Roni Kuikka

Työsuoriteseurantasovelluksen kehittäminen Power Appsilla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

4.5.2023

Tiivistelmä

Tekijä:	Roni Kuikka
Otsikko:	Työsuoriteseurantasovelluksen kehittäminen Power Appsilla
Sivumäärä:	38 sivua
Aika:	4.5.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisovellukset
Ohjaaja:	Lehtori Miikka Mäki-Uuro

Insinöörityössä kehitettiin toimeksiantajayritykselle uusi työsuoriteseurantasovellus Power Apps -kehitystyökalulla. Toimeksiantajayritys on kotimainen henkilöstö- ja asiantuntijapalveluihin sekä talousalaan keskittyvä osakeyhtiö. Sovellus luotiin yhdelle yrityksen palvelukokonaisuuksista.

Sovelluksen tarkoituksena oli sekä korvata palvelun senhetkinen Excel-pohjainen työsuoriteseurantaratkaisu että huomioida yrityksen saman vuoden aikana käyttöön otettava uusi työajanseurantajärjestelmä.

Insinöörityö toteutettiin Power Appsilla pääasiassa sen Teams- ja Dataverse for Teams -alustojen yhteensopivuuden vuoksi, mutta sovelluksen yhtenäistäminen yrityksen muiden Power Apps -työsovellusten kanssa oli myös tavoitteena. Sovelluksen kehittämisen lisäksi työhön lukeutui työtehtäville ja -kirjauksille tietokantataulurakenteen luominen em. Dataverse for Teams -tietokanta-alustalla.

Kehitystöiden jälkeen työssä suoritettiin pienimuotoinen pilotointivaihe, jossa testattiin sovelluksen ja taulujen toiminnallisuutta rajatulla käyttäjämäärällä ja kerättiin palautetta. Pilotissa ei ilmennyt ongelmia toiminnallisuuden osalta, mutta mm. työtehtävien suodatustapaa mukautettiin saadun palautteen pohjalta.

Insinöörityöksi kehitettyä työsuoriteseurantasovellusta ylläpidetään ja mahdollisesti laajennetaan jatkossakin palvelukokonaisuuden työnkuvan muuttuessa.

Avainsanat: Microsoft Dataverse, Microsoft Dataverse for Teams, Microsoft Power Apps, Microsoft Power Platform, Microsoft Teams, Power Fx, sovelluskehitys, työsuoriteseuranta,

Abstract

Author:	Roni Kuikka
Title:	Developing a work performance tracking application with Power Apps
Number of Pages:	38 pages
Date:	4 May 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information & communication technologies
Professional Major:	Game applications
Supervisor:	Miikka Mäki-Uuro, Senior Lecturer

This thesis covers creating a new work performance tracking application developed with Power Apps for a client company. The client is a Finnish limited liability company focused on personnel and specialist services as well as finance. The application was built for one of the client company's teams.

The purpose of the application was to both replace the team's current Excel based work performance tracking solution and to take into consideration the company's new employee time tracking system set to be introduced in the current year.

The project was developed with Power Apps mainly for its Teams and Dataverse for Teams compatibility but unifying the application with the company's other Power Apps work applications was also key. In addition to developing the application the project included creating a database table structure for the work tasks and logs with Dataverse for Teams.

After the development there was a small-scale pilot phase for the application and the data tables where a small group of testers tested the functionality and gave feedback. No technical issues occurred during the pilot, but some changes were made based on the gathered feedback, like the way tasks are filtered in the application among other things.

The work performance tracking application built for this thesis will be maintained and possibly expanded upon in the future as well as the team's job description changes.

Keywords: application development, Microsoft Dataverse, Microsoft Dataverse for Teams, Microsoft Power Apps, Microsoft Power Platform, Microsoft Teams, Power Fx, work performance tracking

Sisällys

1 Johdanto

Työn ja sen suoritteiden seuraaminen on oleellinen ja tärkeä osa liiketoimintaa, koska siten konkreettisesti nähdään, onko työ kannattavaa yritykselle vai ei [1]. Työsuoritteiden seuraamiseen on useita eri tapoja ja keinoja työnkuvasta, yrityksestä ja työntekijöistä riippuen aina kynä-ja-paperi-kirjanpidosta täysin automatisoituihin raportteihin. Työssä, jossa tehdään lukuisia työtehtäviä, joilla on omia tehtäväkohtaisia palvelutasosopimuksia (engl. service-level agreement, SLA) sekä päivätasolla seurattavia suoritteita, on järkevää käyttää esim. erillistä sovellusta, jonka avulla voidaan tallentaa työkirjauksia ja pitää niistä kirjaa. Tämän insinöörityön tarkoituksena oli luoda uusi Microsoft Power Apps -pohjainen työsuoriteseurantasovellus erään toimeksiantajayrityksen tietyn palvelukokonaisuuden käyttöön.

Insinöörityön toimeksiantajayrityksenä oli kotimainen henkilöstö- ja asiantuntijapalveluihin sekä talousalaan keskittynyt osakeyhtiö, joka on toiminut jo yli kahden vuosikymmenen ajan. Yritys työllistää useita satoja työntekijöitä eri paikkakunnilla ympäri Suomea ja yrityksen liikevaihto on kymmeniä miljoonia euroja vuosittain. Toimeksiantajayrityksellä oli (ja on edelleen raporttia kirjoittaessa toukokuussa 2023) meneillään sisäisiä palvelukohtaisia kehitystöitä yrityksen uuden työajanseurantajärjestelmän käyttöönoton ja tulevien työehtosopimusmuutosten vuoksi, joten työsuoriteseurannan uudistaminen oli ajankohtainen aihe insinöörityölle.

Uuden työsuoriteseurannan oli tarkoitus korvata senhetkinen Microsoft Excel -pohjainen työkirjausjärjestelmä, jolla seurattiin suoritteiden lisäksi myös työaikaa. Uuden erillisen työajanseurannan vuoksi työajan kirjaamista ei enää edellytetty uuteen suoriteseurantasovellukseen. Insinöörityön keskeisimpinä tavoitteina oli luoda selkeän käyttäjäkokemuksen tarjoava työkirjaussovellus Microsoft Power Appsilla ja päivittää kirjausten tietokantataulurakenne yhteensopivampaan muotoon uuden sovelluksen kanssa Microsoft Dataverse -tietokanta-alustaa hyödyntäen. Microsoft Teams -integraation varmistaminen ja

yrityksen muiden palveluiden vastaavien työsovellusten kanssa yhtenäisyyteen pyrkiminen olivat myös oleellisia tavoitteita projektissa.

Raportissa avataan aluksi insinöörityössä käytettyjä kehitysmenetelmiä eli Microsoft Power Platform -työkaluja ja -ohjelmistoja, tarkennettuna Power Appsia ja Dataversen Teams-pohjaista versiota, Dataverse for Teamsia. Käsittelyosiossa pureudutaan suoriteseurannan määrittely- ja kehitystyövaiheisiin niin Power Apps -sovelluksen kuin Dataverse-tietokantataulurakenteenkin osalta sekä tavoiteltuun käyttäjäkokemukseen sovellustoimintokohtaisesti. Lopuksi tarkastellaan insinöörityön lopputulemaa työn aikana järjestetyn pilotin pohjalta ja yhteenvedon muodossa sekä pohditaan hieman potentiaalisia jatkokehitysaihioita projektille.

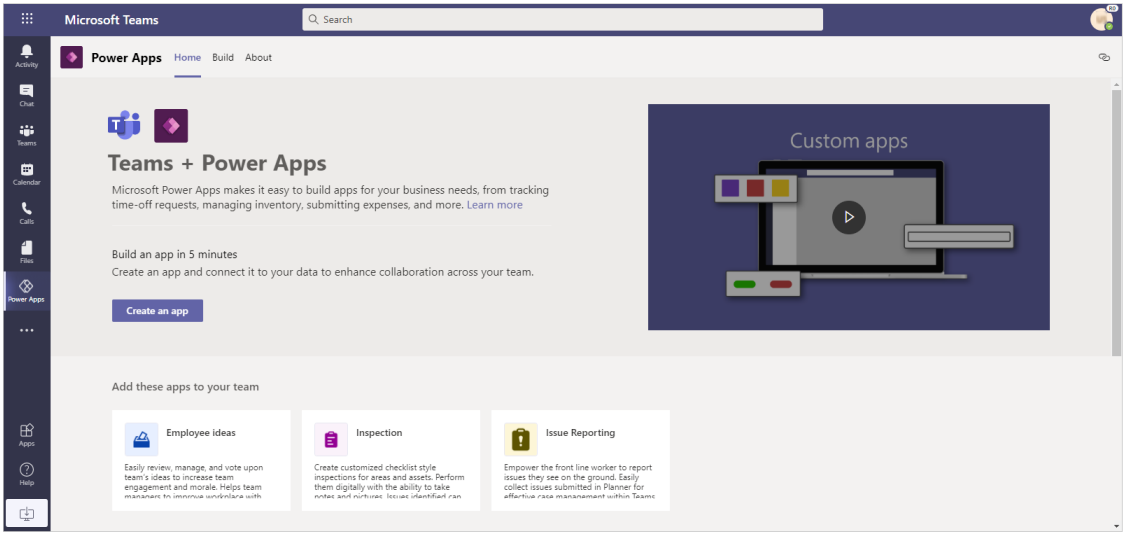
2 Microsoft Power Platform -ohjelmistot

Power Platform on Microsoftin kehittämä ohjelmistokokonaisuus, johon kuuluu pääasiassa liiketoiminnan kehitykseen, ylläpitämiseen ja raportoimiseen keskittyviä keskenään yhteensopivia kehitystyökaluja ja -ohjelmistoja. Power Platform koostuu Power Apps-, Power Automate-, Power BI-, Power Pages-, Power Virtual Agents- ja Microsoft Dataverse -työkaluista ja -ohjelmista. Power Platform -työkalut on suunniteltu nk. vähäkoodisen ohjelmoinnin (engl. low-code development) mukaisesti eli saavutettaviksi ja helposti käytettäviksi ja luettaviksi käyttäjän teknisestä osaamisesta riippumatta. Insinöörityössä paneudutaan pääasiassa kahteen Power Platform -työkaluun: Power Appsiin, jota käytetään nimenmukaisesti sovelluskehitystä varten, ja relaatiotietokanta-alustaan, Microsoft Dataverseen. Raportointiin ja datan visualisointiin käytettävään Power BI -ohjelmaan tutustutaan myös, mutta vain pintapuolisesti. [2.]

2.1 Microsoft Power Apps -kehitystyökalu

Power Apps on vuonna 2016 julkaistu kehitystyökalu, jolla voidaan luoda Microsoft Teamsissa, selaimella ja mobiililaitteilla käytettäviä sovelluksia [3]. Power Appsin kehitystyökaluja on mahdollista käyttää suoraan selaimen tai

Teamsin kautta (kuva 1), ja kuten muutkin Power Platform -työkalut, Power Apps on kohdistettu pääasiassa liiketoimintakäyttöön. Uutta sovellusta luotaessa käyttäjä voi valita kahden päävaihtoehdon väliltä: canvas apps tai model-driven apps. Canvas-sovelluksissa kehitys aloitetaan tyhjältä ruudulta, johon käyttäjä voi tuoda eri elementtejä ja datalähteitä, joista sovellus tulee koostumaan. Model-driven apps on datalähtöinen tapa aloittaa sovelluksen kehittäminen, jossa lähtökohtana on jokin annettu datalähde, kuten Dataverse-taulu. Insinöörityössä luodaan canvas apps -pohjainen sovellus, ja raportissa käsitellään Power Appsia siitä näkökulmasta. [4.]

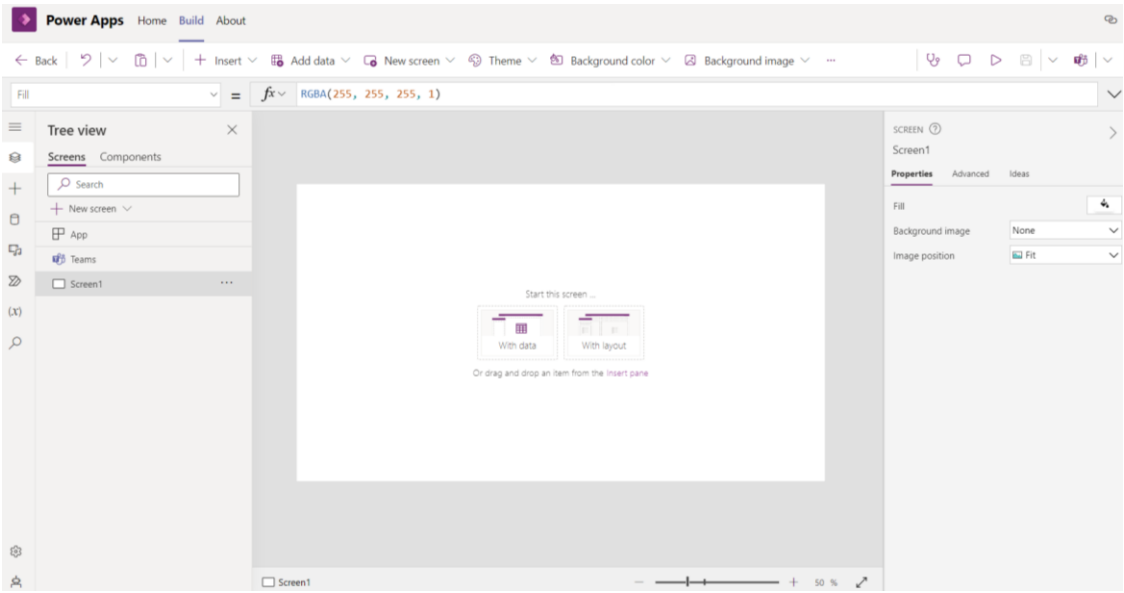


Kuva 1. Power Appsin aloitussivu Microsoft Teamsissa [5].

