SAVONIA
ammattikorkeakoulu

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

KETTU

Käyttöliittymä erilaisille työtilauksille

TEKIJÄ Roosa Turunen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä Roosa Turunen	
Työn nimi Kettu: Käyttöliittymä erilaisille työtilauksille	
Päiväys 21.4.2023	Sivumäärä/Liitteet 29/0
Toimeksiantaja/Yhteistyökumppani Savon Voima Oyj	

Tiivistelmä

Tämän opinnäytetyön tavoitteena oli toteuttaa erilaisia lomakkeita sisältävä web-sovellus yrityksen sisäiseen käyttöön. Tarkoituksena oli, että laajempi käyttäjäjoukko pystyisi syöttämään tietoja web-sovellukseen ja sen kautta käynnistämään prosesseja Frends-integraatioalustalla. Web-sovelluksen vaadittuja ominaisuuksia oli, että se pystyisi viemään tietoja Frendsiin, sovellusta olisi helppo muokata ja siihen voisi kirjautua Azure AD:n kautta. Tavoitteena oli myös tutkia, minkälaisia toteutustapoja muissa vastaavanlaisissa töissä on käytetty ja millä toteutustavoilla tämä opinnäytetyö olisi parasta toteuttaa.

Web-sovelluksen nimeksi muodostui Kettu. Kettu ohjelmoitiin käyttämällä Blazor-käyttöliittymäkehystä. Tiedot päätettiin siirtää Frendsille HTTP POST-kutsujen avulla. Sivusta saatiin helposti muokattava, kun lomakkeilla olevien komponenttien tietoja päätettiin ylläpitää JSON-tiedostossa. Muokattavuutta helpottaa myös se, että koodi päätettiin viedä palvelimelle Azure DevOps Pipelinesin avulla. Kettu-sovellus käyttää käyttäjienhallintaan Azure AD:tä. Muita käytettyjä toteutustekniikoita olivat HTML ja CSS. Muiden käyttämiä toteutustapoja ja tekniikoita on tutkittu netistä tietoa hakemalla ja niitä vertailemalla.

Ketun ensimmäinen versio julkaistiin IIS-palvelimella, jonka jälkeen toimistotyöntekijät pystyivät sitä käyttämään. Projektin aikana syntyi paljon erilaisia ideoita, mitä web-sovellus voisi sisältää, mutta opinnäytetyön rajallisuuden vuoksi joitain ominaisuuksia jouduttiin karsimaan. Sovellusta tehdessä olikin jo selvää, että sitä tullaan jatkokehittämään ja myös siksi, yksi sovelluksen tärkeimmistä ominaisuuksista oli sovelluksen helppo muokattavuus.

Avainsanat

Frends, web-sovellus, käyttöliittymä, JSON, front-end

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author Roosa Turunen	
Title of Thesis Kettu: User Interface for Various Work Orders	
Date 21 April 2023	Pages/Appendices 29/0
Client Organisation/Partner Savon Voima Oyj	

Abstract

The aim of this thesis was to implement a web application that includes different kinds of forms for internal company use. The purpose was that more users could fill in forms and start processes in Frends integration platform. Required features were that it would export data to Frends, it would be dynamic, and it would use Azure AD for user authentication. The aim was also to research what kinds of implementation methods others have used in the same kinds of projects and which kinds of implementation methods would be best for this thesis.

The web application got the name Kettu. Kettu was programmed with the Blazor user interface framework. It was decided that the data is exported from the form to Frends using HTTP POST requests. Kettu is dynamic because its form components were decided to be maintained in a JSON file, and because the code is exported to the server by Azure DevOps Pipelines. Kettu uses Azure AD to manage users. Other methods of implementations that were used, are HTML and CSS. Methods of implementations that other developers have used were searched on the internet.

After the first version of Kettu was published on the IIS server, office workers were able to use it. During the project, there were some ideas about what this application should include. Because of the limitation of time, some ideas needed to be rejected. It was clear that the application is going to be further developed and that is also why it was important that the application was dynamic.

Keywords

Frends, web application, user interface, JSON, front-end

SISÄLTÖ

KUVALUETTELO

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on tehdä erilaisista lomakkeista koostuva web-sovellus yrityksen sisäiseen käyttöön. Tarkoituksena on, että laajempi käyttäjäjoukko pystyisi täyttämään tietoja kyseiseen web-sovellukseen ja sen kautta käynnistämään prosesseja Frends-integraatioalustalla. Käyttäjäjoukkona voivat toimia esimerkiksi kenttätyöntekijät tai toimistotyöntekijät. Tässä opinnäytetyössä tullaan myös pohtimaan mitkä toteutustavat ja tekniikat olisivat parhaita juuri tälle työlle.

Web-sovelluksen nimeksi muodostui sana Kettu, joka on lyhenne sanoista käyttöliittymä erilaisille työtilauksille. Työn tarkoituksena on yksinkertaistaa ja nopeuttaa Frends-prosessin käynnistykseen tarvittavaa työvaihetta. Tarkoituksena on myös päästä pois tavasta, jossa tietoja kirjoitetaan suoraan Frendsiin.

Työn toimeksiantajana toimii Savon Voima Oyj. Savon Voima on vuonna 1947 perustettu Itä-Suomessa toimiva konserni, jonka tehtävänä on tarjota energiapalveluita. Sillä on kaksi tytäryhtiötä Savon Voima Verkko Oy ja Itä-Suomen Biomassa Oy. Savon Voima Oyj:n pääkonttori sijaitsee Siilinjärvellä Toivalassa. (Savon Voima, ei pvm)

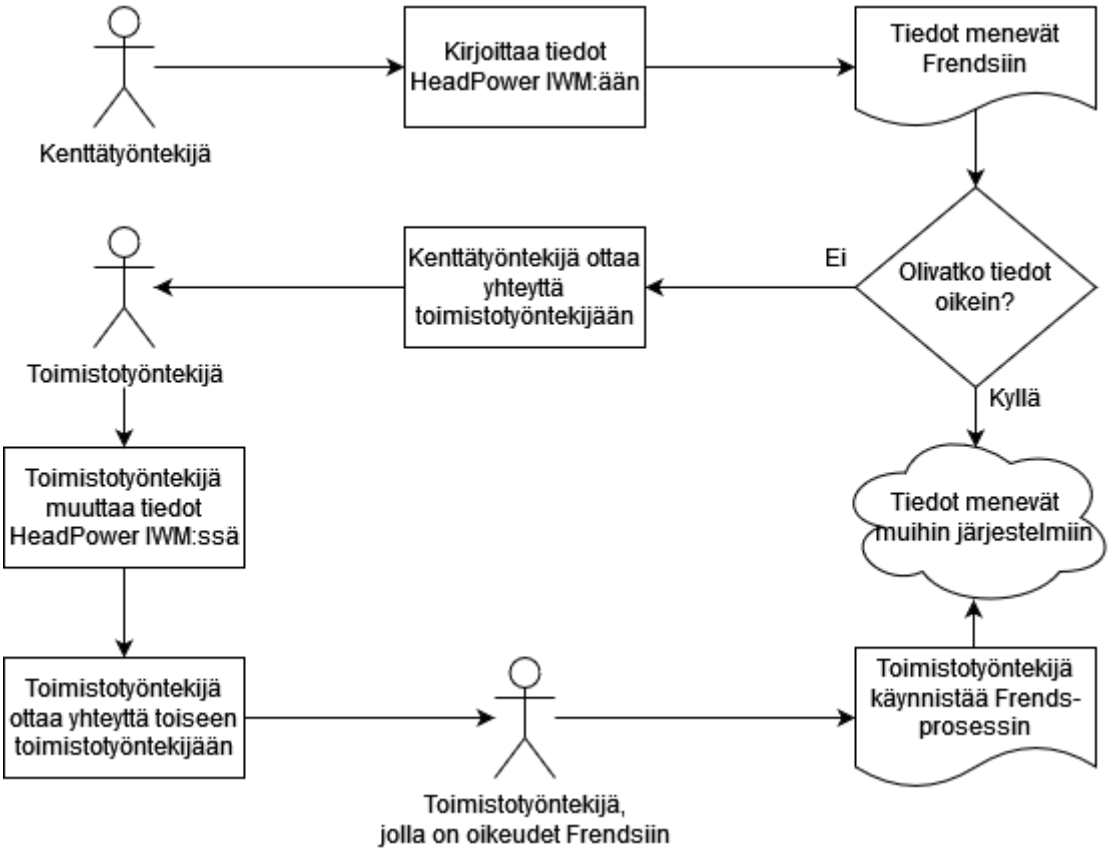
2 TYÖN TAUSTAA

Opinnäytetyön tarkoitus ja tarpeellisuus tulevat ilmi, kun tarkastelee tähän opinnäytetyöhön liittyviä lähtötilanteita. Lähtötilanteina toimivat erilaiset tapahtumasarjat, joita tarvitaan, jotta eri Frends-prosessit pystyttäisiin käynnistämään. Tapahtumasarjoissa on joitain ongelmakohtia, joita tässä opinnäytetyössä rakennettavan Kettu-sovelluksen on tarkoitus ratkoa.

Ensimmäisenä tapauksena on tilanne, jossa kenttätyöntekijä kirjaa mittarin tiedot HeadPower IWM-sovellukseen. Kun tiedot on kirjattu, lähtevät ne Frends-integraatioalustan kautta muihin järjestelmiin. Jos syötetyissä tiedoissa onkin virhe, Frends-prosessi kaatuu. Näin ollen tietoja ei saada vietyä muihin järjestelmiin. Kenttätyöntekijä ei voi itse jälkikäteen muuttaa näitä tietoja ja käynnistää Frends-prosessia uudestaan, jotta tiedot menisivät muihin järjestelmiin, vaan kenttätyöntekijän on otettava yhteyttä Savon Voiman toimistotyöntekijään. Toimistotyöntekijä muuttaa tiedot oikeiksi HeadPower IWM:ssä, mutta ei voi käynnistää prosessia itse, sillä hänellä ei ole oikeuksia päästä Frendsiin. Tämän tulee ottaa yhteyttä toiseen Savon Voiman toimistotyöntekijään, jolla on oikeudet päästä sinne. Kyseinen toimistotyöntekijä käynnistää Frends-prosessin manuaalisesti, jotta IWM:ssä muutetut mittarin tiedot siirtyvät muihin järjestelmiin. Alempana olevassa kuvassa (kuva 1) kuvataan kyseinen tapahtumasarja.

Headpower IWM (Infra Work Manager) on tehtävienhallintasovellus. Sen avulla voi muun muassa välittää ja kuvata työtilauksia, sekä parantaa viestintää työn eri osapuolien kanssa. Sovellusta voidaan käyttää kaikilla eri päätelaitteilla. Sen voi myös integroida toisiin järjestelmiin. (HeadPower, ei pvm) HeadPower Oy on suomalainen vuonna 2001 perustettu pilvipalveluita infraverkkoyhtiöille toimittava osakeyhtiö (HeadPower, ei pvm).

