTURER

If manufacturers of electronic equipment wish to affix the CE marking to their product, they must create technical documentation concerning the equipment to assess whether the equipment satisfies all the relevant requirements. This thesis was carried out in collaboration with Pietiko Ltd, where the information required by the CE marking has always been recorded in an Excel file. The purpose of this thesis was to find a new production management system that would replace the production database in Excel, the paper-based assembly instructions, and the cumbersome inventory accounting.

In the thesis, the current production practices were studied in more detail and their shortcomings were clarified. Once the shortcomings and problems of the current system were identified, the most popular production management systems and their advantages and disadvantages were examined. The ready-made systems proved to be more targeted at large companies and corporations, making them very expensive for a small company such as Pietiko. After comparing the development of a completely new system with the deployment and configuration of ready-made systems, it became clear that the workload could be in the same range, but maintenance would become significantly more expensive.

After reflection, it was concluded that the best choice for the company is the development of a completely new system. This will save money in the future and make the system fully customized for the company. The Django application framework was chosen as the basis for the new system, as it provides tools for development of a lightweight system for internal use in the company.

Order management, inventory accounting, production database and digital assembly instructions were developed for the new system. Most of the shortcomings of the old system were remedied and the manufacturing steps were optimized. The new system should save time and manual work in the company’s production process in the future.

KEYWORDS:

Electronics industry, development platforms, system design, software development, application frameworks

SISÄLTÖ

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET TAI SANASTO

CE-merkintä	Merkintä, jolla tuotteen valmistaja tai valtuutettu edustaja vakuuttaa, että tuote täyttää tuotetta koskevien EU:n direktiivien ja asetusten olennaiset vaatimukset.
ERP	Yrityksen tietojärjestelmä, joka integroi eri toimintoja, esimerkiksi tuotantoa, jakelua, varastonhallintaa, laskutusta ja kirjanpitoa.
Hostaus	Palvelu, jossa käyttäjä vuokraa verkkosivujaan varten tarvittavan verkkopalvelintietokoneen kiintolevytilan, tietoliikennekaistan, palvelinohjelmistot sekä niihin liittyvän ylläpidon.
Loggerijärjestelmä	Järjestelmä, joka koostuu keskusyksiköstä sekä langattomista lähettimistä, joilla voidaan mitata eri suureita.
MRP	Manufacturing Resource Planning, Tuotannonohjaukseen käytettävä materiaalinhallintajärjestelmä.
Rajapintakutsu	Rajapinta mahdollistaa integraation ohjelmistoon. Rajapinnan avulla voidaan tehdä pyyntöjä ohjelmistolle, josta halutaan noutaa, tai tuoda tietoja.
Sisällönhallintaohjelmisto	Sisällönhallintaohjelmistolla (englanniksi CMS, eli Content Management System) voidaan rakentaa dynaaminen ja responsiivinen verkkosivu helppokäyttöisellä käyttöliittymällä.
Sovelluskehys	Muodostaa rungon sen päälle rakennettavalle tietokoneohjelmalle. Sovelluskehys on ohjelmoinnin apuväline, jonka tarkoituksena on nopeuttaa uusien ohjelmistotuotteiden valmistusta.
URL	Uniform Resource Locator. Yksilöllinen osoite internetissä olevalle tiedostolle, tai verkkosivulle.

1 JOHDANTO

Elektroniikkalaitteiden valmistajan on halutessaan CE-merkinnän tarkastettava, että sähkölaite vastaa EU:n lainsäädännön olennaisia vaatimuksia. Valmistajan on laadittava sähkölaitetta koskevat tekniset asiakirjat, joiden perusteella voidaan arvioida vastaako laite kaikkia sitä koskevia vaatimuksia. (Wikipedia n. d.) Tämä opinnäytetyö on tehty yhteistyössä Pietiko Oy nimisen yrityksen kanssa, joka valmistaa elektroniikkalaitteita, joissa on CE-merkintä.

Pietiko on kirjannut vuosien ajan kaikki CE-merkinnän vaatimat tiedot ylös taulukkoon Microsoft Excel -ohjelmaan. Laitteiden valmistusohjeet ovat paperisena ja kaikki tietojen kirjaaminen tapahtuu manuaalisesti, joka vie paljon aikaa tuotannon prosessista. Jos tuotannon työntekijöiden aika saataisi käytettyä tehokkaammin itse laitteiden valmistukseen ja testaukseen tietojen kirjaamisen sijasta parantaisi se yrityksen toimitusaikoja ja sitä myötä myös myyntiä.

Yritys tarvitsee uuden tuotannonhallintajärjestelmän valmistettaville laitteille. Tarkoitus on selvittää, löytyykö olemassa olevista tuotannonhallintaohjelmistoista sellaista vaihtoehtoa, joka voisi sopia Pietikolle, vai onko täysin uuden yritykselle räätälöidyn järjestelmän kehitys tarpeellista.

Opinnäytetyö alkaa nykyisen järjestelmän läpikäynnillä, joka koostuu kolmesta osa-alueesta: tuotantotietokanta, koontaohjeet ja varastokirjanpito. Nykyisen järjestelmän puutteiden ja ongelmien selvittyä käydään läpi mahdollisia ratkaisuja, joihin kuuluu kolmannen osapuolen valmistaman tuotannonhallintajärjestelmän käyttöönoton tai täysin uuden järjestelmän kehitys. Nopein ja yksinkertaisin ratkaisu olisi valmiin järjestelmän käyttöönotto, mutta se voi koitua epäkäytännölliseksi suosittujen tuotannonhallintajärjestelmien kalliiden hintojen, tai niiden yritykselle yhteensopimattomuuden myötä.

Viimeisenä valittu tuotannonhallintajärjestelmä kehitetään toimimaan yrityksen tuotannossa ja tarkoitus on myös mitata, kuinka paljon aikaa uusi järjestelmä säästää tuotannon työvaiheista, ja sitä kautta arvioida, kuinka paljon tehokkaampi yrityksen tuotanto sen ansoista tulee olemaan.

2 NYKYINEN JÄRJESTELMÄ

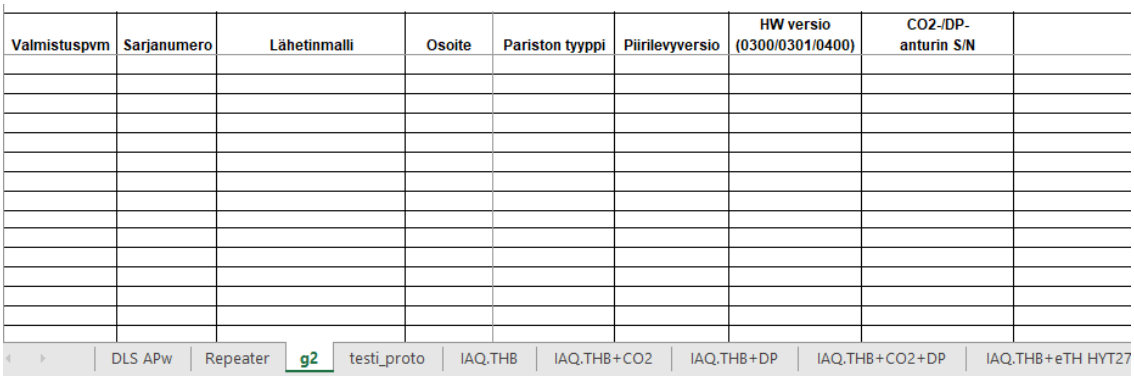
2.1 Tuotantotietokanta

Pietiko Oy toimi maahantuojana ilmanlaatumittauslaitteille melkein 30 vuotta, kunnes vuonna 2016 yritys päätti tehdä strategisen muutoksen ja aloittaa tuotekehitys- ja tuotantotoiminnan maahantuonnin ohelle. Aloitus oli hyvin maltillinen ja aluksi tuotteita olikin vain yksi. (Pietilä 2020.) Niin pienelle tuotannolle on turha hankkia kalliita isoille konserneille tarkoitettuja tuotannonhallintaohjelmia, joten päätös tehtiin pelkästä tietojen kirjaamisesta Microsoftin Excel-ohjelmaan, sillä yrityksellä oli jo Office 365 -paketti ja sitä myötä oli helppo pitää kaikki tiedot synkronoituina eri työpisteiden ja työntekijöiden välillä. Nykyään Pietiko valmistaa jo yli tusinaa eri tuotetta, joten uuden paremmin suunnitellun tuotantotietokannan tarve on suuri.

Vuosien myötä Excel-pohjaa on muokattu senhetkisten tarpeiden mukaiseksi, ja sen seurauksena eri aikakausien ja laiteversioiden tiedot saattavat olla eri muodossa, tai puuttua kokonaan. Kommenttien ja huomioiden, kuten asiakkaan laitteen ohjelmiston päivityksen tai laitteen vian diagnoosin kirjaamiseen ei myöskään ole mitään standardia, joten tietokanta voi osittain olla erittäin sekava.

Tiettyjen laitteiden tai tietyn asiakkaan laitteiden löytäminen on myös työlästä, sillä jokaiselle laitteelle on aikoinaan ollut omat Excel-taulukot, kunnes vuoden 2019 syksyllä uuden G2 (generation 2) laiteversion myötä päätettiin alkaa jatkossa kirjata Pietikon kehittämän Miran loggerijärjestelmän kaikkien 11 lähettimen tiedot samaan Excel-taulukkoon (Taulukko 1). Tämä tietyllä tapaa yksinkertaisti tietojen kirjaamisen, mutta samalla vanhat taulukot jäivät jäljelle, eikä niistä siirretty tietoja uuteen, joten asiakkaan tuodessa vanha laite korjaukseen tai kalibrointiin, työntekijä joutuu etsimään, onko tuote uudessa vai vanhassa taulukossa. Pietiko valmistaa loggerijärjestelmän lisäksi kahta muuta tuotetta, joilla on myös omat erilliset Excel-taulukkonsa, jotka aiheuttavat lisää sekaannusta.

Taulukko 1. Esimerkki nykyisin käytetystä Excel-taulukosta tietojen kirjaamiseen.



Valmistuspvm	Sarjanumero	Lähetinmalli	Osoite	Pariston tyyppi	Piirilevyversio	HW versio (0300/0301/0400)	CO2-/DP-anturin S/N	

DLS APw | Repeater | g2 | testi_proto | IAQ.THB | IAQ.THB+CO2 | IAQ.THB+DP | IAQ.THB+CO2+DP | IAQ.THB+eTH HYT27

2.2 Tuotteiden koontaohjeet

EU-direktiivin sähkölaitteissa vaatima CE-merkintä edellyttää sitä, että tuotteen tekniset asiakirjat ovat kunnossa ja vaatimusten mukaiset. Näihin asiakirjoihin kuuluu muun muassa suunnittelu- ja valmistuspiirustukset sekä kaaviot osakokoonpanoista, piireistä jne. ja tarvittavat kuvaukset ja selitykset piirustusten ja kaavioiden ymmärtämiseksi. (Tukes 2014 s. 12.)

