Nea Rantala

SISÄISEN HANKINTAPROSESSIN TEHOSTAMINEN LOWCODE-RATKAISULLA

Opinnäytetyö

Tradenomi

Tietojenkäsittelyn koulutus

2024

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Nea Rantala
Työn nimi	Sisäisen hankintaprosessin kehittäminen LowCode-ratkaisulla
Toimeksiantaja	T-Drill Oy
Vuosi	2024
Sivut	32 sivua, liitteitä 9 sivua
Työn ohjaaja(t)	Marjo Puikkonen

TIIVISTELMÄ

Tässä opinnäytetyössä kehitettiin T-Drill Oy hankintaprosesseja hyödyntämällä Microsoft Power Platformin tarjoamia LowCode-ratkaisuja. Työn tavoitteena oli yksinkertaistaa ja automatisoida hankinta-aloitteiden käsittelyä, mikä parantaisi prosessien tehokkuutta, vähentäisi virheitä ja tukisi yrityksen strategisia tavoitteita. Hankintaprosessit olivat aiemmin monimutkaisia ja hajanaisia, mikä aiheutti viivästyksiä ja lisäkustannuksia.

Opinnäytetyössä sovellettiin sovelluskehityksen elinkaarimallia (SDLC) ja ketteriä menetelmiä, joiden avulla projekti eteni vaiheittain sidosryhmien tarpeiden mukaan. Tärkeimmät työkalut olivat Power Apps, Power Automate ja SharePoint, jotka valittiin niiden helppokäyttöisyyden, integroituvuuden ja skaalautuvuuden vuoksi. Suunnitteluvaiheessa painotettiin erityisesti käyttäjäystävällisyyttä ja tietoturvaa. Prosessien automatisoinnin ansiosta hankintaprosessit nopeutuivat ja resurssit optimoitiin tehokkaammin.

Projektin tuloksena syntynyt järjestelmä paransi hankintaprosessien läpinäkyvyyttä ja tiedonkulkua eri osastojen välillä. Uusi järjestelmä tukee yrityksen pitkän aikavälin kasvustrategiaa ja parantaa sen kilpailukykyä globaaleilla markkinoilla. Työn aikana ilmenneet haasteet, kuten SharePointin hallinta ja tietoturva, otettiin huomioon jatkokehityksessä ja ratkaistiin parannetuilla menettelyillä.

Asiasanat: LowCode, hankintaprosessit, Power Platform, sovelluskehitys, automaatio

Kaakkois-Suomen ammattikorkeakoulu

Degree title	Bachelor of Business Administration
Author (authors)	Nea Rantala
Thesis title	Improving internal procurement process with LowCode
Commissioned by	T-Drill Oy
Time	2024
Pages	32 pages, 9 pages of appendices
Supervisor	Marjo Puikkonen

ABSTRACT

This thesis focused on improving the internal procurement processes at T-Drill Oy by utilizing Low-Code solutions provided by Microsoft Power Platform. The primary objective was to simplify and automate the procurement initiation process to enhance efficiency, reduce errors, and to support the company's strategic goals. The previous procurement processes were complex and fragmented, leading to delays and additional costs.

The thesis applied the Software Development Life Cycle (SDLC) and agile methodologies to guide the project through its various phases that were tailored to the needs of the stakeholders. The main tools used were Power Apps, Power Automate, and SharePoint which were chosen for their ease of use, integration capabilities, and scalability. The design phase emphasized user-friendliness and data security. As a result of automation, the procurement processes were expedited, and resources were optimized more effectively.

The project outcome was a system that improved the transparency and communication of procurement processes across different departments. The new system supports the company's long-term growth strategy and enhances its competitiveness in the global market. Challenges encountered during the project, such as SharePoint management and data security, were addressed in subsequent development phases and resolved with improved methods.

Keywords: LowCode, procurement processes, Power Platform, application development, automation

SISÄLLYS

1 JOHDANTO

Liiketoimintaympäristö muuttuu nopeasti, ja yritysten on jatkuvasti sopeudutta-
tava näihin muutoksiin pysyäkseen kilpailukykyisinä. Teknologian kehitys ja di-
gitalisaatio tarjoavat uusia mahdollisuuksia, mutta ne asettavat myös vaati-
muksia prosessien tehokkuudelle ja ketteryydelle. Valmistavassa teollisuu-
dessa hankintaprosessit ovat erityisen kriittisiä, sillä ne vaikuttavat suoraan
yrityksen operatiiviseen toimintaan ja lopulta myös sen kykyyn vastata asiak-
kaittensa tarpeisiin. Tässä opinnäytetyössä keskitytään T-Drill Oy:n sisäisen
hankintaprosessin kehittämiseen, jossa hyödynnettiin Microsoft Power Platfor-
min tarjoamia LowCode-ratkaisuja.

T-Drill Oy on maailman johtava putkentyöstökoneiden valmistaja. Yrityksen si-
säiset hankintaprosessit ovat kuitenkin nykyisellään monimutkaisia ja hajanai-
sia eri kommunikaatiokanavien vuoksi. Työntekijät käyttävät hankintapyyntö-
jen tekemiseen erilaisia menetelmiä kuten perinteisiä paperilomakkeita, PDF-
lomakkeita, sähköposteja sekä suullisia pyyntöjä. Tämä hajanaisuus tekee
prosessin hallinnasta vaikeaa ja lisää virheiden riskiä, mikä voi johtaa aikatau-
lujen viivästymisiin, lisäkustannuksiin ja työntekijöiden kuormittumiseen.

Hankintaprosessien kehittäminen on T-Drill Oy:lle ensisijaisen tärkeää, sillä
nykyinen järjestelmä kuormittaa sekä työntekijöitä että resursseja tarpeetto-
masti. Manuaalisten ja hajanaisten prosessien yhdenmukaistaminen ja auto-
matisointi voi tuoda merkittäviä parannuksia toiminnan sujuvuuteen, virheiden
vähentämiseen ja kustannusten hallintaan. Tämän opinnäytetyön tavoitteena
on kehittää ratkaisu, joka yksinkertaistaa hankinta-aloitteiden lähettämisen
niin, että se tehostaa yrityksen liiketoimintaprosesseja, parantaa sisäistä kom-
munikaatiota ja tukee yritykset strategisia tavoitteita.

LowCode-ratkaisuiden hyödyntäminen mahdollistaa hankintaprosessin tehos-
tamisen ilman, että yrityksen tarvitsee investoida merkittävästi ulkopuoliseen
kehitysosaamiseen. Power Platformin käyttöön liittyvä oppimiskynnys on ma-
tala, ja sen työkalut integroituvat saumattomasti yrityksen olemassa olevaan
Microsoft-ekosysteemiin. Tämä tekee ratkaisusta paitsi kustannustehokkaan,
myös helposti skaalautuvan yrityksen tulevaisuuden tarpeisiin.

Opinnäytetyön keskeisenä tavoitteena on parantaa T-Drill Oy:n sisäisten hankintaprosessien tehokkuutta, vähentää virheiden määrää ja helpottaa työntekijöiden työtä. Nykyisten hajanaisten ja manuaalisten prosessien korvaaminen automatisoidulla ja yhdenmukaisella järjestelmällä odotetaan tuovan merkittäviä hyötyjä. Ensinnäkin prosessien käsittelyaika nopeutuu, kun kaikki hankintapyynnöt voidaan käsitellä keskitetysti ja automatisoidusti. Tämä vähentää myös manuaalisten virheiden mahdollisuutta, mikä parantaa prosessien tarkkuutta ja luotettavuutta.

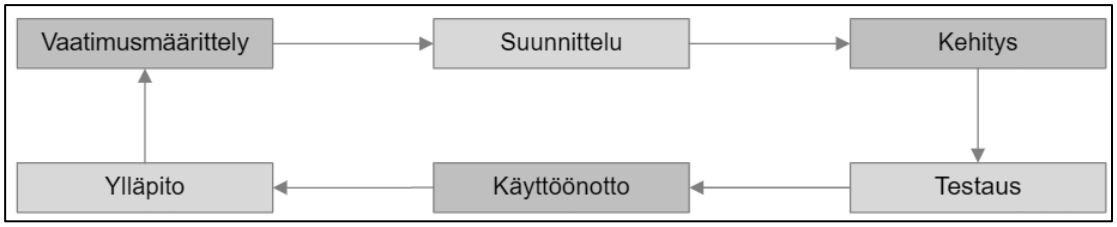
Toiseksi uusi järjestelmä parantaa prosessien läpinäkyvyyttä, kun kaikki tiedot tallennetaan yhteen paikkaan, josta ne ovat helposti saatavilla kaikille prosessiin osallistuville. Tämä vähentää kommunikointivirheitä ja parantaa tiedonkulkua eri osastojen välillä, mikä puolestaan tehostaa koko organisaation toimintaa. Kolmanneksi resurssien käyttö optimoituu, kun työntekijöiden aikaa vapautuu manuaalisista ja rutiininomaisista tehtävistä vaativampiin ja tuottavampiin työtehtäviin. Näiden tavoitteiden saavuttaminen tukee yrityksen pitkän aikavälin kasvustrategiaa ja parantaa sen kilpailukykyä globaalissa markkinassa.

2 SOVELLUSKEHITYS JA LOWCODE

Tämän opinnäytetyön teoreettisessa viitekehyksessä tarkastellaan kahta keskeistä käsitettä: sovelluskehityksen elinkaari sekä NoCode- ja LowCode-kehitys. Nämä käsitteet valittiin niiden tarjoaman laajan näkökulman vuoksi, sillä ne kuvaavat monipuolisesti sovelluskehityksen prosesseja ja menetelmiä, joita hyödynnettiin opinnäytetyönprojektin aikana. Sovelluskehityksen elinkaari tarjoaa selkeän rakenteen kehitysprosessille, mikä auttaa varmistamaan projektin vaiheet johdonmukaisesti ja systemaattisesti. NoCode- ja LowCode-kehitys puolestaan selittävät kehitystyylin erityispiirteet, kuten kehitysajan lyhentämisen ja alhaisemmat kustannukset, jotka voivat olla ratkaisevia tekijöitä projektin onnistumiselle. Yhdessä nämä käsitteet luovat pohjan ymmärrykselle siitä, miten projektin tekninen toteutus suunniteltiin ja toteutettiin.

2.1 Sovelluskehityksen elinkaari

Sovelluskehityksen elinkaari, Software Development Life Cycle (SDLC), on vaiheittainen prosessi, jonka avulla kehitystiimit luovat tehokkaita ja laadukkaita sovelluksia. Dooleyn (2011, 7) mukaan jokaisella sovelluksella on elinkaari, joka koostuu seuraavista vaiheista: alkuidea, vaatimusten määrittely, suunnittelu, kehitys, testaus, käyttöönotto, ylläpito ja kehitystyön päättäminen. Yleisempi, Pereran ja Eadien (2023, 73) esittelemä, SDLC-malli (Kuva 1) keskittyy elinkaaren teknisiin vaiheisiin, jotka ovat suoraan yhteydessä sovelluksen luomiseen ja toteuttamiseen. Tästä mallista puuttuvat Dooleyn laajan mallin alkuidea ja kehitystyön päättäminen.



Kuva 1. Geneerinen SDLC-malli (mukaillen Perera & Eadie 2023, 73)

On tärkeää huomioida, että vaikka SDLC on loogisesti jäsennelty vaiheesta toiseen etenevä prosessi, todelliset projektit eivät aina etene täysin sen mukaisesti. SDLC toimiikin lähinnä yleisenä viitekehyksenä, joka ohjaa kehitystyötä. Toteutustapa voi vaihdella merkittävästi sen mukaan, käytetäänkö perinteistä vesiputousmallia vai ketteriä menetelmiä. (Sommerville 2016, 45-51.)

Elinkaaren vaiheet

SDLC-mallin ensimmäinen vaihe on vaatimusten määrittely, jossa kerätään ja dokumentoidaan sidosryhmien tarpeet ja odotukset. Tyypillisesti tämä prosessi alkaa tunnistamalla sidosryhmät, johon voi kuulua asiakkaita, käyttäjiä, liiketoimintajohtoa ja kehitystiimiä. Sommervillen (2016, 114-119) mukaan on tärkeää käyttää erilaisia menetelmiä, kuten haastatteluja, kyselyitä ja työpajoja, jotta kaikki olennaiset tarpeet saadaan huomioitua. Booch ym. (2005, 4–7) korostavat mallintamisen tärkeyttä, sillä se auttaa visualisoimaan järjestelmän vaatimukset ja varmistamaan, että kaikki sidosryhmät ymmärtävät projek-

tin tavoitteet samalla tavalla. Dooley puolestaan (2011, 99-101) korostaa käyttötapausten soveltuvuutta käyttäjien ja heidän toimintojensa välisten suhteiden kuvaamiseen käyttäjiä osallistavalla tavalla.

Näiden tarpeiden pohjalta luodaan vaatimusmäärittelydokumentti, jossa vaatimukset jaetaan toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Se luo selkeän perustan myöhemmille suunnittelu-, kehitys- ja testausvaiheille. Hyvin määritellyt vaatimukset vähentävät projektin aikana ilmeneviä epäselvyyksiä ja muutostarpeita, mikä parantaa projektin onnistumismahdollisuuksia. Vaatimusten priorisointi auttaa kehitystiimiä keskittymään olennaisiin asioihin, ja hyväksymiskriteerien määrittäminen tarjoaa selkeät mittarit vaatimusten toteuttamisen arvioimiseksi.

Suunnittelu on SDLC-mallin toinen vaihe, jossa kehitetään projektisuunnitelma, joka määrittelee projektin laajuuden, aikataulun ja resurssit (Srinam 2023), sekä sovelluksen yleisen rakenteen ja yksityiskohdat (Sommerville 2016, 56-58). Suunnitteluvaiheessa käytetään erilaisia työkaluja ja menetelmiä projektin osa-alueiden visualisoimiseksi ja organisoimiseksi. Booch ym. (2005, 89–101) esittävät, että UML-diagrammit, kuten luokkadiagrammit ja vuokaaviot, ovat yksi tärkeimmistä työkaluista, ohjelmiston rakenteen suunnittelussa. Sommerville (2016, 62-63) puolestaan nostaa esiin prototyypit, jotka mahdollistavat sidosryhmien varhaisen palautteen. Laitteistovaatimusten määrittely on myös olennainen osa suunnittelua, ja se voi sisältää ohjelmistojen ja laitteistojen yhteensopivuuden arvioinnin lisäksi tarvittavien resurssien, kuten palvelimien ja tietokantojen, suunnittelua (Sommerville 2016; 570-574).

SDLC:n kolmannessa vaiheessa, kehitysvaiheessa, sovellussuunnitelma muutetaan toimivaksi koodiksi. Sovelluksen jokainen osa tai toiminnallisuus koodataan ja testataan yksittäin, jotta voidaan varmistaa, että ne toimivat suunnitellusti. Onnistuminen tässä vaiheessa edellyttää huolellista suunnittelua, tehokasta tiimityötä ja jatkuvaa kommunikaatiota sidosryhmien välillä. (Srinam 2023.)

Testaus on SDLC:n neljäs vaihe, jossa varmistetaan, että sovellus täyttää sille asetetut vaatimukset ja toimii odotetusti. Testausprosessin tulee olla monivaiheinen ja siihen tulisi sisältyä vähintään yksikkötestaus, integraatiotestaus ja

järjestelmätestaus (Sommerville 2016, 56-60; Jorgensen & DeVries 2021, 463). Sen on hyvä sisältää myös regressiotestausta, jolla varmistetaan, että uudet muutokset eivät ole rikkoneet aiemmin toimivia ominaisuuksia (Sommerville 2016, 244). Testauksen tavoitteena on havaita ja korjata virheet ennen sovelluksen käyttöönottoa, jotta vältetään virheellinen toiminta tuotantovaiheessa ja varmistetaan, että lopputuote on laadukas ja luotettava.

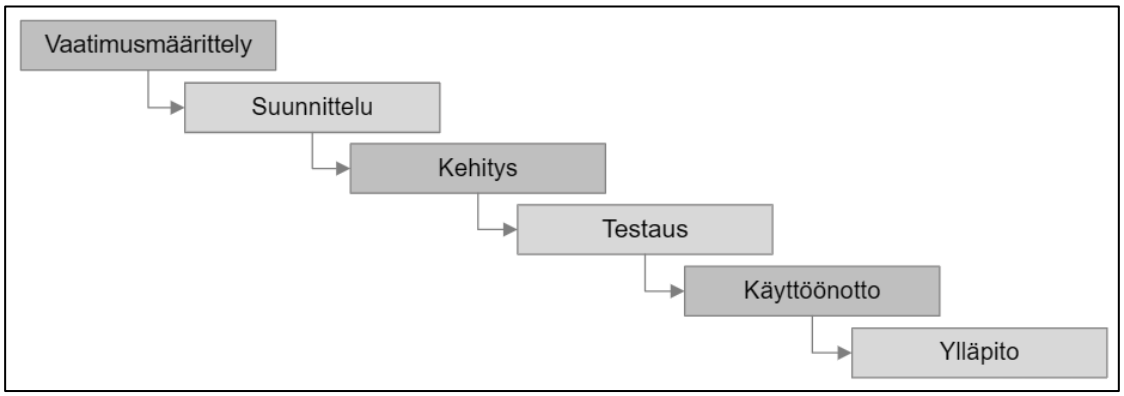
Käyttöönotto on SDLC:n viides vaihe, jossa valmis sovellus tuodaan käyttäjien saataville. Tämä vaihe vaatii huolellista suunnittelua ja koordinointia, jotta siirtyminen kehitysvaiheesta tuotantoon tapahtuu sujuvasti ilman merkittäviä käyttökatkoksia tai virheitä (Srinam 2023). Sommerville (2016, 573-574) korostaa, että käyttöönotto voi olla ennakoitua haastavampaa, sillä käyttäjäympäristö voi poiketa kehittäjien oletuksista, mikä vaatii järjestelmän sopeuttamista. Tämän vuoksi käyttäjäpalautteen kerääminen heti käyttöönoton jälkeen on oleellista mahdollisten ongelmien ja parannustarpeiden tunnistamiseksi.

