Sovellus mehiläistarhaajien käyttöön Microsoft Power Appsilla

Tua Ek-Taiminen

Haaga-Helia ammattikorkeakoulu

Opinnäytetyö

2023

Tradenomin tutkinto

Tiivistelmä

Tekijä

Ek-Taiminen, Tua

Tutkinto

Tradenomi

Opinnäytetyön nimi

Sovellus mehiläistarhaajien käyttöön Microsoft Power Appsilla

Sivu- ja liitesivumäärä

32 + 8

Aihe tälle opinnäytetyölle syntyi käytännön tarpeesta tehdä rajatulle joukolle mehiläistarhaajia sovellus, jolla he voivat dokumentoida tekemiään hoitotoimenpiteitä neljän mehiläispesän muodostamassa tarhassa. Mehiläistarhaajien tarkoitukseen sopivaa, maksutonta suomenkielistä sovellusta ei ollut entuudestaan saatavilla.

Opinnäytetyöni on toiminnallinen ja se koostuu raportista ja sovelluksesta. Työn alkaessa minulle oli selvää, että käyttäisin sovelluksen luomiseen niin kutsuttua koodivapaata sovellustyökalua, alustaa, jonka käyttäminen ei vaadi juurikaan ohjelmointiosaamista tai aiempaa kokemusta sovelluksen tekemisestä. Microsoft Power Apps valikoitui työkaluksi sen maksuttomuuden ja Microsoftin entuudestaan tuttujen sovellusten käyttökokemuksen takia. Saatavilla oli riittävästi tietoa ja tukea sovelluksen luomiseen ja ongelmatilanteisiin. Microsoft Power Apps on ulkoasultaan ja käyttökokemukseltaan hyvin tyypillinen M365-tuoteperheen osa, joten sen käyttäminen oli aikaisemman Microsoft-kokemuksen perusteella intuitiivista.

Työssäni kuvasin vain tämän opinnäytetyön tuotoksena syntyneen sovelluksen työvaiheet, yleisellä tasolla koodivapaiden alustojen määritelmän, niiden käyttämisen hyödyt ja haasteet, sovelluksessani käyttämäni toiminnot pääpiirteissään ja päätin raportointiosuuden pohdintaan. Opinnäytetyöni ei ole tarkoitus olla seikkaperäinen kuvaus Microsoft Power Appsin käytöstä tai sen sisältämistä mahdollisuuksista sovelluksia luotaessa.

Asiasanat

Power Apps, sovelluskehitys, vaatimusmäärittely, no code, low code

Sisällys

1 Johdanto

Opinnäytetyöni tarkoitus on tehdä Microsoft Power Appsilla sovellus, jolla mehiläistarhaajat (myöhemmin tekstissä tarhaajat) voivat seurata hoitamiensa mehiläispesien (myöhemmin tekstissä pesät) kokonaistilannetta, tallentaa reaaliajassa tietoa toteutuneista toimenpiteistä ja luoda, muokata tai tarvittaessa poistaa tehtäviä pesäkohtaisesti. Sovelluksen on tarkoitus helpottaa tarhaajien työskentelyä mahdollistaen reaaliaikaisen tietojen tallentamisen yhtä, kaikille tarhaajille saavutettavaa ja maksutonta alustaa käyttäen.

Tarve kerätä ja tallentaa pesien tietoja tuli ajankohtaiseksi kevään ja kesän 2022 aikana, kun mehiläispesien rakentaminen ja hoitaminen Keravalla alkoi. Toiminta perustui Keravan opiston ja kestävän elämän yhteisön Jalotus Ry:n yhteistyönä järjestämälle mehiläistenhoitokurssille ” Hoida pörriäisiä, hanki hunajaa”, jolle oli 15 aloituspaikkaa. Ryhmä kokoontui Jalotuksen tiloissa Keravan kartanon alueella toukokuun 2022 ja syyskuun 2022 välillä noin kymmenen kertaa, minkä lisäksi pari ryhmäläistä otti enemmän vastuuta mehiläispesien huolehtimisesta ryhmän kokoontumiskerojen välillä ja teki kurssin ohjaajan opastamana tarvittavia toimenpiteitä pesillä kurssin päättymisen jälkeen vapaaehtoispohjalta. Keväällä 2023 järjestetään samanlainen aloittelijoille soveltuva mehiläistenhoitokurssi, sekä sen rinnalle jatkokurssi vuoden 2022 alkeiskurssin käyneille.

Kurssin tapaamiskertoja muokattiin alkuperäisistä aikatauluista tilannekohtaisesti poiketen sen mukaan, millainen tilanne pesissä oli. Pesien tilanteeseen vaikutti merkittävästi niiden sijainti, ympäristön olosuhteet ja sää, eikä jälkimmäiseen kahteen voi merkittävästi vaikuttaa. Satokauden lähestyessä loppukesällä heinä-elokuun vaihteessa mehiläispesissä tehtiin tiheämmin mm. kerättäväksi soveltuvien, peitettyjen hunajakakkujen laskentaa. Pesien tiheälle tarkkailulle ja on perusteltu syy, sillä liian aikaisin kerätty, peittämätön ja kostea hunaja ei ole ominaisuuksiltaan oikeanlaista, jolloin mm. sen maku, koostumus ja säilytys kärsivät merkittävästi ja hunaja voi muuttua nopeasti käydessään käyttökelvottomaksi.

Pesien erilaisten hoitotoimien tarve vaihtelee vuodenaikojen sekä pesien tilanteen mukaan, mutta jokaisella pesäkäynnillä tulisi tarkistaa onko pesässä emo eli kuningatar, tehdä karkea arvio mehiläisten määrästä, käyttäytymisestä ja lisätilan tarpeesta hunajalle tai lisääntymiselle, sekä mehiläisten ruokatilanne. Käynnin päätteeksi tulisi tehdä pesäkohtaista kirjanpitoa, josta tulisi ilmetä ainakin tehdyt havainnot ja laatikoiden lisäykset. (Savolainen 2016, 91–92.)

Viestintä tarhaajien välillä tapahtuu ja on tapahtunut tähän asti erilaisilla sosiaalisen median alustoilla, tekstiviestitse tai puhelinkeskusteluin. Tehdyistä toimenpiteistä, esimerkiksi tarkastuskäynneistä, havaituista poikkeamista pesän toiminnassa, hunajasadon määrästä, tuholaistorjunnoista tai talviruokinnasta ei ole mitään järjestelmällisiä muistiinpanoja, tieto on ollut ainoastaan tarhaajien

muistin varassa ja erilaisia toimenpiteitä on jälkikäteen tarkastettu mm. puhelimilta valokuvista ja viesteistä.

Tällä hetkellä ei ole saatavilla tarhaajien tarpeisiin sopivaa valmista, maksutonta alustaa tai sovellusta, eikä käytetyistä viestintäsovelluksista mikään varsinaisesti kyennyt vastaamaan tarpeeseen dokumentoida pesien tietoja ja tarhaajien työtä, sillä niistä puuttui myös mahdollisuus tehdä tulevaisuuteen tehtäviä ja tarkastella aiemmin tehtyjä toimenpiteitä helposti.

Opinnäytetyön tuotoksena syntyvä sovellus on tarkoitettu ensisijaisesti rajatulle, muutamasta vapaaehtoisista koostuvalle joukolle tarhaajia, jotka huolehtivat neljästä mehiläispesästä Keravalla. Aktiivisen tarhaajajoukon kasvaessa sovellusta voi tulevaisuudessa käyttää suurempikin määrä mehiläisharrastajia, kuitenkin niin että sen käyttö on edelleen vain Jalotus ry:n ja Keravan opiston yhteistyössä järjestämien kurssilaisten käytettävissä. Mehiläispesien tarkka ja reaaliaikainen dokumentointi auttavat tarhaajia toteuttamaan tarhamehiläisten hoitoon liittyvät, rutiininomaiset toimenpiteet, välttämään tarpeettomat ja mehiläisyhdyskuntien elämää häiritsevät käynnit mehiläispesillä sekä havaitsemaan mahdolliset puuttumista vaativat ongelmatilanteet ajoissa edistäen mehiläisyhdyskuntien hyvinvointia.

Sovelluksen on tarkoitus sisältää tarhaajien tärkeimmiksi ja työtään eniten helpottaviksi kokemansa toiminnot ja sovellusta edeltävä vaatimusmäärittely on tehty ketterän ohjelmistokehityksen menetelmiä soveltaen. Power Apps valikoitui sovellusalustaksi sen nk. no code/low code -ominaisuuksien, saatavilla olevan lähdemateriaalin, oppaiden ja helpon käytettävyyden, sekä mobiiliyhteensopivuuden vuoksi. Halusin käyttää sellaista alustaa, jota pystyn tarvittaessa muokkaamaan tarhaajien tarpeiden muuttuessa tai tarkentuessa myös opinnäytetyön valmistumisen jälkeen.

Termiparilla no code/low code viitataan uusien sovellusten nopeaan kehittämiseen sovellustyökaluja käyttäen ja ne vaativat tyypillisesti hyvin vähän koodaamista tai aikaisempaa kokemusta sovellusten kehittämisestä. Power Apps ei ollut minulle entuudestaan tuttu työkalu, joten opinnäytetyölleni asettamani tavoitteen, käyttövalmiin sovelluksen lisäksi odotan aiheeseen perehtymisen myötä oppivani uusia taitoja ja pystyväni hyödyntämään näitä taitoja työelämässä.

Microsoftin tuotteiden ja palveluiden käyttäminen vaatii voimassa olevan lisenssin ja tätä opinnäytetyötä tehdessä olisi ollut mahdollista luoda sovellus oppilaitoksen tuotantoympäristöön. Tein sovelluksen kuitenkin omaan Microsoftin Sandbox -ympäristöön, joka on varsinaisesta tuotannosta eristetty ympäristö. Sandbox-ympäristössä voi turvallisesti kehittää ja testata sovelluksia ilman että mitään rakenteita tai testidataa päätyy tuotantoympäristöön. (Leung 2022, luku ”Sharing Apps”.)

En halunnut kehitysvaiheessa olevan sovellukseni päätyvän tuotantoympäristöön missään työvaiheessa, joten käytin Sandbox-ympäristöä sovelluksen luomiseen ja testaamiseen. Sandbox-

ympäristö on kaikille ilmainen kehitysympäristö ja se sisältää kaikki tarvittavat Microsoftin pilvipalvelut sovelluksen rakentamiseen. Sandbox valikoitui ympäristöksi myös siksi, että koulun tarjoaman lisenssin päättyessä voin pitää sovellukseni tallessa ja siirtää tarvittaessa toiseen ympäristöön ilman aikataulupaineita. Otettaessa sovellus tuotantokäyttöön, voidaan sovellus siirtää toiseen ympäristöön Microsoft 365 Power Apps -portaalin export/import-toiminnallisuuden avulla.

Opinnäytetyöni aihe lähtee todellisesta tarpeesta ollen samalla ajankohtainen kokonaisuus sekä pölyttäjäkadon, että luonnon monimuotoisuuden säilyttämisen kannalta. Tarhamehiläiset paikkaavat luonnonvaraisia pölyttäjiä, mutta eivät missään nimessä korvaa niitä. Tarhamehiläisiä voidaan käyttää kaupunkiympäristöissäkin paitsi pölyttämisen tukena, myös kasvien tuholaistorjunnassa kemiallisten torjunta-aineiden asemesta. Aivan pienistä summista ei puhuta, kun tarkastellaan sekä luonnonvaraisten, että tarhattujen pölyttäjien taloudellista merkitystä; Savolaisen (2016, 163) mukaan vuonna 2016 arvioitiin, että hyönteispölytyksen globaali taloudellinen arvo on 153 miljardia euroa, josta yksinään Suomessa viljely- ja puutarhakasvien mehiläispölytyksen tuoma sadonlisäyksen tuoma arvo on 18,3 miljoonaa euroa.

2 Sovelluksen suunnittelu

Sovelluksen luominen edellyttää suunnitelmaa, jossa määritellään ensin sovelluksen käyttötarkoitus ja toteutustapa ennen kuin edetään tekniseen toteutukseen. Opinnäytetyöprosessissani mukailin Luukkaisen & Ilveksen (2022) esittelemää listaa ohjelmiston ohjelmistotuotannon vaiheista soveltaen vaiheet oman työni laajuuden ja käyttäjäryhmän koon huomioiden. Koin vaatimusten määrittelyn helpottavan sovelluksen luomista ja antavan sille tarvittavat raamit.

Sovelluskehityksen vaiheet on jaettu vaatimusten määrittelyyn, suunnitteluun, toteutukseen, testaukseen ja ylläpitoon. Ylläpitovaiheen jätin pois vaiheista pohtien myöhemmin opinnäytetyössäni sovelluksen jatkokehitysmahdollisuuksia. Lisäksi avasin myös lyhyesti ketteriä menetelmiä, käyttöliittymäsuunnittelua, käyttäjäkokemusta, käytettävyyttä ja saavutettavuutta osana ohjelmiston tai sovelluksen kehitysprosessia, koska koin ne olennaisiksi omassa opinnäytetyössäni ja sovelluksen luomisessa.

Sovelsin opinnäytetyöni toiminnallisessa osuudessa ketteriä menetelmiä suunnitteluvaiheesta alkaen; halusin työskentelyn olevan joustavaa, mutta toisaalta myös johdonmukaista. Olin yhteydessä tarhaajiin sovelluksen edetessä; varmistin mitkä toiminnallisuudet ovat tarpeellisia, pyysin tarkennuksia ja varmistin että olin ymmärtänyt oikein. Työhöni ei sisältynyt taloudellisia vaatimuksia tai toimeksiantajan asettamia tavoitteita, joten koin ketterien menetelmien antavan riittävän tuen ja työjärjestyksen opinnäytetyöprosessin aikana.

Ketteryyttä ohjelmistokehityksessä pidetään ensisijaisesti näkemyksenä tai toimintatapana, jota hyödynnetään projektinhallinnan välineenä. Ketteryyden pääperiaatteita ovat jatkuva kehittyminen sekä oppiminen kokeilujen, onnistumisten ja epäonnistumisten kautta, joustavuus, muutosmyönteisyys sekä suora kommunikaatio. Ketterät menetelmät lähtevät siitä oletuksesta, että vaatimukset tulevat todennäköisesti muuttumaan monesta syystä kehitysprosessin varrella, jonka vuoksi kaikkien vaatimusten selvittäminen kattavasti etukäteen ei ole kannattavaa tai edes mahdollista.

Ketterissä menetelmissä tarkoin kuvattuja, yksityiskohtaisia suunnitelmia tai muuttumattomia rakenteita tärkeämpää on kehittää työskentelyä projektin edetessä ja näiden toimintatapojen ansioista ketterien menetelmien katsotaan soveltuvan erityisen hyvin pieniin ja keskikokoisiin luoviin projekteihin. Esimerkkejä ketterien menetelmien viitekehyksistä ovat esim. Scrum ja XP. (Stellman & Greene 2015, luku ”Understanding Agile Values”.)

Ketterät menetelmät poikkeavat ohjelmistosuunnittelussa perinteisesti käytetystä lineaarisesta vesiputousmallista juuri kyvyllään vastata muutostarpeisiin suunnittelun kaikissa vaiheissa. Vesiputousmallissa tyypillisesti määritellään vaatimukset ensin hyvin tarkasti, minkä jälkeen ohjelmisto rakennetaan ja tuote testataan. Ketteryys vastaa tarpeeseen hallita monimutkaisia ja

arvaamattomia projekteja, joihin perinteiset, yksityiskohtaiset tai suunnitelmavetoiset mallit eivät toimi. (Stellman & Greene 2015, luku ”Understanding Agile Values”.)

2.1 Vaatimusmäärittely

Opinnäytetyöni alkoi vaatimusmäärittelyllä, jonka tarkoitus oli määritellä sovelluksen käyttäjien tarpeet ja tuoda esiin ne ongelmat ja vaatimukset, joihin sovelluksella haetaan ratkaisuja. Vaatimusmäärittelyn yhteydessä kävin läpi kaikki ne seikat, joiden selvittämisen koin olevan paitsi olennaisia toimivan lopputuloksen kannalta, myös helpottavan itse sovelluksen luomista ja sen ominaisuuksien määräytymistä.

Luukkaisen ja Ilveksen mukaan vaatimusmäärittely muodostuu ohjelmiston vaatimusten selvittämisestä, dokumentoinnista ja hallinnoinnista ja se tehdään aina ennen ohjelmiston suunnittelua ja toteuttamista. Se miten vaatimukset kerätään, analysoidaan, dokumentoidaan, validoidaan ja miten niitä hallinnoidaan, vaihtelee ohjelmistoprojektin luonteesta ja koosta riippuen. Niin kutsutussa vesiputousmallissa vaatimusmäärittely tehdään kokonaisuudessaan ennen ohjelmiston suunnittelua ja toteutusta, kun taas ketterässä ohjelmistokehityksessä vaatimusmäärittelyä voi tapahtua prosessin edetessä ohjelmiston toiminnallisuuden kasvamisen, tai käyttäjien tarpeiden tarkentumisen myötä. (Luukkainen & Ilves 2022.)

Vaatimukset jaetaan tyypillisesti kahteen luokkaan, toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnallisilla vaatimuksilla kuvaillaan niitä asioita, joita ohjelmistolla voi tehdä ja ei-toiminnallisia vaatimuksia ovat puolestaan mm. ohjelmiston käytettävyys ja tietoturva sekä ohjelmiston toimintaympäristön asettamat rajoitteet. (Luukkainen & Ilves 2022.)

Toiminnallisilla vaatimuksilla määritellään mitä järjestelmä tekee vastatakseen käyttäjän tarpeisiin ja odotuksiin. Toiminnalliset vaatimukset ovat tyypillisesti ominaisuuksia, jotka käyttäjä havaitsee ja ne kuvaavat nimensä mukaisesti suunniteltavan järjestelmän toimintoja. Ne tarjoavat kuvauksen siitä, miten järjestelmän tulisi reagoida tiettyyn komentoon. Toiminnalliset vaatimukset voidaan jakaa edelleen kahteen osaan, toimintaan ja käyttäytymiseen. Toiminto on se, mitä järjestelmä tekee ja käyttäytyminen on sitä, miten järjestelmä sen tekee. (Visure 2023a.)