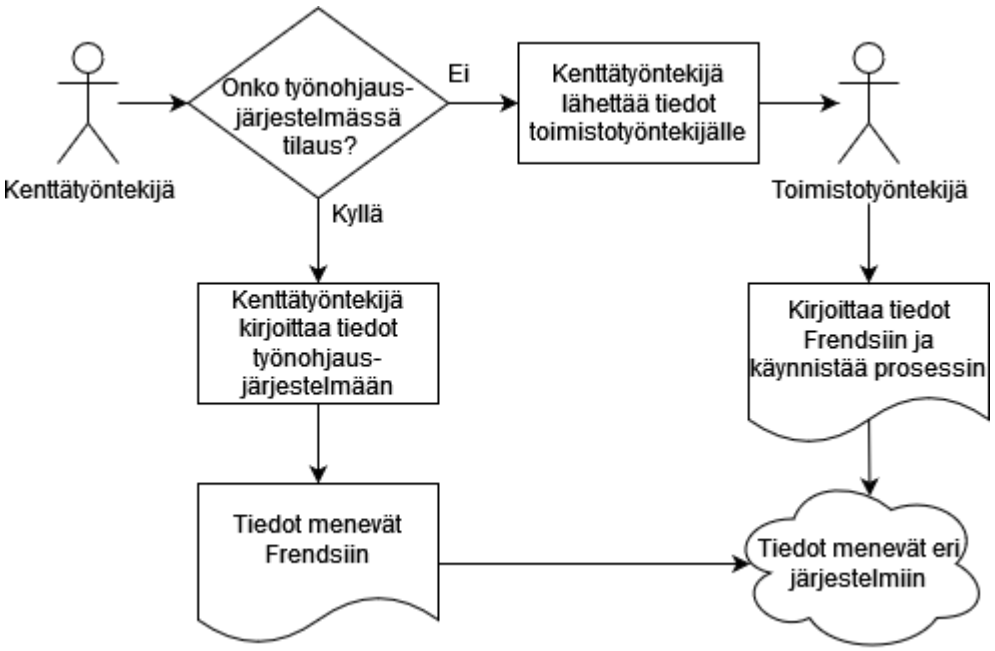
Frends iPaaS (integration Platform as a Service) on integraatioalusta, joka on selainpohjainen ohjelmointirajapintojen integrointeihin ja prosessien automatisointiin tarkoitettu alusta. Kyseisiä integrointeja ja automatisointeja voidaan Frendsin avulla hallita, turvata ja kehittää. (Frends, ei pvm) Frends on HiQ:n omistama ja kehittämä integraatioalusta (Frends, ei pvm). HiQ on vuonna 1995 perustettu suomalainen teknologiakonsulttiyritys, jonka on vuodesta 2020 alkaen omistanut Triton (HiQ, ei pvm).



Kuva 1. HeadPower IWM -tietojen kulku

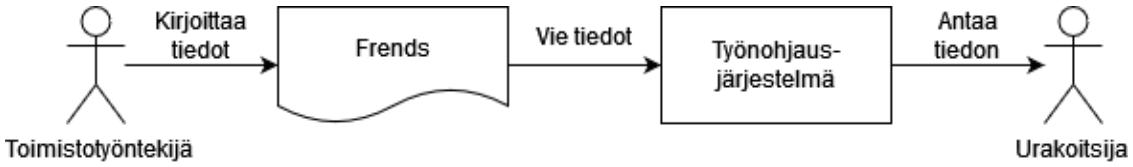
Toisena esimerkkitapauksena on tilanne, jossa kenttätyöntekijä kirjaa työsuorituksen tiedot työnohjausjärjestelmään. Työnohjausjärjestelmiä ovat HeadPower IWM ja WiseMaster. Voi kuitenkin olla tilanne, jossa työnohjausjärjestelmässä ei ole tilausta, jonka tietoja täyttää. Tällöin kenttätyöntekijän täytyy lähettää sähköpostilla työn tiedot Savon Voiman toimistotyöntekijälle. Toimistotyöntekijä kirjaa tiedot Frends-integraatioalustalle, josta tiedot menevät muihin järjestelmiin. Alempana olevassa kuvassa (kuva 2) on kuvattu kyseinen tapahtumasarja.

WiseMaster on tietojärjestelmä, joka on suunniteltu vesihuollon, asennus- ja huoltopalveluyritysten, sekä energiateollisuuden käyttöön. WiseMasteria voidaan käyttää toiminnanohjaukseen ja eri alojen kunnossapitoon. Se on myös mobiilikäyttöinen ja sillä voidaan hallita hajautettua tietoa. WiseMasterin on kehittänyt M-Technology Oy. (WiseMaster, ei pvm)



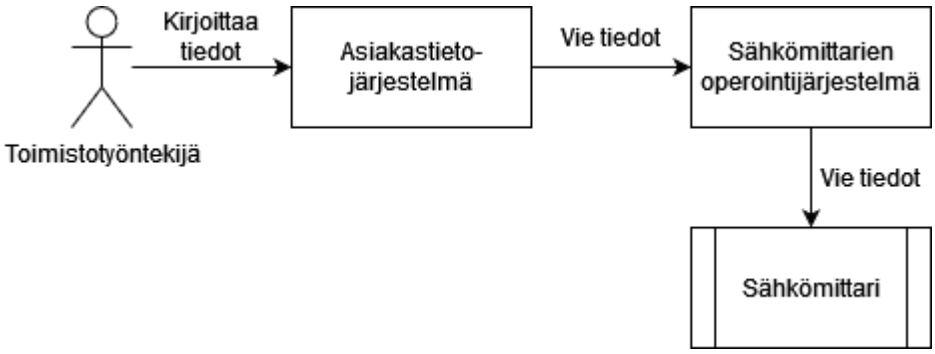
Kuva 2. Työsuoritustietojen kulku

Urakoitsijoille täytyy antaa erilaisia työmääräimiä työnohjausjärjestelmiin. Näitä työmääräimiä voivat olla muun muassa kaukolämpömittarin asennus ja poisto. Työmääräimiä on annettu kirjaamalla tietoja suoraan Frendsiin ja käynnistämällä sen prosessi manuaalisesti. Tiedot kulkeutuvat Frendsistä työnohjausjärjestelmään, josta urakoitsija voi ottaa työmääräimen työn alle. Alla olevassa kuvassa (kuva 3) on kuvattu, kuinka tapahtumasarja etenee.



Kuva 3. Työmääräimen kulku urakoitsijalle

Sähkömittareille voidaan etäyhteyden avulla antaa erilaisia ohjelmointikäskyjä, joiden avulla mittarit saadaan suorittamaan erilaisia toimintoja. Näistä toiminnoista osa annetaan asiakastietojärjestelmän kautta. Asiakastietojärjestelmästä tieto menee sähkömittarien operointijärjestelmään ja sieltä sähkömittarille. Alla oleva kuva (kuva 4) esittää kyseisen tapahtumasarjan. Savon Voima on kuitenkin luopumassa tavasta antaa ohjelmointikäskyjä asiakastietojärjestelmän kautta, joten tilalle tulisi saada toinen käyttöliittymä, josta käskyjä voitaisiin viedä sähkömittareille.



Kuva 4. Ohjelmointikäskyn antaminen sähkömittarille

Kyseisistä tapahtumasarjoista löytyy joitain ongelmakohtia. Esimerkiksi, kun tietoja joudutaan kirjoittamaan moneen kertaan monen henkilön kautta, voi virhenäppäilyjä syntyä helposti. Mikäli Frends-integraatioalustalle syöttää virheellistä tietoa, se kaatuu virheeseen, jonka selvittäminen aiheuttaa ylimääräistä työtä.

Toisena ongelmakohtana on se, että tapahtumasarja olisi mahdollista toteuttaa yksinkertaisemminkin. Tiedot voisivat mennä suoraan Frendsille kulkematta ylimääräisen henkilön kautta. Ulkopuolisia käyttäjiä ei kuitenkaan haluta päästää suoraan Frendsiin käsiksi, sillä Frends on tärkeä järjestelmä Savon Voimalle ja siksi sinne pääsyä halutaan rajata. Tällä hetkellä käyttäjien oikeuksia ei pystytä rajaamaan tarpeeksi, jotta muiden käyttäjien pääsy Frendsiin olisi turvallista. Frends on myös prosessien automatisaatioon tarkoitettu integraatioalusta (Frends, ei pvm). Sen prosesseja ei siis ole tarkoitettu käynnistettäväksi manuaalisesti suoraan Frendsistä.

Olisi hyvä päästä pois siitä, että tiedot joutuvat kulkemaan monen henkilön kautta. Tämä säästäisi työaikaa ja yksinkertaistaisi Frends-prosessin käynnistykseen vaadittavaa työvaihetta. Myös tarvittavat validoinnit helpottaisivat työprosessia, sillä se estäisi Frends-prosessia kaatumasta vääränlaisen tiedon takia. Olisi myös hyvä päästä pois tavasta, jossa tietoja joudutaan kirjoittamaan suoraan Frendsiin. Näihin prosessin käynnistykseen tarvittaviin tapahtumasarjoihin halutaan saada muutos.

3 TYÖN TAVOITE

3.1 Kettu-sovelluksen vaadittavat ominaisuudet

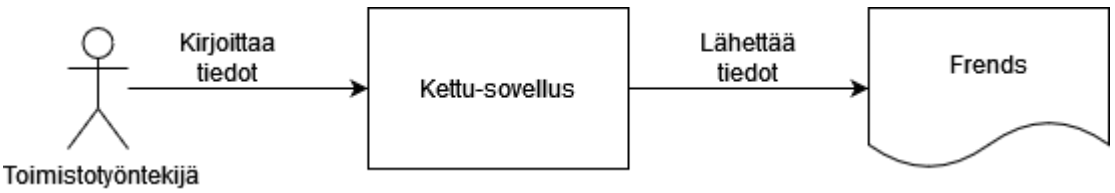
Sovelluksen tulee koostua erilaisista lomakkeista. Lomakkeet, joita käyttäjälle näytetään riippuvat käyttäjästä. Lomakkeet koostuvat erilaisista kentistä, kuten pudotusvalikoista, tekstinsyöttökentistä, sekä päivämääräkentistä. Jokaisella lomakkeen kentällä tulee olla oikeanlainen validointi, jotta virhenäppäilyiltä säästyttäisiin. Kenttiin syötetty data siis validoidaan, eli katsotaan, onko data sellaista, kun sen kuuluisi olla (Tahvanainen, 2018). Kun lomakkeen kentät on täytetty oikein, tulee tietojen lähteä napista painamalla Frendsille ja käynnistää siellä prosessi. Sovelluksesta halutaan myös tyylikäs ja selkeä, jotta sitä olisi helppo käyttää. Tyylin tulee olla Savon Voiman brändin mukainen.

Kettu-sovellukseen tulee päästä kirjautumaan Azure AD:n kautta, jotta käyttäjien oikeuksia pystyttäisiin rajaamaan. Azure AD eli Azure Active Directory on palvelu, jolla voi hallita käyttöoikeuksia ja identiteettejä. Azure AD:n avulla yrityksen työntekijät pystyvät käyttämään ulkoisten resurssien lisäksi myös yrityksen sisäisiä resursseja. (Microsoft, 2022)

Sovelluksesta halutaan saada helposti muokattava. Aikaisemmin mainitut esimerkkitilanteet prosessin käynnistykseen tarvittavista tapahtumasarjoista, ovat vain esimerkkejä monien muiden vastaavanlaisten tilanteiden joukossa. Kettuun halutaan siis pystyä myöhemmin lisäämään muunlaisia lomakkeita ja yhdistämään ne eri prosesseihin. Lisäksi lomakkeiden sisältö tulisi olla helposti muokattavissa. Kettu-sovelluksesta halutaan myös sellainen, ettei se tarvitsisi jokaisen muokkauskerran jälkeen päivityskatkoa.