SDLC:n viimeinen vaihe, ylläpito, alkaa heti käyttöönoton jälkeen ja jatkuu koko sovelluksen elinkaaren ajan. Tämä vaihe varmistaa, että sovellus pysyy toimivana, turvallisena ja ajan tasalla muuttuvien vaatimusten ja teknologioiden kanssa (Perera & Eadie 2023, 75). Ylläpitovaiheeseen sisältyy virheiden korjaaminen, päivitysten tekeminen ja uusien ominaisuuksien lisääminen (Srinam 2023; Sommerville 2016, 271). Ylläpidossa on tärkeä seurata sovelluksen suorituskykyä ja käyttäjäpalautetta jatkuvasti.

Vesiputous ja ketterät menetelmät

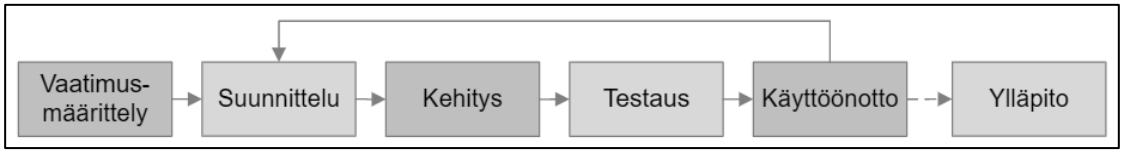
Elinkaaren hallintaan on olemassa useita eri malleja. Ne voidaan jakaa kahteen päätyyppiin: perinteisiin suunnitelmalähtöisiin malleihin ja ketteriin menetelmiin. Vesiputousmalli (Kuva 2) on yksi tunnetuimmista perinteisistä malleista. Vesiputousmallissa jokainen SDLC:n vaihe suoritetaan loppuun ennen seuraavaan vaiheeseen siirtymistä. Tämä voi aiheuttaa ongelmia, sillä projektin alussa ei usein pystytä ennakoimaan kaikkia myöhemmissä vaiheissa ilmeneviä haasteita, mikä voi johtaa merkittäviin ongelmiin projektin edetessä (Dooley 2011, 10). Vesiputousmallin ymmärretäänkin soveltuvan parhaiten projekteihin, joiden vaatimukset ovat selkeät ja vähän odotettavissa olevia

muutoksia. Tällaisia projekteja ovat usein hyvin määritellyt ja teknisesti suoraviivaiset hankkeet, joissa riskit ovat alhaiset. Kuitenkin monimutkaisemmissa projekteissa, joissa vaatimukset voivat muuttua, vesiputousmallin jäykkyys saattaa hidastaa kehitysprosessia ja vaikeuttaa muutosten toteuttamista projektin aikana (Dooley 2011, 10; Stellman & Greene 2015, 16-19).



Kuva 2. Vesiputousmalli (mukaillen Perera & Eadie 2023, 76)

Vesiputousmallista poiketen, ketterissä menetelmissä projekti jaetaan toistuviin kehitysjaksoihin, iteraatioihin (Stellman & Greene 2015, 87). Dooleyn (2011, 7) mukaan tietyt vaiheet, kuten suunnittelu, kehitys, testaus ja käyttöönotto voidaan toistaa useita kertoja ennen sovellusprojektin valmistumista (Kuva 3). Tämä joustava lähestymistapa auttaa tiimejä reagoimaan nopeasti muutoksiin ja parantaa lopputuotteen laatua. Ketterät menetelmät sopivat hyvin projekteihin, joissa vaatimukset voivat muuttua, ja joissa tarvitaan jatkuvaa asiakaspalautetta (Perera & Eadie 2023, 95-97). Iteratiivinen työskentely mahdollistaa myös sen, että sidosryhmät voivat tarkastella ja arvioida työn tuloksia useissa vaiheissa, mikä auttaa varmistamaan, että sovellus täyttää sille asetetut vaatimukset ja vastaa käyttäjien tarpeisiin.



Kuva 3. Ketterä menetelmä

Yhteenvetona voidaan todeta, että vesiputousmalli ja ketterät menetelmät edustavat kahta erilaista lähestymistapaa ohjelmistokehityksessä. Vesiputousmalli tarjoaa vakautta ja järjestelmällisyyttä, mutta sen jäykkyys voi muodostua haasteeksi monimutkaisemmissa ja muutoksiin taipuvaisissa projekteissa.

Ketterät menetelmät sen sijaan mahdollistavat nopean reagoinnin muutoksiin, mikä tekee niistä erityisen sopivia projekteihin, joissa asiakkaan tarpeet ja projektin vaatimukset voivat muuttua kehityksen aikana. Lopulta menetelmän valinta riippuu kuitenkin projektin luonteesta ja sen asettamista vaatimuksista. (Sommerville 2016, 43-71; Perera & Eadie 2023, 71-103.)

2.2 NoCode- ja LowCode-kehitys

NoCode- ja LowCode-kehityksestä on tullut keskeinen osa yritysten työkalupakkia, kun ne tavoittelevat nopeaa ja kustannustehokasta liiketoimintasovellusten kehittämistä. Näiden alustojen avulla myös ei-tekniset asiantuntijat voivat osallistua sovellusten kehittämiseen. Gartner ennustaa, että vuoteen 2025 mennessä 70 prosenttia yrityksissä luoduista sovelluksista toteutetaan NoCode- ja LowCode-alustoilla (Gartner 2021). Tämän kehityssuunnan myötä nämä alustat ovat nousseet merkittäväksi trendiksi nykyaikaisessa sovelluskehityksessä (Gartner 2023; Bratincevic & Guidice 2023).

NoCode- ja LowCode-kehitys ei ainoastaan nopeuta sovelluskehitysprosessia, vaan myös mahdollistaa liiketoimintayksiköiden nopeamman reagoinnin markkinoiden muutoksiin. Bratincevicin ja Guidicen (2023) mukaan yritykset voivat ottaa käyttöön uusia sovelluksia ja päivittää olemassa olevia ratkaisuja huomattavasti nopeammin kuin perinteisillä kehitysmenetelmillä. Tämä ketteryys on erityisen tärkeää nykypäivän kilpailuympäristössä, jossa teknologian ja liiketoiminnan tarpeet voivat muuttua nopeasti ja ennakoimattomasti. Lisäksi, koska NoCode- ja LowCode-alustat antavat mahdollisuuden kehittää sovelluksia ilman syvällistä ohjelmointiosaamista, yritysten sisäiset tiimit voivat kehittää ja hallita ratkaisujaan itsenäisemmin, mikä Forresterin (2020) mukaan vähentää riippuvuutta IT-osastoista ja ulkoisista toimittajista.

Ominaisuudet ja erot

NoCode- ja LowCode-alustat tarjoavat valmiita visuaalisia komponentteja, joilla voidaan rakentaa toimivia sovelluksia drag and drop -tyyppisesti. Näiden työkalujen ansiosta liiketoiminnan asiantuntijat voivat osallistua kehitykseen ilman perinteisten ohjelmointikielten hallintaa. Alustat voidaan jakaa kahteen

pääkategoriaan: täysin koodittomat ratkaisut – noCode – ja vähän koodia sisältävät ratkaisut – LowCode. (Team Kissflow 2024.; IMB 2022).

Kissflown blogin (Team Kissflow 2024) mukaan NoCode- ja LowCode-ratkaisut eroavat toisistaan vaadittavan teknisen osaamisen, sovelluksen räätälöintimahdollisuuksien, sekä abstraktion tason osalta. NoCode-alustat on suunniteltu tukemaan yksinkertaisia ja täysin visuaalisia, mallipohjaisia ratkaisuja, joissa kaikki toiminnallisuus voidaan toteuttaa ilman koodin kirjoittamista. LowCode-alustat puolestaan tarjoavat visuaalisen kehitysympäristön lisäksi mahdollisuuden lisätä yksityiskohtaista ohjelmointilogiikkaa. Microsoft Power Platform hyödyntää Excelin kaltaista Power Fx -kaavakieltä, jonka omaksuminen on helppoa – erityisesti käyttäjille, joilla on kokemusta taulukkolaskentaohjelmista (Microsoft 2024a). LowCode-alustojen tarjoamat räätälöintimahdollisuudet antavat kehittäjille vapauden rakentaa monimutkaisempia ja vaativampia projekteja samalla kun he hyötyvät perinteisiä ohjelmointimenetelmiä nopeammasta kehityksestä (Rokis & Kirikova 2023).

Hyödyt ja haasteet

NoCode- ja LowCode-kehitys tarjoaa merkittäviä etuja, kuten nopeamman sovelluskehityksen, alhaisemmat kustannukset ja vähentyneen riippuvuuden ohjelmistoresursseista (Rokis & Kirikova 2023). Forresterin tutkimuksessa (2020) havaittiin, että Power Apps -alustaa käyttävät yritykset voivat vähentää kehityskustannuksiaan jopa 74 % ja nopeuttaa sovellusten käyttöönottoa merkittävästi. Näiden alustojen avulla yritykset voivat lisätä liiketoimintaketteryyttä, kun myös ei-tekniset asiantuntijat voivat osallistua kehitysprosesseihin, mikä vähentää riippuvuutta IT-osastoista ja ammattiohjelmoijista sekä vahvistaa yrityksen sisäistä osaamista (Forrester 2020).

Samalla NoCode- ja LowCode-alustat ovat syventäneet yhteistyötä IT-osastojen ja liiketoimintayksiköiden välillä. Ne mahdollistavat tiiviimmän yhteistyön liiketoiminta-asiantuntijoiden ja teknisten tiimien kesken, mikä varmistaa, että kehitettävät sovellukset vastaavat tarkasti liiketoiminnan tarpeisiin (Rokis & Kirikova 2023). Tämä lähestymistapa kaventaa perinteistä kuilua teknisen ja lii-

ketoiminnallisen osaamisen välillä, mikä nopeuttaa kehitysprosessia ja parantaa sovellusten laatua ja käytettävyyttä, kun käyttäjät ovat mukana kehityksen jokaisessa vaiheessa.

Vaikka NoCode- ja LowCode-kehitys tarjoaa monia etuja, niihin liittyy myös merkittäviä haasteita, kuten tietoturva- ja skaalautuvuusongelmat. Vaikka nämä alustat mahdollistavat nopean kehityksen, ne eivät aina sovellu monimutkaisten sovellusten kehittämiseen, mikä voi rajoittaa niiden käyttöä erityisesti vaativissa projekteissa. Lisäksi riippuvuus tietystä toimittajasta (eng. vendor lock-in) voi sitoa yrityksen yhteen palveluntarjoajaan, mikä vähentää joustavuutta ja vaihtoehtoja tulevaisuudessa. Vaikka LowCode-alustat voivat tuoda kustannussäästöjä lyhyellä aikavälillä, niiden lisenssikustannukset voivat nousta korkeiksi pitkällä aikavälillä, mikä kasvattaa kokonaiskustannuksia. (Martinez & Pfister, 2023.)

Organisaatioiden on tärkeää harkita tarkkaan NoCode- ja LowCode-alustojen tarjoamia hyötyjä suhteessa niiden rajoituksiin. Vaikka nämä alustat tukevat tehokkaasti yksinkertaisia ja keskinkertaisen monimutkaisia projekteja, ne eivät välttämättä sovi kaikkiin sovellustyyppeihin, mikä voi rajoittaa niiden käyttöä tietyissä liiketoimintaskenaarioissa. Martinez ja Pfister (2023) huomauttavat lisäksi, että nopeasti kehitetyt sovellukset voivat jäädä hallinnan ja ylläpidon osalta puutteellisiksi, mikä vaarantaa niiden toimivuuden ja turvallisuuden. Tietoturva on erityisen tärkeä huolenaihe, sillä nämä alustat voivat altistaa sovelluksia tietomurroille, jos turvallisuuskäytäntöjä ei noudateta asianmukaisesti. Oikein käytettyinä NoCode- ja LowCode-alustat voivat kuitenkin merkittävästi tehostaa sovelluskehitystä ja parantaa liiketoiminnan ketteryyttä, mikä on keskeistä pitkän aikavälin kilpailukyvyn kannalta (Martinez & Pfister 2023).

Tulevaisuuden näkymät ja kehityssuunnat

NoCode- ja LowCode-alustojen tulevaisuus vaikuttaa erittäin lupaavalta, ja markkinoiden odotetaan kasvavan merkittävästi. Gartnerin ennusteiden mukaan NoCode- ja LowCode-markkinoiden arvo nousee jopa 44,5 miljardiin dollariin, ja yli 80 % kaikista sovelluskehitysprojekteista toteutetaan näillä alustoilla vuoteen 2026 mennessä (Gartner 2023). Lisäksi uusien teknologioiden, kuten tekoälyn ja koneoppimisen, integroiminen näihin alustoihin mahdollistaa

entistä älykkäämpien ja automatisoidumpien sovellusten kehittämisen (Bratincevic & Guidice 2023).

Ohjelmistokehityksen kenttä muuttuu väistämättä NoCode- ja LowCode-alustojen yleistyessä. Tämä suuntaus luo uusia mahdollisuuksia ja haasteita sekä kehittäjille että liiketoimintayksiköille. Uudet roolit, kuten "Citizen Developer" eli ei-ammattilainen kehittäjä, yleistyvät, kun taas perinteiset ohjelmointiroolit saattavat keskittyä yhä enemmän alustan hallintaan, tietoturvaan ja monimutkaisten logiikkojen kehittämiseen (Berardi ym. 2023). Tämä muutos vaatii ohjelmistokehittäjiltä uusien taitojen omaksumista, kuten alustaosaamisen syventämistä ja liiketoiminnan tarpeiden syvällisempää ymmärtämistä.

3 ALKUIDEA, VAATIMUSMÄÄRITTELY JA SUUNNITTELU

Projektin alkuidea syntyi IT-päällikön ja hankintapäällikön aloitteesta. Heillä oli selkeä näkemys, että sisäistä hankinta-aloiteprosessia pitäisi kehittää, mutta tarkkaa vaatimusmäärittelyä ei ollut tehty, eikä toteutusalustaa tai kehitystyökaluja ollut vielä valittu. Toiveena oli, että ratkaisua kehitettäisiin olemassa olevilla työkaluilla ja käytössä olevien käyttäjälisenssien puitteissa. Koska yrityksellä oli laajasti käytössä Microsoftin tuoteperheen työkaluja ja lisenssejä, toteutusalustaksi valikoitui Microsoft Power Platform. Tämä alusta valittiin sen takia, että se integroitui saumattomasti yrityksen olemassa olevaan IT-infrastruktuuriin, jolloin ei tarvittu erillisiä lisenssejä tai ulkopuolista asiantuntija-apua. Lisäksi Power Platformin LowCode-lähestymistapa mahdollisti nopean kehitysprosessin ja mukautui hyvin projektin tarpeisiin. Projektin pääasiallisina kehitystyökaluina käytettiin Power Apps- ja Power Automate -työkaluja, ja SharePoint-luetteloita. Myös muita Microsoft tuoteperheen työkaluja käytettiin projektityön tukena.

Projektissa oli aktiivisesti mukana kaksi sidosryhmän edustajaa sekä yksi kehittäjä. Pienen tiimikoon vuoksi dokumentointi ja hierarkia pidettiin mahdollisimman vähäisinä, mikä mahdollisti ketterän ja joustavan työskentelytavan. Etenemisen seuranta toteutettiin pääasiassa suullisesti projektitapaamisten yhteydessä. Vaikka tiimi oli pieni, vastuut jaettiin jäsenten kesken seuraavasti: Kehittäjä vastasi sovelluksen toteutuksesta, IT-päällikön valvoi projektin etenemistä ja aikataulutti projektitapaamiset, ja sekä IT-päällikkö että hankinta-

päällikkö antoivat jatkuvaa palautetta ja täydensivät vaatimuksia projektin edetessä. Tämä lähestymistapa varmisti, että kehitys pysyy linjassa yrityksen tarpeiden kanssa ja muutoksiin voitiin reagoida nopeasti.

Tässä projektissa sovelluskehityksen elinkaarimallia sovellettiin niin, että jokainen vaihe muokattiin yrityksen tarpeiden mukaan. Vaatimusten määrittely käynnistyi ensimmäisten projektitapaamisten aikana, joissa kartoitettiin nykyisiä hankintaprosesseja vapaamuotoisten haastatteluiden avulla. Haastateltavina olivat projektin avainhenkilöt: IT-päällikkö ja hankintapäällikkö. IT-päällikön näkökulmasta tärkeintä oli varmistaa, että valittu ratkaisu integroitui saumattomasti yrityksen olemassa oleviin järjestelmiin ja infrastruktuuriin. Hankintapäällikkö puolestaan korosti prosessin sujuvuuden ja yksinkertaisuuden merkitystä, mikä oli keskeistä työntekijöiden ajan säästämiseksi ja virheiden vähentämiseksi. Tämä johti käyttöliittymän huolelliseen suunnitteluun, jossa keskiössä olivat selkeys, intuitiivisuus ja prosessin automatisointi. Näiden tarpeiden pohjalta luotiin sekä toiminnalliset että ei-toiminnalliset vaatimukset, ja määritettiin järjestelmävaatimukset sekä projektin rajoitukset.