Ei-toiminnallisilla vaatimuksilla (engl. NFR, non- functional requirements) määritellään ohjelmiston laatukysymyksiä, kuten skaalautuvuutta, suorituskykyä, (tieto)turvallisuutta ja luotettavuutta. Ei-toiminnalliset vaatimukset voivat olla toisinaan hankalia hahmottaa, mutta niiden tarkoitus on toiminnallisten vaatimusten tapaan tärkeitä varmistaa, että sovellus vastaa käyttäjän tarpeita, valittu järjestelmä tai ohjelma on helppo käyttää ja tulevaisuutta ajatellen ylläpitää. (Visure 2023b.)

Omassa työssäni toteutin vaatimusmäärittelyn haastattelemalla kolmea tarhaaja. Haastateltavat saivat kertoa vapaassa keskustelussa tarkentavien apukysymysten avulla mitä haluaisivat sovelluksella tehdä, millaisia ominaisuuksia toivovat sovellukselta ja millaisia tietoja pitävät keskeisinä tarhaamisen kannalta. Tarhaajien tuomat vaatimukset olivat lähinnä toiveita, mutta käsittelin ne vaatimusten tavoin ja jaoin ne toiminnallisiin ja ei-toiminnallisiin.

Toiminnallisia vaatimuksia oli odotettavasti enemmän kuin ei-toiminnallisia, ja ne liittyivät tiiviisti mehiläishoidollisiin toimenpiteisiin mukaillen vuoden kulkua. Tarhaajien toiveena oli voida kirjata ylös tekemänsä toimenpiteet pesien avaamisen yhteydessä. Näitä toimenpiteitä oli kuningattaren löytäminen, muninta, sikiökennot, hunajakakut, peitettyjen hunajakakkujen määrä, hätäkennot, laatikoiden määrä (lisääminen ja poistaminen), sulkuristikkojen lisääminen, talvipohjien tai kesäpohjien vaihtaminen, talviruokinta, lisäruokinta, punkkitorjunnat, muut havainnot, mehiläisten hoitoon tarkoitettujen pukujen ja välineiden huolto ja sijainti.

Ei-toiminnallisena vaatimuksena tuli ilmi helppokäyttöisyys, jota pyysin tarkentamaan. Helppokäyttöisyyttä kuvailtiin mahdollisuudeksi käyttää sovellusta tarvittaessa hanskat kädessä tekemällä kyllä/ei valintoja pesien avaamisen ja hoitamisen yhteydessä niin, että valintoja voi tarvittaessa täydentää tekstillä jälkikäteen. Tämä siksi, että pesien hoitamisen yhteydessä käytetään lähes aina suojapukuja ja etenkin paksuja hanskoja. Kirjaamisen tulee olla reaaliaikaista, mutta tarpeeksi yksinkertaista, jotta se ei vaadi suojavarusteiden riisumista keskellä kymmeniätuhansia aktiivisia mehiläisiä.

Kun vaatimukset on kerätty, ne analysoidaan ja dokumentoidaan. Analysoinnissa tarkastellaan vaatimuksia keskenään; onko niissä ristiriitoja, ovatko ne tarpeeksi kattavia tuoden esiin kaikki käyttöskenaariot? Tässä vaiheessa voidaan myös varmistaa vaatimusten toteuttamisen mahdollistaminen ja taloudelliset ulottuvuudet. Vaatimusten tulisi olla todennettavissa, eli pystytäänkö valmiista järjestelmästä toteamaan noudattaako järjestelmä tätä vaatimusta. (Luukkainen & Ilves, 2022.)

Analysoin vaatimusmäärittelyn yhteydessä tarhaajien haastatteluista saamani toiveet ja vaatimukset ryhmittelemällä ne samankaltaisuuden perusteella toiminnallisiin ja ei-toiminnallisiin vaatimuksiin, eikä niissä esiintynyt keskinäisiä ristiriitaisuuksia.

Vaatimusdokumentin tarpeellisuus korostuu suuremmissa ohjelmistokehitysprojekteissa ja vielä voimakkaammin nk. vesiputousmallissa, jossa mm. sovelluksen hintaan vaikuttaa vaatimusmäärittelyssä kuvattu toiminnallisuus. Vesiputousmallissa toiminnallisuuden muuttaminen määrittelyn hyväksynnän jälkeen lisää työmäärä venyttäen projektin aikataulua ja sen myötä kustannuksia.

Ketteriä menetelmiä käytettäessä vaatimusten dokumentointi ei ole niin korostetussa roolissa ohjelmistokehityksen aikana. (Luukkainen & Ilves, 2022.)

Opinnäytetyössäni tein dokumentoinnin saadakseni koottua niiden pohjalta käyttäjätarinat. Tein dokumentoinnin yksinkertaisesti listaamalla analysoidut ja ryhmitellyt vaatimukset omaksi asiakirjakseen ja niiden perusteella tein lyhyet, yhden lauseen mittaiset käyttäjätarinat.

Vaatimukset on myös oleellista validoida, eli tulee varmistaa, että kerätyt vaatimukset todellakin vastaavat asiakkaan mielipidettä ja kuvaavat sellaista järjestelmää, minkä asiakas kokee tarvitsevansa. Vaatimuksia on myös tavalla tai toisella hallinnoitava, erityisesti jos vaatimukset muuttuvat kesken sovelluskehitysprosessin. Hallinnoinnilla tarkoitetaan esimerkiksi uusien asiakkaalle mieleen tulevien vaatimusten kirjaamista, jo kirjattujen vaatimusten muokkaamista ja vaatimusten priorisointia. Muutospyynnöt käsitellään mahdollisuuksien mukaan osana toteutusta tai jatkokehityksenä. (Luukkainen & Ilves 2022.)

Vaatimusten validointia pyrin varmistamaan esittämällä tarkentavia kysymyksiä vaatimusmäärittelyn yhteydessä varmistaakseni, että ymmärsin oikein ja annoin haastateltaville mahdollisuuden palata asiaan missä vaiheessa tahansa, mikäli heille tulisi jotain lisättävää sovelluksen ominaisuuksiin liittyen.

Käytin projektinhallinnan työkaluna lähinnä uteliaisuudesta Microsoftin Azure DevOpsia. Azure DevOps on parhaimmillaan projekteissa, joissa on mukana työntekijöitä sovelluskehittäjistä projektipäälliköihin, sillä sen avulla voidaan hallita huomattavasti suurempia kokonaisuuksia ja projekteja, kuin mitä omassa yhden hengen sovellusprojektissani oli tarve.

Omassa työssäni tämän tyyppinen ratkaisu toimi kuitenkin hyvin visualisoimaan käyttäjätarinat konkreettisesti tarinoista ratkaisuiksi ja antoi yhdellä silmäyksellä kuvan sovellusprojektini etenemisestä ja aikataulussa pysymisestä. Vaihdoin näkymää sen mukaan mikä vaihe sovelluksen kanssa oli menossa, työn edetessä board-näkymä antoi aikataulua ajatellen paremman kokonaiskuvan, kun taas työskentelyn alkuvaiheessa backlog-näkymä listasi käyttäjätarinat ja ikään kuin keskusteli vaatimusmäärittelyn kanssa.

2.2 Käyttäjätarinat

Opinnäytetyössäni käytin sovelluksen vaatimusmäärittelyssä käyttäjätarinoita (engl. user story) kuvaamaan sovellukselta haluttuja, konkreettisia toiminnallisuuksia. Käyttäjätarinoissa asiakkaiden tai käyttäjien ääni pääsee hyvin kuuluville, niissä käytetään asiakkaiden ja käyttäjien sanastoa ja vältetään ottamassa kantaa tekniseen toteutukseen. Poiketen perinteisestä vaatimusmäärittelystä,

käyttäjätarina on lyhyt lause, joka kuvaa yleensä käyttäjän näkökulmasta, mitä käyttäjä haluaa ohjelmalla tehdä. (Luukkainen & Ilves, 2022.)

Dokumentoinnin pohjalta tein lyhyet, yhden lauseen mittaiset käyttäjätarinat. Käyttäjätarinoita syntyi 6 kpl:

- Tarhaaja voi tarkastella (pesäkohtaisesti) edellisen käynnin merkintöjä.
- Tarhaaja voi lisätä (pesäkohtaisen) tehtävän.
- Tarhaaja voi muokata (pesäkohtaista) tehtävää.
- Tarhaaja voi päivittää pesän tiedot kyllä/ei -valintoja käyttäen.
- Tarhaaja voi lisätä tiedon hunajasadon määrästä ja ominaisuuksista.
- Tarhaaja voi tarkastella, muokata ja lisätä välineitä.

2.3 Käyttöliittymäsuunnittelu ja käyttäjäkokemus

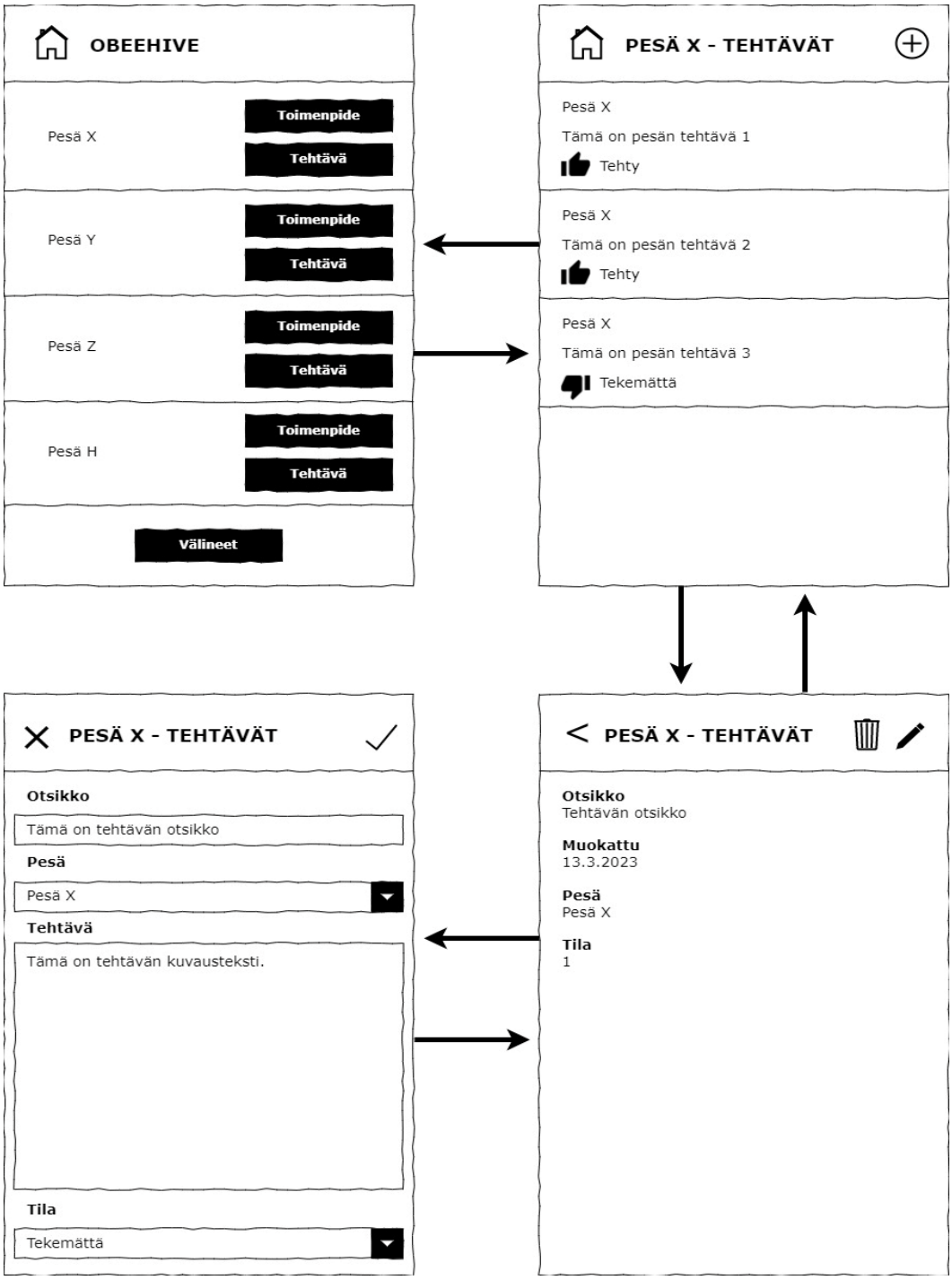
Käyttöliittymäsuunnittelun (engl. User Interface Design, UI) tavoitteena on tehdä palvelun käyttämisestä mahdollisimman yksinkertaista sen käyttäjille ja sitä voidaan kuvailla havainnollistamalla mitä käyttäjä näkee näytöllä sovellusta käyttäessään. Käyttöliittymäsuunnittelussa vaikutetaan visuaalisten keinojen avulla siihen, miten käyttäjä on vuorovaikutuksessa esim. ohjelmiston tai verkkosivustojen kanssa, kuinka vaivatonta tarvittavan tiedon löytäminen on ja kuinka saavutettavissa tieto on. (Hurja 2021.)

Käytin reilusti aikaa tähän sovelluksen luomista edeltävään käyttöliittymäsuunnitteluun. Jokisen (2022) mukaan käyttöliittymäsuunnittelu Power Apps -sovellusta luotaessa parantaa sen käytettävyyttä ja saavutettavuutta merkittävästi; esteettisesti miellyttävä käyttöliittymä on tutkitusti käyttäjäkokemukseltaan parempi. Niin kutsuttu *aesthetic usability effect* kuvaa tätä tutkittua ilmiötä, jossa käyttäjä antaa anteeksi sovelluksen pienet käytettävyysongelmat, mikäli sovellus on intuitiivinen ja luo käyttäjälle kokemuksen ammattimaisesta suunnittelusta.

Sovelluksen näyttöjen suunnittelu ennen sovelluksen luomista on suositeltavaa tehdä rautalankamallilla (engl. wire frame). Rautalankamalli tuo esiin suurimman osan ongelmista jo tässä vaiheessa säästäen työmäärää siinä vaiheessa, kun sovellusta rakennetaan. Rautalankamallin toteutustapa voi olla mitä vain perinteisen piirtäen tehdyn tai hyvin tyylitellyn version väliltä, mutta jos tarkoituksena on myydä sovellus, voi tyylitelty rautalankamalli antaa hyvät lähtökohdat myymiselle. (Jokinen 2022.)

Oma rautalankamallini hahmottui käyttäjätarinoista käsin ja mukaili Power Appsin pohjaa. Kävin tutustumassa Power Appsiin ennen kuin aloitin opinnäytetyön raportin kirjoittamisen, joten tiesin jo etukäteen millaisesta näkymästä sovellustyökalussa lähdetään etenemään. Käsin piirrettyjä

vedoksia ja hahmotelmia syntyi puoli tusinaa, ennen kuin siirsin rautalankamallin maksuttomalla, selainpohjaisella Draw.io -ohjelmalla viimeisteltäväksi. Viimeistelty versio toimi pohjana varsinaiselle sovellukselle ja tuki sovelluksen rakenteen hahmottamisessa työn edetessä. Rautalankamallin käyttäminen oli itselleni sovelluksen rakentamisessa erittäin hyödyllinen apuväline, ikään kuin kuva valmiista palapelistä, jota lähdin rakentamaan osa kerrallaan. Alla esimerkki tekemästäni rautalankamallista (kuva 1), loput rautalankamallin näytöt löytyy liitteistä (liite 1).



Kuva 1. Rautalankamalli sovelluksen Tehtävät-näytöistä.

Käyttöliittymäsuunnittelulle läheinen käsite on käyttäjäkokemus (engl. User experience, UX), jolla tarkoitetaan käyttäjän kokemusta verkkosivuston, mobiilisovelluksen tai minkä tahansa ohjelmiston käytöstä. Käyttäjäkokemus voidaan toteuttaa digitaaliselle tai fyysiselle tuotteelle. Käyttäjäkokemuksessa asiakkaan matka palvelussa määritellään yksityiskohtaisesti ja samalla otetaan huomioon mm. ihmiselle tyypilliset psykologiset toimintatavat ja yrityksen palvelulle asettamat tavoitteet. Käyttäjäkokemussuunnittelussa pyritään vaikuttamaan juuri näihin elementteihin, jotka muovaavat käyttökokemuksen. (Hurja 2021.)

Käyttäjäkokemuksen suunnittelussa tulisi pyrkiä suunnittelemaan interaktiivisia järjestelmiä ja palveluita, joita on miellyttävä käyttää, jotka ovat hyödyllisiä ja parantavat niitä käyttävien ihmisten elämää. Järjestelmien, ohjelmistojen ja sovellusten tulisi olla saavutettavia, käyttökelpoisia ja kiinnostavia. Sen saavuttamiseksi järjestelmien suunnittelun pitäisi olla ihmiskeskeistä, jolloin teknologian sijaan asiakas on suunnitteluprosessin keskipisteenä. Suunnittelussa tulee ottaa huomioon se, että palveluita käyttävien ihmisten kokemukset ja taidot digitaalisista palveluista voivat olla hyvin vähäisiä ja poiketa käyttäjäjoukon sisällä keskenään hyvinkin paljon. (Benyon 2019, 2.)

2.4 Käytettävyys ja saavutettavuus

Käytettävyydellä tarkoitetaan subjektiivista kokemusta, joka käyttäjälle tulee tuotteesta. Benyonin (2019, 108) mukaan korkean käytettävyyden omaava järjestelmä on tehokas niin, että käyttäjä pystyy tekemään sillä asioita sopivalla määrällä vaivaa, se sisältää asianmukaiset toiminnot ja tietosisällön asianmukaisesti ja loogisesti järjestettynä. Järjestelmien, ohjelmistojen ja sovellusten käytettävyyttä voidaan mitata tarkastelemalla, onko niiden avulla helppo tehdä toivottuja asioita, onko järjestelmän käyttäminen turvallista ja saavutetaanko sen käyttämisellä suurta hyötyä.

Suunnitteluvaiheessa tulee tehdä valintoja sen perusteella mikä tuotteen käyttäjäryhmä on, mitä tuotteen käyttämisellä tavoitellaan ja millaisessa käyttöympäristössä tuotetta käytetään. Suunnittelijan on pyrittävä ymmärtämään kohderyhmää, jolle tuote suunnitellaan ja valittava tavoitteet, mitä tuotteelle asetetaan, sekä otettava huomioon ne olosuhteet, millaisessa käyttöympäristössä tuotetta tullaan käyttämään. Hyvä käytettävyys ohjaa käyttäjää tuotteen käytön opettelemisessa ja tekee tuotteen käyttämisestä tehokasta. (Niemelä, 2020.)

