LOW-CODE -SOVELLUSKEHITYS

PK-YRITYKSEN LIIKETOIMINNASSA

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
syksy, 2021

Laura Ruotsalainen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Hämeenlinnan korkeakoulukeskus

Tekijä Laura Ruotsalainen Vuosi 2021

Työn nimi Low-code -sovelluskehitys pk-yrityksen liiketoiminnassa

Ohjaajat Mirlinda Kosova-Alija

TIIVISTELMÄ

Digitalisoituvan maailman alati kiihtyvään sovellustarpeeseen vastaaminen on haaste varsinkin pienille ja keskisuurille yrityksille perinteisen ohjelmoinnin hitauden ja hintavuuden vuoksi. Opinnäytetyön tavoitteena oli tutustua pk-yrityksen näkökulmasta low-code -sovelluskehitykseen ja toteuttaa low-code -alustalla laskutusprosessia tehostava tietokantapohjainen web-sovellus. Opinnäytetyön toimeksiantaja oli toimistosiivousalalla toimiva perheyritys.

Opinnäytetyön tietopohjassa pureuduttiin low-code -sovelluskehitykseen, sen pk-yrityksille tarjoamiin mahdollisuuksiin ja low-code -työkalujen valintaperusteisiin. Teoreettisessa osuudessa ei syvennytty muuhun sovelluskehitykseen tai tarkemmin esitelty sovellus-, käyttöliittymä- tai tietokantasuunnittelua. Kehittämisprojektissa käytettiin konstruktiivista tutkimusmenetelmää ja projektimallina kanban-menetelmää. Opinnäytetyö oli toiminnallinen.

Lopputuloksena syntynyt web-sovellus rakennettiin kolmella eri low-code -alustalla, joista valittiin toimeksiantajan tarpeisiin sopivin. Kehittämistyö onnistui yli odotusten ja toimeksiantaja oli erittäin tyytyväinen käyttöönotettuun sovellukseen, jonka aikaansaama aikasäästö oli merkittävä. Johtopäätöksenä voidaan todeta, että low-code -sovelluskehityksen nopeudellaan, kustannustehokkuudellaan ja helppoudellaan pk-yrityksille tuoma kilpailuetu on kiistaton.

Avainsanat low-code -sovelluskehitys, low-code -alusta, web-sovellus

Sivut 60 sivua ja liitteitä 12 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Abstract

Author Laura Ruotsalainen
Year 2021
Subject Low code development for SMEs
Supervisors Mirlinda Kosova-Alija

ABSTRACT

The ongoing digital transformation rapidly increases the need for business applications which can be challenging particularly for small and medium-sized enterprises because traditional coding is often time-consuming and expensive. The purpose of this thesis was to create a web database application to help accelerate invoicing. The project was commissioned by a family-owned office cleaning company.

The thesis describes what low code development is, how SMEs can streamline their business with low code development and what to consider when choosing a low code platform. The study was carried out as a development project and both constructive research and Kanban method were used.

The application was created with three different low code development platforms and the best suited for the commisioner was implemented. The application exceeded expectations and resulted in significant time savings. The conclusion was that the speed, cost effectiveness and simplicity of low code development undeniably create a competitive advantage for the SMEs.

Keywords low code development, low code application platform, web application
Pages 60 pages and appendices 12 pages

Sanasto

aPaaS	sovellusalusta palveluna (Application Platform as a Service)
appi	yksinkertainen, rajattuun käyttötarkoitukseen sopiva sovellus
back-end	palvelinpuoli, koodi ajetaan sivuston palvelimella
CRUD	neljä perustoimintoa, joilla tietokannan dataa käsitellään (Create, Read, Update, Delete)
end-to-end	alusta loppuun (päästä päähän)
front-end	selainpuoli, koodi ajetaan verkkoselaimessa
kanban	ketterä projektinhallintamenetelmä
kansalaiskehittäjä	liiketoiminnan puolelta tuleva loppukäyttäjä, joka ilman koodaustaitoja pystyy low-code -työkalujen avulla rakentamaan sovelluksia usein omaan tai tiimin käyttöön (citizen developer)
kansalaiskehittäminen	sovelluskehitystä, jossa yrityksen valveutunut työntekijä rakentaa erilaisia sovelluksia itselleen tai tiimilleen tärkeisiin prosesseihin low-code -ratkaisuja hyödyntämällä (citizen development)
käytettävyys	sovelluksen helppokäyttöisyys
LCAP	low-code -alusta (Low-Code Application Platform)
LCNC	low-code/no-code
low-code	visuaalinen mallintaminen mahdollisimman vähäisellä koodin kirjoittamisella

low-code -alusta	graafinen käyttöliittymä, jossa sovellus visuaalisesti rakennetaan pudottamalla ja raahaamalla valmiiksi koodattuja elementtejä
low-code -työkalu	sovelluskehitystyökalu, joka kirjoittaa koodin visuaalisen määrittelyn perusteella
no-code	visuaalinen mallintaminen ilman koodin kirjoittamista
RAD	nopean kehityksen malli, joka pyrkii nopeuttamaan applikaatioiden kehitysprosessia (Rapid Application Development)
roadmap	keskipitkän tähtäimen suunnitelma, kehityksen suuntaviivat, usein aikajanalle aseteltu joukko tavoitteita
SaaS	pilvessä sijaitseva vuokrattava ohjelmisto, jota palveluntarjoaja ylläpitää (Software as a Service)
SWOT	nelikenttämenetelmä (Strengths, Weaknesses, Opportunities, Threats)
UI	käyttöliittymä (User Interface)
vähäkoodinen	low-coden suomennos
web-sovellus	käyttöliittymän kautta verkkosivulla käytettävä sovellus

Sisällys

Kuvat ja taulukot

Liitteet

Liite 1 Aineistonhallintasuunnitelma

Liite 2 Käyttäjätarinat

Liite 3 Luonnokset sovelluksen käyttöliittymäruuduista

Liite 4 Outsystems-alustalla rakennettu sovellus

Liite 5 Joget DX - alustalla rakennettu sovellus

Liite 6 Saltcorn-alustalla rakennettu sovellus

1 Johdanto

Sovelluskehitystä on perinteisesti pidetty aikaa ja rahaa vievänä prosessina, johon pk-yrityksillä ei ole ollut resursseja. Low-code -työkalujen myötä tilanne on kuitenkin muuttumassa, koska niiden avulla voi käytännössä kuka tahansa luoda sovelluksen jopa ilman erityisiä koodaustaitoja. Näin myös pk-yritysten ulottuville on tullut mahdollisuus toteuttaa tarvitsemiaan sovelluksia nopeasti ja kustannustehokkaasti.

Low-code -sovelluskehitys on aiheena erittäin ajankohtainen, koska tarve erilaisille sovelluksille kiihtyy jatkuvasti eikä vähiten pandemian vauhdittaman digiloikan ansiosta. Toimiva prototyyppi voidaan aiempaa pienemmillä kustannuksilla määrittää low-code -alustoilla visuaalisesti jopa muutamassa tunnissa. Sovellus on heti käyttövalmis ja sen ylläpito on vaivatonta. Low-code tarjoaa helpotusta kasvavaan koodaripulaan ja paineeseen saada sovelluksia nopeammin käyttöön (Reinikainen, 2021). Kansainvälisen tutkimus- ja konsultointiyhtiö Gartnerin ennustuksen mukaan low-code -tekniikalla toteutetaan yli 65 prosenttia sovelluksista ja vähintään 4 low-code -työkalua käyttää 75 prosenttia suurista yrityksistä vuoteen 2024 mennessä (Vincent et al., 2019).

Low-code -sovelluskehitystä on käsitelty vielä verrattain vähän tutkimuksissa eikä low-code -alustojen käyttöä juurikaan ole vertailtu keskenään. Low-code -sovelluskehitystä käsittelevissä artikkeleissa ja tutkimuksissa on keskitytty lähinnä suuriin yhtiöihin ja lähes unohdettu pk-yritykset, joille kuitenkin low-code -tekniikoiden käyttöönoton vaikutukset voivat yrityksen mittakaavassa olla hyvinkin merkittäviä.

Työn aihe syntyi toimeksiantajan tarpeesta tehokkaampaan laskutusprosessiin ja työssä toteutettavalla web-sovelluksella pyritään lisäksi parantamaan toimeksiantajan raportointia ja tilausten sekä tuotemenekin seurantaa.

Tutkimuskysymykseni ovat seuraavat:

- Mitä on low-code -sovelluskehitys?
- Miten pk-yritys voi tehostaa liiketoimintaansa low-code -sovelluskehityksen avulla?
- Mitä pk-yrityksen kannattaa huomioida low-code -alustan valinnassa?

2 Low code -sovelluskehitys

Low-code -sovelluskehityksessä sovelluksia rakennetaan perinteisen käsinkoodaamisen sijaan lähes kokonaan konfiguroimalla eli muokkaamalla asetuksia low-code -alustan pyörittämässä visuaalisessa käyttöliittymässä. Low-coden ansiosta työ tehostuu, perinteisen koodauksen tarve vähenee, sovelluskehitys nopeutuu, kulut pienenevät ja sovelluksien ylläpito helpottuu. (Leskinen, 2017) Kotilainen (2018) kiteyttää low-code -ratkaisut ytimekkäästi yhteen lauseeseen: ”Low-code-työkalut eivät sovellu kaikkeen kehitystyöhön, mutta ne tekevät jo monissa paikoissa ohjelmoinnista niin nopeaa ja edullista, ettei vanhaan ole paluuta.”

Joulukuussa 2020 Arrow ECS Finland ja Microsoft kartoittivat Suomen low-code -markkinaa toteuttamalla kyselyn pk-sektorin IT-hankinnoista vastaaville henkilöille. Siinä selvisi, että valtaosaa tietotekniikkapäättäjistä kiinnostavat low-code -ratkaisut ja niiden tarjoamat mahdollisuudet mutta jopa joka viides on niistä tietämätön. Vain noin 20 % suomalaisista pk-yrityksistä hyödyntää low-code -alustoja liiketoiminnassaan, vaikka yli puolet vastaajista näkee etteivät perinteiset sovellukset enää täysin vastaa nykyisen toimintaympäristön tarpeisiin. Lisäksi lähes kolme neljäsosaa kokee manuaalisten ja paperiprosessien hitauden ja virheiden aiheuttavan liiketoiminnalle haasteita. (Arrow ECS Finland Oy, 2020)

Low-code -termille ei ole vakiintunutta suomenkielistä ilmaisua ja toisinaan se suomennetaan muotoon vähäkoodinen. Wennströmin (2021) mielestä englanninkielinen sana ei ole erityisen kuvaava, mistä syystä jotkut alustatoimittajat puhuvat nykyisin modernista ohjelmistokehityksestä. Tässä opinnäytetyössä käytän low-code -käsitettä alkuperäisessä kirjoitusasussaan ns. sitaattilainana. Korhosen (2019) mukaan englantia lainataan yhä matalammalla kynnyksellä omaan kieleen ja tietotekniikassa kynnys on nykyisin olematon. On kuitenkin huomattava, että sitaattilainan ääntäminen sekä taivuttaminen voivat tuottaa suomalaisille vaikeuksia eivätkä englantia taitamattomat sitä välttämättä ymmärrä.

2.1 Low-code -sovelluskehityksen toimintaperiaate

Linna (2021) tiivistää low-code -sovelluskehityksen olevan ohjelmien tekemistä nopeasti myös vähäisillä tai jopa olemattomilla ohjelmointitaidoilla ja nostaa sen yhdeksi tietotyöläisen valmiuksien kehittämisen viimevuosien taikasanoista. Low-code -kehitysalustoilla sovellusten visuaalinen ilme ja logiikka rakennetaan käyttöliittymässä pääasiassa raahaamalla ja pudottamalla valmiita elementtejä näkymään. Visuaalisuus sujuvoittaa työn hahmottamista ja muokkaamista siinä määrin, että low-coden arvioidaan olevan 3 - 5 kertaa perinteistä ohjelmointia nopeampaa. Low-code -alusta huolehtii taustalla natiivin koodin tuottamisesta, mikä vähentää virheiden määrää merkittävästi. Low-code -työkalut on helppo omaksua, mutta tehokäytössä vaaditaan vähintään perustietoja koodauksesta ja ohjelmointilogiikasta. Koodaustaitoja tarvitaan esimerkiksi tyylejä tai ulkonäköä räätälöitäessä, mutta täysin toimivia sovelluksia voidaan luoda ilmankin niitä. (Kailio, 2020)

Low-code siis poistaa valtaosan manuaalisen koodauksen tarpeesta hyödyntämällä sen sijaan tietovarastoja, rajapintoja ja kolmannen osapuolen infrastruktuuria. Low-code -alustan graafisessa käyttöliittymässä tehtyjen workflow-, datamalli- ja UI-määrittelyjen perusteella koodi tuotetaan ohjelmoijan puolesta valmiista koodinpalasista. Sovelluksen toimintalogiikka rakennetaan erilaisten vuokaavioiden, toisiinsa kytkeytyvien elementtien ja excel-tyyppisten kaavojen avulla. Koodia voi halutessaan tarkastella ja tarpeen mukaan muokata tai tehdä siihen lisäyksiä ja laajennuksia manuaalisesti. Huippukoodareita tarvitaan kuitenkin edelleen vaativien ja monimutkaisten järjestelmien rakentamisessa, johon low-code ei ainakaan vielä taivu. (a.i.mater Oy, 2018)

Low-code -työkalujen avulla sovelluskehittäminen on melko suoraviivaista ja näkyviä tuloksia saadaan aikaan nopeasti. Yksinkertaisia prosesseja, joiden tietovarastona voi olla vaikkapa excel-tiedosto tai Sharepoint-lista, voidaan muuttaa mobiiliapplikaatioiksi jopa tunneissa ja samalla päästään eroon esimerkiksi excel-taulukoista tai paperilomakkeista. Työkalujen käyttöönotto on vaivatonta ja luoduilla sovelluksilla voidaan parantaa sisäisten prosessien käytettävyyttä. Omaa työtä helpottavan sovelluksen kehittäminen kuitenkin edellyttää, että työntekijä ymmärtää siihen liittyvät prosessit ja ylätasolla järjestelmien, sovellusten sekä datan syy-yhteyksiä. Tähän kykenevillä työntekijöillä on huima liiketoimintapotentiaali

ideoida omasta perspektiivistään sovelluksia, jotka joko tehostavat yrityksen nykyistä tai synnyttävät kokonaan uutta liiketoimintaa. (Lyytinen, 2020)