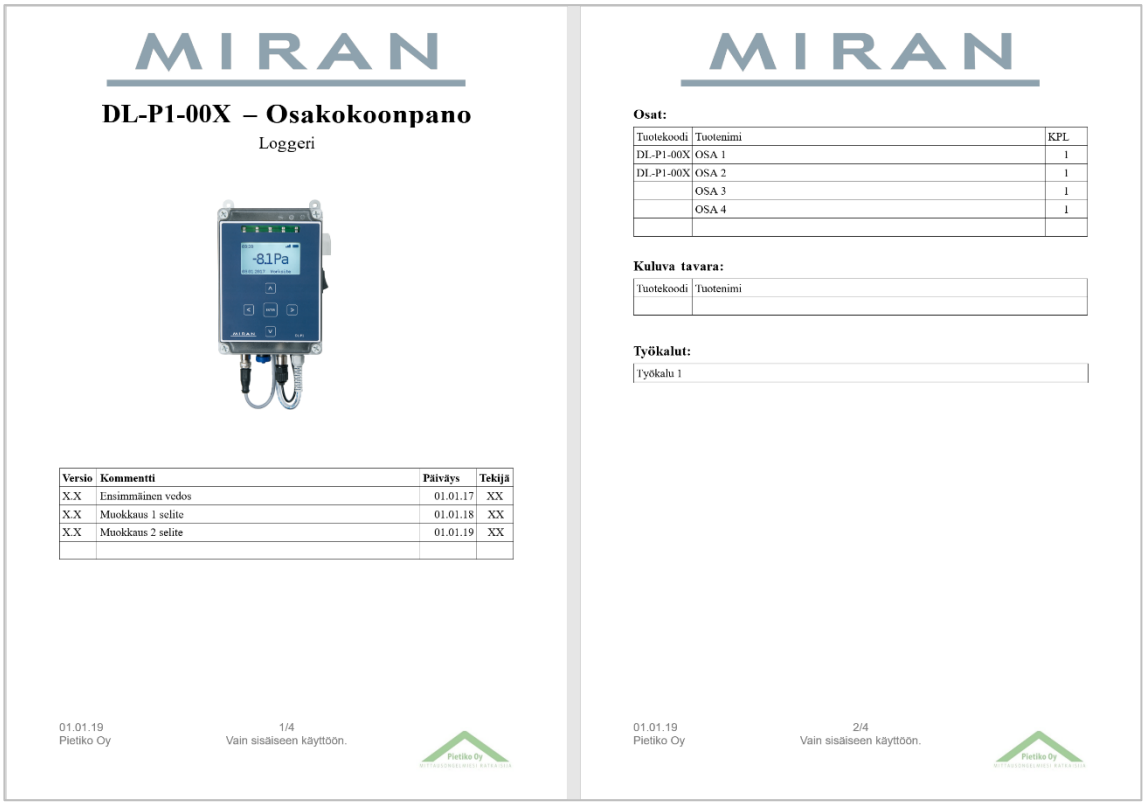
Käytännössä koontaohjeiden osalta tämä tarkoittaa sitä, että jokaisesta laitteesta vaaditaan dokumentti, joka sisältää osaluettelon ja tarkat kuvaukset siitä, miten laite valmistetaan alusta loppuun saakka. Ohjeiden tulisi olla sellaiset, että kuka tahansa tarvittavat taidot omaava henkilö osaisi koota laitteen vaadittavilla työkaluilla. Aikaisemmin mainittu vaatimus tuotantotietokantaan kirjattavasta laiteversiosta pätee myös koontaohjeisiin, joten dokumentissa täytyy olla kirjattuna koontaohjeen versionumero ja jokaisessa versiossa tehdyt muutokset.

Pietikon tuotannossa on panostettu koontaohjeisiin ja ne ovat olleet hyvin kattavat. Ohjeissa on kaikissa sama yhteinen pohja, josta löytyy kaikki direktiivien vaatimat tiedot, ja ne ovat myös yrityksen yleisen ilmeen mukaiset logoineen ja värimaailmoineen (Kuva 1). Ohjeet on kirjattu perinteiseen tapaan tekstinkäsittelyohjelmalla, ja sen jälkeen tulostettu paperille. Niissä on hyvin kuvia ja niiden mukana tekstiä havainnoimaan jokaista työtehtävää.

Ongelmaksi on kuitenkin muodostunut koontaohjeiden säännöllinen päivitys ajan myötä. Yleensä uuden laiteversion myötä tulee jotain parannuksia laitteiden koontavaiheeseen, kuten esimerkiksi tietyn sensorin käsin juottaminen voidaan poistaa työvaiheista, sillä se

saadaankin juotettua jo piirilevyjen valmistajan toimesta. Tässä tapauksessa koontaohje täytyy tietenkin päivittää, ja se onkin usein helppo muistaa, sillä uuden laiteversion myötä myös tuotannon työntekijät joutuvat opettelemaan uudet valmistusvaiheet. Ongelma syntyy siinä vaiheessa, kun eteen tulee pienempiä muutoksia, kuten esimerkiksi jokin laitteessa havaittu virhe, joka edellyttää lisätoimenpiteitä valmistusvaiheessa, jotta virhettä ei asiakkaalla laitteessa ilmenisi. Monesti pienemmät muutokset kommunikoidaan vain suullisesti, eikä niitä muisteta kirjata koontaohjeeseen, joka aiheuttaa ongelmia, jos yritykseen tulee uusia tuotannon työntekijöitä, tai jos muutoksesta on ilmoitettu aikana, jolloin kaikki tuotannon työntekijät eivät ole olleet paikalla.

Paperisessa muodossa on myös havaittu puutoksia, kuten se ettei erittäin pienien muutosten takia viitsisi aina tulostaa uutta ohjetta ja myös se, että pelkät kuvat ja tekstit eivät aina havainnoi työvaihetta täydellisesti. Osa työvaiheista on sellaisia, että niitä on hyvin vaikea kuvailla tekstimuodossa niin, että työntekijä ymmärtäisi täysin mitä haetaan takaa ja kuvakin on vain yksittäinen hetki työvaiheesta, eikä se aina ole tarpeeksi.



MIRAN

DL-P1-00X – Osakokoonpano

Loggeri

Versio	Kommentti	Päiväys	Tekijä
X.X	Ensimmäinen vedos	01.01.17	XX
X.X	Muokkaus 1 selite	01.01.18	XX
X.X	Muokkaus 2 selite	01.01.19	XX

01.01.19 Pietiko Oy — 1/4 Vain sisäiseen käyttöön.

MIRAN

Osat:

Tuotekoodi	Tuotenimi	KPL
DL-P1-00X	OSA 1	1
DL-P1-00X	OSA 2	1
	OSA 3	1
	OSA 4	1

Kuluva tavara:

Tuotekoodi	Tuotenimi

Työkalut:

Työkalu 1

01.01.19 Pietiko Oy — 2/4 Vain sisäiseen käyttöön.

Kuva 1. Esimerkki laitteen koontaohjeen kannesta ja osaluettelosta.

2.3 Varastokirjanpito

Pietiko Oy käyttää kirjanpito-ohjelmistoa, johon sisältyy myös varastonhallinta. Kirjanpito-ohjelmistossa on mahdollista liittää valmistettavaan tuotteeseen alituotteita, eli tuotteen osia, jotka kuluvat aina kun tuotetta valmistetaan. Näin teoriassa on helppoa seurata valmistettavien tuotteiden osavarastoa ja varmistaa, ettei tuotteiden osat lopu.

Käytännössä asia on kuitenkin huomattavasti monimutkaisempi, sillä kirjanpito-ohjelmiston käyttöliittymä on hieman sekava ja jokaisen laitteen ja osan varastotilanteen löytäminen on vaivalloista. Tämän seurauksena valitettavan usein käy niin, että vasta laitetta valmistaessa käy ilmi, että jokin osista on loppunut tai loppumaisillaan eikä laitteita saada valmistettua tarvittavaa määrää ennen kuin osia saadaan tilattua lisää, mikä puolestaan hidastaa tuotantoa ja antaa huonon vaikutelman asiakkaille. Kirjanpito-ohjelmistossa on mahdollista asettaa hälytysrajoja tuotteille, jolloin ohjelma hälyttää, kun tuotetta on jäljellä enää tietty määrä, mutta kyseinen toiminto ei ole koskaan toiminut.

Tuotannossa valmiiden tuotteiden kirjaus täytyy tehdä kirjanpito-ohjelmistoon manuaalisesti, ja se vaatii sisäänkirjautumisen. Sisäänkirjautuminen tapahtuu joko pankkitunnuksilla tai mobiilivarmenteella. Valmistuskirjaus on hyvin monimutkainen ja joka kerta erikseen kirjautuminen pankkitunnuksilla, tai mobiilivarmenteella vie aikaa, mutta jos sen haluaa välttää, voi myös pyytää muita ohjelmistoon jo kirjautuneita työntekijöitä kirjaamaan laitteet, mutta silloin on aina riski, että kirjaus unohtuu tai tulee jokin muu väärinymmärrys. Näin ollen varastotilanne ei aina vastaa todellisuutta ja myös alituotteiden määrä on monesti virheellinen.

3 JÄRJESTELMÄVAIHTOEHDOT

Tämän opinnäytetyön tarkoitus on kehittää uusi toimivampi järjestelmä varastonhallintaan ja tuotantotietokantaan. Järjestelmään on lähtökohtaisesti kaksi vaihtoehtoa: valmis ratkaisu kolmannelta osapuolelta, joka muokataan yritykselle sopivaksi, tai itsekehitetty järjestelmä alusta loppuun. Opinnäytetyön ensimmäinen vaihe koostuu näiden kahden vaihtoehdon tarkemmasta tutkinnasta ja sopeutuvuudesta yrityksen tarpeisiin.

3.1 Valmiit tuotannonhallintajärjestelmät

Tuotannon tietojärjestelmiä on olemassa valmiina paketteina ja niitä kutsutaan nimellä Manufacturing Resource Planning (MRP). MRP-ohjelmistot sisältyvät useasti suurempaan kokonaisuuteen nimeltä Enterprise Resource Planning (ERP). Aiemmin yrityksillä oli jonkinlainen kirjanpito, taloushallinto tai HR-prosessi, mutta ohjelmistojärjestelmät toimivat usein toisistaan erillään. Moderni ERP-ohjelmisto sen sijaan yhdistää kaikki nämä prosessit muodostaen yhden mukautuvan järjestelmän. ERP-ohjelmistot ovat suurimmaksi osaksi suunniteltu suurille yrityksille ja siksi ne ovat myös hyvin kalliita. (Microsoft n. d.)

ERP-järjestelmä eli toiminnanohjausjärjestelmä on koko yrityksen tietojärjestelmä, joka integroi eri toimintoja, esimerkiksi tuotantoa, varastonhallintaa, reskontraa ja kirjanpitoa. Tyypillistä on, että nykyaikaisissa järjestelmissä osiot ovat siis erillisiä moduuleita, joita voidaan ostaa ja ottaa käyttöön vaiheittain. ERP-järjestelmissä onkin erilaisia maksumuotoja, kuten kertamaksu, jolla saa kaiken tarvittavan heti, ja sen jälkeen kuukausimaksu järjestelmän ylläpitoa varten, tai mahdollisuus pienempään maksuun alussa millä saa muutaman moduulin ja myöhemmin mahdollista valita lisää moduuleita, jolloin kuukausimaksu myös nousee. ERP-järjestelmät laskuttavat usein myös käyttäjien määrästä erikseen. Pienimmällä maksulla saa yleensä 1–3 käyttäjää ja lisäkäyttäjistä veloitetaan enemmän. ERP-järjestelmä täytyy myös hostata joko yrityksen sisäisellä palvelimella tai järjestelmän kehittäjän tarjoamalla pilvipalvelimella, joka luonnollisesti maksaa ekstraa.

Skaalautuvan maksujärjestelmän takia ERP-järjestelmät tulevat nopeasti hyvin kalliiksi ja siksi ne ovatkin suunniteltu lähinnä suurille yrityksille, jotka voivat tienata maksamansa rahat nopeasti takaisin järjestelmän ansiosta nopeutuneella tuotannolla ja myynnillä. Pietiko Oy on pieni yritys, joten ERP-järjestelmän tulisi olla mahdollisimman halpa, jotta se

pystyisi maksamaan itsensä takaisin tuotannon nopeutumisessa. Järjestelmään pitäisi pystyä myös kehittämään omia moduuleita täyttämään yrityksen tarpeet. Näiden kriteerien täyttämiä ERP-järjestelmiä on hyvin harvassa, mutta käydään seuraavaksi läpi kaksi suosituinta järjestelmää pienelle, tai keskisuurelle yritykselle.

3.1.1 Odoo

Odoo on maailman käytetyin yritysohjelmisto yli 4,5 miljoonalla käyttäjällään. Se sopii kaikenkokoisille yrityksille yhdestä työntekijästä aina yli 300 000 työntekijään. Odoo on siitä erityinen ERP-järjestelmä, että se tarjoaa myös täysin avoimen lähdekoodin version ohjelmistostaan. Odoo sisältää itsessään 30 päämoduulia ja lisäksi sen yhteisön yli 1 500 aktiivista jäsentä on kehittänyt yli 16 000 lisämoduulia, jotka kattavat kaikki mahdolliset yritysten tarpeet. (Odoo n. d.)

Odoo tarjoaa järjestelmästään kahta eri versiota: Enterprise suljetulla ja Community avoimella lähdekoodilla. Suljetun lähdekoodin järjestelmä on Odoon pääprioriteetti kehityksessä, joten se on selvästi huolitellumpi, ja sen asiakastukeen on panostettu huomattavasti enemmän. Se on myös helpompi asentaa ja Odoo tarjoaa konfigurointipalvelua, jonka avulla järjestelmä on helppoa saada yrityksen tarpeidenmukaiseksi. Kaikki tämä ei tietenkään tule ilmaiseksi vaan kaikesta täytyy maksaa, mutta edelleen isolle yritykselle tai konsernille se ei tuota ongelmaa.