3.2 Kettu-sovelluksen lomakkeet ja vaikutus tapahtumasarjoihin

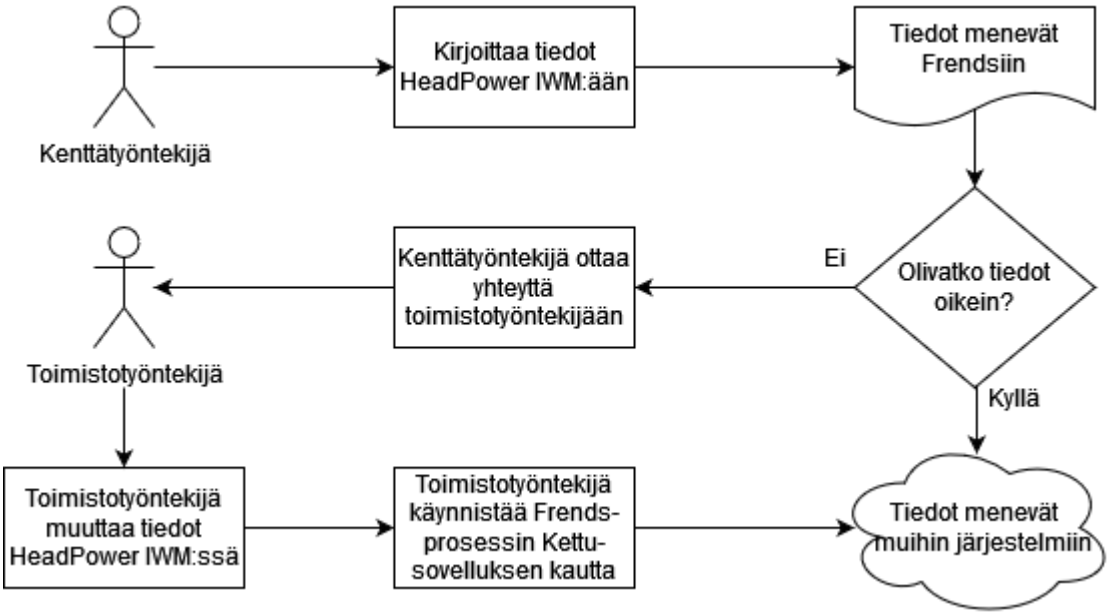
Projektin tavoitteena on tehdä web-sovellus yrityksen sisäiseen käyttöön. Tarkoituksena on, että laajempi käyttäjäjoukko pystyisi itse kirjoittamaan tietoja kyseiseen web-sovellukseen ja siten käynnistämään Frends-prosesseja sen kautta. Tapahtumaketjuissa Kettu-sovellus korvaa tapahtuman, jossa toimistotyöntekijä joutuisi käynnistämään prosessin syöttämällä tietoja suoraan Frendsiin.



Kuva 5. Tavoitteena on korvata Kettu-sovelluksella tapahtumasarjoista vaihe, jossa tietoja joudutaan kirjoittamaan suoraan Frendsiin.

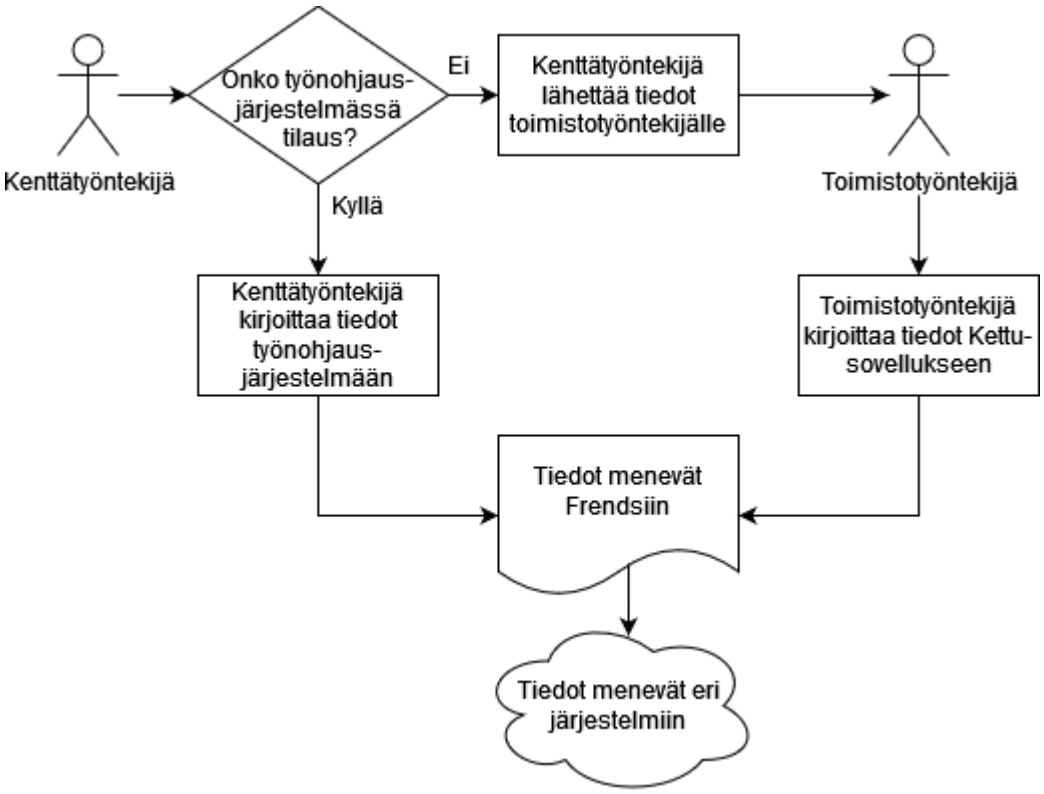
Kettu-sovelluksen avulla halutaan pystyä käynnistämään useita eri prosesseja. Sovellukseen halutaan siis erilaisia lomakkeita, joista jokainen käynnistäisi omanlaisensa Frends-prosessin. Yhdellä näistä lomakkeista halutaan saada käynnistettyä prosessi IWM id:n avulla. Tätä lomaketta kutsutaan nimellä ”Palauta IWM”. Lomakkeessa tulee olla kenttä, johon halutun IWM id:n pystyy syöttämään. Kyseisen lomakkeen tulisi pystyä lähettämään syötetty IWM id Frendsille ja käynnistämään sen prosessi. Alla olevassa kuvassa (kuva 6) on kuvattu, kuinka Kettu-sovelluksessa

oleva “Palauta IWM” -lomake korvaisi toisen toimistotyöntekijän osuuden HeadPower IWM -tietojen viemiseen tarvittavasta tapahtumasarjasta. Kuvaa (kuva 6) voi verrata lähtötilanteeseen (kuva 1).



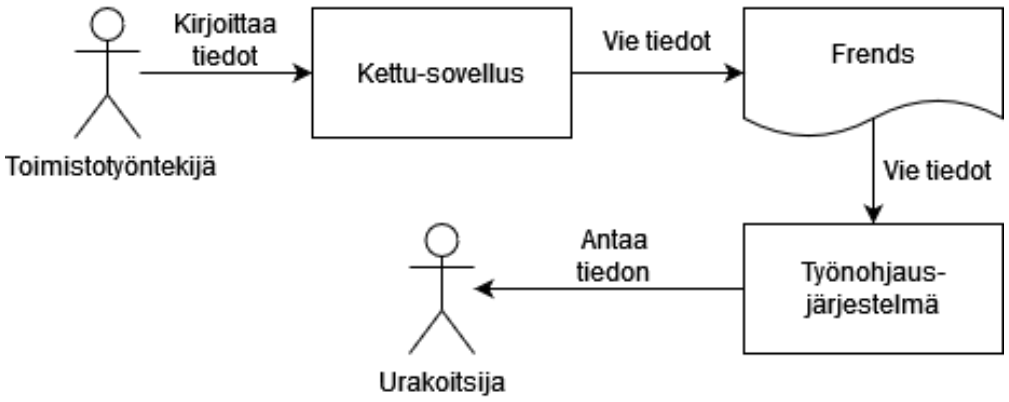
Kuva 6. Kettu-sovellus korvaa toisen toimistotyöntekijän osuuden tapahtumasarjasta, jossa HeadPower IWM -tietoja viedään muihin järjestelmiin.

Toisessa lomakkeessa halutaan käynnistää Frends-prosessi useamman tiedon avulla. Näitä tietoja ovat käyttöpaikka, vaihtoehtoinen käyttöpaikka, työn suoritusaika, vanhan mittarin sarjanumero, uuden mittarin sarjanumero, vanhan mittarin energialukema, vanhan mittarin vesilukema, asentaja, sekä lisätiedot. Tämän lomakkeen nimi on ”Palauta laitevaihto”. Alla olevassa kuvassa (kuva 7) on kuvattu kuinka Kettu-sovellus korvaisi tavan, jossa tietoja syötetään suoraan Frendsiin. Kuvaa (kuva 7) voi verrata lähtötilanteeseen (kuva 2). Kettu-sovelluksessa oleva ”Palauta laitevaihto” -lomake veisi siis tiedot Frendsiin ja käynnistäisi prosessin ilman, että käyttäjän tulisi mennä kyseiseen integraatioalustaan tekemään samat asiat manuaalisesti.



Kuva 7. Kettu-sovelluksen ”Palauta laitevaihto” -lomake korvaisi sen, ettei tietoja tarvitsisi kirjoittaa suoraan Frendsiin.

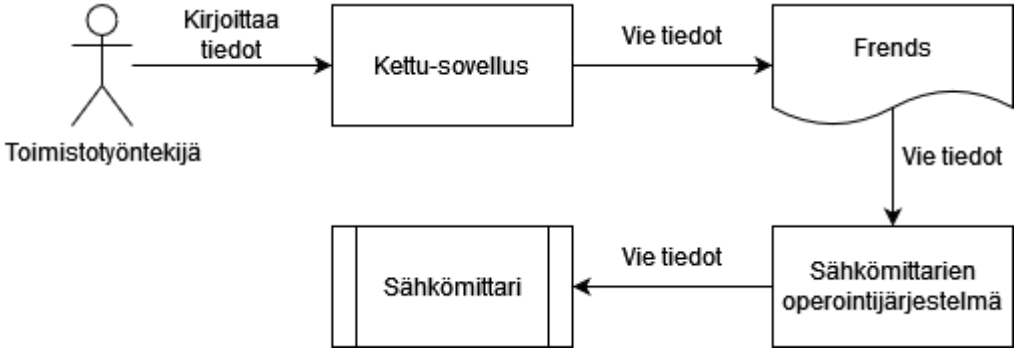
Kolmas ja neljäs lomake koskee kaukolämpömittarin vaihdon tilausta. ”Tilaa kaukolämpömittarin asennus” -nimisellä lomakkeella voidaan tilata kaukolämpömittarin asennus täyttämällä seuraavat kentät: käyttöpaikka, työlista, työtilaus, mittarintyyppi, katuosoite, kaupunki ja postinumero. ”Tilaa kaukolämpömittarin poisto” -nimisellä lomakkeella on samat kentät kuin ”Tilaa kaukolämpömittarin asennus” -lomakkeella, mutta siinä on lisäksi “Vanhan mittarin sarjanumero” -kenttä. Alla olevassa kuvassa (kuva 8) on kuvattu, kuinka kaukolämpömittarin vaihdon tilauksessa tiedot kirjataan Frendsin sijaan Kettu-sovellukseen. Kuvaa (kuva 8) voi verrata tämän tapahtumasarjan lähtötilanteeseen (kuva 3).



Kuva 8. Kaukolämpömittarin vaihdon tilauksessa tietoja voidaan kirjoittaa Frendsin sijaan Kettu-sovellukseen.

