Roope Vaarama

Tuotteen elinkaaren hallintasovelluksen jatkokehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

1.11.2021

Tiivistelmä

Tekijä:	Roope Vaarama
Otsikko:	Tuotteen elinkaaren hallintasovelluksen jatkokehitys
Sivumäärä:	37 sivua
Aika:	1.11.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mobile Solutions
Ohjaajat:	Toimitusjohtaja Mika Maanonen Tutkijaopettaja Hannu Markkanen

---

Insinöörityön aiheena oli tuotteen elinkaarenhallintasovellukset ja toimeksiantajana oli suomalainen informaatioteknologian alan yritys. Aihetta tutkittiin jatkokehityskohteiden määrittelemiseksi ja taustatietojen kartoittamiseksi. Työn tavoitteena oli tuotteen elinkaaren hallintasovelluksen jatkokehityksen määritteleminen, rajapintadokumentointi ja sovelluksen käyttöohjeen luomisen ohje. Lisäksi perehdyttiin tuotteen elinkaaren hallintasovelluksiin kirjallisuuden ja markkinoilla olevien sovelluksien avulla.

Työn alussa tutustuttiin tuotteen elinkaaren- ja tuotetiedonhallinnan kirjallisuuteen sekä markkinoilta löytyviin tuotteen elinkaaren hallintasovelluksiin. Tietoa saatiin kirjallisuudesta ja internetistä eri palveluntarjoajien sivustojen artikkeleista ja videoista. Markkinoilla olevien sovelluksien tutkimista varten hyödynnettiin saatavilla olevien ohjelmistojen testi- tai täysversioita.

Kun kehitettävän sovelluksen tavoitteet saatiin selkeiksi, siirryttiin sovelluksen teknologiaan perehtymiseen, dokumentoitiin ja kehityskohteisiin. Sovelluksen rajapintadokumentaatio ja käyttöohjeen kirjoittamisen ohje toteutettiin onnistuneesti. Ohjeen avulla voidaan kirjoittaa käyttöliittymän käyttöohje. Jatkokehityksen suunnittelemiseen käytettiin hyödyksi lähteistä koottua tietoa ja tutkittuja sovelluksia. Yhdessä kehitystiimin kanssa määriteltiin kehityskohteet, jotka työssä pohjustettiin käyttökohtaisilla vaatimuksilla. Kehityskohteiksi valittiin hakuominaisuudet, revisiointi, lokikirja, käyttöliittymän ulkoasu ja muutostenhallinta.

Työn tuloksena saatiin paljon tietoa muiden yritysten ohjelmistoista ja lähestymistavoista. Tämän perusteella tunnistettiin jatkokehityksen painopiste perusominaisuuksiin ja visuaaliseen esittämiseen. Jatkotutkimuskohteiksi tunnistettiin perehtyminen datan tuonnin rakenteisiin ja suunnittelutapoihin ja lisäksi yritysten eri suunnittelu- ja tuotantoympäristöjen tutkiminen ja niiden perusteella soveltuvia lähestymistapoja tuotetiedon hallintaan. Arviointi yrityksille saavutettavasta lisäarvosta ohjelmiston käyttöönotolla on myös mahdollinen kohde.

Avainsanat: PLM, JavaScript, ohjelmistokehitys, ReactJS, TypeScript, ERP, web, tuotteen elinkaaren hallintasovellus, jatkokehitys

Abstract

Author:	Roope Vaarama
Title:	Product Lifecycle Management Application Development
Number of Pages:	37 pages
Date:	1 November 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Mobile Solutions
Supervisors:	Mika Maanonen, CEO
	Hannu Markkanen, Researching Lecturer

This thesis is about Product Lifecycle Management Applications and it was ordered by Paisto Oy. The subject was researched for further development and background information. The objective of the thesis was determination of further development, creation of API documentation and user manual guide. Another goal for the study was to research Product Lifecycle management literature and available software in the market.

The thesis started with researching product lifecycle and product management literature and available software of product lifecycle management from the market. Information was gathered from literature and software providers' internet pages. Information about product lifecycle management software on the market was found via demos or actual software that were available for free.

This study introduces technology used in the project, documentation, and planning of future development subjects. API documentation and a user manual guide were created successfully. For future development, features such as search, revision, log, user interface and change management were chosen. The knowledge from previous features was used to create the plan of features to implement into the software.

As result of the thesis, information of different software used by companies was gathered. Based on the results, it seems to be a good practice to keep the focus of software development on basic features and visual presentation. Further research subjects could be data planning and importing it to different structures. Also gaining knowledge of different design and production environments to find the best approach in PLM needs more research. This information could be used to appraise the added value to the company if they start to use our PLM software.

Keywords: PLM, JavaScript, Software development, ReactJS, TypeScript, ERP, Web, Product lifecycle management, development

Sisällys

Lyhenteet ja käsitteet

PDM: Tuotetiedon hallinta (Product Data Management)

ERP: Toiminnanohjausjärjestelmä (Enterprise Resource Planning)

PLM: Tuotteen elinkaaren hallinta (Product Lifecycle Management)

ReactJS: ReactJS eli myös nimellä React on avoimen lähdekoodin JavaScript-kirjasto käyttöliittymien ja komponenttien rakentamiseen

CSS: Verkkosivuille kehitetty tyylisivu (Cascading Style Sheets)

FP: Funktionaalinen ohjelmointi (Functional Programming)

OOP: Olio-ohjelmonti (Object Oriented Programming)

FRP: Funktionaalinen reaktiivinen ohjelmointi (Funtional Reactive Programming)

NestCLI: Komentorivin käyttöliittymän työkaluohjelmisto

GNU GPL: GNU GPL tai pelkkä GPL on vapaiden ohjelmistojen julkaisemiseen tarkoitettu lisenssi (General Public License)

SaaS: Ohjelmisto tarjottuna palveluna (Software as a service)

HTTP: Hypertekstin siirtoprotokolla, jota selaimet ja palvelimet käyttävät tiedonsiirtoon (Hypertext Transfer Protocol).

CRM: Asiakkuudenhallinta (Customer relationship management).

BOM: Tuoterakenne eli suunnitteluohjelman luoma osarakenne tuotteesta (Bill of Material).

ECO: Muutosmääräyksen avulla ilmoitetaan tuotteeseen tulevista muutoksista (Engineering Change Order).

1 Johdanto

Insinöörityön aiheeksi valikoituivat tuotteen elinkaaren hallintasovellukset, koska työn tilaajayrityksen oman sovelluksen jatkokehitys on koko ajan käynnissä ketterällä mallilla. Työn avulla on tarkoitus kerätä olennaista tietoa aiheesta ja markkinoilla olevista sovelluksista sekä antaa uutta näkökulmaa olennaisille ominaisuuksille ja kehitystavoille.

Työssä on tavoitteena saada kattava käsitys eri yritysten markkinoilla olevista sovelluksista sekä tunnistaa niiden vahvuuksia ja heikkouksia. Kun työssä on saatu selville sovelluksen tarpeelliset ominaisuudet, luodaan jatkokehityssuunnitelma Paisto Oy:n sovellusta varten. Tarpeena on myös luoda ohje käyttöliittymän käyttöohjetta varten. Näiden lisäksi selvitetään rajapinnan dokumentointia varten soveltuvat työkalut ja toteutetaan dokumentaation luonti. Lopputuloksena saadaan kattava tietopohja tuotteen elinkaaren hallintasovelluksesta, sen käyttötarkoituksesta ja muiden yritysten lähestymistavasta, käyttöohjeen luontiin tarvittava ohje, rajapintadokumentaatio ja jatkokehityssuunnitelma.

Tuotteen elinkaaren hallintasovellus (Product Lifecycle Management, PLM) pyrkii hallitsemaan kaikki tuotteeseen liittyvät tiedot ja suunnitteluprosessit. Ohjelmisto kerää kaikki tiedot tuotteen elinkaaren vaiheista, jotka ovat määrittely, suunnittelu, tuotanto, huolto ja käytöstä poisto. PLM huolehtii, että tuotetietojen säilytys ja hallinnointi löytyvät yhdestä paikasta ja ovat helposti saatavilla sekä muokattavissa. PLM käyttää hyödykseen myös tuotetiedon hallinta (Product Data Management, PDM)- ja toiminnanohjaus (Enterprise Resource Planning, ERP) -järjestelmiä.

Insinöörityön kohdeorganisaationa on informaatioteknologian alalla toimiva vuonna 2017 perustettu Paisto Oy. Se on erikoistunut teollisuuden ohjelmistojen ja konsultoinnin räätälöityihin toteutuksiin yritysasiakkaille ja työllistää tällä hetkellä neljä työntekijää. Yrityksen työntekijät työskentelevät joustavasti toiveidensa mukaan ja pääasiassa etätyönä. Paisto Oy tarjoaa

ohjelmistokehityksen lisäksi teollisuusyritysten tiedonhallinnan analyysipalveluita, kehitystarpeiden määrittelyä ja projektointeja. (18.)

2 Tuote ja tuotetieto

2.1 Tuotteen elinkaaren hallinta

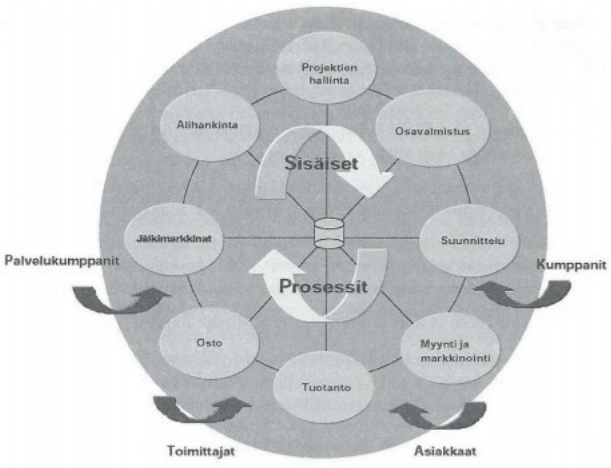
PLM on lyhenne englannin kielen sanoista Product Lifecycle Management eli suomennettuna tuotteen elinkaaren hallinta. Ennen tuotteen elinkaaren hallinnan käsitettä viitattiin tuotetiedon hallintaan. Tuotteiden ja komponenttien elinkaaret lyhenevät samaan aikaan, kun uusien tuotteiden vieminen markkinoille nopeutuu entisestään. Tämä johtaa yritykset muodostamaan verkostoja, joissa kukin valmistaja keskittyy tiettyyn osa-alueeseen suunnitellussa tai tuotannossa. Tiedot tuotteesta pitää pystyä jakamaan nopeasti, virheettömästi ja automaattisesti, jotta pystytään tehokkaasti kilpailemaan kansainvälisillä markkinoilla. Uusia ja parempia tuotteita pitää pystyä tarjoamaan entistä nopeammin, paremmalla tuotolla ja vähemmällä työllä. Tässä laajassa yritysverkostossa tieto pitää pystyä jakamaan sähköisesti ja tietoturvallisesti. PLM voidaan siis tulkita yhteistyöverkoston työkaluksi. (1, s. 6.)

Tuotteen elinkaaren hallinta käsitetään yleisimmin ohjelmistoksi tai järjestelmäksi, joka kattaa kaiken tarpeellisen tiedon tuotteen elinkaaresta. PLM on yleinen prosessi tekniikassa ja valmistuksessa tuotetiedon hallitsemista varten läpi sen elinkaaren. Prosessi kattaa yleensä määrittelyn, suunnittelun, tuotevaatimukset, piirustukset, tekniikan, valmistuksen, tuotannon, huollon, ylläpidon, käytöstä poiston ja uudelleenvalmistamisen. Prosessin ja tuotteen kaikki tieto tulisi saada tallennettua yhteen paikkaan, josta se on vaivattomasti saatavilla kaikille sitä tarvitseville. Tämä toteutetaan, jotta jokapäiväinen käyttö on vaivatonta ja nopeaa. Edellä mainitut asiat puoltavat PLM-järjestelmän ydinideaa. Elinkaaren perusvaiheet näkyvät kuvassa 1. Kaikki tuotteen elinkaareen liittyvät tiedot on tärkeää saada tallennettua yrityksessä. Tiedot löytyvät yhdestä järjestelmästä ja ovat vaivattomasti saatavilla. (1, s. 45.)

Kuva 1. Esimerkki tuotteen elinkaaren vaiheista.

2.2 Tuotetiedon hallinta

PDM on lyhenne englannin kielen sanoista Product Data Management eli suomennettuna tuotetiedon hallinta. Tuotteiden ja komponenttien elinkaaret lyhenevät jatkuvasti, ja samalla uusia tuotteita on saatava entistä nopeammin markkinoille. Tämän vuoksi yritykset muodostavat verkostoja, joissa toimittajat erikoistuvat tuotteiden tiettyihin osa-alueisiin. Yhteistä lopputuotetta koskevien tietojen täytyy kulkea näitten yritysten välillä nopeasti, virheettömästi ja automaattisesti, jotta pystytään kilpailemaan kansainvälisillä markkinoilla. Kuvassa 2 näkyvät PDM-järjestelmän alueet valmistavan teollisuuden yrityksessä. (2, s. 9.)