Power Appsissa on graafinen käyttöliittymä, johon käyttäjä voi reaaliaikaisesti lisätä kontrolleja eli ohjelman osia, joilla on erilaisia toiminnallisuuksia. Kontrollien ulkoasua, sommittelua ja kokoa pystyy muokkaamaan koskematta lainkaan koodiin, mutta esim. painikkeen toiminnallisuus tai syötekentän virhetarkistus tulee käyttäjän määrittää itse ko. kontrollien funktioissa. Kontrolleja käytetään viemällä ne sovelluksen ruudulle, ja ruutuja voi olla useampi kuin yksi sovellusta kohti.

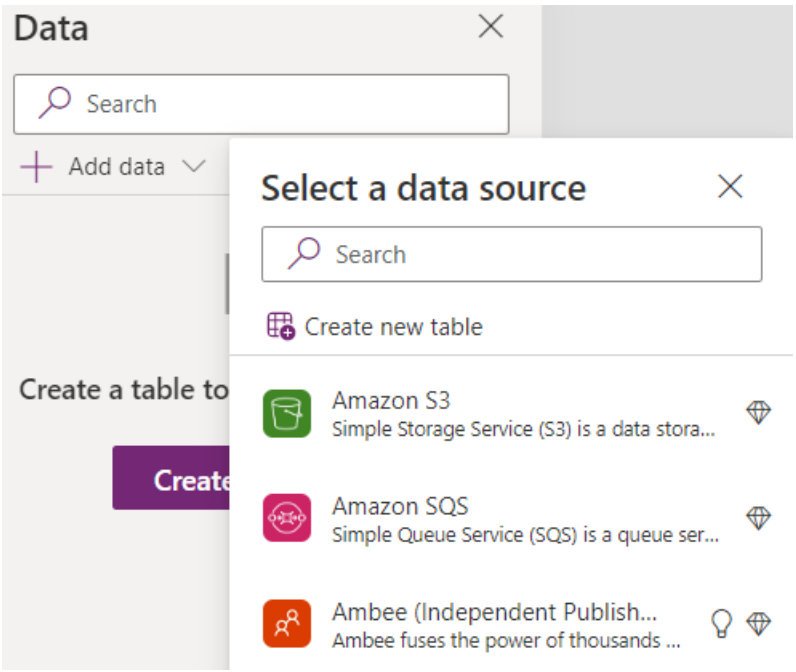
Power Appsin vakiokehitysympäristönäkymä (kuva 2) koostuu kontrollikohtaisten asetusten valikosta ja nk. puuhierarkiasta, jossa ruudut, komponentit ja

kontrollit ovat listattuna allekkain. Puuhierarkian vasemmalta puolelta käyttäjä voi myös avata valikon, josta löytyvät sovelluksessa käytettävät mediatiedostot, datalähteet, muuttujat ja Power Automatella luodut automaatiot. Valikosta voi myös syöttää ruudulle uusia kontrolleja, kuten ruudun yllä olevasta palkistakin, ja hakea kaikkia sovelluksessa käytettäviä osia niiden nimillä. Käyttäjä voi testata sovellusta milloin tahansa painamalla play-painiketta yläpalkista.



Kuva 2. Uuden tyhjän canvas apps -pohjaisen Power Apps -sovelluksen kehitysympäristönäkymä.

Power Appsissa on useita erilaisia liittimiä, joiden avulla voidaan tuoda sovellukseen datalähteitä ja yhteyksiä eri palveluihin, kuten Dynamics 365 -yrityssovelluksiin. Kehittäjä voi esim. kytkeä Power Apps -sovellukseen Office 365 Outlook -liittimen, jolloin käyttäjä voi lähettää sähköpostin sovelluksen sisällä sallittuaan yhteyden Office 365 -tiliinsä. Liittimiä on kolmea erilaista: kaikkien käytössä olevia vakioliittimiä, mukautettuja liittimiä ja maksullisen lisenssin vaativia liittimiä. Maksullisen liittimen tunnistaa Power Appsin sisällä erillisestä timanttikuvakkeesta liittimen nimen yhteydessä (kuva 3). Tiettyjä maksullisia liittimiä, kuten insinöörityössä käytettävää Dataverse-liitintä, voidaan käyttää ilman maksullista lisenssiä, kun kehitystyö tehdään Microsoft Teamsin sisällä [6].



Kuva 3. Maksullisia liittimiä Power Appsissa.

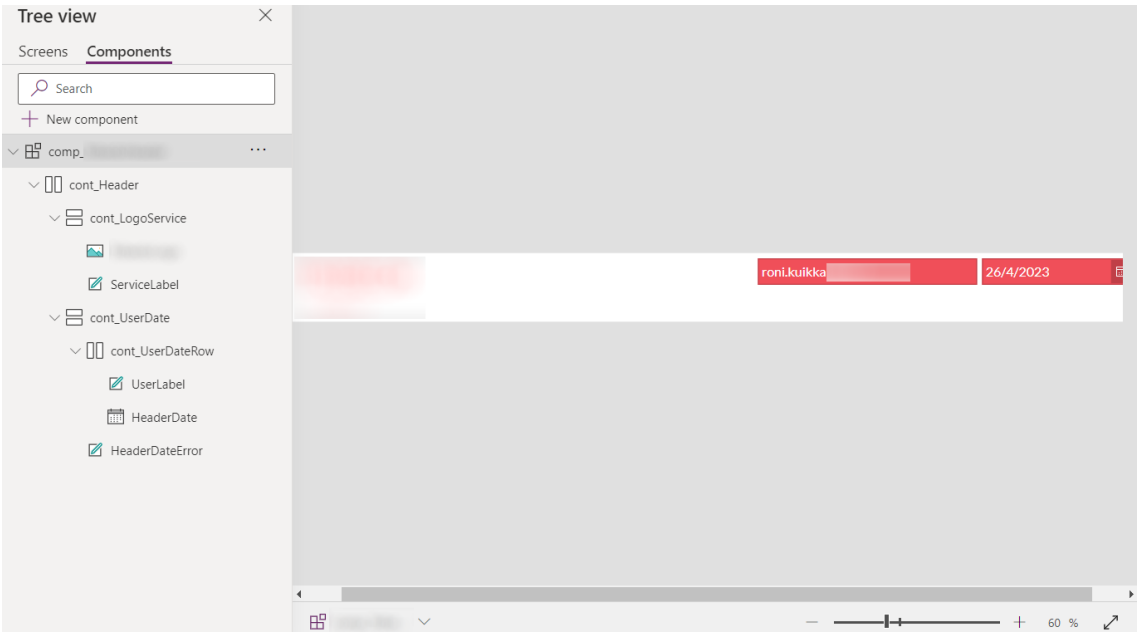
Maksullisia liittimiä on mahdollista kytkeä sovellukseen muokkausnäkymässä ilman lisenssiäkin, mutta tällöin sovellusta ei voi avata tai testata. Teamsin sisällä kehitettyjä Power Apps -sovelluksia, joissa on käytössä liittimiä, jotka vaativat maksullisen lisenssin Teams-ympäristön ulkopuolella, ei siis myöskään voi avata ulkoisessa selaimessa ilman vaadittuja lisenssejä.

Ohjelmointikielenä Power Appsissa käytetään nykyään Power Fx -nimistä kieltä, joka muistuttaa vanhemmissa Microsoftin ohjelmissa, kuten Excelissä ja Accessissa, käytettävää VBA-kieltä (Visual Basic for Applications). Yksi Power Fx:n keskeisimmistä tavoitteista ja hyödyistä on kielen helppolukuisuus kaikille sen käyttäjille [7]. Power Platform -työkalujen vähäkoodisesta periaatteesta huolimatta toimivan sovelluskokonaisuuden kehittäminen Power Appsissa edellyttää käyttäjältä jonkintasoista ymmärrystä mm. muuttujien käytöstä, ehtolauseista, toimintojen suoritusjärjestyksestä ja muuttujien arvojen väliaikaisesta tallettamisesta.

Power Appsin kaltaiset vähäkoodiset ja ei koodia lainkaan vaativat työkalut ovat saaneet myös kritiikkiä osakseen, sillä vähäkoodinen suunnittelu voi olla

toteutettu sillä kustannuksella, että ohjelman muokattavuus on rajoitetumpaa kuin perinteisissä kehitystyökaluissa tai sen tietoturva puutteellista [8]. Power Appsissa rajoituksia ilmenee mm. tiettyjen kontrollien, kuten päivämäärävalintaan käytettävän DatePicker-kontrollin, muokkausvaihtoehtojen suppeudessa ja Teams-ympäristöjen tuomien tietoturvahaavoittuvuuksien muodossa. Työntekijä, joka on omistajana Teams-kanavalla, jossa on Power Apps -sovelluskokonaisuus toiminnassa, saattaa esim. vahingossa poistaa sovelluksen kanavalta vaivatta. Liiketoimintakäytössä vähäkoodiset sovellukset ovat kuitenkin kasvaneet suosiossa viime vuosina; kansainvälinen IT-alan tutkimus- ja konsultointiyritys Gartner ennusti joulukuussa 2022 vähäkoodisten teknologioiden markkinoiden kasvavan jopa 20 prosenttia vuonna 2023 [9].

Power Appsissa voi myös luoda komponenttikirjastoja, joihin käyttäjä voi rakentaa komponentteja eli sovelluksesta riippumattomia irrallisia kontrollikokonaisuuksia (kuva 4). Komponentteja voi viedä ja tuoda käytettäväksi eri sovelluksiin tai saman sovelluksen eri ruutuihin. Komponenttia luodessa tulee kuitenkin huomioida mahdollisten muuttujien näkyvyys, riippuvuudet ja käyttötavat komponentin sisällä ja ulkopuolella; jos komponentissa on muuttujia, niitä ei voi kutsua tai niiden arvoa muokata sovelluksen puolella pelkkää muuttujaa kutsumalla, ellei komponentissa ole kytketty päälle asetus, joka sallii komponentille pääsyn koko sovellukseen (engl. ”Access app scope" -asetus) [10].



Kuva 4. Power Apps -komponentti, joka näyttää sovellusta käyttävän käyttäjän sähköpostiosoitteen ja kirjauspäivämäärän. Komponentin voi kätevästi sommitella bannerityylisesti sovelluksen ruudulle.

Komponenttien tuomat hyödyt riippuvat sovelluksen käyttötarkoituksista ja rakenteesta. Jos sovelluksella on riippuvuuksia useisiin eri tauluihin ja muuttujiin, joita käytetään koko sovelluksen laajuudella, sen toiminnallisuuden erotteleminen omiin (suljettuihin) irrallisiin osiin ei kenties ole paras tapa toteuttaa sitä.

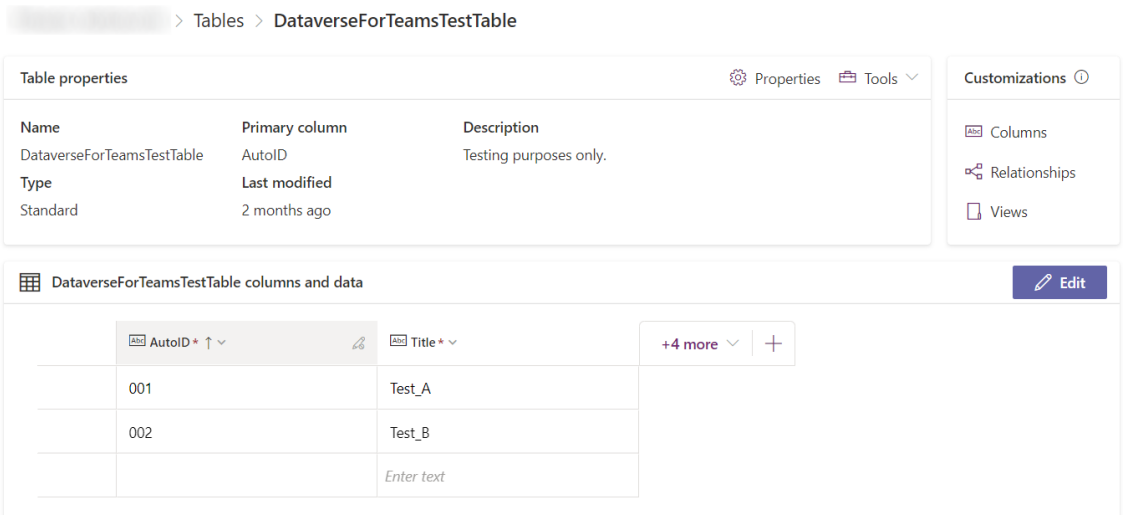
2.2 Microsoft Dataverse for Teams -tietokanta-alusta

Microsoft Dataverse on relaatiotietokanta-alusta, jota voidaan käyttää yhdessä Power Platform -liiketoimintasovellusten kanssa mm. yrityksen tietojen säilyttämiseen ja päivittämiseen. Dataverse-tauluja saa liitettyä esim. Power Apps -sovelluksiin ja Power BI -raporteille ilman nk. välittäjäohjelmia tai -ratkaisuja. Raportissa viitataan Dataverseen ilman Microsoft-etuliitettä, mutta sitä ei tule kuitenkaan sekoittaa samannimiseen tutkimus- ja tutkijakäyttöön tarkoitettuun avoimen lähdekoodin verkkosovellukseen, Dataverseen [11].