Pietikolle parempi vaihtoehto olisi Odoon avoimen lähdekoodin versio, sillä siinä on paremmat mahdollisuudet kehittää itse omiin tarpeisiin ilman ylimääräisiä maksuja. Avoimen lähdekoodin version ongelma on siinä, etteivät ohjeet ole kovin kattavat järjestelmän asennukseen ja useimmissa tilanteissa tukifoorumeilla kysymykset ja vastaukset koskevat suljetun lähdekoodin järjestelmää, eivätkä sen takia päde monin paikoin eri tavalla toimivaan avoimen lähdekoodin versioon.

Odoon hinnoittelu toimii niin, että Enterprise maksaa kolmella käyttäjällä ja kahdella moduulilla (Inventory ja Manufacturing) itse hostattuna opinnäytetyön kirjoitushetkellä 110 € kuukaudessa. Jos haluaa itse kehittää moduuleita, täytyy hankkia myös Studio-moduuli, joka maksaa 48 € kuukaudessa. Sen jälkeen jokainen lisämoduuli maksaa 8–24 € kuukaudessa moduuli kohden ja jokainen lisäkäyttäjä maksaa 18 € kuukaudessa. Hinta siis nousee helposti erittäin suureksi.

Avoimen lähdekoodin Community-versio taas ei maksa mitään aluksi, kun sen hostaa omilla palvelimilla. Siihenkin on saatavilla Odoon maksullisia päämoduuleita, mutta niitä ei ole pakko käyttää ja siihen pystyy lataamaan yhteisön jäsenien tekemiä moduuleita ilmaiseksi, tai kehittää niitä itse.

3.1.2 Scoro

Scoro perustettiin vuonna 2013, joten se on aika nuori yritys, mutta on saanut jo osakseen suurta huomiota ollessaan yksi nopeimmin kasvavista keskieurooppalaisista yrityksistä (Deloitte 2018). Scoro on myös nimetty helppokäyttöisimmäksi yritysohjelmistoksi vuosina 2018 ja 2019 (Ruuse, Liisi 2018; Milk, Liis 2019).

Scoro sisältää 25 moduulia ja yritys mainostaa hyviä integrointejaan suosittuihin yritysmaailmassa käytettyihin sovelluksiin kuten esimerkiksi Google-kalenteri, Microsoft Outlook ja MS Exchange. Scorolla ei ole avoimen lähdekoodin versiota, eikä Scoroon pysty kehittämään omia moduuleita, mutta Scoro on kehitetty sellaiseksi, että suurinta osaa käyttöliittymästä ja moduulien ominaisuuksista pystyy muokkaamaan suoraa ohjelmistossa ilman koodaamista. Muokkaukset ovat tietenkin rajoittuneempia kuin esimerkiksi Odoon muokkaukset, koska niitä voi tehdä vain Scoron asettamien rajoitusten puitteissa, mutta monelle yritykselle ne kuitenkin varmasti täyttävät tarpeet. Myös suurempia muokkauksia on mahdollista saada, jos keskustelee Scoron kanssa asiasta, mutta se tulee taas huomattavasti kalliimmaksi, jos ohjelmisto räätälöidään erikseen yrityksen tarpeiden mukaan.

Scoron hinnoittelu on hieman erilainen Odoohon verrattuna siinä, että moduuleita ei voi yksitellen valikoida mielensä mukaan, vaan Scoro tarjoaa 3 eri pakettia. Paketteihin kuuluu Essential, Work Hub ja Sales Hub. Essential sisältää vain välttämättömimmät moduulit, joilla pieni yritys voi päästä alkuun, ja sen perusmaksu opinnäytetyön kirjoitushetkellä on 22 € kuukaudessa käyttäjää kohden, mutta Odoosta poiketen Scorolla on minimikäyttäjämäärä, joka on viisi käyttäjää. Work Hub sisältää enemmän moduuleja kuin Essential ja siinä on painotettu enemmän tuotantoon, kun taas Sales Hub on painotettu enemmän yritykselle, joka hakee vain kirjanpitoon ja myynnin puolelle yhtenäistä järjestelmää. Molemmista näistä paketeista on olemassa standard- ja pro -versiot, joissa standard-versiossa on hieman vähemmän moduuleita ja mahdollista pienestä maksusta lisätä niitä, kun taas pro-versiossa on kaikki version moduulit sisällytettynä. Opinnäytetyön kirjoitushetkellä standard-versio maksaa molemmissa paketeissa 33 €

kuukaudessa käyttäjää kohden ja pro-versio maksaa 44 € kuukaudessa käyttäjää kohden. Scoro tarjoaa myös Ultimate-versiota mihin sisältyy aikaisemmin mainittu yrityksen tarpeisiin räätälöity ohjelmisto, mutta sille ei ole suoranaisesti annettu hintaa vaan se riippuu täysin lopullisesta järjestelmästä.

3.2 Itse kehitetty järjestelmä

Itse kehitetty järjestelmä vaatii luonnollisesti paljon enemmän aikaa, kuin valmiin järjestelmän asennus, tietojen siirto ja käyttöönotto. Siinä on kuitenkin hyvät puolensakin, sillä järjestelmä voidaan alun perin tehdä yrityksen tarpeiden mukaan sopivaksi, eikä siinä ole mitään ylimääräistä. Jos järjestelmän kehittäjä jatkaa yrityksen työntekijänä järjestelmän alustavan valmistuksen jälkeen, on järjestelmän kehitys mahdollista myös jatkossa. Järjestelmän ohjelmointikielestä riippuen tuki voi olla myös paljon laajempaa muilta yhteisön jäseniltä ja näin ollen mahdollisiin ongelmiin voi löytää ratkaisun huomattavasti helpommin. Järjestelmästä ei myöskään koidu mitään kuukausittaisia kuluja lukuun ottamatta mahdollisia hostauksen pilvipalvelukuluja.

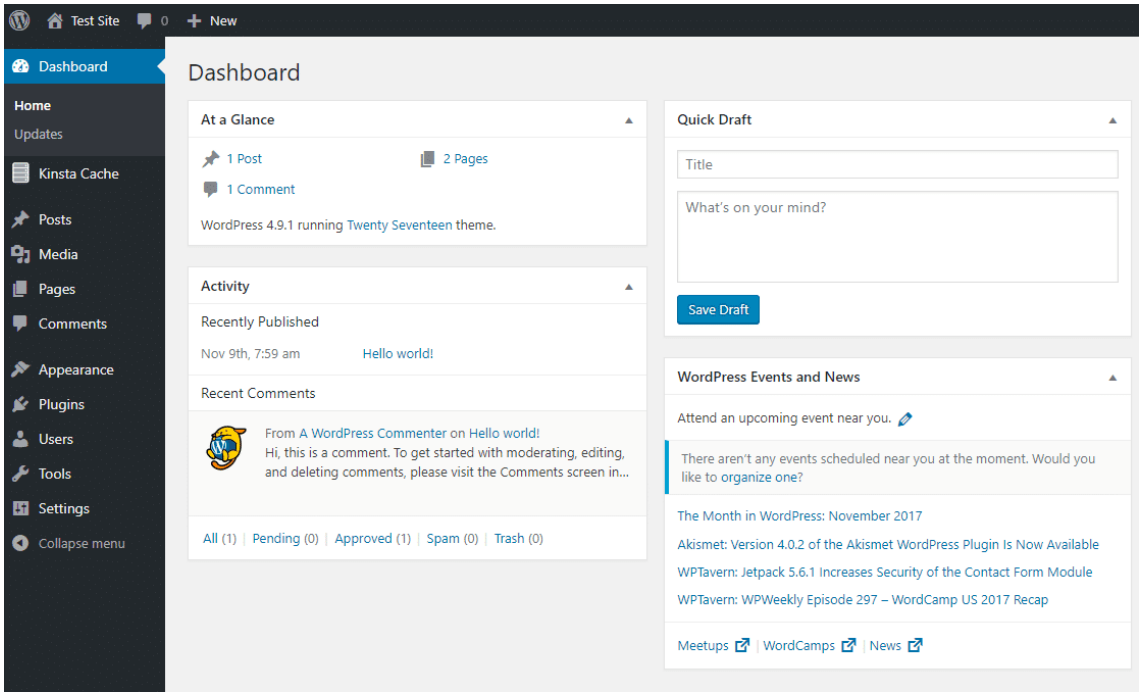
Yrityksen tarpeisiin paras ERP-järjestelmä toimisi internetin kautta selainpohjaisena, jolloin järjestelmää voi käyttää mistä vain puhelimella, tai tietokoneella, kunhan nettiyhteys löytyy. Nettisivuja on koodattu aikoinaan pelkästään HTML- ja JavaScript-yhdistelmällä, mutta myöhemmin mukaan tuli muita kieliä, kuten CSS ja PHP**.** Nykyään suurin osa nettisivuista on tehty jonkin sisällönhallintaohjelmiston kautta, kuten WordPress, Wix tai Squarespace, jotka tarjoavat yksinkertaisen käyttöliittymän nettisivujen kehittämiseen ilman koodausta. Sisällönhallintaohjelmistot tarjoavat myös mahdollisuuden sivujen lähdekoodin muokkaamiseen, mikä tarkoittaa sitä, että sivujen kehittäjät voivat muokata koodia mielensä mukaan, kun taas yrityksen muut työntekijät voivat muokata sisältöä helppokäyttöisen käyttöliittymän kautta.

Yksi hyvin suosittu kehityspohja on nimeltään Django. Se on Python-kielellä toimiva sovelluskehys (engl. framework), joka mahdollistaa monipuolisten nettisivujen kehittämisen. Django on hyvin käytännöllinen, sillä heti asennuksen jälkeen se on jo valmis nettisivupohja, mitä voi sen jälkeen muokata haluamakseen. Djangolla nettisivut täytyy kehittää koodin kautta täysin lukuun ottamatta asennuksen jälkeistä peruspohjaa, johon kuuluu hyvin riisuttu ylläpidon hallintapaneeli (Kuva 2). WordPress on myös suosittu nettisivujen kehitysalusta, mutta sen ylläpidon hallintapaneeli on huomattavasti laajempi

asennuksen jälkeen ja periaatteessa käyttäjän ei tarvitse koodata mitään vaan hän voi alkaa heti tuottamaan sisältöä ja sivuja käyttöliittymän kautta (Kuva 3).

Django olisi paras vaihtoehto itsekehitetylle järjestelmälle, sillä sen pystyisi kehittämään alusta loppuun sisältäen vain ne ominaisuudet mitä yritys kaipaa, jolloin se olisi nopea ja yksinkertainen. WordPressillä kehitetyt sivut sisältäisivät paljon sellaisia ominaisuuksia mitä ei tarvitse ja mitkä saattaisivat hidastaa sivuja. Djangolle löytyy myös paljon enemmän yhteisön tukea, joten ongelmia on helpompi ja nopeampi ratkaista, jos ja kun niitä eteen tulee.

Kuva 2. Djangon ylläpidon hallintapaneeli asennuksen jälkeen.



Kuva 3. WordPressin ylläpidon hallintapaneeli asennuksen jälkeen.

4 JÄRJESTELMÄN VALINTA

Valmiista järjestelmistä järkevin vaihtoehto olisi Odoon tarjoama ohjelmisto, sillä sitä on mahdollista käyttää täysin ilmaiseksi, jos valitsee Community-version ja hostaa sen omilla palvelimilla. Sitä on myös helpompi muokata yritysten tarpeiden mukaiseksi, koska se on avoimen lähdekoodin ohjelmisto ja siihen löytyy suuren yhteisön tukea omien moduulien tekoon.

Päätin aluksi tutkia mahdollisuutta ottaa valmis Odoon järjestelmä käyttöön ja muokata sitä tarpeen mukaan, sillä se voisi olla huomattavasti kivuttomampi prosessi verrattuna kokonaan uuden järjestelmän kehitykseen. Uuden järjestelmän kehitys tulisi kyseeseen vain, jos syystä tai toisesta Odoon käyttöönotto osoittautuisi vaikeammaksi, tai muulla tavalla enemmän aikaa vieväksi.