Saavutettavuudella tarkoitetaan asiakaslähtöistä näkökulmaa, jossa pyritään huomioimaan mahdollisimman monien erilaisten ihmisten mahdollisuus käyttää verkkosivuja ja mobiilisovelluksia helposti. Saavutettavuutta on ottaa huomioon käyttäjien erilaisuus ja moninaisuus jo digipalveluiden suunnitteluvaiheessa ja toteutuksessa. Parhaimmillaan saavutettavuus parantaa yhdenvertaisuutta digitaalisessa ja voimakkaasti digitalisoituvassa yhteiskunnassa. (Aluehallintavirasto 2023.)

Mobiilisovelluksen saavutettavuutta tukee mm. se, että sovellus on responsiivinen. Responsiivisuudella tarkoitetaan sitä, että sovellus asettuu käytettävälle päätelaitteelle oikein, kuten se on tarkoitettukin. Tekstin tulee olla tarpeeksi suurta, jotta se on helposti luettavissa, sivuston navigointielementtien tulee olla hyvin näkyvissä ja käytettävissä ja lomakkeiden tulisi toimia ongelmitta. Ongelmatilanteissa sovellus antaa käyttäjälle ohjeita toimia oikein tai mahdollisuuden korjata tilanteen. Saavutettavuus mahdollistaa digipalveluiden käyttämisen niiden käyttäjille ja toisaalta myös avaa mahdollisuuden sellaisille käyttäjille, joille digipalvelut eivät ole ennen olleet ajankohtaisia tai mahdollisia käyttää. (Aluehallintavirasto 2020.)

Pyrin huolellisella suunnittelulla pitämään sovelluksessa navigoinnin yksinkertaisena ja loogisena ja rautalankamallin avulla hahmottelin saavutettavuutta. Jokisen ohjeistamana kiinnitin huomiota sovelluksen värimaailmaan välttäen liian voimakkaita, täyteläisiä värejä sekä punaisen ja vihreän rinnakkaista käyttöä. Yhtenäistin typografian käyttämällä vain yhtä fonttia ja kiinnittämällä huomiota ns. visuaaliseen hierarkiaan. Visuaalisella hierarkialla tarkoitetaan tässä yhteydessä otsikoiden ja leipätekstin välistä suhdetta; otsikon tulee erottua ja näin ohjata sovelluksessa navigointia. (Jokinen 2022.)

Omassa projektissani saavutettavuus tuli konkreettisesti mukaan vaatimusmäärittelyn käyttäjätarinoista alkaen. Tarhaajien valmiudet käyttää sovellusta olivat tiedossa etukäteen, kaikilla oli aikaisempaa kokemusta mobiilisovellusten käyttämisestä, ja he pystyivät nimeämään ja kuvailemaan hyvin tarkasti ominaisuuksia, joita sovellukselta toivoivat. Tarhaajien toiveena oli toiveena saada mahdollisimman helppokäyttöinen sovellus.

Muiden kuin kyllä/ei -valintoja vaativien asioiden kirjaaminen sovellukseen tapahtuu pääasiallisesti sellaisissa tilanteissa, joissa suojapuvun käyttäminen ei ole tarpeellista tai vaadi välitöntä kirjaamista, esimerkiksi pesien hoitoon tarvittavien välineiden huolto ja säilyttäminen tai hunajan kerääminen, linkoaminen ja purkittaminen voivat tapahtua täysin erillään pesien tarkastuskäynneistä ja avaamisesta riippumatta tai vasta näiden toimenpiteiden päätteeksi.

3 Koodivapaat kehitysalustat

Koodivapaa (engl. no code tai low code) kehitysalusta on tyypillisesti visuaalinen, pitkälti automatisoitu ympäristö, jossa sovelluksen tekeminen tapahtuu valmiiden elementtien sijoitteluun alustalla. Englanniksi koodivapaa alusta on usein määritelty vielä tarkemmin no code -alustoihin, joissa koodaamista ei vaadita lainkaan ja siitä on erotettu vielä omaksi käsitteekseen low code -alustat, joissa sovellusta voidaan halutessa rikastaa lisäämällä siihen koodia. Tässä opinnäytetyössä käytän koodivapaan alustan käsitettä viitatessani molempiin, sekä no code, että low code -alustoihin.

Koodivapaalla alustalla kehittäjällä on käytössään yleensä laaja valikoima valmiita elementtejä, joita lisäämällä, siirtämällä tai poistamalla kehittäjä voi koota (engl. drag & drop) vaatimukset täyttävän sovelluksen. Koodivapaa alusta sisällyttää elementteihin kaiken tarvittavan taustalla olevan koodin ja tukitehtävät, kuten testauksen ja käyttöönoton. Nämä vähäistä koodaamista vaativat alustat mielletään helpoiksi käyttää ja niiden käyttäminen yksinkertaisiin prosesseihin vapauttaa kehittäjätiimejä keskittämään resurssejaan suurempiin ja monimutkaisempiin projekteihin. (Bigelow 2023.)

Kaikki koodivapaat alustat eivät ole keskenään samanlaisia ja oikean alustan löytyminen tarjonnasta riippuu luonnollisesti käyttäjän tarpeista. Koodivapaiden alustojen tarjoajia on markkinoilla satoja ja määrä kasvaa jatkuvasti. Sopivan alustan kokeilemista ja valitsemista helpottaa onneksi myös helposti verkkohaulla löytyviä sivustoja, jotka ovat tehneet erilaisia kuvailuja ja kategorisointeja alustojen ominaisuuksien selailemiseen.

Alustojen kustannukset vaihtelevat muutaman euron tai dollarin kuukausimaksuista kuusinumeroisiin lukuihin, riippuen alustojen tarjoamista ominaisuuksista ja käyttäjien määrästä. Sopivan koodivapaan alustan löytäminen voi vaatia kokeiluja, mutta yleensä alustojen käyttöönottoon sisältyy maksuton kokeilujakso, jonka aikana käyttäjät voivat testata alustoja ja nähdä, kuinka ne toimivat. On todennäköistä, että jonakin päivänä liiketoimintaketjun jokainen osa on tekoälyn (engl. Artifical intelligence, AI) avulla optimoitu ja koodivapaat alustat ovat yhtä yleisiä kuin tekstinkäsittely- tai taulukkolaskentaohjelmistot nykyään. (Reilly 2021.)

3.1 Koodivapaan alustan hyödyt ja haasteet

Bigelow luettelee koodivapaan alustan käyttämisen etuja ja niistä nopeus on yksi koodivapaan alustan ehdottomista hyödyistä verrattuna ns. perinteiseen koodaamiseen. Yksittäisten koodirivien kirjoittaminen on hitaampaa kuin koodivapaiden alustojen käyttö, joissa ideat ja työnkulut on koottu käyttövalmiiksi visuaalisiksi paketeiksi. Koodivapaiden alustojen käyttö mahdollistaan myös laajemmin yrityksen henkilöstön osallistumisen (kansalaiskehittäjänä) ohjelmistokehitysprosesseihin,

koska sen käyttö ei vaadi huipputason ohjelmointiosaamista. Ohjelmointiosaajia voidaan pyytää tarvittaessa tueksi, esim. mukautettaessa käsin kirjoitettua koodia alustalle, muuten heidät voidaan resursoida yrityksen sisällä syvällisempää osaamista vaativiin tehtäviin. (Bigelow 2023.)

Koodivapaan alustan käyttäminen on usein kustannustehokasta; sitä käyttämällä on mahdollista kokeilla matalalla kynnyksellä erilaisia ideoita, joiden toteuttaminen perinteisinä ohjelmistoprojekteina vaatisivat runsaasti aikaa ja taloudellisia resursseja. Usein koodivapaisiin alustoihin sisältyy jo oletuksena erilaisia analytiikkaan ja raportointiin liittyviä ominaisuuksia, joiden käyttäminen puolestaan auttaa keräämään tärkeitä tietoja projektin suorituskyvystä ja käytöstä, mikä voi edelleen auttaa tiimejä suunnittelemaan päivityksiä ja suorittamaan vianetsintää. (Bigelow 2023.)

Vaikka koodivapaa alusta menetelmänä vaikuttaa näennäisen yksinkertaiselta käyttää, tulee muistaa, ettei alusta tee mitään käyttäjän puolesta ja sen käyttäminen ohjelmistokehityksessä tulisi olla aina harkittua ja alustojen käytön tulisi olla suhteutettu yrityksen tarpeisiin. Koodivapaan alustan tehokkuus on sen valmiissa komponenteissa, jotka sopivat useisiin eri käyttötapauksiin. Komponenttien muokkaaminen ja lisääminen on usein mahdollista, mutta tällainen räätälöinti lisää väistämättä kehitystyötä ja kustannuksia, mikä puolestaan heikentää kehitysvapaan alustan tarkoitusta ja ominaisuutta olla nopea ja yksinkertainen. Vaarana on myös se, virhetilanteissa joudutaan kuluttamaan aikaa enemmän vianmääritykseen, kuin jos olisi luotu laadukasta koodia heti alusta alkaen. Käytettäessä koodivapaita alustoja tulee olla myös suunnitelma sen varalta, ettei sovellustyökalu tai sovellus toimi odotetulla tavalla. Jollakin henkilöllä yrityksessä tulisi olla tieto ja ymmärrys siitä, miten virhetilanteissa toimitaan ja miten virheet korjataan. (Bigelow 2023.)

Johannessen & Davenport muistuttavat, että vaikka koodivapaiden alustojen käytössä on suuria etuja, niiden laajaan käyttöön liittyy myös hallintahaasteita. Koodivapaiden alustojen ja sovellustyökalujen laaja käyttö voi vahvistaa "varjo-IT"-ilmiötä (engl. shadow IT). Kyseessä on ilmiö, jossa yrityksessä lähdetään hakemaan liiketoiminnan tarpeisiin nopeita ratkaisuja, joihin yrityksen IT:ltä ei löydy ratkaisua tai ratkaisun saaminen on tehty hankalaksi. Mitä enemmän varjo-IT:tä syntyy, sitä enemmän syntyy dataa eri paikkaan, mikä puolestaan lisää haittoja ja riskejä; tietoturvasta huolehtiminen hankaloituu ja tietoa häviää monimutkaiseen, järjestäytymättömään ympäristöön. Tällaisissa olosuhteissa työntekijät toimivat osittaisen tiedon varassa, mikä heikentää tehokkuutta ja vaikeuttaa työn tekemistä. (Johannessen & Davenport 2021.)

3.2 Koodivapaiden alustojen tulevaisuus

Lähimenneisyyttä tarkasteltaessa trendinä on ollut suurten yritysten kuten Amazonin, Microsoftin ja Googlen ilmaantuminen kilpailemaan pienempien koodivapaiden alustojen toimittajien, esimerkiksi OutSystemsin, Mendixin ja Appianin kanssa. Ennusteiden mukaan matalakooditeknologian myynti

kasvaa vuoden 2019 9,1 miljardista dollarista vuoteen 2025 mennessä 29 miljardiin dollariin. Kasvua vauhdittaa paitsi pandemia ja sen myötä muuttunut työkulttuuri, automaation lisääntyminen, ja joustavampien liiketoimintastrategioiden lisääntyminen. Uusia palveluntarjoajia tulee markkinoille koko ajan vastaamaan kasvavaan kysyntään. (Bigelow 2023.)

Ohjelmistorobotiikkaa (engl. Robotic Process Automation, RPA) pidetään yhtenä nopeimmin kasvavista koodivapaiden alustojen hyötyjistä. Ohjelmistorobotiikka antaa käyttäjille mahdollisuuden suunnitella automatisoituja työnkulkuja, jotka ovat ominaisuuksiltaan usein toistuvia, sääntöihin pohjautuvia ison volyymin tehtäviä ja ne voivat ulottua useisiin tietojärjestelmiin. (Johannessen & Davenport 2021.)

Koodivapailla menetelmillä luotuja ohjelmia käytetään verkko- ja mobiilisivustojen kehittämiseen jo nyt laajalti ja tällaisten ohjelmien kehittyneimmät versiot voivat jopa käsitellä asiakastapahtumia. Verkkosivustojen suunnittelutyökaluja tarjoavat yritykset tarjoavat usein myös isännöintipalveluita, ja ne voivat myös tarjota lisäarvoa tuottavia koodivapaita ominaisuuksia, jotka auttavat hakukoneoptimoinnissa ja sosiaalisen median markkinoinnissa, sekä mahdollistavat digitaalisen analytiikan määrittämisen ja hallinnan. Jotkut koodivapaat sovellustyökalut helpottavat jo nyt yrityksiä automatisoimaan markkinointia, kuten verkkosivustojen personointia, sähköpostimarkkinointia ja digitaalista mainosliikennettä. On todennäköistä, että koodivapaat alustat tarjoavat kehittyessään mahdollisuuksia verkko- ja mobiilisivustojen käyttöön koko ajan kattavammin. (Johannessen & Davenport 2021.)

3.3 Microsoft Power Apps

Power Apps valikoitui opinnäytetyöni tuloksena syntyneen sovelluksen koodivapaaksi alustaksi pitkälti siitä syystä, että sen käyttämiseen on saatavilla runsaasti tukimateriaalia ja ohjeita. Lisäksi sen käyttäminen oli maksutonta ja aiempi käyttökokemukseni Microsoftin ohjelmista tuki uuden alustan ominaisuuksiin tutustumista ja käyttöönottoa.

Power Apps on Microsoftin koodivapaa sovellustyökalu, joka toimii Microsoftin pilvipohjaisella kehitysalustalla, Power Platformilla. Power Platform tarjoaa käyttäjälleen kehitysalustan, jossa sovellusten toteuttaminen Power Appsilla ei välttämättä vaadi aiempaa koodausosaamista. Sovelluksia voi tehdä valmiiden mallipohjien avulla tai tyhjältä työpöydältä. Power Appsin sovelluslogiikkaa voi rikastaa käyttämällä Power Appsin kaavakieltä, joka muistuttaa Microsoftin taulukkolaskentaohjelma Excelissä käytettyä kaavakieltä. Sovellukset muodostavat yhteyden joko taustalla olevaan tietovarastoon, erilaisiin verkkolähteisiin tai paikallisiin lähteisiin tallennettuihin tietoihin. (Leung 2021, luku ”Introducing Power Apps”.)

Power Apps on suunniteltu kaikille avoimeksi sovellustyökaluksi ja Power Appsia käyttäviä sovelluskehittäjiä nimitetään Microsoftin omissa lähteissä usein kansalaiskehittäjiksi (engl. citizen developers) ikään kuin korostamaan sitä, että Power Appsin avulla voidaan suunnitella sovelluksia, joilla vastata liiketoiminnan haasteisiin ja vaatimuksiin nopeasti ja ketterästi kehittäjän lähtötasosta riippumatta. (Church 2023.)

Power Apps -sovellukset luodaan koodivapaiden alustojen tapaan käyttämällä valmiita elementtejä ja lisäämällä tarvittaessa erilaisia toimintoja suorittavia kaavoja (engl. functions), jotka noudattavat muodoltaan Microsoft Excelin kaavoja. Sovellusta voi tarkastella esikatselutoiminnolla jokaisen muutoksen jälkeen ja saada reaaliaikaisen käsityksen sovelluksen toiminnasta. Power Apps tarjoaa automaattisesti myös erilaisia väriteemoja, joissa on etukäteen huomioitu värien yhteensopivuus ja kontrasti sekä visuaalinen käyttäjäystävällisyys.

Power Apps tarjoaa tuen useille kielille ja sovelluksen käyttämä kieli määräytyy käytössä olevan verkkoselaimen perusteella. Selaimen kielivalinnalla on sovelluksen kehittämisen edetessä merkitystä Power Appsin käytössä. Jos käytössä on kieli, jossa käytetään pilkkua desimaalipisteen erottimena (esim. suomi, saksa tai espanja), tulee vastaavasti käyttää puolipistettä paikoissa, joissa yleensä käytetään pilkkua englannin kielessä kaavoja kirjoitettaessa. Kaavarivillä kielivalinta määrittelee puolipisteen käytön; lähes kaikissa eurooppalaisissa kielissä tulee puolipisteen sijaan käyttää kahta puolipistettä peräkkäin. (Leung 2022, luku ”Sharing Apps”.)

Sovelluksen kielen myötä määräytyvä kielituki aiheuttaa myös omat haasteensa työkalun käyttämiselle; vaikka valtaosa kielestä kääntyy sovelluksessa suomeksi, osa jää myös englanniksi. Power Appsin toimintalogiikka aukeaa varsin helposti kahden kielen sekamelskasta huolimatta, mikäli Microsoftin ohjelmistot ovat käyttäjälle entuudestaan yhtään tuttuja.

Pohjaan perustuva sovellus

Pohjaan perustuva sovellus (engl. canvas) on Power Appsin käytettävissä olevista sovellustyypeistä suosituin ja se voi muodostaa yhteyden lukuisiin tietolähteisiin, esimerkiksi Exceliin, SharePointiin ja SQL-serveriin. Pohjaan perustuvat sovellukset ovat yhteensopivia iOS- ja Android-laitteiden kanssa ja ovat sekä toiminnallisuuksiltaan että ulkonäöltään pitkälle muokattavissa. Helpoimmillaan pohjaan perustuvan sovelluksen rakentamisen voi aloittaa jopa oman piirroksen pohjalta, kyseessä on siis hyvin matalan kynnyksen sovellustyökalu. (Leung 2021, luku ”Introducing Power Apps”.)

Mallipohjainen sovellus

Mallipohjaisten sovellusten (engl. model-driven) suunnittelukokemus Power Appsissa on hyvin erilainen kuin pohjaan perustuvien sovellusten luominen. Mallipohjaisessa sovelluksessa ei ole pohjaan perustuvasta sovelluksesta tuttuja valikoita tai painikkeita, eikä näyttöjen ulkoasun hienosäätöön ole samanlaista mahdollisuutta. Sen etuna on, että sovelluksen rakennuskokemus on yksinkertainen, mutta toisaalta se tarjoaa huomattavasti vähemmän mahdollisuutta vaikuttaa sovelluksen ulkoasuun. Mallipohjaiset sovellukset rakennetaan Dataverse-tietolähteessä määriteltyjen tietojen, suhteiden ja prosessien ympärille. (Leung 2021, luku ”Introducing Power Apps”.)

Mallipohjaisia sovelluksia voisi kutsua myös pohjaan perustuvien sovellusten datakeskeisiksi sisaruksiksi. Mallipohjaiset sovellukset tarjoavat monia ominaisuuksia, jotka keskittyvät tietopohjaisiin ratkaisuihin ja nämä ratkaisut käyttävät Dataversen ominaisuuksia. Power Appsissa on mahdollisuus käyttää myös ns. konvergerssimallia, jossa voi yhdistää pohjaan perustuvien sovelluksien ominaisuudet mallipohjaisiin sovelluksiin. (Eickhel 2021, luku” Building from Data with Model-Driven Apps”.)