Tavallisesti Dataverse edellyttää joko erillistä käyttäjä- tai sovelluskohtaista maksullista lisenssiä, mutta Teams-ympäristöissä käytettävä Dataverse for

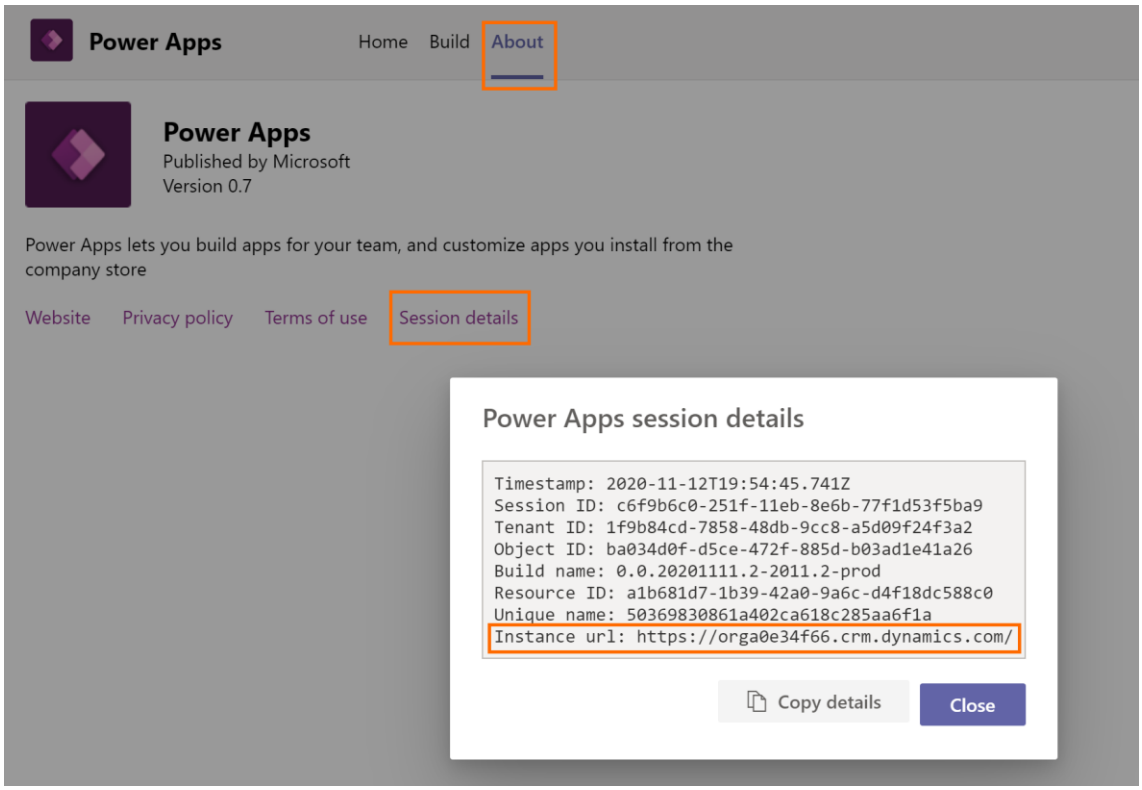
Teams sisältyy Microsoft 365 -lisenssiin, kuten Power Appsikin. Dataverse for Teamsissa on kuitenkin mm. taulu-, kehitys- ja tietoturvarajoituksia, joita Dataversessa ei ole. Keskeisiä Teams-version rajoituksia ovat esim. miljoonan rivin tai kahden gigatavun kokoraja Teams-ympäristössä, kenttä- ja saraketason suojauksen puute ja Dataversen rajapintaan pääsyn estäminen. Taulujen käyttötarkoituksista, laajuuksista ja määrästä riippuen Dataverse for Teams voi kuitenkin olla riittävä taulurakenteen toteutukseen. Insinöörityössä käytettiin Dataverse for Teamsia. [12.]

Dataverse-taulut luodaan niin Teamsissa kuin selaimellakin Power Apps -sivun kautta. Dataverse for Teams -taulut ovat Teams-ympäristökohtaisia, ja niitä pääsee tarkastelemaan ja muokkaamaan Power Apps -sivun kautta valittuaan ympäristön, jossa taulut sijaitsevat (kuva 5). Uutta taulua luotaessa määritetään pääsarake eli perusavain sekä taulun nimi ja kuvaus. Sarakkeita voi lisätä ja niiden asetuksia muokata taulun luonnin jälkeenkin.



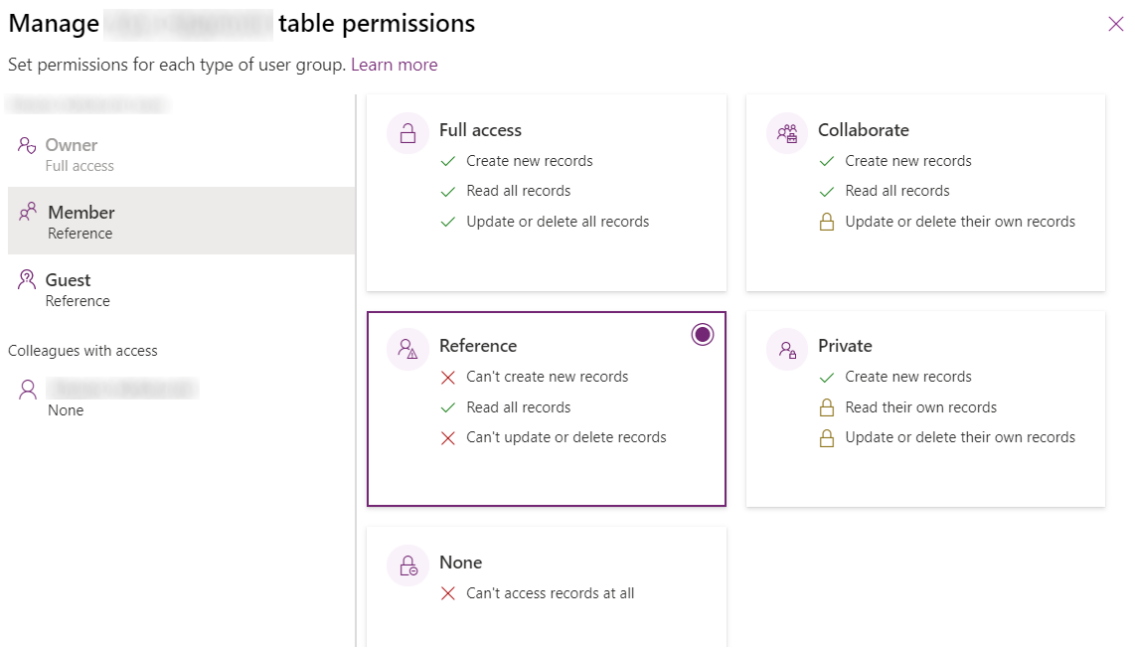
Kuva 5. Vakionäkymä, kun Dataverse-taulu on avattu Teamsissa, kuvassa esimerkkitaulu.

Dataverse for Teams -tauluja voi kytkeä Power BI -raportteihin hakemalla instanssiosoitteen siitä Teams-ympäristöstä, missä taulut sijaitsevat (kuva 6) ja syöttämällä sen raportille Power BI Desktop -ohjelmassa ”Nouda dataa” -painikkeen kautta [13].



Kuva 6. Esimerkki Teams-ympäristön instanssiosoitteen hakemisesta [13].

Yksi Dataverse for Teamsin merkittävimmistä heikkouksista on sen puutteelliset tietoturva-asetukset. Tauluille voi asettaa omistaja-, käyttäjä- ja vieraskohtaisia käyttöoikeuksia (kuva 7), mutta Teamsissa taulujen sijaintiin käsiksi pääsevät henkilöt voivat silti lisätä ja poistaa sarakkeita, muokata taulun kuvausta ja jopa poistaa tauluja, vaikka heille olisi myönnetty vain lukuoikeudet tauluihin, jos taulut eivät ole suljetun järjestelmän sisällä [14]. Siitä riippumatta onko todennäköistä, että näin tapahtuisi, on se silti suuri tietoturvahaavoittuvuus Power Apps -sovellusten ja Power BI -raporttien käyttämien taulurakenteiden kannalta.



Kuva 7. Dataverse-taulun käyttöoikeuden hallintanäkymä.

3 Työsuoriteseurantasovelluksen kehittäminen

Insinöörityön kehitysprosessi oli melko vapaamuotoista. Kiteytettynä projekti koostui vaatimusmäärittelyn muodostamisesta, säännöllisistä palavereista, Power Apps -sovelluskehityksestä, Dataverse for Teams -tietokantataulurakenteen luomisesta, kahden em. kokonaisuuden yhdistämisestä ja lopulta suoriteseurannan pilotoinnista.

3.1 Vaatimusmäärittely

Projektin alkaessa vaatimusmäärittelyä työsuoriteseurannalle ei ollut vielä luotu, mutta palvelun, jolle sovellus oli tarkoitus tulla käyttöön, työtehtävät ja niiden edellyttämät seurattavat suoritteet oli valmiiksi listattu. Sovelluksen määrittely aloitettiin tarkastelemalla aiemman Excel-pohjaisen työkirjausseurannan toteutusta ja toiminnallisuutta sekä koottua työtehtävälistaa ja seurattavia suoritetyyppejä. Suoritteita, joita sovelluksessa tulisi tehtäväkohtaisesti seurata, oli kolmea erilaista: kappale- tai sivu-, dokumentti- ja asiakasmäärä. Kappalemäärien lisäksi tietyistä töistä saatetaan myös seurata SLA:ta eli palvelutasosopimusta,

ja sitä varten tarvittiin kirjattavan työn vanhimman käsitellyn materiaalin saapumispäivämäärätieto. Työtehtävälistasta karsittiin ne tehtävät, joista ei seurata laskutettavia suoritteita lainkaan; nämä työt kuuluvat vain toimihenkilöiden tuntitöihin työajanseurannan puolelle.

Suoriteseurannan kehitysprosessi aloitettiin kartoittamalla työn tilanneen palvelukokonaisuuden liiketoiminnallisia vaatimuksia, tarpeita ja toiveita sovellukselle. Ensin koottiin lista palvelun työtehtävistä, joiden suoritteita, eli kappalemääriä ja työkohtaisia päivämäärätietoja, tulisi pystyä seuraamaan. Palvelun aiemmassa työseurannassa kirjauksista seurattiin myös työaikaa, mutta toimeksiantajayrityksen sisäisien uudistusten takia työajanseurantaa ei uuteen sovellukseen edellytetty. Seuraavana vaiheena oli yrityksen tehtäväkohtaisten palvelutasosopimusten huomioiminen, jota vaaditaan laskutuksen kannalta. SLA:lla tarkoitetaan enimmäispäivämäärää, jonka sisällä työtehtävässä käsiteltävä materiaali tulee olla loppuun asti käsitelty. Siten toimeksiantajayritys seuraa, ovatko palvelukohtaiset työt pysyneet yrityksen asiakkailleen ja kumppaneilleen lupaamien sopimusten puitteissa.

Vaatimusmäärittelyn teko aloitettiin käyttäjätarinatyyppisesti eli kuvaillen ominaisuuksia, joita sovelluksesta tulisi löytyä ”käyttäjä voi” -muotoisesti, esim. ”käyttäjä voi muokata tekemäänsä työkirjausta”. Käyttäjätarinat koottiin palvelun kokoaman vaatimus- ja tarvelistan mukaan määrittelyä varten luotuun dokumenttiin (kuva 8).

- Käyttäjä pystyy valitsemaan työtehtävän valittuaan työlajin
    - Sovellus aktivoi työtehtävän valinnan yhteydessä tehtävän vaatimat syötekentät kirjausnäkymässä
        - -> **Toteutuu**
- Käyttäjä pystyy tallentamaan kirjauksen työtehtävän valittuaan ja vaaditut tiedot täytettyään
    - Sovellus antaa virheilmoituksen ja estää kirjauksen, jos tallennettavan kirjauksen tiedot ovat puutteelliset
        - -> **Toteutuu**
    - Sovellus antaa virheilmoituksen ja estää kirjauksen, jos tallennettavan kirjauksen tiedot ovat syötetty väärässä muodossa (esim. kokonaisluvun sijaan kirjain tai desimaaliluku)
        - -> **Toteutuu**, kentät sallivat vain pos. kokonaislukuja
- Käyttäjä pystyy vaihtamaan kirjauspäivämäärää
    - Käyttäjä pystyy syöttämään valitulle päivälle työkirjauksia
        - -> **Toteutuu**
    - Huomioitavaa: muutokset edellisiin kirjauksiin kirjaushistoriasivun kautta
        - -> **Huom.!** kirjausten muokkaus mahdollista myös työseurantasivun kautta
- Käyttäjä pystyy hakemaan työtehtävää hakukentän avulla valittuaan työlajin
    - -> **Toteutuu**
- Käyttäjä pystyy poistamaan päivän aikana tehdyn kirjauksen -> **Toteutuu**, ei rajoitu vain saman päivän aikana tehtyihin kirjauksiin
    - Sovellus avaa ponnahdusikkunan, joka kysyy varmistusta ennen kirjauksen poistoa
        - -> **Toteutuu**
    - Käyttäjä **ei** pysty kumoamaan kirjauksen poistoa
        - -> **Toteutuu** (käyttäjän tulee luoda uusi kirjaus ”palauttaakseen” poistetun)
- Käyttäjä pystyy muokkaamaan päivän aikana tehtyä kirjausta
    - Sovellus avaa valitun kirjauksen ponnahdusikkunassa, jossa tiedot ovat muokattavissa omissa tekstikentissään
        - -> **Toteutuu**
    - Käyttäjä pystyy tallentamaan tehdyt muutokset
        - -> **Toteutuu**