Toiminnalliset vaatimukset (Taulukko 1, T1-T6) keskittyivät käyttäjäkokemuksen optimointiin ja tiedonkäsittelyn tehostamiseen. Tärkeitä ominaisuuksia olivat mm. tietojen syöttö- ja tarkastelulomakkeet, sekä tiedon tallennus, muokkaus ja haku. Esimerkiksi lomakkeiden helppokäyttöisyys ja nopea tiedon käsittely olivat avainasemassa, jotta hankinta-aloitteet voitiin käsitellä tehokkaasti. Ei-toiminnalliset vaatimukset (Taulukko 1, E1-E3) painottivat suorituskykyä, skaalautuvuutta ja käytettävyyttä, jotta järjestelmä voisi mukautua kasvaviin tarpeisiin ja pysyä ajan tasalla ilman merkittäviä muutoksia infrastruktuuriin. Järjestelmävaatimuksissa (Taulukko 1, J1-J3) korostettiin tietoturvaa ja SharePoint-yhteensopivuutta. Rajoitukset määrittelivät projektin laajuuden hankinta-aloitteen lähettämiseen ja tallennukseen. Aikataulu rajattiin opinnäytetyön aikataulun mukaiseksi, ja resurssit rajoitettiin olemassa oleviin käyttäjälisensseihin ja kehitystyökaluihin.

Taulukko 1. Vaatimukset

ID	Päätoiminto	Ominaisuudet	Tärkeys
T1	Käyttöliittymä	Lomakkeet tietojen syöttämiseen	Kriittinen
T2	Käyttöliittymä	Listausnäkymä käyttäjän lisäämien tuotteiden tarkasteluun	Kriittinen
T3	Tiedonkäsittely	Tietojen tallennus, muokkaus ja poisto	Kriittinen
T4	Tiedonkäsittely	Tietojen haku ja suodatus	Kriittinen
T5	Notifikaatiot	Ilmoitusten lähettäminen käyttäjille	Tärkeä
T6	Integraatiot	Integraatio muihin järjestelmiin	Tärkeä
E1	Suorituskyky	Nopea reagointi ja tietojen käsittely	Kriittinen
E2	Skaalautuvuus	Mahdollisuus lisätä uusia toimintoja	Tärkeä
E3	Käytettävyys	Helppokäyttöinen ja intuitiivinen käyttöliittymä	Kriittinen
J1	Käyttöympäristö	Toimii SharePoint -ympäristössä	Kriittinen
J2	Tietoturva	Tietojen suojauksen ja salauksen toteuttaminen	Kriittinen
J3	Tietoturva	Tietojen turvallinen tallennus	Kriittinen

Kun vaatimukset oli määritelty, alettiin suunnitella sovelluksen arkkitehtuuria, joka täyttäisi nämä vaatimukset. Vaatimusmäärittelyn pohjalta tavoitteeksi muodostui kehittää sähköinen hankinta-aloitelomake, määrittää tietojen tallennuspaikka ja automatisoida tietyt toistuvat toiminnot. Näiden tavoitteiden saavuttamiseksi ratkaisun arkkitehtuuria alettiin kehittää kolmella keskeisellä tasolla: tietokerros, käyttöliittymä ja logiikkakerros.

Tietokerros on olennainen osa sovellusarkkitehtuuria, sillä se vastaa tietojen tallennuksesta, hausta ja hallinnasta. Suunnittelun painopisteenä oli varmistaa tietojen eheys ja turvallisuus. Tietojen tallennus ja hallinta toteutettiin SharePoint-luetteloiden avulla niiden joustavuuden ja tehokkuuden ansiosta. Tietokantarakenne suunniteltiin siten, että luettelot liittyivät toisiinsa yksi-moneen-suhteessa, mikä mahdollisti tietojen tehokkaan hallinnan ja organisoinnin.

Käyttöliittymä toimii käyttäjän ja järjestelmän välisenä rajapintana. Suunnittelussa keskityttiin käyttäjäystävällisyyteen ja intuitiivisuuteen, jotta työntekijät voisivat käyttää sovellusta ilman erityistä teknistä koulutusta. Käyttäjäpalautetta ja tuoteomistajien tarkentamia vaatimuksia kerättiin koko kehitysprosessin ajan, ja sovellusta muokattiin saadun palautteen perusteella vastaamaan paremmin käyttäjien tarpeita. Esimerkiksi tuotteen nimikenumero-kenttää muutettiin hyväksymään aakkosnumeerinen syöte aiemman numeerisen syötteen sijaan.

Logiikkakerroksen suunnittelu keskittyi sovelluksen toimintalogiikkaan, kuten tietojen käsittelyyn ja käyttäjän toiminnan perusteella tehtäviin toimintoihin. Suunnittelussa huomioitiin erityisesti sovelluksen dynaaminen käyttäytyminen ja syötekenttien validointi, sekä toistuvien toimintojen automatisointi Power Automate -työnkulkujen avulla. Lisäksi suunnittelussa otettiin huomioon mahdollisuus laajentaa ja mukauttaa sovellusta tulevaisuudessa ilman merkittäviä rakenteellisia muutoksia.

4 KEHITYS

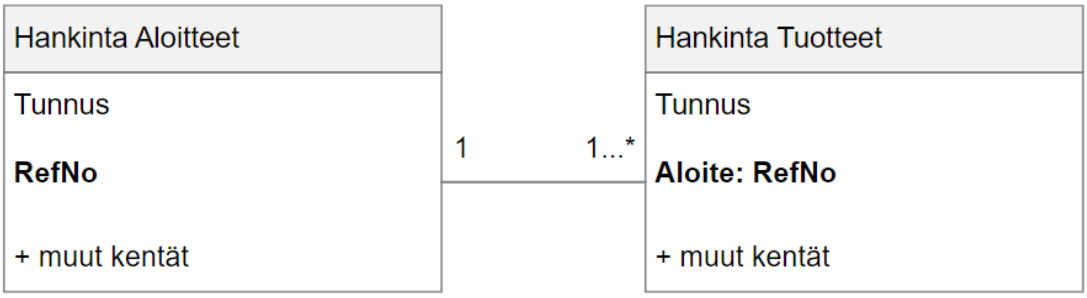
Sovellusta kehitettiin iteratiivisesti hyödyntäen ketteriä menetelmiä. Tämä mahdollisti sovelluksen kehittämisen vaiheittain, jolloin jokainen uusi toiminnallisuus testattiin ja validoitiin ennen siirtymistä seuraavaan vaiheeseen. Tällä tavalla vältettiin perinteisen vesiputousmallin jäykkyys ja voitiin reagoida nopeasti muutoksiin ja käyttäjäpalautteeseen. Projektin tekninen toteutus perustui SharePoint-luetteloihin sekä Power Apps- ja Power Automate -työkaluihin, joiden avulla muodostettiin sovelluksen keskeiset komponentit: tietokerros, käyttöliittymä ja logiikkakerros.

4.1 Tietokerros

Tietokerros on sovelluksen ydin, joka mahdollistaa tiedon tehokkaan hallinnan ja turvallisen käytön. Tämä kerros on ratkaisevan tärkeä sovelluksen arkkitehtuurille, sillä se huolehtii tiedon tallennuksesta, hallinnasta ja käytettävyydestä kaikissa sovelluksen osissa. Hyvin suunniteltu ja toteutettu tietokerros takaa, että kaikki hankinta-aloitteisiin liittyvä tieto on oikein tallennettu, helposti saatavilla ja luotettavasti käytettävissä. Tietojen eheys oli tässä projektissa erityisen tärkeää, koska pienikin virhe hankinta-aloitteiden käsittelyssä voisi johtaa merkittäviin kustannusylityksiin tai viivästyksiin. Käytetyt ratkaisut, kuten johdannaispoisto ja tiukat käyttöoikeudet, varmistavat, että tiedot pysyivät luotettavina ja ajantasaisina, mikä parantaa koko järjestelmän tehokkuutta ja luotettavuutta. Tämä kokonaisuus tarjoaa käyttäjille turvallisen ja sujuvasti toimivan työkalun päivittäisiin tehtäviin, varmistaen, että kaikki sovelluksen osat toimivat saumattomasti yhdessä.

Luettelot

Tietokerros koostuu kahdesta SharePoint-luettelosta: Hankinta Aloitteet ja Hankinta Tuotteet, jotka muodostavat sovelluksen tietovaraston. Luettelot liittyvät toisiinsa yksi-moneen-suhteessa, mikä tarkoittaa, että yksi Hankinta Aloitteet -luettelon aloite voi sisältää useita Hankinta Tuotteet -luettelon tuotteita. Tämä rakenne mahdollistaa tiedon tehokkaan hallinnan ja selkeän organisoinnin, varmistaen, että jokainen tuote liittyy oikeaan aloitteeseen, sekä tehostaa tietojen hallintaa ja hakua. Kuva 4 havainnollistaa, kuinka yksi-moneen-suhde toteutuu näiden luetteloiden välillä.



Kuva 4. Yksi-moneen-suhde Hankinta Aloitteet- ja Hankinta Tuotteet -luetteloiden välillä

Hankinta Aloitteet-luettelo on projektin ydin, sillä se sisältää kaikki sovelluksen kautta tehdyt hankinta-aloitteet. Jokainen aloite tallennetaan omaan tietueeseensa, jossa on useita kenttiä, jotka tallentavat aloitetta koskevia tietoja. Yksi tärkeimmistä kentistä on RefNo, joka toimii aloitteen yksilöllisenä tunnisteena, ja luettelojen välisenä viiteavaimena. Kenttien rakenteet ja ominaisuudet, kuten tietotyyppien määrittely ja kenttien pakollisuus, on suunniteltu varmistamaan, että tiedot ovat yhdenmukaisia ja oikein syötettyjä.

Hankinta Tuotteet -luettelo täydentää Hankinta Aloitteet -luetteloa tallentamalla kaikki aloitteisiin liittyvät tuotteet. Jokainen tuote tallennetaan omaan tietueeseensa, ja Power Apps -sovellus hakee automaattisesti oikean aloitteen viiteavaimen ja liittää sen tuotetietueen Aloite:RefNo -hakukenttään. Tämän automaattisen viittauksen ansiosta tuotteet voidaan helposti yhdistää oikeisiin aloitteisiin, mikä parantaa tiedon hallittavuutta ja mahdollistaa monipuolisemmat hakutoiminnot.

Tietojen eheys

Tietojen eheyden varmistamiseksi SharePoint-luetteloihin määritettiin useita ominaisuuksia, kuten tietotyyppien määrittely ja pakollisten kenttien asettaminen, jotka estävät virheellisen tiedon tallentamisen. Näiden ominaisuuksien ansiosta järjestelmä varmistaa, että syötetyt tiedot ovat yhtenäisiä ja sovelluksen toiminnallisuudet pysyvät luotettavina. Esimerkiksi pakolliset kentät, kuten RefNo, varmistavat, että jokaisella aloitteella ja siihen liittyvällä tuotteella on oma yksilöllinen tunniste, mikä mahdollistaa tietueiden oikean yhdistämisen ja haun.

RefNo -hakusarakkeelle määritettiin johdannaispoisto, joka pitää huolen tietojen eheydestä. Microsoftin (2024, s.a.) dokumentaation mukaan johdannaispoisto poistaa kohdeluettelon kaikki lähdeluettelon poistettuun tietueeseen liittyvät tietueet. Toisin sanoen, jos Hankinta Aloitteet -luettelosta poistetaan tietue, kaikki siihen liittyvät tuotteet poistetaan Hankinta Tuotteet -luettelosta. Tämä toiminto estää "orpojen" tietueiden syntymisen, jotka voisivat muuten jäädä tietokantaan ilman viittausta ja aiheuttaa virheitä tai epäjohdonmukaisuuksia hankintaprosessissa.

Johdannaispoisto ei ainoastaan suojaa tietokannan eheyttä, vaan se myös yksinkertaistaa tietojen hallintaa. Kun poistettu aloite automaattisesti poistaa siihen liittyvät tuotteet, järjestelmänvalvojien ei tarvitse huolehtia manuaalisesta tiedon siivouksesta, mikä vähentää virheiden mahdollisuutta ja säästää aikaa. Tämä varmistaa, että tietokanta pysyy siistinä ja toimintakykyisenä, mikä on erityisen tärkeää järjestelmissä, joissa käsitellään suuria määriä tietoa. Lisäksi tämä ominaisuus parantaa käyttäjäkokemusta, sillä se vähentää tarpeettomien tietojen säilyttämisen riskiä ja varmistaa, että kaikki järjestelmässä säilytetty tieto on relevanttia ja ajantasaista.

Käyttöoikeuksien hallinta

Jotta tiedot pysyvät turvallisina, SharePoint-luetteloihin määritettiin tarkat käyttöoikeudet, jotka estävät luvattoman pääsyn tietoihin. Käyttäjäryhmät ja roolit suunniteltiin siten, että vain valtuutetuilla henkilöillä on oikeus muokata tai

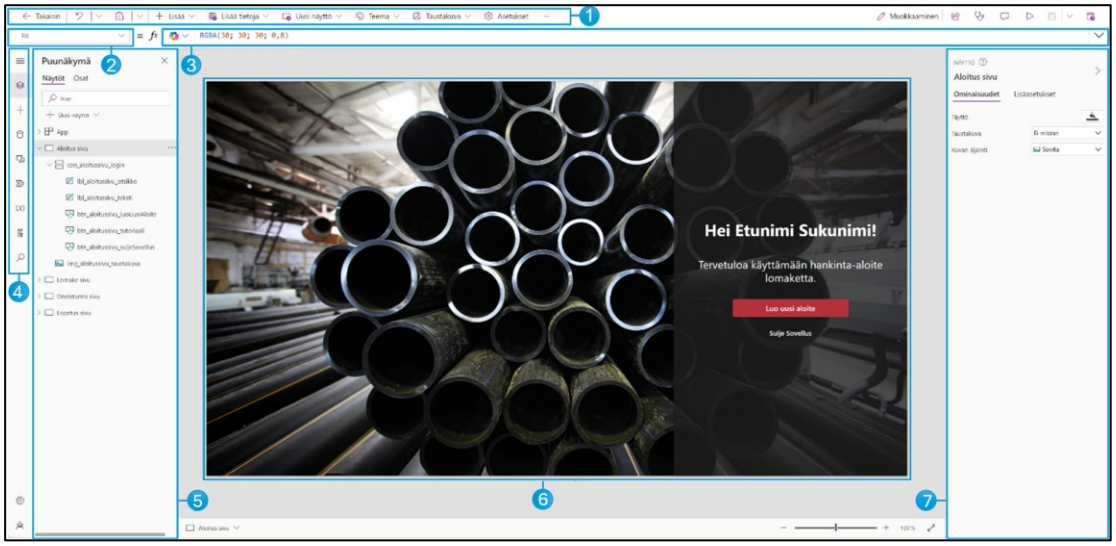
poistaa tietoja. Käyttöoikeuksien hallinta on yksi keskeisimmistä tietoturvatoimenpiteistä, sillä se varmistaa, että arkaluontoiset tiedot ovat suojassa ja että järjestelmässä toimitaan yrityksen tietoturvapolitiikan mukaisesti. Kaikille käyttäjille annettiin oikeudet tallentaa aloitteiden tiedot järjestelmään, mutta muokkaus- ja poisto-oikeudet rajattiin ainoastaan käsittelijöille. Tämä takaa, että vain valtuutetut ja osaavat henkilöt voivat tehdä muutoksia, mikä vähentää inhimillisten virheiden ja tietoturvariskien mahdollisuutta. Käyttäjäryhmien ja roolien määrittely tehtiin tiiviissä yhteistyössä yrityksen tietohallinnon kanssa, jotta kaikki tietoturvavaatimukset täyttyvät.

4.2 Käyttöliittymä ja Power Apps

Lähtökohtana käyttöliittymän suunnittelussa oli varmistaa, että se vastaa T-Drill Oy:n hankintaprosessin erityistarpeita. Päätavoitteena oli luoda selkeä ja intuitiivinen rajapinta, joka ohjaa käyttäjiä täyttämään hankinta-aloitteen tiedot nopeasti ja oikein. Tärkeimmät tavoitteet olivat käyttäjäystävällisyys, nopeus ja virheiden vähentäminen, jotta jokainen hankinta-aloite voidaan käsitellä sujuvasti ja tehokkaasti. Näiden tavoitteiden saavuttamiseksi käyttöliittymän suunnittelu perustui perusteelliseen käyttäjäpalautteen analysointiin sekä Power Apps -ympäristön tarjoamien teknisten mahdollisuuksien tehokkaaseen hyödyntämiseen.

Power Apps Studio, ohjausobjektit, kaavat ja muuttujat

Power Apps Studio tarjoaa kattavan ympäristön, jossa sovelluksen käyttöliittymä voidaan rakentaa ja jossa sitä voidaan hallita joustavasti. Sen avulla pystyttiin luomaan dynaaminen, käyttäjäystävällinen rajapinta, joka tukee hankintaprosessin sujuvuutta ja vähentää käyttäjien tekemien virheiden määrää. Tämän opinnäytetyöprojektin kannalta keskeisimmät elementit olivat (Kuva 5): komentopalkki (1), ominaisuusluettelo (2), kaavarivi (3), muokkausvalikko (4), puunäkymä (5), näyttö (6) ja ominaisuuspaneeli (7).