Kuva 2. PDM-järjestelmän alueet valmistavan teollisuuden yrityksissä (2, s. 21).

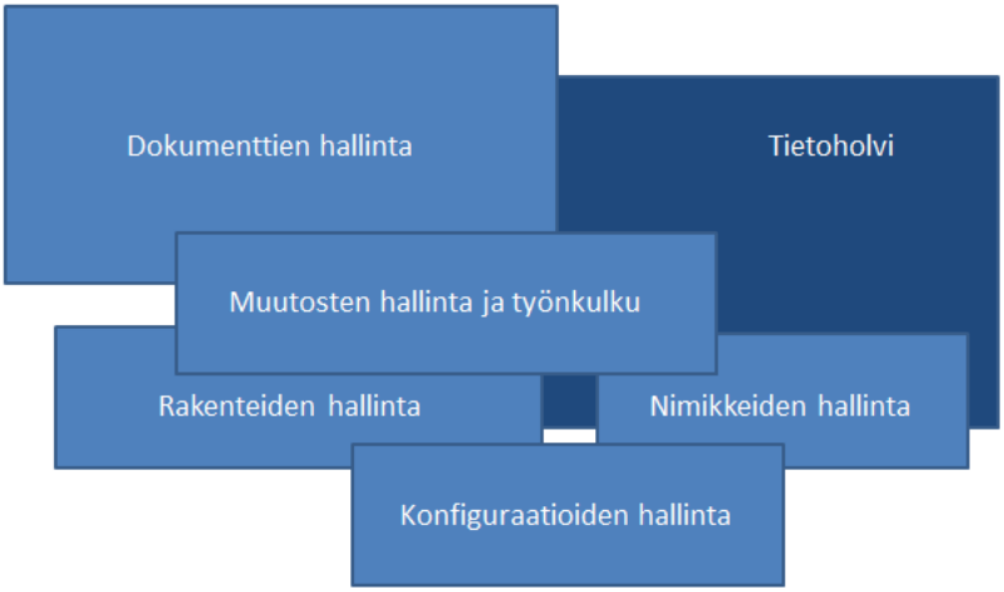
Kenneth McIntoshin määritelmän mukaan tuotetiedon hallinta on ”systemaattinen tapa suunnitella, hallita, ohjata ja valvoa kaikkea sitä tietoa, jota tarvitaan tuotteen dokumentoimiseksi, tuotteen kehittämis-, suunnittelu-, valmistus- ja

testausprosessien sekä käytön aikana, tuotteen koko elinkaaren ajan” (2, s. 18). Tämän määritelmän perusteella huomataan, että lähdeteoksessa käytetään termiä PDM, vaikka siinä huomioidaan tuotteen elinkaariajattelua. Teoksen alussa myös mainitaan, että PDM:a voidaan käyttää puhtaasti tuotetiedon hallintaan tai myös elinkaaren hallintaan, ja yritetään vakiinnuttaa alan vaihtuvaa ja epämääräistä terminologiaa. (2, s. 9.)

Tuotetiedonhallintajärjestelmiä otetaan käyttöön eri yrityksissä eri syistä, riippuen yrityksen toimialasta, tuotteista ja ennen kaikkea siitä, mihin järjestelmää halutaan käyttää. Järjestelmän tarkoituksena on tuoda kaikki tuotteeseen liittyvä tieto yhteen paikkaan, jossa sen oikeellisuus, ajantasaisuus ja saatavuus ovat mahdollisia. Monissa isoissa yrityksissä ongelmana on tietojen säilytys: jos tiedot tallennetaan paikalliselle kiintolevylle, ne eivät ole kaikkien saatavilla ja aina uusimpia versioita. Monille yrityksille PDM on työkalu selvitä päivän töistä tehokkaammin ja toisille se on investointi tulevaisuutta varten. Asiakkaat odottavat jatkuvasti parempia, kehittyneempiä tuotteita ja niihin räätälöintimahdollisuuksia. Asiakkaiden tarpeisiin vastaamista varten tarvitaan järjestelmä, jossa tieto on heti saatavilla uusimpien versioiden ja räätälöintien kanssa. (2, s. 28.)

2.3 Tuotetietojärjestelmä

Tuotetietojärjestelmässä on usein laaja kokonaisuus toimintoja ja ominaisuuksia, joilla pyritään tukemaan tiedon luomisen, tallentamisen, päivittämisen, jakelun, hyödyntämisen ja etsinnän prosesseja. PLM- ja PDM-järjestelmät yleensä sisältävät kyseisen tuotetietojärjestelmän jollain tapaa ja järjestelmästä löytyvät kuvan 3 mukaiset tyypilliset ominaisuudet. (1, s. 13–16.)



Kuva 3. Tuotetietojärjestelmän hallinnan osa-alueet (1, s. 16).

Nimikkeiden hallinta on yksi järjestelmän perustoiminnoista. Nimikkeillä tarkoitetaan tuotteesta luotua yksilöllistä tietoa, jonka sisällä ovat tuotteen ominaisuudet. Järjestelmällä hallitaan nimikkeiden tietoja ja elinkaarta sekä kontrolloidaan käyttöoikeuksien ja muutoshallinnan kanssa nimikkeiden luomiseen ja ylläpitoon liittyviä prosesseja.

Tuoterakenteen hallinta ja ylläpito tunnistaa yksittäiset tiedot ja niitten yhteydet toisiin tietoihin tuoterakenteen avulla, joka muodostuu hierarkkisesti yhteen liitetyistä nimikkeistä.

Käyttöoikeuksien hallinnan avulla määritellään käyttäjien oikeudet järjestelmän hallitsemaan tietoon. Oikeuksia voidaan määritellä sen mukaan, ketkä saavat luoda uutta tietoa, tehdä muutoksia, tarkastaa ja hyväksyä tehtyjä muutoksia sekä vain katsella järjestelmän piirissä olevia tietoja tai dokumentteja.

Dokumenttien ja nimikkeiden tilan, statuksen ja ylläpidon avulla järjestelmä saa tietoa kunkin dokumentin ja nimikkeen tilasta ja versiosta sekä tilaan tehdyistä muutoksista: kuka teki, mitä teki, milloin.

Tiedonhaku on yksi järjestelmän päätehtävistä, ja se tehostaa ja helpottaa tiedonkulkua prosessien aikana. Luomisvaiheessa hyvä tiedonhaku mahdollistaa uusien nimikkeiden luomisen sijasta käyttämään olemassa olevia päivitettyjä ja laadukkaiksi todettuja nimikkeitä. Haun avulla voidaan helposti tuoda esille yksittäiseen nimikkeeseen liittyvät tuotteet tai kokoonpanot sekä nimikkeeseen liittyvät suunnitelmat ja dokumentaatiot.

Muutosten hallinta on keskeinen työkalu elinkaaren tiedon oikeudellisuuden varmistamiseksi. Tämä mahdollistaa tuotteista ja osista sen viimeisimmän tiedon ja tarkistuksen, kuka on tehnyt ja millaisia muutoksia viimeksi ja kuka on muutosten hyväksyjä.

Konfiguraation hallinta mahdollistaa samaan käyttötarkoitukseen tarkoitetun tuotteen räätälöinnin asiakkaan toiveiden mukaan, fyysisiä ominaisuuksia muuttamalla, niille sopivilla vaihtoehtoisilla kokoonpanoilla tai komponenteilla.

Viestien hallinta on järjestelmän perusominaisuus, jolla mahdollistetaan organisaation tiedonvälitys tehostamista varten erityisesti hajautetussa toimintaympäristössä, jopa maailmanlaajuisesti.

Tiedostojen/dokumenttien hallinta yleensä pitää sisällään indeksitietoa järjestelmän sisältämistä tiedostoista. Kyseessä on toisin sanoen metatieto, joka tarkoittaa tietoa siitä, missä tieto sijaitsee.

Tiedon katoamisen esto päivitysten aikana. Tiedostoja kopioidessa ohjelmisto valvoo, että alkuperäinen kopio säilyy niin kauan, kunnes tiedostot on onnistuneesti päivitetty.

Varmuuskopioiden hallinnan järjestelmä pitää automaattisia varmuuskopioita ja lokia tehdyistä varmuuskopioista sekä mahdollisista muokkauksista varmuuskopioihin.

Lokikirjanpito pitää sisällään tehdyt muutokset tietokantaan, jotta kaikki hallinnan piirissä olevat muutokset tallentuvat ja voidaan tarvittaessa jäljittää.

Tietoholvi eli tiedostojen tallennuspaikka on paikka, johon varsinaiset data- ja liitetiedostot tallennetaan.

3 PLM-ohjelmistot

Tässä luvussa tutkitaan kolmea erilaista markkinoilla olevaa tuotteen elinkaaren hallintasovellusta sekä niiden ominaisuuksia ja toiminnollisuuksia. Sovelluksiksi valittiin nopeasti kasvavan yrityksen Roiman Aton, avoimen lähdekoodin ilmainen Odoo ja Siemensin Teamcenter X. Tutkimista vaikeutti se, että vain kahteen kolmesta sovelluksista on suora pääsy ja siksi Roiman Aton-ohjelmistoa varten jouduttiin perehtymään internetistä löytyviin artikkeleihin, videoihin ja arvosteluihin. Odoo avoimen lähdekoodin ohjelmistona mahdollisti vapaan pääsyn ohjelmistoon ja lisäksi verkkosivuston kautta oli saatavilla kokeiluversio. Siemensin Teamcenter X:ää varten oli saatavilla 30 päivän kokeilujakso rajoitetuilla ominaisuuksilla, ja sen avulla yritettiin saada mahdollisimman kattava käsitys ohjelmiston pätevyydestä ja lisäksi haettiin tietoa internetistä ominaisuuksia varten.

3.1 Roima Aton -ohjelmisto

Roima Intelligence Inc. tarjoaa Aton-nimistä PLM-ohjelmistoa. Ohjelmisto on saatavilla SaaS-palveluna RoimaCloudin kautta pilvipalveluna tai RoimaSoftwaren kautta yrityksen omaan ympäristöön.

RoimaCloud-pilvipalvelussa saa tuotteen nopeasti käyttöön ja maksaa palvelusta käyttäjämäärän mukaan, joten kuukausitasolla kustannukset ovat ennustettavissa ilman suuria vaihteluja vuosittain. Palveluun liittyvät järjestelmät ja ylläpito ovat Roiman vastuulla eivätkä vaadi asiakkaalta osaavia henkilöitä tähän. Roima tarjoaa järjestelmän integroimista muihin järjestelmiin. Roima ylläpitää kaikkia pilvipalveluna tarjoamiaan Aton-järjestelmiä vakioidussa palvelinympäristössä, mikä takaa ohjelmiston toimivuuden, ja datan varmuuskopiointi on erittäin helppoa. Tämän lisäksi asiakkaalla on aina käytettävissä uusin versio.

RoimaSoftware tarjoaa Aton-järjestelmää yrityksen omaan ympäristöön asennettaviksi, ja tuote räätälöidään asiakkaiden tarpeiden mukaan. (17.)

Käyttöönotto

Roima Aton -sovelluksen käyttöönottoa ei voitu työssä käydä läpi, koska testiversiota ei ollut saatavilla. Vertailua varten tieto sovelluksesta haettiin eri lähteistä, joista yleisimmät olivat Roiman omat sivut, webinaarit ja Youtube-videot.