Pohjaan perustuvia ja mallipohjaisia sovelluksia luodaan, päivitetään, testataan ja otetaan käyttöön Power Apps Studiossa. Power Apps Studio on täysin selainpohjainen ympäristö ja kaikki nykyaikaiset selaimet tukevat sitä. Power Apps Studion käyttäminen mahdollistaa Power Appsin uusimpien päivitysten ja ominaisuuksien käyttämisen automaattisesti ilman erillistä päivitystarvetta. Se on korvannut edeltäjänsä, Power Apps for Windowsin, jota käytetään nykyään enää sovellusten kehittämisen sijaan lähinnä sovellusten katseluun. Power Apps Studio avautuu aina sovellusta luotaessa tai muokattaessa. (Valto 2023.)

Portaali

Portaalia (nyk. Power Pages) käytetään julkisten verkkosivustojen rakentamiseen ja se hakee tiedot Dataverse-tietolähteistä. Useimmiten verkkosivuston yhdistäminen tietokantaan vaatii kohtuullisen kokemuksen ohjelmoinnista, mutta portaalin käyttäminen ei edellytä tätä kokemusta. Portaalisovellusten rakennustyökalu on hyvin visuaalinen ja muistuttaa ominaisuuksiltaan niitä työkaluja, joita yleensä käytetään blogien tai verkkosivustojen rakentamiseen. (Leung 2021, luku ”Introducing Power Apps”.)

4 Sovelluksen luominen

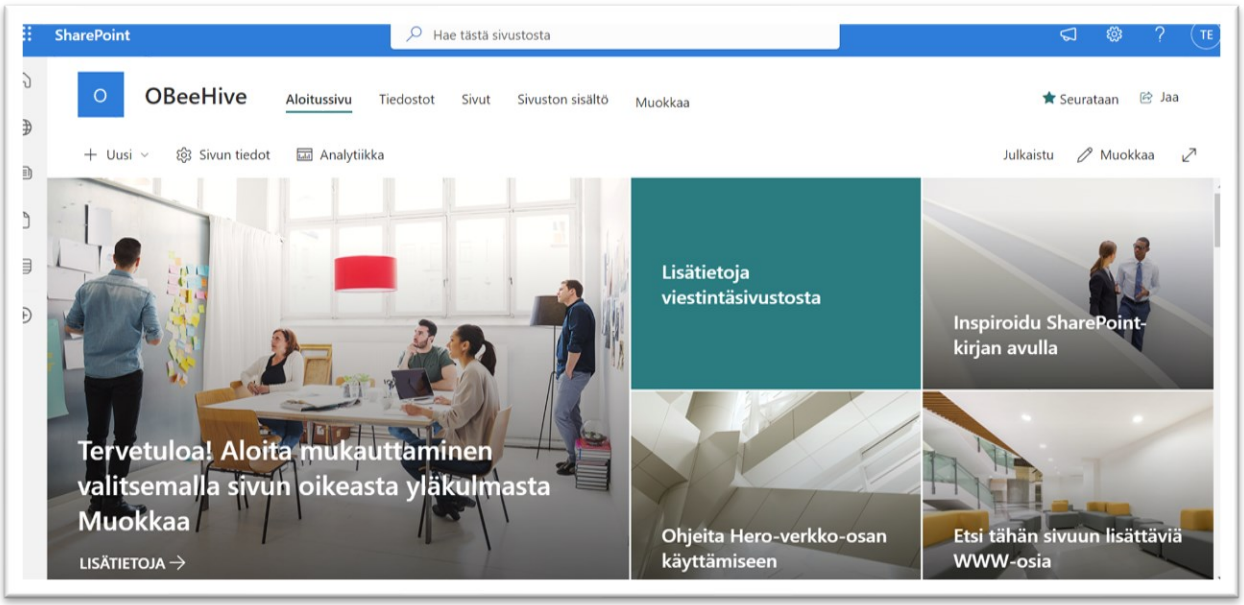
Valitsin opinnäytetyössäni käytettäväksi Power Appsin pohjaan perustuvan sovelluksen, koska se on responsiivinen, mobiiliyhteensopiva ja sen tietovarastona voi käyttää SharePointia. Pohjaan perustuva sovellus luo automaattisesti tietovarastoon luoduista luetteloista käsin näytöt, kun sovellusta avataan ensimmäisen kerran muokkaustilaan. Tämä ominaisuus on etenkin uusille sovellusten tekijöille suureksi avuksi ensimmäistä sovellusta luodessa. (Eickhel 2021, luku ”Improving User Experience”.)

Pohjaan perustuvan sovelluksen luominen Power Appsissa alkoi tietovaraston valitsemisella. Tietovaraston valitsemiseen vaikuttaa käytettävän pohjan lisäksi sovelluksen koko ja sen tarvitsema tallennustila.

4.1 SharePoint ja luettelot

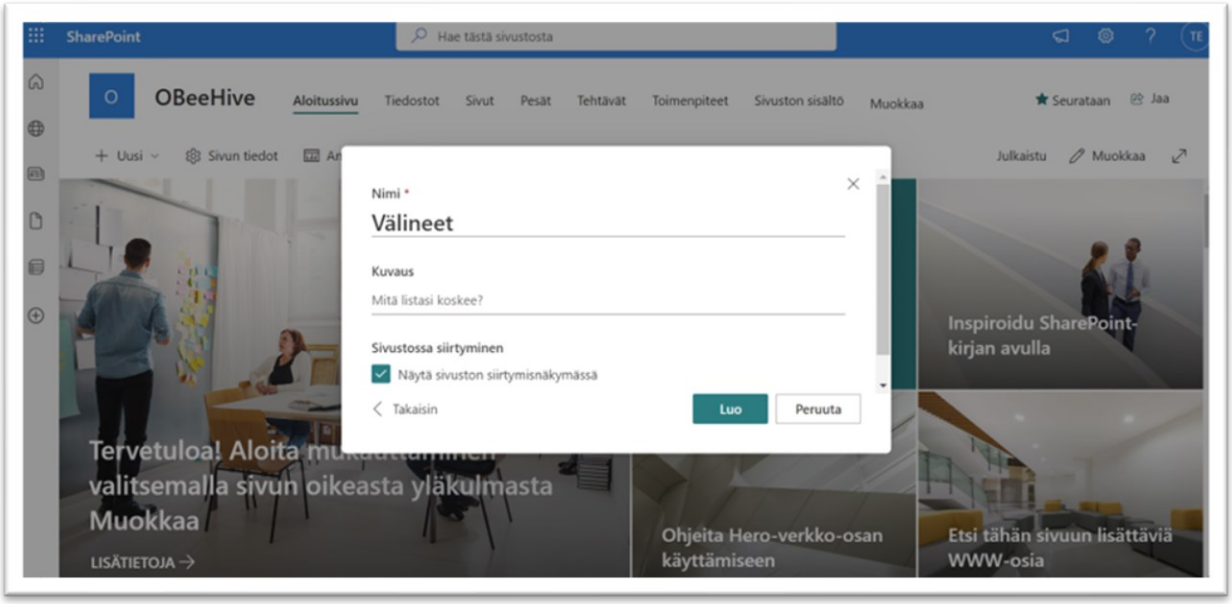
Sovelluksen tietojen tallennuspaikaksi valikoitui SharePoint, sillä se tarjosi sovellustani varten riittävät resurssit, oli maksuton käyttää ja oli minulle ympäristönä jo entuudestaan tuttu. SharePoint on osa Microsoftin 365 -palveluja ja tarjoaa suojatun ympäristön tiedostojen yhteiskäyttöön, tallentamiseen ja jakamiseen ja se on yhteensopiva kaikkien yleisimpien selainten kanssa.

Sovelluksen tietovaraston luominen alkoi SharePoint-viestintäsivuston tekemisellä SharePointin ylläpitoportaalin kautta. Viestintäsivuston kautta luodaan ja muokataan SharePoint-luetteloita. Nykyään SharePointissa käytetään moderneja sivustoja aiempien klassisten sivustojen sijaan.



Kuva 2. Sovellusta varten tehty moderni SharePoint-sivusto.

Sovelluksen tarvitsemaa tietovarastoa varten loin SharePoint-sivustolle neljä luetteloa (engl. list). Luettelot nimesin kuvaavasti helpottamaan jatkotyöskentelyä sovelluksen parissa. Luetteloiden nimeksi tuli pesät, toimenpiteet, tehtävät ja välineet.



Kuva 3. Välineet-luettelon luominen SharePoint-sivustolle.

Luetteloiden luomisen jälkeen lisäsin luettelokohtaisesti tarvittavat sarakkeet. Tässä työvaiheessa nimesin sarakkeet ja valitsin sarakkeiden tietotyypit. Sarakkeiden tietotyypiksi voi valita esimerkiksi tekstin, valintalistan tai päivämäärän. Loin jokaiseen luetteloon malliksi kohteita, esimerkiksi välineet-luetteloon listasin muutamia tarhaajien tärkeimpiä ja välttämättömimpiä työvälineitä (mehiläispuku, hanskat, harja, taltta). Etenin samaan tyyliin jokaisen luettelon kanssa lisäten niihin tarvittavat sarakkeet sopivilla tietotyypeillä.

Toimenpiteet ☆

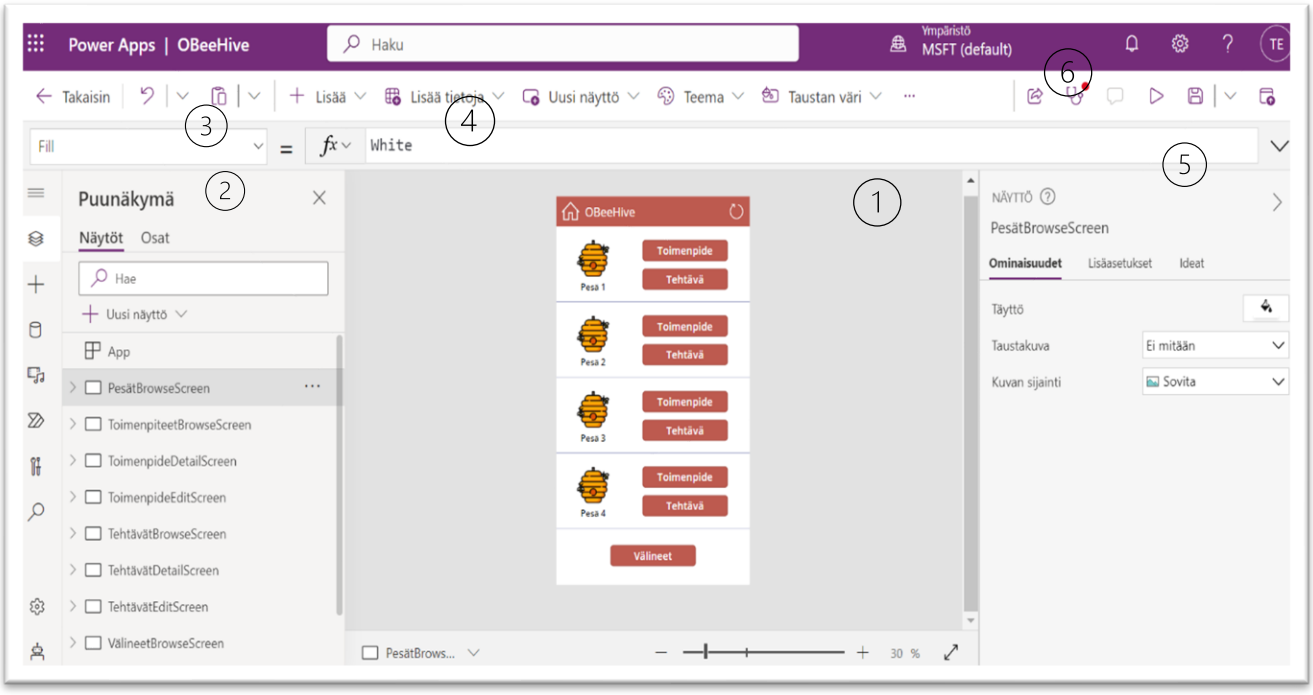
Otsikko	Pesä	Laatikot	Emo	Munintaa	Sikiökennoja	Näyte
Tarkastuskäynti	Pesä 1	1	✓	✓		13.3.2023
Tarkastuskäynti	Pesä 2	6	✓	✓	✓	13.3.2023
Tarkastuskäynti	Pesä 1	6				13.3.2023
Tarkastuskäynti	Pesä 1	4	✓	✓	✓	21.3.2023

Kuva 4. Esimerkki sovelluksen Toimenpiteet-luettelosta ja mallikohteista.

4.2 Puunäkymä ja näytöt (screens)

SharePoint-luetteloiden tekemisen jälkeen lähdin luomaan varsinaista sovellusta Power Appsilla. Loin uuden pohjaan perustuvan sovelluksen ja yhdistin sen aikaisemmin SharePointissa tekemääni luetteloon, minkä jälkeen pohja avautui automaattisesti varsinaiseen työskentelytilaan, Power Apps Studioon. Power Apps Studio on työtilana hyvin saman kaltainen kuin muutkin Microsoft 365 -ohjelmat. Sovelluksen luomista Power Appsilla verrataan usein PowerPoint-esitelmän tekemiseen ja Power Apps Studio on käyttäjäkokemuksena kuin yhdistelmä PowerPointia ja Exceliä.

SharePoint – Kolmen näytön sovelluksen luonti -vaihtoehto teki valmiiksi kolme näyttöä tietojen esittämistä ja käsittelyä varten mobiilisovelluksessa. Sovellus näyttää tässä vaiheessa jo hyvin paljon muodollisesti valmiilta sovellukselta näyttöineen. Oletuksena on sinivalkoinen värimaailma ja Power Appsin automaattisesti luoma asettelu.



Kuva 5. Power Apps Studio.

1. Näyttö
2. Puunäkymä
3. Ominaisuusluettelo
4. Kaavarivi
5. Ominaisuusruutu
6. Sovellustoiminnot (jakaminen, vianmääritys, esikatselu, tallentaminen, julkaisu)

Power Apps Studiossa sovelluksen muokkaus tapahtuu pääasiallisesti kolmella eri näytöllä (engl. screen); browse-näytöllä, detail-näytöllä ja edit-näytöllä. Browse-näytöllä listataan kaikki luettelon kohteet, detail-näytöllä näytetään yksittäisen kohteen tiedot ja edit-näytöllä muokataan yksittäisen kohteen tietoja. Näytöt ja kohteet voi työskentelyään ja muistiaan helpottaakseen nimetä kuvaavammin. Power Apps muuttaa automaattisesti kaikki kaavaviittaukset käyttämään uutta nimeä, jolloin kehittäjän ei tarvitse manuaalisesti vaihtaa kaikkien muiden näyttöön tai kohteisiin viittaavien sovelluksen osia. (Leung 2021, luku ” Creating Your First App.)

Näytön vasempaan reunaan avautuu oletuksena puunäkymä, jonka kautta hallitaan sovelluksen elementtejä. Puunäkymässä näkyy kaikki sovellukseen lisätyt näytöt ja näyttöjen ominaisuudet. Puunäkymästä valittu ominaisuus näkyy näytöllä aktivoituna ja päinvastoin, näytöllä valittu ominaisuus näkyy puunäkymässä aktivoituna.

Ohjausobjekteiksi kutsutaan sovelluksen näytöille lisättyjä painikkeita, kuvia, tekstikenttiä, ikoneita ja taulukoita. Ohjausobjekteilla luodaan sovellukseen erilaisia toimintoja ja jokaisen ohjausobjektin ulkoasua sekä käytöstä voi muokata sen ominaisuutta säätämällä. Valittuja ohjausobjekteja voidaan muokata sivun oikealle reunalle avautuvassa ruudussa ja niille voidaan asettaa erilaisia ominaisuuksia puunäkymän yläpuolella olevasta alasvetovalikosta, jota kutsun ominaisuusluetteloksi.

Ylänavigaatiosta tai puunäkymän vasemmalta puolelta voi lisätä sovellukseen erilaisia ohjausobjekteja, kuten painikkeita ja näyttöjä tai valita (väri)teeman. Ylänavigaatiopalkin oikeassa reunassa on mahdollisuus sovellustoimintojen käyttämiseen, kuten jakamiseen, vianmääritykseen, esikatseluun ja tallentamiseen.