Työntekijätasolla low-code muuttaa tietojärjestelmien käyttäjiä asteittain kehittäjiksi, jotka aktiivisesti osallistuvat järjestelmien parantamiseen ja puutteiden korjaamiseen. Low-code on usein it-kumppanin hallinnoima kehitysalusta, joka synergisesti tarjoaa yritykselle mahdollisuuden kehittää sovelluksia. Ne ovat usein nopeasti kehitettyjä täsmäratkaisuja, joita liiketoiminnan muuttunut tilanne vaatii. (Kuivainen & Laukkanen, 2020)

2.2 Low-code -sovelluskehityksen kehittyminen ja tulevaisuus

Low-code ei ole uusi keksintö, vaan siihen on johtanut pitkä kehityskulku. Internetin ja sen käyttäjämäärän kasvun myötä alettiin jo vuosikymmeniä sitten vaatia ohjelmointikieliltä ja sovelluksilta yhä enenevässä määrin yhteensopivuutta, helppokäyttöisyyttä ja toimintakeskeisyyttä. Tuolloin ryhdyttiin kehittämään niin sanottuja neljännen sukupolven ohjelmointikieliä kehitystyön tehostamiseksi. Tämä avasi tietä nopean kehityksen mallille (RAD) ja low-code -ratkaisuille, joilla koodaajan työtä pystyttiin merkittävästi nopeuttamaan ja keventämään. Viime vuosien aikana markkinoille on saatu useita toimivia low-code -työkaluja, joilla uusia sovelluksia voidaan luoda nopeasti ja helposti. (Kissflow, 2018)

Terminä low-code esiintyy ensimmäisen kerran Forrester Researchin raportissa 2014, jossa todetaan low-code -alustojen mahdollistavan nopean sovelluskehittämisen vähäisellä koodaamisella ja pienillä asennus-, koulutus- ja käyttöönottoinvestointikuluilla. Raportissa tuodaan esiin low-code -alustojen suosion kiihtyminen kuluttajille suunnattujen sovellusten rakentamisessa, joista useimpien kehittäminen perinteisesti koodia käsin kirjoittamalla on liian hidasta. Low-code -alustat sekä nopeuttavat kehittämistyötä että helpottavat sovelluksesta saatuun käyttäjäpalautteeseen reagoimista. (Forrester, 2014)

Liiketoiminnan digitalisaatio on lisännyt räätälöityjen sovellusten kysyntää ja luonut painetta nostaa sovellusten toimitusnopeutta merkittävästi. Tämä on saanut aikaan low-code -ratkaisujen ja kansalaiskehittäjien ilmaantumisen. Gartnerin (2021) tutkimuksen mukaan keskimäärin jo 41 % niin sanotuista tavallisista työntekijöistä osallistuu teknologiaratkaisujen kehitystyöhön. Joustavasti hinnoitellut low-code -ratkaisut

mahdollistavat etätyöskentelyä tukevien prosessien uudistamisen käyttäjälähtöisemmiksi ja ennustetaankin, että puolet kaikista uusista low-code -asiakkuuksista tulee olemaan it-organisaatioiden ulkopuolelta vuoden 2025 loppuun mennessä. Gartnerin varatutkimusjohtaja Biscotti arvelee, että maailmanlaajuisesti katsoen valtaosalla suurista organisaatioista tulee olemaan käytössään useita low-code -työkaluja jo vuoden 2021 lopulla. Pidemmällä aikavälillä yritykset tulevat omaksumaan jatkuvan muutoksen periaatteen ja siirtyvät käyttämään low-code -teknologioita, jotka tukevat sovellusinnovaatioita ja -integraatioita. (Gartner, 2021)

Low-code -alustojen voidaan katsoa syntyneen paikkaamaan koodarivajetta, jonka on luonut yritysten, koulutuksen ja yhteiskunnan kaikkien osa-alueiden alati kasvava sovellustarve. Kun kännykkäsovellus syntyy valmiista palasista, ei vaikkapa henkilöstöosaston loma-anomusapin tekemiseen vaadita enää ohjelmointitaitoista ihmistä. Legopalikoiden tapaan toimivia elementtejä yhdistellen organisaation teknologiamyönteisimmät työntekijät voivat nopeasti itse rakentaa käyttöönsä yksinkertaisia sovelluksia korvaamaan esimerkiksi paperiset lomakkeet ja samalla helpottaa jokapäiväistä työtään. Low-code -maailmassa ammattikehittäjät pystyvät jopa minuuteissa rakentamaan tuleville käyttäjille testiversioita, joita sitten parannellaan käyttäjäpalautteen perusteella. (Åkerman, 2021)

Kansainvälinen tutkimus- ja konsultointiyhtiö Gartner (2021) ennustaa low-code -markkinoiden kasvavan pandemian vauhdittamana maailmanlaajuisesti lähes 23 % vuonna 2021 huolimatta siitä, että yrityksissä pyritään optimoimaan kuluja. Kuten taulukosta 1 selviää on koko low-code -markkinan ja erityisesti low-code -alustojen (LCAP) tuloskasvu ollut lukujen valossa merkittävää ja sen odotetaan edelleen kiihtyvän. Koska kaikki suurimmat Saas-toimittajat tarjovat jo low-code -kehitystekniikkaa sisältäviä ominaisuuksia, hyödyttää pilvipalveluiden lisääntyminen välillisesti myös low-code -markkinaa.

Taulukko 1. Low-code -markkinan tulos milj. USD (Gartner, 2021)

	2019	2020	2021
Low-Code Application Platforms (LCAP)	3,473.5	4,448.2	5,751.6
Intelligent Business Process Management Suites	2,509.7	2,694.9	2,891.6
Multiexperience Development Platforms (MDXP)	1,583.5	1,931.0	2,326.9
Robotic Process Automation (RPA)	1,184.5	1,686.0	2,187.4
Citizen Automation and Development Platform (CADP)	341.8	438.7	579.5
Other Low-Code Development (LCD) Technologies*	59.6	73.4	87.3
Overall	**9,152.6**	**11,272.2**	**13,824.2**

**Other LCD technologies include rapid mobile app development (RMAD) tools and rapid application development (RAD) tools. Low-code is the evolution of RAD to cloud and SaaS models.*

Forresterin (2020) mukaan jo 75 % sovelluskehityksestä tehdään low-code -alustoilla vuoden 2021 lopulla ja nousua edelliseen vuoteen tulee olemaan 44 %. Yritykset pystyvät säilyttämään kilpailukykynsä investoimalla modernisaatioon ja low-code voi nopeuttaa sovelluskehitystä jopa kymmenkertaisesti. Pandemian aikana on korostunut, että low-codea hyödyntämällä reagointi muutostarpeisiin on nopeampaa ja tehokkaampaa. Syyskuussa 2020 Microsoftin yritysjohtaja Lamanna (Ashbaugh, 2020) mainitsi esityksessään low-code -vallankumouksen ja kertoi, että Microsoft tulee julkaisemaan 500 miljoonaa uutta appia seuraavan viiden vuoden sisällä. Määrä on suurempi kuin 40 vuoden aikana julkaistut sovellukset yhteensä.

Laitila (2021) toteaa osuvasti: ”Low-coden lupaus on abstraktiotason nostaminen koodin syntaksin miettimisestä bisneslogiikan rakentamiseen ja kaiken mahdollisen automatisointiin. Samoin optimitilanteessa myös muut kuin kuukausien koulutusputken läpikäyneet voisivat kehittää hyödyllisiä palveluita itselleen ja muille töiden ohessa. Siihen on kuitenkin vielä matkaa. Sinänsä vaikuttaa siltä, että esimerkiksi web-kehityksen työkalut ja low-code -alustat kulkevat kohti samaa maalia, mutta eri suunnista. Oleellista on oikea työkalu oikeassa paikassa. ”

2.3 Low-code ja no-code -sovelluskehitys

Outsystemsin Forsythin (2021) mukaan low-code tarjoaa kaikentasoisille kehittäjille oikopolkuja, jotka nopeuttavat tuntuvasti kehitysprosessia ja vähentävät käsityönä tehtävää koodausta. Low-code -alusta tuottaa tarvittavan, oikeamuotoisen koodin taustalla eikä varsinainen sovelluskehittäminen juurikaan poikkea perinteisestä. Vähäisillä koodaustaidoilla alustojen käytön oppiminen voi kuitenkin olla hidasta, vaikka varsinaisen sovelluksen rakentaminen on nopeaa. Low-code sopii käyttäjäystävällisten, responsiivisten sovellusten ja portaalien kehittämiseen.

Termejä low-code ja no-code käytetään usein virheellisesti toistensa synonyymeinä ja toki molemmissa sovellus rakennetaan visuaalisesti raahaamalla ja pudottamalla elementtejä graafisessa käyttöliittymässä. Techrepublicin (2020) näkemyksen mukaan termien ero näkyy selvimmin alustojen käyttötarkoituksessa ja niiden käyttäjissä. Low-code -alustoja hyödyntävät useimmiten koodaustaitoiset it-ammattilaiset, jotka rakentavat alustoilla räätälöityjä, liiketoimintakriittisiä sovelluksia. No-code -työkaluja käyttävät pääsääntöisesti kansalaiskehittäjät luoden sovelluksia täysin ilman koodaustaitoja omiin ja yrityksen sisäisiin tarpeisiin ja optimoimaan päivittäisiä toimintoja.

No-code onkin suunnattu työntekijöille, jotka eivät välttämättä osaa ohjelmointikieliä. Heillä on kuitenkin tarve kehittää sovelluksia tiettyyn käyttöön, usein omalle osastolleen. No-code tarjoaa sovelluskehitykseen työkaluja, joiden käyttöön ei tarvita erillistä koulutusta. Se muistuttaa valmiiksi rakennettuja blogi- tai verkkokaupparatkaisuja ja alustan käyttö vaatii vain vähän harjoittelua, jonka jälkeen käytännössä kuka tahansa pystyy rakentamaan yksinkertaisen, vakiomuotoisen sovelluksen. Näennäinen helppous voi kuitenkin johtaa muun muassa tietoturva-, integraatio- ja vaatimuksenmukaisuusongelmiin, kun sovelluksia kehitetään ilman asianmukaista valvontaa ja liiketoiminnallista harkintaa. Tämän vuoksi no-code -työkaluja tulisi käyttää vain front-end -projekteihin. (Forsyth, 2021)

Kansainvälisen tutkimus- ja konsultointiyhtiö Forresterin analyytikko Rymerin ja tutkija Seguinin (2019) mukaan no-code on vain markkinointitermi, minkä lupaukset harvoin todellisuudessa toteutuvat. Yhtiön lanseeraama termi low-code päinvastoin myöntää jo nimellään, että aina tehokkaimmallakaan low-code -alustalla ei pystytä rakentamaan sovellusta täysin ilman koodausta vaan sitä saatetaan tarvita esimerkiksi räätälöitäessä käyttöliittymiä, integroitaessa toisiin sovelluksiin ja järjestelmiin tai vastattaessa erityisiin raportointivaatimuksiin.

2.4 Kansalaiskehittäminen

Kansalaiskehittämisestä tai -koodaamisesta (citizen coding) puhutaan silloin, kun niin sanottujen valkokaulusammattien teknisesti kyvykkäät työntekijät itse kehittävät liiketoimintaprosesseja ja työtään tukevia sovelluksia. Ilmiöllä pystytään helpottamaan yrityksen tietohallinnon työtaakkaa, kun usein resurssipulasta kärsivän it-osaston on tarjottava vain kehitystyökalut, käyttökoulutus ja valmiiden sovellusten tekninen tuki. Vaikka varsinaista koodausta low-code -työkalujen käytössä tarvitsee osata vain vähän, sitäkin tärkeämpää kehittäjän on ymmärtää liiketoimintaprosessi, jota varten sovellusta kehitetään. (Viestintätoimisto Manifesto, 2020)

Sovelluskehityksen povataan tulevaisuudessa olevan korkean teknologian maissa vääjäämättä osa työnkuvaa ja Outsystemsin helmi-maaliskuussa 2020 toteuttaman kyselytutkimuksen mukaan tähän uskoo lähes 20 % suomalaisistakin. Intoa tuntuisi kotimaisesta yritysmaailmasta löytyvän, sillä lähes 30 % kyselyyn vastanneista olisi halunnut ideoida ja koodata työtään helpottavan sovelluksen. Kansalaiskoodaaminen edellyttää kuitenkin liiketoimintayksiköiden ja tietohallinnon välillä sujuvaa yhteistyötä, mikä Suomessa valitettavasti on vielä lapsenkengissään. Jos henkilöstö kehittää sovelluksia IT-osaston tietämättä, on vaarana tietoturvariskien ja jopa kustannusten lisääntyminen. (Viestintätoimisto Manifesto, 2020)

2.5 Low-code -sovelluskehityksen hyödyt

Tampereen korkeakouluyhteisön Digitalisaatio-kehityshankkeen puitteissa järjestettyjen pilottikoulutusten tavoitteena oli selvittää Microsoftin low-code -työkalujen tuottamaa

lisäarvoa. Tiedekuntia ja palveluorganisaatioita edustavilta pariltakymmeneltä osallistujalta kerätyn kyselypalautteen perusteella Powerapps -alustan peruspotentiaali oli hyvin ymmärrettävissä ja yhdeksän kymmenestä piti sitä yleisesti hyödyllisenä. Oman työn kannalta kuitenkin vain kuusi kymmenestä näki hyötypotentiaalia ja muun muassa sovellusten päivitys- ja korjaustarpeet aiheuttivat pohdintaa. (Linna, 2021)

Low-coden suurimmiksi hyödyiksi voidaan katsoa tehokkuus, ketteryys ja nopeus, monipuolisuus ja -käyttöisyys, kustannustehokkuus sekä koodaripulaan vastaaminen. Kehittäjille tarjoutuu tilaisuus keskittyä täysipainoisesti luomiseen, kun infrastruktuuri ja uudelleenimplementoinnit voidaan ohittaa. Low-code mahdollistaa runsaamman määrän testausta, kokeilua ja pilotointia sekä ketterän reagoinnin bugeihin ja käyttäjäpalautteeseen kuten myös prosessien ja ympäristöjen muutoksiin. Low-code -työkaluja voivat eritasoisesti hyödyntää organisaation sisällä monentasoiset työntekijät ja synnyttää uusia innovaatioita useamman henkilön voimin esimerkiksi kehitystiimeissä. Low-coden avulla toteutettuja turvallisia ja laadukkaita sovelluksia on kevyt ylläpitää ja niiden elinkaarta on helppo hallita. (a.i.mater Oy, 2018)

Koska kehitystyö, testaus ja muutoksiin reagointi on low-coden ansiosta nopeaa ja tehokasta, luonnollisesti myös kustannustehokkuus paranee. Tämä taas mahdollistaa räätälöityjen sovellusten rakentamisen jopa hyvinkin pienille käyttäjämäärille ja saattaa tarjota ratkaisun myös nykypäivänä yleiseen kokeilukulttuuriin ja startup-henkiseen nopealiikkeisyyteen. (Tapanainen, 2018)