Odoon Community-version voi ladata Linuxille ja Windowsille. Mutta tässä toteutuksessa käytetään Windows 10 käyttöjärjestelmää. Odoon voi asentaa kahta eri kautta, joko suoraan asennustiedostona, tai lataamalla lähdekoodin ja asentamalla sen itse. Moduulikehittäjälle paras vaihtoehto on ladata lähdekoodi omalle koneelle ja asentaa se itse, sillä näin lähdekoodi on helposti saatavilla ja on myös helpompaa asentaa ja ylläpitää erikseen kahta versiota ohjelmistosta, kuten testaus ja tuotanto. Lähdekoodin voi ladata joko pakattuna kansiona tai Git-ohjelmalla.

Odoo vaatii toimiakseen muutamia muita ohjelmistoja, kuten Pythonin 3.6 tai uudemman version ja PostgreSQL-ohjelman tietokantaa varten. Odoo vaatii myös huomattavan paljon eri Python-lisäosia, jotka asennetaan pip nimisellä paketinhallintaohjelmalla. Odoon mukana tulee requirements.txt tiedosto, joka sisältää kaikki tarvittavat Python-lisäosat ja teoriassa asennus toimii niinkin yksinkertaisesti, kuin antamalla komentokehotteessa komennon ”pip install -r requirements.txt”. Käytännössä asia on kuitenkin eri, sillä monet lisäosat ovat päivittyneet sitten lähdekoodin viimeisen päivityksen ja myös kehitysympäristön Python-versio saattaa olla eri kuin mitä ohjelmiston viime päivityksen aikaan, joten usein asennuksessa tapahtuu virheitä ja kehittäjä joutuu itse selvittämään, mikä ongelman tuottaa ja miten se on korjattavissa. Tämä tapahtui myös nyt Odoota asentaessa ja ongelmaksi koitui muutama lisäosa, jonka asennettu versio ei toiminut kehitysympäristössä olleen uusimman Python 3.8.3-version kanssa. Tämän selvittämisestä koitui ylimääräistä työtä ja aikaa meni hukkaan. Ongelma ratkesi kuitenkin päivittämällä kyseiset

lisäosat uusimpaan versioonsa ja kokeilemalla asennusta uudelleen, jolloin asennus onnistui.

Usein isompien yritysten maksullisen ohjelmiston avoimen lähdekoodin versioissa ohjeet ovat hyvin lyhyet ja puutteelliset. Tästä esimerkkinä on myös Odoon ohjeet, joissa ohjelmiston asennuksen jälkeen vain mainitaan lyhyesti, miten ohjelmiston saa käynnistettyä, kun todellisuudessa asennuksen jälkeen tietokanta täytyy vielä alustaa, ennen kuin ohjelmistoa voi käyttää. Tämä puute ohjeissa aiheutti sen, ettei Odoon järjestelmään pääse kirjautumaan, ennen kuin luo uuden tietokannan PostgreSQL-tietokantaohjelmistossa, ja sen jälkeen alustaa sen Odoon komentokehotteen kautta. Tämän ratkaisun löytäminen kesti huolestuttavan kauan ja sai minut epäilemään, miten käytännöllinen järjestelmä voisi olla, jos jo asennuksessa tulee eteen tämän kaltaisia ongelmia, eikä ohjeita ratkaisuun meinaa löytää mistään.

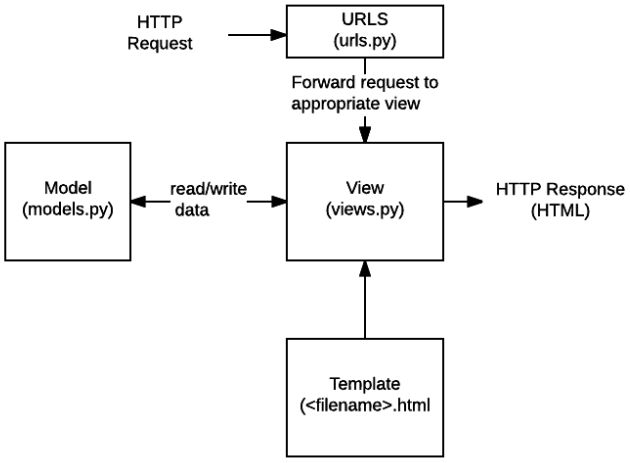
Vihdoin päästyäni kirjautumaan Odoon järjestelmään sisään aloin tutkia moduuleita ja yleistä toimintaa ja ulkoasua. Hyvin nopeasti kävi ilmi, että Community-version moduulit olivat hyvin pelkistettyjä ja niissä ei ollut lähellekään kaikkea yrityksen tarvitsemista ominaisuuksista. Moduulit olivat hyvin yksinkertaisia, esimerkkinä varastokirjanpitomoduuli, jossa oli mahdollista nähdä oikeastaan vain tuotteen tuotekoodi, ja kuinka monta tuotetta varastossa on. Moduulista puuttui täysin mahdollisuus kirjata tuotteen tietoja, kuten ohjelmisto- tai piirilevyversiota, jotka yritys täytyy kirjata ylös täyttääkseen CE-merkinnän vaatimukset.

Mitä enemmän tutkin moduuleita ja käyttöliittymää, sitä selvemmin tulin siihen tulokseen että, jotta Odoon järjestelmästä saataisiin yrityksen tarpeet täyttävä, vaatisi se vähintään yhtä paljon työtä, kuin täysin uuden ja paljon kevyemmän järjestelmän kehitys. Tämän takia olen päättänyt kehittää täysin uuden järjestelmän ja yrityksen tarpeisiin sopii parhaiten Django-sovelluskehyksellä kehitetty, sillä siinä tulee valmiina turvallinen kirjautumisjärjestelmä, ja se on hyvin kevyt ja helposti muokattava. Eli Odoon valmiin järjestelmän muokkauksen sijaan tämä opinnäytetyö keskittyy oman tuotannonhallintajärjestelmän kehitykseen.

5 DJANGO SOVELLUSKEHYS

5.1 Djangon toiminta

Djangon toiminta perustuu Model-View-Template-arkkitehtuuriin (MVT) (Kuva 4). Djangon MVT-arkkitehtuuri eroaa suositummasta Model-View-Controller-arkkitehtuurista (MVC) siinä, että Django itsessään korvaa ohjaimen eli contoller-osan MVC-arkkitehtuurissa ja jäljelle jää Djangon template-osio. Selainpyynnöt tulevat Djangoon URL-muodossa ja Django käsittelee URL-osoitteet urls.py tiedostossa, joka lähettää ne prosessoinnin jälkeen eteenpäin osoitetta vastaavalle näkymälle (View). Näkymät prosessoivat kaiken mitä käyttäjälle näytetään views.py tiedostossa. Näkymät voivat olla niinkin yksinkertaisia, kuin vain tekstin palauttaminen käyttäjälle. Ne voivat olla myös hyvinkin monimutkaisia sisältäen tietokantakyselyjä, lomakkeen prosessointeja, tai vaikka luottokorttitietojen käsittelyä. Tietokantakyselyitä tehdessä näkymä käyttää models.py tiedostoa, jossa on konfiguroitu kaikki objektit, jotka käyttävät tietokantaa. Kun näkymä on valmis, tulos lähetetään vastauksen muodossa käyttäjän selaimeen. Vastaus, joka lähetetään, on HTML-sivu, joka sisältää tekstiä, kuvia ja muuta sisältöä. Nämä HTML-sivut muodostetaan Djangon mallijärjestelmällä (Template system). (Mozilla 2020.)



Kuva 4. Havainnekuva Djangon toiminnasta (MDN web docs, Mozilla).

Django toimii sovellusperiaatteella, eli jokainen sivusi ominaisuus on oma sovelluksensa, jonka voi asentaa helposti uuteen projektiin tulevaisuudessa, jos tarvitset samanlaista ominaisuutta uudestaan. Tämä tarkoittaa sitä, että esimerkiksi sivun blogi voi olla oma sovelluksensa, chat on omansa ja verkkokauppa omansa jne. Näin voidaan kehittää

kerran toimivan chat-sovellus ja asentaa sama sovellus myös toisen asiakkaan sivuille ja toiminnot toimivat ilman ongelmia, mutta ulkoasun voi muokata helposti asiakkaan sivujen teeman mukaiseksi.

5.2 Kehityksen aloitus

Django toimii Python-kielellä ja kehittäessä Pythonilla suositellaan aina käyttämään virtuaalista ympäristöä (engl. virtual environment). Virtuaalinen ympäristö tarkoittaa sitä, että jokaisen Python-projektin tarvitsemat lisäosat asennetaan vain projektin virtuaaliseen ympäristöön, eikä globaalisti koko tietokoneelle. Tämä mahdollistaa sen, että vaikka monta projektiasi käyttäisi samaa lisäosaa voit silti tarvittaessa päivittää yhden projektin lisäosan samalla, kun toisessa projektissa lisäosa pysyy vanhassa versiossa vähentäen mahdollisia yhteensopivuusongelmia. Virtuaaliympäristön asennus toimii helposti kirjoittamalla Pythonin asennuksen jälkeen komentokehotteeseen ”pip install virtualenv” ja painamalla Enteriä. Asennuksen jälkeen voit luoda virtuaaliympäristön haluamaasi kansioon antamalla komennon ” virtualenv ’virtuaaliympäristön_nimi’ ” komentokehotteessa. Tämä komento asentaa kaiken tarpeellisen Pythonin lisäosien asennusta varten kyseiseen kansioon. Tämän jälkeen virtuaaliympäristö täytyy vielä aktivoida menemällä komentokehotteessa juuri luodun kansion sisällä sijaitsevaan Scripts-kansioon ja antamalla komennon ”activate”. Virtuaaliympäristön pitäisi nyt olla aktivoitu, ja sen huomaa siitä, jos komentokehotteen rivin alussa lukee virtuaaliympäristön nimi (Kuva 5).

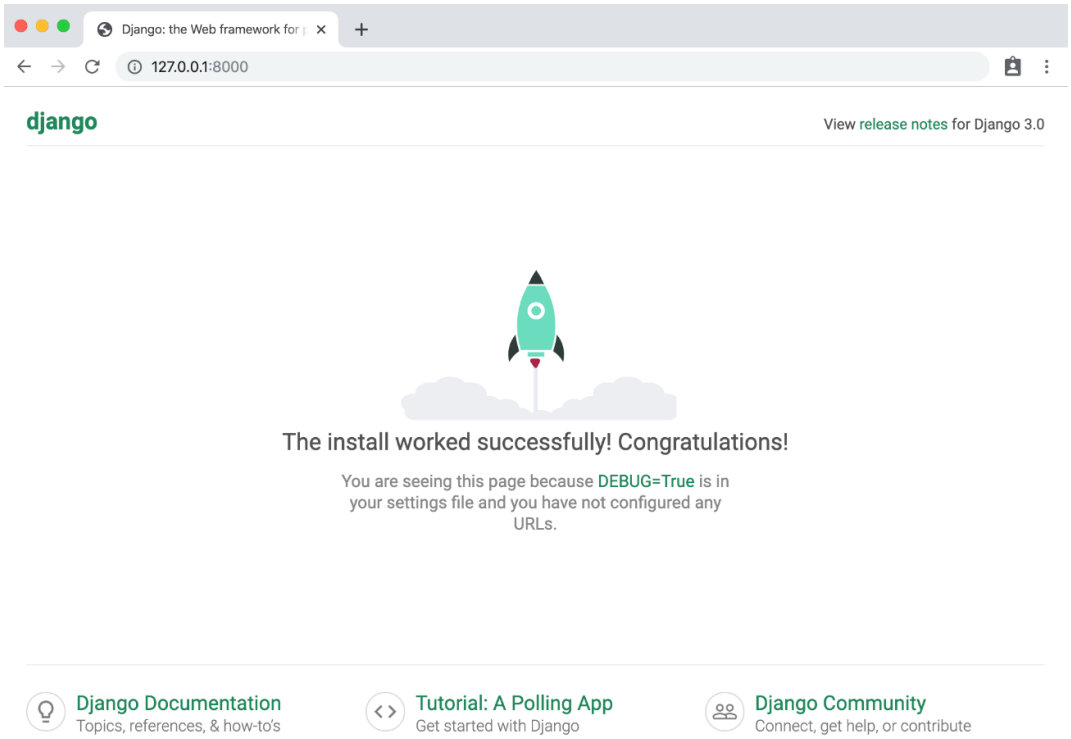
Kuva 5. Virtuaaliympäristön nimi korostettuna rivin alussa.