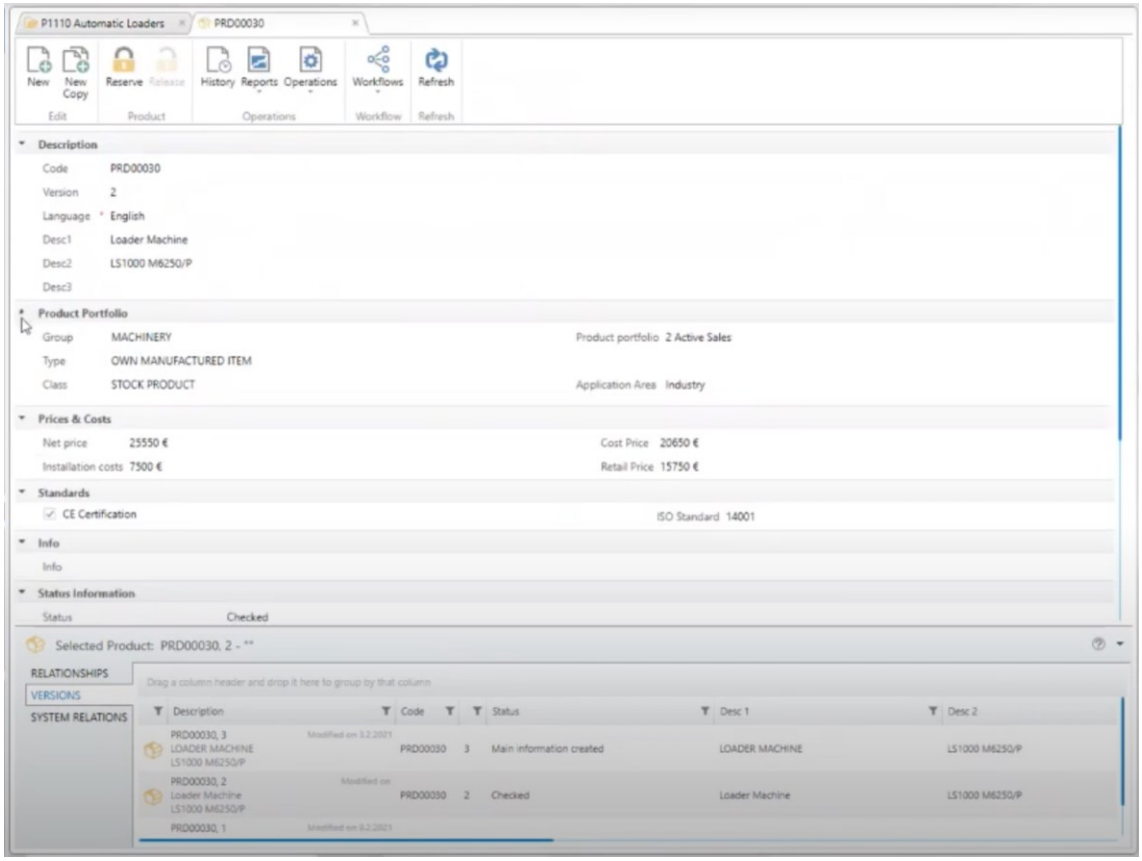
Ominaisuudet

Tiedonhaku

Tiedonhakua varten Aton-sovelluksen oikeassa yläkulmassa on hakukenttä, joka on joskus pienenetty vain suurennuslasiksi. Hakukenttään voi kirjoittaa hakusanan, joka etsii vastaavia tietoja nimikkeiden nimistä, selitteistä, tyypeistä ja muista tiedoista. Hakukentän vasemmalta puolelta löytyy heti kaksi suodatinta, joista toisella voi suodattaa, kuka on nimikkeen omistaja, ja toisessa voi valita nimikkeen kategorian. (15.)

Tuotetieto

Atonin tuotetiedot löytyvät tuoterakenteen tuotteista tai hakutoiminnallisuudella. Kuvassa 4 on esimerkki ohjelmiston tuotetietosivusta, joka sisältää tuotteen kuvauksen, tuotekoodin, version, kielen, osto- ja tuotantohinnan, vaatimukset, portfolion, standardit, sertifikaatit, tilan, muokkauspäivät ja muokkaajat sekä työnkulun. Tuotteelle on myös täysi versiointituki, eli tuotteessa näkyvät versiointivälilehdellä kaikki vanhat sekä mahdolliset uudemmat versiot. Tuotteelle on myös järjestelmän relaatiot, joista löytyy tieto, mihin tuotantorakenteisiin tuote on liitetty. Yhteysvälilehdeltä löytyy eri materiaaleja, projekteja ja muita yhteyksiä, joihin tuote on liitetty. (15.)



Kuva 4. Aton-ohjelmiston tuotetietosivu (16).

Tiedosto

Aton-järjestelmästä on dokumenttienhallinta, joka on yksi tärkeimmistä ominaisuuksista tuotetietojen hallinnassa. Tuotteeseen voi liittää yrityksiä, projekteja, muutoslomakkeita, toisia tuotteita, sertifikaatteja ja tarpeellisia dokumentaatioita. Dokumenttien yhteiskäyttö on helposti hallittavissa ja visuaalisesti toteutettu työtila- ja kansiorakenteiden avulla. (15.)

Muutostenhallinta

Tuotteisiin liittyvät muutospyynnöt ja palautteet hallinnoidaan keskitetysti Aton-järjestelmässä. Täten muutoksiin aina löytyvät kaikki vaiheet ja vastuuhenkilöt, mikä pitää tuotteen ajan tasalla ja näyttää tuotteeseen liittyvät tapahtumat. (15.)

Bom

Tuoterakenteet luodaan joko manuaalisesti sovelluksessa tai ne voi vaihtoehtoisesti tuoda CAD-järjestelmästä, joita Aton-ohjelmisto tukee. Tuoterakenteiden muokkaus onnistuu suoraan ohjelmiston kautta. (15.)

Datan tuonti

Aton-järjestelmä mahdollistaa integraatioita mekaniikkasuunnitteluohjelmistoihin, kuten SolidWorks, Invertor, AutoCad, Catia, Vertex, ProE/Creo ja Microstation, sähkö- ja elektroniikkasuunnittelua varten CADS, ePlan, E3 ja Pads sekä toiminnanohjausjärjestelmien IFS, Lean, SAP, Oracle, Microsoft Dynamics, Epicor ja Lemonsoft integraatiot. Näiden integraatioiden avulla saadaan luotua tehokas tuotteen elinkaaren hallinta, jolla saa kaiken tarvittavan tiedon ja datan tuotua samaan paikkaan käytettäväksi. (15.)

Datan vienti

Aton-järjestelmä mahdollistaa tiedostojen, tuotteiden ja tuoterakenteiden viemisen eri muodoissa, kuten Excel, PDF ja Word (15).

3.2 Odoo-ohjelmisto

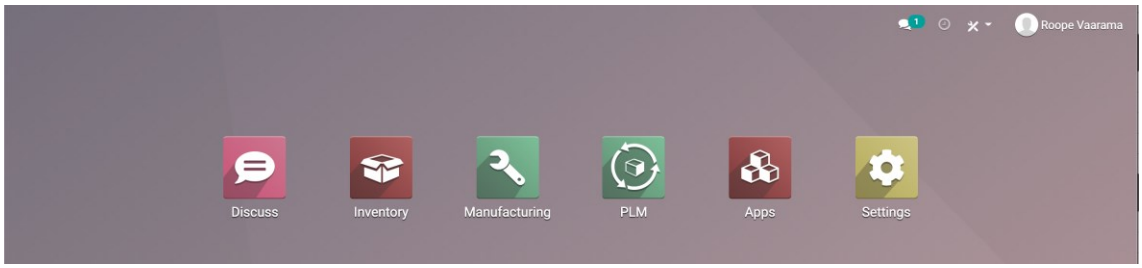
Odoo on avoimen lähdekoodin yritysohjelmisto, joka tarjoaa monipuolisia liiketoimintasovelluksia. Se sopii parhaiten pienille ja keskisuurille yrityksille, koska ohjelmiston käyttöönotto on nopeaa ja ilmaista. Odoon laaja liiketoimintasovellustarjonta pitää sisällään ohjelmistot tuotetiedonhallintaan, asiakkuudenhallintaan, verkkosivuihin, verkkokauppoihin, laskutukseen, kirjanpitoon, valmistukseen, varastohallintaan ja projektihallintaan. Tässä vertailussa keskitytään vain tuotetiedon hallintasovellukseen. (9.)

Odoo on saatavilla kahtena eri palveluversiona: Odoo Enterprise ja Community. Enterprise-versio on lisensoitu ja maksut alkavat 300 eurosta käyttäjää kohti vuodessa ja vaativat vähintään viisi käyttäjää. Versio pitää sisällään kaikki

sovellukset, ja ylläpito on asiakkaan vastuulla. Odoo Community on avointa lähdekoodia, jossa palveluntarjoaja huolehtii ylläpidosta ja tiedon säilytyksestä. Odoo Community on ilmainen ja vaatii käyttäjän tiedot rekisteröitymistä varten. Versiossa on rajoitetut ominaisuudet ja ainoastaan verkkoselaimen versio. (13.)

Käyttöönotto

Työssä käytetään vertailua varten Odoon pilvipalveluversiota, joka toimii suoraan selaimesta. Kuvassa 5 näkyy etusivu käytössä olevilla palveluilla. Ilmaisia palveluita ovat kuvassa näkyvät osat ja ”Apps”-sovellusvalikon alta saa lisättyä kuukausiveloitukseen perustuvia lisäosioita.



Kuva 5. Odoon etusivu (13).

Ominaisuudet

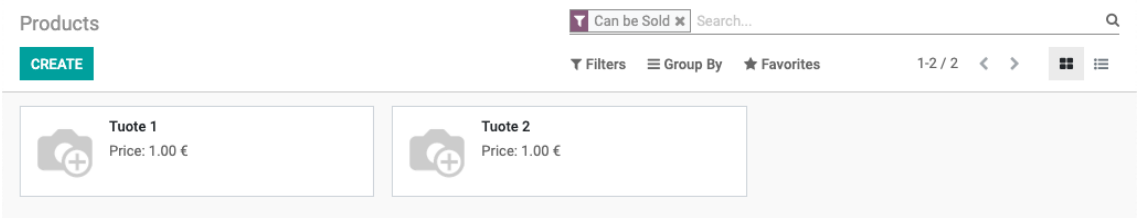
Odoon avatessa huomaa, että sovellus on tehty erittäin yksinkertaiseksi ja helppokäyttöiseksi. Sovellukseen on yksinkertaista luoda tuotteita ja tuoterakenteita. Sovelluksen keveys ja helppokäyttöisuus soveltuu hyvin pienille ja keskisuurille yrityksille.

Tiedonhaku

Järjestelmässä ei ole saatavilla yleisellä tasolla tiedonhakua, vaan tiedonhaku toimii jokaisessa osiossa sen keskeisille asioille. Osioissa oleva tiedonhaku sisältää suodatuksen, ryhmittelyn ja suoritetun haun tallennuksen uudelleenkäyttöä varten.

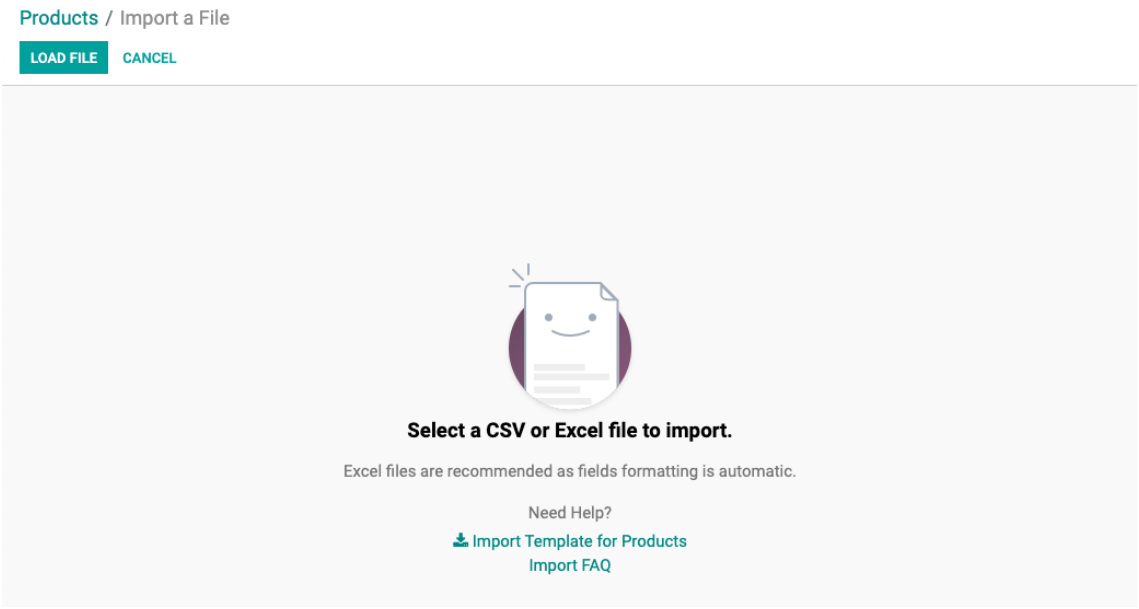
Tuotetieto

Tuotetieto löytyy sovelluksessa valikosta ”Master Data” ja sen alta tulevasta alavalikosta ”Products”. Kuvassa 6 näkyy tuotelistaus ruudukkonäkymässä, oikealta voi myös valita tuotteet listana. Tuotteita voi hakea, suodattaa, ryhmitellä ja lisätä hakuvalinnat suosikeiksi, minkä avulla voi nopeasti aina suorittaa saman haun.



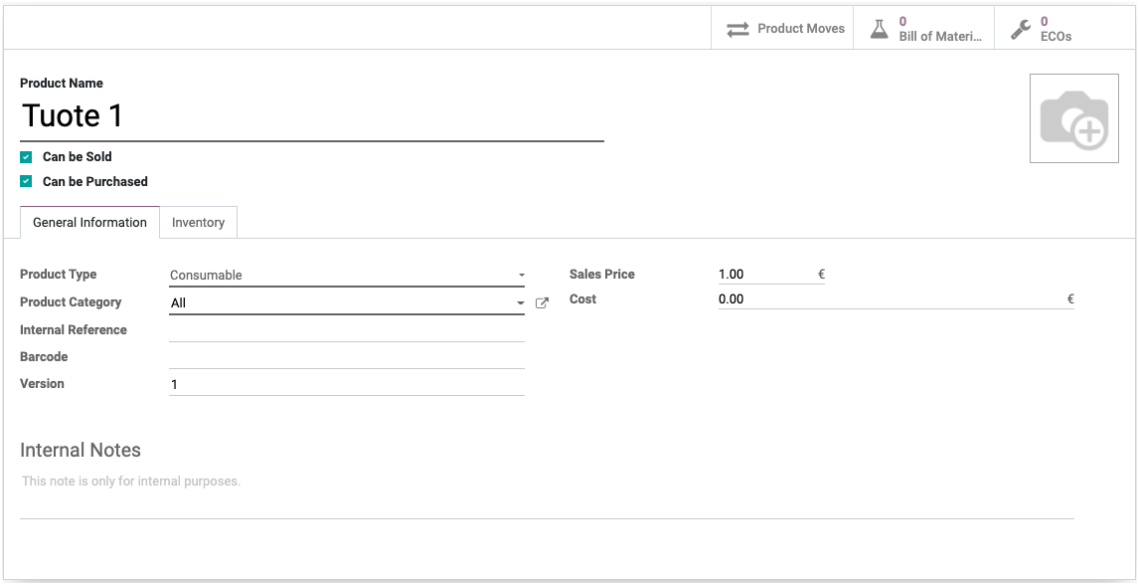
Kuva 6. Odoon tuotenäkymä (13).