4.3 Funktiot

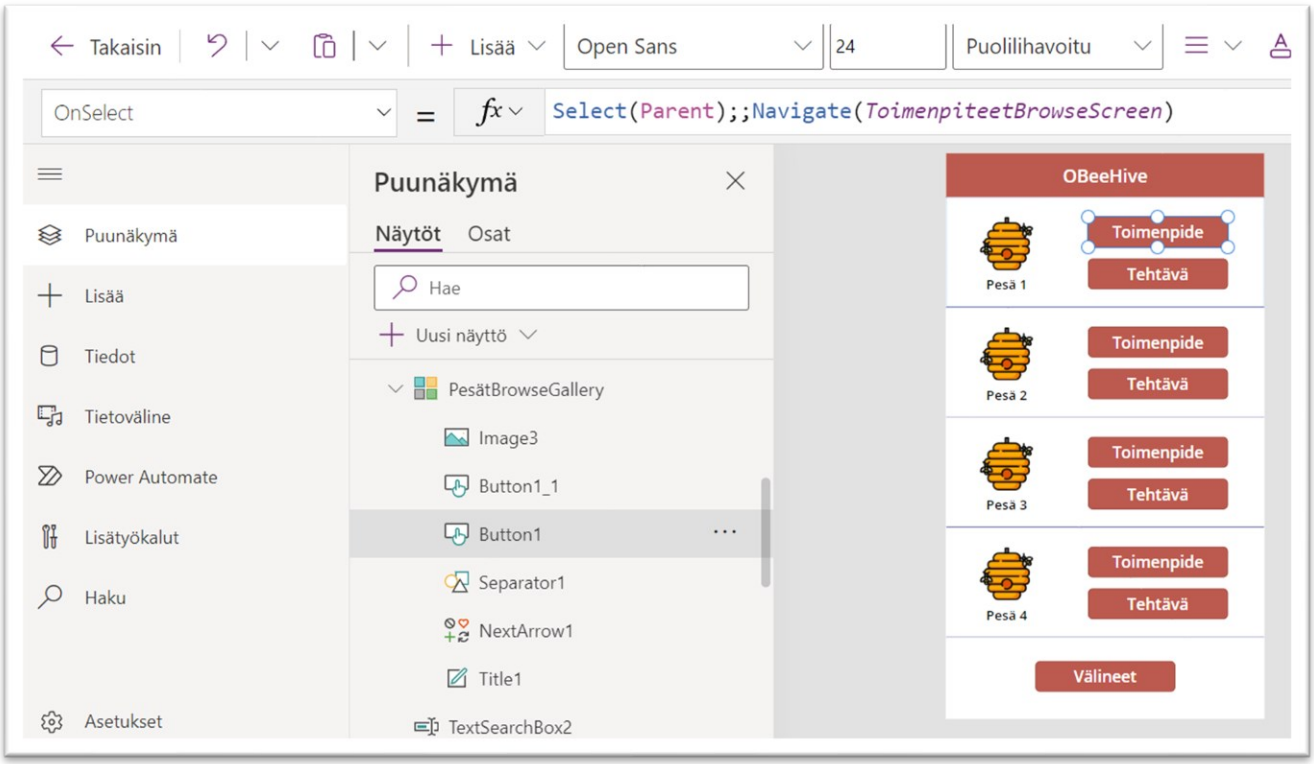
Näytön yläosassa, ylänavigaation alapuolella on Excelistäkin tuttu kaavarivi (engl. functions) ja kaavojen kirjoittaminen onkin käytännössä hyvin samanlaista kuin Excelissä. Kaavarivillä voi rakentaa ja yhdistellä funktioita sovelluksen toimintojen luomiseen; niitä käytetään mm. lisäämään sovellukseen laskelmia ja ohjaamaan ohjausobjektien toimintoja. Kaavoja voidaan käyttää missä tahansa sovelluksen ominaisuudessa arvojen asettamiseen, joko kutsumalla funktiota, dataa tietolähteestä tai olemalla vuorovaikutuksessa toisen näytön ohjausobjektin kanssa. (Weston, M. luku ”Exploring formulas”.)

Pitkien kaavojen lukemisen helpottamiseksi kaavariviä voi laajentaa käyttämällä hiirtä korkeuden säätämiseen. Tekstiä voi muotoilla helpottamaan kaavojen lukemista esim. lisäämällä rivinvaihtoja ja sisennyksiä. Kaavarivi sisältää myös koodin värityksen ja tarjoaa apua IntelliSensen kautta. IntelliSense tarkoittaa mahdollisuutta siirtyä lukemaan tarkemmin funktion toiminnasta Microsoftin Formula Reference for Power Apps -sivulle. Kaavarivillä voi käyttää yleisimpiä pikakomentoja, joilla

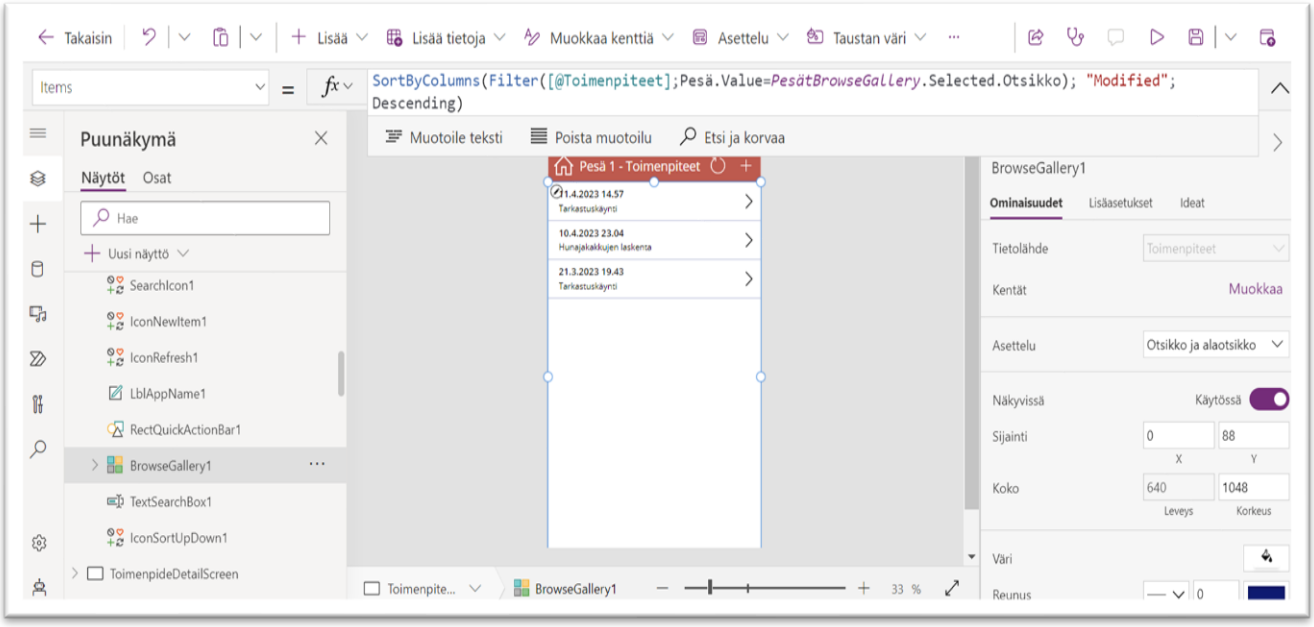
kopioida (Ctrl + C) ja liittää (Ctrl + V) kaavoja. Tämän lisäksi kaavarivi käyttää myös automaattista täydennystä tarjoten argumentit, joita funktio odottaa, ohjaten näin kehittäjää reaaliajassa. (Leung 2021, luku 5. ” Using Formulas”.)

Käytin sovelluksessani funktioita laajentamaan ohjausobjektien toiminnallisuuksia sekä tiettyjen oletusarvojen asettamiseen. Sovelluksen käynnistyksen yhteyteen on määritelty sovelluksessa käytettävät värit omiin muuttujiin. Muuttujiin viitataan eri ohjausobjektien asetuksissa. Näin värien muuttaminen onnistuu yhdestä paikasta niin, ettei värejä muutettaessa tarvitse käydä kaikkia ohjausobjekteja läpi.

Gallerioiden (gallery) pesäkohtaisessa filtteröinnissä käytin myös funktiota (kuva 6 ja kuva 7) filtteröimään pesäkohtaiset toimenpiteet ja tehtävät.

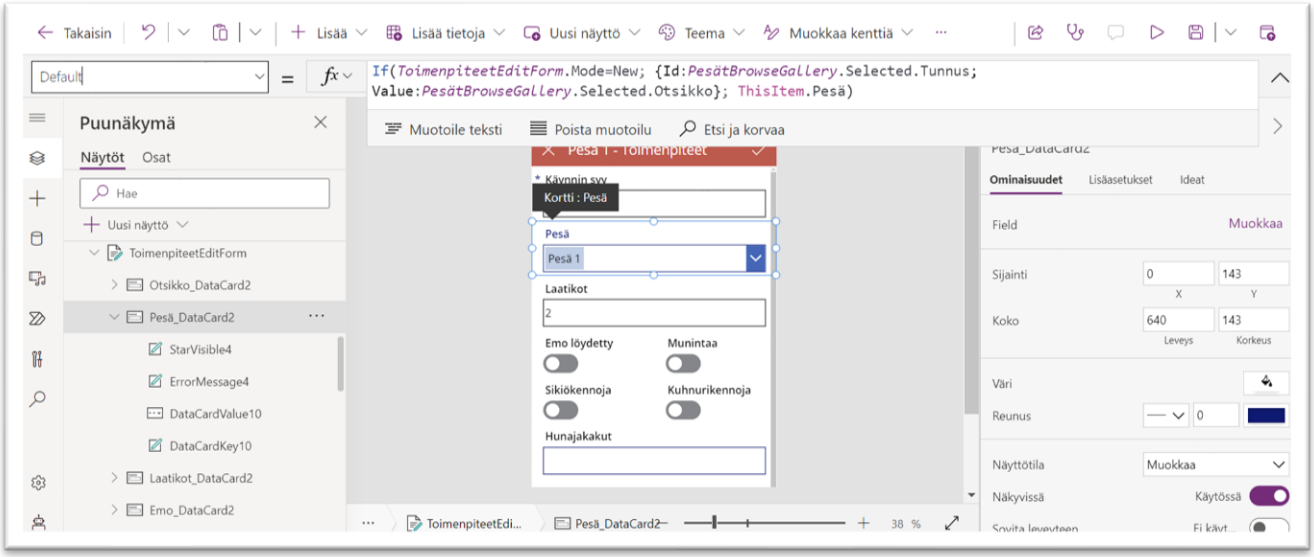


Kuva 6. Toimenpide-painikkeeseen on lisätty Select(Parent)-funktio, joka valitsee pesän. Tästä siirrytään pesän toimenpiteet-listaukseen.



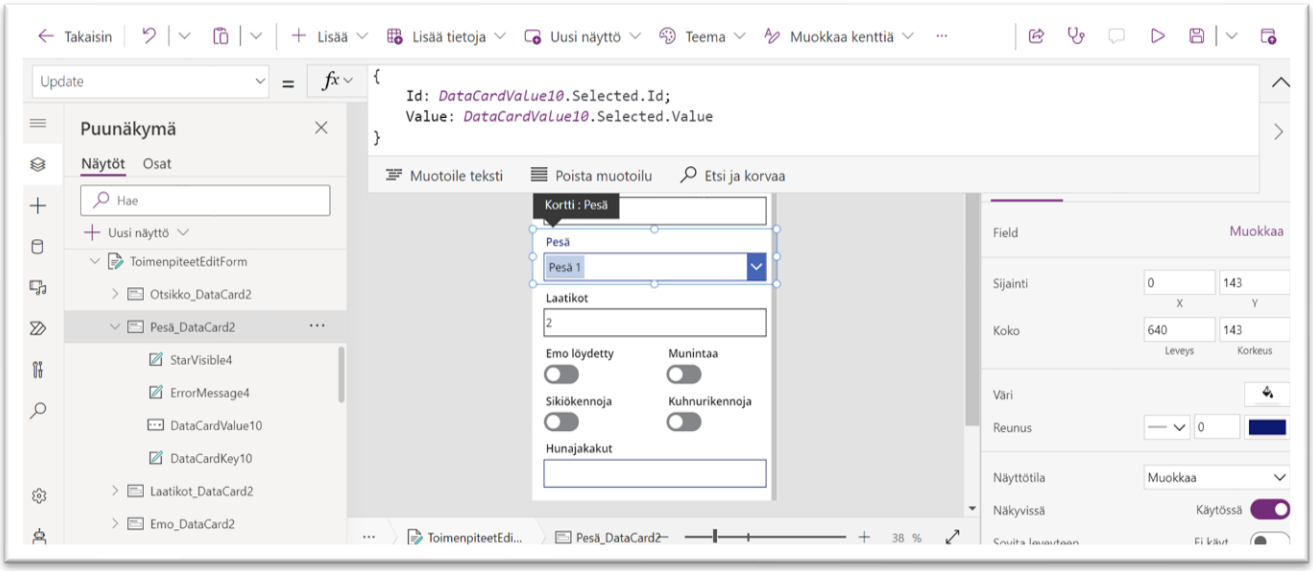
Kuva 7. Toimenpiteet-listausnäytön galleria filtteröidään valitun pesän mukaisesti. Toteutuneet toimenpiteet on järjestetty muokkauspäivämäärän mukaan laskevaan järjestykseen.

Käytin funktioita myös asettamaan oletuspesän uutta toimenpidettä luotaessa (kuva 8). Tällöin käyttäjän ei tarvitse enää aloitusnäytöltä pesän valittuaan toistaa valintaa toimenpiteen kirjauksen yhteydessä, vaan pesä on automaattisesti sama kuin minkä käyttäjä on ensimmäisenä valinnut. Tämä estää ns. ristiin kirjaamisen, jolloin käyttäjä olisi valinnut aloitusnäytöltä pesän 1, mutta epähuomiossa valitseekin esimerkiksi pesän 3 vielä toimenpidettä tallentaessaan.



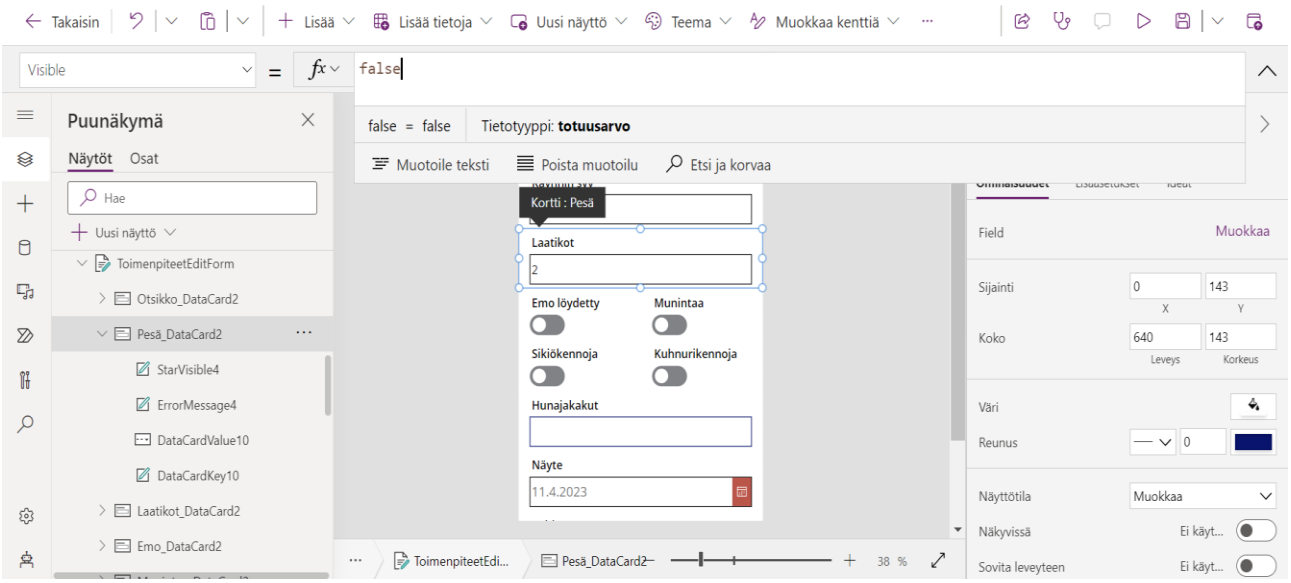
Kuva 8. Pesät-valintalistan oletusarvon asettaminen.

Oletuspesän asettamisen jälkeen tulee vielä muokata tiedon päivittämistä niin, että valitun pesän tunnus (id) ja arvo (value) menevät SharePoint-luetteloon oikein (kuva 9). Ellei tätä muokkausta tee, tallentuu tiedot osittain väärin pesätiedon jäädessä tyhjäksi. Tiedot siis tallentuvat, mutta pesätieto jää tyhjäksi.



Kuva 9. Pesän *id* ja *value* -tietoparin tallentaminen.

Tämän jälkeen pesän tieto piilotetaan Toimenpiteet -lomakkeelta (kuva 10), koska se on tarpeeton.

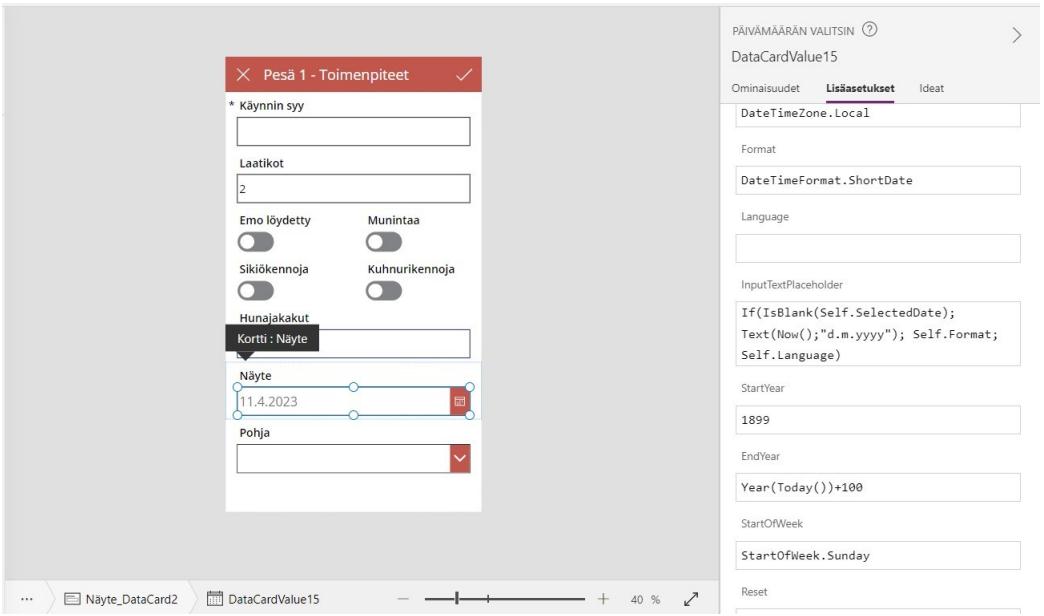


Kuva 10. Pesä-tietoa ei näy enää Toimenpiteet-lomakkeella.

Asetin sovellukseen oletuspäivämääräksi kuluvan päivän niissä tilanteissa, kun päivämäärän kirjaamiselle on tarve (kuva 11). Tehdyt toimenpiteet kirjautuvat automaattisesti reaaliajassa, mutta

esimerkiksi näytteenottopäivämäärä ja pesäkohtaisentehtävän tallennuspäivämäärä saattoivat olla jotain muuta kuin kuluva päivä.