Kotilainen (2019) teroittaa nimenomaan low-code -sovelluskehityksen nopeuden olevan useille yrityksille kustannussäästöjä tärkeämpää, koska vikkelästi markkinoille pääsemällä ehtii liikevaihtoa kerätä pidemmältä ajalta. Myös markkinoiden ja käyttäjätrendien muutoksiin pystytään vastaamaan ripeästi, kun erilaiset sovellusversiot ja jopa 80 – 90 prosenttia ohjelmakoodista luodaan low-code -alustan toimesta automaattisesti. Tarve rekrytoida ammattitaitoisia kehittäjiä pienenee, kun alusta huolehtii laadukkaan ja tasalaatuisen koodin tuottamisesta sovelluksen tekijästä riippumatta. Lisäksi tietoturva on sisäänrakennettu alustoihin ja ohjelmakoodi on lähtökohtaisesti varsin turvallista, testattua ja koeteltua vakiokoodia. (Kotilainen, 2018)

Low-code -ajattelu pitää sisällään myös työnkulkujen mallintamisen ja niiden automatisoinnin sekä ohjelmistorobotiikan. Low-code -työkaluilla pystytään kätevästi määrittelemään monimutkaisiakin prosesseja ja hyväksyntäketjuja sekä automatisoimaan toistuvia prosesseja, mikä vähentää puuduttavaa rutiinityötä ja inhimillisiä virheitä. Low-code avaakin ennennäkemättömiä digitaalisia mahdollisuuksia liiketoiminnan prosessien kannattavaan modernisointiin. (Åkerman, 2021)

Moni pk-yritys kamppailee edelleen digitaalisen muutoksen ristiaallokossa. Low-code on ratkaisu asiantuntijoiden tai oman tietotaidon puutteeseen ja jopa sopivan ohjelmiston valinnassa koettuun epävarmuuteen. Low-code tarjoaa valtavan edun pk-yrityksille, jotka saavat pienillä investoinneilla suurten yritysten kanssa tasavertaiset ja monipuoliset sekä rajoittamattomat mahdollisuudet prosessien, tuotteiden ja palvelujen digitaaliseen kehittämiseen. Kun yrityksessä hyödynnetään ketteriä menetelmiä ja yhteistyötiimejä, toimii low-code -alusta siltana liiketoiminnan ja it:n välillä synnyttäen samalla kilpailuetua. (Garms, 2021)

Pk-sektorille tyypilliset rajalliset resurssit edellyttävät tehokkaita toimintatapoja, joita low-code -ratkaisut tarjoavat mahdollistaen samalla kustannustehokkaan keinon kilpailla suurempien yhtiöiden kanssa. Kilpailukykyä nostaa myös pk-yritysten kyky ketterämmin uudistaa it-strategiaansa ja käyttöönottaa uusia low-code -ratkaisuja. Esimerkiksi Microsoftin Power Platform -alusta voidaan koostaa nimenomaan omalle organisaatiolle ja sen työntekijöille sopivista tuotteista, joilla saadaan ketterästi kokeiltua ja toteutettua sekä innovaatioita että täsmäratkaisuja työtehtäviin. (Nyholm et al., 2021)

Liiketoiminnan uudistaminen vastaamaan nykyajan teknologisia vaatimuksia mahdollisimman pienin kustannuksin on jokaiselle yritykselle haaste. Poikkeuksellisen vaativaa se on pk-yrityksille, joille perinteinen sovelluskehitys on usein liian aikaavievä ja kallis prosessi. Pilvipohjaisilla low-code -alustoilla voi kuitenkin aloittaa pienin askelin ja siirtyä vähitellen vaativimpiin toteutuksiin. Low-code -alustojen tarjoamia hyötyjä ovat kustannustehokkuus, nopeus, koulutuksen tarpeen vähyys, päätelaiteriippumattomuus ja skaalautuvuus. Sisäiset tiimit voivat itse nopeasti ja vaivattomasti kehittää ja päivittää sovelluksia liiketoiminnan vaatimusten pohjalta. Sovelluksia on mahdollista ennen käyttöönottoa testata skaalautuvassa ja integroidussa ympäristössä. Sovelluskehitys

onnistuu yrityksen nykyisillä resursseilla, mikä säästää kustannuksia. On kuitenkin huomattava, että low-code -sovelluskehitys soveltuu parhaiten perustason digitalisointiin jonka monimutkaisuusaste on pieni. Low-code ei ole omimmillaan vaativammassa koodaustyössä kuten yksityiskohtaisesti räätälöidyissä projekteissa tai backendin rajapintojen luomisessa. (Kachot, 2020)

Low-code -alustojen avulla pk-yritykset voivat siis kehittää liiketoimintansa tueksi sovelluksia, johon niillä ei aiemmin ole ollut resursseja. Sovelluksilla voidaan muun muassa virtaviivaistaa sisäisiä prosesseja, automatisoida erilaisia toimintoja ja parantaa kaupankäyntiä ja vuorovaikutusta asiakkaiden kanssa. Alustasta riippuen sovelluskehitys on mahdollista aloittaa hyvinkin pienillä kustannuksilla eikä ulkopuolisiin kalliisiin sovelluskehittäjiin tarvitse turvautua. Työntekijöiltä ei vaadita erityisiä koodaustaitoja, mikä tarkoittaa, että käytännössä kuka tahansa yrityksen tietorakennetta ja prosesseja ymmärtävä voi omalla koneellaan rakentaa kattavan ja toimintakykyisen sovelluksen käyttöönotettavaksi. Pilvipohjaisilla alustoilla sovelluskehitys onnistuu samanaikaisesti myös useilla eri koneilla, mikä sopii erinomaisesti pandemian aikana yleistyneeseen etätyökonseptiin. Visuaaliset työkalut on nopea omaksua ja sovellusta pystytään yrityksen olemassa olevilla resursseilla päivittämään vastaamaan erityisiä tai muuttuvia liiketoiminnan tarpeita. (Carroll, 2021)

Näin ollen low-code -alustat ovat pk-yrityksille arvokkaita työkaluja, joiden avulla ne voivat omin voimin ilman koodaustaitoja luoda hetkessä räätälöityjä sovelluksia tietyn tehtävän suorittamiseen tai erityisen ongelman ratkaisemiseen tiivistää Guta (2021). Tarjolla olevista kaupallisista sovelluksista kun ei välttämättä löydy juuri yrityksen tarpeeseen sopivaa ratkaisua ja usein täsmäsovelluksen kehittämiseen vaaditaan lisäksi yrityksen sisäistä tietotaitoa.

2.6 Low-code -sovelluskehityksen käyttökohteita

Nykyisin low-code -alustoja käytetään monipuolisesti niin yksinkertaisten kuin kehittyneiden sovellusten ja jopa yritysten ydinjärjestelmien nopeaan kehittämiseen. Käyttö aloitetaan tyypillisesti yrityksen sisäisistä aikataulutuksen ja resurssoinnin työkaluista tai vaikkapa myyjille tehdyistä sovelluksista. Alustojen käyttöalueiden ja ominaisuuksien jatkuvasti

laajentuessa on helppo lähteä rakentamaan yritysten välisiä b2b-sovelluksia sekä kuluttajille suunnattuja mobiilisovelluksia ja web-palveluja. (Kotilainen, 2019)

Low-code -alustoja hyödynnetään erityisesti digitaalisen muutoksen tukena tuottamaan asiakkaille lisäarvoa, suoraviivaistamaan työnkulkuja, automatisoimaan datan integrointia ja tukemaan tiedon visualisoimista. Niiden avulla sovelluskehitystä pystytään nopeuttamaan ja ne tarjoavat myös useita muita lisäetuja. Low-code-alustat soveltuvat ohjelmiston elinkaaren eri vaiheisiin aina sovellusten suunnittelusta niiden käyttöönottoon ja seurantaan. Ne voivat pyöriä yrityksen omassa pilvessä, julkisessa pilvessä tai palvelinkeskuksissa enemmän tai vähemmän SaaS-palveluna. Alustoissa on kuitenkin tuntuviakin eroja ja jotkut esimerkiksi generoivat koodia toisten tuottaessa ennemminkin malleja. Osa alustoista on suunnattu vain ohjelmistokehityksen ammattilaisille, kun taas toiset on tarkoitettu sekä huippukoodarien että kansalaiskehittäjien käyttöön. Kehittyneimmillä alustoilla pystytään nykyisin rakentamaan monimutkaisiakin sovelluksia, ohjelmistologiikkaa, tietokanta-automatiikkaa ja integraatioita. Huomionarvoista on, että hiljattain myös julkiset pilvipalvelut kuten AWS, Azure ja Google ovat panostaneet low-code -ratkaisuihin. (Sacolick, 2020)

Wennström (2021) luettelee low-code -lähestymistavan soveltuvan yrityksen omien operatiivisten järjestelmien ja mobiiliapplikaatioiden kehitykseen, liiketoimintaprosessien digitalisaatioon, modernin käyttöliittymäkerroksen rakentamiseen vanhojen taustajärjestelmien päälle, sähköisten palvelujen, asiakasportaalien ja lomakkeiden kehitykseen sekä nopeisiin kokeiluihin, joissa toimivaa ohjelmaa halutaan käyttää oikeassa ympäristössä. Low-coden tehokkuus aikaansaadaan paketoimalla yhteen kaikki ohjelmistokehitykseen tarvittavat elementit ja monipuolisimmat alustat tarjoavat työkalut ohjelmistojen suunnittelusta ja kehityksestä aina valmiiseen PaaS-alustaan sekä niiden ajoon ja ylläpitoon.

Toisaalta osa low-code -työkaluista on kovin spesifisiä ja parhaimmillaan vain tietyillä toimialoilla muistuttaa Kotilainen (2018). Lisäksi low-codella voidaan luonnollisesti rakentaa vain sellaisia sovelluksia, joiden vaatimat elementit on koodattu alustalle valmiiksi. Jos pyritään maksimoimaan suorituskykyä tai optimoimaan ajoalustan hyödyntäminen, kannattaa hänen mielestään valita ennemmin perinteinen sovelluskehitys. Laitila (2021) lisää, että low-code -työkaluilla ei ainakaan vielä voi rakentaa kauttaaltaan räätälöityä

ulkoasua tai toimintaa eikä low-codea käyttämällä pystytä saamaan täyttä kontrollia infrastruktuuriin tai taustapalveluihin.

2.7 Low-code -alustan valintakriteereitä

Low-code -alustojen tarjoamat edut säästävät sovelluskehitykseen normaalisti kuluvaa aikaa, vaivaa ja rahaa. Kannattaa kuitenkin ottaa huomioon, että kaikki alustat eivät ole saman tasoisia vaikka niistä parhailla onkin tiettyjä yhteisiä piirteitä. Sopivaa alustaa etsittäessä keskeisinä valintakriteereinä tulisi pitää datan käsittely- ja yhdistämismahdollisuuksia, pilvipohjaista arkkitehtuuria, järjestelmäriippumatonta tukea, joustavuutta ja laajennettavuutta sekä alustalle valmiiksi koodattujen komponenttien määrää ja laatua. (Salesforce, n.d.)

Jotta low-code -työkalulla kyettäisiin täyttämään kaikki kaavaillut tarpeet ja mahdollisesti myös korvaamaan jo käytössä olevia ratkaisuja, on ensisijaisen tärkeää selvittää, mitä low-code -alustalta odotetaan ja mihin tarkoitukseen sitä tullaan käyttämään. Samalla tulee pohtia minkätyyppisiä sovelluksia yritys on aikeissa kehittää, koska alustat eroavat toisistaan melkoisesti. On suositeltavampaa sijoittaa monipuoliseen työkaluun, jolla pystyy ratkomaan vaihtelevia haasteita nyt ja tulevaisuudessa kuin sitoutua vain yhden käyttöalueen alustaan ja näin rajoittaa muutostarpeisiin reagointia. Lisäksi tietoturvallisuuteen tulee perehtyä huolellisesti ja varmistaa, että ainakin käyttäjähallinta ja käyttäjän määrittelemän tiedon suojaaminen on ongelmatonta. Niin ikään olennaista on tutustua tarjolla oleviin tukipalveluihin ja alustan hinnoitteluun sekä erityisesti käytön perusteella perittäviin maksuihin, vaikkei hintaa tulekaan tule pitää ainoana valintaperusteena. (Rakshit, 2020)

Nykyisin varsinkin pilvipalvelujen hinnoittelu perustuu useimmiten joko kapasiteettiin tai käyttäjämäärään. Tämänkaltaisen hinnoittelumallin etuina voidaan pitää edullisempia aloituskustannuksia ja olematonta ylläpitotarvetta. Yrityksen tarvitsemien ohjelmistojen kokonaishintaa voi kuitenkin olla vaikea arvioida ja Capterran yhdysvaltalaisille pk-yrityksille vuonna 2019 tekemän tutkimuksen mukaan suurin este ohjelmistojen hankinnalle onkin pelko budjetin ylittymisestä. Hinta on harvoin ilmoitettu yksiselitteisesti ja lisäksi hinnoitteluperusteet vaihtelevat toimittajittain, mikä tekee vertailusta hankalaa. (Westfall, 2020) Alustojen lisenssimallit on monesti rakennettu niin, että sovelluskohtainen hinta

laskee alustan käytön lisääntyessä. Näin ollen yksittäisen tai toisaalta massiiviselle käyttäjämäärälle suunnatun sovelluksen toteutus voi olla kannattamatonta. (Kotilainen, 2019)

Gartnerin raportissa (Wong et al., 2021) neuvotaan valintaa tehdessä keskittymään erityisesti alustan arkkitehtuuriin ja tekniikkaan, koska alustan avulla tulisi pystyä ratkaisemaan keskeiset liiketoimintaongelmat, integroitumaan ulkoisiin toimittajiin ja vastaamaan tulevaisuuden haasteisiin. Alustoja vertaillessa on ajateltava myös niiden avoimuutta, skaalautuvuutta ja kuormituksen kestoa. Kotilainen (2019) tähdentää vielä, että monen pelkäämää riskiä lukittumisesta tiettyyn alustaan voidaan pienentää suosimalla kerroksellisuutta ja komponenttiajattelua.

Valintatilanteessa ei liioin kannata unohtaa ulkonäköseikkoja, joiden vaikutus käyttäjiin voi olla yllättävän suuri. Ammattimaisen kuvan antavaa ja yrityksen näköistä käyttöliittymää on niin työntekijöiden kuin asiakkaidenkin mukava käyttää tehokkaasti. Vaikka juuri sen oikean low-code -toimittajan löytäminen tarjonnan paljoudesta saattaa ajoittain tuntua ylivoimaiselta, maksaa valintaan käytetty aika itsensä varmasti takaisin. (Enright, n.d.)