”Favorites”-painikkeen alta löytyy myös ”Import records”, jonka avulla voi tuoda tuotteita. Kuvassa 7 näkyy tuotteidentuontisivu, josta voi ladata malli-Excel-tiedoston. Excel-tiedostoon voi syöttää tuotteita ja niiden tietoja. Sivun vasemmasta reunasta ”Load file” -painikkeesta voi ladata täytetyn Excelin, joka tuo tiedot järjestelmään.



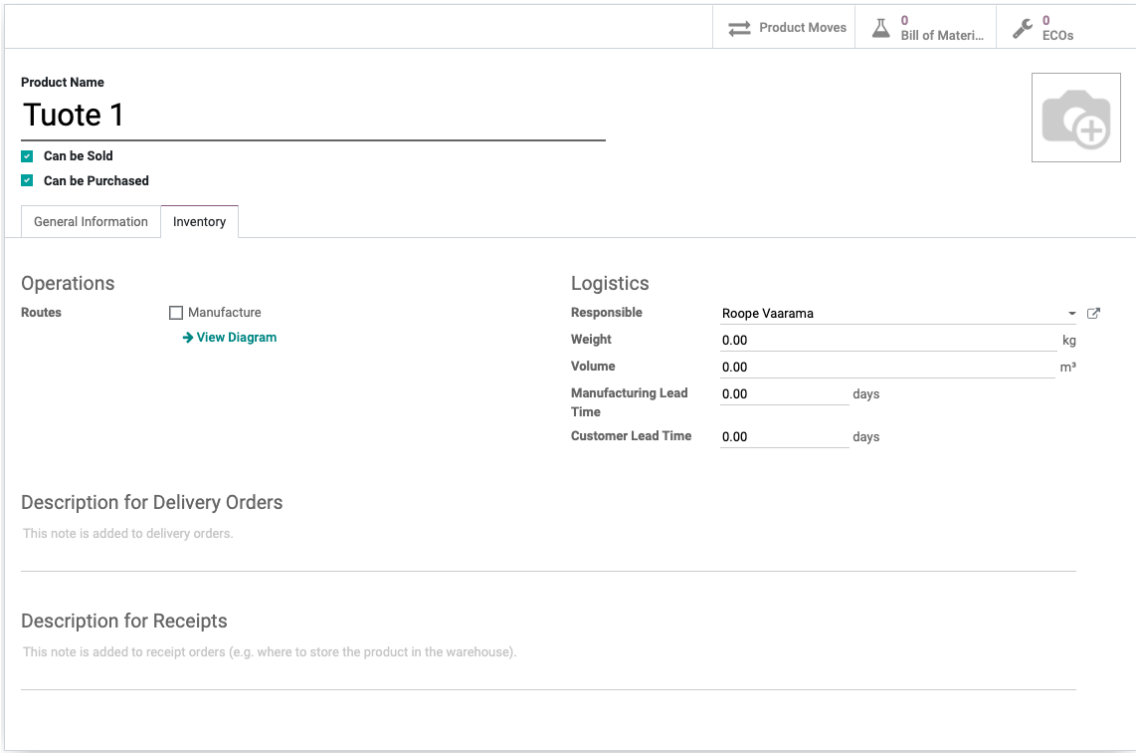
Kuva 7. Odoo-ohjelmiston tuotteidentuontisivu (13).

Kuvassa 8 näkyvät kaikki tiedot, joita tuotteelle voi antaa eli nimen, valinnan, voiko tuotetta myydä tai ostaa, tyypin, kategorian, sisäisen koodin, viivakoodin, version, myyntihinnan, valmistuskustannukset ja sisäisiä muistiinpanoja.



Kuva 8. Odoo-ohjelmiston tuotteen tietosivu (13).

Tuotteelle on myös kuvan 9 mukaiset varastotiedot, josta löytyvät tuotteen vastuuhenkilö, paino, tilavuus, valmistusaika, toimitusaika, toimitusreitti, toimitus muistiinpano ja kuittiselite.



Kuva 9. Odoo-ohjelmiston tuotteen varastotiedot (13).

Tuotteen yläreunasta näkyy, mihin tuotantorakenteisiin tuote on liitetty, ja muutosmääräysten määrä sekä esikatselu. Tuotteen poisto ei onnistu, jos tuote on yhdistetty tuoterakenteeseen.

Tiedosto

Tiedostoja voi liittää lokikirjassa oleviin muistiinpanoihin, joka on käytössä kaikissa ominaisuuksissa. Tiedostoja varten ei ole hakutoimintoja.

Muutostenhallinta

Odoo-ohjelmisto mahdollistaa muutostenhallinnan ECO- eli muutosilmoituksilla. Muutosilmoituksella annetaan otsikko, tyyppi, tuote, tuoterakenne,

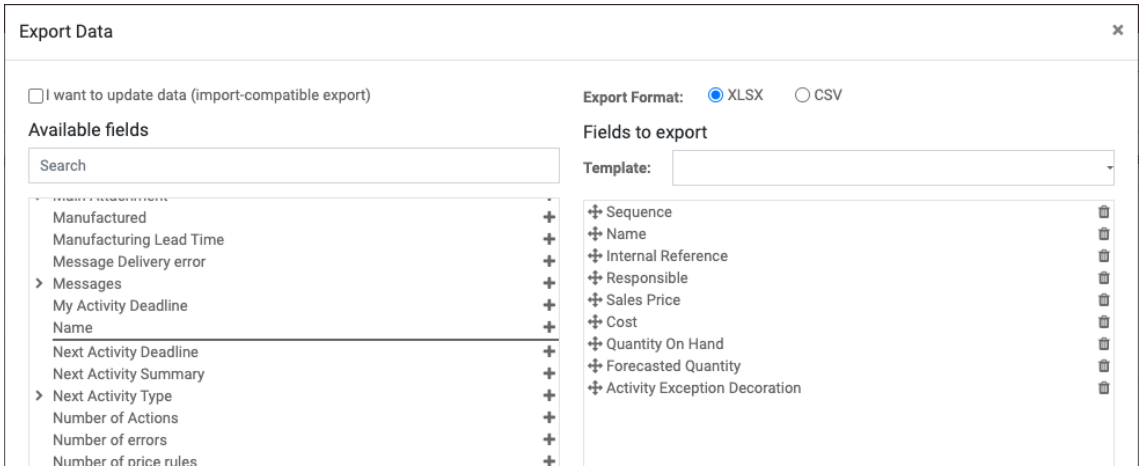
vastuuhenkilö, voimaantuloaika, tagi, muistiinpano ja liittyykö se tuotteeseen vai tuoterakenteeseen. Muutosilmoituksista jää revisiointihistoria tuotteelle tai tuoterakenteelle.

BOM

Tuoterakenteet luodaan suoraan sovelluksessa, tai vaihtoehtoisesti rakenteen voi tuoda Excel-tiedostossa olevilla tiedoilla, johon löytyy esimerkkipohja. Tuoterakenteella on kaksi eri tyyppiä, tuotettava tuote tai tuotesarja. Tuoterakenteelle voi antaa nimen, määrän, referenssin, version ja siihen kuuluvat osat tai komponentit. Tuoterakenteesta näkee suoraan varastotilanteet kaikille rakenteen osille vaadittavien ja saatavilla olevien suhteen.

Datan vienti

Tietojen vientiä varten sovelluksessa voi valita tietorivin tai rivejä, jotka voi viedä Excel- tai CSV-muodossa valittujen kenttien kanssa, kuten kuvassa 10 näkyy.



Kuva 10. Odoo-ohjelmiston datanvientiasetukset (13).

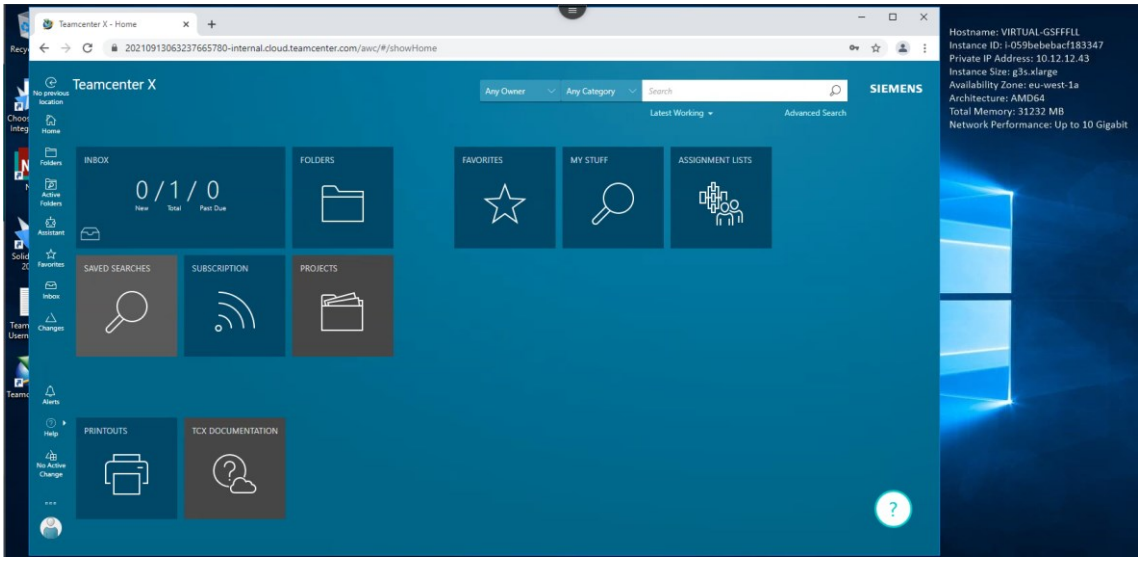
3.3 Siemens Teamcenter X -ohjelmisto

Siemens tarjoaa Teamcenter X -nimistä PLM-ohjelmistoa. Ohjelmisto on saatavilla SaaS-pilvipalveluna. Siemens mainostaa sen avulla käyttäjän säästävän

kuluissa ja maksavan vain siitä, mikä on tarpeen. Palvelumuodossa saatavilla oleva ohjelmisto myös siirtää ylläpidon pois käyttäjän vastuulta. (10.)

Käyttöönotto

Teamcenter X:n käyttöönotto on erittäin yksinkertaista: se vaatii vain selaimen, joka tukee HTML5:tä. Demoversiossa ohjelmistoon pääsy toimii virtuaalikoneen kautta, minkä näkee kuvasta 11 sovelluksen etusivulla. Tietoa itse yrityksen käytössä olevista ympäristöistä ei löytynyt internetistä, joten jäi auki, toimiiko Teamcenter aina virtuaalikoneen kautta vai onko yrityksillä suora pääsy selaimesta omaan ympäristöönsä.



Kuva 11. Teamcenter X:n aloitusnäkymä virtuaalikoneessa (10).

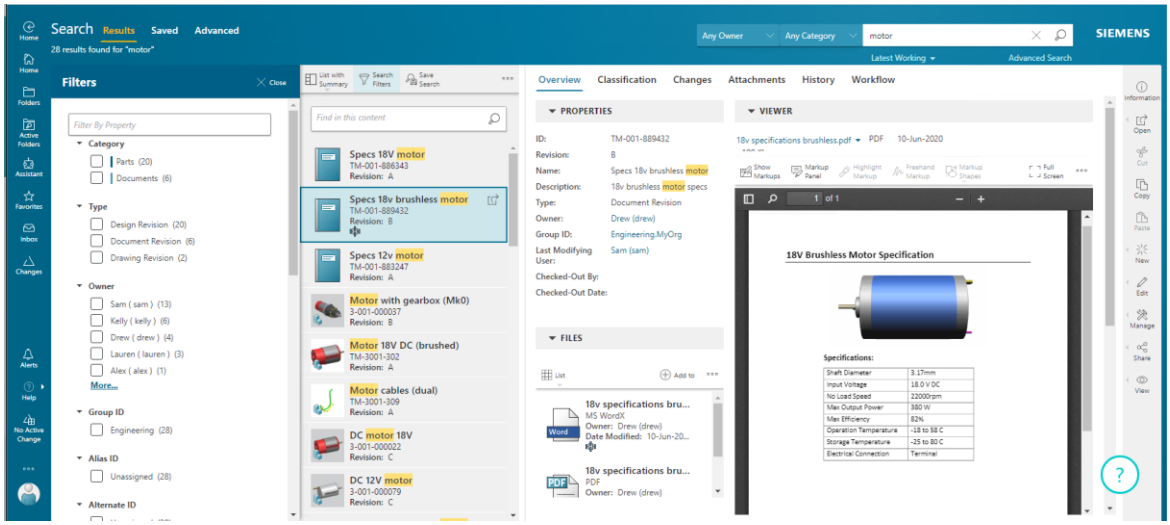
Ominaisuudet

Heti ensimmäisen kerran, kun avaa Teamcenterin, huomaa sovelluksesta sen olevan erittäin Windows-tyyppinen värimaailmaltansa, kuvakkeiltansa ja painikkeiltansa. Pienen läpikäynnin jälkeen mallidatojen kanssa sovelluksesta voi todeta, että se sopii erittäin hyvin isoille projekteille ja yrityksille.