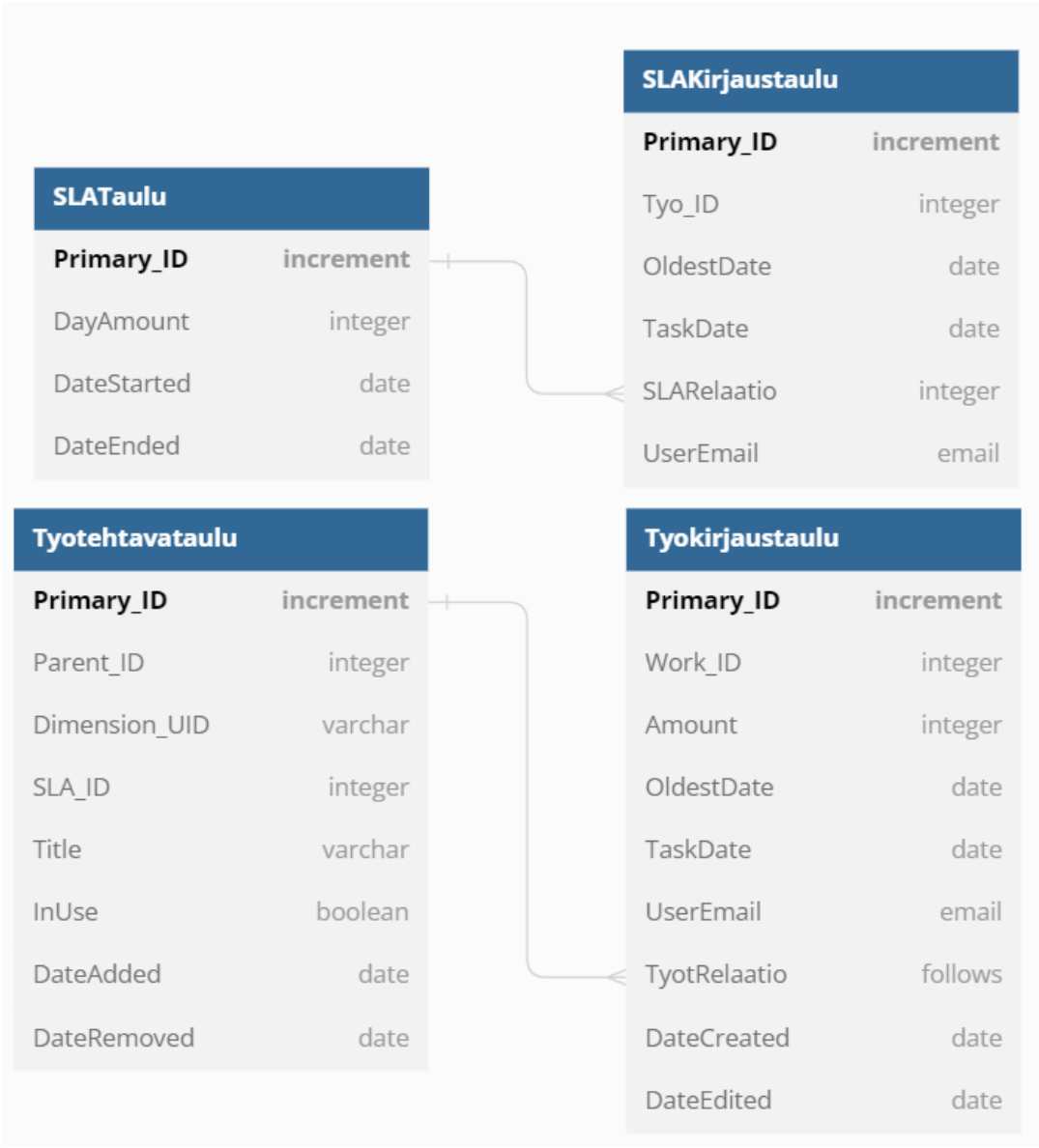
Kuva 8. Ote suoriteseurannan vaatimusmäärittelydokumentista, joka sisältää esimerkkikäyttäjätarinoita sovelluksen käyttäjäkokemuksesta. Korostetut toteutuu-merkinnät ovat määrittelydokumenttiin myöhemmissä palavereissa lisättyjä mainintoja sovelluksessa toteutuvista kohdista.

Vaatimusmäärittely käytiin yhdessä läpi kehityspäällikön ja suoriteseurannan tilanneen palvelun team leadin kanssa helmikuussa 2023, minkä jälkeen se hyväksyttiin. Viikoittaiset palaverit suoriteseurannan toteuttamiseen liittyen jatkuivat kuitenkin tämän jälkeenkin, ja niissä katselmoitiin työn edistystä ja ajoittain nostettiin esiin asioita, joita olisi tullut mainita vaatimusmäärittelyä koostaessa.

3.2 Tietokantataulujen rakenne

Toimeksiantajayrityksen aiemmissa Power Apps -työseurantasovelluksissa käytettiin Microsoft SharePoint-listoja työkirjausten tallentamiseen. SharePoint-listoissa on 5 000 rivin kokorajoitus, ja kirjausten tuominen Power BI -raporteille edellytti välikäsiratkaisuna niiden viemistä PostgreSQL-tietokantaan, joka vaati käyttäjältä lisävaltuuksia ja erillisiä (suurimmaksi osaksi automatisoituja) Python-skriptejä, joilla hallita vietäviä ja vietyjä kirjauksia. Näistä syistä haluttiin tehdä siirto Dataverse for Teamsiin.

Työsuoriteseurantaa varten tuli pitää kirjaa tehtäviin liittyvistä tiedoista, kuten työlajeista (eli minkä yläkategorian alle tehtävä kuuluu), seurattavista suoritteista ja palvelutasosopimuksista. Sovellus käyttää neljää eri taulua tietojen kirjaamiseen, säilyttämiseen ja ylläpitämiseen: työtehtävätaulu, josta katsotaan kaikkien tehtävien nimikkeet ja niiden edellyttämät suoritteet; kirjaustaulu, johon viedään käyttäjän sovelluksessa syöttämät työkirjaukset; SLA-taulu, josta tarkistetaan eri palvelutasosopimukset, ja SLA-kirjaustaulu, johon viedään työkirjausten yhteydessä kirjattavat SLA-tiedot, jos sellaisia seurataan ko. työtehtävässä. Suurin osa taulujen välisistä relaatioista muodostetaan vasta Power BI -raportilla, mutta työkirjaustaulussa on viiteavain työtehtävätauluun ja SLA-kirjaustaulussa vastaavasti SLA-tauluun (kuva 9).



Kuva 9. Yksinkertainen relaatiokaavio työsuoriteseurannassa käytettävistä Dataverse-tauluista. Työkirjaustaulussa on viiteavain TyotRelaatio työtehtävätauluun ja SLA-kirjaustaulussa SLARelaatio SLA-tauluun. Loput taulujen väliset yhteydet muodostetaan vasta Power BI -raportin datamallissa.

3.2.1 Työtehtävä- ja SLA-taulut

Työtehtävätaulu koostuu kahdeksasta eri sarakkeesta (taulukko 1). Taulussa on ensimmäisenä listattuna palvelukokonaisuuden työlajit, minkä jälkeen seuraavat työtehtävät työlajijärjestyksessä ja viimeisenä tehtäväkohtaiset seurattavat suoriterivit esim. dokumenttien kappalemäärä työlle X. Taulussa siis viitataan

taulunsisäisesti aiempien rivien päätunnisteisiin. Keskeisimmät syyt taulun rakenteelle olivat sekä yksinkertainen työtehtävien lisäämis- ja ylläpitomahdollisuus että datan yhdenmuotoisena pitäminen yrityksen aiempien vastaavien tietokantataulujen kanssa. Työtehtävätaulussa on yhteensä yli 100 riviä.

Taulukko 1. Työtehtävätaulun rakenne. Tähdellä merkityt sarakkeet vaaditaan rivin luomiseen.

Sarakkeen nimi	**Sarakkeen tyyppi**	**Sarakkeen selite**	**Sarakkeen tarkoitus**
Primary_ID *	Auto-number	Pääavain ja rivin päätunniste, automaattinen	Toimia päätunnisteena riville
Parent_ID	Numero	Tunniste, jolla yhdistää rivi aiemman rivin alle	Kertoa, minkä työlajin työ tai työn suorite on kyseessä
Dimension_UID	Numero	Kuusinumeroinen tunniste, identtinen työajanseurannan vastaavan tunnisteen kanssa	Yhdistää rivi työajanseurantakirjauksiin raportilla
SLA_ID	Numero	Työtehtäväkohtainen SLA-tunniste	Kertoa työn palvelutasosopimus
Title *	Teksti	Työn, tehtävän tai suoritteen nimike	Kertoa työlajin, -tehtävän tai -suoritteen nimike
InUse	Boolean	Rivin voimassaolon kertova tieto	Kertoa, onko rivi aktiivinen vai passivoitu
DateAdded	Päivämäärä	Työn, tehtävän tai suoritteen lisäyspäivämäärä	Kertoa, milloin rivi on otettu käyttöön
DateRemoved	Päivämäärä	Työn, tehtävän tai suoritteen passivointipäivämäärä	Kertoa, milloin rivi on poistettu aktiivisesta käytöstä

Palvelutasosopimustaulu toimii käytännössä yksinkertaisena selitetauluna: taulussa on vain neljä saraketta (taulukko 2), jotka kertovat eri SLA-tyyppien enimmäispäivämäärät sopimuksen täyttymiselle ja sen, milloin SLA on astunut

voimaan (ja mahdollisesti passivoitu). SLA:n ylittymistä seurataan vain Power BI -raportilla vastaisuudessa.

Taulukko 2. Palvelutasosopimustaulun rakenne. Tähdellä merkityt sarakkeet vaaditaan rivin luomiseen.

Sarakkeen nimi	**Sarakkeen tyyppi**	**Sarakkeen selite**	**Sarakkeen tarkoitus**
Primary_ID *	Autonumber	Pääavain ja rivin päätunniste, automaattinen	Toimia päätunnisteena riville
DayAmount *	Numero	Enimmäispäivämäärä SLA:n toteutumiselle	Kertoa enimmäispäivämäärä SLA:n toteutumiselle
DateStarted	Päivämäärä	SLA-tyypin lisäyspäivämäärä	Kertoa, milloin SLA on otettu käyttöön
DateEnded	Päivämäärä	SLA-tyypin passivointipäivämäärä	Kertoa, milloin SLA on poistettu aktiivisesta käytöstä

3.2.2 Työtehtävä- ja SLA-kirjaustaulut

Sovelluksessa tehtäviä työkirjauksia varten on kaksi erillistä taulua, joihin tiedot tallennetaan: työkirjaus- ja SLA-kirjaustaulu. SLA-kirjaukset eli tehtävä- ja kirjauskohtaiset SLA-tiedot viedään erilliseen tauluun tietojen erottelun helpottamista varten myöhemmin Power BI -raportilla.

Työtehtäväkirjaustaulu koostuu yhdeksästä eri sarakkeesta (taulukko 3). Tauluun vietäviin riveihin kirjataan tehtäväkohtaisesti suoritesarakkeet, kirjaus- ja luontipäivämäärä, kirjauksen tekijä ja työtehtävän tunnistetiedot. Työtehtävätaulun rakenteen takia tehtävät, joista seurataan useampaa kuin yhtä suoritetta, kirjataan omina työkirjauksinaan, minkä takia riveissä käytetään viiteavainta työtehtävätauluun. Viiteavaimen avulla saadaan haettua kirjatun työn nimike, vaikka kirjattava työ itsessään olisi vain haettavan työn ”sivut”-suorite.

Taulukko 3. Työkirjaustaulun rakenne. Tähdellä merkityt sarakkeet vaaditaan rivin luomiseen.

Sarakkeen nimi	**Sarakkeen tyyppi**	**Sarakkeen selite**	**Sarakkeen tarkoitus**
Primary_ID *	Autonumber	Pääavain ja rivin päätunniste, automaattinen	Toimia päätunnisteena riville
Work_ID *	Numero	Kirjatun työn tunniste, työtehtävätaulun *Primary_ID*	Kertoa kirjattu työtehtävä, asetetaan sovelluksessa
Amount	Numero	Kirjattavan rivin kappalemäärä	Kertoa kirjatun suoritteen määrä
OldestDate	Päivämäärä	Kirjattavan työn vanhimman käsitellyn materiaalin saapumispäivämäärä	Kertoa rivin vanhin saapumispäivämäärätieto
TaskDate *	Päivämäärä	Työn kirjauspäivämäärä	Kertoa, mille päivämäärälle työ on kirjattu
UserEmail *	Sähköposti	Rivin kirjanneen työntekijän sähköpostiosoite	Kertoa kirjauksen luonut henkilö
TyotRelaatio	Viiteavain	Viiteavain työtauluun	Hakea työtaulusta tehtävän nimike *Primary_ID*:n tai *Parent_ID*:n kautta
DateCreated *	Päivämäärä	Rivin luontipäivämäärä	Kertoa, milloin kirjaus on luotu
DateEdited	Päivämäärä	Rivin muokkauspäivämäärä, vakiona sama kuin luontipäivämäärä	Kertoa, milloin kirjausta on muokattu sen luomisen jälkeen

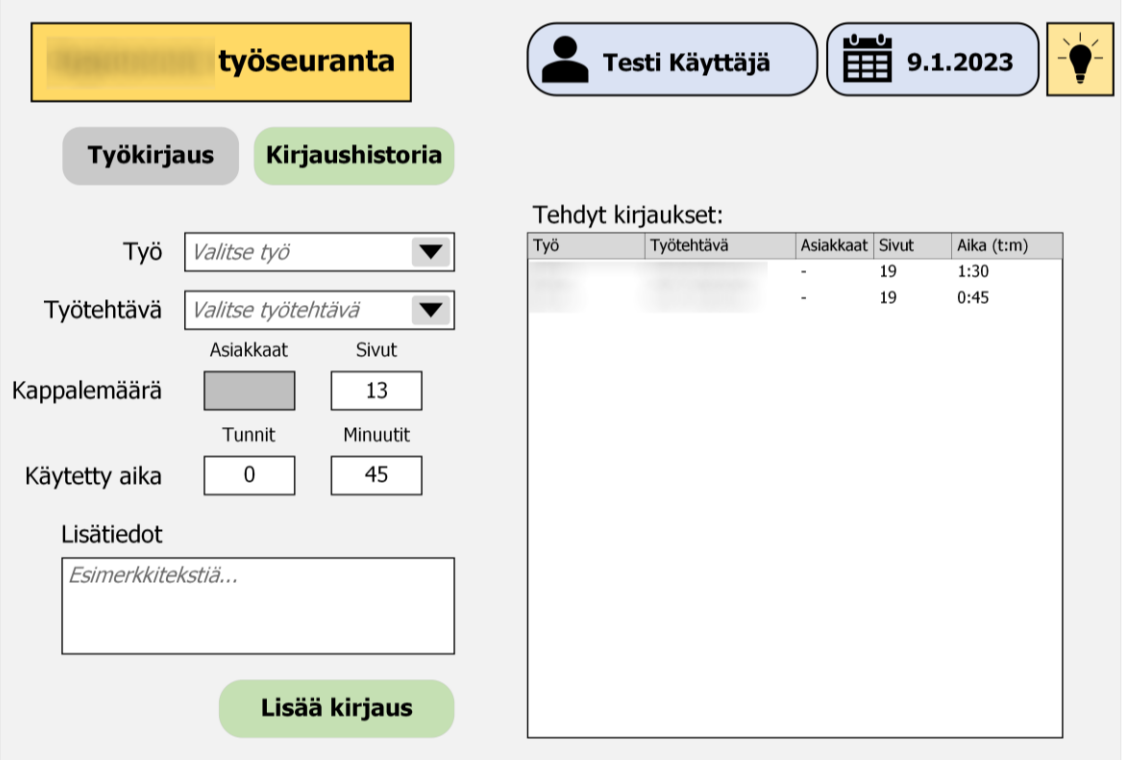
SLA-kirjaustaulussa on kuusi eri saraketta, joista jokainen on pakollinen (taulukko 4). Tauluun viedään kirjaus aina niiden työkirjausten yhteydessä, joista seurataan palvelutasosopimusta. Töistä, joista seurataan useampaa kuin yhtä suoritetta, viedään kuitenkin vain yksi SLA-kirjaus.