Viides lomake on nimeltään ”Sähkömittarin tuotteenvaihto”. Kyseisen lomakkeen on tarkoitus olla korvaava tapa antaa ohjelmointikäskyjä sähkömittareille. Lomakkeella tulee olla kolme kenttää nimeltä, ”Käyttöpaikka”, ”Vanha tuote” ja ”Uusi tuote”. ”Käyttöpaikka” määrittää mittarin,

jota ohjelmoidaan. ”Vanha tuote” kuvaa nykyistä ohjelmaa ja ”Uusi tuote” uutta haluttua ohjelmaa. Tiedot kulkevat Frendsin kautta sähkömittarien operointijärjestelmään käsiteltäväksi. Alla oleva kuva (kuva 9) näyttää, kuinka Kettu-sovellus korvaisi asiakastietojärjestelmän. Tapahtumasarjaa (kuva 9) voi verrata lähtötilanteeseen (kuva 4).



Kuva 9. Ohjelmointikäskyn vieminen sähkömittarille Kettu-sovelluksen avulla

Kettu-sovellus tullaan julkaisemaan IIS-palvelimella, joka pyörii Azuressa olevalla virtuaalikoneella. Virtuaalikoneet ovat samanlaisia kuin muutkin tietokoneet, mutta ne eivät ole fyysisiä. Virtuaalikoneet pyörivät fyysisillä palvelimilla, mutta itse tietokone on olemassa vain koodina. (Microsoft, ei pvm) IIS (Internet Information Services) on web-palvelin Windows-palvelimille (Microsoft, ei pvm).

Ainoastaan Savon Voiman henkilökunnalla on pääsy Kettu-sovellukseen. Sivulle pääsyä rajoitetaan entisestään, sillä vain Azure portalissa määritetyt käyttäjät pääsevät kirjautumaan sivulle. Näille käyttäjille on myös määritetty Kettu-sovelluksessa, mille lomakkeille heillä on oikeus päästä. Azure portal on graafinen käyttöliittymä, joka on vaihtoehto komentorivityökaluille. Sen kautta voi hallita Azure-tilauksia. Azure portalin kautta voi lisäksi hallita, valvoa ja rakentaa muun muassa web-sovelluksia. (Microsoft, 2022)

4 TOTEUTUSTAPOJEN JA TEKNIIKOIDEN SELVITYS

4.1 Selvitettävät asiat

Tämän opinnäytetyön yhdeksi vaiheeksi kuuluu tutkia ja pohtia millä mahdollisilla toteutustavoilla ja tekniikoilla tätä työtä olisi parasta lähteä tekemään. Tähän vaikuttaa tietysti se, mitä tullaan tekemään ja minkälaisia ominaisuuksia sovellus tarvitsee. Tutkin netin avulla mitä tapoja ja tekniikoita muut ovat käyttäneet vastaavanlaisissa projekteissa. Eniten tähän lopputulokseen tulee kuitenkin vaikuttamaan se, mitä toteutustapoja ja tekniikoita Savon Voima itse on käyttänyt samankaltaisissa töissä. Tämä on tärkeää työn jatkokehityksen kannalta, sillä yrityksestä löytyvä osaaminen on otettava huomioon, jotta mahdollinen jatkokehitys olisi vaivatonta.

Aina kun jotain ryhdytään ohjelmoimaan, täytyy valita millä ohjelmointikielellä se tullaan tekemään. Von Kügelgenin ja Laukkosen (2020, s. 53) kirjassa Sallinen mainitsee, että kun valitsee ohjelmointikieltä, tulee miettiä mihin tarkoitukseen ohjelmointikieltä käytetään ja sen mukaan valita oikeanlainen ohjelmointikieli. Von Kügelgenin ja Laukkosen kirjassa mainitaan myös, että ohjelmointikieliä on olemassa paljon ja niitä kehitetään koko ajan enemmän. Pääosin niistä kuitenkin vain muutamaa kymmentä käytetään yleisesti. Ohjelmointikielten määrä selittyy suurimmaksi osaksi sillä, että kielten käyttötarpeita on paljon erilaisia. (Kügelgen (von Kügelgen) & Laukkonen, 2020, s. 53)

Kielen valintaan vaikuttaa myös se, millä kirjastolla tai ohjelmistokehyksellä web-sovellusta aiotaan ryhtyä tekemään. Von Kügelgenin ja Laukkosen (2020, s. 47) mukaan kirjastot (library) ovat aliohjelmista koostuvia kokoelmia. Kirjaston kokoelmiin kootaan yleensä ratkaisuja, jotka liittyvät samaan ongelmakenttään. Esimerkiksi React ja SQLite ovat kirjastoja. Ohjelmistokehykset (software frameworks) ovat taas niin sanottuja runkoja ohjelmistoille. Näissä rungoissa on kohtia, joihin voi täydentää omaa koodia ja näin rakentaa ohjelmistoja helposti. Näitä kohtia kutsutaan laajennoskohdiksi (hot spot). (Kügelgen (von Kügelgen) & Laukkonen, 2020, s. 47) Monesti web-sovellusten ohjelmointiin käytetään ohjelmointikieliin perustuvia ohjelmistokehyksiä. Ruby-nimiseen ohjelmointikieleen pohjautuvat ohjelmistokehykset Ruby on Rails sekä Sinatra. PHP-ohjelmointikieleen perustuu taas Laravel-niminen ohjelmistokehys ja Python kieleen Django-niminen ohjelmistokehys. (Kügelgen (von Kügelgen) & Laukkonen, 2020, s. 70)

Ohjelmistokehyksilläkin on paljon eroja ja niiden valinta perustuu myös siihen, mihin tarkoitukseen ohjelmistokehystä halutaan käyttää. Boyarko (2023) käsittelee artikkelissaan ohjelmistokehyksiä front-end ja back-end ohjelmistokehyksinä. Front-end tarkoittaa sivun osaa, jonka käyttäjä näkee. Back-end käsittelee taas palvelinpuolta, mitä käyttäjä ei näe. Front-endin ja back-endin yhdistelmää, eli koko pakettia, kutsutaan nimellä full-stack. (Kügelgen (von Kügelgen) & Laukkonen, 2020, ss. 69-70) Ohjelmistokehyksen käyttö ei myöskään ole mitenkään pakollista web-sovelluksia tehdessä, mutta joidenkin ominaisuuksien kohdalla oikeanlainen ohjelmistokehys voi helpottaa sivun ohjelmoimista huomattavasti.

Tässä projektissa tietokantayhteyttä ei erikseen tarvitse tehdä, sillä sovellus on yhteydessä Frends-integraatioalustalle, joka on sitten yhteydessä tietokantaan ja muihin järjestelmiin. Sovellus tulee kommunikoimaan Frendsin kanssa ja saamaan palautetta Frendsiltä, mikäli jokin menee vikaan tietojen siirrossa. Tässä tulee kuitenkin selvittää, miten web-sovellus tulee olemaan yhteydessä

Frends-integraatioalustaan, eli millä tavalla tietoa voi siirtää Frendsiin ja kuinka sen prosessi pystytään käynnistämään tietojen siirron jälkeen.

Toteutustavan valinnassa täytyy myös ottaa huomioon se, että sivusta haluttiin helposti muokattava. Tähän tietysti liittyy yrityksessä jo oleva osaaminen, eli tarvitseeko muokkausta varten erillisen käyttöliittymän vai riittäkö, että muutoksia tehtäisiin suoraan koodiin. Web-sovelluksesta halutaan myös sellainen, ettei se tarvitsisi suurempaa päivityskatkoa jokaisen muutoksen jälkeen. Myös koodin kommentointi ja selkeys tekevät sovelluksesta helpommin muokattavamman.

4.2 Muiden käyttämät toteutustavat ja tekniikat

Nettisivuja ohjelmoidaan yleensä käyttäen HTML-, CSS- ja Javascript-kieliä (Kügelgen (von Kügelgen) & Laukkonen, 2020, s. 70). Myös Kantola (2020) ja Tilli (2022) ovat käyttäneet opinnäytetyö projektissaan HTML-kieltä. Tilli (2022) on käyttänyt myös CSS- ja Javascript-kieliä. Javascript on yleisin nettisivuihin käytetty ohjelmointikieli, sillä lähes kaikki verkkoselaimet osaavat kääntää kyseistä ohjelmointikieltä (Kügelgen (von Kügelgen) & Laukkonen, 2020, s. 54).

Nettisivujen ohjelmointiin käytettyjä kirjastoja ja ohjelmistokehyksiä ovat muun muassa React.js, Angular.js, Bootstrap ja jQuery (Kügelgen (von Kügelgen) & Laukkonen, 2020, s. 70). Esimerkiksi Tilli (2022) oli käyttänyt opinnäytetyössään Bootstrap:ia ja jQuery:ä. Boyarkon (2023) mukaan vuoden 2023 parhaat front-end ohjelmistokehykset ovat React, Angular, Vue, Ember ja jQuery. Hän kuitenkin mainitsee, että React ei käytännössä ole ohjelmistokehys vaan kirjasto. (Boyarko, 2023)

Sisällönhallintajärjestelmä eli Content Management System, joka lyhennetään myös CMS, on tietojärjestelmä, jonka avulla voidaan muun maussa muokata nettisivuja. Sisällönhallintajärjestelmän avulla sivun muokkaus ei edellytä sen suurempia ohjelmointitaitoja. Esimerkiksi sosiaalisen median alustat ovat jonkinlaisia sisällönhallintajärjestelmiä. (Folcan, ei pvm) Fellin (2023) mukaan parhaat sisällönhallintajärjestelmät vuonna 2023 ovat HubSpot CMS Hub, WordPress, Joomla!, Drupal ja WooCommerce. Sivun tekeminen sisällönhallintajärjestelmällä voisi siis helpottaa sivun jatkokehitystä ja myöhempää muokkausta.

Ahlstén (2022) kertoo insinöörityössään lähettävänsä Hubspot-palvelussa olevan lomakkeen tiedot POST-tyyppisellä HTTP-kutsulla Frendsille. Ahlstén myös kertoo, että Frends voi vastaanottaa ja lähettää HTTPS-kutsuja ja näin siihen on helppo yhdistää ainakin Hubspot ja Power Automate -rajapinnat, joita hän on käyttänyt insinöörityössään. (Ahlstén, 2022) Myös Ludwig (2022) mainitsee insinöörityössään, että on saanut Frends-prosessin aktivoitumaan, kun on lisännyt HTTP-triggerin kyseiseen prosessiin. Tällöin hänen työssään käyttämänsä SlackBot, kutsuu lisätyn HTTP-triggerin HTTPS-osoitetta, jolloin prosessi lähtee käyntiin. (Ludwig, 2022)

4.3 Työn toimeksiantajan käyttämät toteutustavat ja tekniikat

Työn toimeksiantaja on käyttänyt vastaavanlaisissa töissä Blazoria. Blazor on .NET-tiimin kehittämä käyttöliittymäkehys (user interface framework) (dotnet, 2021). Blazorin avulla voidaan rakentaa full-stack sovelluksia käyttäen C#:ia ja .NET:iä (Microsoft, ei pvm). Ohjelmointiympäristönä työn toimeksiantaja on käyttänyt usein Visual Studiota, mutta myös Visual Studio Codea.

Frendsiin työn toimeksiantaja on ollut yhteydessä Azure Service Bussin kautta, mutta myös suoraan HTTP-kutsujen kautta. Käyttämällä HTTP-kutsuja Frendsiin on lisätty HTTP-triggeri, johon on voitu olla yhteydessä. Työn toimeksiantaja on joissain töissä tehnyt sivun muokkaamista varten toisen sivun. Sivun muokkaaminen käyttöliittymän kautta helpottaa esimerkiksi sellaisen henkilön työtä, jolla ei ole kokemusta ohjelmoinnista.