Kuva 5. Power Apps Studion elementit (mukaillen Microsoft 2024b)

Ohjausobjektit ovat yksittäisiä käyttöliittymäelementtejä, joiden kautta käyttäjä on vuorovaikutuksessa sovelluksen kanssa, ja joita voidaan lisätä sovellukseen komentopalkin (Kuva 5, elementti 1) tai muokkausvalikon (Kuva 5, elementti 4) kautta. Niitä ovat muun muassa painikkeet, tekstikentät, kuvat ja galleriat, jotka auttavat käyttäjää syöttämään ja tarkastelemaan tietoja sujuvasti. Ohjausobjektin ominaisuuksia ja toiminnallisuutta voidaan muokata kaavarivillä (Kuva 5, elementti 3). Kaavojen avulla voidaan määrittää sovelluksen toiminnallisuutta ja logiikkaa, kuten, mitä tapahtuu, kun painiketta painetaan tai miten tietoja suodatetaan. Kaavat voivat sisältää funktioita, operaattoreita, ja viittauksia ohjausobjekteihin ja muuttujiin.

Muuttujat ovat tietovarastoja, joita käytetään sovelluksen sisällä tietojen väliaikaiseen tallentamiseen. Ne jaetaan kolmeen päätyyppiin: globaalit muuttujat, paikalliset muuttujat ja kokoelmat. Globaaleja muuttujia ja kokoelmia voidaan käyttää kaikkialla sovelluksessa, mikä tekee niistä hyödyllisiä tietojen jakamiseen eri näyttöjen välillä. Paikalliset muuttujat sen sijaan ovat käytettävissä vain tietyssä näytössä (Microsoft 2023), mikä rajoittaa niiden käyttöä, mutta parantaa tietojen hallittavuutta. Kokoelmat ovat monipuolisia tietorakenteita, jotka voivat sisältää useita tietueita. On tärkeää huomioida, että muuttujat ja kokoelmat nollautuvat sovellusistunnon päätyttyä (Microsoft 2023).

Sovelluksen toiminnallisuus ja näytöt

Power Apps Studion työkalut mahdollistivat käyttöliittymän suunnittelun ja toteutuksen T-Drill Oy:n hankintaprosessin tarpeiden mukaisesti. Sovelluksen toiminnallisuus jaettiin neljään näyttöön: aloitussivu (liite 1), lomakesivu (liite 2), onnistumissivu (liite 3) ja lopetussivu (liite 4). Näistä keskeisin on lomakesivu, joka toimii tiedon syöttämisen ja tallentamisen rajapintana. Muut näytöt tukevat tätä prosessia tarjoamalla käyttäjälle selkeän navigointipolun ja palautetta suoritettujen toimien onnistumisesta.

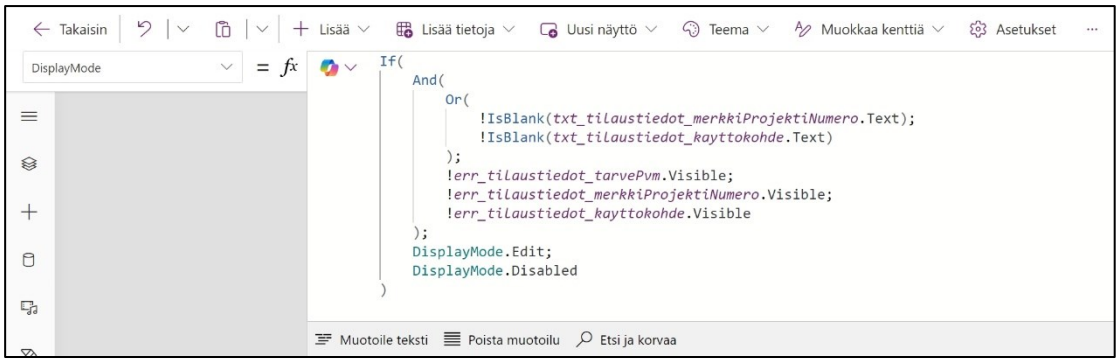
Lomakesivun suunnittelussa haluttiin varmistaa, että käyttäjä pääsee käsiksi vain olennaisiin tietoihin ja että tiedonsyöttöprosessi olisi mahdollisimman intuitiivinen ja virheetön. Tämän saavuttamiseksi käyttöliittymässä hyödynnettiin ehdollista muotoilua ja dynaamisia näkymiä, jotka mukautuvat käyttäjän toimien perusteella. Lomakesivulle luotiin neljä näkymää: perustietonäkymä (liite 2/1), gallerianäkymä (liite 2/2), dialoginäkymä (liite 2/3) ja vahvistusnäkymä (liite 2/4).

Perustietonäkymässä on sovelluksen ainoa lomake, joka on yhdistetty tietolähteenä toimivaan Hankinta Aloitteet -luetteloon, ja se sisältää vain täytettäväksi vaadittavat sarakkeet. Pakolliset kentät on merkitty asteriskilla, ja muut toiminnot aktivoituvat vasta, kun nämä kentät on täytetty hyväksytysti. Tämä varmistaa, että eteneminen tapahtuu vasta, kun kaikki tarvittavat tiedot on syötetty oikein. Samankaltaista validointia hyödynnetään myös tuotetietojen osalta dialoginäkymässä. Lisättyjen tuotteiden tiedot tulevat näkyville galleriaan. Gallerianäkymässä käyttäjä voi tarkastella, muokata ja poistaa tuotteita, sekä lähettää lomakkeen ja tuotetiedot käsiteltäväksi. Dialogi- ja vahvistusnäkymät tukevat näitä toimintoja. Lopuksi tietojen lähettäminen ohjaa käyttäjän Onnistumissivulle, josta siirrytään Lopetussivulle.

Ehdollinen muotoilu ja tietojen validointi

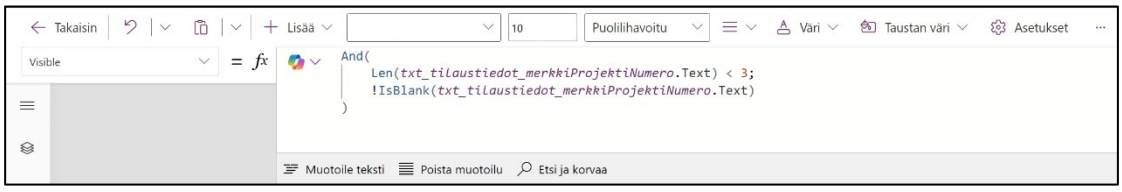
Ehdollinen muotoilu ja dynaamiset näkymät suunniteltiin erityisesti käyttäjien tarpeet huomioon ottaen. Tämä lähestymistapa poistaa tarpeettomat kentät ja toiminnot näkyvistä, tehden käyttöliittymästä intuitiivisen ja selkeän. Esimerkiksi Kommentit-kentän muokattavuuden määritys toteutettiin lisäämällä Kuva

6 näkyvä ehtolause sen DisplayMode-ominaisuuteen. Ehtolause varmistaa, että kenttä on muokattavissa vain, jos MerkkiProjektiNumero- tai Käyttökohde-kenttä ei ole tyhjä, eikä minkään pakollisen kentän virheviesti ole näkyvissä.



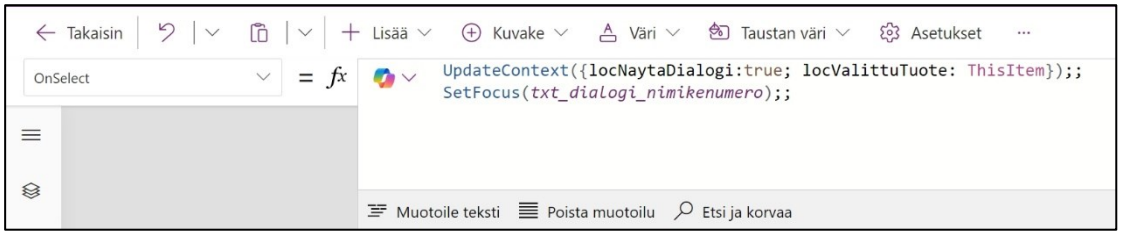
Kuva 6. Kommentit-kentän DisplayMode-ominaisuuden ehdollinen muotoilu

Visible-ominaisuuden avulla ohjausobjektien näkyvyyttä hallitaan hyvin samankaltaisesti, jotta vain olennaiset tiedot ja toiminnot ovat käyttäjille näkyviä. Tätä hyödynnettiin muun muassa lomakekenttien validoinnissa, jossa esimerkiksi MerkkiProjektiNumero-kentässä virheviesti aktivoituu vain, jos syöte on alle kolme merkkiä pitkä eikä kenttä ole tyhjä (Kuva 7). Näin käyttäjää opastetaan korjaaman virheet ennen etenemistä, mitä parantaa syötettyjen tietojen laatua.



Kuva 7. MerkkiProjektiNumero-kentän validointi

Visible-ominaisuuden ehdollista muotoilua hyödynnettiin virheviestien lisäksi erilaisten ohjausobjektien, etenkin säilöjen, näkyvyyden hallinnassa. Esimerkiksi Lomakesivun perustiedot -lomake, dialogi ja vahvistusikkuna on asetettu näkymään paikallisten muuttujien arvojen perusteella. Sovellus käyttää useita paikallisia ja globaaleja muuttujia, joiden avulla hallitaan eri ohjausobjektien näkyvyyttä ja välitetään tietoa ohjausobjektien ja näyttöjen välillä. Esimerkiksi gallerian muokkaa- kuvake hyödyntää paikallista locValittuTuote-muuttujaa, jonka arvoksi tulee gallerian valittu tuote ja sen tiedot (Kuva 8).



Kuva 8. locValittuTuote-muuttujan arvon asettaminen muokkaa -kuvakkeen OnSelect-ominaisuudessa

Muuttujan locValittuTuote arvoksi asetetaan gallerian ThisItem-olio. Se sisältää kaikki valitun tietueen kentät, ja sen avulla voidaan viitata gallerian tietyn tuotteen arvoihin. Esimerkiksi ThisItem.Name viittaisi Name-kenttään. Kun ThisItem asetetaan muuttujan arvoksi, voidaan näihin arvoihin viitata vastaavasti locValittuTuote.Name viittauksella.

Oletussisällön asettaminen

Lomakkeen kentille voidaan määrittää oletusarvoinen sisältö, joka täyttää kentän automaattisesti, jos käyttäjä jättää sen tyhjäksi. Tämä ominaisuus on tärkeä, sillä se varmistaa, että kentissä on aina järkevä arvo, mikä vähentää käyttäjän virheiden mahdollisuutta ja nopeuttaa lomakkeen täyttämistä. Oletusarvojen käyttö on erityisen hyödyllistä silloin, kun tietyn kentän täyttäminen on kriittistä prosessin etenemiselle, mutta sen täyttäminen voi helposti unohtua tai jäädä huomaamatta käyttäjältä. Tätä hyödynnettiin esimerkiksi perustietolomakkeen päivämäärävalitsimessa. Lisäksi oletusarvon avulla mahdollistui jo lisättyjen tuotteiden muokkaaminen (Kuva 9).

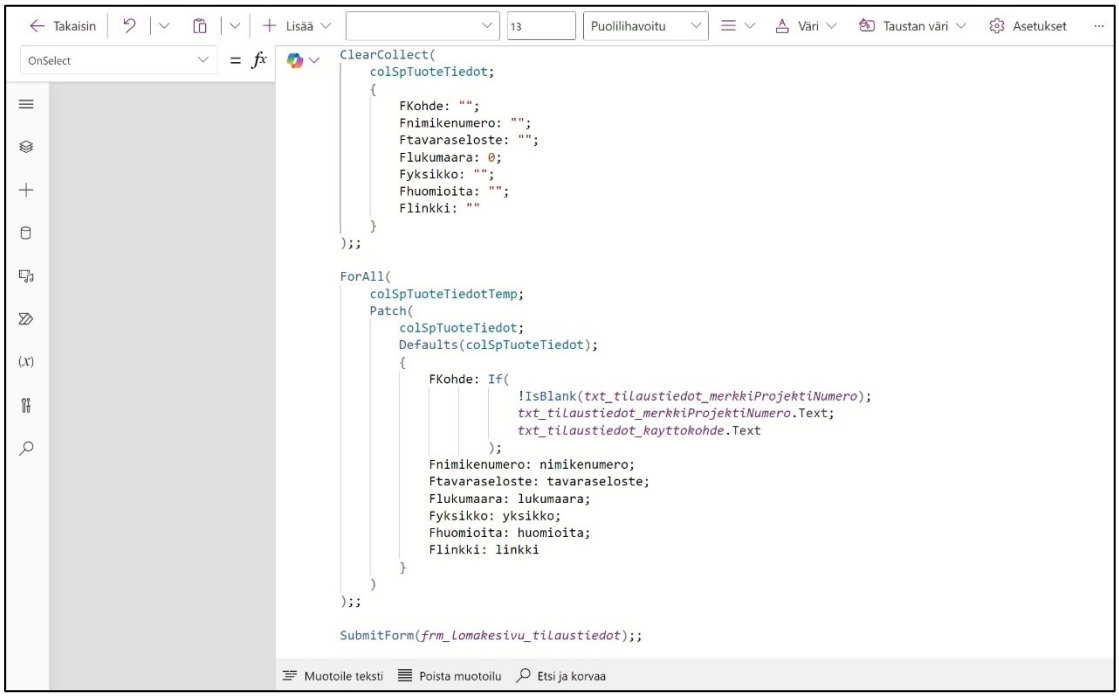
Kuva 9. Default-arvon asettaminen locValittuTuote-muuttujan avulla

Oletusarvon ja paikallisen locValittuTuote-muuttujan avulla käyttäjän aiemmin syöttämät tiedot voitiin asettaa dialogin tekstikenttiin automaattisesti. Tämä tarkoittaa, että muokattavan tuotteen tiedot tuodaan dialogin kenttiin, kuten nimikenumero-kenttään, jolloin käyttäjä voi nopeasti tarkistaa ja päivittää tiedot. Tämä toiminto nopeuttaa lomakkeiden käsittelyä huomattavasti.

Tietojen tallennus ja virheiden hallinta

Kun lomakesivun toiminnallisuus ja ohjausobjektien ehdolliset muotoilut oli määritelty, siirryttiin tietojen tallennukseen ja virhetilanteiden hallintaan. Sovellusistunnon aikana käyttäjän syöttämät tuotetiedot tallentuvat kokoelmaan colSpTuoteTiedotTemp, jonka tiedot näytetään käyttäjälle lomakesivun galleriassa. Näin annetaan käyttäjälle mahdollisuus tarkistaa ja muokata lisäämiään tietoja ja varmistetaan, että tiedot ovat oikein ennen niiden tallennusta lopulliseen tietovarastoon.

Aloitteen perustietojen ja tuotetietojen tallennus SharePoint-luetteloihin tapahtuu painamalla Lähetä -painiketta. Lähetä -painikkeen onSelect-ominaisuudessa (Kuva 10) alustetaan ja täytetään colSpTuoteTiedotTemp kaltainen kokoelma colSpTuoteTiedot, joka luotiin estämään virheellisen datan siirtyminen SharePoint-luetteloihin, jos perustietoja muokataan kesken tuotteiden lisäyksen. Uuteen kokoelmaan poimitaan lisäksi FKohde-sarakkeeseen perustiedoista ensisijaisesti MerkkiProjektiNumero-kentän arvo, tai sen puuttuessa käyttökohde-kentän arvo. Tämä kopioitu kokoelma vastaa nyt paremmin tietolähteenä toimivan Hankinta Tuotteet -luettelon rakennetta. Lopuksi lomakkeen tiedot lähetetään SubmitFrom -funktiolla SharePoint-luetteloon.



Kuva 10. Lähetä-painikkeen onSelect-ominaisuus: Tietojen tallennus kokoelmaan ja lomakkeen lähetys

SubmitForm-funktio välittää lomakkeen sisältämät tiedot Hankinta Aloitteet -luetteloon. Tuotetietojen tallennus Hankinta Tuotteet -luetteloon tapahtuu lomakkeen onSuccess-ominaisuudessa (liite 8), jossa ForAll-funktio iteroi colSpTuoteTiedot-kokoelman läpi tietue kerrallaan ja tallentaa ne yksitellen SharePoint-luetteloon Patch-funktion avulla. Tämän jälkeen tarkistetaan, että kaikki tuotetiedot on tallennettu SharePoint-luetteloon ilman virheitä.

Koska data validoidaan jo syöttövaiheessa, virheellisen datan aiheuttamia virheitä ei pitäisi ilmetä. Mahdollisia virheitä voivat kuitenkin aiheuttaa yhteysongelmat, jotka estävät tietojen onnistuneen tallentamisen. Jos tällaisia virheitä ilmenee, Hankinta Aloite -luetteloon juuri lisätty tietue poistetaan, ja sen mukana poistetaan RefNo-kentän johdannaispoiston myötä myös mahdollisesti osittain tallentuneet tuotetiedot Hankinta Tuotteet -luettelosta. Näin varmistetaan, ettei luetteloihin jää tyhjiä tai puutteellisia tietoja, jotka rikkoisivat tietojen eheyden. Poiston jälkeen käyttäjälle näytetään virheviesti epäonnistuneesta tietojen lähettämisestä (liite 3) ja ohjataan lopetussivulle, jossa neuvotaan täyttämään aloitteen tiedot uudelleen tai sulkemaan sovellus.

4.3 Logiikkakerros ja Power Automate

Logiikkakerros on ratkaiseva osa sovelluksen arkkitehtuuria, sillä se hallitsee sekä sovelluksen toiminnallisuuksia että liiketoimintasääntöjen toteuttamista. Tässä projektissa logiikkakerros oli erityisen tärkeä, koska sen avulla varmistettiin tiedon käsittelyn tarkkuus ja toistuvien prosessien automaatio. Logiikkakerros jakautui kolmeen keskeiseen osa-alueeseen: validointilogiikka, dynaaminen käyttäytyminen ja Power Automate -työnkulut.