2.8 Low-code -alustat

Melkoisen laajalla low-code -termillä voidaan viitata eri asioihin ja sen alta löytyy useita erilaisia tuotteita ja alustoja. Kaikille yhteinen ajatusmaailma on visuaalinen mallintaminen koodin käsinkirjoittamisen sijaan, mutta osa työkaluista soveltuu paremmin kevyiden yritysten sisäisten järjestelmien kehitykseen ja osa taas isompien järjestelmien ja digitaalisten palvelujen kehitykseen. Joitakin alustoja voi käyttää minkä tahansa liiketoiminnan apuna, mutta jotkut on selkeästi suunnattu vain tiettyjen alojen järjestelmäkehitykseen. Alustoilla työskentelykään ei ole täysin samanlaista ja vaatii tuotteesta riippuen käyttäjältä erilaista osaamista. (Wennström, 2021) Alla esitellään tarkemmin kolme erityyppistä low-code -alustaa, joilla opinnäytetyön toiminnallinen osuus toteutettiin.

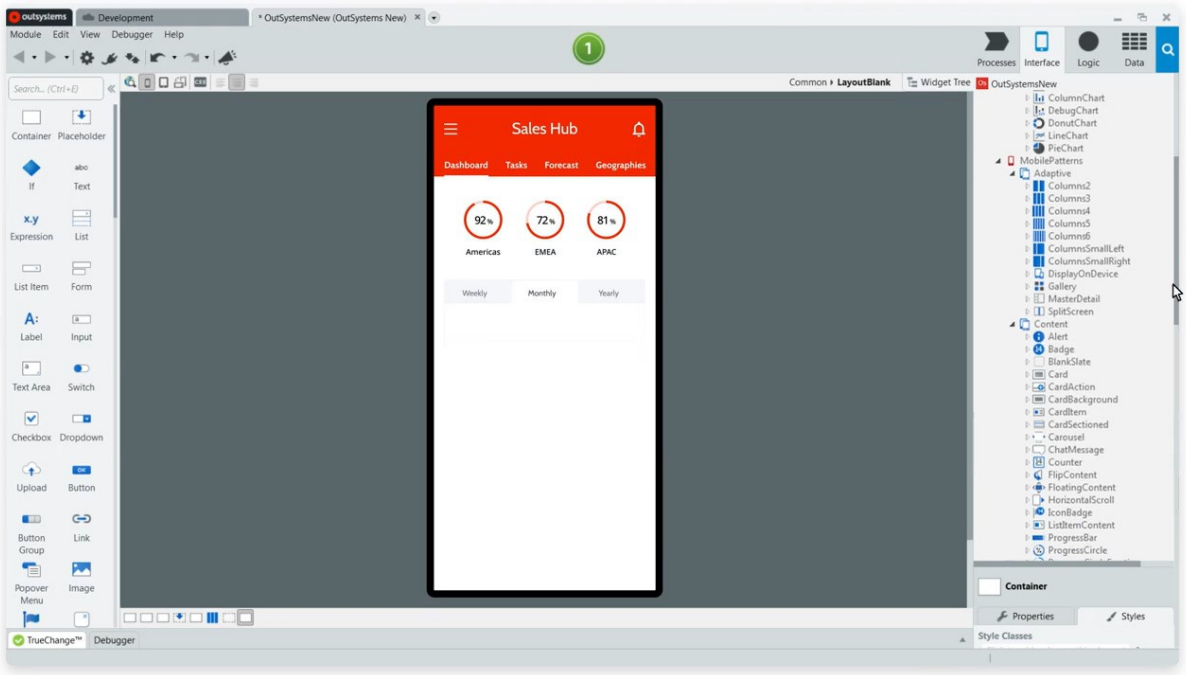
2.8.1 Outsystems

Outsystems on nopea ja tehokas työkalu yritysten reaktiivisten web- ja mobiilisovellusten kehitykseen ja tuotantoon sekä niiden hallinnointiin ja analysointiin. Se sopii sekä pienten osastokohtaisten että suurten monimutkaisten sovellusten rakentamiseen. Avoin alusta on helppo integroida olemassa oleviin yritysjärjestelmiin ja ulkoisiin tietokantoihin. Koska alustaa ei ole lukittu, voidaan sillä tehtyjä sovelluksia suorittaa tai päivittää myös muilla työkaluilla tai alustoilla. Outsystemsillä on tarjolla 2 GB:n tietokannan sisältävä ilmainen versio yrityksen sisäiseen käyttöön. Laajempi käyttö vaatii erillisen yrityssopimuksen, jolloin kuukausihinnat alkavat 4000 dollarista. (OutSystems, n.d.)

Sovellusten käyttöliittymä, toimintalogiikka ja tietomallit luodaan visuaalisesti raahaamalla ja pudottamalla suunnittelunäkymään valmiita elementtejä (kuva 1), joita voidaan rajattomasti laajentaa lisäämällä JavaScript-, Java-, C#-, SQL-, CSS- tai HTML-koodia käsin. Alusta generoi taustalla optimoidun fullstack-koodin käyttäen muun muassa HTML-, CSS-, JavaScript- ja .NET-kieliä, mikä varmistaa sovellusten virheettömyyden. Viimeisimpiä standardeja ja teknologioita hyödyntävä ReactJS-pohjainen sovellus voidaan tietoturvallisesti julkaista yhdellä napin painalluksella joko pilvessä tai yrityksen omassa ympäristössä. Sovellusten suorittamiseen alustan sovelluspalvelin käyttää perinteisiä tietokantoja ja ulkoisia järjestelmiä. Outsystems myös ylläpitää arkistoa avoimen lähdekoodin laajennuksista, joita voidaan ottaa käyttöön kaikissa sovelluksissa. (OutSystems, n.d.)

Outsystems on julkaistu ensimmäisen kerran Portugalissa vuonna 2001 ja se on nykyisin yksi low-code -työkaluja tarjoavista markkinajohtajista. Maailmanlaajuisella yhtiöllä on yli 1500 työntekijää ja asiakkaina useita suuryrityksiä. Outsystems tarjoaa erittäin kattavan ja monipuolisen maksuttoman online-kurssimateriaalin, jonka eri osioista on mahdollista suorittaa maksullisia sertifikaatteja työelämän tarpeita ajatellen. Kehittäjäyhteisössä on yli 60 000 jäsentä ja käyttäjäfoorumin tarjoama tuki on vahva. (OutSystems, n.d.)

Kuva 1. Outsystems-alustan suunnittelunäkymä (OutSystems, n.d.)



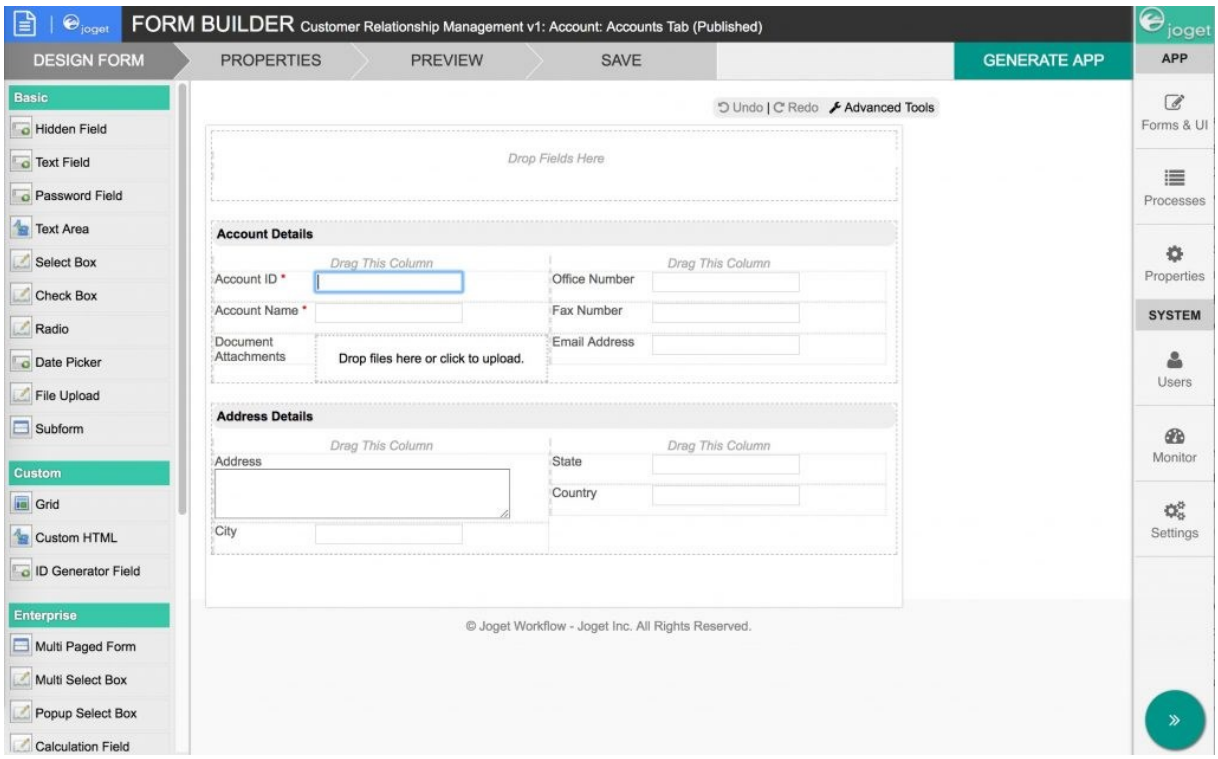
2.8.2 Joget DX

Joget DX on seuraavan sukupolven avoimeen lähdekoodiin perustuva low-code -alusta, jossa prosessiautomaation, työnkulun hallinnan ja low-code -sovelluskehityksen parhaat puolet yhdistyvät yksinkertaisella ja joustavalla tavalla. Nimeen lisätty DX kuvaa alustan digitalisoitumisen tueksi tarjoamia kattavia ominaisuuksia. Sen verkkopohjainen käyttöliittymä tarjoaa sekä ammattikoodareille että koodia taitamattomille mahdollisuuden rakentaa ja ylläpitää sovelluksia missä tahansa. Alusta tukee täydellisiä end-to-end -sovelluksia eikä rajoitu prosessi- tai datakeskeisiin projekteihin. Sen painopisteet ovat progressiivisissa verkkosovelluksissa ja käyttökokemuksessa, DevOpsissa ja sovellusten suorituskyvyn hallinnassa sekä tekoälyssä. Sovellukset luodaan raahaamalla ja pudottamalla elementtejä suunnittelunäkymään (kuva 2) ja rakentamisessa on mahdollista hyödyntää sekä valmiita pohjia että erilaisia käyttäjäteemoja. Alustaan saumattomasti yhdistyvältä Joget Marketplacelta voi lisäksi ladata käyttöönsä lisäsovelluksia ja liitännäisiä (plugin). Java-pohjainen Joget DX pyörii niin yksityisessä kuin julkisessakin pilvessä tai yrityksen omalla palvelimella eikä se ole käyttöjärjestelmä-, tietokanta- tai sovelluspalvelin-riippuvainen. Julkaistut sovellukset toimivat sekä selaimella että mobiilissa. (Joget, n.d.)

Joget DX -alustasta on tarjolla eri versioita yrityksen kokoon suhteutettuna ja niiden hinnoittelu on porrastettu sekä käyttäjämäärän että käyttötavan mukaan. Omalla serverillä pyöritettäessä alustan käyttäjäkohtaiset vuosihinnat alkavat 80 eurosta, kun taas sovellusalusta palveluna (aPaaS) kustantaa yhteensä 5 käyttäjälle halvimmillaan 10 euroa kuussa. Tässä pienille ryhmille tarkoitetussa mallissa on varattu 1 GB tilaa käyttäjää kohti. (Joget, n.d.)

Joget on aloittanut automatisoidun työnkulun moottorina Yhdysvalloissa vuonna 2009 ja julkaissut ensimmäisen täysimittaisen sovelluskehitysalustan vuonna 2011. Yli 400 yrityskäyttäjän joukkoon mahtuu suuri kirjo niin valtion virastoja kuin suuryhtiöitä ja pk-yrityksiä. Jogetilla on sivuillaan lyhyitä opetusvideoita ja kattavat ohjeet sekä ilmainen online-kurssimateriaali. Käyttäjäyhteisössä on yli 12 000 jäsentä, joilta saa tukea oppimiseen ja alustan käytön omaksumiseen. (Joget, n.d.)

Kuva 2. Jodet DX -alustan suunnittelunäkymä (Joget, n.d.)