5 KÄYTETYT TOTEUTUSTAVAT JA TEKNIIKAT

Työn toimeksiantajalla oli aika selkeä linja, miten ja millä toteutustavoilla työ tullaan tekemään. Myös tässä työssä päädyttiin käyttämään Blazor-käyttöliittymäkehystä, sekä sen käyttämää C#-kieltä. Sivu rakennettiin myös käyttämällä HTML ja CSS -kieliä. Ohjelmointiympäristönä päätettiin käyttää Visual Studiota, sillä työn toimeksiantaja yrityksessä sitä on pääosaksi käytetty. Työskentelyssä apuna käytettiin Azure DevOpsia, jonka avulla pystyi käyttämään koodin versionhallintaa, sekä seurata ja hallita työtehtäviä.

Frendsiin päätettiin olla yhteydessä suoraan POST-tyyppisten HTTP-kutsujen kautta. Mikäli Frendsiin olisi päätetty olla yhteydessä Azure Service Bussin kautta, olisi se koonnut sille tulevat viestit jonoon. Azure Service Bus olisi myös purkanut jonoja vain silloin, kun vastaanottava sovellus olisi ollut valmis vastaanottamaan viestejä. Azure Service Bussissa viestit liikkuvat molempiin suuntiin samalla tavalla. (Microsoft, 2023) HTTP-protokolla toimii taas siten, että käyttäjäpuolelta avataan yhteys palvelimelle, tehdään pyyntö ja odotetaan, kunnes palvelimelta saadaan vastaus (Mozilla, ei pvm).

Voi olla, että Azure Service Bussin avulla responsiivisuus olisi ollut monimutkaisempi toteuttaa ja se ei välttämättä olisi ollut niin nopea vastaamaan. Oletettiin myös, ettei viestejä sattuisi olemaan yhtäaikaisesti niin monta, että jono ominaisuudesta olisi ollut sen suurempaa hyötyä. Sovelluksesta haluttiin responsiivinen, eli mikäli viestin lähetys epäonnistuisi, voitaisiin käyttöliittymään saada viesti virheestä. Tämä viesti voisi olla esimerkiksi Frendsissä määritetty kaatumisilmoitus, joka kertoisi missä kohtaa prosessia virhe syntyi. Näin käyttäjä saisi tietää, miksi viesti ei mennyt perille. Niinpä yhteys Frendsiin päätettiin toteuttaa HTTP-kutsujen avulla.

Sivusta haluttiin helposti muokattava. Työn toimeksiantajalta löytyy sen verran osaamista, että sisällönhallintajärjestelmää tai erillistä sivua sovelluksen muokkaukseen ei ole tarvittavaa käyttää. Sivusta haluttiin kuitenkin sellainen, ettei se erikseen tarvitsisi suurempaa päivityskatkoa, mikäli uusia ominaisuuksia siihen lisättäisiin. Vaikka sivun päivittäminen ei aiheuttaisi sen isompaa käyttökatkoa, haluttiin opinnäytetyöhön saada laajuutta ja hieman haastavuutta. Ratkaisuna tähän, päätettiin lomakkeiden komponenttien joitain perustietoja ylläpitää JSON-tiedostossa, josta niiden tietoja voi muokata. Tällöin sivu hakee käynnistyessään komponenttien tiedot JSON-tiedostosta. Tämä mahdollistaa sen, ettei erillisiä päivityskatkoja tarvita, mikäli komponentteja muokataan ja näin ollen sivun muokkaaminen ei haittaa käyttäjiä. Lisäksi koodin kommentointi ja englanninkielisyys tekevät koodista helpommin ymmärrettävämmän kaikille.

Mikäli sivulle tarvitsee tehdä suurempia muutoksia, kuten lisätä uusi alasivu, ei se onnistu JSON-tiedostoa muokkaamalla. Tällöin suuremmat muutokset joudutaan tekemään suoraan koodiin. Jotta päivityskatkot olisivat lyhyitä ja päivitykset menisivät suoraan palvelimelle, päätettiin ottaa käyttöön Azure DevOps Pipelines. Pipelinesit yhdistettiin menemään IIS-palvelimelle. Tämä tekee sen, että kun koodi pusketaan Azure DevOps Reposiin, pipeline aktivoituu ja vie sivun uuden version automaattisesti palvelimelle. Tämän jälkeen sivun päivitetty versio on käytettävissä. Tämä on nopeampi ja helpompi tapa julkaista uusi versio web-sovelluksesta, kuin että uusi versio vietäisiin palvelimelle manuaalisesti.

6 KETTU-SOVELLUS

6.1 Yleisnäkymä ja ulkoasu

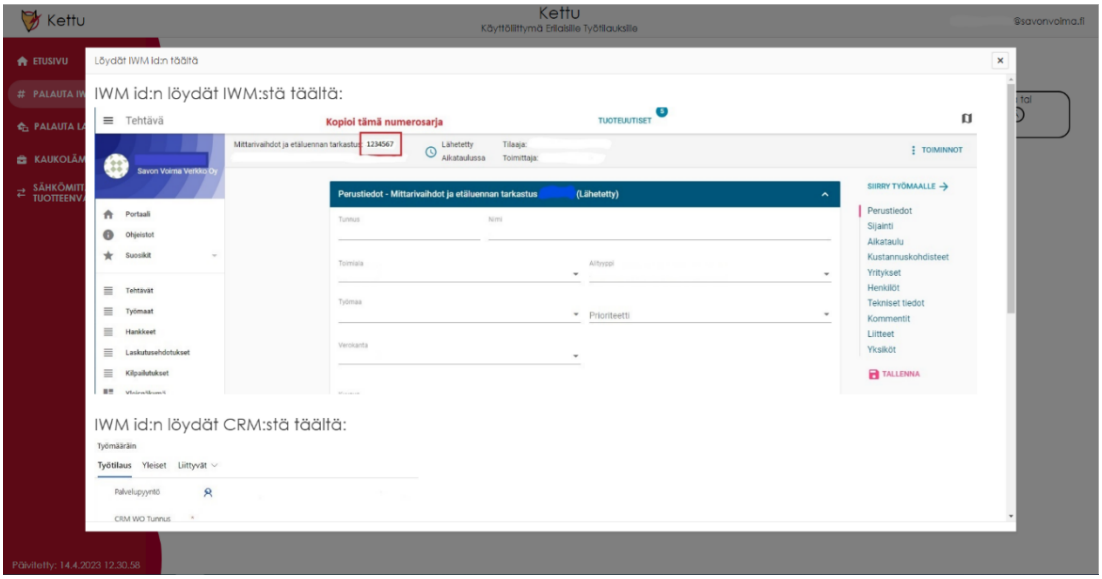
Sivun yleisnäkymässä näkyy aina ylä- ja sivupalkki. Yläpalkissa ensimmäisenä on Ketun logo ja nimi. Logo on suunniteltu ja toteutettu yhdessä Savon Voiman työntekijän kanssa. Keskellä yläpalkkia on web-sovelluksen nimi. Oikeassa yläkulmassa näkyy käyttäjän sähköpostiosoite, jolla käyttäjä on kirjautuneena web-sovellukseen.

Kettu-sovelluksessa ei ole ”Kirjaudu ulos” -nappia. Mikäli sovelluksesta kirjauduttaisiin ulos, se kirjaisi käyttäjän ulos myös kaikista muista sovelluksista, jotka käyttävät Azure AD:ta identiteettien hallinnointiin. Tämä koettiin epäkäytännölliseksi ja siksi se päätettiin jättää pois. Mikäli käyttäjä ei ole kirjautuneena sisään, ohjautuu sivu automaattisesti Azure AD -kirjautumisnäkymään. Käyttäjän ei tarvitse kirjautua kuin kerran, jonka jälkeen hän voi käyttää kaikkia sovelluksia ja sivuja, jotka käyttävät Azure AD:ta ilman, että käyttäjän tarvitsee jokaiselle sivulle kirjautua erikseen.

Sivupalkki koostuu linkeistä alasivuille. Perusnäkymässä linkkejä on viisi: Etusivu, Palauta IWM, Palauta laitevaihto, Kaukolämpömittari ja Sähkömittarin tuotteenvaihto. Kaukolämpömittari-napin alta löytyy vielä ”Asennus” ja ”Poisto” -linkit kyseisille alasivuille. Sivupalkissa alhaalla näkyy myös sovelluksen viimeisin päivityspäivämäärä, joka määrittyy automaattisesti, kun sovellukseen tehdään päivitys.

Sivun ulkoasussa on pyritty noudattamaan Savon Voiman brändiohjeistoa. Sovelluksen värimaailma koostuu suurimmaksi osaksi Savon Voiman brändiohjeistossa määritellystä värimaailmasta (Savon Voima). Sivun tyyli on myös pyritty tekemään Savon Voiman kotisivujen mukaiseksi. Esimerkiksi napit ja tekstikentät ovat pyöristettyjä. Napeissa ja tekstikentissä on myös samanlainen väriteema, kuin Savon Voiman kotisivuilla on käytetty. (Savon Voima, ei pvm) Fonttina on käytetty Savon Voiman brändiohjeistossa mainittua Century Gothic -fonttia (Savon Voima).

Jokainen lomake koostuu erilaisista kentistä, infonapeista ja Lähetä-napista. Kenttiä voi olla tavallinen tekstikenttä, päivämäärävalitsin tai pudotusvalikko, jossa arvot ovat valmiina valittavaksi. Jokaisella kentällä on myös oma infonappi, jota painamalla aukeaa tekstilaatikko. Tekstilaatikosta saa tietoa minkälaista dataa kyseiseen tekstikenttään tulee syöttää ja mistä kyseinen tieto voisi mahdollisesti löytyä. Osassa tekstilaatikoista on myös sisällä nappi, joka aukaisee dialogin. Dialogin sisällä on esimerkkikuvia, mistä kenttiin syötettävä tieto löytyy.



Kuva 10. Dialogi

Jokaisessa lomakkeessa on Lähetä-nappi, jota painamalla lomakkeen tiedot lähtevät Frendsille. Tiedot lähtevät vain silloin, kun lomakkeen tiedot ovat validointisääntöjen mukaiset. Jokaisella kentällä on omanlaisensa validointisääntö. Mikäli syötetty tieto ei ole kyseisen kentän validointisääntöjen mukainen, huomauttaa kenttä tästä punaisella tekstillä ja kertoo, minkälaista tietoa kenttään tulisi syöttää.

Kuva 11. Validointiviesti ilmoittaa, mikäli syöte on kentän validointisääntöjen vastainen.

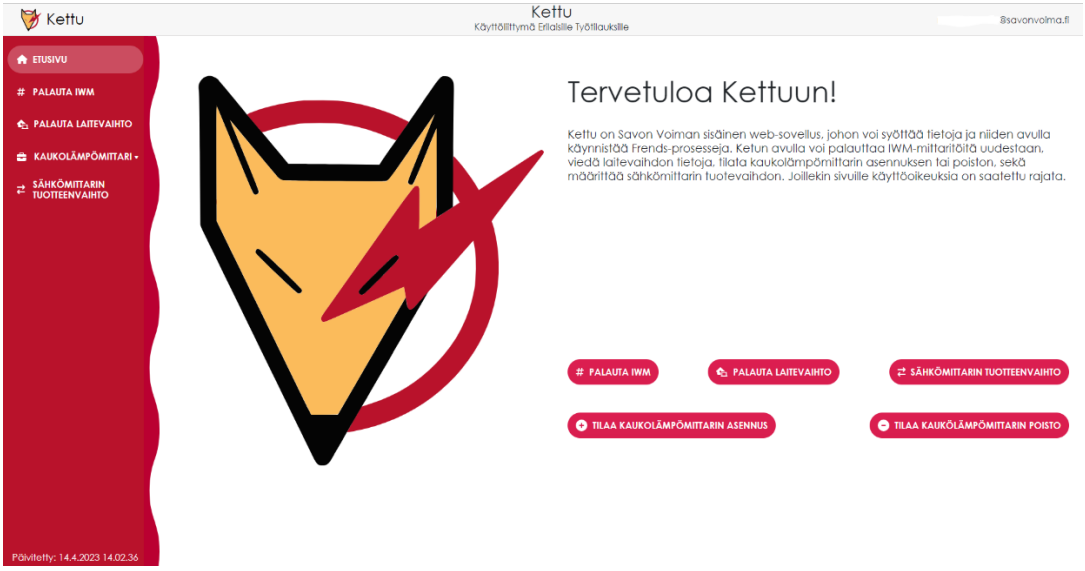
Kun Lähetä-nappia on painettu, nappi häviää ja tilalle tulee loader, joka visualisoi tiedon siirtymistä Frendsille. Loaderin vieressä on myös statusteksti ”Lähetetään…”. Lähetä-nappi piilotetaan käyttäjältä viestin lähetyksen ajaksi, jotta vältyttäisiin useamman viestin lähettämiseltä päällekkäin.