Validointilogiikka tarkistaa käyttäjän syöttämät tiedot ennen niiden tallentamista tietolähteeseen. Esimerkiksi, jos käyttäjä täyttää lomakkeen puutteellisesti, järjestelmä estää lomakkeen lähetyksen ja antaa selkeän virheilmoituksen. Tämä vähentää merkittävästi virheellisen tiedon päätymistä tietokantoihin, mikä on erityisen tärkeää hankinta-aloitteiden kaltaisissa prosesseissa, joissa virheelliset tiedot voivat johtaa viivästyksiin ja lisäkustannuksiin

Logiikkakerros vastaa myös sovelluksen dynaamisesta käyttäytymisestä, mikä tarkoittaa, että käyttöliittymä mukautuu käyttäjän toimien perusteella. Esimerkiksi, kun käyttäjä aikoo poistaa lisäämänsä tuotteen listalta, näytölle avautuu vahvistusikkuna, joka estää tahattoman poistamisen. Muut sovelluksen elementit de-aktivoituvat, jolloin käyttäjän huomio kiinnittyy vain kyseiseen toimintaan. Tämä varmistaa, että käyttäjä voi toimia sovelluksessa turvallisesti ja ilman pelkoa virheistä.

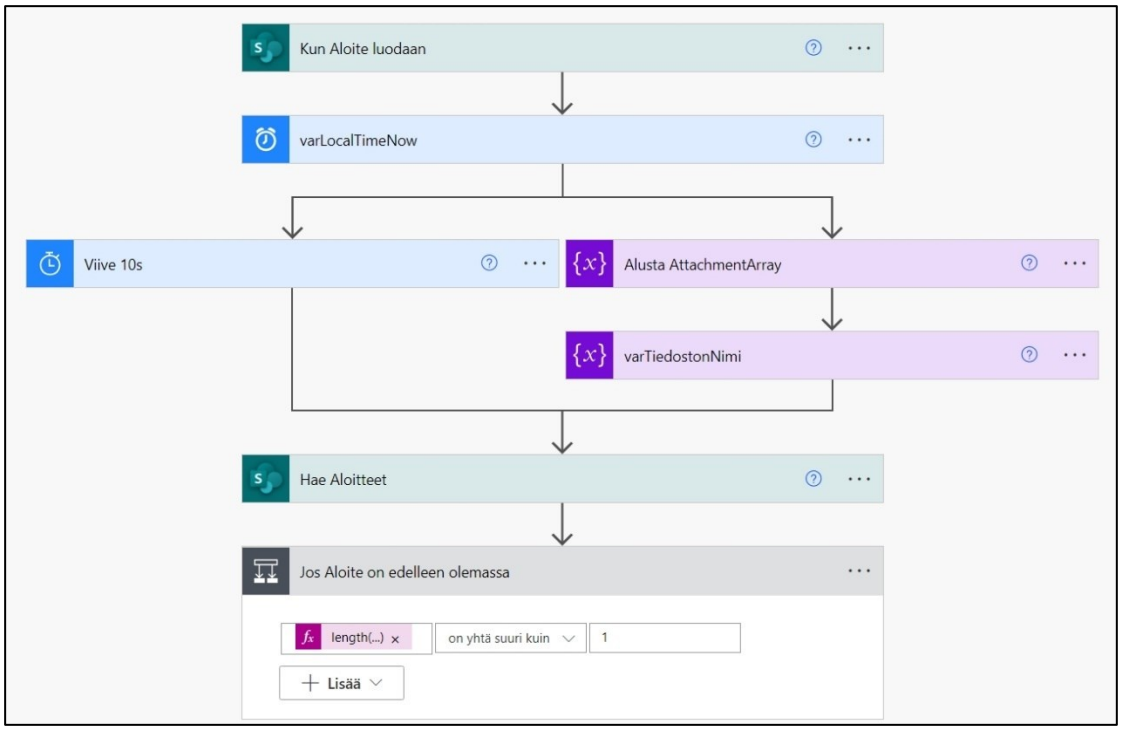
Projektissa hyödynnettiin Power Automate -työnkulkuja, jotka mahdollistavat hankinta-aloitteiden käsittelyn automatisoinnin ja tehostavat viestintää ilman merkittävää manuaalista työtä. Esimerkiksi aloitteen vahvistusviesti lähtee automaattisesti aloitteen tekijälle, mikä parantaa prosessin läpinäkyvyyttä ja vähentää inhimillisten virheiden riskiä. Näiden työnkulkujen ansiosta hankinta-aloitteet käsitellään nopeammin, ja tiedot siirtyvät saumattomasti osastojen välillä.

Työnkulut vähentävät huomattavasti käsittelijöiden manuaalista työtä, mikä vapauttaa aikaa tärkeämpiin tehtäviin. Lisäksi ne on suunniteltu joustaviksi, joten niitä voidaan helposti mukauttaa tai laajentaa yrityksen muuttuviin tarpeisiin ilman suuria uudelleenkoodausprojekteja. Tämä takaa, että järjestelmä tukee yrityksen liiketoimintaprosesseja tehokkaasti nyt ja tulevaisuudessa, tarjoten samalla käyttäjille luotettavan ja helppokäyttöisen työkalun päivittäiseen työhön.

Aloitteen vahvistusviesti -työnkulku

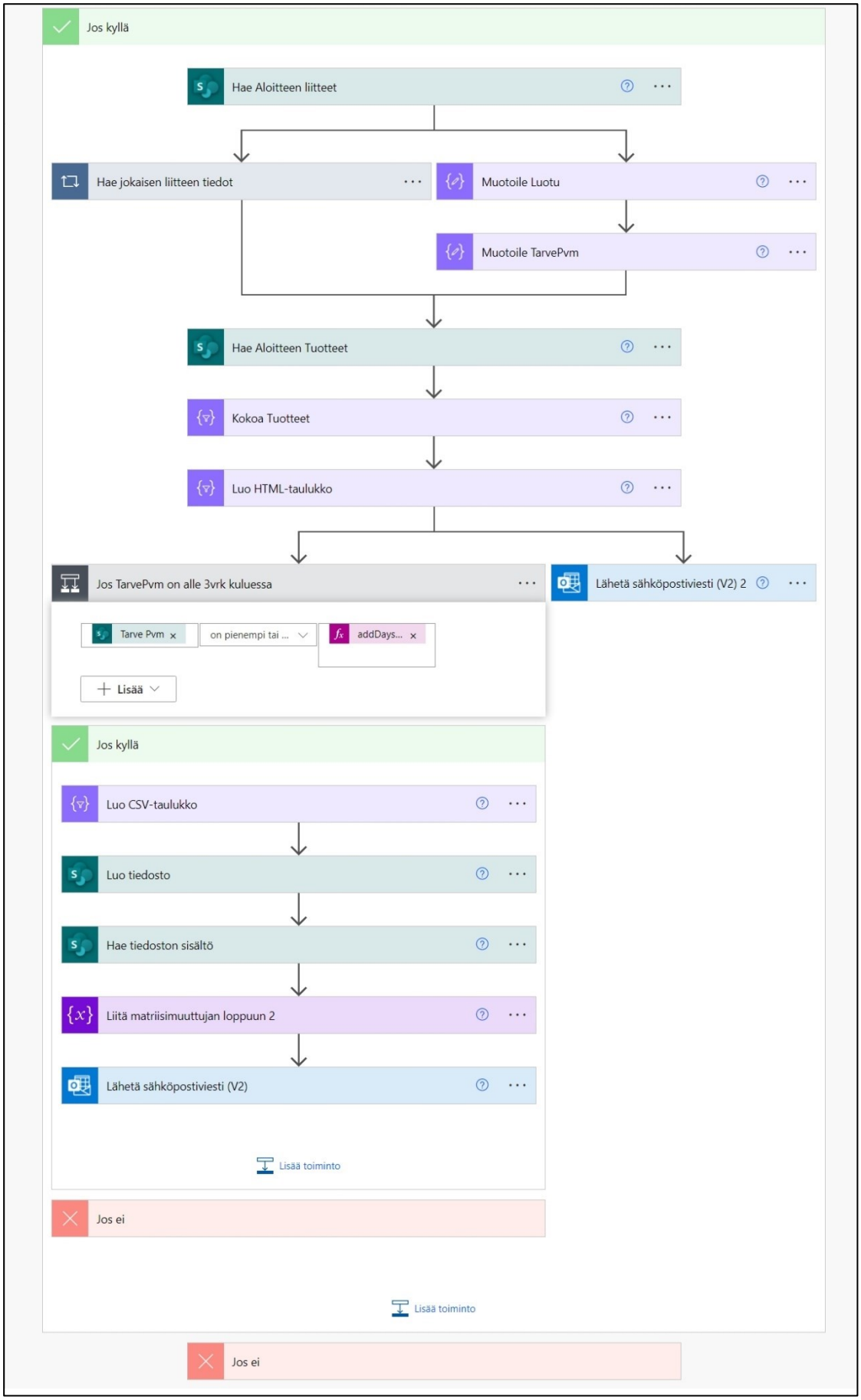
Työnkulun tehtävänä on lähettää aloitteen tekijälle vahvistusviesti onnistuneen aloitteen tallentamisen jälkeen sekä lähettää tiedot liitteineen käsittelijöille, jos aloitteen tarvepäivämäärä on kolmen vuorokauden kuluessa aloitteen luomisesta. Työnkulku luo käsittelijöille myös csv-tiedoston, jonka avulla tuotetiedot voidaan viedä tilausohjelmaan. Hankinta App -sovelluksen toiminnallisuuden vuoksi aloitteen perustiedot saatetaan poistaa luettelosta pian niiden tallentamisen jälkeen. Tämän vuoksi työnkulun tulee tarkistaa, että tietue on edelleen olemassa myös hetki työnkulun käynnistymisen jälkeen. Työnkulun vaiheet esitetään kuvissa Kuva 11 ja Kuva 12.

Työnkulku käynnistyy automaattisesti, kun Hankinta aloite-luetteloon luodaan uusi tietue. Tämän jälkeen määritetään tarvittavat muuttujat ja odotetaan viiveen ajan luettelon mahdollista päivittymistä. Odotuksen jälkeen työnkulku hakee aloitteen tiedot käyttämällä ODATA-suodatuskyselyä, jossa hakuparametreiksi on asetettu tietueen ID, joka vastaa työnkulun käynnistäneen tietueen ID:ä. Jos ehtolause osoittautuu todeksi ja aloite on edelleen olemassa, työnkulku jatkuu Jos Kyllä -lohkossa, muuten työnkulku päättyy.



Kuva 11. Aloitteen vahvistusviesti-työnkulun vaiheet käynnistymisestä Ehto-toimintoon.

Seuraavaksi työnkulku hakee aloitteen liitteet (Kuva 12). Koska liitteitä voi olla useita, käytetään Käytä jokaiseen -toimintoa, joka hakee liitteiden sisällön ja liittää ne AttachmentArray-matriisimuuttujan loppuun. Tämä vaihe toistetaan, kunnes kaikki liitteet on käsitelty. Samalla päivämäärät muotoillaan käyttäjäystävälliseen muotoon. Tämän jälkeen työnkulku hakee Hankinta Tuotteet-luettelosta ne tuotteet, joiden AloiteRefNo-arvo vastaa työnkulun käynnistäneen aloitteen RefNo-arvoa. Haettavat tiedot määritetään avain-arvo-pareina, ja lopuksi niistä luodaan HTML-taulukko, joka kokoaa aloitteen tiedot yhteen. Kun HTML-taulukko on luotu, työnkulku lähettää aloitteen tekijälle vahvistusviestin ja tarkistaa, onko aloitteeseen liittyvä tarvepäivämäärä alle kolmen vuorokauden päässä.



Kuva 12. Aloitteen vahvistusviesti -työnkulun loput vaiheet.

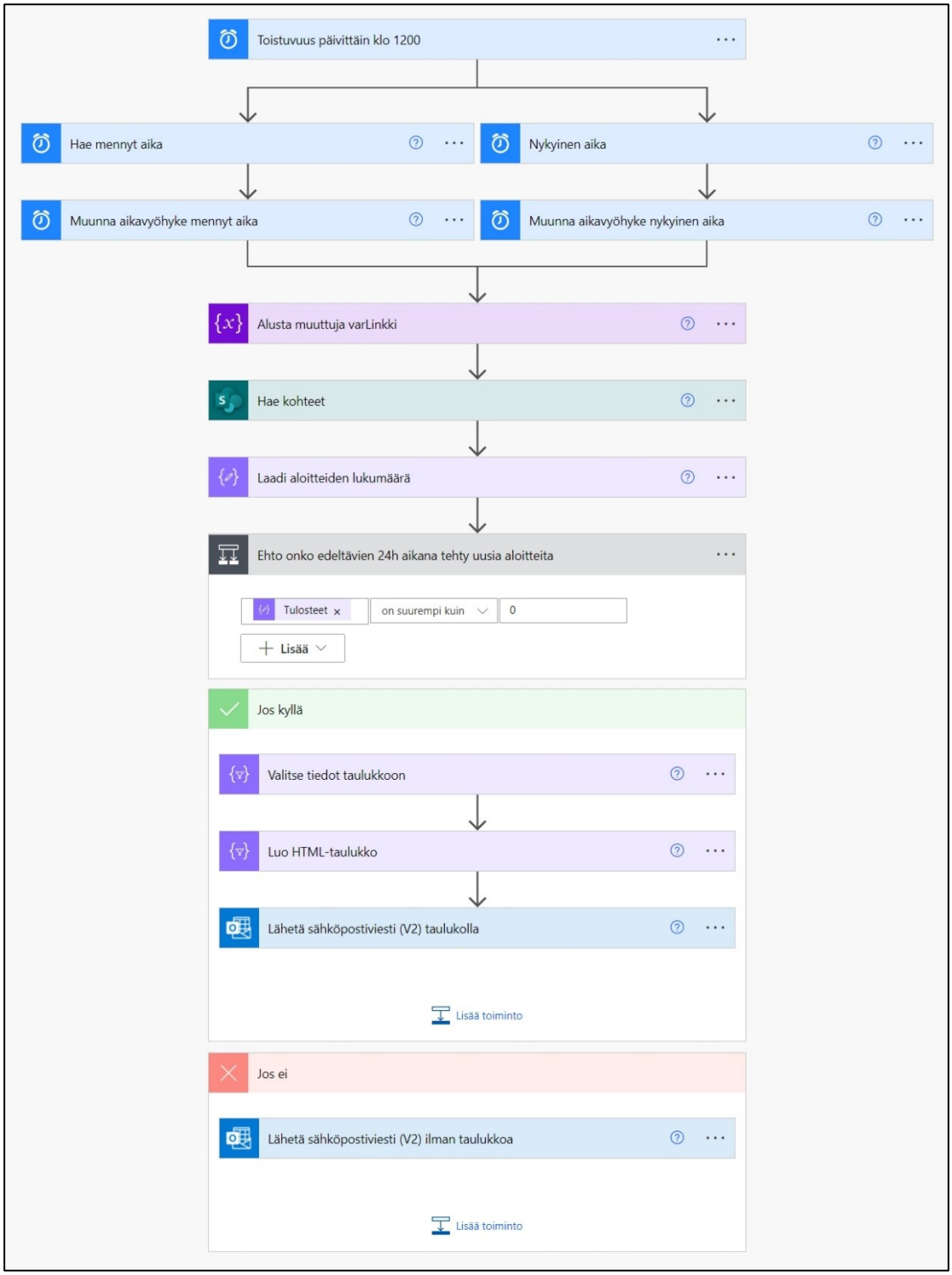
Jos näin on, työnkulku jatkuu Jos kyllä -lohkossa, muuten se päättyy. Mikäli työnkulku jatkuu se luo csv-tiedoston Kokoa Tuotteet -toiminnon tuloksen perusteella ja liittää sen AttachmentArray -matriisimuuttujan loppuun. Lopuksi työnkulku lähettää käsittelijöille sähköpostin, jossa on aiemmin luotu taulukko aloitteen tiedoista sekä liitteenä aloitteen liitetiedostot ja juuri luotu csv-tiedosto.

Aloitteiden päivittäinen koontiviesti -työnkulku

Aloitteiden päivittäinen koontiviesti -työnkulun (Kuva 13) tarkoituksena on kerätä päivittäin edeltävän 24 tunnin aikana luodut aloitteet ja lähettää niistä koostesähköposti käsittelijöille. Työnkulku ei ota huomioon sitä, jos aloitteesta on jo lähetetty sähköpostiviesti käsittelijöille toisen työnkulun kautta. Aloitteiden päivittäinen koontiviestityönkulku varmistaa, että käsittelijät saavat ajantasaisen tiedon uusista aloitteista, mikä mahdollistaa tehokkaan seurannan ja raportoinnin.

Työnkulku käynnistyy automaattisesti päivittäin klo 12:00. Ensimmäiseksi työnkulku hakee aikaleimat nykyiselle ajalle sekä ajalle 24 tuntia sitten ja muuttaa ne Suomen aikavyöhykkeelle. Tämän jälkeen työnkulku alustaa muuttujan, joka toimii linkkinä Hankinta Aloitteet -luetteloon. Seuraavaksi työnkulku hakee suodatuskyselyn avulla Hankinta Aloitteet -luettelosta ne kohteet, jotka on luotu menneen ja nykyisen ajan aikaleiman välisenä aikana. Hakutulosten perusteella työnkulku laskee aloitteiden lukumäärän. Työnkulku jatkuu ehdollisesti aloitteiden lukumäärän perusteella.

Jos aloitteita on enemmän kuin 0, työnkulku valitsee taulukkoon haettavat tiedot, luo niistä HTML-taulukon ja lähettää käsittelijöille sähköpostiviestin, joka sisältää taulukon edeltävän 24 tunnin aikana luoduista aloitteista sekä linkin Hankinta Aloitteet -luetteloon. Viestin otsikossa mainitaan uusien aloitteiden määrä. Jos uusia aloitteita ei ole, työnkulku lähettää käsittelijöille sähköpostiviestin, joka ilmoittaa, ettei uusia aloitteita ole, mutta sisältää linkin Hankinta Aloitteet -luetteloon mahdollisten aikaisempien aloitteiden tarkastelua varten.