Djangon asennus alkaa hyvin samaan tapaan kuin virtuaaliympäristön, antamalla komentokehotteeseen komennon ”pip install django”. Asennuksen jälkeen täytyy luoda uusi Django-projekti antamalla haluamasi kansion sisällä komento ” django-admin startproject ’projektin nimi’ ” komentokehotteeseen. Käytäntönä on, että Django-projekti asennetaan samaan kansioon, minkä sisälle virtuaaliympäristö on asennettu, näin voidaan helpommin erottaa, mikä virtuaaliympäristö kuuluu millekin projektille.

Djangon asennus ei ole tätä vaikeampi ja nettisivun saa toimintaan antamalla komentokehotteessa projektin kansiossa komennon ”python manage.py runserver”. Tässä vaiheessa huomaamme Djangon ja Odoon suurimman eron, Django varoittaa, ettei

tietokantaa ole alustettu ja antaa suoraan ohjeen komentokehotteessa, miten alustus suoritetaan. Django silti käynnistää nettisivun vaatiman palvelimen ja kertoo, että nettisivut ovat näkyvissä kirjoittamalla selaimen osoiteriville ”http://127.0.0.1:8000/”, tai ”http://localhost:8000/”. Django siis suoraan kertoo mikä on vialla ja miten se korjataan, toisin kuin Odoo, johon täytyi etsiä vastausta netin tukifoorumeilta.

Djangon alustavaa nettisivua voi tarkastella myös ilman tietokannan alustusta ja ensimmäisenä http://localhost:8000/ sivulle mentäessä vastaan tulee teksti, joka kertoo asennuksen onnistuneen ja antaa linkit Djangon ohjeisiin, tutoriaaliin ja yhteisön foorumeille (Kuva 6). Djangon hallintapaneelin kirjautumissivu on myös nähtävissä osoitteessa http://localhost:8000/admin, mutta hallintapaneeliin ei voi vielä kirjautua, sillä pääkäyttäjää ei ole tehty. Pääkäyttäjän luontia varten tietokanta täytyy alustaa komennolla ”python manage.py migrate”. Nyt tietokantaan voi lisätä pääkäyttäjän tiedot komennolla ”python manage.py createsuperuser”, joka pyytää kirjoittamaan käyttäjänimen, sähköpostiosoitteen ja salasanan. Tämän jälkeen hallintapaneeliin voi kirjautua ja näkymä pitäisi olla samanlainen kuin aikaisemmin kuvassa 2. Nyt Django on asennettu, tietokanta alustettu ja pääkäyttäjä luotu, joten kaikki on valmista tuotannonhallintajärjestelmän kehitystä varten.



Kuva 6. Djangon aloitussivun näkymä asennuksen jälkeen.

6 JÄRJESTELMÄN KEHITYS

6.1 Vaatimukset

Ensimmäinen asia, jonka käyttäjän tulisi nähdä, kun hän kirjautuu uuteen tuotannonhallintajärjestelmään, on tiivistelmä kaikesta tärkeimmästä, mitä järjestelmä pitää sisällään, kuten tilaukset ja varastokirjanpito. Kutsun tätä tästä eteenpäin hallintapaneeliksi, sillä tässä alkunäkymässä olisi tarkoitus pystyä lisäämään ja muokkaamaan tilauksia ja tarkastelemaan ja muokkaamaan tuotteita.

Järjestelmän käyttöliittymä täytyisi olla mahdollisimman yksinkertainen, siisti ja intuitiivinen. Sen tulisi olla myös yhteneväinen kaikkialla järjestelmässä. Tässä auttaa Djangon mallijärjestelmä, joka mahdollistaa yhden yleisen niin kutsutun base-mallin luomisen. Yleinen malli toimii sivujen pohjana, johon voi luoda esimerkiksi navigointipaneelin, ylä- ja alatunnisteen ja muita ominaisuuksia, jotka pysyvät samana sivuston kaikilla sivuilla. Yleisessä mallissa pystyy myös sisällyttämään sivujen käyttämät tyyliohjeet, eli CSS-tiedostot, jotka määrittävät sivujen ulkonäön ja muut yleiset sivujen määritykset, jolloin niitä ei tarvitse kirjoittaa jokaiseen malliin erikseen. Yleisen mallin lisäksi luodaan jokaiselle sivulle oma malli, joka sisältää uniikin tiedon ja datan, mitä käyttäjälle halutaan näyttää.

Haluan myös, ettei sivuille pääse, ellei ole kirjautunut sisään, joten sivut pitää konfiguroida niin, että jokainen sivu ohjaa käyttäjän kirjautumissivulle, jos hän ei ole kirjautunut. Tämä uudelleenohjaus täytyy tietenkin poistaa itse kirjautumissivulta, jottei eteen tule päättymätöntä uudelleenohjausta, joka hajottaisi sivut. Djangoon on mahdollista asentaa niin kutsuttuja Middleware-lisäosia, jotka ovat ihmisten itsenäisesti kehittämiä lisäosia, jotka voi lisätä mihin tahansa Django-projektiin. Myös kirjautumisen vaatimiseen on tällainen lisäosa, ja sen asennus ei vaadi muuta kuin kooditiedoston lisäämisen Djangon middleware-kansioon ja Djangon asetuksissa kertoa uudesta lisäosasta lisäämällä sen nimen lisäosa-listaan. Konfiguroin asetuksista myös niin, että kirjautuessa sisään sivusto vie käyttäjän hallintapaneeli-sivulle ja kirjautuessa ulos käyttäjä viedään sisäänkirjautumissivulle. Näin varmistetaan, ettei käyttäjä vahingossakaan jätä yrityksen salaista tietoa sisältävää sivua auki poistuessaan.

Nettisivujen tyyliohjeet voi luoda itse, mutta paljon helpompi ja aikaa säästävämpi vaihtoehto on ladata netistä valmiit tyyliohjeet, jotka kattavat laajalti eri tyylejä ja

muokkauksia. Yksi eniten käytetyistä tyyliohjepaketeista on Bootstrap. Se mahdollistaa nopean nettisivujen kehityksen ilman turhaa miettimistä, minkä muotoisia nappeja haluaa tai minkä näköisiä taulukoista haluaa. Kehittäjän ei tarvitse kertoa muuta kuin, että kyseessä on punainen nappi ja Bootstrap muotoilee napin sen mukaisesti. Bootstrap tarjoaa sivuillaan myös paljon esimerkkejä erilaisista nettisivujen ulkoasuista, mitä heidän tyyliohjepaketillaan voi saavuttaa. Esimerkit ovat vapaasti ladattavissa ja niitä voi käyttää mielensä mukaan, joten päätin käyttää heidän hallintapaneeliesimerkkiään, jota muokkaan mieleisekseni tarpeen mukaan.

6.2 Hallintapaneeli

Tuotannonhallintajärjestelmässä olen päättänyt kehittää kolme Djangon sovellusta: hallintapaneeli, tuotantotietokanta ja digitaaliset koontaohjeet. Uuden sovelluksen asennus toimii antamalla virtuaaliympäristön sisällä komento ”python manage.py startapp sovelluksen_nimi”. Django luo kaikki sovelluksen tarvitsemat tiedostot, kuten models.py, views.py jne.

Sovelluksen kehitys alkaa urls.py-tiedoston muokkauksella, jotta Django osaa ohjata selainpyynnön oikealle näkymälle. Projektin urls.py tiedostossa ohjataan kaikki URL-osoitteet ilman päätettä suoraan hallintapaneeli-sivulle, eli localhost:8000/dashboard/. Kaikki dashboard-tekstin sisältävät URL-osoitteet ohjataan hallintapaneelisovelluksen omaan urls.py tiedostoon, jossa niistä leikataan pois dashboard ja sitä edeltävä teksti ja loppuosa puolestaan ohjataan eteenpäin oikealle näkymälle. Tässä vaiheessa hallintapaneelille ei ole aliosoitteita, joten tyhjä pääte ohjataan hallintapaneelin näkymälle. Näkymät ovat funktioita, joissa määritellään kaikki näkymän käyttämät muuttujat ja tietokantakutsut ja lopuksi palautetaan selaimelle haluttu malli, joka sisältää lopullisen HTML-koodin (Kuva 7).

```python
# mrpproject/urls.py
urlpatterns = [
    path("admin/", admin.site.urls),
    path("dashboard/", include("dashboard.urls")),
    path("", lambda request: redirect("dashboard/", permanent=False)),
]
```

Ohjataan kaikki dashboard-tekstin sisältävät URL-osoitteet hallintapaneelin urls.py tiedostoon ja URL ilman mitään päätettä ohjataan aina hallintapaneelisivulle.

```python
# dashboard/urls.py
urlpatterns = [
    path("", views.dashboard, name="dashboard"),
]
```

Hallintapaneelin urls.py ohjaa pelkällä dashboard-päätteellä olevan URL-osoitteen dashboard-nimiselle näkymälle.

```python
# dashboard/views.py
def dashboard(request):
    context = {
        # Muuttujat ja tietokantaviittaukset lisätään tähän
    }
    return render(request, "dashboard/dashboard.html", context)
```

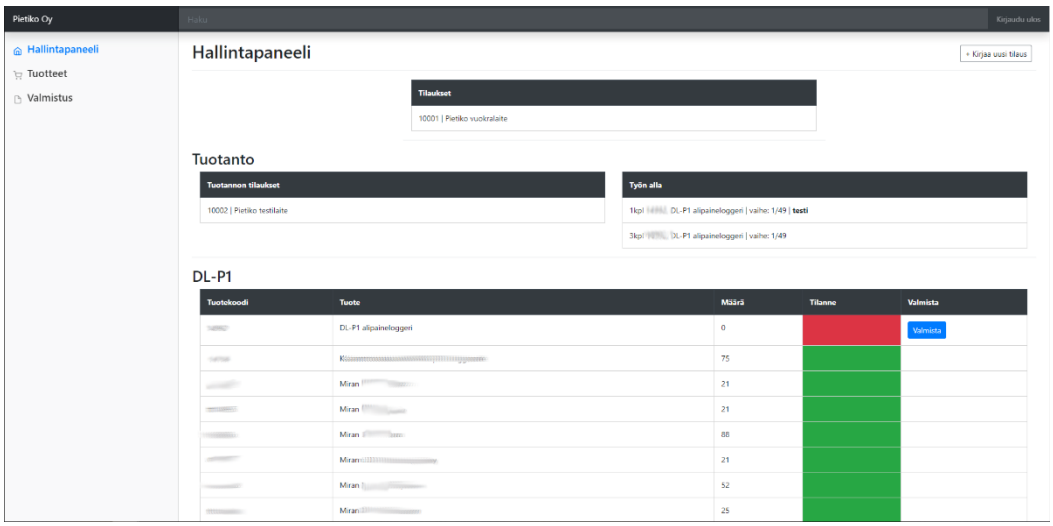
Näkymän funktio käsittelee muuttujat ja tietokantaviittaukset ja palauttaa dashboard.html sivun selaimeen.

Kuva 7. Havainnointi projektin URL-osoitehallinnasta.

Hallintapaneelin HTML-koodi on lisätty dashboard.html-tiedostoon, ja nyt nettisivu toimii. Tällä hetkellä siinä kylläkin toimii vain hallintapaneeli eikä mikään muu sivu, kuten tuotantotietokanta tai koontaohjeet. Keskityn ensin hallintapaneelin toimintaan ja ulkoasuun, ja sen jälkeen muihin sivuihin/sovelluksiin. Sivuston base-mallissa tulee olemaan ylätunnisteessa yrityksen nimi, hakukenttä ja kirjautumislinkki. Sivupalkissa on linkki jokaiseen sovellukseen ja muu tila täyttyy sovelluksien omalla datalla. Base-mallissa täytyy olla {% block content %} {% endblock %} siinä kohtaa, mihin sovellusten data halutaan. Djangon HTML-tiedostoissa käytetään formaattia {% %} merkkaamaan python funktiota ja {{ }} formaattia muuttujalle. Ensimmäinen {% block content %} kertoo Djangolle, että tästä alkaa sovelluksen HTML-koodi, tässä tapauksessa dashboard.html-tiedoston sisältö ja jälkimmäinen {% endblock %} merkkaa sovelluksen datan loppumista. Toisin sanoen hallintapaneelin HTML-koodi ilmestyy näiden kahden merkinnän väliin lopullisessa HTML-sivussa. Hallintapaneelin HTML-tiedostossa puolestaan täytyy kertoa, että se kuuluu base.html tiedoston yhteyteen. Tämä tehdään kirjoittamalla tiedoston alkuun {% extends "dashboard/base.html" %}, sen alle {% block content %} ja tiedoston loppuun {% endblock %}. Näin ollen kahden block-funktion väliin tuleva HTML-koodi lisätään base-mallin.