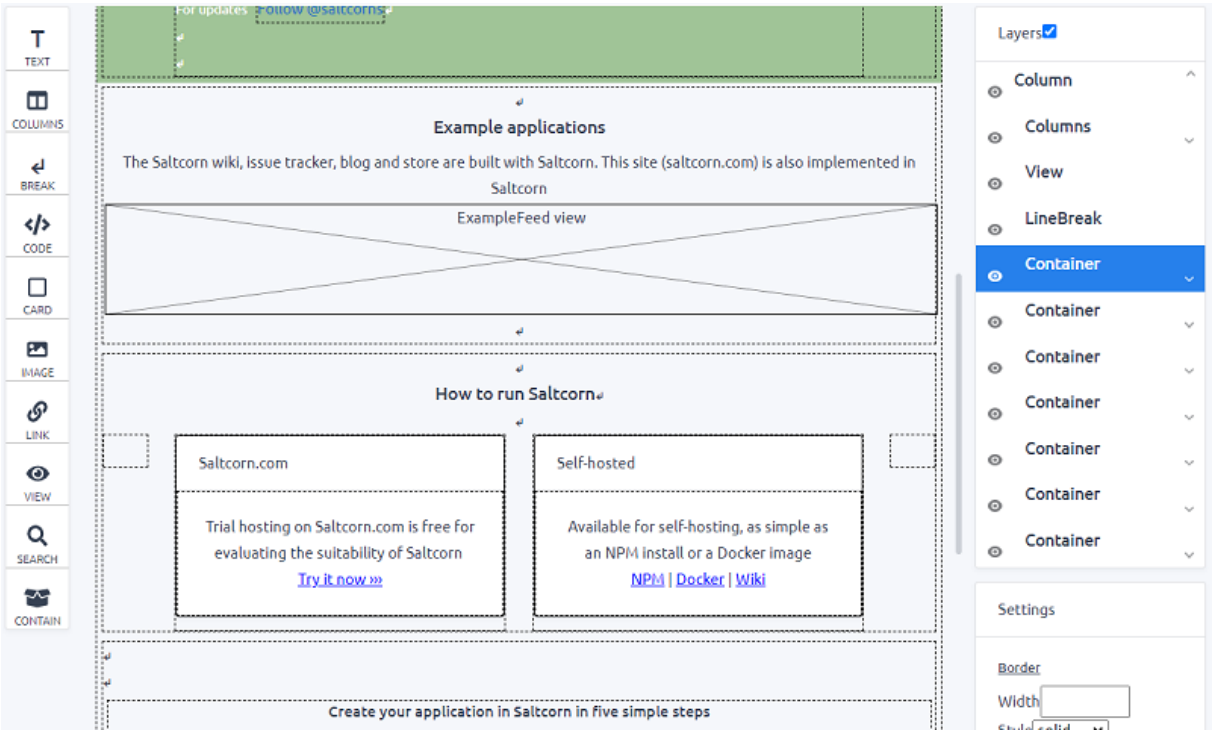
2.8.3 Saltcorn

Saltcorn on avoimen lähdekoodin no-code -alusta, joka sopii varsinkin relaatiotietoihin perustuvien tietokantasovellusten rakentamiseen. Sillä voidaan joustavasti muokata tietotyyppejä ja web-sovelluksen ulkonäköä. Alustan käyttö ja testisovelluksen julkaiseminen on ilmaista. Ohjelman esiasennettu versio on mahdollista ottaa yhdellä klikkauksella käyttöön DigitalOcean-pilvipalvelussa, jossa perustasoinen 1 GB:n virtuaaliserveri maksaa 5 dollaria kuussa. Vaativampaan käyttöön suositellaan kuitenkin ohjelman lataamista omalle koneelle ja sovellusten tallentamista omavalintaiseen web-hotelliin. (GitHub, 2021)

Saltcornissa sama ohjelma huolehtii sekä serveri- että selainpuolesta ja alustan voidaan ajatella olevan jäsennellyn tiedon sisällönhallintajärjestelmä. Hyvin suunnitellun tietokannan luominen onnistuu web-sovelluksen rakentamisen yhteydessä ja tietokanta on tarpeen vaatiessa siirrettävissä muihin järjestelmiin. Alusta käyttää relaatiomallin perustuvaa PostgreSQL avoimen lähdekoodin SQL-tietokantaa. Tietokantaan tallennettua dataa näytetään erilaisten esimääriteltyjen näkymien avulla, joita lisätään sovellukseen raahaa ja pudota -menetelmällä (kuva 3). Näin toimien voidaan toteuttaa monimutkaisiakin käyttöliittymiä kirjoittamatta lainkaan koodia. Sovellusten rakentamista on helpotettu ja nopeutettu eri elementeistä valmiiksi kootuilla paketeilla (packs) ja lisäksi alustan toimintoja on mahdollista laajentaa erityisillä javascriptillä koodatuilla liitännäisillä (plugin). Sovellusten ulkoasut perustuvat teemoihin, jotka voi vaivatta vaihtaa toiseen ja näin hetkessä muuttaa ilmettä. Alusta tarjoaa useita esimerkkisovelluksia kuten osoitekirjan, tehtävälistan, projektinhallintajärjestelmän, wikin ja raportointinäkymän. (Saltcorn, n.d.)

Alusta on julkaistu vuonna 2020 eikä se vielä sovellu mobiilisovellusten toteuttamiseen. Sillä luodut web-sovellukset ovat kuitenkin reaktiivisia ja toimivat teemasta riippuen myös mobiilissa. Usean sivuston hallinnointi kerralla on mahdollista multitenancy-tilassa. Avoimen lähdekoodin ohjelmana Saltcorn käyttää sallivaa MIT-lisenssiä eikä sillä ole takuuta. Turvallisuusasioihin suhtaudutaan vakavasti muun muassa etsimällä haavoittuvuuksia ja tietoturva-aukkoja. Alustan kehittäminen on edelleen kesken, mikä näkyy selvästi roadmapista. Tehtyjä päivityksiä voi seurata alustan blogista ja twitteristä. Käyttäjätukena toimii alustan sivustolta löytyvien suppeiden ohjeiden lisäksi käyttäjäfoorumi. (Saltcorn, n.d.)

Kuva 3. Saltcorn-alustan suunnittelunäkymä (Saltcorn, n.d.)



3 Kehittämistyön tavoite ja menetelmät

Opinnäytetyön aihe syntyi toimeksiantajan tarpeesta tehostaa laskutuskäytäntöjään ja kehittämistyönä toteutettavalla tietokantapohjaisella web-sovelluksella pyritään sekä sujuvoittamaan merkittävästi toimeksiantajan nykyistä käsityönä tehtävää excel-pohjaista laskutusprosessia että nopeuttamaan ja laajentamaan raportointia sekä edistämään tilausten ja tuotemenekin seurantaa. Oppimisprosessin tavoitteena on vahvistaa, syventää ja täydentää tekijän nykyistä osaamista sekä hankkia uusia taitoja ammatillisen kasvun tueksi.

Opinnäytetyön toiminnallisessa osuudessa suunnitellaan, toteutetaan ja käyttöönotetaan toimeksiantajan sisäiseen käyttöön yksinkertainen web-sovellus. Se tallentaa työntekijän lomakkeella syöttämän datan tietokantaan, josta on mahdollista laskutusta ja seurantaa varten tulostaa raportteja asiakkaittain, tuotteittain tai ajanjaksoittain eriteltynä. Lisäksi tietokantaan tallennettuja tuotteita, asiakkaita ja toimituksia voidaan päivittää, poistaa tai lisätä. Lopputuloksena syntyvä reaktiivinen web-sovellus skaalautuu eri päätelaitteille, joten erillistä mobiilisovellusta ei ole tarpeen luoda.

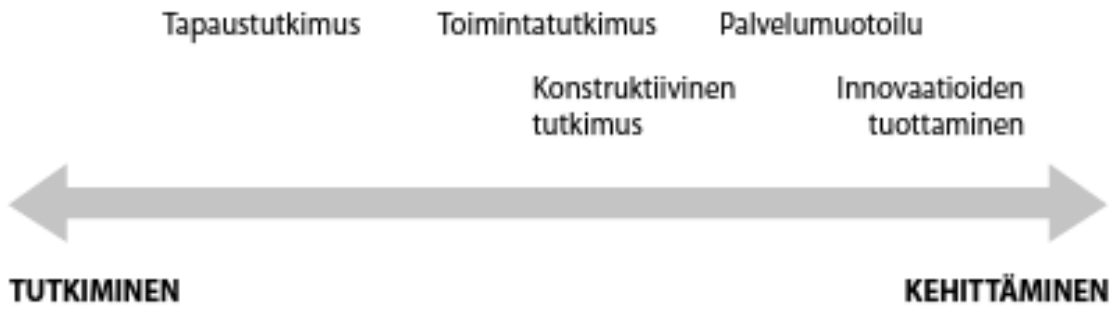
Kehittämisprojektissa hyödynnetään ketteriä menetelmiä ja kanbania. Sovellus rakennetaan kolmella eri low-code -alustalla, joita lopuksi vertaillaan SWOT-analyysin avulla. Erityisesti tarkastellaan alustojen toiminnallisuutta, käyttökokemusta, hinnoittelua ja syntynyttä lopputulosta. Pyrkimyksenä on löytää toimeksiantajan tarpeisiin mahdollisimman tarkasti sopiva ja kustannustehokas alusta.

3.1 Konstruktiivinen tutkimusmenetelmä

Kehittämisprojektin menetelmänä käytetään konstruktiivista tutkimusmenetelmää, koska projektin tavoitteena on tuottaa toimeksiantajan käyttöön uusi sovellus ja haluttu lopputulos on ennalta tiedossa. Kehittämisen kohteena on yrityksen laskutusprosessi ja lopputuloksena syntyy sekä projektisuunnitelma että käyttöönotettava web-sovellus, joka muuttaa ja parantaa vanhaa käytäntöä. Ojasalo et al. (2015, s. 66) kirjoittaa konstruktiivisen tutkimuksen olevan sopiva lähestymistapa, kun tehtävänä on saada aikaan konkreettinen tuotos. Lähestymistavalla pyritään muuttamaan organisaation toimintaa ja ratkaisemaan

aito käytännön ongelma teoreettista tietämystä hyödyntäen. Lähestymistavan valitsemista havainnollistetaan kuvassa 4.

Kuva 4. Lähestymistavan valitseminen (Ojasalo et al., 2015, s.36)



Konstruktiivinen tutkimus pyrkii ratkaisemaan ongelmia käytännönläheisesti ja luomaan uusia rakenteita sekä teoreettisen tiedon että käytännöstä kerättävän tiedon avulla. Lähestymistapa muistuttaa innovaatioiden tuottamista ja palvelumuotoilua, mutta niistä poiketen tuloksena syntyneitä rakenteita arvioidaan käytännön hyödyn perusteella. Konstruktiivinen tutkimus pitää sisällään suunnittelua, käsitteellistä mallintamista sekä mallien toteutusta ja testaamista. Vuorovaikutusta ja kommunikaatiota korostetaan ja ratkaisun laatimiseen otetaan aktiivisesti mukaan käytännön toimijat, joita tässä tapauksessa ovat toimeksiantajayrityksen työntekijät. (Ojasalo et al., 2015, s. 65)

Kehittämistyössä resursseja on tärkeä suunnata varsinkin muuutoksen toteuttamiseen, jotta tavoite saavutetaan. Oleellinen osa prosessia on säännöllinen raportointi, jonka avulla sekä kuvataan kehittämistyötä että viedään sitä eteenpäin. Kirjallinen raportointi auttaa jäsentämään ajatuksia, synnyttämään keskustelua ja antamaan palautetta. (Ojasalo et al., 2015, s. 25)

3.2 Ketterät menetelmät ja kanban

Ketterät menetelmät pitävät sisällään useita menetelmiä, joiden avulla on mahdollista sekä ajatella ja työskennellä tehokkaammin että tehdä parempia päätöksiä. Ne keskittyvät

sovelluskehityksen eri alueisiin, mutta kaikille tärkeä yhteinen tavoite on pyrkiä muuttamaan kehitystiimin ajattelutapaa. (Stellman & Greene, 2014, s. 2)

Ketterissä menetelmissä dokumentointia ei tehdä yksityiskohtaisesti ja se keskittyykin lähinnä järjestelmän ydinalueisiin. Vähäistä dokumentointia korvataan asiakkaan kanssa käydyillä keskusteluilla, jotka ovat elintärkeitä projektin onnistumiselle ja aikataulussa pysymiselle. Asiakas kertoo, mitä järjestelmältä odottaa ja vaatimukset kirjataan käyttäjätarinoiksi. Käyttäjätarina (user story) kuvaa yhden järjestelmän liiketoiminnallista arvoa tuottavan toiminnallisuuden ymmärrettävästi lausemuodossa. Isossa roolissa on myös toiminnallisuuksien priorisointi tärkeysjärjestykseen usein niin, että eniten liiketoiminnallista arvoa tuottava on ylimpänä. Priorisointia toistetaan koko kehitysprosessin ajan, jotta se pysyy ajan tasalla kun asiakkaan vaatimukset lisääntyvät tai muuttuvat. Dokumentoinnin tukena voidaan käyttää mallintamista, jonka tarkoituksena on toimia suunnittelun ohjenuorana. (Paetsch et al., 2003)

Yritys saa ketteriä menetelmiä suosimalla low-codesta parhaan hyödyn irti, koska silloin nopea päätöksenteko yhdistyy nopeaan sovelluskehitykseen. Low-coden vauhdikas tahti vaatii sitoutumista uusien lähestymistapojen jatkuvaan kehittämiseen kaikissa projektin vaiheissa. Aiemmin sovelletut, tutut käytännöt eivät välttämättä toimi low-codessa yhtä tehokkaasti ja niitä tulisikin pyrkiä muokkaamaan, mikä valitettavan usein jää kuitenkin tekemättä. (Huff, 2019)

Ketteristä menetelmistä esitellään tarkemmin vain projektimalliksi valittu kanban-menetelmä, joka on alunperin prosessin parannusmenetelmä. Sen ydinperiaatteita noudattaen voidaan nykyistä toimintatapaa kaiken aikaa kehittää ja sujuvoittaa. Periaatteisiin kuuluvat työnkulun ja eri vaiheiden visualisointi, työn alla olevien tehtävien rajoittaminen, tehtävien läpimenoajan mittaus, prosessikäytäntöjen havainnollistaminen, prosessin jatkuva kehittäminen ja valmiiden mallien käyttö uusien kehitysmahdollisuuksien tunnistamiseen. (Stellman & Greene, 2014, s. 315 – 316)

Yksinkertaisena menetelmänä kanban soveltuu erinomaisesti ohjelmistoprojekteihin ja etenkin ylläpitovaiheen töihin Juvonen (2018) kertoo kirjassaan. Työn kulku visualisoidaan kuvassa 5 esitetyn kaltaisella kanban-taululla, jonka yksiselitteisesti nimetyissä sarakkeissa

tehtävät liikkuvat etenemisen mukaan. Samanaikaista työtä rajoitetaan eikä uusia tehtäviä aloiteta ennen kuin keskeneräiset on saatu valmiiksi. Kanban-taulun avulla kaikki voivat hahmottaa tehtävät selkeästi, seurata työjonoa tai tehdä siihen muutoksia. Juvonen viittaa tutkimukseen projektin onnistumistodennäköisyydestä (Standish Group, 2015) ja kehottaa suosimaan ketteriä menetelmiä. Toteutustapa on tärkeä valita huolella siitä huolimatta, että esimerkiksi pienitöisessä yhden hengen projektissa toteutusmenetelmällä ei käytännössä olekaan väliä.

Kuva 5. Esimerkki kanban-taulusta, bugifiksi tarkoittaa ohjelmointivirheen korjausta ja implementointi varsinaista ohjelmointityötä (Juvonen, 2018)

Myös Ikosen (2011) väitöskirjatutkimuksessa on todettu, että ohjelmistokehitys hyötyy virtaviivaisuuteen perustuvan kanban-menetelmän visuaalisuudesta, intuitiivisuudesta ja yksinkertaisuudesta. Menetelmän avulla pystytään sekä säästämään resursseja, aikaa ja kustannuksia että lisäämään työmotivaatiota ja kommunikaatiota. Lisäksi ongelmanratkaisu ja päätöksenteko tehostuu sekä turha työ ja vastoinkäymiset vähenevät. Kokonaisvaltaiseen ohjelmistoprojektin hallitsemiseen kanban ei yksinään riitä, mutta sen avulla resursseja voidaan organisoida joustavammin kuin esimerkiksi scrum-mallissa.