Kuva 13. Aloitteiden päivittäinen koontiviesti -työnkulun vaiheet

Nämä Power Automate -työnkulut paransivat sovelluksen tehokkuutta ja luotettavuutta vähentämällä manuaalista työtä. Ne varmistavat, että hankinta-aloitteet käsitellään ajallaan ja oikein, mikä tukee yrityksen tavoitteita entistä paremmin. Lisäksi automatisointi vähentää inhimillisten virheiden riskiä, mikä osaltaan parantaa prosessien laatua ja luotettavuutta pitkällä aikavälillä.

5 TESTAUS, KÄYTTÖÖNOTTO JA YLLÄPITO

Tässä projektissa testaus toteutettiin pääasiassa manuaalisesti, lukuun ottamatta muutamia Power Apps Test -ympäristön testiautomaatioita. Testaustoimien tarkoituksena on varmistaa, että sovellus täyttää kaikki sille asetetut vaatimuksen ja toimii moitteettomasti eri käyttöympäristöissä. Yksikkötestauksessa keskityttiin yksittäisten komponenttien testaamiseen. Power Apps -sovelluksessa yksikkötestaus kohdistui erityisesti lomakekenttien validointiin ja painikkeiden toimintoihin. Esimerkiksi validointia testattiin syöttämällä eri tyyppisiä virheellisiä tietoja, kuten tyhjiä kenttiä ja virheellisiä syöteformaatteja. Tämän avulla voitiin varmistaa, että sovellus ilmoittaa käyttäjälle selkeästi syötteen virheistä ja estää väärän tiedon tallentamisen. Integraatiotestauksessa testattiin erityisesti SharePoint-luetteloiden ja Power Automate-työnkulkujen välistä yhteistyötä. Power Automate-työnkuluissa testattiin, että ne aktivoituvat odotetusti, ja käsittelevät tiedot ja lähettävät sähköpostit virheettömästi.

Testauksen ja validoinnin jälkeen sovellus siirrettiin tuotantoympäristöön. Projektille luotiin oma SharePoint sivusto, johon sijoitettiin sovelluksen tietolähteenä toimivat SharePoint-luettelot. Tämä sivusto toimii samalla myös käsittelijöiden uutena työympäristönä, jossa he voivat käsitellä saapuneita aloitteita. Power Apps -sovellus tuotiin käyttäjien saataville luomalla linkki yrityksen intranet-sivustolle. Käyttöönottoa varten laadittiin kattavat ohjeet, jotka toimitettiin kaikkien käyttäjien saataville yrityksen intranettiin. Käyttäjäohjeet sisältävät vaiheittaiset kuvaukset sovelluksen tärkeimmistä toiminnoista, kuten aloitteen luomisesta, tietojen syöttämisestä ja tallennuksesta.

Ylläpidon alkaessa toiminta keskittyi seuraamaan sovelluksen toimintaa, keräämään käyttäjäpalautetta ja reagoimaan nopeasti mahdollisiin ongelmiin. Koska sovelluksen laajamittainen käyttöönotto on vasta alkamassa, palautetta on toistaiseksi saatu rajoitetusti. Palautteen vähyys ei kuitenkaan estä ylläpidon tavoitteena olevaa sovelluksen jatkuvaa kehittämistä.

6 POHDINTA JA JOHTOPÄÄTÖKSET

Power Platform osoittautui kokonaisvaltaiseksi työkalupaketiksi, joka mahdollisti sovelluksen nopean kehityksen. Sen avulla koodaustarvetta pystyttiin merkittävästi vähentämään, jolloin kehitystyössä pystyttiin keskittymään enemmän

liiketoimintaprosessien automatisointiin ja optimointiin. Power Apps ja Power Automate osoittautuivat tehokkaiksi työkaluiksi sovelluksen eri osa-alueiden toteuttamisessa.

Tietokerroksen ratkaisut, kuten SharePoint-luetteloiden käyttäminen tietovarastona ja yksi-moneen-suhteisten tietorakenteiden hyödyntäminen, paransivat merkittävästi sovelluksen kykyä käsitellä suuria tietomääriä ja taata tiedon eheys. Hankinta Aloitteet -luettelo ja siihen liittyvä Hankinta Tuotteet -luettelo mahdollistivat tietojen loogisen jäsentelyn ja tehokkaan haun, mikä oli keskeinen tekijä hankintaprosessien optimoinnissa. Näiden ratkaisujen ansiosta tietojen hallinta selkeytyi ja virheiden määrä saatiin minimoitua, mikä on yksi projektin tärkeimpiä onnistumisia.

Käyttöliittymän suunnittelussa käytetyt menetelmät, kuten ehdollinen muotoilu, muuttujien ja kokoelmien hallinta, sekä tiedon validointi, paransivat merkittävästi sovelluksen käytettävyyttä ja soveltuvuutta hankintaprosessien ratkaisuksi. Käyttäjäpalautteen perusteella tehdyt muutokset osoittautuivat erittäin hyödyllisiksi ja paransivat sovelluksen käytettävyyttä huomattavasti. Näiden ratkaisuiden ansiosta käyttäjäystävällisyys, prosessin nopeus ja virheiden vähentäminen toteutuivat odotetusti.

Logiikkakerroksen ratkaisut, kuten validointilogiikan tarkka toteutus, dynaaminen käyttäytyminen ja automaattiset työnkulut, paransivat sovelluksen toimintavarmuutta ja vähensivät manuaalisen työn määrää. Esimerkiksi virheellisen tiedon syöttämisen estäminen ja automaattisten vahvistusviestien lähettäminen vähensivät virheiden määrää ja paransivat tiedon käsittelyn tarkkuutta. Näiden ratkaisujen ansiosta sovelluksen luotettavuus lisääntyi merkittävästi, mikä tukee yrityksen liiketoimintatavoitteita.

Onnistumiset ja haasteet

Projektin aikana saavutettiin merkittäviä onnistumisia, erityisesti tietojen hallinnan ja prosessien automatisoinnin osalta. SharePoint-luetteloiden ja Power Platformin hyödyntäminen osoittautui tehokkaaksi ratkaisuksi, ja ne mahdollistivat tiedonhallinnan ja käytettävyyden parantamisen. Sovellus on linjassa T-Drill Oy tarpeiden kanssa.

Haasteita ilmeni kuitenkin erityisesti SharePoint-luetteloiden hallinnassa. Esimerkiksi johdannaispoiston puuttuva varoitus käyttäjille, jotka käsittelevät tietoja suoraan SharePoint-sivustolla tai Lists-sovelluksessa, on merkittävä ongelma, joka vaatii pikaista korjausta. Tietueiden vahingollinen poisto ilman ennakkovaroitusta voi aiheuttaa vakavia ongelmia tiedon eheyden ylläpitämisessä, mikä korostaa tarvetta tiukemmalle käyttöoikeuksien hallinnalle.

Jatkosuositukset ja kehitysehdotukset

Projektin aikana ilmenneiden puutteiden korjaaminen on ensiarvoisen tärkeää sovelluksen toimivuuden kannalta. Ensisijaisena suosituksena on varmistaa, että SharePoint-luetteloihin tehtävät muutokset, kuten tietueiden poistot, varoittavat käyttäjiä mahdollisista johdannaispoistoista. Poisto-oikeudet tulisi rajoittaa tietylle käyttäjäkunnalle, kuten järjestelmänvalvojille tai vastuuhenkilöille, jotta voidaan minimoida vahingossa tapahtuvat tärkeiden tietueiden poistot. Tämä parantaisi järjestelmän turvallisuutta ja tietojen säilyvyyttä.

Jatkokehityksessä voisi keskittyä myös tietokannan suorituskyvyn parantamiseen, esimerkiksi lisäindeksointia hyödyntämällä ja hakualgoritmien optimoinnilla. Käyttöliittymän osalta visuaalisuutta ja navigointia on mahdollista vielä kehittää, jotta sovellus vastaisi entistä paremmin käyttäjien tarpeisiin ja tukisi T-Drill Oy:n liiketoimintatavoitteita.

Lisäautomaation tuominen monimutkaisempien prosessien käsittelyyn voisi myös parantaa sovelluksen käyttökokemusta ja automatisointia. Esimerkiksi kiireellisten aloitteiden käsittelyssä voitaisiin hyödyntää Power Automaten sähköposti valinnoilla -toimintoa, jolloin käsittelijät voisivat hyväksyä aloitteen suoraan sähköpostista. Tämä vähentäisi käsittelijöiden kuormitusta ja nopeuttaisi prosessia merkittävästi. Toisaalta aloitteiden käsittelyn integroiminen suoraan tilausohjelmaan olisi myös suositeltavaa jatkokehityksen kannalta. Tämä vaatisi kuitenkin laajempaa osaamista API-rajapintojen käytössä ja ylläpidossa, mutta voisi merkittävästi parantaa koko prosessin tehokkuutta ja tarkkuutta.

Reflektio

Tämän projektin toteuttaminen on tarjonnut minulle arvokasta oppia sekä teknisten taitojen että projektinhallinnan näkökulmasta. Alusta alkaen oli selvää, että sovelluksen kehittäminen Microsoft Power Platform -ympäristössä vaatii monipuolista osaamista. Projektin aikana syvensin merkittävästi ymmärrystäni käyttämistäni työkaluista ja opin hyödyntämään niitä tehokkaasti hankintaprosessien automatisoinnissa ja optimoinnissa.

Erityisesti tietokerroksen suunnittelu ja toteutus antoivat minulle arvokkaita kokemuksia SharePoint-luetteloiden ja tietorakenteiden hallinnasta. Opin, kuinka tärkeää on varmistaa tiedon eheys ja järjestää tietojen hallinta loogisesti niin, että sovellus pystyy käsittelemään suuria tietomääriä ja suorittamaan monimutkaisia hakuja tehokkaasti. Lisäksi jouduin pohtimaan, miten mahdolliset ongelmat, kuten tiedon poiston hallinta ja tietokannan suorituskyky, voivat vaikuttaa koko sovelluksen toimintaan.

Käyttöliittymän suunnittelun osalta opin, kuinka käyttäjäkokemuksen parantaminen edellyttää jatkuvaa iterointia ja käyttäjäpalautteen huolellista analysointia. Käyttöliittymän ehdollisen muotoilun ja tiedon validoinnin toteuttaminen ei ollut pelkästään tekninen haaste, vaan vaati myös syvällistä ymmärrystä siitä, miten käyttäjät vuorovaikuttavat sovelluksen kanssa. Tämä projekti auttoi minua ymmärtämään, kuinka tärkeää on asettua käyttäjän asemaan ja suunnitella käyttöliittymä siten, että se on intuitiivinen ja vähentää virheiden mahdollisuutta.

Logiikkakerroksen suunnittelussa opin, kuinka monimutkaiset liiketoimintaprosessit voidaan automatisoida Power Automate -työkaluilla, ja kuinka tärkeää on varmistaa, että sovelluksen toiminnallisuudet tukevat liiketoimintatavoitteita. Prosessien automatisoinnin avulla opin vähentämään manuaalisen työn määrää ja lisäämään sovelluksen toimintavarmuutta, mikä oli keskeinen osa projektin onnistumista.

Kokonaisuudessaan tämä projekti on vahvistanut osaamistani sovelluskehityksessä ja erityisesti Microsoftin ekosysteemin työkaluissa. Olen oppinut yhdis-

tämään teknisen osaamisen ja käyttäjäkokemuksen parantamisen tehokkaaksi kokonaisuudeksi, mikä on parantanut kykyäni ratkaista monimutkaisia ongelmia ja toteuttaa käytännönläheisiä ratkaisuja. Tulevaisuudessa hyödynnän näitä oppeja kehittäessäni entistä parempia ja käyttäjäystävällisempiä sovelluksia.

LÄHTEET

Booch, G., Rumbaugh, J., & Jacobson, I. 2005. The Unified Modeling Language User Guide. 2.Painos. New Jersey: Addison-Wesley Professional.

Berardi, V., Kaur, V., Thacker, D., & Blundell, G. 2023. Towards a citizen development andragogy: Low-code platforms, design thinking and knowledge-based dynamic capabilities. *International Journal of Higher Education Management* 2, 1-21. Verkkolehti. Saatavissa: https://doi.org/10.24052/ijhem/v09n02/art-1 [viitattu 27.8.2024].

Bratincevic, J. & Guidice, D. 2023. New Research: Will AI Kill The Low-Code Market. Blogi. Päivitetty 20.9.2023. Saatavissa: https://www.forrester.com/blogs/new-research-will-ai-kill-the-low-code-market/ [viitattu 15.8.2024].

Dooley, J. 2011. Software Development and Professional Practice. New York: Apress.

Forrester. 2020. The Total Economic Impact Of Power Apps. PDF-dokumentti. Saatavissa: https://www.scribd.com/document/523681070/The-Total-Economic-Impact-of-Power-Apps [viitattu 15.8.2024].

Gartner. 2021. Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences. WWW-dokumentti. Päivitetty 10.11.2021. Saatavissa: https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences [viitattu 30.7.2024].

Gartner. 2023. Magic Quadrant for Enterprise Low-Code Application Platforms. Gartner, Inc. Saatavissa: https://www.gartner.com/doc/reprints?id=1-2F7NELJY&ct=231004&st=sb [Viitattu 04.8.2024].

IBM. 2022. Low-Code vs. No-Code: What's the difference? Blogi. Päivitetty 23.5.2022. Saatavissa: https://www.ibm.com/think/topics/low-code-vs-no-code [viitattu 27.8.2024].

Jorgensen, P. & DeVries, B. 2021. Software testing: A Craftsman's Approach, 5. Painos. Lontoo: CRC Press.

Kissflow Team. 2024. Difference Between Low-Code and No-Code Platform. Blogi. Päivitetty 29.7.2024. Saatavissa: https://kissflow.com/low-code/low-code-vs-no-code/ [viitattu 27.8.2024].

Martinez, E. & Pfister, L. 2023. Benefits and limitations of using low-code development to support digitalization in the construction industry. *Automation in Construction* 152 2-17. Verkkolehti. Saatavissa: https://doi.org/10.1016/j.autcon.2023.104909 [viitattu 02.8.2024].

Microsoft. 2023. (2023, February 7). Understand variables in canvas apps - Power Apps. WWW-dokumentti. Päivitetty 07.2.2023. https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/working-with-variables [viitattu 15.8.2024].

Microsoft. 2024a. Microsoft Power Fx overview - Power Platform. WWW-dokumentti. Päivitetty 22.3.2024. Saatavissa: https://learn.microsoft.com/en-us/power-platform/power-fx/overview [viitattu 15.8.2024].

Microsoft. 2024b. Understand Power Apps Studio - Power Apps. WWW-dokumentti. Päivitetty 22.2.2024. Saatavissa: https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/power-apps-studio [viitattu 15.8.2024].

Microsoft. s.a.. Create list relationships by using lookup columns. WWW-dokumentti. Saatavissa: https://support.microsoft.com/en-us/office/create-list-relationships-by-using-lookup-columns-80a3e0a6-8016-41fb-ad09-8bf16d490632 [viitattu 15.8.2024].

Perera, S., & Eadie, R. 2023. Managing Information Technology Projects: Building a Body of Knowledge in IT Project Management. Singapore: World Scientific Publishing Co.

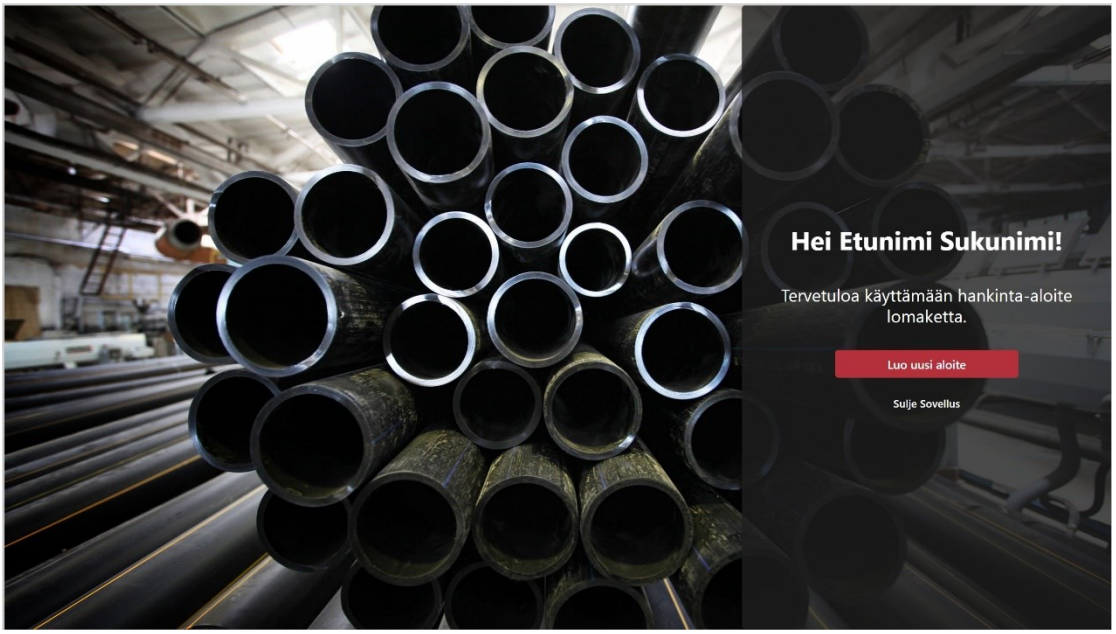
Rokis, K., & Kirikova, M. 2023. Exploring Low-Code Development: A Comprehensive Literature review. Complex Systems Informatics and Modeling Quarterly, 36, 68–86. Saatavissa: https://doi.org/10.7250/csimq.2023-36.04 [viitattu 27.8.2024].