Hallintapaneelin kehitys alkoi ja Djangon loistavan yhteisötuen ansiosta ongelmat ja kysymykset saatiin ratkaistua nopeasti ja kehitys eteni sulavasti. Kehityksen aikana sivun ulkonäkö vaihtui muutaman kerran, ominaisuuksia tuli lisää ja niitä myös karsittiin ajan puutteessa. Yrityksen työntekijöiden kesken pidettiin palavereita, joissa tarkasteltiin ulkoasua ja toimintoja. Palautteen perusteella muokkasin sivuja ja paransin käytettävyyttä.

Hallintapaneelista nähdään uudet tilaukset, keskeneräiset tuotannon valmistustyöt ja varastossa olevat laitteet ja niiden osat (Kuva 8). Tilauksissa näkyy normaalisti vain tilauksen numero ja asiakkaan nimi, mutta jos tilausta klikkaa hiirellä, avautuu se niin, että siitä näkee tilaukseen tulevat tuotteet, mahdolliset tilaukselle varatut tuotteet, niiden määrän ja kolme nappia: kerää, muokkaa ja poista. Kerää-nappi avaa ponnahdusikkunan missä kerrotaan mitä laitteita tilaus sisältää ja mitkä niistä on saatavilla varastosta. Sen jälkeen on nappi, jota painamalla voi varata tuotteet tilaukselle, jos niitä on varastossa ja jos varastossa ei ole tarpeeksi, puuttuvat tuotteet siirtyvät tuotannon tilausjonoon. Django tarjoaa hyvät objektien muokkaus, poisto ja lisäyssivut pääkäyttäjän hallintapaneelissa, mutta on myös mahdollista tehdä omat sivut objektien hallintaa varten. Omien objektinhallintasivujen teko tulisi kyseeseen, jos sivut olisivat julkiset, tai muuten vaatisivat hienomman ulkoasun. Tässä projektissa ainoat sivujen käyttäjät ovat yrityksen työntekijät, joten Djangon tarjoamien pohjien hyvin hillitty ulkoasu ei haittaa ja näin säästämme aikaa huomattavasti. Valmiita pohjia pystyy käyttämään niin, että muokkaus- tai poistonappia painaessa luetaan tilauksen tunnistenumero, ja sen perusteella muodostetaan linkki Djangon muokkaus- tai poistosivulle, johon käyttäjä ohjataan. Tilauksen lisäysnappi ohjaa vain suoraan Djangon tilausobjektin lisäyssivulle.



Kuva 8. Kuva kehitetystä hallintapaneelista.

Hallintapaneelin kolme tilauksiin liittyvää osaa toimii niin, että uuden tilauksen tullessa se kirjataan ensin järjestelmään, jolloin tilaus ilmestyy tilaukset taulukkoon. Kun työntekijä aloittaa tilauksen keräyksen hän painaa kerää-nappia, joka avaa ponnahdusikkunan ja tarkistaa onko varastossa tilaukselle tulevia tuotteita. Keräys-ikkuna kertoo mitä tuotteita tilaukseen kuuluu ja onko kyseisiä tuotteita varastossa. Jos varastossa on kaikki tilaukseen kuuluvat tuotteet, tulee ikkunaan uudestaan kerää-nappi, jota painamalla varastossa olevat tuotteet varataan tilaukselle, tilaus poistetaan ja käyttäjä ohjataan oikeisiin ohjeisiin laitteiden lopullista konfigurointia varten. Jos kaikkia tilauksen tuotteita ei löydy varastosta, kerää-nappi varaa varastossa olevat tuotteet tilaukselle, vähentää tilauksen tuotteista varatut ja siirtää tilauksen tuotannon valmistustöihin. Jos tilaukseen ei ole yhtään tuotetta varastossa, siirretään se suoraan sellaisenaan tuotannon valmistustöihin ja jos tilaus ei sisällä yhtään tuotetta, vaan esimerkiksi vain tuotteiden huollon, tai muun vastaavan työn, tilaus kirjataan valmiiksi ja poistetaan nappia painamalla. Tilauksen poisto sen valmistumisen jälkeen ei tuota ongelmia, sillä alkuperäiset tilaukset pysyvät silti yrityksen käyttämässä kirjanpitopalvelussa. Tuotteet varataan niin, että järjestelmä kirjaa tilauksen asiakkaan ja tilausnumeron laitteen tietoihin niin kuin ennen tehtiin manuaalisesti Excel-taulukossa.

Kun tilaus on siirretty tuotannon töihin, voi tuotannon työntekijä katsoa tilausta klikkaamalla, mitä tuotteita tilaus vaatii. Klikkaamalla tilaus avautuu samalla tavalla, kuin uudetkin tilaukset, mutta nyt tilauksesta näkyy sille mahdollisesti varatut tuotteet, niiden sarjanumerot ja puuttuvat tuotteet. Alas ilmestyy myös kerää-, muokkaa- ja poista-napit. Kerää-napista käyttäjä viedään samaan ponnahdusikkunaan, kuin uudessa tilauksessa, josta voi varata puuttuvat tuotteet valmistuksen jälkeen. Kun tuotannon työntekijä haluaa valmistaa puuttuvat tuotteet hän painaa valmista-nappia tuotteen vieressä tuotelistauksessa, tai hän voi mennä valmistussivulle sivun vasemmasta valikosta. Kun tuotteita on alettu valmistaa, mutta se keskeytetään syystä tai toisesta, esimerkiksi työpäivän päättyessä, tai jonkun laitteen osan puuttuessa, voi valmistuksen senhetkisen tilanteen tallentaa ja jatkaa myöhemmin. Tällöin valmistustyö siirtyy ”työn alla” -tilaan, ja sen mukaiseen palkkiin hallintapaneelissa ja työhön voi lisätä kommentin, miksi se on keskeytetty, kuten esimerkiksi ”odottaa paineantureita”. Kommentti näkyy valmistustyön tiedoissa lihavoidulla tekstillä, jotta se näkyy selvästi heti, ilman erikseen avaamista. Kun valmistustyötä haluaa jatkaa voi painaa jatka-nappia ja valmistusohjeet jatkavat siitä kohdasta mihin edellisellä kerralla jäätiin.

Tilausten ja tuotannon töiden alapuolelle tulee listaus eri tuotteista, jossa näkyy tuotekoodi, tuotenimi ja varastotilanne. Varastotilanne näytetään selkeästi määränä, ja sen jälkeen värikoodilla, joka vaihtelee määrän mukaan punaisen, keltaisen ja vihreän välillä. Jokaisen valmistettavan tuotteen kohdalla on myös valmista-nappi, joka ohjaa käyttäjän kyseisen laitteen koontaohjeeseen. Kun käyttäjä painaa tuotteen tuotenimeä, avautuu alle listaus kaikkien varastossa olevien tuotteiden sarjanumeroista. Avautuneista sarjanumeroista painettaessa pääsee kyseisen laitteen kohdalle tuotantotietokannassa, jossa voi tarkistaa tarkempia tietoja laitteesta.

Hallintapaneeli laskee varastossa olevien tuotteiden määrän sen perusteella, onko tuotteen tietoihin kirjattu asiakasta ja tilausnumeroa. Jos tuotteella ei ole merkattu tilausta, niin se on vielä varastossa myymättömänä. Tuotteet, jotka on koottu, mutta testauksessa huomattu vika, ja sen takia myymäkelvottomia, karsitaan pois varastotuotteista tuotteisiin lisätyllä ”viallinen” parametrillä. Niiden tuotteiden, minkä tietoja ei erikseen kirjata tuotantotietokantaan, kuten virtalähteet ja laitteiden osat, varastomäärä haetaan kirjanpito-ohjelmistosta, mihin valmistuskirjaukset tehdään. Tieto haetaan rajapintakutsulla, johon vaaditaan eri parametrejä, kuten rajapinta-avaimet ja käyttäjätunnukset. Nämä arkaluonteiset tiedot täytyy suojata poistamalla ne suorasta koodista. Se toimii tekemällä uuden tiedoston projektin pääkansioon nimeltä ”.env”, ja sinne lisätään tunnukset muodossa ”muuttuja = rajapintatunnus”, jokaiselle riville vain yksi tunnus. Tämän jälkeen Djangon asetuksissa asetetaan muuttujat projektille muodossa ”muuttuja = config("muuttuja", default="")”. Tämä kertoo Djangolle, että config-tiedostossa on arkaluontoinen muuttuja, jolloin Django etsii muuttujan arvon ensisijaisesti .env-tiedostosta. Itse koodissa pystyy käyttämään asetuksissa asetettua muuttujaa muodossa ”settings.muuttuja”. Jos projektin haluaa lisätä esimerkiksi GitHubiin, voi Gitin .gitignore-tiedostoon lisätä päätteen .env, jolloin Git ei lisää arkaluonteisia tietoja sisältävää tiedostoa julkisesti näkyville.

6.3 Tuotantotietokanta

Aikaisempi tuotantotietokanta Excel-pohjaisena oli aikojen saatossa muuttunut hyvin sekavaksi, sillä uusia sarakkeita oli vain lisätty loppuun aina kun uusia kirjattavia tietoja tuli uuden laiteversion myötä. Näin ollen sarakkeiden määrän lisääntyessä taulukkoa joutuu usein selaamaan hyvin paljon tietoja lisättäessä sen sijasta, että kaikki harvemmin käytetyt tiedot olisivat lopussa, kuten asiakkaan ja pilvipalvelun tiedot. Tietojen lisäämisessä ei myöskään ole ollut mitään standardia tai tyyliopasta, joten jokainen tuotannon

työntekijä on lisännyt tietoja oman parhaan arvionsa mukaisella tyylillä, joka aiheuttaa sekaannusta siitä, mitä on merkitty ja milloin. Esimerkiksi laitteiden ohjelmistopäivitykset on aluksi merkattu vain korvaamalla vanha versio tietokannassa, mutta myöhemmin on kirjattu vanha ja uusi versio ja milloin se on päivitetty. Joissakin laitteissa on myös merkitty, kuka laitteen on päivittänyt, mutta kaikkia näitä merkintöjen eri muotoja on sekaisin tietokannassa ja uuden tietokannan olisi tarkoitus ratkaista näitä ongelmia.

Tuotantotietokanta on yksi kolmesta sovelluksesta, jotka kehitän järjestelmään, joten sen kehitys aloitetaan samalla tavalla, kuin hallintapaneelinkin, eli antamalla komento ”python manage.py startapp sovelluksen_nimi” projektin sisällä. Tämän sovelluksen tapauksessa sovelluksen nimi on products. Tämän jälkeen asetetaan taas URL-osoitteet tuotantotietokantasovelluksen urls.py tiedostossa.

Tarkoitukseni on tehdä uudesta tuotantotietokannasta mahdollisimman samannäköinen, kuin vanha Excel-pohjainen, jotta työntekijöiden on helppo sopeutua uuteen tuotannonhallintajärjestelmään (Kuva 9). Kaikki tai ainakin niin monta puutetta tai ongelmaa kuin mahdollista täytyy saada silti korjattua. Ensimmäisenä sarakkeet täytyy järjestää paremmin, että useammin tarvittavat tiedot löytyvät taulukon alusta ja harvemmin käytetyt taulukon lopusta. Näin vältetään turhaa taulukon selausta edestakaisin ja selkeytetään taulukon rakennetta jakamalla sarakkeet luokittain niin, että aluksi on laitteen tiedot ja eri sarjanumerot, sen jälkeen valmistajan nimi ja kommentit ja viimeiseksi asiakkaan ja pilvipalvelun tiedot. Taulukon selaamisen helpottamiseksi sarakkeiden otsikot täytyy pysyä aina näkyvissä ja myös laitteiden sarjanumerot täytyy näkyä, jotta käyttäjä erottaa minkä laitteen tietoja hän selaa. Jokainen rivi näkyy myös hieman tummennettuna, kun hiiren osoitin on sen päälle helpottaakseen rivien erottuvuutta.