Taulukko 4. Palvelutasosopimuskirjaustaulun rakenne. Kaikki taulun sarakkeet vaaditaan rivin luomiseen.

Sarakkeen nimi	**Sarakkeen tyyppi**	**Sarakkeen selite**	**Sarakkeen tarkoitus**
Primary_ID	Autonumber	Pääavain ja rivin päätunniste, automaattinen	Toimia päätunnisteena riville
Tyo_ID	Numero	Kirjattavan työn tunniste	Kertoa, mille työtehtävälle SLA-kirjaus kuuluu
OldestDate	Päivämäärä	Kirjattavan työn vanhimman käsitellyn materiaalin saapumispäivämäärä	Kertoa rivin vanhin saapumispäivämäärätieto
TaskDate	Päivämäärä	SLA:n kirjauspäivämäärä	Kertoa, mille päivämäärälle työ on kirjattu
SLARelaatio	Viiteavain	Viiteavain SLA-tauluun	Hakea SLA-taulusta oikea SLA-tyyppi työn *SLA_ID*:n perusteella
UserEmail	Sähköposti	Rivin kirjanneen työntekijän sähköpostiosoite	Tunnistaa kirjauksen tehnyt henkilö

3.3 Sovelluksen suunnittelu ja kehittäminen Power Appsissa

Sovelluksen alustavaan ulkoasun suunnitteluun käytettiin projektin alussa Microsoft Visio -luonnostelutyökalua (kuva 10) ja referenssimateriaalina yrityksen muiden palvelujen Power Apps -sovelluksia sekä aiempaa työsuoriteseurantaa. Visuaalisessa ilmeessä oli oleellista huomioida yrityksen käyttämä väripaletti, mutta muuten tyyli oli pitkälti vapaa ja priorisointi keskittyi selkeän käyttäjäkokemuksen tuottamiseen, niin toiminnallisuuden kuin visuaalisen puolenkin osalta.



Kuva 10. Microsoft Visio -luonnosteluohjelmalla luotu hahmotelma mahdollisesta työsuoriteseurantanäkymästä. Työajanseurannan poisjättämisestä ei ollut vielä täysin selkeyttä, minkä takia se on huomioitu luonnoksessa.

Sovelluksen lopullinen layout muodostui melko varhaisessa vaiheessa kehitystä. Sovellus koostui kahdesta välilehdestä, työseurannasta ja kirjaushistoriasta, ja sovelluksen käynnistäessä aukeaa näistä ensimmäinen. Työseurantavälilehti sisälsi neljä osaa: työlajit, työtehtävät, työn kirjausnäkymä ja tehdyt kirjaukset (kuva 11).

Kuva 11. Työseurantavälilehden eri osat (vasemmalta oikealle): työlajit, työtehtävät, työn kirjausnäkymä ja tehdyt kirjaukset.

3.3.1 Tehtävien suodattaminen ja hakeminen

Työtehtävät suodatetaan sovelluksessa näkyviin TaskGallery-nimisen Gallery-kontrollin Items-funktiossa. Sovelluksen käynnistyessä alustetaan FilteredTasks -niminen muuttuja, jonka arvoksi asetetaan vakiona 0. Käyttäjän painaessa työlajipainiketta annetaan ko. muuttujalle arvo 1–3 valitun työlajin mukaan painikkeen OnSelect-funktiossa (esimerkkikoodi 1).

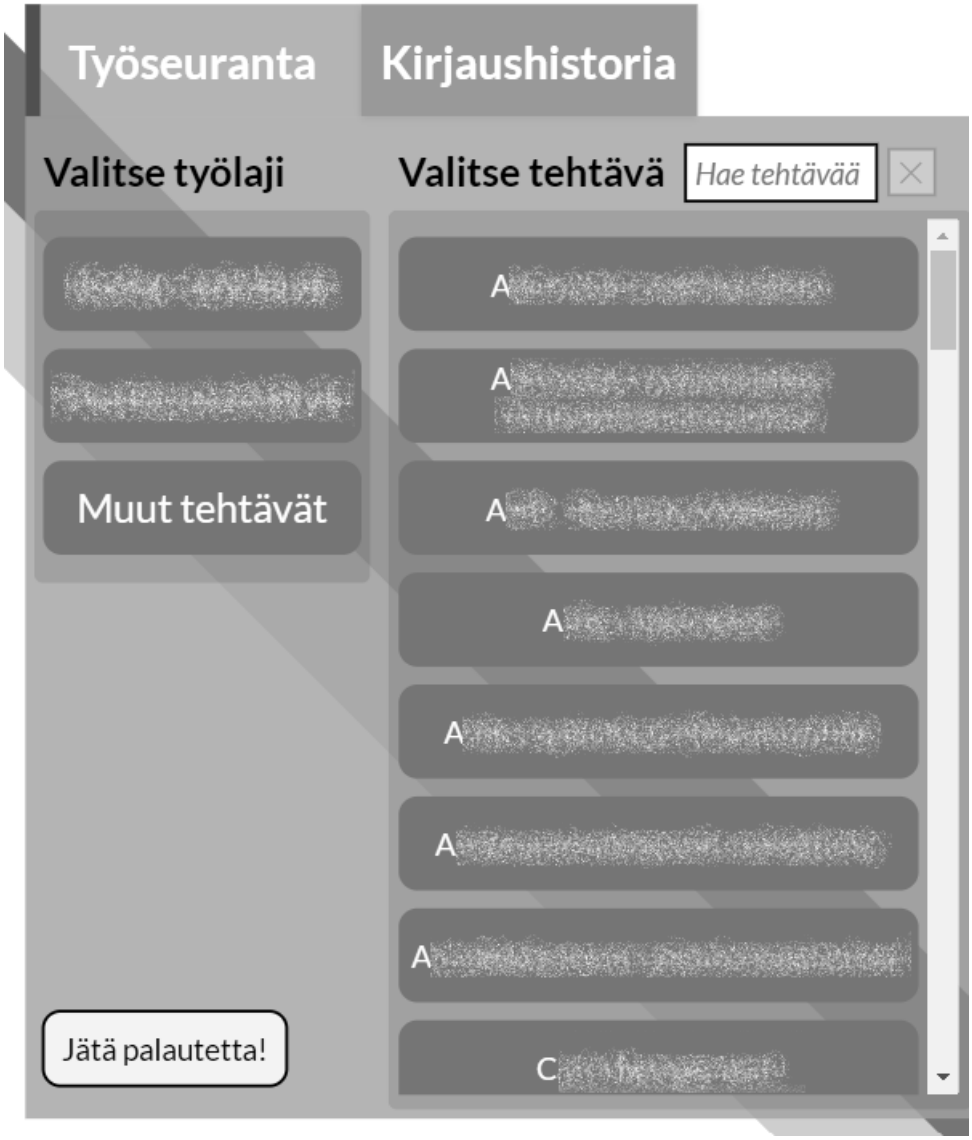
```
If(
     (!WorkCat1Chosen || (WorkCat2Chosen || WorkCat3Chosen)),

     Set(WorkCat1Chosen, true) && Set(FilteredTasks, 1) &&
     Set(WorkCat2Chosen, false) &&
     Set(WorkCat3Chosen, false) &&
     Set(LogFormVisibility, false),

     Set(WorkCat1Chosen, false) && Set(FilteredTasks, 0) &&
     Reset(TaskGallery) && Set(LogFormVisibility, false)
)
```

Esimerkkikoodi 1. Ehtolauseessa tarkistetaan, onko ko. työlaji valittu ja jos ei ole, muutetaan FilteredTasksin arvoa ja asetetaan työlaji aktiiviseksi, muutoin nollataan FilteredTasks ja tyhjennetään työlajivalinta.

Items-funktio hakee työt vertaamalla FilteredTasksin arvoa TaskGalleryyn liitetyn työtehtävätaulun Parent_ID-sarakkeeseen ja suodattaa aktiiviset tehtävät näkyviin (esimerkkikoodi 2). Tehtävälista tulostuu aakkosjärjestyksessä tehtävien nimikkeiden mukaisesti (kuva 12). Listan yllä oleva hakukenttä toimii siten, että tehtävälistaa suodatetaan Items-funktiossa reaaliajassa hakukenttään syötettävän tekstin mukaan. Sama FilteredTasks-muuttuja on huomioitu myös hakukentässä, eli jos työlaji on valittu, haku suoritetaan vain valitun työlajin alle kuuluviin tehtäviin.

```
Sort(
     Search(
          Filter(
               Tyotaulu,
               Parent_ID = FilteredTasks,
               InUse
          ),
          TaskSearch.Text,
          "crabd_title"
     ),
     Title
)
```

Esimerkkikoodi 2. Työtehtävät suodattuvat Parent_ID:n eli työlajin mukaan. Search-funktion sisällä oleva TaskSearch.Text on hakukenttään syötetty teksti, ja hakua vastaavat tehtävät tulostuvat näkyviin aakkosjärjestyksessä.



Kuva 12. Työtehtävälista, kun työlajia ei ole valittu, jolloin kaikki työt ovat listattuna aakkosjärjestyksessä.

3.3.2 Kirjauksen tekeminen

Kun käyttäjä on valinnut kirjattavan työtehtävän työseuranta-välilehdellä, kirjausnäkymän eri kentät asetetaan näkyviin tehtävän edellyttämien tietojen mukaan ja tehtävän Primary_ID-tunniste tallennetaan CurrentTaskID-nimisen muuttujan arvoksi. Käyttäjän sähköpostiosoite ja työtehtävän nimi tulevat vakiona näkyviin työstä riippumatta. Kappalemääräkentät ovat omissa säiliöissään (engl. container) piilotettuna, ja niistä aktivoidaan ne, joita valitussa tehtävässä

seurataan (esimerkkikoodi 3). Vastaavaa logiikkaa käytetään myös vanhimman päivämäärän kentän kohdalla.

```
If(
     IsBlank(
          LookUp(
               Tyotehtavataulu,
               (Parent_ID = Value(CurrentTaskID) &&
               (Title = "Sivut" || Title = "Kappaleet"))
          )
     ),
     false,
     true
)
```

Esimerkkikoodi 3. Sivu- tai kappalemääräkentän sisältävän säiliön Visible-ominaisuus, jonka arvo määrittyy tarkistamalla, seurataanko valitussa työtehtävässä (CurrentTaskID) ko. suoritetta.

Kirjaus tehdään LogButton-nimisen painikekontrollin OnSelect-funktiossa. Työkirjaus viedään kirjaustauluun käyttäen Patch-funktiota; Patchille annetaan parametreinä kohdetaulun nimi ja Defaults-funktio, joka määrittää, että kyseessä on uusi kirjaus tauluun. Defaults-funktiolle annetaan myös parametriksi sama taulu. Kirjattaviin sarakkeisiin syötetään kirjausnäkymän kentistä haetut arvot (esimerkkikoodi 4).

```
Patch(Tyokirjaustaulu, Defaults(Tyokirjaustaulu),
{
     Work_ID     : SivutKplID,
     Amount      : Value(SivutKplInput.Text),
     OldestDate  : OldestDateInput.SelectedDate,
     TaskDate    : comp_CompanyHeader_T1.ReportingDate +
                   Time(Hour(Now()), Minute(Now()),Second(Now())),
     UserEmail   : EmailBox.Text,
     DateCreated : Now(),
     DateEdited  : Now(),

     TyotRelaatio: LookUp(Tyotaulu, CurrentTaskID = Value(Primary_ID))
}
```

Esimerkkikoodi 4. Patch-funktio sivu- tai kappalemäärän sisältävän työn kirjaamiseen. TaskDate-sarakkeeseen tallennetaan kirjausaika, jotta käyttäjä näkee sen rivin yhteydessä. Alimpaan sarakkeeseen haetaan LookUp-kyselyllä tehtävän päätunniste, jotta saadaan työn nimike näkyviin kirjauksen yhteydessä.