Statusteksti kertoo, mikä tietojen lähetyksen tila on. Kun tieto on mennyt perille onnistuneesti, tulee kyseisen tekstin tilalle vihreällä sana ”Lähetetty”. Mikäli lähetys epäonnistuu, tulee samaiseen kohtaan ”Lähetys epäonnistui” -teksti punaisella. Tällöin statustekstissä näkyy myös viesti Frendsistä, joka kertoo, miksi viestin lähetys epäonnistui. Kun viesti on lähetetty onnistuneesti tai epäonnistuneesti, Lähetä-nappi tulee taas näkyviin ja loader häviää.

Kuva 12. Käyttöliittymä saa vastauksen, miten tietojen lähetys onnistui.

6.2 Etusivu

Kettu-sovellus koostuu kokonaisuudessaan kuudesta eri alasivusta, joista ensimmäinen on Etusivu. Etusivulla ensimmäisenä on Kettu-sovelluksen logo. Etusivulla on myös tietoa kyseisestä sovelluksesta, sekä pikalinkkejä alasivuille.

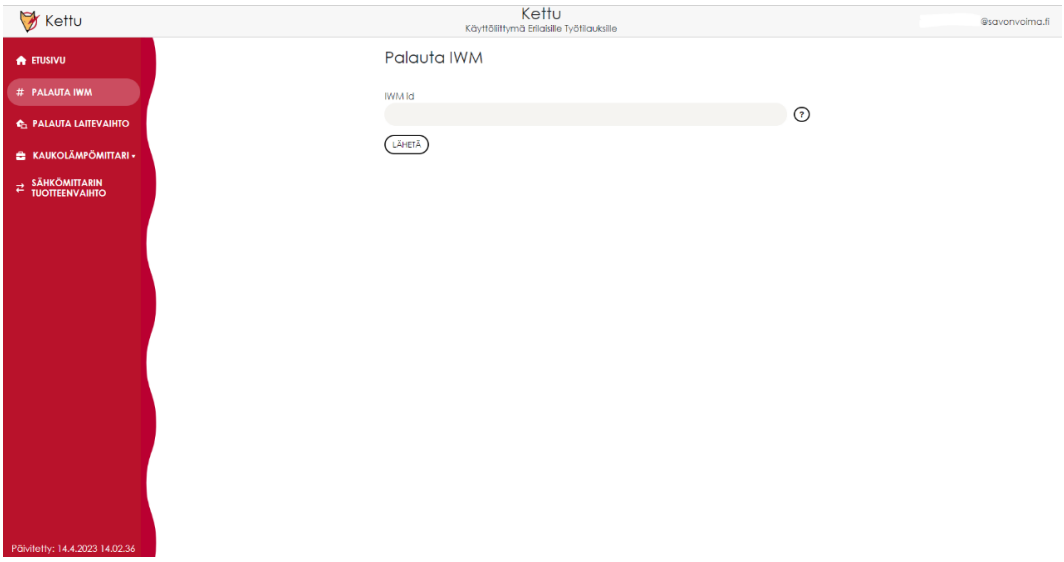


Kuva 13. Etusivu

6.3 Palauta IWM

”Palauta IWM” -nimisellä alasivulla on lomake, joka koostuu vain yhdestä kentästä. Kenttä on nimeltään ”IWM id”, johon tulee syöttää Headpower IWM:stä tai CRM:stä löytyvä IWM id. Kenttä hyväksyy vain numeroita. Merkkijonon tulee olla myös täsmälleen seitsemän merkin pituinen, jotta tieto suostutaan lähettämään.

”IWM id” -kentällä on lisäksi infonappi, jota painamalla aukeaa infolaatikko. Infolaatikosta löytyy tietoa minkälaista dataa ”IWM id” -kenttään tulisi syöttää. Infolaatikon sisällä on myös nappi, joka aukaisee dialogin. Dialogin sisällä on esimerkkikuvat siitä, mistä ”IWM id” -kenttään syötettävän tiedon tulisi löytyä.



Kuva 14. Palauta IWM

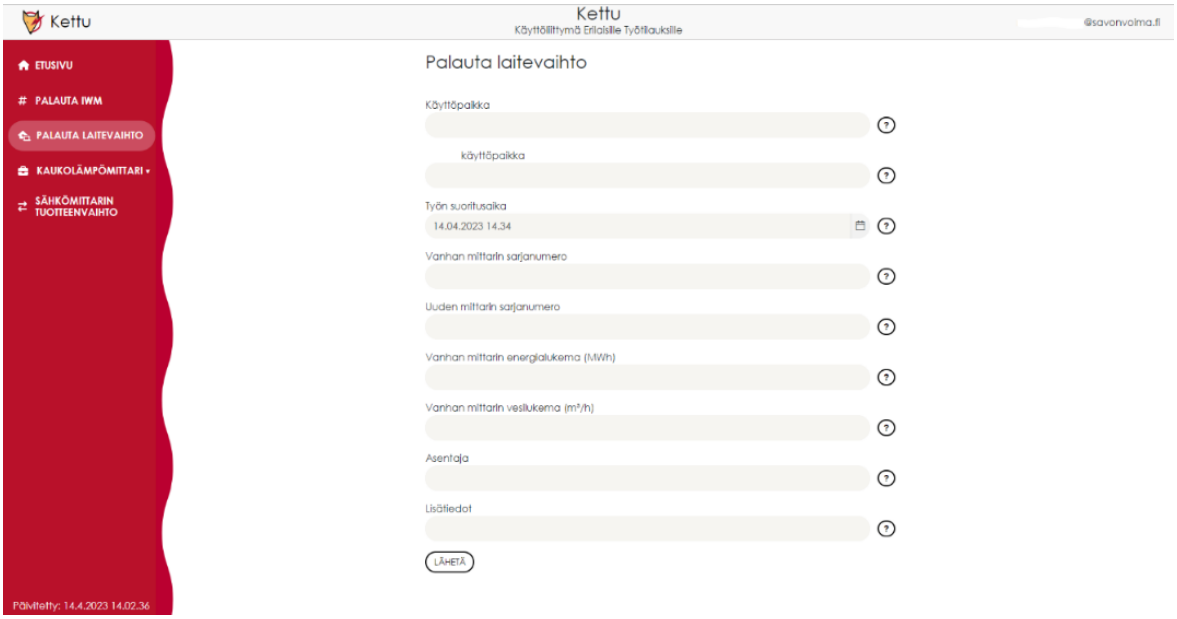
6.4 Palauta laitevaihto

”Palauta laitevaihto” -lomakkeessa on yhteensä yhdeksän kenttää. Ensimmäinen kenttä on nimeltään ”Käyttöpaikka”, johon voi syöttää ainoastaan numeroita. Toinen kenttä on ”Vaihtoehtoinen käyttöpaikka”, jonka validointi on samanlainen kuin “Käyttöpaikka”-kentällä.

Kolmas kenttä on nimeltään ”Työn suoritusaika”. Tämä kenttä on DateTime-tyyppinen, eli siihen syötetty aika on muodossa dd.MM.yyyy HH.mm.ss. Tämä tarkoittaa siis päivämäärä, kuukausi, vuosi ja tunnit, minuutit, sekä sekunnit. Kenttään pystyy syöttämään ajan kirjoittamalla, mutta kentän reunassa on myös painike, josta saa auki kalenterinäkymän. Sen kautta voi valita oikean päivämäärän, sekä kellonajan. Kellonajasta pystyy valitsemaan vain tunnit ja minuutit. Minuutit ovat vain vartin välein valittavissa.

Neljäs ja viides kenttä on ”Vanhan mittarin sarjanumero” ja ”Uuden mittarin sarjanumero”. Kumpikin kenttä hyväksyy vain numeroita syötteenä. Kuudes kenttä on ”Vanhan mittarin energialukema”. Kentän otsikkoon on sulkuihin mainittu, että kyseinen arvo tulee syöttää yksikössä MWh eli megawattituntia. Kyseiseen kenttään pystyy syöttämään vain numeroita, mutta myös pilkun. Pilkku ei kuitenkaan ole pakollinen. Seitsemännessä kentässä nimeltä ”Vanhan mittarin vesilukema” on sama validointi, kuin edellisellä kentällä. Kentän otsikossa on mainittu, että arvo halutaan yksikössä m^3/h eli kuutiometriä tunnissa.

Kahdeksas kenttä on ”Asentaja”-niminen kenttä. Kyseiseen kenttään tulee syöttää asentajan nimi. Kentässä ei ole muuta validointia, kuin että se on pakollinen. Viimeinen kenttä on ”Lisätiedot”-kenttä, johon voi halutessaan syöttää lisätietoja. Kyseinen kenttä ei kuitenkaan ole pakollinen, kun taas kaikki muut kentät kyseisessä lomakkeessa ovat. Mikäli ”Lisätiedot”-kenttä jätetään tyhjäksi, se lähettää Frendsille arvona tyhjän string-tyyppisen muuttujan eli pelkät lainausmerkit.



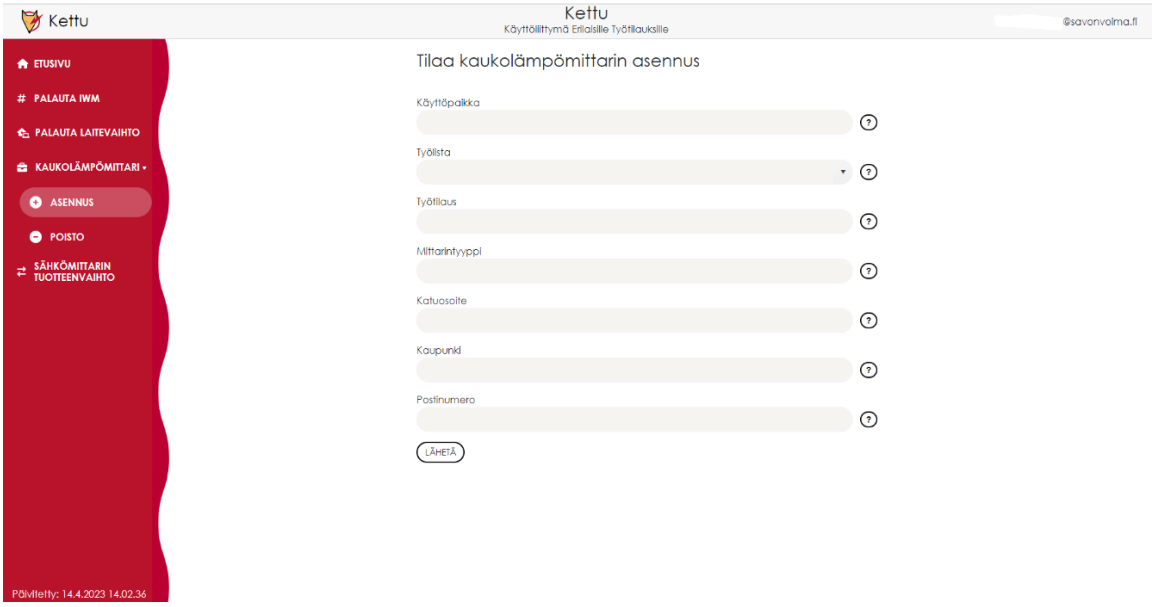
Kuva 15. Palauta laitevaihto

6.5 Kaukolämpömittari

6.5.1 Kaukolämpömittarin asennus

”Kaukolämpömittarin asennus” -lomake sisältää seitsemän eri kenttää. ”Käyttöpaikka”-kenttä on samanlainen kuin edellisessä lomakkeessa, eli siihen pystyy syöttämään vain numeroita. ”Työlista”-kenttä on pudotusvalikkotyyppinen kenttä, jota painamalla aukeaa lista valmiita arvoja, joista valitaan yksi. Seuraavat kentät ovat ”Työtilaus” ja ”Mittarintyyppi” -kentät, joilla ei ole erikseen minkäänlaista validointia, joten niihin syötettävä data voi olla mitä vain. Myös ”Katuosoite”-kentälle ei