DL-P1

Valmistuspäivä	Sarjanumero	SIM kortin sarjanumero	SD-kortin sarjanumero	GSM modeemin IMEI-numero	sarjanumero	HW versio (ladonta)	Piirilevyversio	SW versio	HWID	Huomioita
8.9.2020	17010071									
	17010070									
10.1.2018	17010069									
10.1.2018	17010068									
	17010067									
22.11.2017	17010066									
22.11.2017	17010065									
22.11.2017	17010064									
22.11.2017	17010063									
22.11.2017	17010062									
22.11.2017	17010061									
22.11.2017	17010060									
22.11.2017	17010059									
22.11.2017	17010058									

Kuva 9. Kuva uudesta tuotantotietokannasta.

Seuraava kehityskohde on tietojen yhtenäisyys. Osaan sarakkeista on annettu vain suorat vaihtoehdot vapaan tekstin sijasta, sillä niissä harvoin valitaan muutaman vaihtoehdon ulkopuolelta. Näitä on esimerkiksi piirilevyn ladontaversio, tai SD-kortin mallinumero. Näin ollen vähennetään tietojen kirjoittamiseen kuluvaa aikaa ja vältytään kirjoitusvirheiltä ja muilta kirjoitustyylien eroilta. Aikaisemmin asiakkaan kohdalle on kirjoitettu vaihtelevasti, joko pelkkä asiakkaan nimi, tai asiakkaan nimi ja tilauksen numero. Tarkoitus olisi kirjoittaa molemmat nimi ja tilauksen numero, joten uudessa järjestelmässä tiedot tulevat automaattisesti hallintapaneelin toimesta omiin sarakkeisiinsa. Näin voidaan myös halutessaan löytää kaikki tietyn asiakkaan laitteet, sillä asiakkaan nimi löytyy nyt erikseen omasta sarakkeestaan.

Excelin yksi suurista ongelmista tuotantotietokannan kanssa on se, kun moni sarjanumero sisältää pelkkiä numeroita ja alussa voi olla nollia ja kun Excel tunnistaa sarjanumeron luvuksi, se poistaa edeltävät nollat. Näin ollen sarjanumeron eteen täytyy laittaa lainausmerkki, jotta Excel tunnistaa sen tekstiksi, eikä muuta numeroa missään muodossa. Tämä taas tuottaa uuden ongelman, sillä Excel näyttää solussa virhemerkin ja ilmoittaa, että luku on tallennettu tekstinä. Uudessa tuotantotietokannassa ongelma on ratkaistu niin, että solut ovat alun perinkin tekstisoluja, joten vastaavia virheitä ei synny ja edeltävät nollat säilyvät.

Toinen Excelin heikko osa on kommenttien lisäys tuotteisiin ja tuotteiden muokkaushistoria. Excelissä pystyy kyllä lisäämään kommentteja soluihin, mutta ne eivät ole kovin

käyttäjäystävällisiä eikä niitä saa koko laitteelle, vaan ne koskevat yhtä solua. Excelin muokkaushistoriassa näkyy, milloin taulukkoa on muokattu ja kuka käyttäjistä sitä on muokannut, mutta siitä ei näe mitä soluja tai laitteita on muokattu, joka olisi tärkein ominaisuus tuotantotietokannassa. Uudessa tuotantotietokannassa, kun painaa laitteen sarjanumeroa, avautuu uusi ikkuna, josta laitteen tietoja voi muokata. Tässä ikkunassa voi myös lisätä kommentteja laitteelle, kuten esimerkiksi, milloin laitteen ohjelmisto on päivitetty, tai milloin on vaihdettu jokin osa. Jokaisessa laitteessa näkyy myös, kuka laitteen tietoja on muokannut, milloin niitä on muokattu ja mitä tietoja on muokattu. Näin ollen, jos laitteen kanssa tulee ongelmia tai epäselvyyksiä, voi muokkaushistorian tarkistaa laitteen sivulta.

Tuotantotietokannassa pystyy myös merkitsemään laitteen vialliseksi, jos laitteessa on ilmennyt jotain ongelmia, eikä sitä voida myydä asiakkaalle. Tässä tapauksessa laitteen sarjanumeron sisältävä solu värjäytyy punaiseksi ja sitä ei lasketa varastossa oleviin tuotteisiin.

6.4 Digitaaliset koontaohjeet

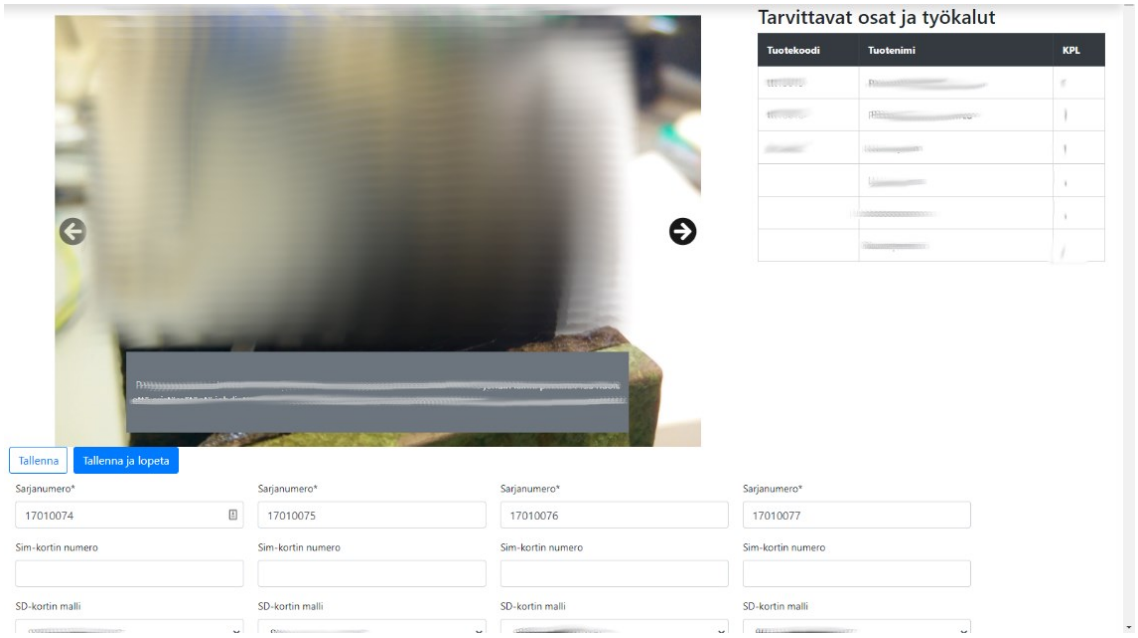
Koontaohjeiden suunnittelu uuteen järjestelmään oli mahdollisesti järjestelmän vaikein osuus. Vanhat ohjeet on aina tehty Microsoft Word-ohjelmistolla, ja sen jälkeen tulostettu paperiseksi versioksi tuotantoon. Suuren kuvilla ehostetun tekstitiedoston muuttaminen nettisivumuotoon tuotti hyvin paljon ongelmia ja päänvaivaa. Parhaaksi vaihtoehdoksi koettiin diaesitysmuotoinen toteutus, jossa jokaisen työvaiheen kuva, tai video on yksi dia, ja sen yhteydessä teksti, jossa ohjeistetaan työvaihe.

Koontaohje-sovelluksen nimeksi tulee manufacturing, ja sen osoitteet menevät muodossa verkkosivu/manufacture/laitteen_nimi/valmistuksen_vaihe/. Kun osoitteen pääte on pelkkä manufacture, käyttäjä ohjataan sivulle, jossa hän voi valita mitä laitetta, ja kuinka monta niitä hän haluaa valmistaa. Samalle sivulle ohjataan myös silloin, kun hallintapaneelissa painaa valmista-nappia tuotteen kohdalla tai menee valmistussivulle sivuston vasemmasta valikosta. Laitteen nimi osoitteessa päättää, minkä laitteen ohjeet järjestelmä lataa ja valmistuksen vaihe taas kertoo, mikä työvaihe ladataan näkyville.

Djangon objektit voivat sisältää myös kuvia, joten työvaiheet voitiin tehdä normaaleina objekteina, joka mahdollistaa helpomman muokkauksen ja yksityiskohtaisemman muokkaustenseurannan. Työvaiheista voi nähdä samanlaisen muokkaushistorian kuin

laitteista, eli kuka on muokannut ja mitä he muokkasivat. Työvaiheiden muokkaus helpottuu siinä mielessä, ettei muokkaajan tarvitse enää huolehtia koontaohjeen asiakirjatyylistä, tai muutenkaan muotoilusta, vaan Django hoitaa sen automaattisesti. Muokkaajan täytyy vain kirjoittaa tekstit ja lisätä kuva. Työvaiheiden järjestyksen muokkaus onnistuu myös helpommin asentamalla lisäosa Djangoon, joka mahdollistaa objektien järjestelyn järjestelmävalvojan hallintapaneelista. Näin ollen työvaiheiden järjestely uudessa järjestelmässä toimii hyvin yksinkertaisella vedä ja pudota -menetelmällä, joka on huomattavasti käyttäjäystävällisempi kuin vanha Word-tiedoston manuaalinen muokkaus.

Vanhoissa paperisissa ohjeissa valmistukseen tarvittavat työkalut ja osat on listattu ohjeen alussa yhtenä suurena listana, jos valmistuksen aikana haluaa esimerkiksi tarkistaa mitä ruuvia ohjeessa tarkoitetaan, täytyy se aina tarkistaa ohjeen alusta. Uudessa järjestelmässä työvaiheen vieressä näkyy aina mitä osia, tai työkaluja kyseiseen työvaiheeseen tarvitaan (Kuva 10). Lista muokataan automaattisesti työvaiheen mukaan aina, kun käyttäjä siirtyy seuraavaan vaiheeseen.



Kuva 10. Kuva uusista digitaalisista koontaohjeista.

Jokaisen työvaiheen tiedoissa on kirjattu sen järjestysnumero ja järjestelmä lukee aina senhetkisen työvaiheen järjestysnumeron ja kirjaa sen muuttujaan. Jokaisessa osa- ja työkaluluettelossa on myös vastaava järjestysnumero ja järjestelmä näyttää vain sen luettelon, jonka järjestysnumero vastaa muuttujassa olevaa aktiivisen työvaiheen

järjestysnumeroa. Laitteiden tietoja tallentaessa, myös nykyisen työvaiheen järjestysnumero tallennetaan laitteiden tietoihin, jotta käyttäjän myöhemmin jatkaessa valmistustyötä järjestelmä osaa palata oikeaan työvaiheeseen.

Koontaohjeen alle listataan kaikki keskeneräiset tuotteet ja niiden valmistusvaiheessa muokattavat kentät. Sellaisia kenttiä ei näytetä, joita ei muokata laitteen valmistusvaiheessa, kuten esimerkiksi laitteen asiakas, tai valmistuspäivä, joka kirjataan automaattisesti tuotteen valmistuessa. Sivulla on kaksi nappia: tallenna- ja tallenna ja lopeta -nappi. Tallenna-napista tallennetaan täytetyt tiedot kyseisille laitteille, mutta pysytään samalla sivulla, joka on käytännöllistä silloin, kun halutaan tallentaa tiedot välillä ja jatkaa hetken päästä uudestaan. Tallenna ja lopeta -nappi taas tallentaa tiedot ja palaa takaisin hallintapaneeliin. Ennen hallintapaneeliin palaamista järjestelmä tarkistaa, onko kaikissa kentissä oikeanmuotoiset tiedot ja jos näin on, se poistaa valmistustyön järjestelmästä ja merkitsee laitteet valmiiksi. Samalla järjestelmä myös kirjaa tuotteet valmistetuksi yrityksen kirjanpitosovellukseen rajapintakutsun avulla. Järjestelmässä on myös mahdollista valmistaa vain osa laitteista loppuun asti esimerkiksi osien puutteen vuoksi, sillä järjestelmä laskee, kuinka monessa laitteessa on tarvittavat tiedot täytetty, ja tekee pelkästään niistä valmistuskirjauksen kirjanpitoon.

Jokaiseen laitteeseen liimataan tarra, joka sisältää laitteen tietoja, kuten esimerkiksi sarjanumeron, laitemallin sekä yksilöivän laiteosoitteen. Jotta nämä tiedot on helppo tulostaa, ollaan yrityksessä käytetty tulostusohjelmaa, joka lukee tiedot tuotantotietokannasta Excel-tiedostossa. Uudessa järjestelmässä ei käytetä vastaavaa Excel-tiedostoa, joten tarrojen tulostukseen täytyi keksiä uusi ratkaisu. Djangoon on saatavilla lisäosa, jonka avulla järjestelmästä voi tulostaa ulos laitteiden tietoja CSV-muodossa. CSV-tiedostot toimivat vastaavasti normaalin Excel-taulukon kanssa, joten tietojen tulostus tarroille onnistuu myös uudessa järjestelmässä. Koontaohjeissa ilmestyy nappi tarrojen tulostusvaiheessa, jota painamalla tiedot tulostuvat tiedostoon, ja sen jälkeen käyttäjä voi tulostaa tarrat samaan tapaan, kuin ennenkin.