OnSelect-funktiossa on neljä erilaista Patch-funktiota ehtolauseiden sisällä, ja kirjattavan tehtävän seurattavat suoritteet määrittävät, mitä niistä kutsutaan. Työtehtävä, josta seurataan SLA-tietoa ja kahta eri suoritetta (esim. sivut ja asiakkaat), vaatii kolme eri Patchia: SLA-, sivumäärä- ja asiakasmääräkirjauksen. CurrentTaskID viittaa aina valitun työtehtävän Primary_ID-tunnisteeseen, minkä takia ko. tehtävän suoritteiden, kuten asiakasmäärän, Primary_ID-tunniste tulee hakea erikseen työtehtävätaulusta LookUp-kyselyllä ja tallentaa omaan muuttujaansa ennen kirjauksen tekoa (esimerkkikoodi 5). Kirjauksen tallennuttua funktiossa nollataan kirjausnäkymän syötekentät sekä CurrentTaskID ja CurrentTaskText-muuttujat.

```
If(
     cont_Asiakkaat.Visible,
     Set(
          AsiakkaatID,
          Value(
               LookUp(
                    Tyotehtavataulu, Parent_ID = CurrentTaskID &&
                    Title = "Asiakkaat"
               ).Primary_ID
          )
     )
)
```

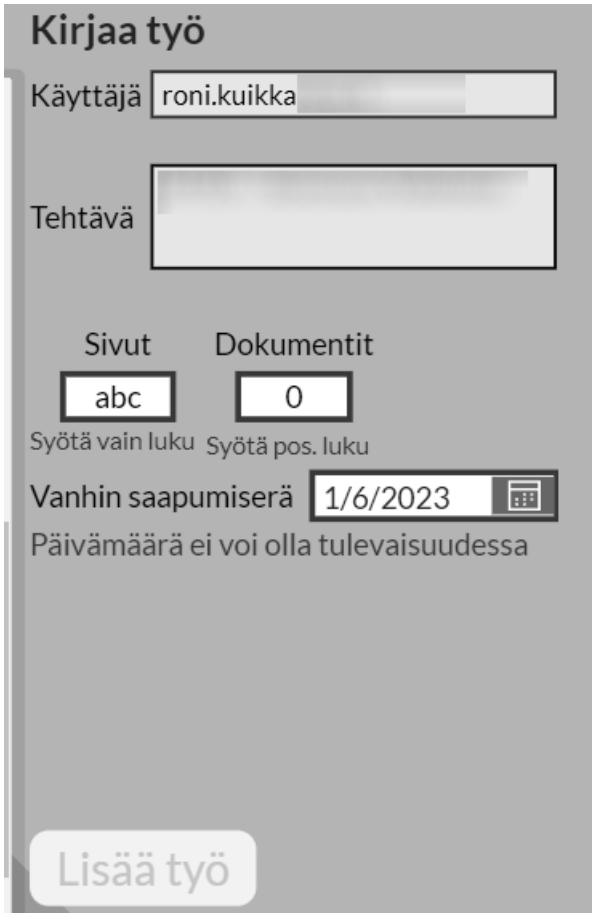
Esimerkkikoodi 5. Kirjauspainikkeen OnSelect-funktiossa luodaan AsiakkaatID-muuttuja, jolle haetaan oikea tunniste, jotta se voidaan syöttää Work_ID-sarakkeeseen Patch-funktiossa.

Jos syötekenttien alapuolella ilmenee virheviestejä tai vaadittuja kenttiä ei ole täytetty, kirjauspainikkeen DisplayMode-funktiossa asetetaan tilaksi ”DisplayMode.Disabled”. Tällöin painiketta ei voi painaa eikä kirjausta viedä tauluun. DisplayMode-funktion virhetarkistukset toimivat IsBlank-tarkistuksilla, joille annetaan tarkistettavaksi joko vakiona piilotettujen virheilmoitusten tekstiarvoja tai kirjausnäkymän syötekenttien arvoja (esimerkkikoodi 6). Lisäksi, jos kirjauspäivämäärä on tulevaisuudessa, kirjauksen tekeminen evätään.

```
If(
     !IsBlank(SivutKplError.Text) || !IsBlank(DokumentitError.Text) ||
     !IsBlank(AsiakkaatError.Text) || !IsBlank(EmailError.Text) ||
     comp_CompanyHeader_T1.ReportingDate > Today() ||
     (IsBlank(SivutKplInput) && cont_SivutKpl.Visible) ||
     (IsBlank(DokumentitInput) && cont_Dokumentit.Visible) ||
     (IsBlank(AsiakkaatInput) && cont_Asiakkaat.Visible) ||
     (IsBlank(OldestDateInput) && cont_OldestDateRow.Visible) ||
     TaskCategoryBox.Text = "",

     DisplayMode.Disabled,
     DisplayMode.Edit
)
```

Esimerkkikoodi 6. Kirjauspainikkeen DisplayMode-funktio. Useat IsBlank-tarkistukset varmistavat, ettei puutteellista tai väärin täytettyä kirjausta tehdä.

Virheilmoitukset tulevat näkyviin käyttäjän täyttäessä kirjattavan tehtävän edellyttämät kohdat väärällä tavalla, esim. syöttämällä kappalemääräkohtaan negatiivisen luvun tai kirjaimia (kuva 13).



Kuva 13. Kirjausnäkymän virheilmoituksia, jotka eväävät kirjauksen tekemisen.

Virheilmoitukset ovat oletuksena tyhjiä tekstilaatikoita, mutta tekstiksi asetetaan virheviesti ehtolauseen havaitessa virheellisen syötteen. OldestDateError-niminen virheilmoitus esim. tarkistaa, onko kirjausnäkymään syötetty vanhin päivämäärä -kenttään päivä, joka on tulevaisuudessa tai myöhemmin kuin työn kirjauspäivämäärä (esimerkkikoodi 7).

```
Coalesce(
     If(
          OldestDateInput.SelectedDate > Today(),
          “Vanhin päivämäärä ei voi olla tulevaisuudessa.”
     ),
     If(
          OldestDateInput.SelectedDate >
          comp_CompanyHeader_T1.ReportingDate,
          “Kirjauspäivämäärä ei voi olla ennen vanhinta päivämäärää.”
     )
)
```

Esimerkkikoodi 7. OldestDateError-tekstilaatikon Text-arvo, johon ilmestyy virheviesti ehtolauseen toteutuessa. Coalesce-funktio määrittää järjestyksen, jossa ehtolauseet käydään läpi (ylhäältä alas).

3.3.3 Kirjausten hakeminen

Tehdyt kirjaukset haetaan työseuranta-välilehdellä ruudun yläosasta löytyvästä DatePickerista valitun kirjauspäivämäärän mukaan. Työkirjaustaulu on liitetty LoggedGallery-nimiseen Gallery-kontrolliin, ja sen Items-funktiossa haetaan sovellusta käyttävän käyttäjän tekemät kirjaukset ko. päivältä (esimerkkikoodi 8).

```
Sort(
     Filter(
          Tyokirjaustaulu,
          Date(
               Day(TaskDate),
               Month(TaskDate),
               Year(TaskDate)
               ) =
          Date(
               Day(comp_CompanyHeader_T1.ReportingDate),
               Month(comp_CompanyHeader_T1.ReportingDate),
               Year(comp_CompanyHeader_T1.ReportingDate)
               ),
          UserEmail = CurrentUser
          ),
     Primary_ID
)
```

Esimerkkikoodi 8. Tehdyt kirjaukset haetaan vertaamalla TaskDate-sarakkeeseen tallennettua päivämäärää valittuun kirjauspäivämäärään. Suoraa vertausta tulee välttää, koska kirjatut kellonajat mitä luultavimmin eroavat toisistaan.

Kirjaushistoria-välilehdellä voi hakea useamman päivän kirjaukset näkyviin, mutta muuten toiminnallisuus on identtinen työseuranta-välilehden kirjausten listauksen kanssa. Kirjaushistoria-välilehdellä asetetaan alku- ja loppupäivämäärä ruudun alaosasta löytyviin DatePicker-kontrolleihin, ja välilehden Gallery-kontrollin Items-funktio hakee käyttäjän työkirjaukset määritetyltä aikaväliltä.

3.3.4 Kirjausten muokkaaminen

Kirjaus muokataan SaveChanges-nimisen painikekontrollin OnSelect-funktiossa. Käyttäjän valittua kirjauksen jommaltakummalta välilehdeltä se annetaan ChosenRecord-nimisen muuttujan arvoksi. Kirjauksen avautuessa muokkausikkunaan sen tiedot asetetaan muokattavaksi sarakekohtaisiin syötekenttiin. Kirjauksen alkuperäistä luontipäivämäärää, työtehtävää tai kirjauksen tehnyttä käyttäjää ei voi muokata. Muutokset kirjaukseen tehdään Patch-funktiolla (esimerkkikoodi 9). Jos muutoksia ei tehty, kirjausta ei voida myöskään päivittää.

```
Patch(
     Tyokirjaustaulu,
     LookUp(Tyokirjaustaulu, Primary_ID = ChosenRecord.Primary_ID),
     {
          TaskDate       : LogDateInput_Edit_T1.SelectedDate +
                               Time(
                                    Hour(ChosenRecord.TaskDate),
                                    Minute(ChosenRecord.TaskDate),
                                    Second(ChosenRecord.TaskDate)
                               ),
          Amount         : Value(AmountInput_Edit_T1.Text),
          OldestDate     : OldestDateInput_Edit_T1.SelectedDate,
          DateCreated    : ChosenRecord.DateCreated,
          DateEdited     : Now()
     }
)
```

Esimerkkikoodi 9. Patch-funktio, jolla kirjauksen muokkaus toteutetaan.

Kirjauksia saatetaan tallentaa kahteen eri sijaintiin, työkirjaus- ja SLA-kirjaustauluun, joten kirjauksia muokatessa tulee tarkistaa, onko myös SLA-kirjaus olemassa (esimerkkikoodi 10). SLA-kirjauksille tehdään vastaavanlainen päivitys Patch-funktiolla.

```
LookUp(
     SLAKirjaustaulu,
     Tyo_ID = Value(ChosenRecord.TyotRelaatio.Primary_ID) &&
     (
          Date(Day(TaskDate), Month(TaskDate), Year(TaskDate))
          =
          Date(
               Day(ChosenRecord.TaskDate),
               Month(ChosenRecord.TaskDate),
               Year(ChosenRecord.TaskDate)
          )
     ) &&
     (
          Date(Day(DateCreated), Month(DateCreated), Year(DateCreated))
          =
          Date(
               Day(ChosenRecord.DateCreated),
               Month(ChosenRecord.DateCreated),
               Year(ChosenRecord.DateCreated)
          )
     ) &&
     OldestDate = ChosenRecord.OldestDate &&
     UserEmail = ChosenRecord.UserEmail
)
```

Esimerkkikoodi 10. LookUp-kysely, jolla tarkistetaan, onko muokattavalla kirjauksella SLA-kirjausvastinetta.

3.3.5 Kirjausten poistaminen

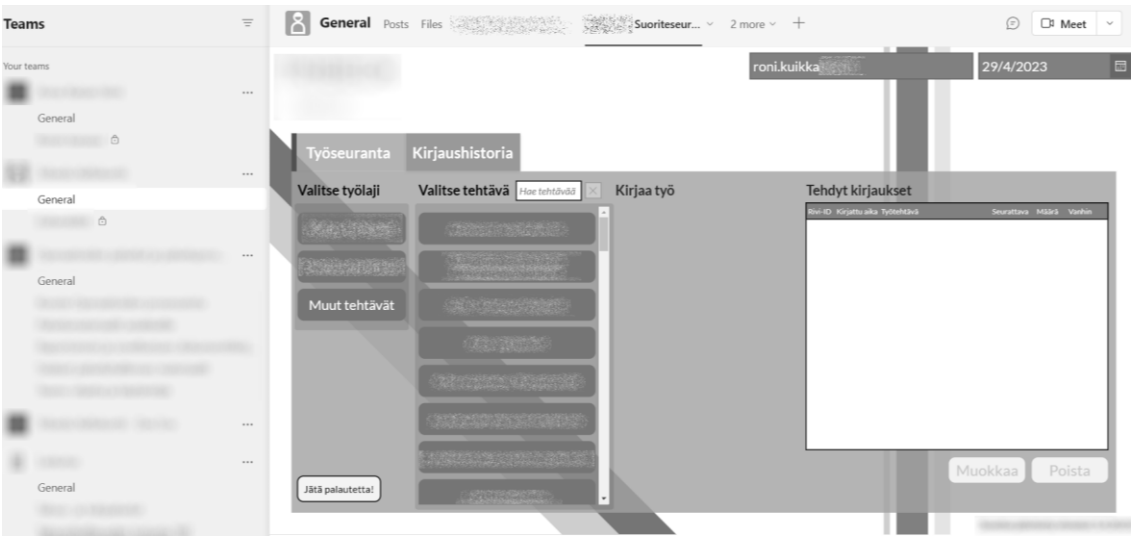
Kuten kirjausta muokatessakin, poistaessa tulee ottaa huomioon, onko kirjauksesta tehty sekä työ- että SLA-kirjaus, ja poistaa molemmat, jos on. Tätä varten kirjauksen poistavan painikekontrollin OnSelect-funktiossa on samanlainen ehtolause kuin muokkauspainikkeessakin. Ehdon täyttyessä suoritetaan myös SLA-kirjauksen poisto Remove-funktiolla ennen työkirjauksen poistamista. Funktiolle annetaan kaksi parametriä: taulu, josta kirjaus poistetaan, ja poistettava rivi, tässä tapauksessa ChosenRecord-muuttujan arvo eli valittu kirjaus. Työkirjausrivin poistamisen yhteydessä nollataan ChosenRecord, nollataan kirjauslista ja suljetaan kirjauksen poistoikkuna (esimerkkikoodi 11).

```
Remove(Tyokirjaustaulu, ChosenRecord);

Reset(LoggedGallery);
Set(LogRowSelected, false);
UpdateContext({BackgroundDimmed : false});
UpdateContext({DeletePopupVis   : false});
```

Esimerkkikoodi 11. Työkirjauksen poistaminen ja jälkitoimenpiteet.