Tiedonhaku

Tiedonhakua varten sovelluksen oikeassa yläkulmassa on hakukenttä, ja joskus hakukenttä on pienenetty vain suurennuslasiksi. Hakukenttään voi kirjoittaa hakusanan, joka etsii vastaavia tietoja esimerkiksi nimikkeiden nimistä, selitteistä ja tyypeistä. Hakukentän vasemmalta puolelta löytyy heti kaksi suodatinta, joista toisella voi suodattaa, kuka on nimikkeen omistaja, ja toisessa voi valita nimikkeen kategorian.

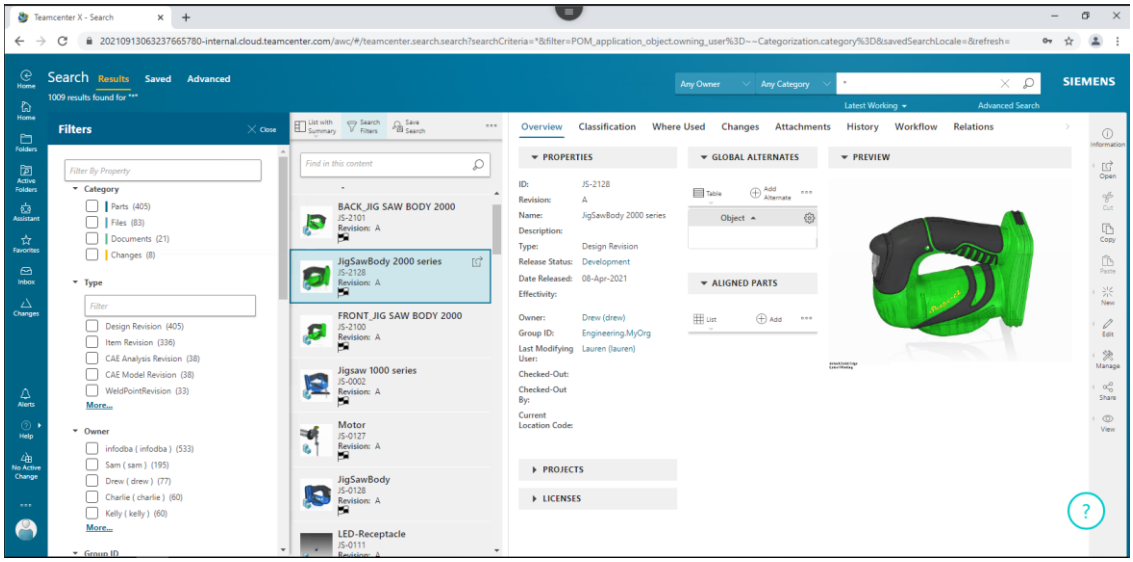
Kuvassa 12 nähdään esimerkkihakutulos hakusanalla ”motor” eli suomeksi moottori. Vasemmalta löytyvät kaikki suodatusvaihtoehdot, joihin kuuluu suodatus seuraavien mukaan: kategoria, tyyppi, omistaja, ryhmätunnus, nimimerkki-ID, vaihtoehtoinen ID, julkaisutila, julkaisupäivämäärä, viimeksi muokannut käyttäjä, viimeksi muokattu ajankohta, projekti, luokitus, IP-luokitus, työn alla, NX-ominaisuudet ja toimipisteen nimi.



Kuva 12. Teamcenter X:n hakutulos sanalla ”motor” (10).

Tuotetieto

Tuotteen tiedon yleisnäkymä kuvassa 13 sisältää ominaisuudet, projektit, lisenssit, varaosat, osat ja tuotteen esikatselun.

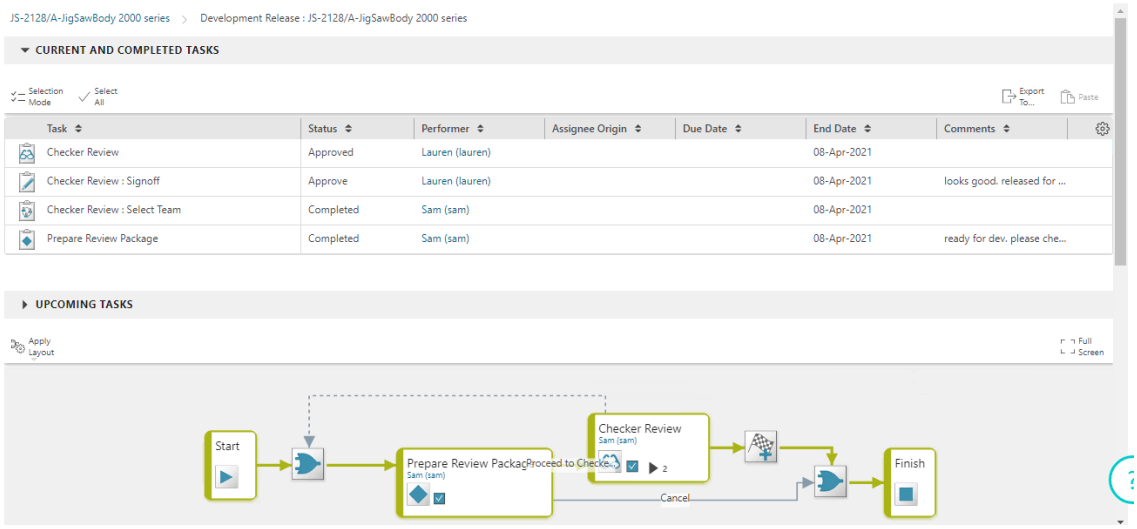


Kuva 13. Tuotetiedon yleisnäkymä (10).

Tuotteelle löytyy myös eri välilehtiä, joissa on seuraavia tietoja:

- Classification- eli luokitus-kohdassa voi antaa ja poistaa tuotteesta eri luokituksia.
- Where used- eli missä käytetty -kohdasta löytyy tietotauluja, missä rakenteissa, konteksteissa ja referensseissä osaa käytetään.
- Changes- eli muutokset-kohdassa voi tarkastella tuotteeseen tehtyjä muutoksia. Tätä varten on neljä eri näkymää, lista, pöytä, vertailu ja kuvat, joissa muutokset esitetään visuaalisesti ja tekstinä.
- Attachments- eli liitteet-välilehdestä löytyvät kaikki tiedostot, jotka ovat tuotteeseen liittyviä ja liitettyjä.
- History- eli historia kohdasta löytyy tuotteen muutos- ja revisiohistoria. Tästä siis näkee kätevästi, mitä muutoksia on tehty.
- Relations eli relaatiot on visuaalinen välilehti, josta löytyvät tuotteeseen liitetyt muut osat ja tieto, mihin tuotteeseen osa on liitetty.
- Participants- eli osallistujat-välilehdeltä löytyvät osan tarkistajaksi merkityt henkilöt sekä osan vastuuhenkilöt.

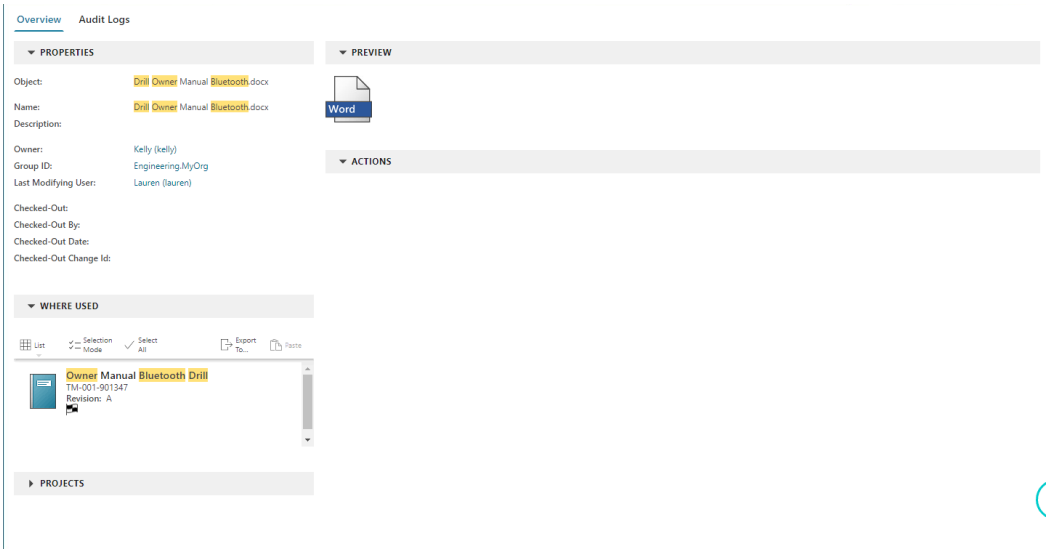
- Audit Logs- eli lokikirjat-välilehdestä löytyy neljä taulua, joissa on lokitiedot työnkulusta, yleisestä, lisenssiviennistä ja rakennemuutoksista.
- Workflow- eli työnkulku-välilehdellä kuvan 14 mukaisesti näkyvät osalle annetut tarkastukset ja muutokset, jotka niistä vastaava henkilö hyväksyy, ja tämä esitetään visuaalisesti myös taulun alla.



Kuva 14. Teamcenterin työnkulku-näkymä (10).

Tiedosto

Tiedostot ovat liitettynä osiin ja tuotteisiin, ja kaikki tiedostot löytyvät etsimällä tiedoston nimellä tai etsimällä kaiken ja suodattamalla pelkät tiedostot hakuun. Kuvassa 15 on poran käyttäjän ohjekirja, jossa on seuraavat ominaisuudet: missä käytetty, projekti, esikatselu ja toiminnat. Tiedostoille on myös tuotetiedon tapaan samanlainen lokikirja.



Kuva 15. Esimerkkitiedosto (10).

Projekti

Projektinhallintatyökalu ei ollut käytettävissä Teamcenterin kokeiluversiossa, ja ominaisuuksien arviointi ja vertailu ei ollut mahdollista saatavilla olleista tiedoista.

Muutostenhallinta

Muutostenhallinta-työkalussa on tarkoituksena muodostaa ja hallita erilaisia muutoksia. Muutokset luovat työnkulkuun yksikölle historian muutoksista ja poikkeuksista. Työkalusta löytyy seuraavat viisi eri vaihtoehtoa muutoksille:

Action item eli mahdollisia kohteen toimenpiteitä varten oleva korostus, esimerkiksi suunnitteluun, yhteensopivuuteen tai korvaamiseen liittyvät asiat. Kohteelle annetaan otsikko, kuvaus ja toimintatyyppi, sen lisäksi voi lisätä kommentin, mihin kohde vaikuttaa, vastuuhenkilön ja alueet. Toimenpiteestä voi avata uuden muutoksen.

Change notice eli muutosilmoituksessa ilmoitetaan tuotteen, osan tai tiedoston muutoksesta. Ilmoitukseen tulee tiedoiksi otsikko ja kuvaus, sen lisäksi

ilmoitukseen voi liittää tiedostoja ja projekteja. Ilmoituksessa on valinta avata uusi muutos.

Change request eli muutospyynnössä pyydetään muutos tietylle tuotteelle, osalle tai tiedostolle. Pyyntöön tulee tiedoiksi otsikko ja kuvaus, sen lisäksi ilmoitukseen voi liittää tiedostoja ja projekteja. Pyynnössä on valinta avata uusi muutos.

Deviation request eli poikkeuspyynnössä pyydetään tekemään poikkeus tiettyyn tuotteeseen, osaan tai tiedostoon. Pyyntöön tulee otsikko, selite ja poikkeustyyppi, sen lisäksi pyynnössä voi liittää tiedostoja ja projekteja. Pyynnössä on valinta avata uusi muutos.