Käyttäjän poistuessa koontaohjeista järjestelmä varoittaa, jos laitteiden tiedoissa on tallentamattomia muutoksia. Tämä on toteutettu JavaScriptillä asettamalla binaarimuuttuja muotoon false ja aina, jos laitteiden tietokenttiin tekee muutoksia JavaScript vaihtaa muuttujan true-muotoon. Kun käyttäjä poistuu sivulta, JavaScript lukee muuttujan ja jos se on true, se varoittaa käyttäjää, että sivulla on tallentamattomia muutoksia ja kysyy, haluaako käyttäjä silti poistua. Jotta tallennetuista muutoksista ei varoiteta, muuttuja asetetaan false-muotoon aina, kun käyttäjä painaa tallenna tai tallenna ja lopeta -nappia.

7 LOPULLINEN JÄRJESTELMÄ

7.1 Kehityksen lopputulos

Työn tavoitteena oli kehittää uusi tuotannonhallintajärjestelmä elektroniikkalaitevalmistajalle. Uuden järjestelmän oli tarkoitus korvata vanha Excel-tietokanta, paperiset koontaohjeet ja hankala varastokirjanpito.

Tässä opinnäytetyössä järjestelmä saatiin siihen pisteeseen, että yrityksen yhden laitteen valmistus voidaan suorittaa alusta loppuun järjestelmää käyttämällä. Tämä oli alun perin tarkoituskin, sillä näin voidaan siirtää yhden laitteen valmistus kokonaan uuteen järjestelmään ja samalla huomataan mahdolliset puutteet, tai ongelmat ja ne voidaan korjata pienemmässä mittakaavassa ja myöhemmin lisätä muut laitteet järjestelmään. Järjestelmää ei vielä ole saatu siirrettyä tuotannon käyttöön, joten varsinaista laitteen valmistuksen kestoa uudella järjestelmällä ei voida mitata, mutta on silti mahdollista arvioida, kuinka paljon aikaa yleisesti säästyy vanhaan verrattuna.

Ensimmäisenä ajan ja työn säästäjänä uudessa järjestelmässä toimii hallintapaneeli, johon voi kuka vain kirjata uuden tilauksen, kun taas vanha tapa oli tulostaa tilaus aina paperille ja kiinnittää se tuotannon työtilan seinälle. Toinen parannus on se, että keskeneräiset valmistustyöt jäävät hallintapaneeliin esille, kunnes ne tulevat valmiiksi. Näin varmistetaan, että valmistustyöt eivät unohdu eikä jää epäselväksi mihin vaiheeseen ne ovat jääneet. Yksi suurimmista parannuksista ja tuotannon optimoinneista uudessa järjestelmässä on hallintapaneelin laite- ja osaluettelo. Siitä pystyy helposti seuraamaan, mitkä osat tai tuotteet ovat loppumassa ja yritys pystyy hyvissä ajoin tilaamaan uusia osia, tai valmistaa laitteita varastoon. Aikaisemmin jokaisen osan ja laitteen varastotilanteen joutui erikseen katsomaan kirjanpitosovelluksesta tai käymään tuotannon varaston läpi ja laskemaan manuaalisesti. Tämä aiheutti viivästyksiä tuotannossa, kun jonkin osan loppuminen jäi huomaamatta ja laitteiden valmistus joutui odottamaan tilattuja osia, joissa saattoi olla jopa viikkojen toimitusaika.

Tuotantotietokannassa parannettiin tietojen yhtenäisyyttä muuttamalla kaikki kirjatut tiedot samaan muotoon koko tuotannon historian ajalta. Laitteiden valmistuspäivien kirjaus korjattiin myös selkeämmäksi niin, että se kirjataan automaattisesti vasta, kun laite on valmistettu, eikä kuten vanhassa tietokannassa kirjattiin jo aloitettaessa valmistus. Kommentointia ja muokkaushistoriaa kehitettiin kattavammaksi ja käytännöllisemmäksi, sillä

ne olivat todella heikolla tasolla aikaisemmin käytetyssä Excel-tiedostossa. Nyt uudessa järjestelmässä tiedot ovat selkeämmin näkyvissä ja kattavan muokkaushistorian avulla voidaan nähdä tarkalleen, kuka tietoja on muokannut ja milloin.

Valmistuksen puolella suurimpia muutoksia olivat valmistuskirjauksen automatisointi ja kirjattavien tietojen tarkistus. Aikaisemmin valmistuskirjaus tehtiin manuaalisesti kirjanpitosovellukseen, ja siitä koitui paljon ongelmia, kun liian usein se unohdettiin tai kirjattiin väärin. Valmistuskirjauksen automatisointi tarkoittaa sitä, että ainakin teoriassa vääriltä, tai puuttuvilta kirjauksilta pitäisi välttyä ja sitä kautta laitteiden ja osien varastosaldo vastaisi todellisuutta. Kirjattavien tietojen tarkistus taas pitää huolen siitä, ettei pitkissä numerosarjoissa esimerkiksi jää puuttumaan, tai tule liikaa numeroita. Myös toistuvaa tietojen kirjaamista on parannettu niin, että käyttäjän tarvitsee vain valita muutamasta vaihtoehdosta, jokaisen tiedon erikseen kirjaamisen sijasta. Viimeisenä parannuksena mahdollistettiin valmistustyön tallennus keskeneräisenä, joka helpottaa tilanteita, joissa työ keskeytyy esimerkiksi kiireellisemmän työn takia. Tämä on paljon parempi ja luotettavampi tapa kuin vanha, jossa laitteiden päälle asetettiin paperilappu, jossa mainittiin, mihin vaiheeseen työ oli jäänyt.

Kaiken kaikkiaan tuotannon työvaiheista on pystytty karsimaan muutamia pois ja osaa on automatisoitu virheiden välttämiseksi. Näin ollen voidaan olettaa, että uuden tuotannonhallintajärjestelmän kehitys on onnistunut ja tuotannon työnkulku on nopeutunut.

7.2 Järjestelmän kehitys tulevaisuudessa

Tuotannonhallintajärjestelmässä on vielä paljon kehittämistä, ja tässä opinnäytetyössä siitä saatiin lähinnä niin kutsuttu minimum viable product (MVP). Järjestelmä toimii yhdelle laitteelle valmistuksen alusta loppuun asti, mutta yritys valmistaa montaa muutakin laitetta, jotka täytyy kaikki saada lopulta lisättyä järjestelmään. Ensimmäinen tehtävä on siirtää järjestelmä tuotannon käyttöön ja alkaa valmistamaan järjestelmän kautta toimivaa laitetta. Näin ollen nähdään mahdolliset ongelmat tai puutteet järjestelmässä, ja ne voidaan korjata ennen muiden laitteiden lisäämistä järjestelmään.

Kun järjestelmä vaikuttaa toimivan ilman selviä ongelmia, voidaan alkaa lisäämään muita laitteita järjestelmään. Prosessi ei ole kovin vaikea, sillä suurimmalle osalle laitteista se koostuu lähinnä muutamasta koodirivistä, joissa vain kerrotaan, mitä tietoja laitteista kerätään ja koontaohjeiden lisäyksestä järjestelmään. Enemmän työtä tulee tuottamaan

yrityksen valmistama loggerijärjestelmä, joka sisältää monta eri laitetta, joiden valmistusprosessi on hyvin samanlainen. Loggerijärjestelmien tilauksissa on useasti montaa eri laitetta, ja tuotannonhallintajärjestelmän täytyisi osata käsitellä kyseiset tilaukset. Loggerijärjestelmän lisäys tuotannonhallintajärjestelmään tulee kyseeseen vasta viimeisenä, sillä tarkoitus on saada mahdollisimman monta yrityksen valmistamaa laitetta järjestelmään heti, kun järjestelmä on valmis siihen ja muiden laitteiden lisäys uuteen järjestelmään onnistuu nopeammin.

Laitteiden määrän lisääntyessä järjestelmässä hallintapaneelin laite- ja osaluettelo muodostuu epäkäytännölliseksi, jos kaikkien laitteiden osat ja varastosaldot listattaisiin päällekkäin. Tämän takia yksi jatkokehityksen kohteista onkin laite- ja osaluettelon muuttaminen niin, että vain eri laitteiden nimet näkyisivät listassa ja niiden kohdalla joko keltainen tai punainen väri, jos osia on loppumassa tai loppunut. Laitteen nimen vieressä ei näkyisi mitään, jos kaikkia osia on tarpeeksi ja nimeä klikkaamalla avautuisi koko osaluettelo näkyviin. Jotta vähissä olevat osat huomattaisiin paremmin, voitaisiin myös tehdä niin, että koko laite- ja osaluettelon alkuun tulisi näkyviin kaikki keltaisella ja punaisella merkityt osat ja vasta sen jälkeen laitteiden täydet osaluettelot.

Asiakastuen helpottamiseksi tarkoitus olisi kehittää järjestelmään myös tietokanta, josta näkisi esimerkiksi, mitä kaikkia laitteita kenelläkin asiakkaalla on. Se toimisi suurin piirtein samalla pohjalla kuin tuotantotietokanta, ja siinä voisi valita asiakkaan listasta, jonka alle ilmestyisi listaus kaikista valitun asiakkaan laitteista tietoineen. Tämä olisi huomattava parannus nykyiseen, sillä tällä hetkellä tietyn asiakkaan tuotteita ei saa mistään esille samanaikaisesti. Järjestelmän yläpalkissa oleva hakutoiminto olisi tarkoitus myös saada toimivaksi. Haku olisi käytännöllinen silloin, kun haluaa nopeasti löytää esimerkiksi päivitykseen tulleen laitteen tiedot tuotantotietokannasta ilman turhaa selaamista. Djangoon löytyy runsaasti ohjeita, miten yleisen haun saa sivustolla toimimaan, joten sen lisäys järjestelmään ei pitäisi tuottaa suurta ongelmaa.

Kun kaikki yrityksen valmistamat laitteet toimivat uuden tuotannonhallintajärjestelmän kautta tarkoitus olisi automatisoida ainakin osan laitteiden tietojen kirjaaminen järjestelmään valmistusvaiheessa. Tämä toimisi niin, että laitteen ohjelmistoa asennettaessa ohjelmointilaite lukisi laitteesta kaikki tarvittavat tiedot, jotka on mahdollista lukea, ja lähettäisi ne rajapintakutsulla tuotannonhallintajärjestelmään. Näin ollen säästettäisiin taas huomattavasti aikaa ja manuaalista työtä laitteiden valmistuksessa ja pienennettäisiin inhimillisten kirjausvirheiden riskiä.

LÄHTEET

Deloitte 2018. Estonia's fastest growing technology company is Scoro Software. Viitattu 1.7.2020. https://www2.deloitte.com/ee/en/pages/about-deloitte/articles/Fast50-report-2018.html

Microsoft N. d. Mikä ERP on ja miksi sitä tarvitaan. Viitattu 26.6.2020. https://dynamics.microsoft.com/fi-fi/erp/what-is-erp/

Milk, Liis 2019. Scoro Once Again the Easiest to Use Business Software by G2 Crowd. Viitattu 1.7.2020. https://www.scoro.com/blog/scoro-easiest-to-use-business-software-2019/

Mozilla 2020. Django introduction. Viitattu 10.7.2020. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Odoo N. d. Making companies a better place, one app at a time. Viitattu 30.6.2020. https://www.odoo.com/page/about-us

Pietilä, Sami 2020. Pietiko Oy toimitusjohtaja. Haastattelu 29.6.2020.

Ruuse, Liisi 2018. Scoro is the Easiest to Use Business Software by G2 Crowd. Viitattu 1.7.2020. https://www.scoro.com/blog/scoro-easiest-to-use-business-software/

Tukes. 2014. Sähkölaitteiden valmistus, maahantuonti ja myynti. Tukes opas. 6/2014. S. 12. ISBN 978-952-5649-58-1.

Wikipedia 2020. CE-merkintä. Viitattu 26.6.2020. https://fi.wikipedia.org/wiki/CE-merkint%C3%A4