Sommerville, I. 2015. Software Engineering, Global Edition. Harlow: Pearson Education Limited.

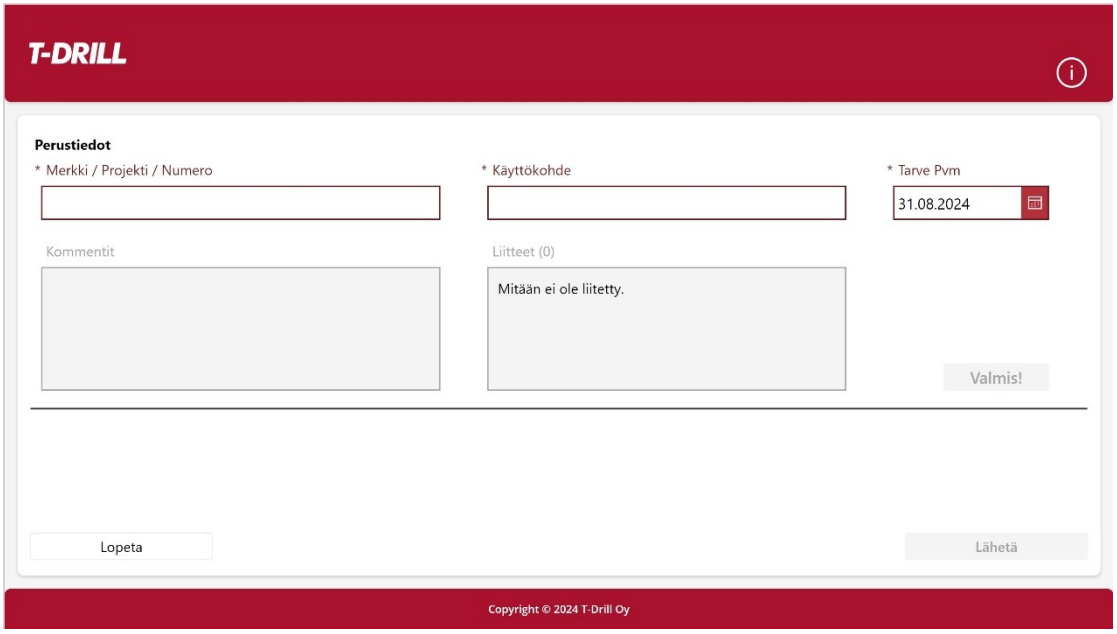
Srinam. 2023. Software development process step by step guide: Requirement, Plan, Design, Develop & Deploy. GeeksforGeeks. WWW-dokumentti. Päivitetty 29.11.2023. Saatavissa: https://www.geeksforgeeks.org/5-steps-of-software-development-process/ [viitattu 6.6.2024].

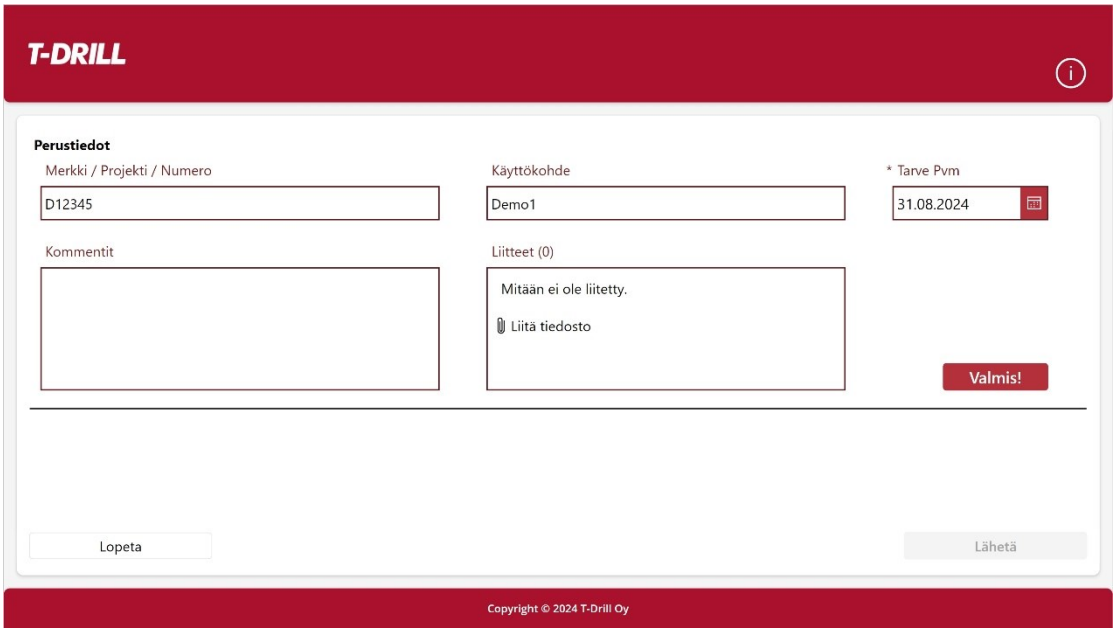
Stellman, A., & Greene, J. 2015. Learning agile: Understanding Scrum, XP, Lean, and Kanban. Sebastopol, CA: O'Reilly Media.

Liite 1

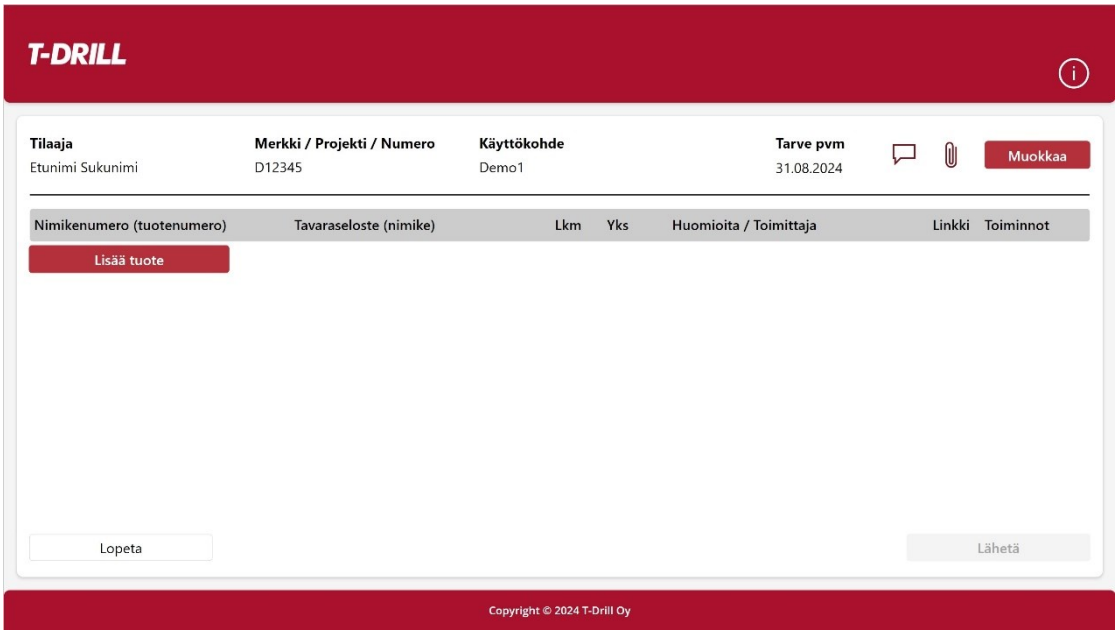

Hei Etunimi Sukunimi!
Tervetuloa käyttämään hankinta-aloite lomaketta.
Luo uusi aloite
Sulje Sovellus

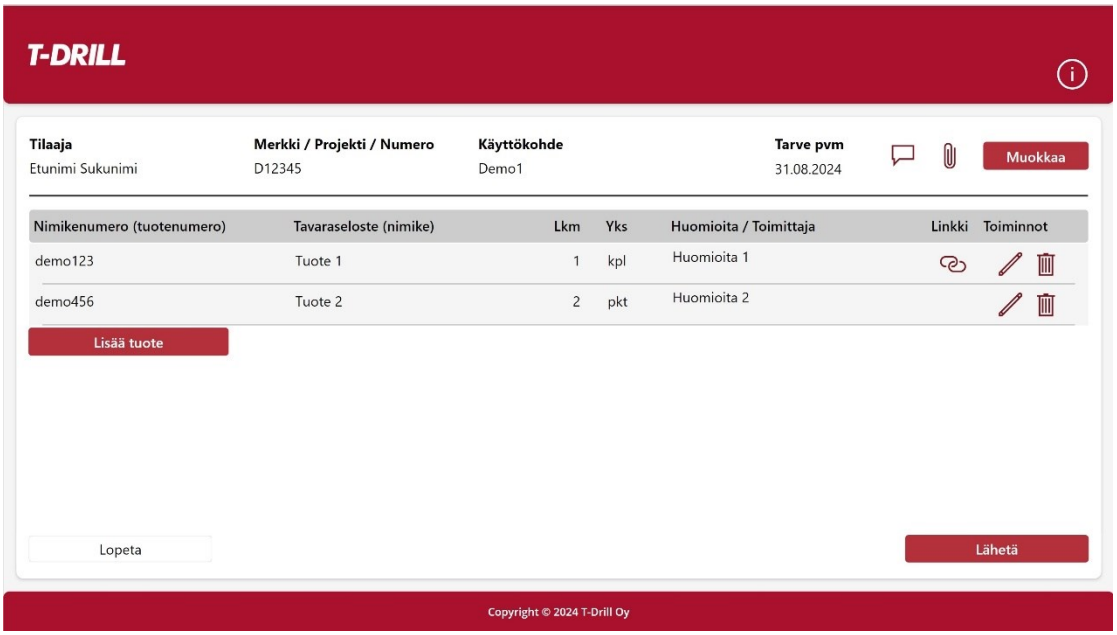
Liite 2/1


T-DRILL
Perustiedot
* Merkki / Projekti / Numero
* Käyttökohde
* Tarve Pvm
31.08.2024
Kommentit
Liitteet (0)
Mitään ei ole liitetty.
Valmis!
Lopeta
Lähetä
Copyright © 2024 T-Drill Oy

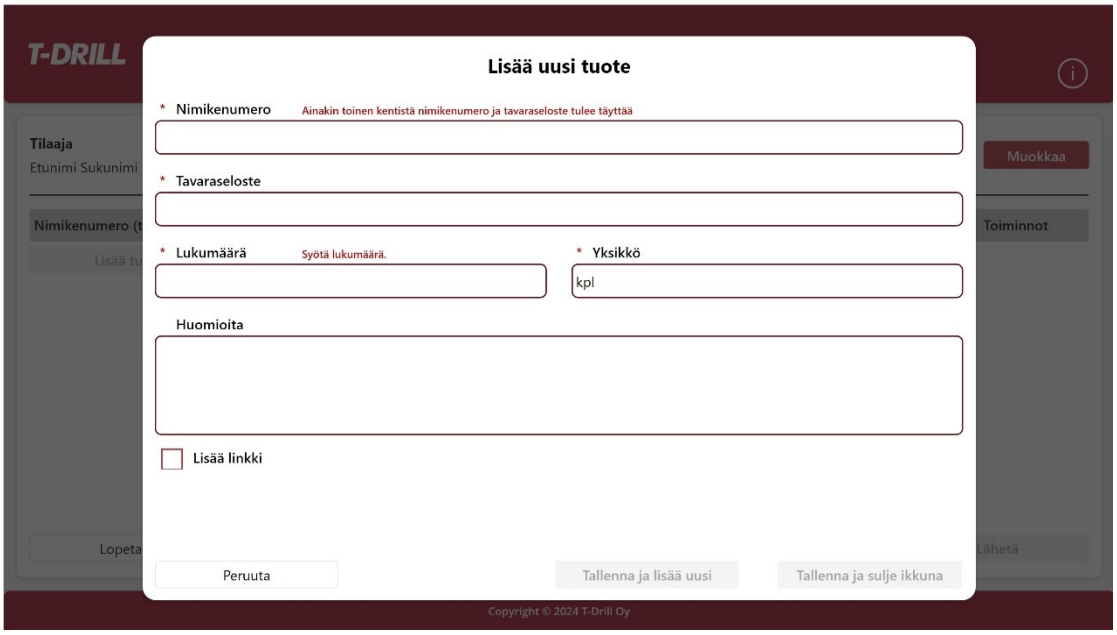

T-DRILL
Perustiedot
Merkki / Projekti / Numero
D12345
Käyttökohde
Demo1
* Tarve Pvm
31.08.2024
Kommentit
Liitteet (0)
Mitään ei ole liitetty.
Liitä tiedosto
Valmis!
Lopeta
Lähetä
Copyright © 2024 T-Drill Oy

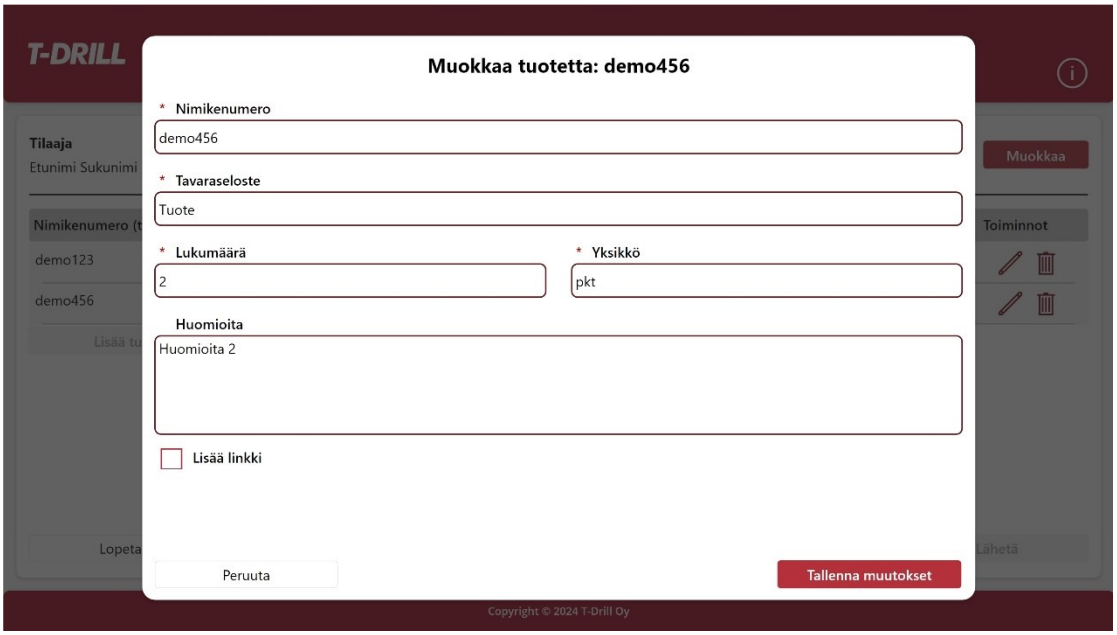
Liite 2/2


T-DRILL
Tilaaja
Etunimi Sukunimi
Merkki / Projekti / Numero
D12345
Käyttökohde
Demo1
Tarve pvm
31.08.2024
Muokkaa
Nimikenumero (tuotenumero)
Tavaraseloste (nimike)
Lkm
Yks
Huomioita / Toimittaja
Linkki
Toiminnot
Lisää tuote
Lopeta
Lähetä
Copyright © 2024 T-Drill Oy

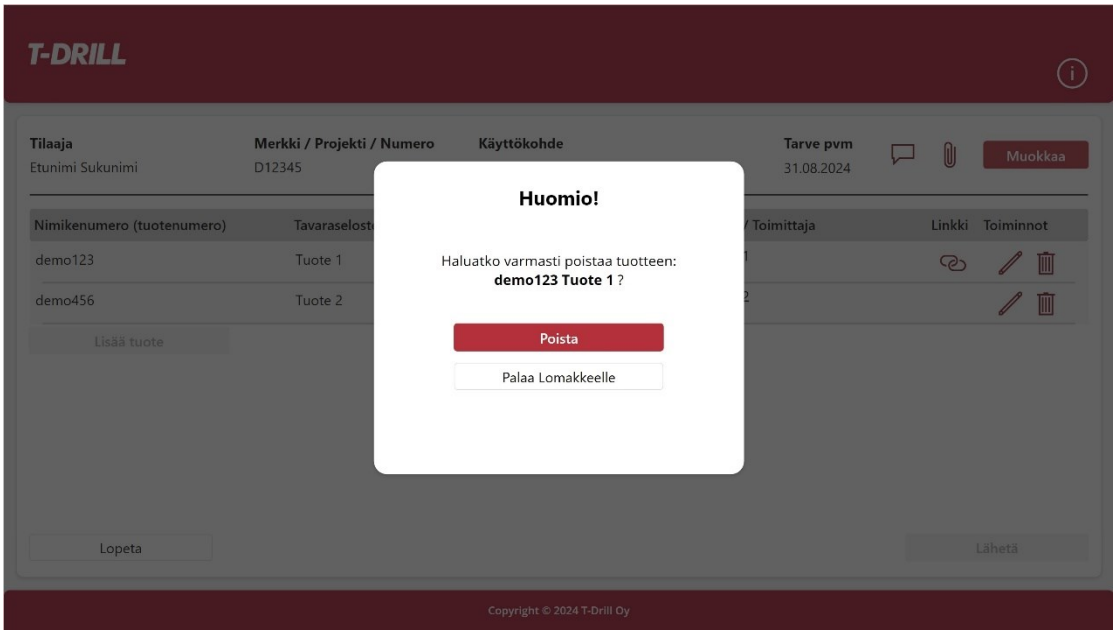

T-DRILL
Tilaaja
Etunimi Sukunimi
Merkki / Projekti / Numero
D12345
Käyttökohde
Demo1
Tarve pvm
31.08.2024
Muokkaa
Nimikenumero (tuotenumero)
Tavaraseloste (nimike)
Lkm
Yks
Huomioita / Toimittaja
Linkki
Toiminnot
demo123
Tuote 1
1
kpl
Huomioita 1
demo456
Tuote 2
2
pkt
Huomioita 2
Lisää tuote
Lopeta
Lähetä
Copyright © 2024 T-Drill Oy