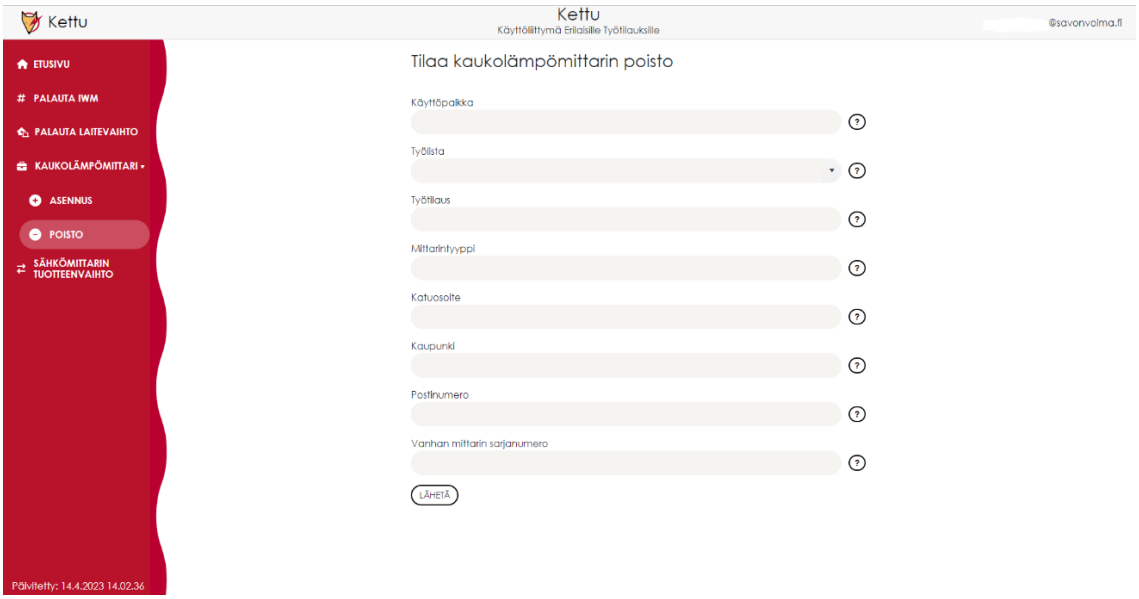
ole asetettu validointia. ”Kaupunki”-kenttään ei ole mahdollista syöttää numeroita. ”Postinumero”-kenttään on mahdollista syöttää vain numeroita. Merkkijonon tulee olla myös tasan viisi merkkiä pitkä. Lomakkeen kaikki kentät ovat pakollisia.



Kuva 16. Tilaa kaukolämpömittarin asennus

6.5.2 Kaukolämpömittarin poisto

”Kaukolämpömittarin poisto” -lomake on muuten täysin samanlainen ”Kaukolämpömittarin asennus” -lomakkeen kanssa, mutta ”Kaukolämpömittarin poisto” -lomakkeessa on lisäksi ”Vanhan mittarin sarjanumero” -kenttä. ”Vanhan mittarin sarjanumero” -kenttään voi syöttää ainoastaan numeroita. Lomakkeen kaikki kentät ovat pakollisia.

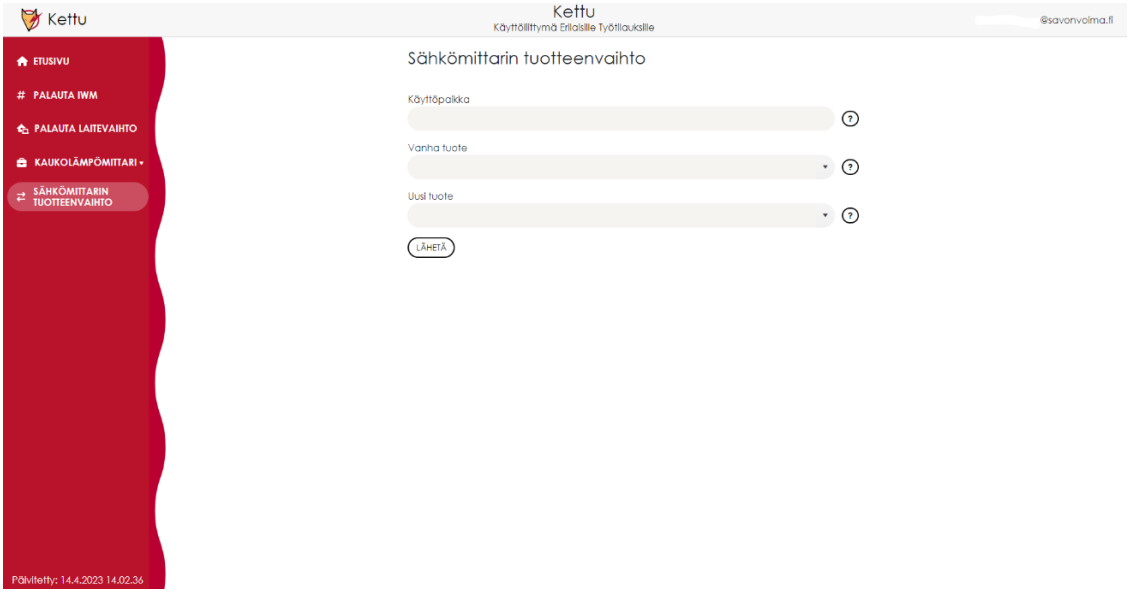


Kuva 17. Tilaa kaukolämpömittarin poisto

6.6 Sähkömittarin tuotteenvaihto

”Sähkömittarin tuotteenvaihto” -lomakkeella on kolme kenttää, joista ensimmäinen on ”Käyttö-paikka”. ”Käyttöpaikka”-kentällä on samanlainen validointi, kuin muiden lomakkeiden ”Käyttö-paikka”-kentillä, eli se sallii syötteenä ainoastaan numeroita. ”Vanha tuote” ja ”Uusi tuote” -kentät

ovat pudotusvalikkotyyppisiä kenttiä. ”Vanha tuote” ja ”Uusi tuote” -kenttien arvot eivät voi olla samoja keskenään. Mikäli käyttäjä valitsee samat arvot kumpaankin, lomake ilmoittaa asiasta, eikä anna viedä tietoja eteenpäin, ennen kuin arvot eroavat toisistaan. Kaikki kyseisen lomakkeen kentät ovat pakollisia.



Kuva 18. Sähkömittarin tuotteenvaihto

7 TIETOJEN YLLÄPITÄMINEN JSON-TIEDOSTOSSA

Kettu-sovellus rakentuu siten, että se hakee latautuessaan joitain komponenttien tietoja JSON-tiedostosta. JSON (JavaScript Object Notation) on tiedonsiirtomuoto. JSON koostuu nimi/arvo -pareista, jotka erotetaan aina kaksoispisteellä. Arvona voi toimia myös objekti, joka koostuu toisista nimi/arvo -pareista. Objekti ympäröidään aaltosuluilla ja nimi/arvo -parit erotetaan toisistaan pilkulla. Sekä ihmisen, että koneen on helppo lukea ja kirjoittaa JSON:ia. JSON ei myöskään ole riippuvainen ohjelmointikielestä. (JSON, ei pvm)

JSON-tiedostoon on lueteltu muun muassa kenttien otsikot, infokenttien tekstit ja pudotusvalikoiden arvot. Tietojen säilyttäminen JSON-tiedostossa edesauttaa sivun muokattavuutta, eli dynaamisuutta. JSON-tiedoston kautta voidaan lisätä, muokata tai poistaa: kenttiä, kenttien otsikoita, pudotusvalikoiden arvoja, validointeja, sekä infokenttien tekstejä.

Arvojen säilytys JSON-tiedostossa auttaa myös siihen, ettei muutosten käyttöönotto vaadi erillistä päivityskatkoa. Tämä on siis mahdollista vain silloin, kun JSON-tiedostoa muokataan suoraan palvelimelta käsin. Seuraavan kerran kun sivu latautuu, se hakee uudet JSON-tiedostoon muutetut tiedot ja lataa sivun niiden perusteella. Tällöin sivun päivittäminen ei haittaa myöskään käyttäjiä.

Alla olevassa kuvassa (kuva 19) on "Kaukolämpömittarin asennus" ja "Kaukolämpömittarin poisto" -lomakkeissa käytetty "Kaupunki"-kentän JSON muotoinen objekti. Jokaisella kentällä on suhteellisen identtiset objektit, mutta tietenkin nimi/arvo -parien arvot ovat jokaisella kentällä yksilölliset. Joitain boolean muuttujia joudutaan säilyttämään kenttien JSON muotoisissa objekteissa. Tämä mahdollistaa sen, että validointiteksti tai infolaatikko tulee esiin oikean kentän kohdalta. Poikkeuksena muihin kenttiin, pudotusvalikko kentillä on lisäksi lueteltu pudotusvalikon arvot kenttien JSON muotoisissa objekteissa.

```
"City": {
  "Variable": "City",
  "Label": "Kaupunki",
  "Value": "",
  "Required": true,
  "Validation": "^[^0-9]+$",
  "ValidationOpen": true,
  "ValidationMessage": "Merkkijono ei saa sisältää numeroita. Kenttä on pakollinen.",
  "InfoDivOpen": true,
  "InfoMessage": "Kaupunki",
  "InputType": "Text"
},
```

Kuva 19. JSON muotoinen objekti "Kaupunki"-kentälle

8 JATKOKEHITYS

Sovellusta tehdessä tuli paljon ideoita ja ominaisuuksia, joita sivulla voisi olla. Näitä täytyi kuitenkin ryhtyä rajaamaan, sillä opinnäytetyöstä olisi muuten tullut liian laaja ja aika ei olisi riittänyt niiden toteuttamiseen. Tähän on siis koottu mahdollisia ideoita ja ominaisuuksia, joita sovellukseen voisi jälkikäteen toteuttaa.

Yhtenä jatkokehitysideana oli, että sivu tultaisiin upottamaan CRM-ohjelmistoon, jota Savon Voiman asiakaspalvelijat käyttävät. CRM (customer relationship management) tarkoittaa asiakkuudenhallintaa (Microsoft, ei pvm). Sovellukseen olisi hyvä saada kielivalikko, josta käyttöliittymän kieli vaihtuisi englanninkieliseksi tarvittaessa. Englanninkielinen käyttöliittymä edistäisi sovelluksen helppokäyttöisyyttä ja selkeyttä. ”Palauta IWM” -sivulle haluttaisiin myös nappi, jota painamalla sivulle tulisi muita tietoja ”IWM id” -kenttään syötetyn numerosarjan perusteella. Näiden tietojen perusteella voitaisiin varmistaa, että syötetty IWM id on oikea.