Oletuksena Power Appsissa oli funktio *If(IsBlank(Self.SelectedDate); Text(Date(2001;12;31); Self.Format; Self.Language))*, jonka korvasin funktiolla *If(IsBlank(Self.SelectedDate); Text(Now();"yyyy;mm;dd"); Self.Format; Self.Language)*

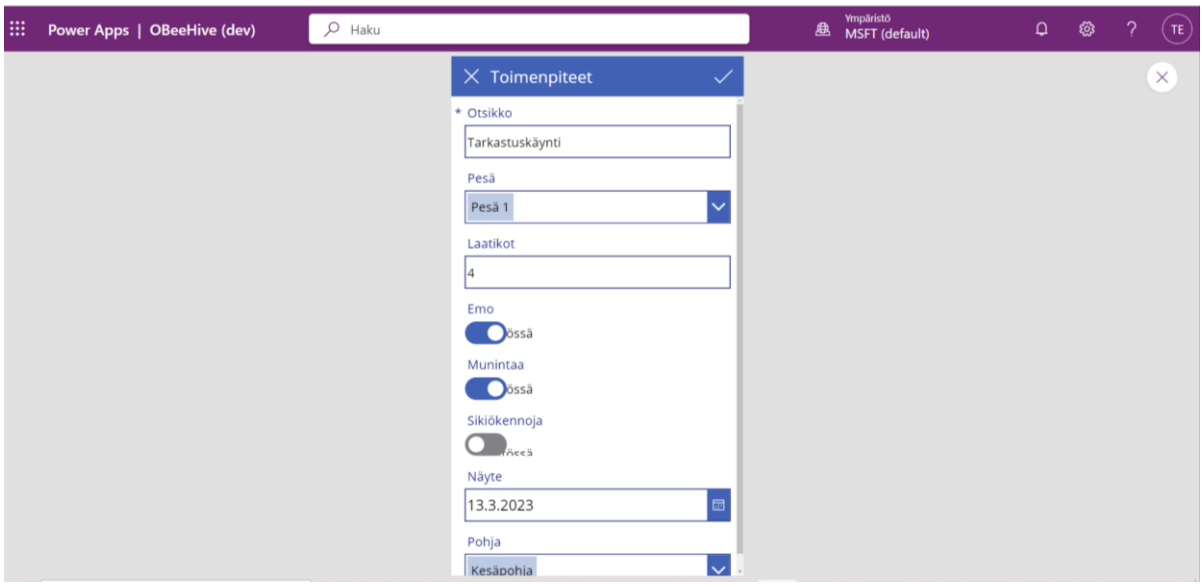


Kuva 11. Päivämäärän muuttaminen lomakkeelle.

4.4 Esikatselu

Sovelluksen ajaminen ja testaaminen Power Appsissa tapahtuu niin kutsutussa esikatselutilassa. Sovellusta kehittäessäni tallensin tekemäni muutokset usein ja käytin tallennuksen jälkeen esikatselutoimintoa, ensisijaisesti siksi, että varmistuin sovelluksen toimivuudesta ja pääsin nopeasti kiinni virheisiin, mutta myös siksi, että tekemiseni pohjautui myös hyvin pitkälti kokeilemiseen ja virheistä oppimiseen.

Esimerkkinä alla olevassa kuvassa sovelluksen luomisen alkuvaiheessa (kuva 12) näkyy, kuinka esikatselutilassa kyllä/ei -valinta näyttää suomenkielisessä versiossa melko huonolta, koska valintapainikkeen oletusteksti tulee näkyviin ikävästi valintapainikkeen alta. Valintapainikkeen oletuskuvailu on ehkä hieman muodollisesti ”käytössä” ja ”ei-käytössä”. Tämän esimerkkitilanteen korjasin poistamalla painikkeen otsikkotiedon kokonaan ja vaihtamalla omaa logiikkaani mukaillen oletustekstit muotoon ”kyllä” ja ”ei” kuvaamaan paremmin painikkeen toimintaa.



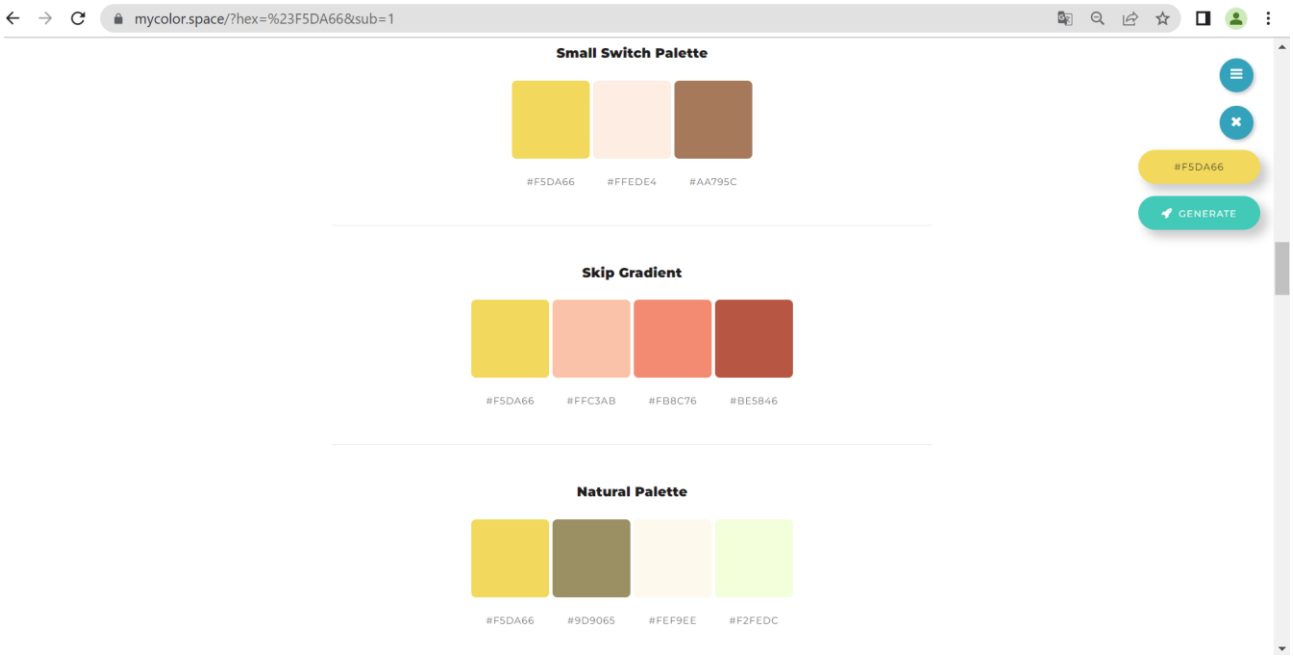
Kuva 12. Esikatselutila.

4.5 Visuaalinen ilme

Sovelluksen ulkoasuun ja värien valintaan vaikutti aikaisemmin sovellusprojektin aikana tekemäni käyttöliittymäsuunnittelu, mikä sisälsi saavutettavuusnäkökulman. Sovelluksen värien suunnittelu oli ennen käyttöönottoa viimeisimpiä viilauksia. Jokinen (2022) neuvoo välttämään voimakkaita, kylläisiä värejä ja pitämään väripaletin hallittuna; värien tulisi olla loogisesti samat läpi sovelluksen ja huomioimaan niiden semantiikka. Semantiikka värimäärittelyssä tarkoittaa esimerkiksi painikkeiden väreissä sitä, että tiettyjä värejä on yleisesti totuttu pitämään joko varoituksina tai ohjaamaan toimintaa. Punainen valintapainike mielletään usein kielteisen päätöksen tekeväksi, vihreä antaa suostumuksen ja keltaisella pohjalla olevaa tekstiä pidetään varoituksena.

Käytin sovellukseni värimaailman sommitteluun maksutonta mycolor.space -sivustoa (kuva 13), jossa voi valita haluamansa värin, jonka ympärille ohjelma luo sopivan väripaletin. Valitsin ns. pääväriksi tiilenpunaisen sävyn, tarhaajilla ei ollut antaa mitään ehtoja tai toiveita värivalinnoille. Niukimmillaan palettiin tulee kolme ja enimmillään kuusi toisiinsa sopivaa väriä.

Hyvin valitut värit on visuaalisesti miellyttävästi yhdistelty ja sovelluksen sisältämä teksti nousee esiin ollen selkeästi luettavissa. Omaan sovellukseeni koin kolmen värin yhdistelmän olevan riittävä, jopa liikaa. Sovellukseni on verrattain yksinkertainen, joten koin vielä parhaaksi rajata värit kahteen ja käyttää tekstiin omaa väriään, jottei yleisilme mene liian kirjavaksi ja luettavuus on kohdillaan. Sovellusta käytetään todennäköisesti ulkona ja eniten kesäaikaan, joten taustan ja tekstien välisen kontrastin tulee olla suuri.



Kuva 13. Esimerkki värien määrittelystä mycolor.space -sivustolla.

Painikkeiden kanssa pyrin huomioimaan saman loogisuuden kuin värimaailman kanssa. Jokaisella sovelluksen näytöllä huomioin painikkeiden paikat niin, että ne olivat johdonmukaisesti samoilla paikoilla ja kyllä/ei -painikkeissa väri muuttui selvästi valintaa tehdessä. Tällä tarkoitetaan sitä, että painikkeet, joilla vahvistetaan tai peruutetaan toimintoja, ovat joka näytöllä samassa järjestyksessä. Pyrin myös välttämään sovelluksen näytöillä tarpeen vierittää sivua, eli sijoittelemaan tekstit ja toiminnot niin, ettei käyttäjän tarvitse käyttää vierityspalkkia (engl. scrollbar), jonka käyttäminen vaatii tarkkuutta ja on sovellukseni käyttäjien tarpeet huomioiden melko haastava ominaisuus.

4.6 Julkaiseminen, laadunhallinta ja testaaminen

Sovelluksen tuotantoversio julkaistaan Power Apps Studion kautta Publish-toiminnolla. Julkaisun yhteydessä käyttäjille annetaan tarvittavat käyttöoikeudet sovellukseen ja SharePoint -tietovarastoon. Käyttäjät voivat käyttää sovellusta mobiililaitteella joko mobiililaitteen selainversion kautta tai sovelluskaupasta ladattavan Power Apps -sovelluksen avulla.

Ohjelmistojen laadunhallinnan (engl. quality assurance, QA) yhteydessä käytetään käsitteitä verifiointi ja validointi. Verifiointi tarkoittaa testaamista, jossa varmistetaan, että ohjelmisto täyttää sille vaatimusmäärittelyssä asetetut toiminnalliset ja ei-toiminnalliset vaatimukset vastaten kysymykseen *are we building the product right?* eli tehdäänkö tuotetta oikein. Validoinnilla varmistetaan, että ohjelmisto vastaa käyttäjän odotuksiin ja tarpeisiin ja sen pitäisi vastata kysymykseen *are we building the right product?* eli ollaanko tekemässä käyttötarkoitukseen soveltuvaa järjestelmää. (Luukkainen & Ilves, 2022.)

Verifioinnin ja validoinnin tärkein tavoite on varmistaa, että rakennettu ohjelma on tarpeeksi hyvä siihen käyttötarkoitukseen, mihin se on tarkoitettu. Riittävän hyvän määrittely on subjektiivista, eikä ohjelman tarvitse olla täydellinen ollakseen kuitenkin riittävän hyvä käyttötarkoitukseensa. (Luukkainen & Ilves, 2022.)

Power Appsilla rakentamani sovelluksen testaamisessa voi käyttää Sandbox-ympäristöön automaattisesti valmiiksi luotuja Azure AD -käyttäjätunnuksia tai luoda testaajille omat tunnukset. Microsoft 365 E5 Developer -lisenssejä on valmiina 25 kpl, joita voi hyödyntää testikäyttäjien tarpeisiin. Testikäyttäjät pitää luvittaa luku- ja kirjoitusoikeuksin SharePoint-tietovarastoon, sekä lukuoikeuksin (käyttäjä) sovellukseen.

Olin testannut sovellusta koko sen luomisen ajan, mikä paljasti välittömästi pieniä virheitä ja epäloogisuuksia. Loppukäyttäjät testasivat maaliskuussa 2023 sovellusta julkaistulla versiolla valmiiksi luoduilla tunnuksilla ja sovellusta korjattiin saatujen havaintojen perusteella vain hyvin vähän. Korjauksia tuli vain sovelluksessa käytettyihin otsikoihin sekä pesäkohtaisesti tallennettaviin tietoihin. Sovellukseen lisättiin mm. pesien lentoaukkojen rajoittamista koskeva valinta. Sovelluksen valmistuminen ajoittui sellaiseen vuodenaikaan ja harmillisesti myös sellaisten sääolosuhteiden aikaan, että sen testaaminen perustui lähinnä vain oletettuihin tilanteisiin ja skenaarioihin.

Pesiä ei ole suositeltavaa avata kovan pakkasen, sateen, navakan tuulen tai muuten haasteellisten sääolosuhteiden aikana. Maaliskuussa 2023 ennen sovelluksen testaamista mehiläiset olivat saaneet jo lisäruokinnan, joka on kevättalven ainoita tehtäviä mehiläistenhoidossa ennen mehiläisten talven jälkeistä puhdistuslentoa ja pesän heräämistä kevääseen. Sovelluksen valmistuessa mehiläiset eivät olleet vielä tehneet kevään ensimmäistä puhdistuslentoaan pesien ulkopuolella tai purkautuneet talvipallomuodostelmastaan. Verifiointi ja validointi tapahtuivat siis niiltä osin kuin se oli mahdollista.

Huhtikuussa 2023 sovellusta päästiin käyttämään sen verran, että kaikkiin pesiin voitiin päivittää tieto talvipohjan vaihtamisesta kesäpohjaan ja välineluetteloon lisättiin tieto talvipohjien sijainnista (navetan ylinen) ja kunnosta (pesua vailla). Lisäksi voitiin lisätä jokaiseen pesään maininta lentoaukon rajoittamisesta. Pesien lentoaukkoa rajoitetaan epävakaiden sääolosuhteiden ja pesien lämmönsäätelyn takia; täysin avoin lentoaukko lisää pesän tuulettumista ja aiheuttaa mehiläisille tarpeetonta työtä pitää lämpötila tarpeeksi korkealla.

5 Pohdinta

Opinnäytetyön tuotoksena syntyi rajoitetun mehiläistarhaajajoukon käyttöön ominaisuuksiltaan heidän toiveisiinsa pohjautuva sovellus. Sovellus vastaa valmistuttuaan tarhaajien tarpeisiin dokumentoida pesissä tehtyjä toimenpiteitä, luoda ja muokata tehtäviä ja listata tarhaajien varusteet, niiden kunnon ja säilytyspaikat. Sovellus luotiin Microsoftin Power Appsilla ja se rakennettiin pohjaan perustuvan mallin päälle. Sovellusta tulee käyttämään aktiivisesti ainakin kolme tarhaajaa omilla tunnuksillaan. Opinnäytetyön valmistuessa ei ole vielä tietoa vuoden 2023 toteutuvista mehiläistenhoitokursseista ja osallistujamääristä, kurssien toteutuessa käyttäjämäärä tulee todennäköisesti kasvamaan.

Suurimmat haasteet opinnäytetyöni tekemisessä liittyivät yllättäviin aikataulumuutoksiin, joihin pystyin varautumaan etukäteen oikeastaan vain asennoitumalla niin, että aikataulu tulee todennäköisesti elämään hyvästä suunnittelusta huolimatta. Kirin aikatauluani kiinni jakamalla työmäärää opinnäytetyön edetessä uudestaan ja omiin aikatauluihini sopivaksi.

Työvaiheiden avaaminen sanalliseen vaiheistukseen aiheutti käännösongelmia opinnäytetyöprosessin aikana. Suurin osa lähdemateriaalista oli englanninkielistä ja osalle englanninkielisiä termejä löytyi suomen kieleen vakiintuneita käännöksiä. Oli myös käsitteitä, joille ei löytynyt vakiintuneita suomenkielisiä käännöksiä ja usein termeistä käytettiinkin niiden englanninkielisiä nimiä myös suomenkielisissä lähteissä.

Microsoftin suomen kielelle kääntämät ohjeet ovat käännösohjelmien tekemiä ja ohjeissa käytetään paikoitellen hyvin suoraan käännettyjä termejä, joita ei käytetä missään muualla kuin kyseisissä ohjeissa. Tähän opinnäytetyöhön määrittelin käyttämäni termistön perustuen lähteissäni toistuvasti vastaan tulleisiin käännöksiin, aikaisempiin aiheesta julkaistuihin opinnäytetöihin sekä osittain suomen kieleen vakiintuneisiin käsitteisiin. Itse sovelluksen luomiseen sekakielisyydellä ei varsinaisesti ollut vaikutusta, koska tekeminen perustuu pitkälti visuaalisuuteen ja intuitioon. Pyrin ottamaan kielikompuroinnin huomioon niin, että käytin vakiintuneita, suomenkielisiä termejä aina kun mahdollista ja kerroin sulkeissa termin alkuperäisen, englanninkielisen nimen.

Opinnäytetyö eteni aluksi teoriavetoisesti teoriaa ja lähteitä kartoittaen, mutta pian vaatimusmäärittelyn jälkeen alkoi sovelluksen luominen. Työskentelyni sovelluksen kanssa perustui pitkälti kokeilemiseen, testaamiseen ja tarvittaessa ohjeiden etsimiseen. Käyttökokemus Power Apps Studiosta oli, että sen käyttäminen on M365 -ohjelmien käyttäneenä melko tuttu ja intuitiivinen. Muutokset pystyi helposti perumaan ja etenkin testaamisen helppous joudutti työn etenemistä. Oma lähestymistapani ei ollut ehkä kaikista nopein tapa toimia, mutta suhtauduin työskentelyyni nimenomaan kokeilemisen ja sitä kautta oppimisen kautta.

Lähteiden kanssa hyödynsin kirjoja työn alkuvaiheessa enemmän hahmottaakseni Power Appsin toimintalogiikkaa ja työvaiheiden järjestystä, sillä en ollut aikaisemmin esim. luonut tietovarastoa ja se oli olennainen osa sovellukseni kehitystä. Kun sain perusteet haltuun siinä määrin, että sovelluksen luomislogiikka hahmottui, käytin enemmän blogeja ja artikkeleita, sillä koodivapaisiin alustoihin liittyvän tiedon määrä on valtava ja ajankohtaista tukea oli saatavilla artikkeleiden ja etenkin YouTubeen ladattujen ohjevideoiden muodossa. Lisäksi painetulla kirjallisuudella on vaikeuksia pysyä Microsoftin päivitysten perässä, sen verran tiuhaan esim. lisenssien sisältöjä päivitetään ja julkaistaan ja ohjelmistoihin tulee nimitysmuutoksia tai kokonaan uusia tuotekokonaisuuksia.

Raportin kirjoittaminen ja sovelluksen luominen kulkivat rinnakkain, kunnes sovellus tuli valmiiksi ensin kirjoittamisen vielä jatkuessa. Työaika jakautui melko tasaisesti kirjoittamisen ja sovelluksen luomisen välille ja tällainen rytmitys ja ajan jakautuminen toimi mielestäni hyvin.