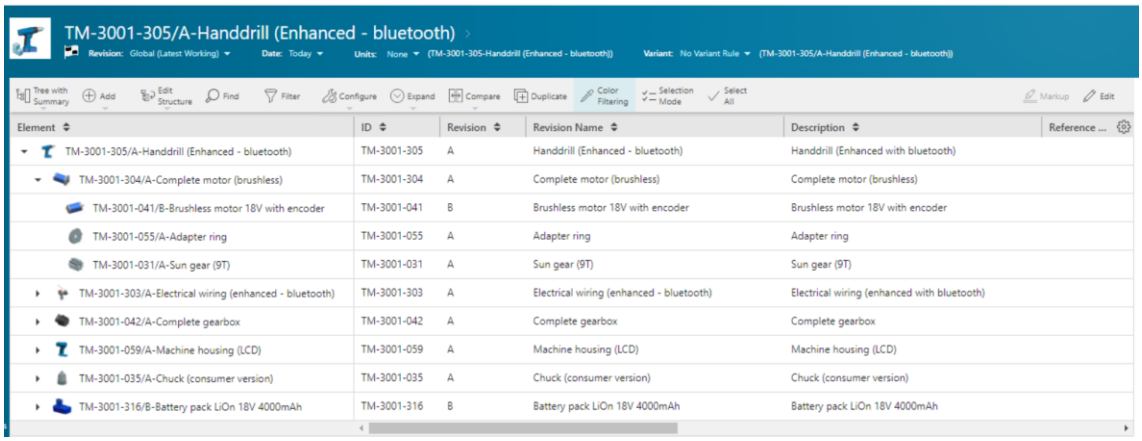
Problem report eli ongelmaraportissa raportoidaan tuotteeseen, osaan tai tiedostoon liittyvästä ongelmasta. Ongelman raportoimista varten tarvitaan otsikko ja selite, sen lisäksi raporttiin voi liittää tiedostoja ja projekteja.

Postilaatikko

Teamcenter X -sovelluksessa on postilaatikko, josta löytyy viisi eri kohtaa: omat, tiimin, seuratut, kaikki ja sijaistehtävät. Työntekijälle osoitetut tehtävät löytyvät omien tehtävien alta. Niihin kuuluvat erilaisten osien valmistukset, muutospyynnöt, muutosilmoitukset, ongelmaraportit ja poikkeamapyynnöt. Tätä kautta työntekijä pääsee helposti hyväksymään hänelle osoitettuja tehtäviä ja tarkastelemaan muilla olevia tehtäviä, sekä hänen itse raportoimiensa asioiden tilannetta.

BOM

Tuoterakenteet luodaan Siemensin NX-suunnitteluohjelmassa ja tuodaan Teamcenteriin. Tuoterakenteesta löytyy osat tuotteen valmistamista varten. Ohjelmistosta voi muokata tuoterakennetta muun muassa poistamalla ja lisäämällä osia, jotka ovat järjestelmässä. Kuvassa 16 on esimerkki tuoterakenteesta.



Kuva 16. Teamcenter X -ohjelmiston käsiporan osista muodostuva tuoterakenne (10).

Datan tuonti

Teamcenter käyttää suunnitteludatan tuontia varten omaa suunnittelujärjestelmäänsä Siemens NX. Teamcenterin saa integroitua suunnitteluohjelmaan ja synkronoitua tiedostot järjestelmien välillä sekä tallennettua päivitetyt versiot suoraan Teamcenterin rakenteisiin.

Datan vienti

Datan vientiä varten sovelluksesta löytyy eri tauluista ja tiedostoista vientityökalut, joista pystyy tallentamaan tiedoston tai tauluista taulun tiedot saatavilla oleviin muotoihin. Saatavilla olevia tiedostotyyppejä ovat muun muassa Excel, Word, PDF ja CSV.

3.4 Yhteenveto

Ohjelmistojen tutkiminen onnistui hyvin, ja jokaisesta sovelluksesta saatiin kattavasti tietoa. Hankaluuksina olivat kaupalliset yrityssovellukset, joiden pääsy oli osittain rajattua. Siitä huolimatta saatiin erinomainen käsitys siitä, mitä varten ja kenelle sovellukset on suunnattu.

Laajoja tuotantoympäristöjä ja prosesseja varten Roiman Aton ja Siemensin Teamcenter X ovat parempi valinta verrattuna Odoon PLM:iin. Aton ja Teamcenter X tarjoavat laajempia ja monipuolisempia ominaisuuksia valmiina sekä mahdollisuutena asiakaskohtaisen jatkokehityksen. Aton-järjestelmässä on selvästi parhaat mahdollisuudet eri sovelluksien yhteensopivuuden ja integroinnin kanssa. Teamcenter X on hyvä yritykselle, jolla on Siemensin muut ohjelmistot käytössä tai joka on valmis ottamaan ne käyttöön. Odoon PLM-järjestelmä on vuorostaan selkeästi edullisempi ja helpompi ratkaisu pienempiin ympäristöihin, joissa prosessit voidaan suorittaa manuaalisemmin ja vähemmällä suunnittelulla. Odoo tarjoaa myös suuren määrän yrityksen eri toimintoihin liittyviä työkaluja, jotka saadaan suoraan integroitua yhteen järjestelmään lisähinnalla. Ohjelmistojen ominaisuudet ovat erilaisia ja ne soveltuvat erilaisiin projekteihin, joten niiden sopivuus eri toimijoille riippuu tilanteesta ja tarpeista.

Ohjelmistoja tutkittaessa tunnistettiin neljä eri kehityskohdetta, joita sovelletaan suunniteltaessa jatkokehitystä. Ensimmäisenä tulivat esille laajemmat hakuominaisuudet, joita sovelluksista löytyi. Sen lisäksi muutosten tekeminen ja hallitseminen on olennainen asia, joista löytyi kolme eri kokonaisuutta jatkokehitykselle. Nämä ovat itse muutostenhallinta sekä tehtyjen muutoksien jäljitys lokikirjan ja revisioiden avulla.

4 Jatkokehityssovelluksen teknologiat

Tässä luvussa käydään läpi työkalut ja teknologiat, jotka ovat käytössä Paiston PLM-järjestelmässä. Järjestelmän selainpuoli on kehitetty ReactJS:n päälle ja palvelinpuoli taas on Node.js-pohjaisen NestJS-ohjelmistokehyksen päälle tehty käyttäen kielenä TypeScriptiä. Sovelluksen tietokantana toimii MySQL. Swagger on rajapintakuvaukseen ja suunnitteluun tarkoitettu ohjelmisto, jota käytetään projektin rajapinnan dokumentointiin ja jatkokehitykseen.

4.1 ReactJS-kirjasto

ReactJS on Facebookin perustama ja ylläpitämä JavaScript-kirjasto, joka julkaistiin toukokuussa 2013. ReactJS tunnetaan myös nimillä React.js tai React. Nopeasti julkaisun jälkeen React nousi yhdeksi suosituimmaksi kirjastoksi, jolla voi tehdä ”single-page” eli yhden sivun sovelluksia ja mobiilisovelluksia. Kirjaston tarkoitus oli ratkoa skaalautuvien datalähtöisten sovellusten ongelmia. (8.) React on rajoitettu vain tilan hallintaan ja näitten tilojen lataamiseen sivulle, joten se yleensä vaatii lisäkirjastoja tukemaan sitä, kuten React Router, jolla hoidetaan sovelluksen navigaatiota (6).

ReactJS:ää käytetään komponenttipohjaiseen rakenteen tekemiseen, joka mahdollistaa geneerisen komponenttien teon ja näiden komponenttien uudelleen käyttämisen. Komponenttipohjainen rakenne mahdollistaa yksittäisten komponenttien päivityksen web-sivulla, mikä keventää sivun rasitusta. ReactJS-sovellus käyttää JSX- eli JavaScript Syntax Extension -syntaksia, joka on JavaScript-laajennos. Syntaksin avulla kirjoitetaan Reactin komponentit, ja syntaksi muistuttaa todella paljon html:ää. (8.)

4.2 TypeScript-kirjasto

TypeScript on Microsoftin kehittämä ja ylläpitämä avoimen lähdekoodin JavaScript-kirjasto. Sen ensimmäinen versio julkaistiin vuonna 2012. TypeScript on ohjelmointikieli, joka rakentuu JavaScriptin varaan. Se ei ole oma kielensä vaan lisää uusia ominaisuuksia JavaScriptiin. Sen päällimmäinen vahvuus on tyypitys, josta kielen nimikin tulee. Ohjelmointikielet voi jakaa kahteen luokkaan tyypityksien mukaan, joko heikosti tai vahvasti tyypitetyt. JavaScript on heikosti tyypitetty, eli yksittäisten muuttujien tyyppejä ei tarvitse erikseen määritellä. Vahvasti tyypitetyissä kielissä, kuten Javassa, jokaisen muuttujan tyyppi on aina määriteltävä sitä luodessa. (7.)

TypeScript lisää JavaScriptiin vahvan tyypityksen tuen ja antaa mahdollisuuden kielessä asettaa muuttujien tyypit sekä ominaisuuksia kuten rajapintaluokat.

Tämä tapahtuu ohjelmointiympäristössä, ja julkaisun yhteydessä kaikki käännetään normaaliksi JavaScriptiksi. Tämä helpottaa käyttöönottoa, ja JavaScript on suoraan käypää TypeScriptiä. (7.)

Myös monet JavaScript-sovelluskehykset käyttävät nykyään TypeScriptiä, kuten Angular 2, Aurelia, Dojo, Ember, Ionic ja NativeScript. Sovelluskehykset tarjoavat kehittäjille rungon, jonka päälle rakentaa ohjelmistonsa. (7.)

4.3 NestJS-kirjasto

NestJS on progressiivinen Node.js-ohjelmistokehys, jonka tehtävänä on rakentaa tehokkaita ja skaalautuvia palvelinpuolen Node.js-sovelluksia. Ohjelmistokehys tukee JavaScript- ja TypeScript-ohjelmointikieliä. NestJS yhdistää funktionaalisen, olio- ja funktionaalisen reaktiivisen ohjelmoinnin elementtejä. (3.)

Seuraavat käsitteet ovat perusasiat rakennettaessa NestJS-sovellusta:

NestCLI on komentorivin käyttöliittymä, joka ei ainoastaan nopeuta web-pohjaisen sovelluksen kehitystä vaan myös automatisoi projektin aloituksen. Vain kahdella komennolla projektikansio luodaan, Node-moduulit ja muutama vakiotiedosto asennetaan. Komentoja varten tarvitsee asentaa npm-paketinhallinta-ohjelmisto. Komennot, joita tarvitaan NestCLIn käytön aloittamista varten ovat ”npm install -g @nestjs/cli”, joka asentaa komentorivin käyttöliittymän käyttöön, ja ”nest new firstProject”, joka luo uuden projektin. ”npm run start:dev”-komento suorittaa sovelluksen. (4.)

Controller eli ohjaimet hallitsevat pyyntöjä ja palauttavat datan asiakasohjelmaan. Ohjaimen tehtävä on vastaanottaa tietty pyyntö sovellukselle. Reititys määrittää, mikä ohjain vastaanottaa minkäkin pyynnön. Välillä ohjaimella on useampi reitti ja voi suorittaa eri tehtäviä. Esimerkkikoodissa 1 näkyy ohjain, johon tuleva reitin ”cats”-pyynnöt. (3.) Luodakseen perusohjaimen Nest käyttää apuna luokkia ja sisustuksia. Sisustukset lisäävät luokkiin tarvittavan

metadatan, jonka avulla Nest rakentaa reitityksen. HTTP-metodien päätepisteen määrittämiseen ovat käytettävissä @Get(), @Post(), @Put(), @Delete() jne. (3.)

```
import { Controller, Get } from '@nestjs/common';

@Controller('cats')
export class CatsController {
  @Get()
  findAll(): string {
    return 'This action returns all cats';
  }
}
```

Esimerkkikoodi 1. Nest ohjaimesta (3).

Providers eli palveluntarjoajat ovat Nestin peruskäsite, ja monia perusluokkia voidaan käyttää palveluntarjoajina. Keskeisin idea on, että ne voidaan lisätä riippuvaisuuksina, mikä tarkoittaa, että objektit voivat luoda erilaisia relaatioita keskenään. Esimerkkikoodissa 2 näkyy palveluntarjoaja, joka pitää sisällään ”cats”-käyttöliittymän tietoja. (3.)

```
import { Injectable } from '@nestjs/common';
import { Cat } from './interfaces/cat.interface';

@Injectable()
export class CatsService {
  private readonly cats: Cat[] = [];

  create(cat: Cat) {
    this.cats.push(cat);
  }

  findAll(): Cat[] {
    return this.cats;
  }
}
```

Esimerkkikoodi 2. Nest palveluntarjoajasta (3).

Service eli palvelu on vastuussa tietokannan hallinnasta, ja sen toiminta määritellään kontrollerin avulla. Palvelu voidaan myös määritellä palveluntarjoajaksi. (3.)

Module eli moduuli on luokka, jolla on @Module()-sisustus. Moduuli on luokka, joka yhdistää palveluita, ohjaimia jne., kuten koodiesimerkissä 3 näkyy. (3.)

```
import { Module } from '@nestjs/common';
import { CatsController } from './cats.controller';
import { CatsService } from './cats.service';

@Module({
  controllers: [CatsController],
  providers: [CatsService],
})
export class CatsModule {}
```

Esimerkkikoodi 3. Nest moduulista (3).