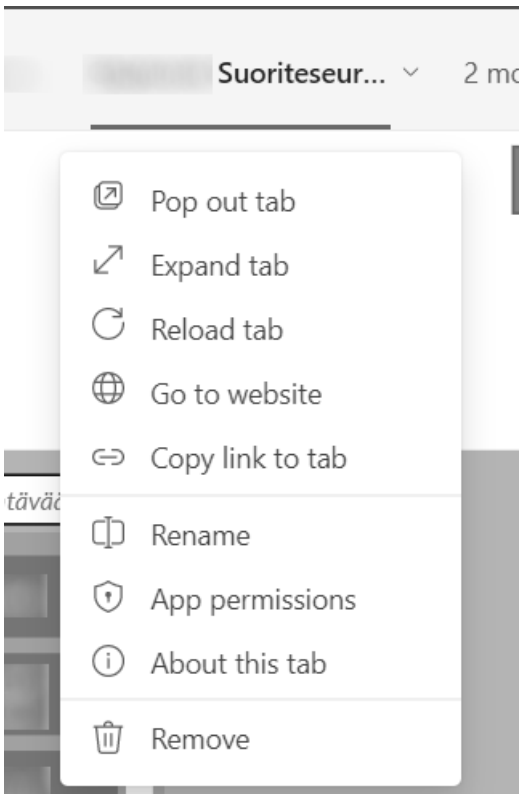
4 Tavoiteltu käyttäjäkokemus sovelluksessa

Käyttäjä avaa sovelluksen työajan- ja työsuoriteseurannalle osoitetun Teams-ympäristön kautta sovelluksen omalta välilehdeltä (kuva 14). Käynnistyessään sovellus hakee senhetkisen käyttäjän tiedot ja asettaa kirjauspäivämäärän vakiona nykyhetken mukaan. Sovellus on heti käyttövalmis.



Kuva 14. Työsuoriteseuranta Power Apps -sovellus avattuna Teamsissa.

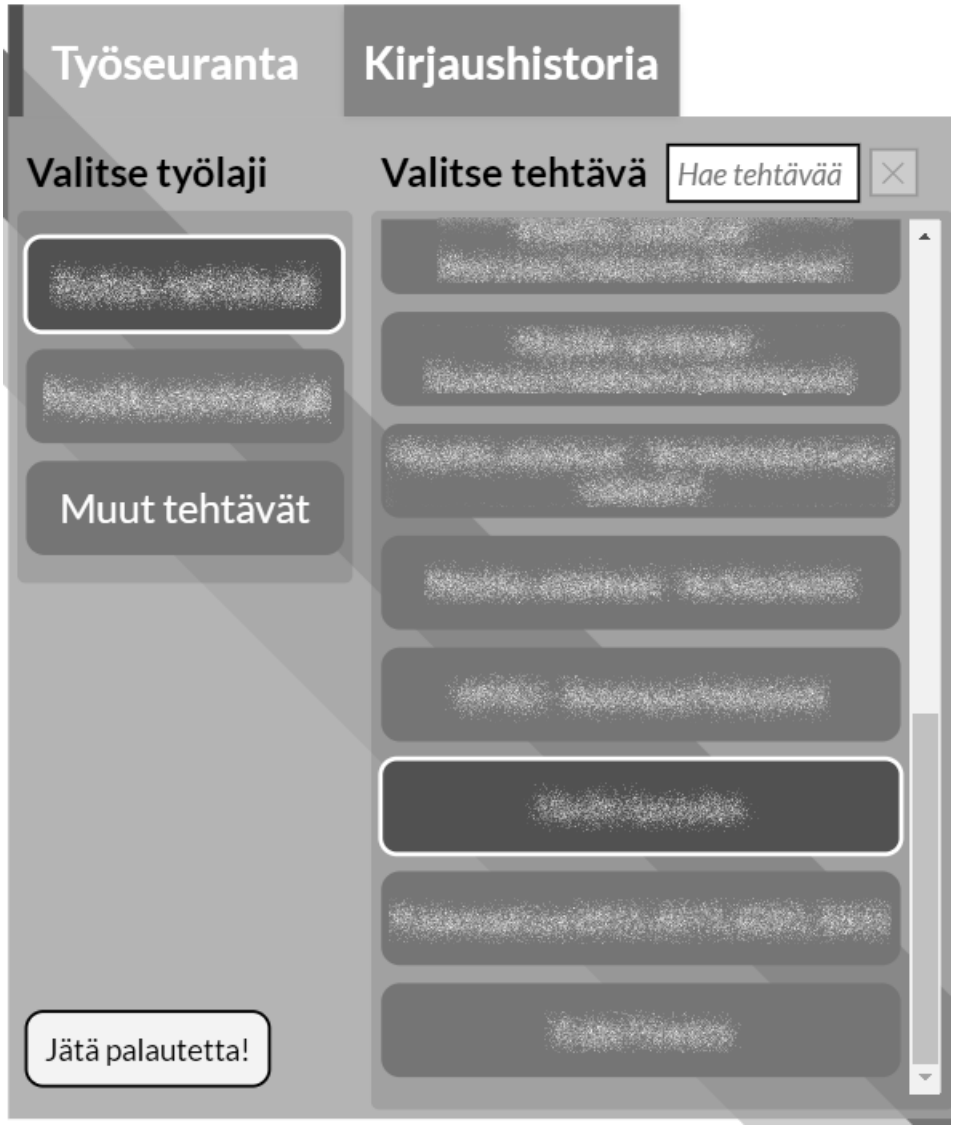
Käyttäjä voi myös avata sovelluksen erilliseen ikkunaan, jolloin se ei tule esim. Teams-keskustelujen tielle (kuva 15).



Kuva 15. "Pop out tab" -valintaa painamalla käyttäjä saa avattua sovelluksen omaan ikkunaan.

4.1 Tehtävien suodattaminen ja hakeminen

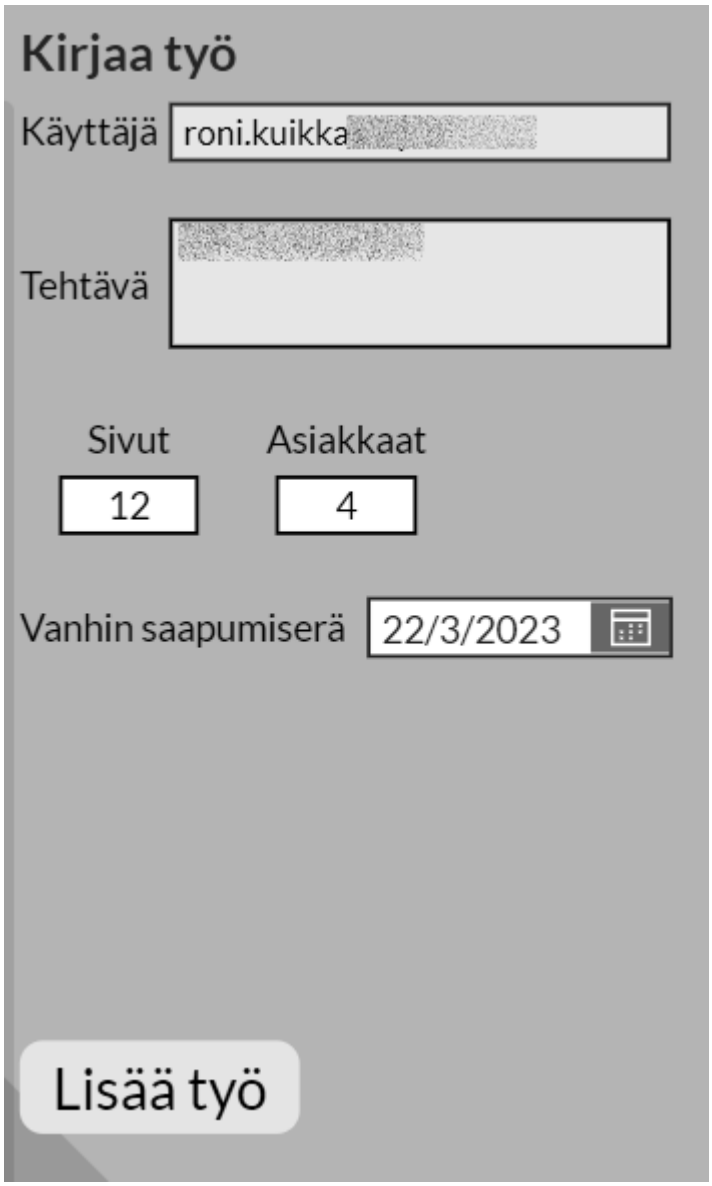
Työseuranta-välilehdellä ovat sovelluksen avautuessa oletuksena kaikki palvelun työtehtävät listattuna ja haettavissa. Hakukenttä toimii reaaliaikaisesti eikä vaadi erillistä painallusta hakeakseen tehtäviä syötetyillä hakusanoilla. Käyttäjän painaessa työlajipainiketta tehtävälista suodatetaan valitun työlajin mukaisesti ja hakukenttä hakee tällöin vain valitun työlajin joukosta hakutuloksia (kuva 16). Vain yksi työlaji voi olla valittuna kerrallaan.



Kuva 16. Työtehtävälista suodatettuna valitun työlajin mukaan.

4.2 Kirjauksen tekeminen

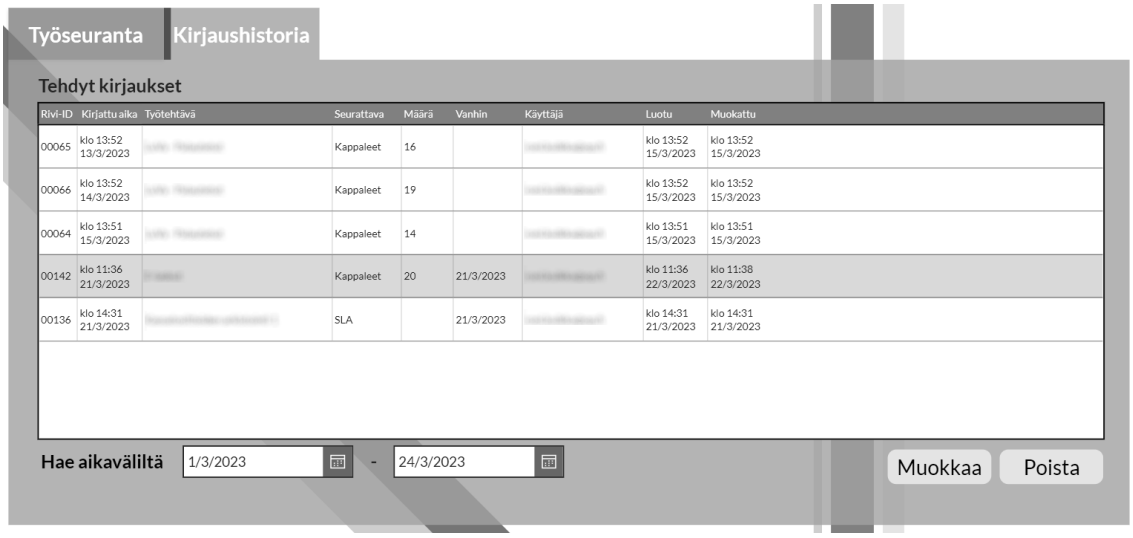
Käyttäjän valittua työtehtävän listalta ko. tehtävä asetetaan kirjattavaksi kirjausnäkymään. Kirjauksen tallentamiseen edellytetään, että kaikki tehtävän vaatimat syötekentät on täytetty oikeassa muodossa, muutoin käyttäjä ei voi kirjata työtä. Käyttäjän valitsemassa tehtävässä seurataan SLA:ta sekä sivu- ja asiakasmäärää. Käyttäjä täyttää syötekentät oikein, jolloin kirjausnappi on painettavissa (kuva 17).



Kuva 17. Tehtävän tiedot täytettynä kirjausnäkymässä valmiina kirjattavaksi.

4.3 Kirjausten hakeminen

Kirjaushistoria-välilehdellä käyttäjä voi asettaa alku- ja loppupäivämäärän ruudun alaosasta löytyvillä DatePicker-kontrolleilla. Kontrolleilla rajataan aikaväli, jolta kirjaukset haetaan (kuva 18). Kontrolleissa on vakioarvona senhetkinen päivämäärä. Kirjauksia ei haeta, jos käyttäjä asettaa aloituspäivämäärän yli kuukauden menneisyyteen tai myöhemmäksi kuin loppupäivämäärän. Virheilmoituksen ilmetessä kirjausnäkymä on tyhjä. Kirjausnäkymä on toiminnallisuudeltaan samanlainen työseuranta-välilehden vastaavan näkymän kanssa, mutta kirjaushistoria-välilehden näkymässä ovat myös sarakkeet UserEmail, DateCreated ja DateEdited käyttäjän nähtävissä.

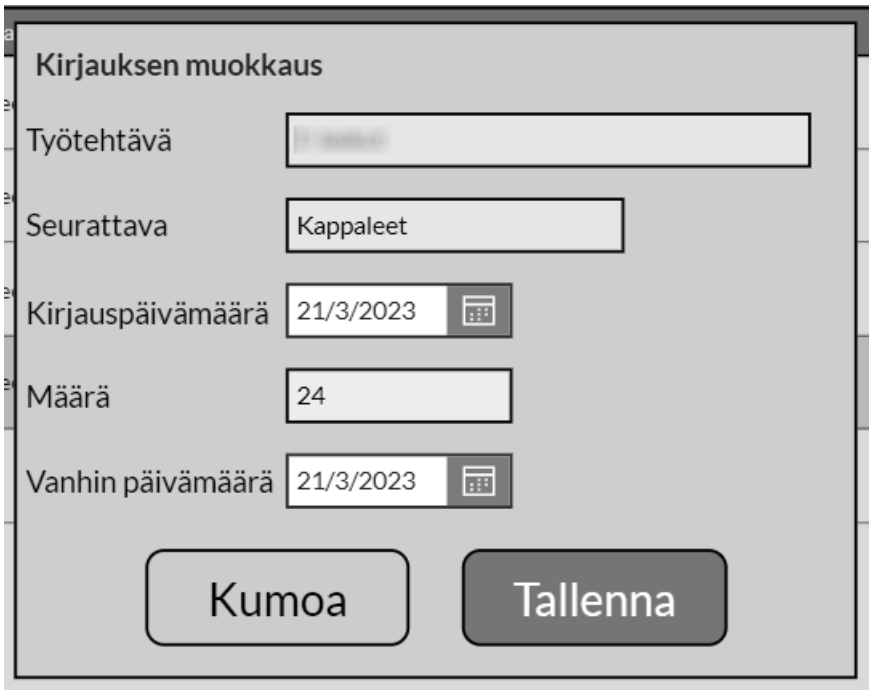


Kuva 18. Kirjaushistoria-välilehden tehdyt kirjaukset.