Opinnäytetyöprosessi oli itsessään omien oppimistapojen ja ajankäytön suunnittelun kannalta antoisa. Käsitykset omista vahvuuksista oppijana saivat vahvistusta ja toivat esiin myös tyypillisiä ajankäyttöön liittyviä sudenkuoppia, joiden työstäminen jatkuu opinnäytetyön jälkeenkin.

Olen tyytyväinen valitsemaani sovelluksen toteutustapaan, sillä koodivapaiden alustojen käyttö tulee hyvin todennäköisesti tulevaisuudessa lisääntymään ja koskettamaan laajempaa käyttäjäkuntaa kuin tällä hetkellä osataan ennustaa. Koodivapailla alustoilla tehtyjä sovelluksia on tullut vastaan opinnäytetyön aikana sellaisissa tilanteissa, joissa en aikaisemmin kiinnittänyt mitään huomiota sovelluksen tai toiminnon toteutustapaan.

Asetin itselleni opinnäytetyön alkaessa tavoitteeksi oppia käyttämään Power Appsia. Tätä tavoitetta on vaikea arvioida subjektiivisesti ja tavoitteena se on melko epätarkka ja aivan liian laaja, kuten tässä vaiheessa työn jo valmistuttua huomaan. Käytännössä harjoittelin ja opin soveltamaan ja skaalaamaan tarjolla olevaa tietoa pohjaan perustuvan sovelluksen luomisessa nimenomaan omassa, rajatussa opinnäytetyössäni. Opinnäytetyöprosessin edetessä ymmärsin kuinka valtavan paljon erilaisia toimintoja ja mahdollisuuksia Power Apps tarjoaa ja kuinka murto-osan siitä olen omassa työssäni pystynyt tai ehtinyt hyödyntämään.

Mielestäni opin enemmänkin hakemaan tietoa ja ratkomaan sellaisia ongelmia, joita sovelluksen luomisessa tuli vastaan. Opin olennaisimmat käsitteet ja toimintalogiikan, joten jatkossa todennäköisesti osaisin toimia sellaisessa sovelluskehitysprojektissa, jossa käytettäisiin Power Appsia sovellustyökaluna. Tässä opinnäytetyössä käytin vain rajattua osaa Power Appsista, koska keskityin tekemään sovelluksen pohjaan perustuvana ja käyttämään tietovarastona SharePointia. En lähtenyt hiomaan yksityiskohtia loputtomiin, vaan olin tyytyväinen sovellukseen, joka toimi

halutunlaisesti ollen kuitenkin valmiimpi ja kehittyneempi kuin MVP, *minimum viable product*, eli pienimmällä mahdollisella työllä saavutettu mutta tarpeeksi käyttötarkoituksessaan toimiva tuote.

Sovelluksen jatkokehittäminen tulee olemaan hyvin todennäköistä, sillä sovellus pääsee todelliseen käyttöön vasta kevään edetessä mehiläisyhdyskuntien järjestäytyessä täyteen toimintaan. Kesä on mehiläispesille aktiivisinta aikaa ja ne vaativat luonnollisesti myös tarhaajilta eniten resursseja touko-syyskuun-välisenä ajanjaksona. Aktiivisen käytön myötä sovelluksen kehitystarpeet tulevat näkyviksi ja toisaalta taas käyttäjien sille asettamat vaatimukset todennäköisesti lisääntyvät, jolloin kehittäminen perustuu todellisiin tarpeisiin.

Tarve tehtyjen toimenpiteiden selaamiseen tai kokoamiseen ratkeaa myös käytön jatkuessa. Nyt tehtyjä käyntejä voi tarkastella taaksepäin kronologisesti viimeisimmän käynnin tallentuessa ylimmäksi, käynnin syy -kenttään voi kirjoittaa kuvailevan tekstin, mikä auttaa selaamaan ja etsimään haluttua toimenpidettä. Mikäli mehiläistenhoito jatkuu säännöllisenä tulevina vuosina, voi tulla tarve esittää tietoa toteutuneista toimenpiteistä eri tavalla kuin pelkästään toteutuneiden käyntien otsikkotietoja selailemalla.

Mikäli molemmat kevään ja kesän 2023 mehiläistenhoitokurssit toteutuvat, tulee sovellukselle reilusti enemmän käyttäjiä ja tarve lisätä pesiä tai luoda kokonaan erilliset tarhat pesineen on todennäköistä. Tämän tyyppinen muutos on sovelluksessa helposti toteutettavissa lisäämällä aloitusnäytöksi pesien sijaan tarhat, luomalla tarhat-luettelon ja ohjaamalla edelleen navigaation tarhoista niille luotuihin pesiin. Jokaiselle käyttäjälle tulee niin ikään tehdä tai antaa omat käyttäjätunnukset sovellukseen. Sovelluksen käyttäjämäärän kasvaessa on myös järkevää pohtia sitä vaihtoehtoa, että se siirretään sopivaan Microsoftin tuotantoympäristöön Sandboxista.

Jatkokehityksessä mehiläispesät voisi sijoittaa karttapalveluun niin, että koordinaatit olisi myös sovelluksessa saatavilla pesän tiedoissa ja jaettavissa eteenpäin. Tällä hetkellä pesät on vain numeroitu, mutta mikäli pesien määrä lisääntyy ja niitä sijoitellaan selvästi eri alueille, on vain järkevää luoda omat tarhat ja niihin joko numeroidut tai esim. QR-koodatut pesät.

Pesät on ilmoitettu mehiläisrekisteriin EU:n eläinterveyssäännöstön (2016/429) mukaisesti, mutta mehiläisrekisterissä ne eivät vielä ole. Mehiläisrekisteri on avoin sekä maksuton rekisteri- ja karttapalvelu, johon on koottu keskitetysti mehiläispesät ja -tarhat. Rekisteröiminen näyttää mehiläisten lentoalueet ja auttaa tarhaajia uusien pesä- ja tarhapaikkojen valinnassa. Riittävän suuren lentoetäisyyden huomioimalla voi paitsi estää eri mehiläisrotujen risteytymisen ja mahdollistaa uusien kuningataremojen onnistuneet pariutumisen, myös rajata mahdollisesti parveilevat pesät ja saattaa ne takaisin tarhaajilleen. (SML ry, 2023.)

Lähteet

Aluehallintavirasto. 2023. Yleistä saavutettavuudesta. Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/#saavutettavuus-on-verkkomaailman-esteettomyytta. Luettu 22.3.2023

Aluehallintavirasto. 2020. Saavutettavat digipalvelut rakentavat yhdenvertaista Suomea. Luettavissa: https://www.saavutettavuusvaatimukset.fi/saavutettavat-digipalvelut-rakentavat-yhdenvertaista-suomea/. Luettu 2.4.2023

Benyon, D. 2019. E-kirja. Designing user experience: A guide to HCI, UX and interaction design. 4th edition. Harlow, England: Pearson Education Limited.

Bigelow, S. J. 2023. What is low-code? A guide to enterprise low-code app development. Luettavissa: https://www.techtarget.com/searchsoftwarequality/What-is-low-code-A-guide-to-enterprise-low-code-app-development. Luettu 6.4.2023

Church, N. 2023. 5 Real-life Microsoft PowerApps Examples. Blogi. Luettavissa: https://www.consultdolphin.com/blog/5-real-life-microsoft-powerapps-examples. Luettu 2.5.2023

Eickhel, M. 2021. E-kirja. Microsoft Power Apps Cookbook: Become a pro Power Apps maker by applying practical use cases to solve ever-evolving business challenges. Packt Publishing, Limited.

Hurja 8.9.2021. UX- ja UI-suunnittelu – mitä ne ovat ja mikä rooli niillä on verkkosivu- ja ohjelmistoprojektissa? Blogi. Luettavissa: https://www.hurja.fi/blogi/kayttoliittymasuunnittelu-vaatii-teknista-ja-visuaalista-osaamista/. Luettu 28.3.2023

Johannessen, C. & Davenport, T. 2021. When Low-Code/No-Code Development Works — and When It Doesn't. Luettavissa: When Low-Code/No-Code Development Works — and When It Doesn't (hbr.org) Luettu 12.4.2023

Jokinen, A. 2022. PowerUp! May Session Antti Jokinen: Power Appsien käyttöliittymäsuunnitelu. YouTube. Katsottavissa: https://youtu.be/Z8zLBI5CEj4. Katsottu 21.3.2023.

Leung, T. 2021. E-kirja. Beginning Power Apps: The Non-Developer's Guide to Building Business Applications. Berkeley, CA. Apress L. P.

Luukkainen, M. & Ilves, K. Ohjelmistotuotanto 2022. Helsingin yliopisto. Luettavissa: https://ohjelmistotuotanto-hy.github.io/. Luettu 6.3.2023.

Microsoft. 2023. Power Appsin kuvaus. Luettavissa: https://learn.microsoft.com/fi-fi/power-apps/powerapps-overview. Luettu 27.2.2023

Niemelä, H. Sovelluksen käytettävyys. 31.1.2020. @SeAMK. Verkkolehti. Luettavissa: https://lehti.seamk.fi/alykkaat-ja-energiatehokkaat-jarjestelmat/sovelluksen-kaytettavyys/. Luettu 28.3.2023.

Reilly, J. 2021. How No-Code Platforms Can Bring AI to Small and Midsize Businesses. Harvard Business Review. Luettavissa: How No-Code Platforms Can Bring AI to Small and Midsize Businesses (hbr.org). Luettu 19.4.2023.

Stellman, A. & Greene, J. 2015. E-kirja. Learning agile. Sebastopol, CA: O'Reilly.

Suomen Mehiläishoitajien liitto SML ry. 2023. Mehiläishoitajan velvollisuudet viranomaiselle. Luettavissa: https://hunaja.net/mehilaistarhaus/mehilaishoitajan-velvollisuudet-viranomaisille/. Luettu 2.4.2023

Valto 2023. PowerApps Studio Expert Support, Consultancy and Advice. Blogi. Luettavissa: https://valto.co.uk/microsoft-powerapps/studio/. Luettu 14.3.2023.

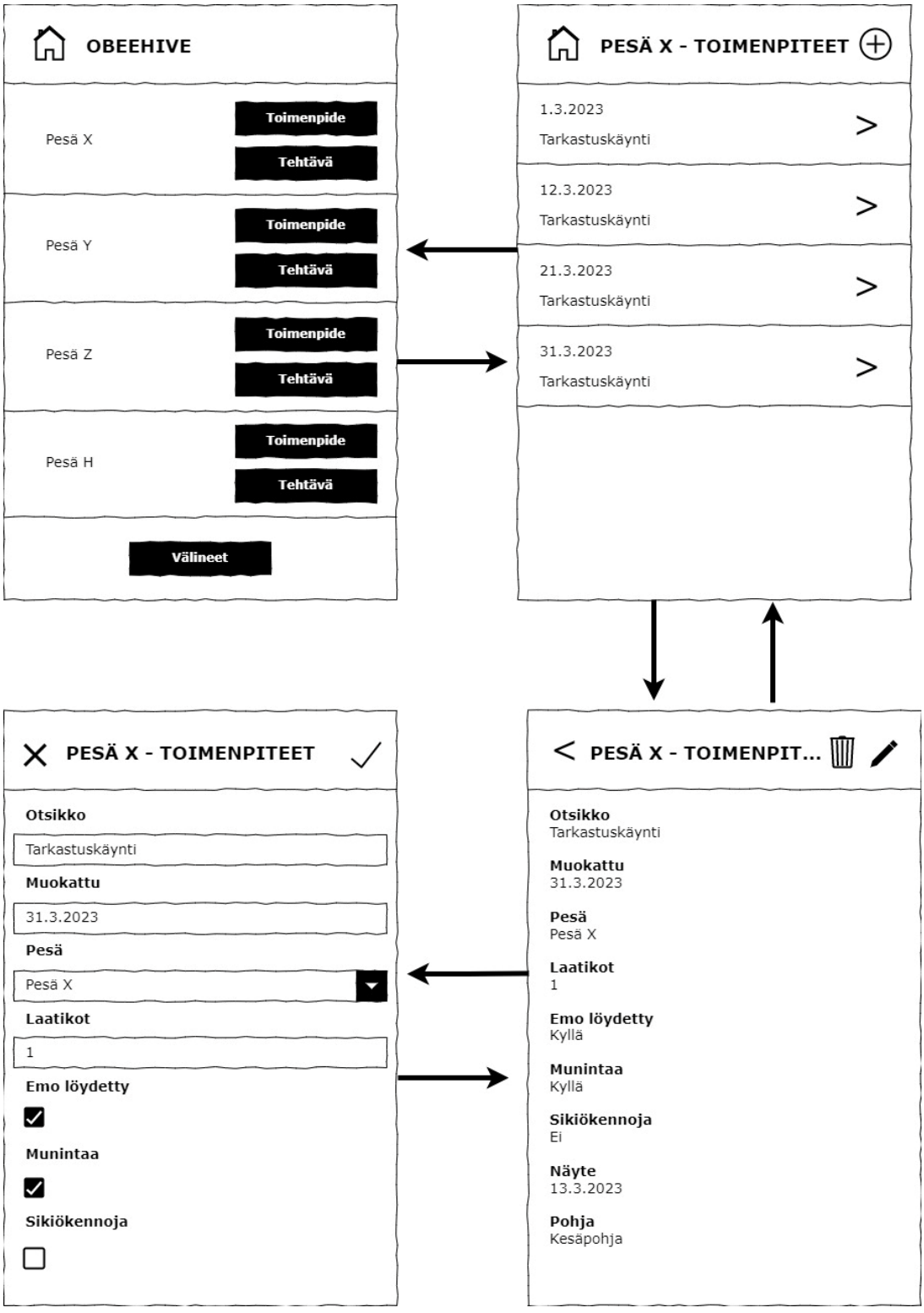
Visure. 2023a. Mitä ovat toiminnalliset vaatimukset: esimerkit, määritelmä, täydellinen opas. Blogi. Luettavissa: https://visuresolutions.com/fi/blog/functional-requirements/ Luettu 2.4.2023

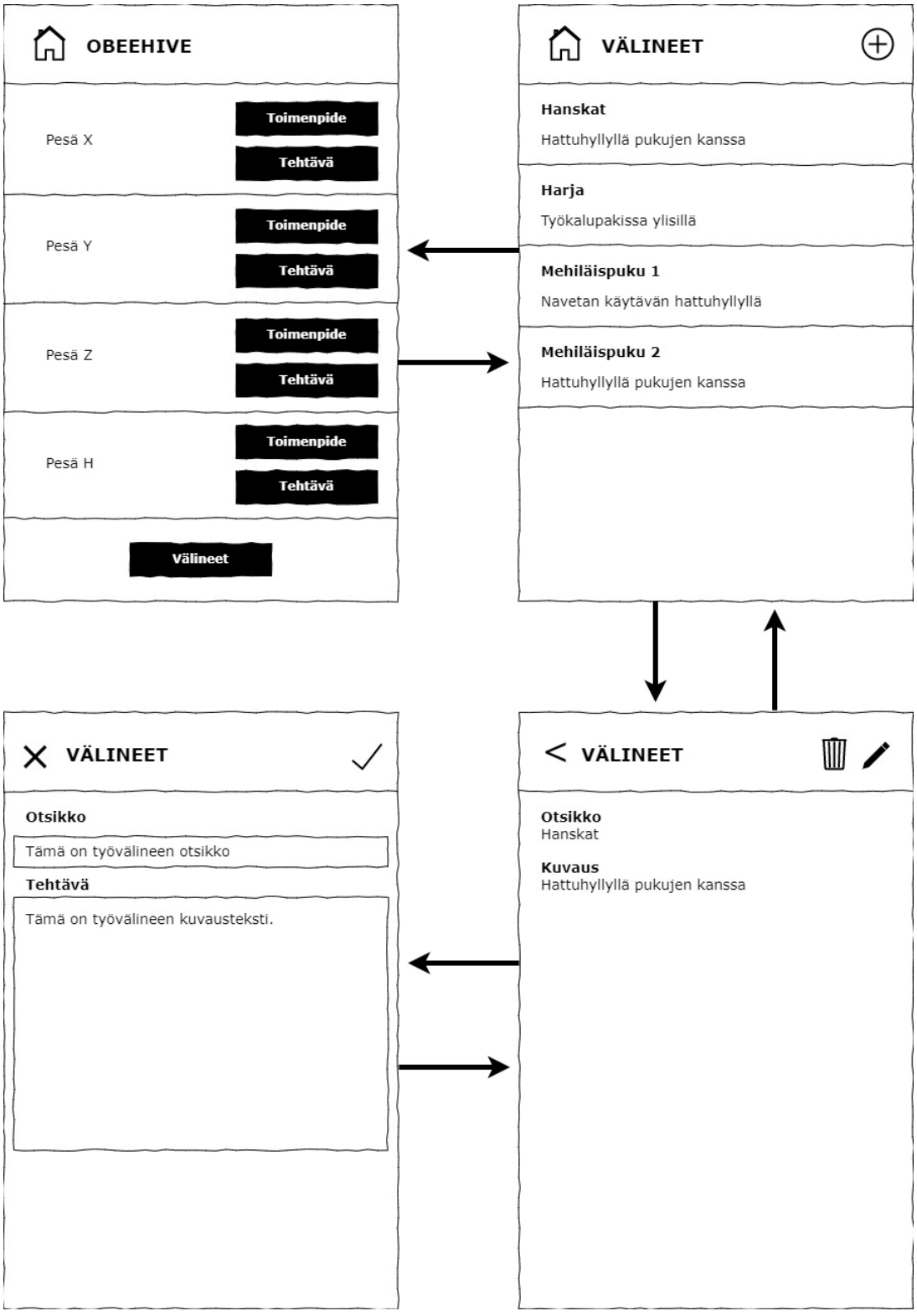
Visure. 2023b. Mitä ovat ei-toiminnalliset vaatimukset: esimerkit, määritelmä, täydellinen opas. Blogi. Luettavissa: https://visuresolutions.com/fi/blog/non-functional-requirements. Luettu 2.4.2023

Weston, M. 2019. E-kirja. Learn Microsoft PowerApps: Build customized business applications without writing any code. Packt Publishing.