4.4 Swagger-ohjelmisto

Swagger on avoimen lähdekoodin ohjelmisto, jota käytetään rajapinnan kuvaamiseen JSON-kielellä. Ohjelmistoa käytetään yhdessä sen muiden ohjelmistotyökalujen kanssa rajapintojen suunnitteluun, rakentamiseen, dokumentoitiin ja käyttöön. Ohjelmiston avulla koodista voidaan luoda automaattisia dokumentaatioita, jotka SwaggerUI-työkalulla esitetään visuaalisesti ja interaktiivisesti. (12.)

5 Rajapinnan dokumentaatio ja käyttöohje

Tässä luvussa luodaan rajapintadokumentaatio OpenAPI eli määritys koneellisesti luettavalle rajapintatiedostolle ja Swagger-työkalujen avulla. Dokumentaation luomista varten tehdyt toimenpiteet käydään läpi ja selitetään. Luvussa suoritetaan myös käyttöohjeen ohjeiden tekeminen, jonka avulla voidaan luoda sovelluksen käyttöohje helposti ohjeiden mukaisesti.

5.1 Rajapinnan dokumentaatio

Nestin päälle rakennetun rajapinnan dokumentointia varten löytyy riippuvaisuus Swaggerille ja SwaggerUI:lle, jotka ovat Nestin dokumentaatiossa nimellä OpenAPI (12). Esimerkkikoodissa 4 asennetaan riippuvaisuudet.

```
npm install --save @nestjs/swagger swagger-ui-express
```

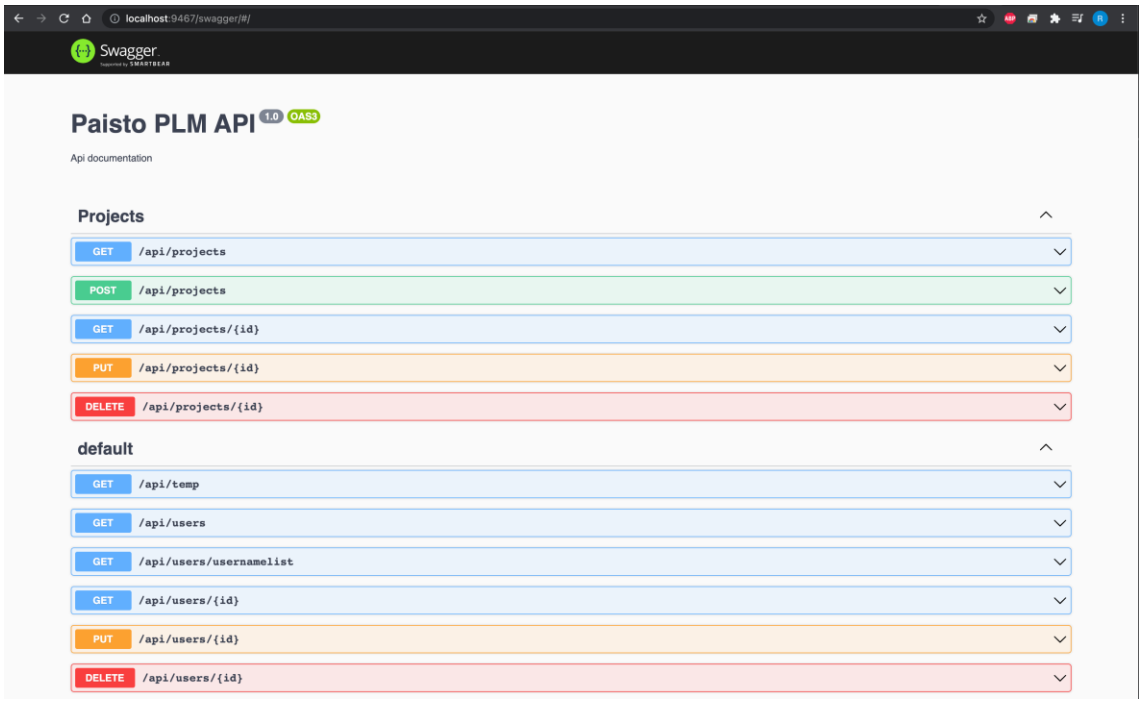
Esimerkkikoodi 4. Swagger-riippuvaisuuksien asennus.

Kun riippuvaisuudet on asennettu sovelluksen Main.ts-tiedostoon tuodaan ”SwaggerModule” ja ”DocumentBuilder” esimerkkikoodin 5 ensimmäisen rivin mukaisesta. Tämän jälkeen luodaan Swagger-konfiguraatio, johon voi määritellä otsikon, selitteen, version ja tagit ja sen jälkeen konfiguraatio rakennetaan. Kun konfiguraatio on valmis, luodaan dokumentti metodilla, jonka parametreiksi tuleva sovelluksen moduuli ja konfiguraatio. Viimeisenä Swaggerin moduuli luodaan metodilla, ja ensimmäiseksi parametriksi tulee reitin polku, josta dokumentaation löytää sivulta, sitten sovellus ja viimeisenä dokumentti.

```
import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';
async function bootstrap() {


const app = await NestFactory.create(AppModule);

// Setup swagger document that can be seen at localhost:${PORT}/swag-
ger
const config = new DocumentBuilder()
.setTitle('Paisto PLM API')
.setDescription('Api documentation')
.setVersion('1.0')
.addTag('Projects')
.build();
const document = SwaggerModule.createDocument(app, config);
SwaggerModule.setup('swagger', app, document);

app.use(compression());

const PORT = process.env.PORT || 8080;
await app.listen(PORT, () => {
console.log(`Server is listening on port ${PORT}`);
});
}
bootstrap();
```

Esimerkkikoodi 5. Sovelluksen Main.ts-tiedoston Swagger luominen.

Dokumentaation luonnin jälkeen kuvan 20 näköinen dokumentti löytyy selaimesta menemällä sivun osoitteeseen ja lisäämällä URL:n perään ”/swagger”, joka on määritelty reitti dokumentille. Sivulta löytyy ensin otsikko ja selite, tämän alla ovat ensimmäisenä tagatut reititykset ja sitten loput tulevat oletuksen alle, jos sovelluksessa on tagaamattomia reittejä.



Kuva 17. Swagger-dokumenttisivu.

Kun dokumenttisivu on luotu ja toimii, lisätään sovelluksen ohjaimiin tagit. Tagien avulla voidaan helposti merkitä, minkä reitin ohjaimen kyselyt löytyvät sen alta. Esimerkkikoodissa 6 ensin tuodaan ”ApiTags” ja sitten tagi liitetään ohjaimeen ja sille annetaan haluttu nimi, tässä tapauksessa ”Projects”. Lopuksi tämä toistetaan koko ohjelmistorakenteen jokaiselle ohjaimelle antaen niille yksilölliset tagit.

```
import { ApiTags } from '@nestjs/swagger';

@ApiTags('Projects')
@Controller('/api/projects')
export class ProjectController {
    constructor(private readonly service: ProjectService){}

    //@UseInterceptors(CacheInterceptor)
    @Get()
    async findAll():Promise<Project[]>{
        return this.service.findAll();
    }

    @Get(':id')
    async findOne(@Param() params):Promise<Project>{
        return this.service.findOne(params.id);
    }…
```

Esimerkkikoodi 6. Sovelluksen tagin lisääminen projektiohjaimeen.

Swagger-dokumentaatiossa on myös sovelluksen DTO (Data transfer object) eli suomeksi tiedonsiirto-objekti. Nämä objektit näkyvät valmiiksi sivulla, mutta niistä puuttuu kaikki sisältö. Sisältöä varten luokan ominaisuuksiin täytyy lisätä ”ApiProperty”, tai jos ominaisuus on valinnainen, niin ”ApiPropertyOptional. Kuten esimerkkikoodissa 7 näkyy, lisätään ”CreateProjectDto” ja ”UpdateProjectDto” eli projektin lisäys- ja päivitysobjekteihin lisätään, ovatko ne pakollisia vai valinnaisia ominaisuuksia.

```
import { ApiProperty, ApiPropertyOptional } from '@nestjs/swagger';

export class CreateProjectDto {
    @ApiPropertyOptional()
    readonly name?: string;
    @ApiPropertyOptional()
    readonly description?: string;
    @ApiPropertyOptional()
    readonly projectState?: ProjectState;
    @ApiPropertyOptional()
    readonly customer?: Customer;
    @ApiPropertyOptional()
    readonly productionProject?: ProductionProject;
    @ApiPropertyOptional()
    readonly salesProject?: SalesProject;
    @ApiPropertyOptional()
    readonly engineeringProject?: EngineeringProject;
}

export class UpdateProjectDto {
    @ApiProperty()
    readonly id: number;
    @ApiPropertyOptional()
    readonly name?: string;
    @ApiPropertyOptional()
    readonly description?: string;
    @ApiPropertyOptional()
    readonly projectState?: ProjectState;
    @ApiPropertyOptional()
    readonly customer?: Customer;
    @ApiPropertyOptional()
    readonly productionProject?: ProductionProject;
    @ApiPropertyOptional()
    readonly salesProject?: SalesProject;
    @ApiPropertyOptional()
    readonly engineeringProject?: EngineeringProject;
}
```

Esimerkkikoodi 7. Ominaisuuden valinnaisen ja pakollisen kentän lisäys.

Ominaisuudet näkyvät dokumentaatiossa kuvan 21 mukaisesti. Jos ominaisuus on pakollinen, nimen perään tulee asteriski.

Kuva 18. Swagger-dokumentin esimerkki-DTO.

5.2 Käyttöohjeen luomisen ohjeet

Monissa yrityksissä, joissa kehitetään sovelluksia ja varsinkin ketterän kehityksen tapaan, monesti käyttöohjeen kirjoitus jää työlistalla alemmalle prioriteetille. Näin on myös sattunut käymään insinöörityössä esitellyn sovelluksen kanssa, ja sen takia työssä tutkittiin, miten käyttäjäystävällinen käyttöohje pitäisi tehdä. Käyttöohjeen luomisen ohjeita varten löytyi Heidi Martikaisen pro gradu -tutkielma, jonka tiivistettyjä tietoja käytettiin apuna ohjeen luomisessa (14).

Käyttöohjeen tarkoituksena on auttaa käyttäjää löytämään tarvittavat tiedot lukutottumuksista riippumatta ja oppia pidemmälle aikavälillä käyttämään käyttöliittymää paremmin kuin ilman käyttöohjetta. Käyttöohjeen kirjoittajana ja päivittäjänä olisi hyvä toimia henkilön, joka ei ole järjestelmäkehittäjä. Tämä pitää

huolen, ettei käyttöohjetta kirjoiteta kehittäjän näkökulmasta tai liian teknisellä ajattelutavalla. Saman henkilön toimiessa kirjoittajana hänen kokemuksensa kertyy myös kirjoittajana. Käyttöohjeen pitää olla riittävän yksinkertainen niin että sen avulla jokainen käyttäjä pystyy suorittamaan tarvitsemansa toiminnot. (14.)

Suunnitellessa ja toteuttaessa käyttöohjetta olisi hyvä pitää mielessä ja hyödyntää seuraavia periaatteita. Yksinkertaisuus pitää tekstin lyhyenä ja nopeana luettavana ja tuo vain oleelliset asiat esille. Kirjoitustyylin tulee olla selkeä ja asiallinen ja oikeinkirjoituksen kunnossa. Tekstissä on hyvä välttää lyhenteitä, jotta kaikki tietävät, mistä puhutaan ilman lyhenteen tarkoituksen etsimistä. Imperatiivilauseet ja käyttäjän suora puhuttelu pitävät lauseet ohjemuodossa. Kun otsikkotaso on osuva ja kekseliäs, käyttäjän on helppo löytää tarvitsemansa. Ohjeen tulee olla helposti selattava ja tiedon havainnollisessa muodossa. Käyttöohjeen tulee olla yhtenäinen kokonaisuudeltansa sanavalinnoissa, kirjoitusasussa, rakenteessa ja ulkoasussa. Käyttöohjeesta kannattaa unohtaa tekninen ammattisanasto, ja käyttöliittymän elementit tulee nimetä. Ohjeesta olisi myös hyvä löytyä johdanto, hakemisto, sisällysluettelo ja sanasto. Virheiden estämistä ja käsittelyä ei tämän projektin osalta kannata suorittaa käyttöohjeen puolella, vaan ne tulisi selkeästi ilmoittaa suoraan käyttöliittymässä niiden tapahtuessa. (14.)

6 Jatkokehityksen suunnitelma

Tässä luvussa pohditaan sovelluksen jatkokehitykseen liittyviä asioita, kuten tarpeellisia ominaisuuksia ja käytettävyyttä. Jatkokehityskohteiden määrittelyssä ja toteutuksessa hyödynnetään lähteissä ja sovellusvertailuissa havaittuja asioita. Osiossa ei tehdä teknisiä ratkaisuja kehitykseen liittyen, vaan suunnitellaan ratkaisut ja pohjustetaan ne, jotta jatkokehityksen suorittaminen on selkeää ja valmiiksi suunniteltua.