4.4 Kirjausten muokkaaminen

Käyttäjän valittua kirjauksen joko työseuranta- tai kirjaushistoria-välilehdellä kirjausten alla oleva muokkauspainike asetetaan päälle ja käyttäjä pystyy avaamaan kirjauksen muokattavaksi. Kirjaus avataan erilliseen ikkunaan (kuva 19), jossa käyttäjä pystyy muokkaamaan päivämääräkenttiä ja suoritteita päivittämällä niiden syötekenttiä. Käyttäjän muokatessa tietoja syötekentän taustaväri tummenee hieman, mikä selkeyttää, että tietoa on muokattu. Käyttäjä voi joko

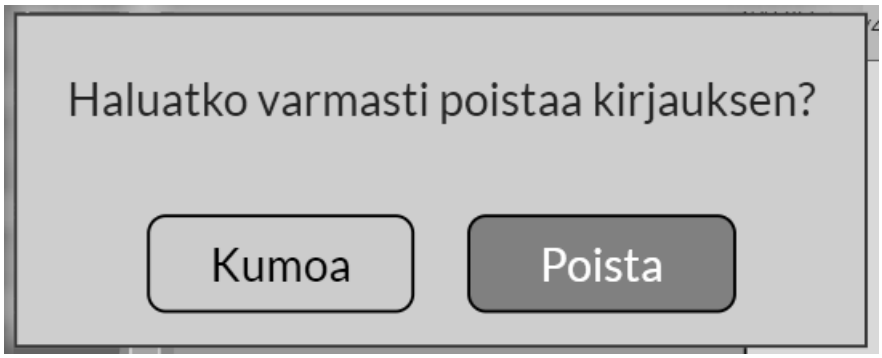
poistua ikkunasta painamalla Kumoa-nappia, jolloin valittu kirjaus nollataan, tai tallentaa kirjaukseen tehdyt muutokset Tallenna-nappia painamalla.



Kuva 19. Kirjauksen muokkausikkuna. Ikkunan ollessa ruudulla muu sovellus on lievästi tummennettuna taustalla.

4.5 Kirjausten poistaminen

Käyttäjän valittua kirjauksen, jonka hän haluaa poistaa joko työseuranta- tai kirjaushistoria-välilehdellä, sovellus asettaa kirjausten alla olevan poistopainikkeen aktiiviseksi. Käyttäjän painaessa painiketta aukeaa ruudulle erillinen ikkuna, jossa kysytään varmistusta ennen kirjauksen poistamista (kuva 20). Käyttäjä voi joko kumota poiston painamalla Kumoa-painiketta, jolloin valinta tyhjennetään ja kirjaus säilytetään taulussa, tai poistaa kirjauksen. Käyttäjä voi poistaa vain yhden kirjauksen kerrallaan.



Kuva 20. Kirjauksen poistovarmistus.

5 Työn pilotointi ja lopputulokset

5.1 Pilotti

Suoriteseurantasovelluksen kehitystyön lähestyessä loppusuoraa järjestettiin pilotti, joka alkoi maaliskuun puolivälissä ja kesti noin kahden viikon ajan. Pilottiin osallistui kolme testaajaa, joista jokainen oli täysipäiväinen työntekijä palvelussa, johon sovellus oli myös tarkoitus ottaa käyttöön. Pilotti aloitettiin pitämällä pienimuotoinen perehdytys sovelluksesta. Yleisen toimivuuden selvittämisen lisäksi pilotin tavoitteina oli kerätä palautetta työntekijöiltä päivittäiskäyttäjäkokemuksen osalta, kuten töiden valitsemisen ja hakemisen selkeydestä, ja mahdollisista puuttuvista ominaisuuksista ja työtehtävistä. Käyttäjät kirjasivat samat työkirjaukset uuteen sovellukseen kuin senhetkiseen työseurantaankin, ja täten saatiin kerättyä todenmukaista dataa myös kirjaustauluihin. Palautetta kerättiin sovellukseen upotetun sähköpostipalautelaatikon kautta (kuva 21) sekä testaajien, palvelun team leadin ja kehittäjän välisellä Teams-kanavalla.

Kuva 21. Sovelluksesta löytyvä palautelaatikko, jonka kautta käyttäjät voivat lähettää palautetta suoraan kehittäjän sähköpostiin.

Pilotin aikana testaajilla ei ilmennyt lainkaan sovelluksen toiminnallisuuteen vaikuttavia virheitä. Työtehtävien osalta ilmeni hieman epäselvyyttä siitä, onko tehtävälista puutteellinen, sillä aiemmassa työsuoriteseurannassa oli huomattavasti kattavampi lista työtehtäviä, ml. eri palveluiden töitä. Testaajien ja team leadin kanssa pidettyjen palaverien myötä kuitenkin selvisi, että suurin osa puuttuvista työtehtävistä ei ollut laskutettavia töitä eivätkä ne täten olleet suoritteiden seuraamisen kannalta oleellisia, joten niitä ei ollut tarvetta lisätä työtehtävälistaan. Ilmeni kuitenkin muutama puuttuva työ, joista tuli seurata palvelutasosopimuksia, ja kyseiset tehtävät päivitettiin työtehtävätauluun.

Vaikka pilotissa ei ilmennytkään suuria puutteita sovelluksessa, palautteen pohjalta tehtiin kuitenkin hienosäätöä ja pieniä korjauksia mm. työtehtävien listaus- ja suodatustapaan sekä kirjausnäkymän syötekenttien virhetarkistuksiin. Yksi tällaisista muutoksista oli mahdollisuus hakea tiettyä työtehtävää kaikkien tehtävien joukosta, jos käyttäjä ei ole vielä erikseen valinnut työlajia, jonka perusteella työtehtävät suodatetaan näkyviin. Sovelluksen käynnistäessä kaikki

työtehtävätaulusta löytyvät tehtävät ovat siis listattuna, kunnes käyttäjä rajaa ne valitsemalla työlajin.

5.2 Lopputulosten arviointia ja pohdintaa

Työsuoriteseurantasovellus toimi sovitun vaatimusmäärittelyn mukaisesti, ja palaute sovelluksesta oli positiivista. Uusi sovellus myös toimi nopeammin kuin edellinen työseuranta, erityisesti aiempia työkirjauksia selatessa ja käsitellessä. Ajoittain työtehtävien ja -kirjausten hakemisessa tauluista ilmeni pientä viivettä, mutta näin tapahtui kuitenkin verrattain harvoin.

Tiettyjä tapoja tarkistaa tehtäväkohtaisia seikkoja sovelluksessa olisi voinut suunnitella ja toteuttaa alusta alkaen paremmin. Erityisesti LookUp-kutsut, joilla haetaan tietoa Dataverse-tauluista tiettyjen funktioiden ehtolauseiden sisällä, aiheuttavat herkästi nk. delegaatiovaroituksia Power Appsissa. Nämä kohdistetuista tietokantakyselyistä aiheutuvat varoitukset voivat osoittautua mittavammaksi ongelmaksi vasta kirjaustaulujen täyttyessä tuhansista riveistä [15]. Väliaikainen ratkaisu tähän ongelmatilanteeseen on työkirjaustauluihin kytketty Power Automate -automaatio, jolla tallennetaan kirjaustietueet omiksi tiedostoikseen viikoittain. Siten virhetilanteiden ilmetessä on mahdollista hakea aiemmat kirjaukset sovelluksessa näistä erikseen tallennetuista .csv-tiedostoista ja tyhjentää aktiivisia tauluja riittävän kaukana menneisyydessä olevista työkirjauksista.

Kaiken kaikkiaan kehittäjä oli tyytyväinen tehtyyn työhön, koska hän pääsi onnistuneesti luomaan uuden työsuoriteseurannan juuri ko. yrityksen palvelukokonaisuudelle, jossa oli itse aiemmin työskennellyt samojen tehtävien parissa ja palvelun työnkuva oli entuudestaan tuttua hänelle. Kehittäjä koki, että hänellä oli intoa uudistaa tätä osaa palvelun työprosessista taustojensa ansiosta ja tämä auttoi myös kehitystyön aikana, sillä hänen oli helpompaa asennoitua sovellusta käyttävän työntekijän rooliin.

Suoriteseurantaa ylläpidetään ja mahdollisesti jatkokehitetään tulevaisuudessakin, sillä palvelussa mm. aloitetaan ajoittain uusia, niin projektiluontoisia kuin vakituisiksikin jääviä, työtehtäviä. Sovelluksen saavutettavuutta voidaan jatkossa edistää esim. lisäämällä vaihtoehtoisia väripaletteja käyttöliittymälle värisokeita työntekijöitä varten sekä useammalla käyttökielivaihtoehdolla, kuten englannin kielellä.

6 Yhteenveto

Insinöörityössä toteutettiin ja testattiin Power Apps -pohjainen työsuoriteseurantasovellus ja sovelluksen taustalla toimiva Dataverse for Teams -tietokantataulurakenne. Itse Power Apps -sovelluksen osalta työ täytti sille asetetut tavoitteet ja vaatimukset: sovelluksella voidaan valita haluttu työtehtävä, se kykenee mukauttamaan kirjausnäkymää valitun tehtävän edellyttämien suoritteiden mukaisesti, sillä voidaan tarkastella, muokata ja poistaa tehtyjä kirjauksia, ja se huomioi mahdolliset käyttäjän virheelliset tai puutteelliset syötteet. Perustoiminnallisuuden toimivuudesta huolimatta sovellukseen jäi kuitenkin jonkin verran parannettavaa ja hiottavaa, kuten raportin luvussa 5 mainitun delegaatiovaroitusongelman ratkominen.

Insinöörityön tavoitteisiin ja koottuun vaatimusmäärittelyyn nähden työ onnistui suunnitellusti. Toimeksiantajayrityksellä on tavoitteena ottaa työsuoriteseuranta käyttöön vuoden 2023 aikana, ja sovelluksen osalta se on jo mahdollista; erillisen Power BI -raportin ja sen käyttämän datamallin valmistuttua sovellus on käyttöönottovalmis palvelulle.

Power Appsiin perehtyminen oli suhteellisen suoraviivaista ja selkeää, sillä työkalut on tehty nopeasti sisäistettäviksi, mikä kulkee käsi kädessä Power Platformin vähäkoodisen periaatteen kanssa. Kehitystyön osalta Teamsin sisällä työskenteleminen oli ajoittain turhauttavaa, sillä pilvityökaluja käyttäessä esiintyy keskivertoa enemmän viivettä toimintojen suorittamisessa, tätä ilmeni erityisesti Dataverse for Teams -taulujen tietueiden päivittämisessä.

Lähteet

1 Carpi, Raffaele; Douglas, John & Gascon, Frédéric. 2017. Performance management: Why keeping score is so important, and so hard. Verkkoaineisto. McKinsey & Company. <https://www.mckinsey.com/capabilities/operations/our-insights/performance-management-why-keeping-score-is-so-important-and-so-hard>. Luettu 21.4.2023.

2 Business Application Platform | Microsoft Power Platform. Verkkoaineisto. Microsoft. <https://powerplatform.microsoft.com/en-us/>. Luettu 20.4.2023.

3 Phillips, James. 2016. Microsoft PowerApps and Flow are generally available starting tomorrow. Verkkoaineisto. Official Microsoft Blog. <https://blogs.microsoft.com/blog/2016/10/31/microsoft-powerapps-flow-generally-available-starting-tomorrow/>. Luettu 21.4.2023.

4 Crosbie, Lisa. 2021. Power Apps: Canvas vs Model-Driven Explained. Verkkoaineisto. YouTube. <https://www.youtube.com/watch?v=D5rHJsBShMQ>. Katsottu 23.4.2023.

5 Microsoft Power Apps. 2016. Microsoft.

6 Hess, Andrew. 2022. Guide to using Power Apps with Standard Connectors, No Premium License. Verkkoaineisto. YouTube. <https://www.youtube.com/watch?v=kzp9-f3gs78>. Katsottu 27.4.2023.

7 Lindhorst, Greg & Stall, Mike. 2021. Power Fx: the Programming Language for Low Code and what it means for Developers. Verkkoaineisto. YouTube. <https://www.youtube.com/watch?v=ik6k89WNjuk>. Katsottu 20.4.2023.

8 Sarabyn, Kelly. What's Wrong with Low and No Code Platforms? Verkkoaineisto. Pandium. <https://www.pandium.com/blogs/whats-wrong-with-low-and-no-code-platforms>. Luettu 25.4.2023.

9 Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023. 2022. Verkkoaineisto. Gartner. <https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023>. Luettu 29.4.2023.

10 Lindhorst, Greg. 2021. Canvas components can now access app scope directly. Verkkoaineisto. Microsoft. <https://powerapps.microsoft.com/en-us/blog/canvas-components-can-now-access-app-scope-directly/>. Luettu 26.4.2023.

11 About | The Dataverse Project. Verkkoaineisto. Dataverse. <https://dataverse.org/about>. Luettu 28.4.2023.

12 How are Dataverse for Teams and Dataverse different? 2022. Verkkoaineisto. Microsoft Power Apps -dokumentaatio. <https://learn.microsoft.com/en-us/power-apps/teams/data-platform-compare>. Päivitetty 16.12.2022. Luettu 29.4.2023.

13 View Dataverse for Teams table data in Power BI Desktop. 2023. Verkkoaineisto. Microsoft Power Apps -dokumentaatio. <https://learn.microsoft.com/en-us/power-apps/teams/view-table-data-power-bi>. Päivitetty 28.2.2023. Luettu 29.4.2023.

14 Edit or delete a table. 2022. Verkkoaineisto. Microsoft Power Apps -dokumentaatio. <https://learn.microsoft.com/en-us/power-apps/teams/edit-delete-table>. Päivitetty 15.2.2022. Luettu 29.4.2023.

15 Understand delegation in a canvas app. 2023. Verkkoaineisto. Microsoft Power Apps -dokumentaatio. <https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/delegation-overview>. Päivitetty 7.2.2023. Luettu 20.4.2023.