Kanban sopii erinomaisesti nimenomaan low-code -sovelluskehitykseen, koska siinä keskitytään nopeaan julkaisutahtiin, reaktiivisuuteen ja järjestelmällisyyteen. Menetelmän avulla low-code -sovelluskehitystä voidaan nopeuttaa entisestään ja se onkin yleensä paras valinta, kun etusijalla on tuotteen saaminen pikaisesti markkinoille. Lisäksi aikaa vapautuu suunnitteluun, kehittämiseen, testaukseen ja käyttäjäpalautteeseen, sillä dokumentointi on

karsittu minimiin. Kanbanin joustavuus mahdollistaa muutokset missä tahansa projektin vaiheessa ja varsinkin kokeneet tiimit hyötyvät sisäisestä yhteistyöstä. Menetelmän ansiosta työntekijöiden osallistuvuus ja itseohjautuvuus kasvaa sekä tehtävien valmistuminen tehostuu. (Huff, 2020)

3.3 SWOT-analyysi

SWOT- eli nelikenttäanalyysi on yksinkertainen ja hyödyllinen työkalu, jolla voidaan selvittää tutkittavan kohteen nykyisiä vahvuuksia ja heikkouksia sekä arvioida tulevaisuuden mahdollisuuksia ja uhkia. Analyysia on mahdollista käyttää useaan eri tarkoitukseen ja se soveltuu esimerkiksi jonkin tuotteen tai palvelun vertailuun, kunhan tulosten vertailukelpoisuuden takaamiseksi muistetaan rajata tarkasti mitä arvioidaan. (Lindroos & Lohivesi, 2010)

SWOT-lyhenne tulee englannin kielen sanoista strengths, weaknesses, opportunities ja threats, joiden suomenkieliset vastineet ovat vahvuudet, heikkoudet, mahdollisuudet ja uhat. Näistä vahvuudet ja heikkoudet ovat sisältä tulevia asioita, kun taas uhat ja mahdollisuudet tulevat ulkopuolelta (kuva 6). Analysoitavaa kohdetta tarkastellaan kaikkien osalta ja kirjataan havainnot ne kunkin otsikon alle. (Visma, 2017)

Kuva 6. SWOT-analyysin kentät (Rousku, 2013)

3.4 Toimeksiantaja

Opinnäytetyön toimeksiantaja on pääkaupunkiseudulla toimiva siivousalan pk-yritys Fresh House Oy, joka on perustettu vuonna 1991. Perheyritys on keskittynyt toimistosiivoukseen, ylläpitosiivoukseen ja niihin liittyviin erikoissiivouspalveluihin. Sen kautta toimeksiantajan asiakas saa halutessaan tilattua kaikki puhtaanapidon palvelut ja tuotteet yhdestä osoitteesta. Yrityksen palveluksessa on suoraan 40-50 työntekijää ja lisäksi se työllistää useamman alihankkijan.

4 Web-sovelluksen suunnittelu ja toteutus low-code -työkaluilla

Toteutettavalla tietokantapohjaisella web-sovelluksella pyrittiin parantamaan ja sujuvoittamaan toimeksiantajan nykyistä laskutuskäytäntöjä ja ratkaisemaan low-code -työkalujen avulla useita tähän asti lähes pelkästään käsityönä excel-taulukoiden avulla tehdyn prosessin ongelmakohtia kuten erityyppisten raporttien tulostus, tuotehintojen ajantasaisuus ja päivittäminen sekä laskutettavien tuotteiden ja kulutettujen tarvikkeiden menekin seuranta.

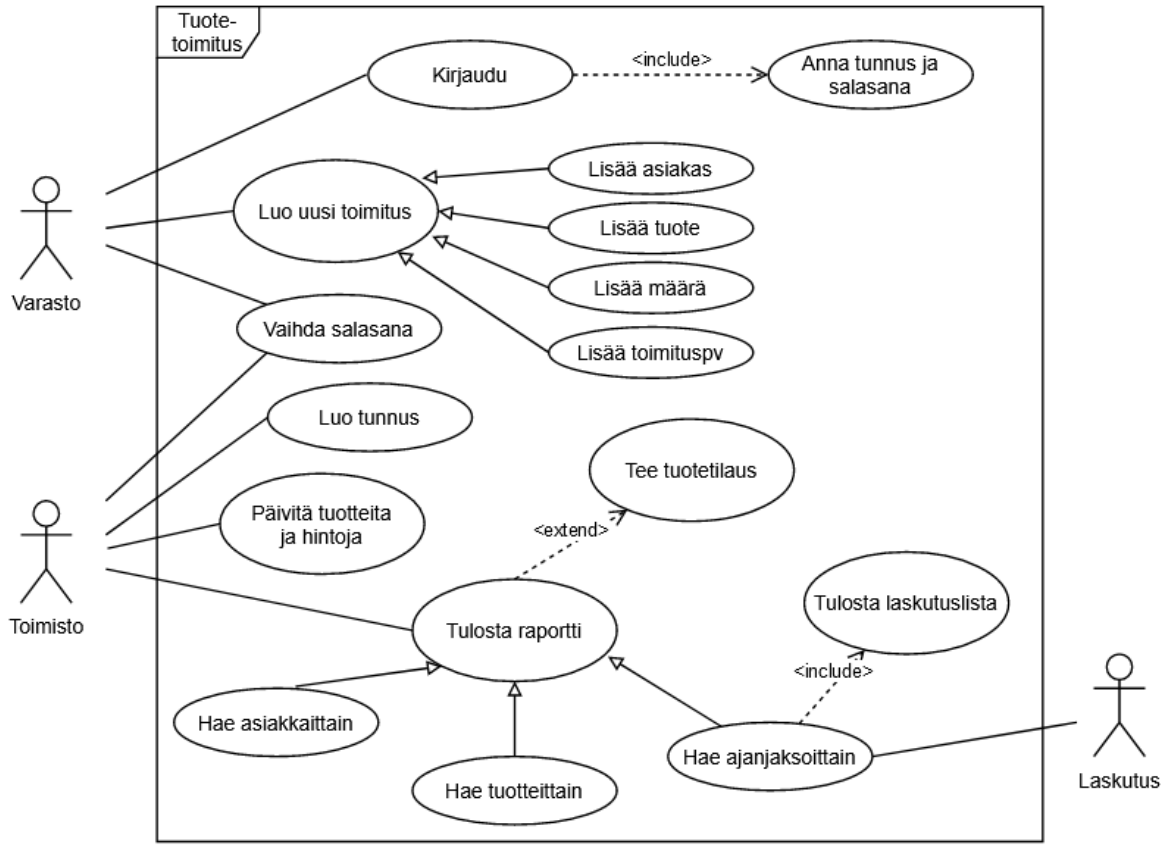
Kehittämistyö käynnistettiin keskustelemalla toimeksiantajan kanssa nykyisen laskutusprosessin toiminnasta ja sen parannustarpeista sekä kehittämistoiveista. Keskustelun pohjalta päädyttiin tietokantapohjaiseen ratkaisuun muun muassa sen tarjoamien erinomaisten hakuominaisuuksien vuoksi, joita pystyttiin hyödyntämään monipuolisesti raporttien tulostuksessa. Web-sovellus päätettiin rakentaa kolmella eri low-code -alustalla toiminnoiltaan mahdollisimman identtiseksi. Toimeksiantaja annettiin testata kaikkia sovelluksia, jonka jälkeen niistä laaditun vertailun perusteella valittiin toimeksiantajalle sopivin vaihtoehto. Ketterän sovelluskehityksen periaatteita noudattaen toimeksiantajalta pyydettiin kommentteja toistuvasti projektin kuluessa, jotta esitetyt kehitystoiveet oli mahdollista huomioida sovelluksessa sekä kehityksen ja toteutuksen aikana että vielä ennen käyttöönottoa.

4.1 Sovelluksen ja käyttöliittymän suunnittelu

Sovelluksen suunnittelu aloitettiin laatimalla yhteistyössä toimeksiantajan kanssa pelkistetty vaatimusmäärittely, jonka tukena käytettiin mallintamista visualisoimaan sovelluksen toimintaa. Toimeksiantajan sovellukselta toivomat ominaisuudet kirjoitettiin käyttäjätarinoiden muotoon kuvaamaan selkeästi vaatimuksia ja toiminnallisuuksia (liite 2). Käyttäjätarinoista käytiin toimeksiantajan kanssa keskusteluja, joiden perusteella ne pystyttiin priorisoimaan hyödyllisyysjärjestykseen. Toimeksiantajan palautteen perusteella käyttäjätarinoita ja priorisointeja myös tarkistettiin ja osin muutettiin prosessin edetessä, mihin ketterä kanban-menetelmä soveltui hyvin.

Sovelluksen toiminnallisuuksia havainnollistettiin UML-mallinnuksen avulla ja piirrettiin kuvassa 7 esitetty käyttötapauskaavio, jossa kuvataan käyttäjien sovelluksen avulla suorittamia tehtäviä.

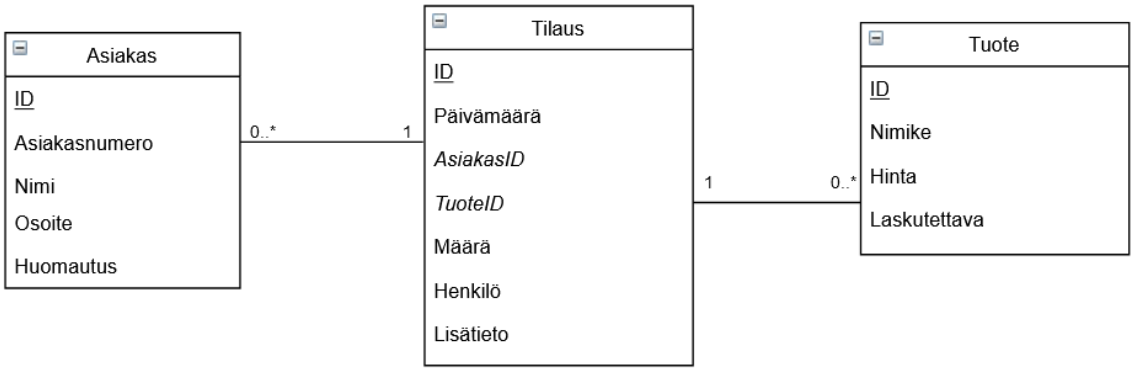
Kuva 7. Sovelluksen käyttötapauskaavio



Sovelluksen käyttöliittymästä pyrittiin tekemään mahdollisimman selkeä ja helppokäyttöinen ja sen ruuduista piirrettiin suunnittelutyökalulla karkeat, suuntaa-antavat luonnokset (liite 3). Luonnokset lähetettiin toimeksiantajalle, jonka kanssa niistä keskusteltiin ja saadun palautteen perusteella tehtiin muutamia lisäyksiä ja parannuksia.

Tietokannan rakenteesta piirrettiin kuvan 8 relaatiokaavio, johon on määritetty taulut ja attribuutit sekä niiden väliset yhteydet. Kussakin taulussa on alleviivattu pääavain, jonka perusteella käsitteet ja niiden attribuutit pystytään yksilöimään.

Kuva 8. Tietokannan relaatiokaavio



4.2 Sovelluksen toteutus

Low-code -alustojen tarjonta on tänä päivänä runsasta ja valinnan tekeminen oli haastavaa. Tätä korosti se, että toteutuksen kohteena oli pieni sovellus yrityksen sisäiseen käyttöön jonka kustannukset haluttiin pitää maltillisina. Lisäksi toteuttamisvaiheessa havaittiin, ettei kaikkien vaatimusten toteuttaminen onnistunut alustavasti valituilla ratkaisuilla niiden puutteellisten elementtien ja heikon muokattavuuden vuoksi. Toivotuista toiminnallisuuksista ei kuitenkaan haluttu luopua ja hyödyntämällä useimpien alustojen tarjoamaa ilmaista kokeilujaksoa saatiin vertailtua ja testailtua erityyppisiä työkaluja, mihin toki kului melkoisesti aikaa.

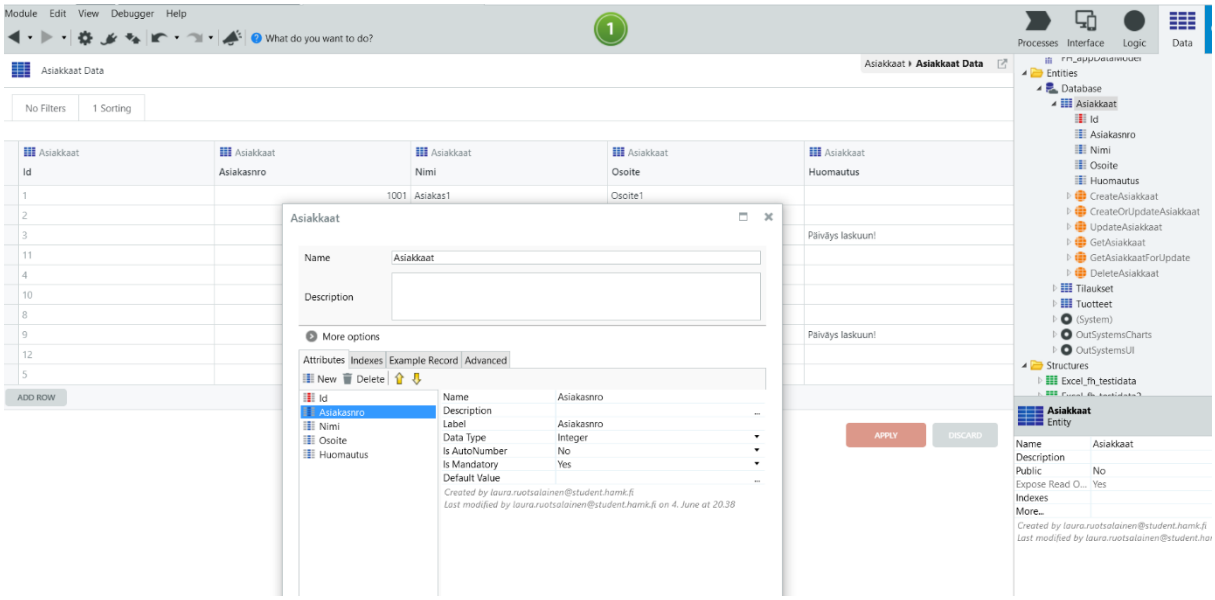
Lopulta opinnäytetyössä käytettäviksi low-code -alustoiksi valikoituivat Outsystems, Joget DX ja Saltcorn, joilla kaikilla toiminnoiltaan mahdollisimman identtisen sovelluksen rakentaminen onnistui. Täysin identtiseen ulkonäköön ei pyritty, eikä se näillä alustoilla olisi ollut mahdollistakaan. Outsystems oli tekijälle ennalta jonkin verran tuttu, mutta muut olivat täysin vieraita. Outsystems toimi lähes rajattomine hyödyntämismahdollisuuksineen hyvänä vertailupohjana, vaikkakin jo etukäteen oli tiedossa ettei sovelluksen toteuttamiseen tarvita kuin murto-osa sen tarjoamista toiminnoista.