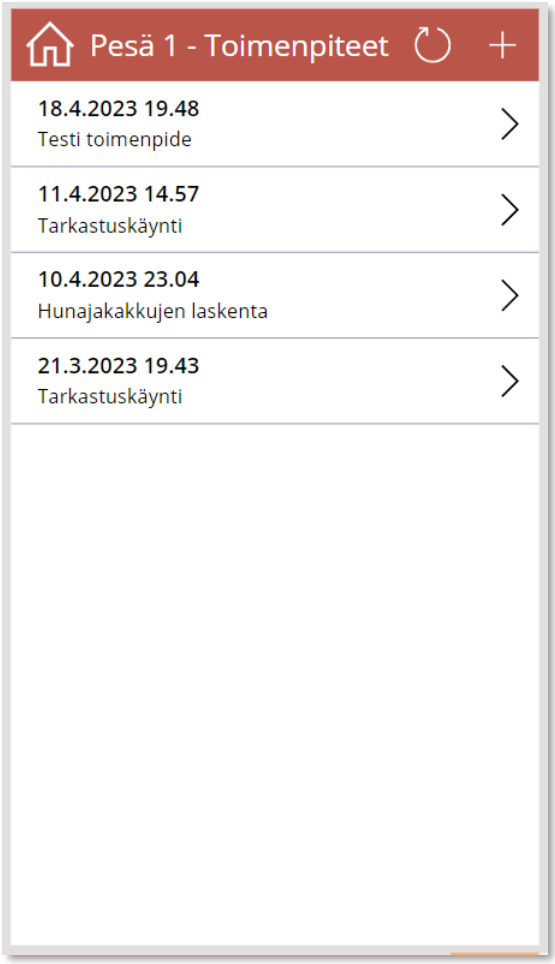
6 Liitteet

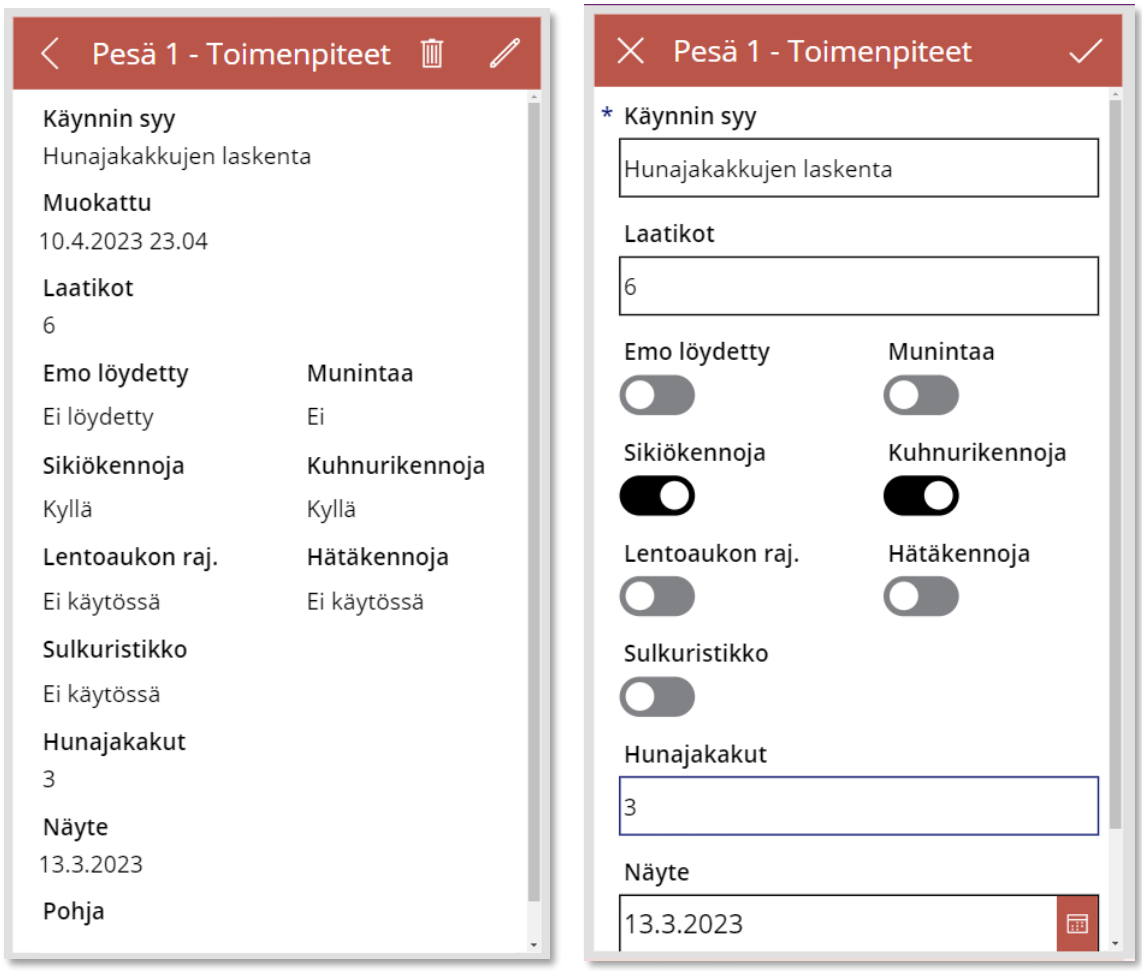
Liite 1. Rautalankamalli

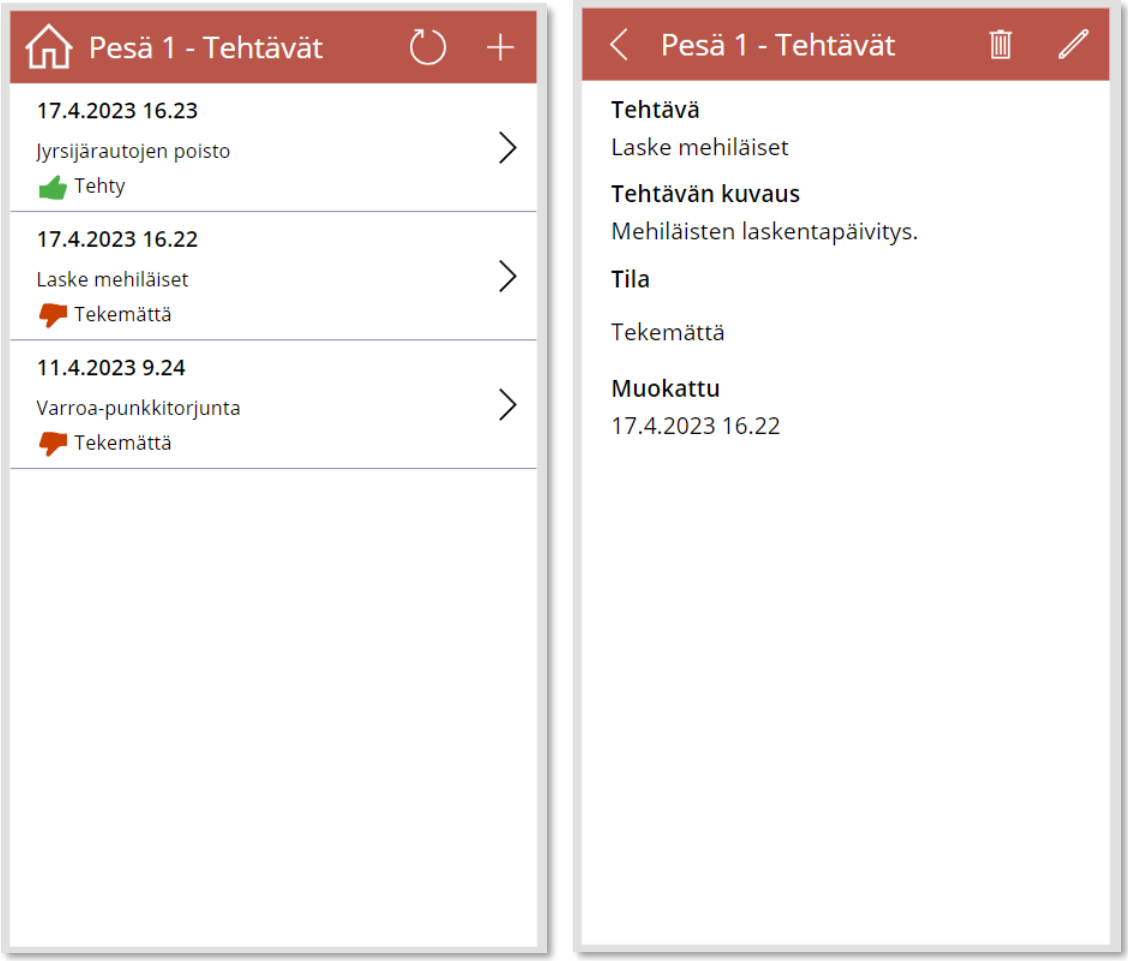


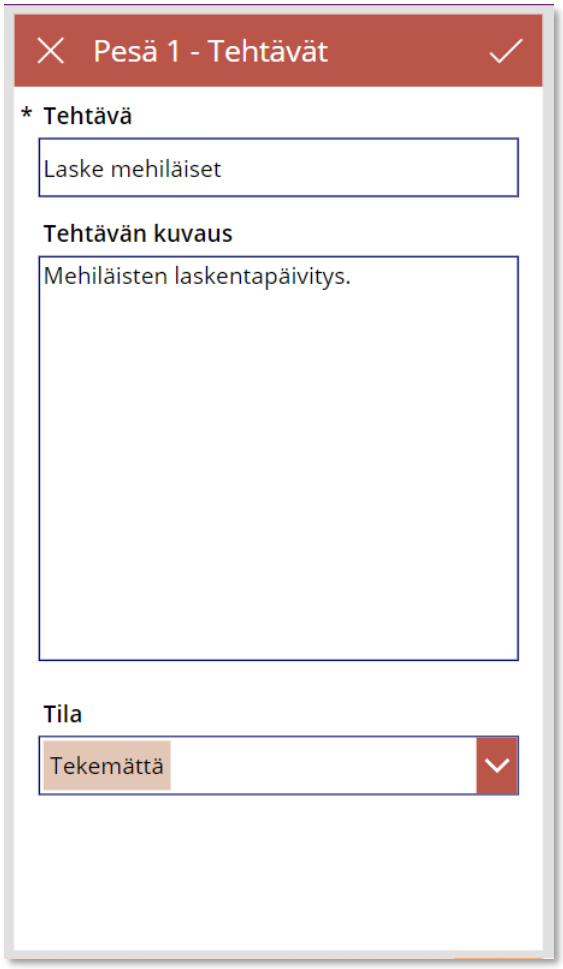

OBEEHIVE
Pesä X
Toimenpide
Tehtävä
Pesä Y
Toimenpide
Tehtävä
Pesä Z
Toimenpide
Tehtävä
Pesä H
Toimenpide
Tehtävä
Välineet
VÄLINEET
Hanskat
Hattuhyllyllä pukujen kanssa
Harja
Työkalupakissa ylisillä
Mehiläispuku 1
Navetan käytävän hattuhyllyllä
Mehiläispuku 2
Hattuhyllyllä pukujen kanssa
VÄLINEET
Otsikko
Tämä on työvälineen otsikko
Tehtävä
Tämä on työvälineen kuvausteksti.
VÄLINEET
Otsikko
Hanskat
Kuvaus
Hattuhyllyllä pukujen kanssa

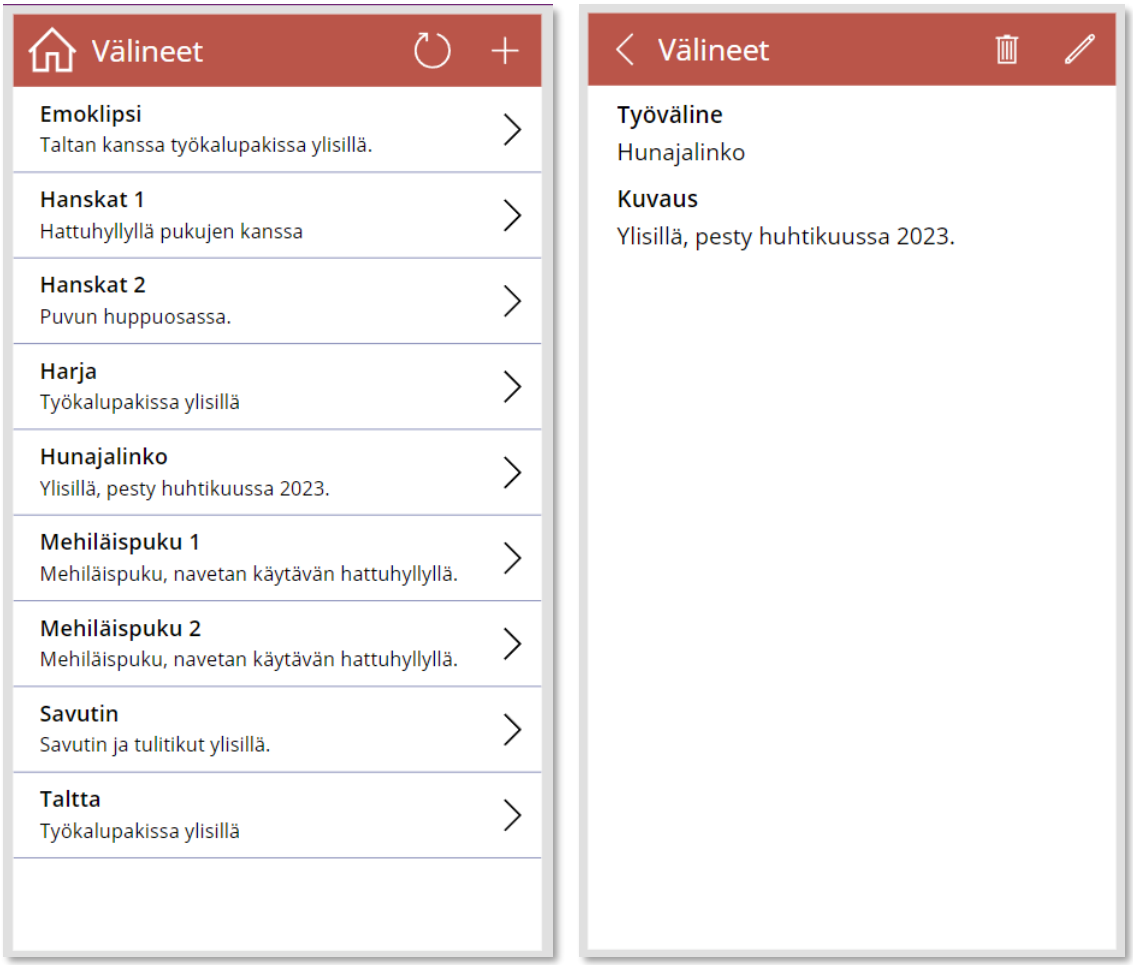
Liite 2. Sovelluksen näytöt

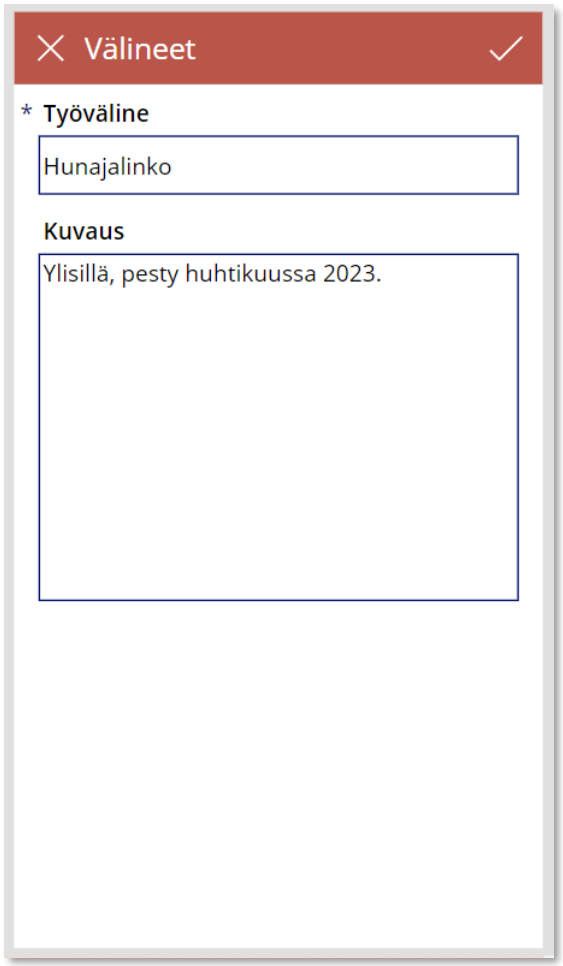



Pesä 1 - Toimenpiteet
Käynnin syy
Hunajakakkujen laskenta
Muokattu
10.4.2023 23.04
Laatikot
6
Emo löydetty
Ei löydetty
Munintaa
Ei
Sikiökennoja
Kyllä
Kuhnurikennoja
Kyllä
Lentoaukon raj.
Ei käytössä
Hätäkennoja
Ei käytössä
Sulkuristikko
Ei käytössä
Hunajakakut
3
Näyte
13.3.2023
Pohja
Pesä 1 - Toimenpiteet
* Käynnin syy
Hunajakakkujen laskenta
Laatikot
6
Emo löydetty
Munintaa
Sikiökennoja
Kuhnurikennoja
Lentoaukon raj.
Hätäkennoja
Sulkuristikko
Hunajakakut
3
Näyte
13.3.2023


Pesä 1 - Tehtävät
17.4.2023 16.23
Jyrsijärautojen poisto
Tehty
17.4.2023 16.22
Laske mehiläiset
Tekemättä
11.4.2023 9.24
Varroa-punkkitorjunta
Tekemättä
Pesä 1 - Tehtävät
Tehtävä
Laske mehiläiset
Tehtävän kuvaus
Mehiläisten laskentapäivitys.
Tila
Tekemättä
Muokattu
17.4.2023 16.22


Pesä 1 - Tehtävät
* Tehtävä
Laske mehiläiset
Tehtävän kuvaus
Mehiläisten laskentapäivitys.
Tila
Tekemättä


Välineet
Emoklipsi
Taltan kanssa työkalupakissa ylisillä.
Hanskat 1
Hattuhyllyllä pukujen kanssa
Hanskat 2
Puvun huppuosassa.
Harja
Työkalupakissa ylisillä
Hunajalinko
Ylisillä, pesty huhtikuussa 2023.
Mehiläispuku 1
Mehiläispuku, navetan käytävän hattuhyllyllä.
Mehiläispuku 2
Mehiläispuku, navetan käytävän hattuhyllyllä.
Savutin
Savutin ja tulitikut ylisillä.
Taltta
Työkalupakissa ylisillä
Välineet
Työväline
Hunajalinko
Kuvaus
Ylisillä, pesty huhtikuussa 2023.


Välineet
* Työväline
Hunajalinko
Kuvaus
Ylisillä, pesty huhtikuussa 2023.