Insinöörityön tietoperustan, tutkittujen sovelluksien ja kehitystiimin kanssa käytyjen jatkokehityspalaverien kautta saatiin viisi eri kehityskohdetta. Hakuominaisuudet ovat tällä hetkellä hyvin perustasolla, mikä tarkoittaa, että kohteita voi

hakea vain niitten otsikolla ja tutkituissa sovelluksissa nähtyjä suodatusominaisuuksia ei ole. Ominaisuudesta on saatu selkeitä kehityspyyntöjä asiakkaalta, ja asiaa on myös käyty kehitystiimin palavereissa läpi tarpeellisena ja tärkeänä. Revisiointi ja lokikirja on kokonaisuutena syntynyt kehitystiimin kanssa miettiessä, miten tallennetaan tiedot erilaisiin muutoksiin liittyen, osa niistä on pienempiä ja osa isompia, jolloin muutoksia pitäisi helpommin pystyä vertailemaan. Tähän samaan osittain liittyy muutostenhallinta, joka on myös ollut kehitystoiveena eri osapuolilta. Muutostenhallinta keskittyy enemmän ongelman raportointiin, ilmoittamiseen ja havaitsemiseen, jolloin projektissa työskentelevät eri tiimit saavat tiedon virheellisestä asiasta omalla osa-alueellaan. Muutostenhallinnasta löytyy muutoksen alkuperä ja prosessin kulku sekä myös tarkemmat einiin-tekniset tiedot. Kehityksessä työskennellessä eri henkilöt ja vakioimattomat ulkoasutyylit ovat johtaneet siihen, että ulkoasu on hajanainen ja vaatii erilaisia päivityksiä.

6.1 Hakuominaisuudet

Hakuominaisuuksien kehitys on tarpeellista, jotta tiedon löytäminen ja hyödyntäminen on nopeaa ja vaivatonta. Hakusanalla sovelluksen pitää löytää hakutuloksia tuoterivin kaikilta kentiltä ja liitetyiltä tiedostoilta, jotta voidaan yhdistää hakusana myös otsikon ulkopuolella esiintyviin kohteisiin. Hakutuloksia halutaan myös suodattaa niissä olevien ominaisuuksien mukaan. Suodatus mahdollistaa erikseen yksilöllisten asioiden hakemisen. Hakuominaisuus pitäisi myös saattaa saataville projektin ulkopuolelle, jotta tietoa voidaan etsiä kaikista projekteista. Tämän avulla voidaan esimerkiksi tarkistaa, mistä projekteista löytyy tietty tuote tai tiedosto.

6.2 Revisio ja lokikirja

Revisiointi on erinomainen ominaisuus muutettaessa kohteen tietoja niin, että jotain merkittävää muuttuu. Jos revisiointia ei ole olemassa, vanha tieto katoaa ja muutosten jäljitys muuttuu vaikeaksi. Projektissa tai sen osassa voi tulla laajamuotoisia muutoksia, jotka on syytä tallentaa myöhempää varten. Tuotteen

huoltamista varten voi olla olennaista tietää, minkä revisioversion mukaan se on rakennettu vaihdettavia osia varten.

Pienempiä muutoksia varten lokikirja on parempi vaihtoehto, koska esimerkiksi selitteen pieni korjaus ei luo revisiota ja revisioiden määrä ei kasva suureksi pitkällä aikavälillä. Lokikirjaa varten tallennetaan tiedot siitä, kuka, milloin ja mitä muutti. Näin on helppo luoda loki, josta voidaan jälkikäteen katsoa pieniä muutoksia, esimerkiksi vahingossa poistetun selitteen teksti ja korjata tämä. Lokikirjaan voi myös lisätä hakuominaisuuden tai suodatuksen tietyn tietoruudun muutoksille.

6.3 Käyttöliittymän ulkoasu

Sovelluksen käyttöliittymän ulkoasu vaatii päivitystä, koska nykyisessä versiossa on vaihtelevia ulkoasuelementtejä ja rajattu responsiivisuus eri näyttöko'oilla. Ulkoasun suunnittelua varten löytyy paljon eri ohjelmia, kuten kuvankäsittelijöiden suosimat Adoben Photoshop tai XD. Ohjelmat sopivat mainiosti käyttäjälle, joka vain suunnittelee visuaalisen osion käyttöliittymästä. Suunnittelua ja toteutusta varten päädyttiin kuitenkin Bootstrap Studioon, joka mahdollistaa suunnittelun ja toteutuksen samanaikaisesti. Sovelluksessa voi käyttää HTML- ja Bootstrap-komponentteja suunnitteluun. Kun käyttöliittymä on suunniteltu ohjelmistossa, sen koodit ja tyylisivut saa vietyä suoraan projektiin. Sovellus tukee myös ulkopuolisia tyylisivukirjastoja, joten suunnittelussa voi myös käyttää muita tyylikirjastoja.

Lisäksi oli syytä tutkia mahdollisia React-komponentteja olemassa olevien tyylielementtien osalta. Bootstrapin korvaavaksi ensisijaiseksi tyyli- ja komponenttikirjastoksi löytyi SAP:n tekemä Fundamental library (17). Kirjastosta löytyy tyylisivu ja React-komponentteja, joita hyödynnetään projektissa. Ensisijaisesti luodaan komponenttitiedostoja, joissa on HTML-elementit Fundamental-kirjaston tyylisivun tyyleillä. Tämä tapa mahdollistaa elementtien yhtenäisen muokkauksen koko projektiin sen omasta tiedosta. Reactin komponentteja käytetään

silloin, kun ne tuovat erilaisia valmiita ominaisuuksia ja helpottavat työskentelyä nopeuttamalla tai vähentämällä koodia.

6.4 Muutostenhallinta

Muutostenhallinta on keskeinen asia tuotteen elinkaaren hallinnassa: se kertoo kaiken, mitä kohteeseen tehdyn muutoksen prosessissa on tapahtunut. Muutoksenhallinta on prosessi eikä pelkästään muutos. Muutosprosessi alkaa jostain viasta tai tarpeesta muuttaa jotain. Muutoskohteen voi luoda kuka tahansa käyttäjä käyttöoikeuksien puitteissa, ja siitä kohde etenee vastaavalle työryhmälle. Työryhmässä muutos suunnitellaan, hyväksytetään ja viimeisenä muutos suoritetaan. Muutoksesta olisi myös tarpeen jäädä lokitieto ja tarvittaessa revisio. Muutoshallinnassa olisi myös hyvä olla ominaisuus, joka suorittaa tekniset muutokset tuotetietoihin suoraan.

7 Yhteenveto

Insinöörityössä perehdyttiin tuotteen elinkaarenhallinnan ja tuotehallinnan sovelluksiin, niiden olennaisiin ominaisuuksiin ja käyttötarkoitukseen sekä markkinoilta löytyviin tuotteen elinkaarenhallintasovelluksiin. Tavoitteena oli saada laaja ja kattava käsitys aiheesta sekä markkinoilla olevista sovelluksista, niiden yleisistä ominaisuuksista ja toteutustavoista, luoda rajapintadokumentaatio Swagger-työkalun avulla ja pohjustaa ohjeet käyttöohjeen luomista varten sekä suunnitella ja pohjustaa jatkokehitystarpeet ja -kohteet, niin että tekninen toteutus voidaan suorittaa suunnitelman perusteella.

Tuloksena saatiin perusteellinen tietopohja tuotteen elinkaarenhallinasta ja tuotetiedonhallinnasta. Markkinoilta löytyvien sovelluksien tutkimisen avulla pystyttiin tunnistamaan toistuvia hyödyttäviä ja tarpeellisia ominaisuuksia, erilaisia lähestymistapoja ja se, millaisiin ympäristöihin sovellukset soveltuvat.

Sovelluksen jatkokehitystä varten työssä käytiin läpi projektissa käytettävät eri teknologiat ja toteutettiin rajapintadokumentaatio. Käyttöohjeen puuttumisen

vuoksi työssä oli myös syytä tutustua olennaisiin asioihin ja käytäntöihin, jotta käyttöohjeen luonti sujuu käyttäjälle ystävälliseen tapaan ja täyttää helppolukuisuuden ja tiedon vaivattoman löytämisen perusteet.

Jatkokehityksen suunnittelemista varten yhdessä kehitystiimin kanssa päädyttiin yhteensä viiteen eri kohteeseen, jotka tunnistettiin lähteiden ja sovellusten tutkimisen perusteella. Suunnitelmassa määriteltiin ominaisuuksien käyttötarkoitukset ja tarpeet. Suunnitelman avulla ominaisuuksista on kehitystyötä aloittaessa helppoa tehdä tekniset määritelmät ja vaatimukset toteutusta varten. Käyttöliittymän päivitystä varten tutustuttiin erilaisiin tyylikirjastoihin ja React-komponenttikirjastoihin, joista päädyttiin Fundamental-kirjastoon. Käyttöliittymää varten myös määriteltiin toimintatapoja tyyliä varten, jotta ulkonäöstä saadaan yhtenäinen.

Työn tuloksena selvisi paljon muiden yritysten ohjelmistoista ja lähestymistavoista. Tämän perusteella on tärkeätä säilyttää jatkokehityksen painopiste perusominaisuuksissa ja visuaalisessa esittämisessä sekä ketterän kehityksen työmallissa. Kehityksessä huomioitavia asioita ovat myös rajapintakehitys ja tiedon sujuva syöttäminen. Vertailtujen ohjelmistojen perusteella tuotetiedon hallinnan ja tuotteen elinkaaren hallinnan ominaisuuksissa on päällekkäisyyksiä sekä erilaisia lähestymistapoja tiedon ja visuaalisuuden suhteen. Laajempi katsaus eri ohjelmistoihin voisi tuoda uusia ominaisuuksia ja lähestymistapoja. Jatkotutkimuskohteiksi tunnistettiin perehtyminen datan tuonnin rakenteisiin ja suunnittelutapoihin ja lisäksi yritysten eri suunnittelu- ja tuotantoympäristöjen tutkiminen ja niiden perusteella soveltuvien lähestymistapojen tunnistus tuotetiedon hallintaan. Arviointi yrityksille saavutettavasta lisäarvosta ohjelmiston käyttöönotolla on myös mahdollinen kohde.

Lähteet

1 Sääksvuori, Antti & Immonen, Anselmi. 2002. Product Lifecycle Management. Springer.

2 Sääksvuori, Antti & Immonen, Anselmi. Tuotetiedonhallinta – PDM. Springer.

3 NestJs Documentation. Verkkoaineisto. Mysliwiec, Kamil. <https://docs.nestjs.com/>. Luettu 18.8.2021.

4 Bouchefra, Ahmed. Introduction to Nest.js for Angular Developers. 2019. Verkkoaineisto. <https://www.sitepoint.com/introduction-to-nest-js-for-angular-developers/>. Luettu 25.8.2021.

5 Mazaika, Ken. ReactJS 101 – Everything you need to know. Verkkoaineisto. <http://blog.thefirehoseproject.com/posts/reactjs-101/>. Luettu 25.8.2021.

6 React (JavaScript library). Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/React_(JavaScript_library)/>. Luettu 5.9.2021.

7 Tarviainen, Jani. 2016. Mikä on TypeScript? Verkkoaineisto. <https://symfony.fi/artikkeli/mika-on-typescript>. Luettu 5.9.2021.

8 Banks, Alex & Porcello, Eve. 2017. Learning React. O'Reilly Media.

9 Odoo. Verkkoaineisto. Wikipedia. </https://fi.wikipedia.org/wiki/Odoo>. Luettu 28.8.2021.

10 Teamcenter X: Instant-on Cloud PLM. Verkkoaineisto. <https://www.plm.automation.siemens.com/global/en/products/teamcenter/cloud-plm.html/>. Luettu 2.9.2021.

11 Mysliwiec, Kamil. NestJs OpenApi documentation. 2021. Verkkoaineisto. <https://docs.nestjs.com/openapi/introduction/>. Luettu 14.9.2021.

12 Swagger documentation. Verkkoaineisto. <https://swagger.io/>. Luettu 10.9.2021.

13 Odoo editions. Verkkoaineisto. <https://www.odoo.com/page/editions/>. Luettu 20.9.2021.

14 Martikainen, Heidi. 2020. Käyttöohjeiden käytettävyys: Suunnitteluperiaatteiden kehittäminen sosiaali- ja terveydenhuollon asiakastietoja käsittelevien järjestelmien käyttöohjeita varten. Pro gradu -tutkielma. Tampereen yliopisto. Trepo-tietokanta.

15 Tuotteen elinkaaren hallinta. Verkkoaineisto. <https://www.roimaint.fi/tuotehallinta/tuotteen-elinkaaren-haltuun/>. Luettu 28.9.2021.

16 Kuuri, Ismo. 2021. Aton 7. Verkkoaineisto. <https://www.roimaint.fi/aton-7-uuden-tuoteobjektin-hyodyntaminen-webinaari/>. Luettu 10.9.2021.

17 Sap Fundamental Library. Verkkoaineisto. <https://sap.github.io/fundamental/>. Luettu 4.10.2021.

18 Maanonen, Mika. 2021. Paisto Oy. <https://paisto.fi/>.