Seuraavassa esitellään pelkistetysti ja pääpiirteittäin sovelluksen visuaalinen koodausprosessi edellä mainittuja low-code -alustoja käyttäen pyrkimättä kuvaamaan sen tarkemmin kaikkia alustojen tarjoamia ominaisuuksia ja mahdollisuuksia.

4.2.1 Toteutus Outsystems-alustalla

Outsystems-käyttäjäksi rekisteröitymisen jälkeen omalle koneelle ladattiin alustan käyttöä varten Service Studio -ympäristö. Uuden web-sovelluksen kehitys aloitettiin luomalla taulut Data-näkymän Entities-kansioon, jossa niitä voidaan muokata kuten kuvassa 9 näkyvän Asiakkaat-taulun Asiakasnumero-kenttää. Muun muassa kentän tietotyyppi, pituus, pakollisuus ja oletusarvo voidaan määritellä jokaiselle erikseen sekä lisätä tai poistaa kenttiä. Alusta luo automaattisesti juoksevasti numeroidun Id-avainkentän ja taulukon kenttien muokkaamiseen perustoiminnot Luo uusi (Create), Muokkaa (Update), Hae (Get) ja Poista (Delete). Taulujen tietoja päästiin tarvittaessa muokkaamaan taulunäkymässä ja taulujen väliset yhteydet saatiin näppärästi perustettua Entity Diagrams-kohdassa asettamalla lapsitaulun tietotyypiksi äititaulun Id.

Kuva 9. Outsystems-alustan tietokantataulut ja asiakas-taulun kentät

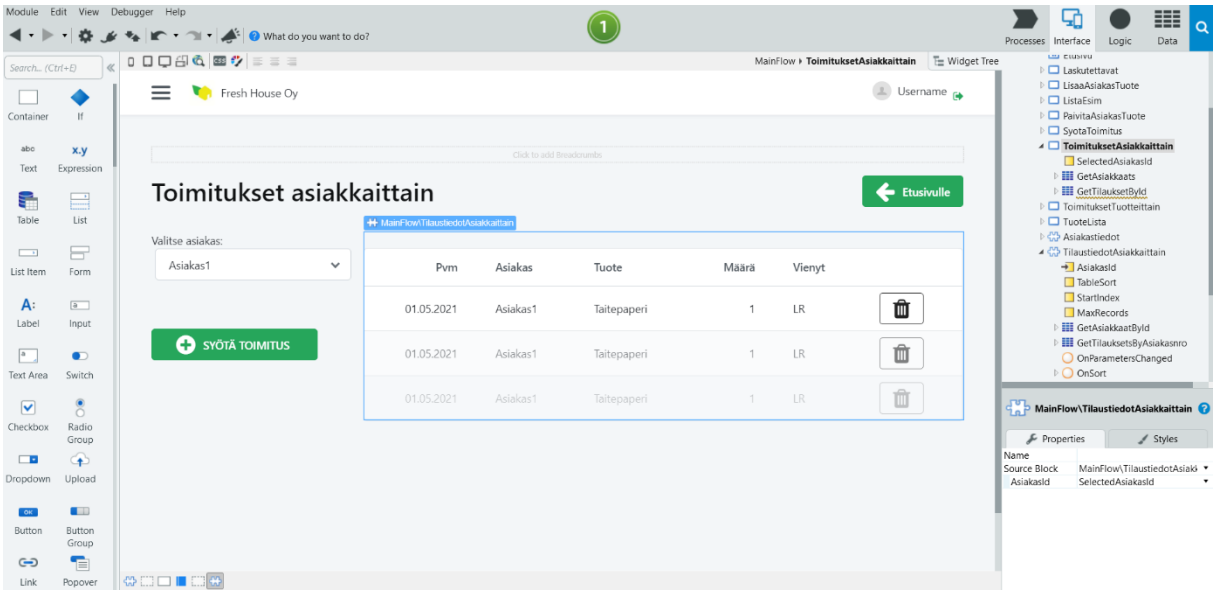


Interface-välilehdellä luotiin käyttäjänäkymät (Screen), joiden pohjana voidaan käyttää jotain lukuisista valmiista malleista tai aloittaa tyhjästä. Näkymien käyttöoikeudet myönnettiin Roles-valinnan alla rekisteröidyille käyttäjille (Registered), jotka olisi ollut tarvittaessa mahdollista jakaa myös alaryhmiin tai myöntää oikeudet kaikille (Anonymous). Eri näkymät koodattiin visuaalisesti raahaten ja pudottaen suunnittelunäkymään tauluja ja elementtejä, joita olisi ollut mahdollista muokata lähes loputtomasti omaa css-koodia kirjoittamalla. Näkymiä pystyi tarkastelemaan selaimessa julkaisemalla ne sivun yläreunan vihreällä

1-Click-Publish -napilla, jolloin alusta teki sekä virhetarkastuksen että tallensi samalla uuden version sovelluksesta. Aiempiin versioihin oli mahdollista palata versiohistorian kautta ja myös pikanäppäinkomento Ctrl + Z (Undo) toimi alustalla.

Useassa Outsystems-näkymässä hyödynnettiin block-elementtejä, joiden avulla samaa koodinpätkää pystyttiin toistamaan kirjoittamatta sitä joka kerta uudelleen. Esimerkiksi kuvassa 10 näkyvä block-elementti määriteltiin näyttämään valitulle asiakkaalle tehdyt toimitukset ja se sijoitettiin sekä raporttinäkymään että uuden toimituksen syöttönäkymään.

Kuva 10. Block-elementtiä hyödyntävä raporttinäkymä

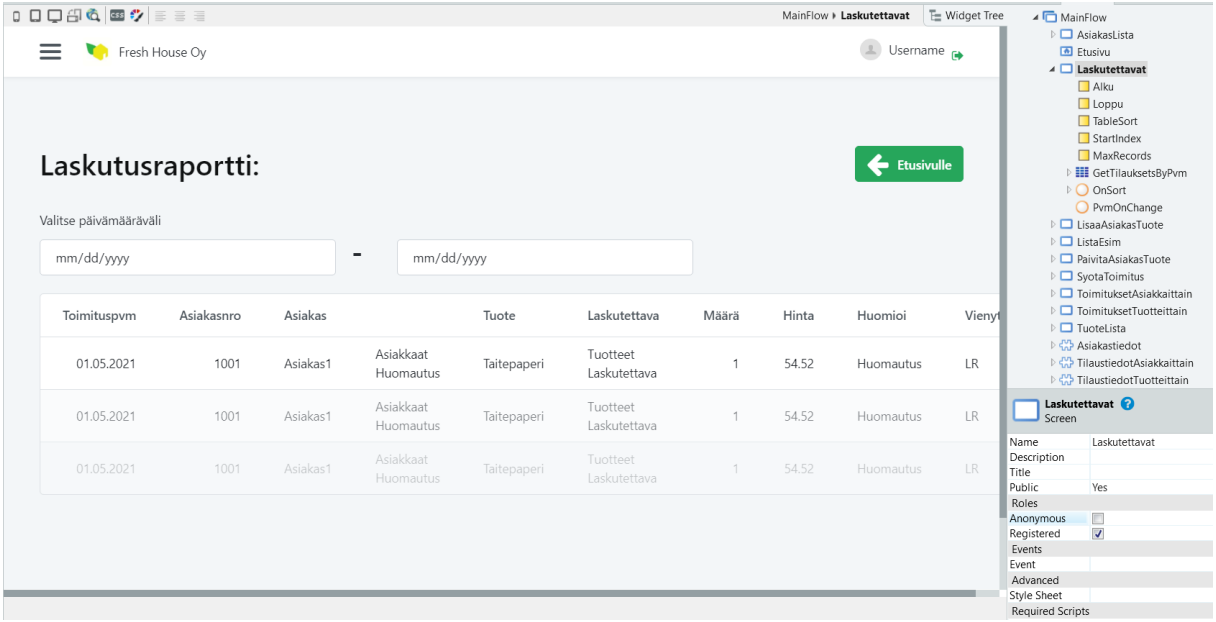


Alusta loi automaattisesti useita tarpeellisia toimintoja kuten kuvassa 11, jossa on vedetty Tilaus-taulu suunnittelunäkymään ja alusta on luonut automaattisesti sen kentistä listan sekä lisännyt lajittelua varten OnSort-toimintalogiikan. Toki toimintalogiikan määrittelyjä joutui lisäämään jonkin verran vielä käsin, mutta alustan huolehtiessa peruskoodauksesta säästyi aikaa melkoisesti.

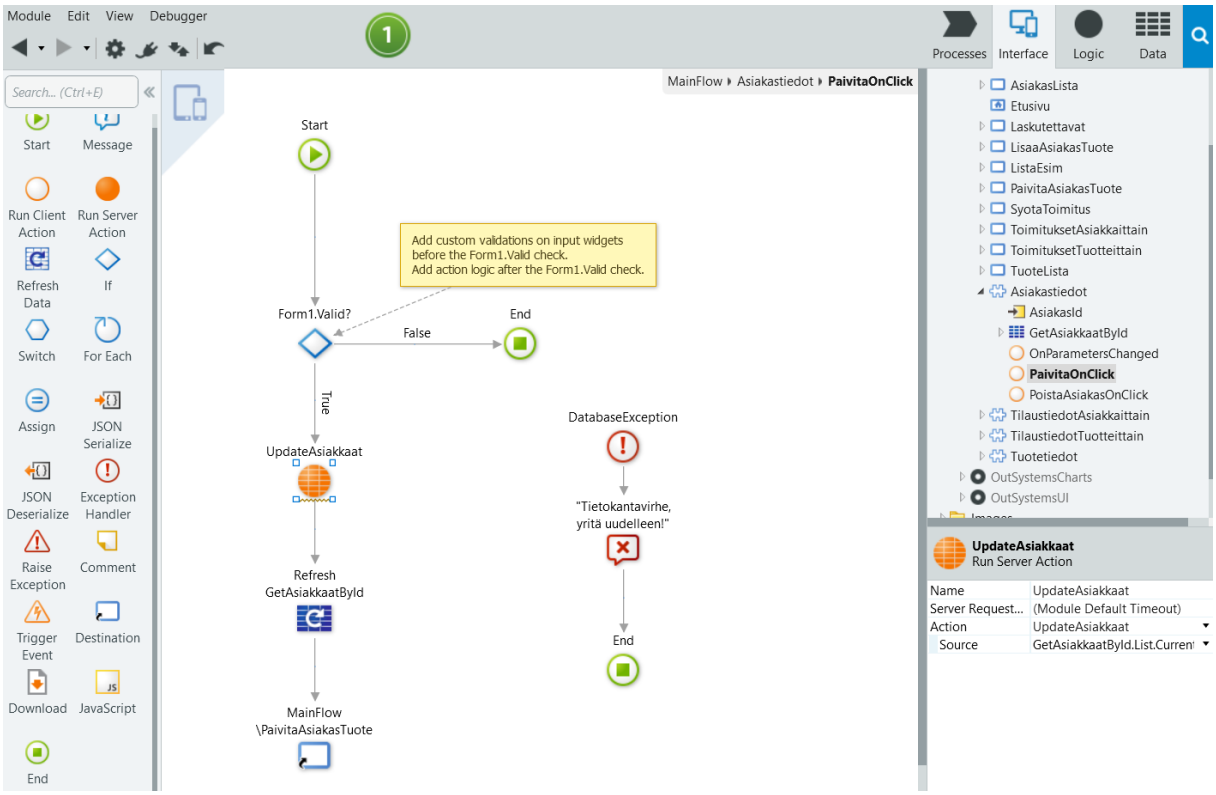
Toimintalogiikoiden määrittely ja rakentaminen vaati jo hieman syvempää perehtymistä varsinkin monimutkaisempien kaavioiden kohdalla. Toimintalogiikat luotiin samalla raahaa ja pudota -periaatteella vuokaaviomaiseen tyyliin, jossa jokaisella polulla oli oltava alku ja yksi tai useampi loppu. Esimerkiksi kuvassa 12 on määritelty Päivitä asiakastiedot -napin toimintalogiikka valmiiden elementtien avulla ensin tarkastamaan lomakkeelle syötettyjen

tietojen kelvollisuus (Form1.Valid) ja vasta sen jälkeen päivittämään tietokanta (UpdateAsiakkaat) ja näyttämään päivitetyt tiedot (Refresh). Lisäksi tietokantavirheen (DatabaseException) varalta määriteltiin erillinen polku näyttämään virhesanoma.