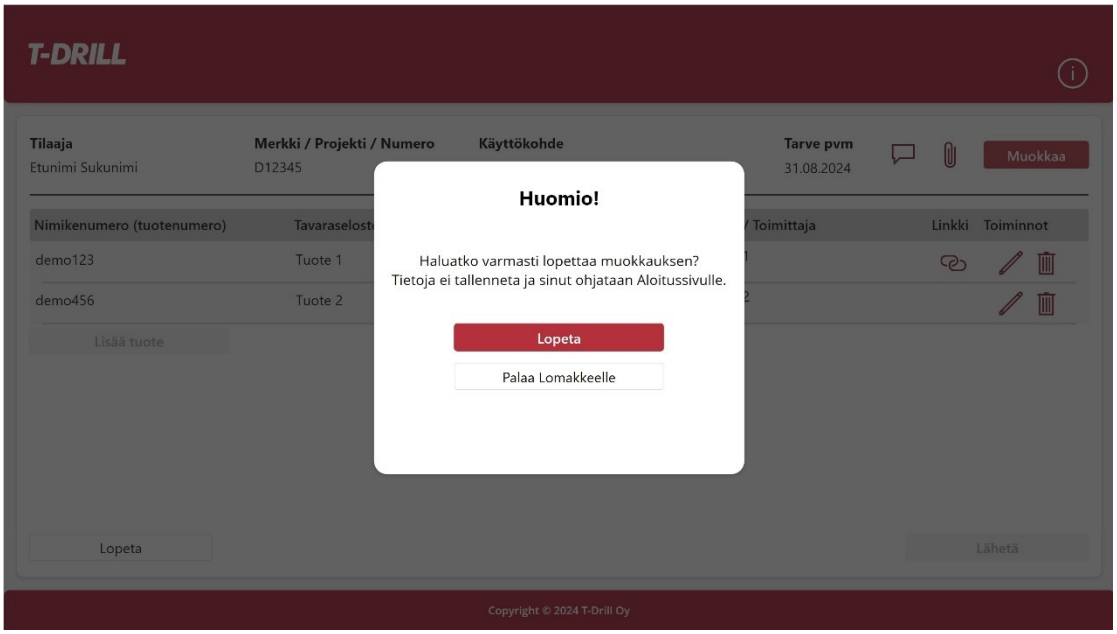
Liite 2/3


T-DRILL
Lisää uusi tuote
* Nimikenumero
Ainakin toinen kentistä nimikenumero ja tavaraseloste tulee täyttää
* Tavaraseloste
* Lukumäärä
Syötä lukumäärä.
* Yksikkö
kpl
Huomioita
Lisää linkki
Peruuta
Tallenna ja lisää uusi
Tallenna ja sulje ikkuna
Copyright © 2024 T-Drill Oy

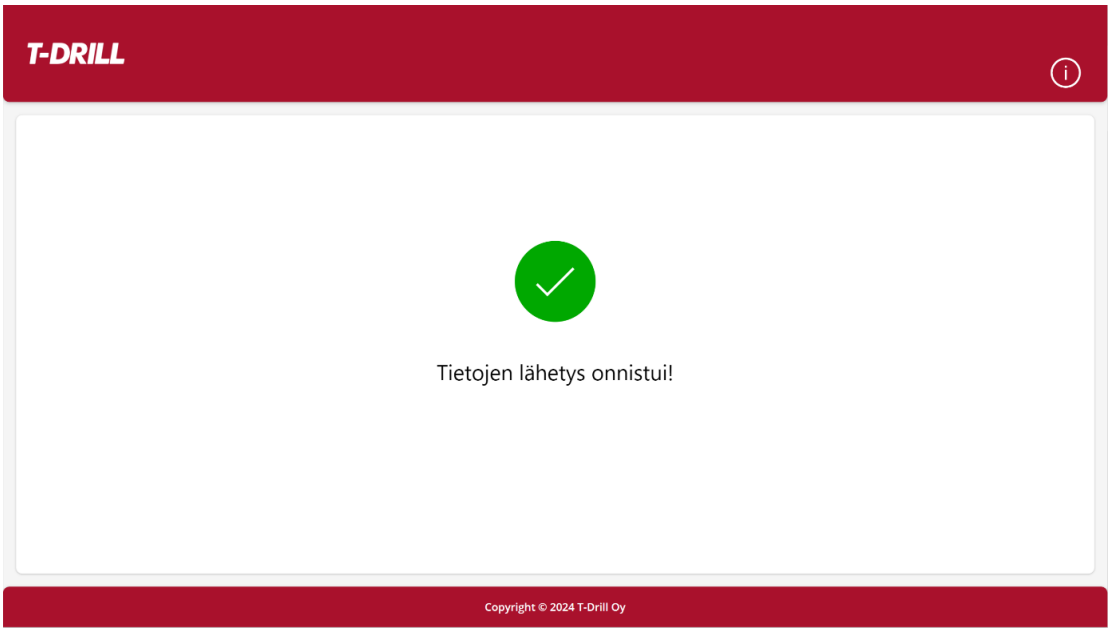

T-DRILL
Muokkaa tuotetta: demo456
* Nimikenumero
demo456
* Tavaraseloste
Tuote
* Lukumäärä
2
* Yksikkö
pkt
Huomioita
Huomioita 2
Lisää linkki
Peruuta
Tallenna muutokset
Copyright © 2024 T-Drill Oy

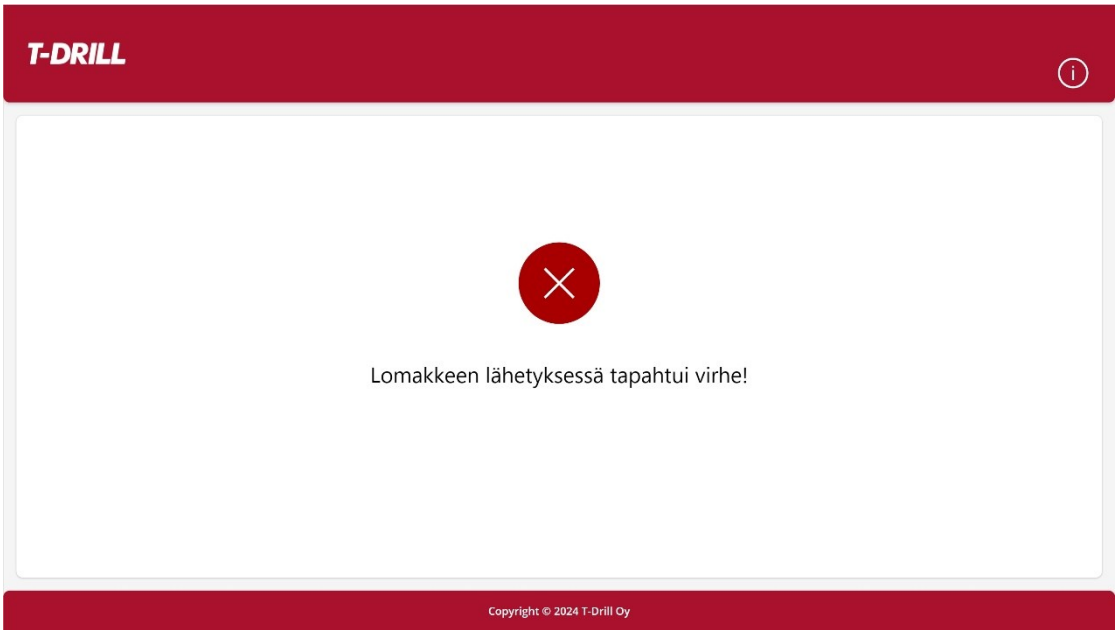
Liite 2/4


T-DRILL
Tilaaja
Etunimi Sukunimi
Merkki / Projekti / Numero
D12345
Käyttökohde
Tarve pvm
31.08.2024
Muokkaa
Nimikenumero (tuotenumero)
Tavaraselost
Toimittaja
Linkki
Toiminnot
demo123
Tuote 1
demo456
Tuote 2
Lisää tuote
Huomio!
Haluatko varmasti poistaa tuotteen:
demo123 Tuote 1 ?
Poista
Palaa Lomakkeelle
Lopeta
Lähetä
Copyright © 2024 T-Drill Oy


T-DRILL
Tilaaja
Etunimi Sukunimi
Merkki / Projekti / Numero
D12345
Käyttökohde
Tarve pvm
31.08.2024
Muokkaa
Nimikenumero (tuotenumero)
Tavaraselost
Toimittaja
Linkki
Toiminnot
demo123
Tuote 1
demo456
Tuote 2
Lisää tuote
Huomio!
Haluatko varmasti lopettaa muokkauksen?
Tietoja ei tallenneta ja sinut ohjataan Aloitussivulle.
Lopeta
Palaa Lomakkeelle
Lopeta
Lähetä
Copyright © 2024 T-Drill Oy

Liite 3/1

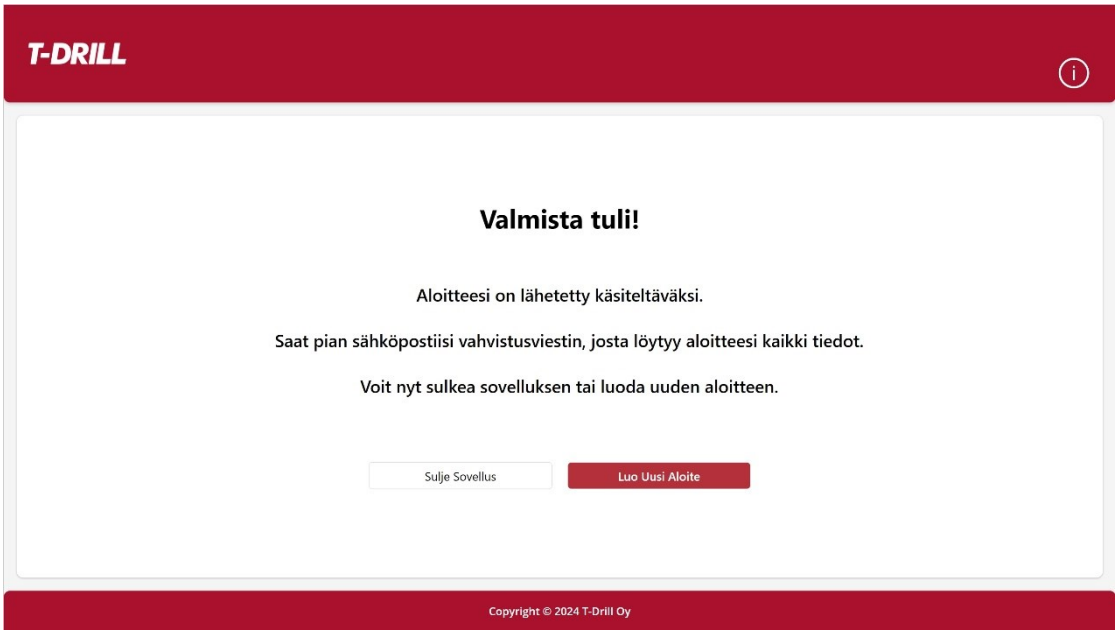

T-DRILL
Tietojen lähetys onnistui!
Copyright © 2024 T-Drill Oy

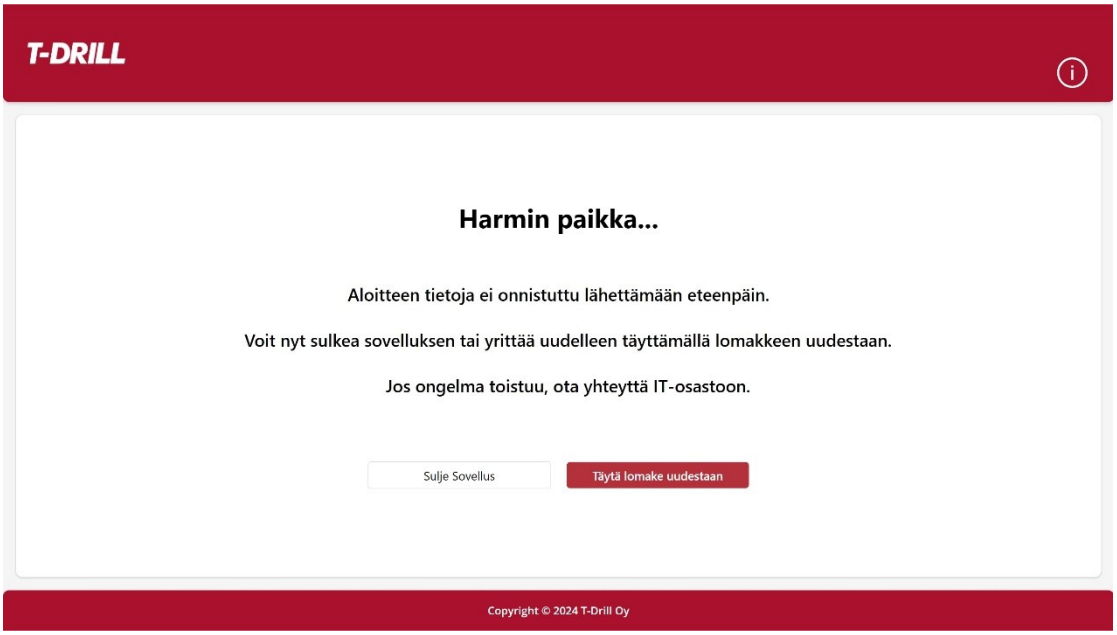

T-DRILL
Lomakkeen lähetyksessä tapahtui virhe!
Copyright © 2024 T-Drill Oy

Liite 3/2


T-DRILL
Tuotteiden tallentamisessa tapahtui virhe!
Copyright © 2024 T-Drill Oy

Liite 4


T-DRILL
Valmista tuli!
Aloitteesi on lähetetty käsiteltäväksi.
Saat pian sähköpostiisi vahvistusviestin, josta löytyy aloitteesi kaikki tiedot.
Voit nyt sulkea sovelluksen tai luoda uuden aloitteen.
Sulje Sovellus
Luo Uusi Aloite
Copyright © 2024 T-Drill Oy


T-DRILL
Harmin paikka...
Aloitteen tietoja ei onnistuttu lähettämään eteenpäin.
Voit nyt sulkea sovelluksen tai yrittää uudelleen täyttämällä lomakkeen uudestaan.
Jos ongelma toistuu, ota yhteyttä IT-osastoon.
Sulje Sovellus
Täytä lomake uudestaan
Copyright © 2024 T-Drill Oy

```
ForAll(
    colSpTuoteTiedot;
    If(Or(!IsBlank(Ftavaraseloste); !IsBlank(Fnimikenumero));
        Patch(
            'Hankinta Tuotteet';
            Defaults('Hankinta Tuotteet');
            {
                'Aloite: RefNo': {
                    Id: frm_lomakesivu_tilaustiedot.LastSubmit.Tunnus;
                    Value: frm_lomakesivu_tilaustiedot.LastSubmit.RefNo
                };
                'Aloite: Tarve Pvm': {
                    Id: frm_lomakesivu_tilaustiedot.LastSubmit.Tunnus;
                    Value: frm_lomakesivu_tilaustiedot.LastSubmit.'Tarve Pvm'
                };
                'Merkki Tai Kohde': FKohde;
                Nimikenumero: Fnimikenumero;
                Tavaraseloste: Ftavaraseloste;
                Lukumäärä: Flukumaara;
                Yksikkö: Fyksikko;
                Huomiot: Fhuomioita;
                'Tuote Linkki': If(!IsBlank(Flinkki);
                    "<a href='" & Flinkki & "' target='_blank'
                    data-interception='off'>" & Flinkki & "</a>";
                    Blank()
                )
            }
        )
    )
);;
//Tarkistetaan onko ylläolevan patch funktion aikana tapahtunut virheitä.
If(!IsEmpty(Errors('Hankinta Tuotteet'));

    //Jos true, eli virheitä on:
    Remove('Hankinta Aloitteet';frm_lomakesivu_tilaustiedot.LastSubmit);;
    Set(gblItemPatchSuccess;false);

    //Jos false, eli ei virheitä:
    Set(gblItemPatchSuccess; true);;
    Set(varTallennaLomake; frm_lomakesivu_tilaustiedot.LastSubmit);;
    Set(varLinkki;
        Concatenate(
            "https://uwira2.sharepoint.com/sites/TDHankintaApp/Lists/
            HankintaTuotteet/AllItems.aspx?FilterField1=TilausRefNo&
            FilterValue1=";
            varTallennaLomake.RefNo;
            "&FilterType1=Lookup"
        )
    );;
    Patch(
        'Hankinta Aloitteet';
        LookUp(
            'Hankinta Aloitteet';
            Tunnus = varTallennaLomake.Tunnus
        );
        {
            LinkkiTuotteisiin: Concatenate(
                "<a href='" & varLinkki & "' target='_blank'
                data-interception='off'>Tuotteet</a>")
        }
    );;

);;
Navigate('Onnistumis sivu')
```