Kettu-sovelluksesta tulisi entistä helppokäyttöisempi, mikäli siinä olisi erikseen sivun muokkaukseen tarkoitettu alasivu, jolle vain määrätyt henkilöt pääsisivät. Sivua pystytään muokkaamaan tällä hetkellä helposti JSON-tiedoston kautta, mutta käyttäjälle sivun muokkaus olisi vieläkin yksinkertaisempaa oman muokkaukseen tarkoitetun käyttöliittymän kautta. Tietojen säilyttäminen JSON-tiedostossa edesauttaa jo mahdollisesti tulevaisuudessa tehtävää muokkaukselle tarkoitettua sivua, sillä komponenttien tietoja tulisi todennäköisesti silloinkin säilyttää JSON-tiedostossa, jotta sivu muokkautuisi helposti.

Sovellukseen tullaan lisäämään myöhemmin lisää alasivuja eli tässä tapauksessa lomakkeita. Tai esimerkiksi pudotusvalikoihin voidaan tulla lisäämään lisää arvoja tai lomakkeisiin uusia kenttiä. Ongelmana tällä hetkellä on se, ettei lomakkeeseen pysty lisäämään kenttää JSON-tiedoston kautta, mikäli lomakkeessa ei jo ole sen tyyppistä kenttää olemassa. Eli esimerkiksi ”Palauta IWM” -lomakkeeseen ei voi lisätä pudotusvalikkoa, sillä siinä ei sellaista tällä hetkellä ole, mutta ”Sähkömittarin tuotteenvaihto” -lomakkeelle sellaisen pystyy lisäämään, koska lomake sisältää jo pudotusvalikoita. Kaikkiin lomakkeisiin pystyy kuitenkin lisäämään perus tekstikentän JSON-tiedoston kautta. Mahdollisesti tulevaisuudessa toteutettava Kettu-sovelluksen muokkaussivu voisi ratkaista myös tämän ongelman.

9 YHTEENVETO JA POHDINTA

Työ alkoi virallisesti 4.1.2023, kun opinnäytetyön aloituspalaveri pidettiin ohjaavan opettajan ja työn toimeksiantajan kanssa. Palaverissa käytiin läpi mitä tullaan tekemään, miksi tullaan tekemään ja miten tullaan tekemään. Nämä asiat katsottiin vielä tarkemmin läpi työn toimeksiantajan kanssa, kun työtä lähdettiin toden teolla tekemään.

Työ eteni hyvin ja tasaiseen tahtiin. Sivu rakentui ensin siten, että se käytti modeleita. Model on tiedonsiirto-objekti, jonka avulla tietoa siirretään toiseen palveluun (dotnet, 2021). Kun sivu oli siinä vaiheessa, että sen perusrakenne oli pystyssä, muutettiin se hakemaan tietyt perustiedot modelin sijaan JSON-tiedostosta. Sivun olisi voinut rakentaa alusta asti hakemaan tietoja suoraan JSON-tiedostosta, mutta mielestäni käyttämäni tapa oli parempi. Tämä siksi, että suurin osa asioista oli itselle uutta. Oli hyvä, että sai ensin tutustua Blazorin tapaan rakentaa sivuja ja sen jälkeen vasta muokata sivu hakemaan tietoja JSON-tiedostosta. Koska tiedot haettiin modelin sijaan JSON-tiedostosta, ei validointejakaan voinut käyttää Blazorin tapaan, sillä validointisäännöt ja validointiviestit ylläpidettiin myös modeleissa. Sen sijaan jokaisen kentän validointisäännöt ja -viestit tuli siirtää ylläpidettäväksi JSON-tiedostoon, ja niiden toiminnallisuudet tuli tehdä itse.

Tietojen hakeminen JSON-tiedostosta oli myös uusi opittava asia. Tämä oli lisäksi ehkä vaikein osuus tässä projektissa, sillä materiaalia löytyi tähän asiaan niukasti. Myös Blazorin tapa toteuttaa web-sovelluksia erosi hieman siitä mihin oli tottunut, joten tämä tuotti välillä hieman hankaluuksia. Ongelmat ratkesivat kuitenkin hyvin ja tarvittaessa sain apua Savon Voiman työntekijöiltä.

Koska aikaa oli rajallisesti, joutui joitain asioita hieman karsimaan. Osa koodin toteutuksesta olisi voitu tehdä ehkä järkevämmälläkin tavalla, mutta sen toteutukseen olisi kulunut enemmän aikaa. Mikäli toteutus olisi tehty toisella tavalla, olisi se tarvinnut tarkempaa suunnittelua ja opiskelua. Ajan rajallisuuden takia päätettiin kuitenkin, että tärkeintä oli saada sovelluksesta toimiva ja sellainen kuin oli suunniteltu.

Kaikki tärkeimmät ja vaaditut ominaisuudet Kettu-sovellukseen saatiin rakennettua. Sovellukseen tuli myös joitain myöhemmin ideoituja ominaisuuksia. Ehkä ainut ominaisuus, joka jäi hieman vajaaksi, oli sivun muokattavuus ja nimenomaan uuden alasivun lisääminen. Mikäli uuden alasivun tekeminen olisi haluttu helposti muokattavaksi henkilölle, jolla ei ole kokemusta ohjelmoinnista, olisi sovelluksen muokkaukseen täytynyt tehdä erillinen muokkaussivu. Tämä kuitenkin nähtiin liian laajaksi kokonaisuudeksi ja ehkä vähän turhaksi ominaisuudeksi, sillä työn toimeksiantajalta löytyy osaamista sivun muokkaukseen koodista käsin.

Työ oli loppujen lopuksi onnistunut. Kettu-sovelluksen ensimmäinen version julkaistiin palvelimella virallisesti 14.4.2023. Sovellus on siis jo käyttövalmis ja tarpeen tullen Frends-prosessit käynnistetään jatkossa Kettu-sovelluksen kautta. Aika näyttää kuinka sovellusta tullaan jatkossa kehittämään.

LÄHTEET

Ahlstén, V. (2022). *Ajanvarauksen automatisointi Frends-integraatioilla*. Haettu 21. 1. 2023 osoitteesta Theseus: https://urn.fi/URN:NBN:fi:amk-2022112423860

Boyarko, E. (11. 1. 2023). *Top 10 web development frameworks: best options in 2023*. Haettu 19. 1. 2023 osoitteesta Upsilonit: https://www.upsilonit.com/blog/top-10-web-development-frameworks-best-options-to-choose

dotnet. (18. 8. 2021). *Forms and Capturing User Data [10 of 11] | Blazor for Beginners*. Haettu 17. 3. 2023 osoitteesta You Tube: https://www.youtube.com/watch?v=H6-R7c0h1fY&list=PLdo4fOcmZ0oUJCA3DCzKT79Oe3kdKEceX&index=10

dotnet. (17. 8. 2021). *What is Blazor [1 of 11] | Blazor for Beginners*. Haettu 2. 3. 2023 osoitteesta You Tube: https://www.youtube.com/watch?v=9BhbGbTsyeE&list=PLdo4fOcmZ0oUJCA3DCzKT79Oe3kdKEceX&index=1

Fell, J. (19. 1. 2023). *5 Best CMS Platforms In 2023*. Haettu 20. 1. 2023 osoitteesta Entrepreneur: https://www.entrepreneur.com/guide/small-business-tools/5-best-cms-platforms-in-2023

Folcan. (ei pvm). *Mikä on CMS?* Haettu 20. 1. 2023 osoitteesta Folcan: https://folcan.fi/mika-on-cms/

Frends. (ei pvm). *About us*. Haettu 28. 3. 2023 osoitteesta Frends: https://frends.com/about-us

Frends. (ei pvm). *Frends*. Haettu 18. 2. 2023 osoitteesta Frends: https://frends.com/

HeadPower. (ei pvm). Haettu 18. 2. 2023 osoitteesta HeadPower: https://headpower.fi/iwm-2/

HeadPower. (ei pvm). *HeadPower Oy*. Haettu 28. 3. 2023 osoitteesta HeadPower: https://headpower.fi/headpower-oy/

HiQ. (ei pvm). *Yritys*. Haettu 28. 3. 2023 osoitteesta HiQ: https://hiq.fi/yritys/

JSON. (ei pvm). Haettu 17. 3. 2023 osoitteesta JSON: https://www.json.org/json-en.html

Kantola, M. (2020). *Projektipalvelutoiminnan web-lomakkeen suunnittelu ja toteutus*. Haettu 24. 1. 2023 osoitteesta Theseus: https://urn.fi/URN:NBN:fi:amk-2020113025059

Kügelgen (von Kügelgen), M.;& Laukkonen, V. (2020). *Kaikki koodaa : päivitä itsesi - Käytännön opas ajankohtaisiin digitaitoihin.* Helsinki: Into Kustannus Oy

Ludwig, D. (2022). *Slackbot-toteutus Frends-integraatioalustalla*. Haettu 22. 1. 2023 osoitteesta Theseus: https://urn.fi/URN:NBN:fi:amk-2022112924805

Microsoft. (15. 9. 2022). *What is Azure Active Directory?* Haettu 15. 1. 2023 osoitteesta Microsoft: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis

Microsoft. (13. 9. 2022). *What is the Azure portal?* Haettu 16. 3. 2023 osoitteesta Microsoft: https://learn.microsoft.com/en-us/azure/azure-portal/azure-portal-overview

Microsoft. (28. 2. 2023). *What is Azure Service Bus?* Haettu 24. 3. 2023 osoitteesta Microsoft: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview

Microsoft. (ei pvm). *Blazor*. Haettu 26. 2. 2023 osoitteesta Microsoft: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor

Microsoft. (ei pvm). *IIS*. Haettu 3. 4. 2023 osoitteesta IIS: https://www.iis.net/

Microsoft. (ei pvm). *Mitä on CRM?* Haettu 23. 4. 2023 osoitteesta Microsoft: https://dynamics.microsoft.com/fi-fi/crm/what-is-crm/

Microsoft. (ei pvm). *What is a virtual machine (VM)?* Haettu 3. 4. 2023 osoitteesta Microsoft: https://azure.microsoft.com/en-au/resources/cloud-computing-dictionary/what-is-a-virtual-machine/

Mozilla. (ei pvm). *HTTP*. Haettu 3. 4. 2023 osoitteesta MDN: https://developer.mozilla.org/en-US/docs/Web/HTTP

Savon Voima. (ei pvm). Haettu 3. 3. 2023 osoitteesta Savon Voima: https://savonvoima.fi/

Savon Voima. (ei pvm). *Media*. Haettu 3. 3. 2023 osoitteesta Savon Voima: https://savonvoima.fi/wp-content/uploads/2020/11/SV-brandiohjeisto.pdf

Savon Voima. (ei pvm). *Savon Voima*. Haettu 4. 4. 2023 osoitteesta Savon Voima: https://savonvoima.fi/tietoa/savonvoima/

Tahvanainen, M. (2018). *Tietovarastojen Validiteetti*. Haettu 6. 2. 2023 osoitteesta Helda: http://urn.fi/URN:NBN:fi-fe201804208644

Tilli, T. (2022). *eVikavihko : sovellus koneistajan ja huoltohenkilökunnan työn tukemiseksi konepajaympäristöön*. Haettu 28. 1. 2023 osoitteesta Theseus: https://urn.fi/URN:NBN:fi:amk-202203314284

WiseMaster. (ei pvm). *WiseMaster*. Haettu 28. 3. 2023 osoitteesta WiseMaster: https://www.wisemaster.fi/