Kuva 11. Tilaus-taulusta luotu listanäkymä

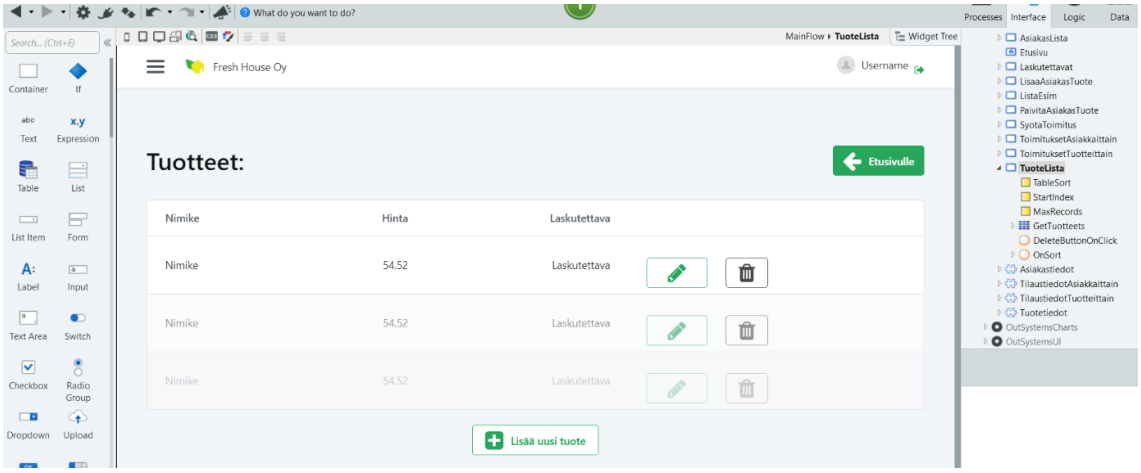


Kuva 12. Päivitä-napin toimintalogiikka

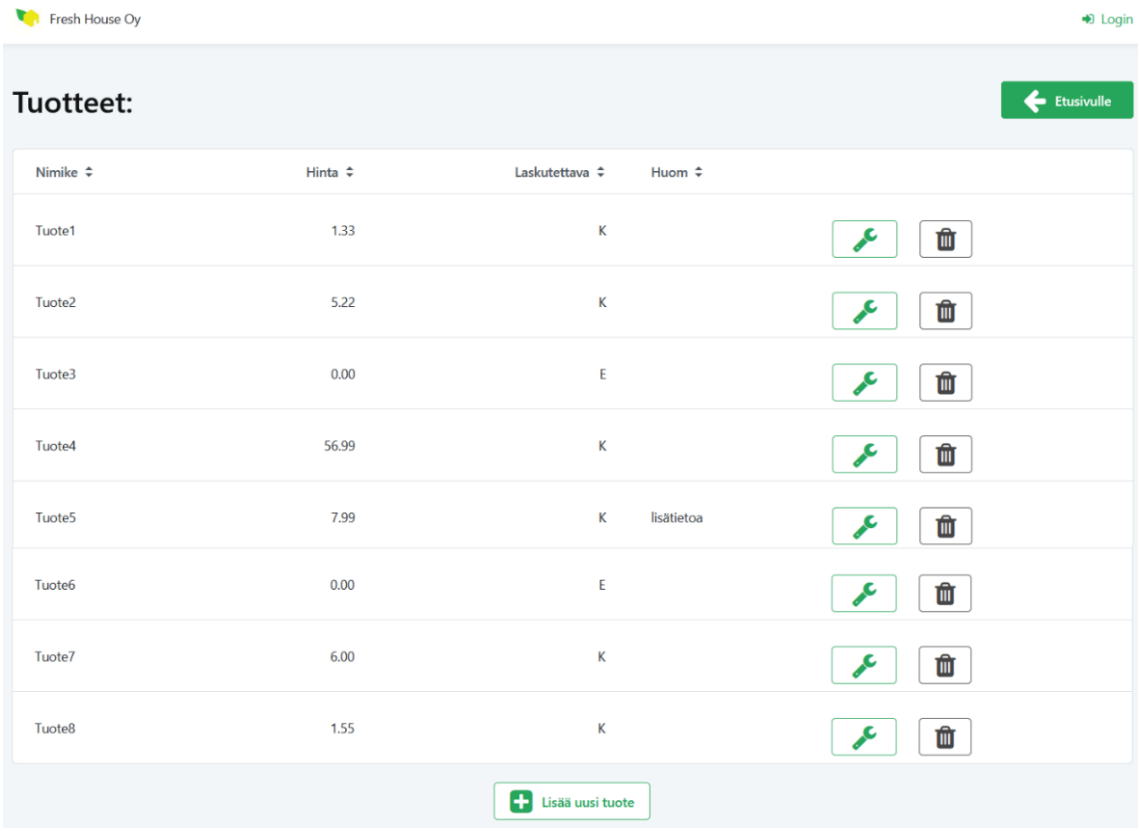


Outsystems tarjosi erittäin runsaasti monipuolisia elementtejä ja ominaisuuksia sekä käytännössä loputtomat muokkausmahdollisuudet, joiden avulla pikkutarkastikin räätälöityjen sovellusten rakentaminen tarvittaessa onnistuu. Suunnittelu- ja selainnäkymä olivat lähes identtiset, jonka voi todeta vertaamalla kuvia 13 ja 14. Outsystems-alustalla toteutetun sovelluksen kaikki käyttöliittymäruudut ovat liitteessä 4.

Kuva 13. Tuotelista-sivun suunnittelunäkymä



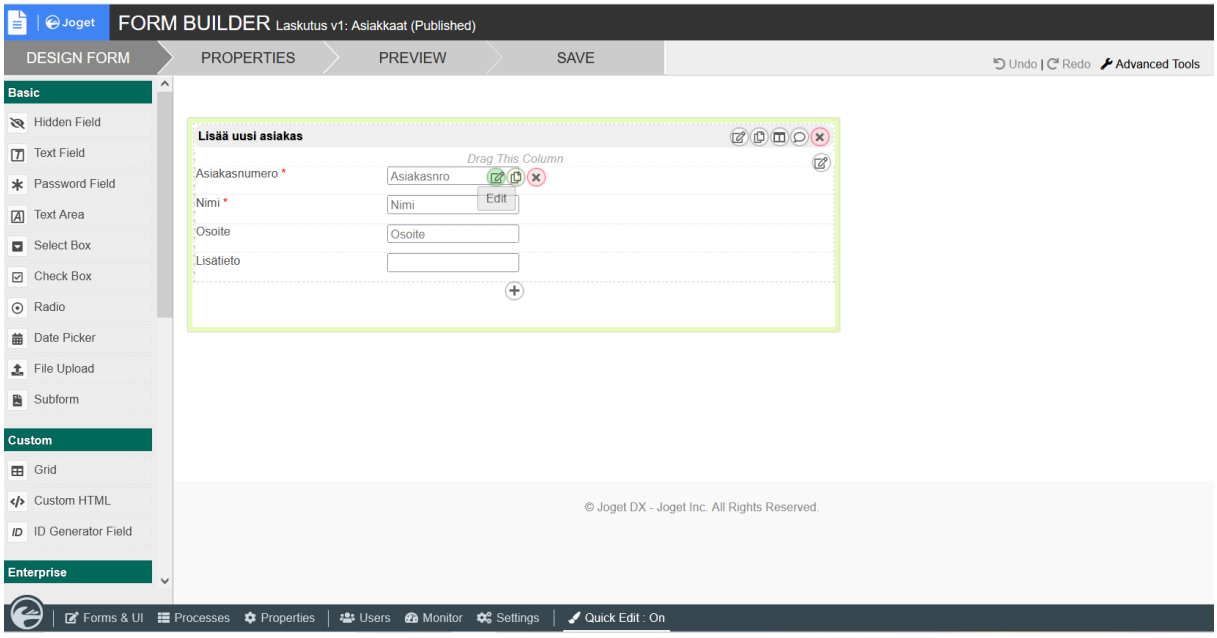
Kuva 14. Tuotelista-sivu selaimessa



4.2.2 Toteutus Joget DX-alustalla

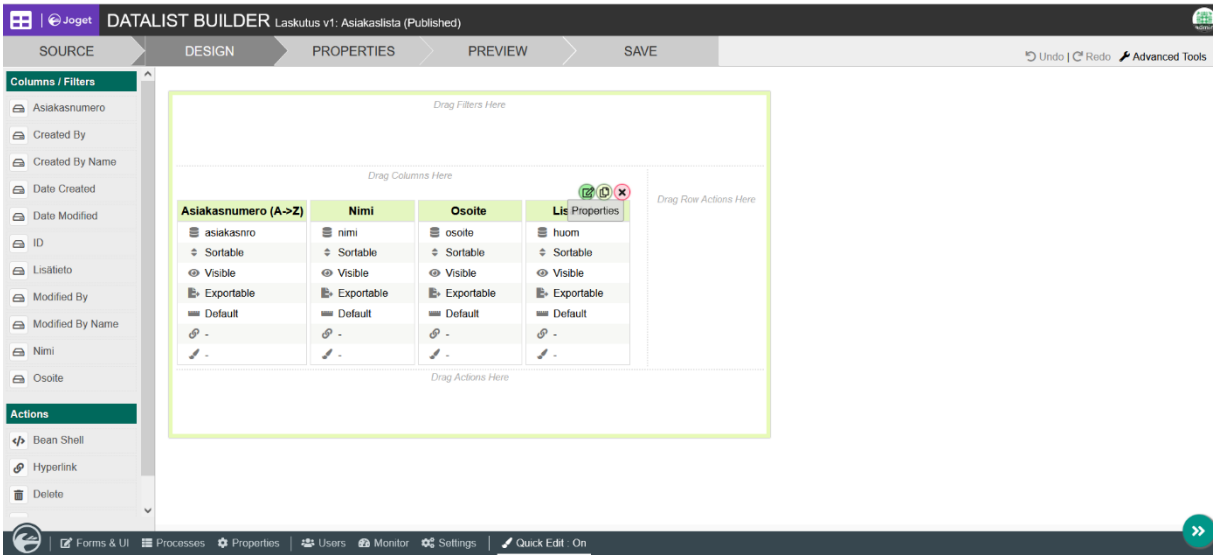
Sovelluksen toteutus aloitettiin rekisteröitymällä käyttäjäksi ja luomalla osoite xx.cloud.joget.com, jossa sijaitsevat kaikki saman yrityksen käyttäjien alustalla luomat sovellukset. Alusta toimii selaimessa ja sovelluksen rakentaminen aloitettiin lisäämällä ensimmäiseksi lomakkeet, joiden pohjalta tietokannan taulut automaattisesti alustan toimesta syntyivät. Lomakkeille raahattiin ja pudotettiin kuvan 15 kaltaisessa suunnittelunäkymässä elementtejä, joihin syötettävä tieto ja tyyppi määriteltiin tarkemmin erikseen tehtävillä valinnoilla. Kenttiin lisättiin täyttövihjeitä ja osa niistä määriteltiin pakolliseksi. Alusta loi automaattisesti lomakkeisiin Tallenna-napin ja sen toimintalogiikan, eikä näistä kumpaakaan päässyt itse muokkaamaan.

Kuva 15. Form-suunnittelunäkymä



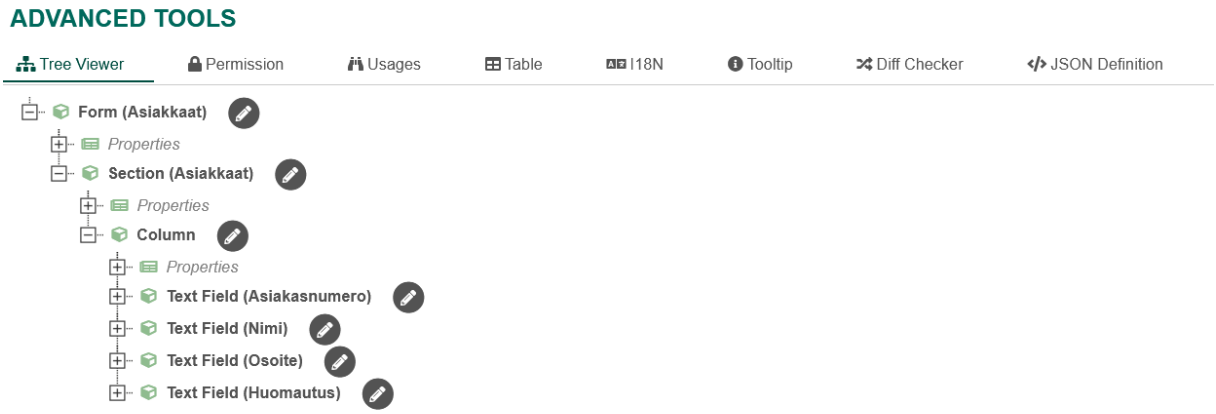
Tauluihin syötettyä dataa näytetään tietolistoilla (datalist), jotka luotiin kuvassa 16 esitellyssä suunnittelunäkymässä kenttä- ja toimintoelementtejä raahaten ja pudottaen. Elementtien oletusominaisuuksia pystyi muokkaamaan melko monipuolisesti ja lisäämään niihin omaa koodia ja css-määrittelyjä. Olemassa olevia listoja oli myös mahdollista kopioida uusien pohjaksi, mikä nopeutti samantyyppisten listojen tekoa. Eri taulujen tietojen pystyttiin näyttämään samalla listalla, kun source-välilehdellä luotiin listakohtaisesti yhteydet taulujen välille.

Kuva 16. Datalist-suunnittelunäkymä



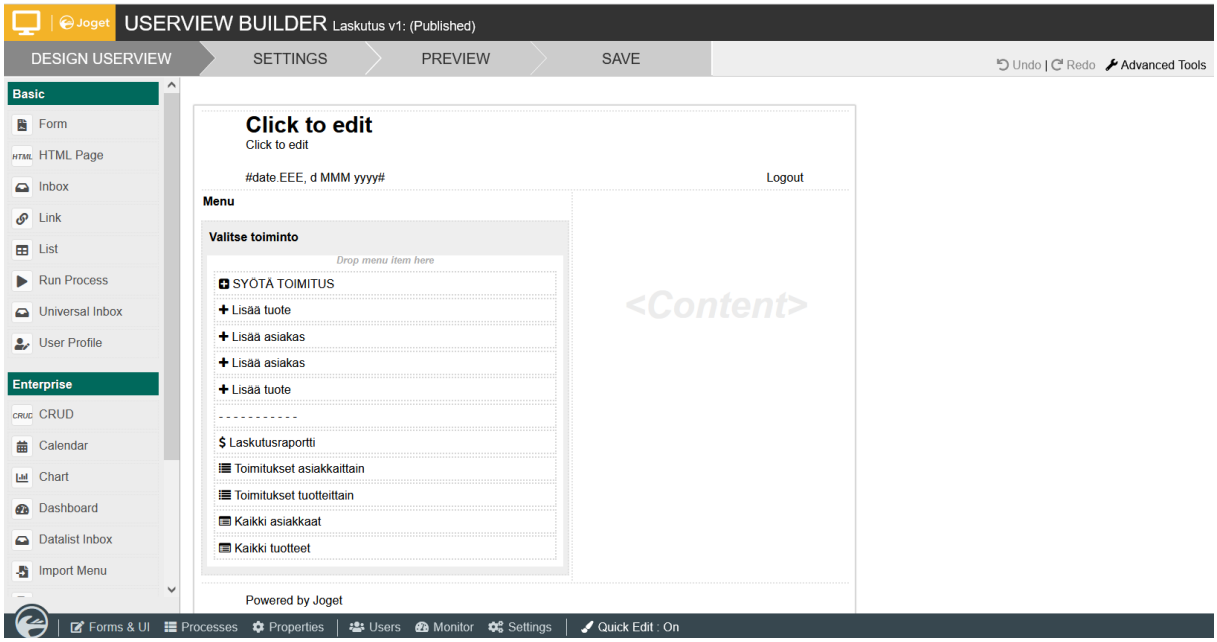
Lomakkeita ja listoja oli mahdollista käsitellä myös lisätyökalujen alta löytyvässä kuvan 17 puumallisessa näkymässä (Advanced Tools), jossa esimerkiksi kenttien päivittäminen oli mielestäni selkeämpää kuin suunnittelunäkymässä. Puunäkymässä pystyi myös tarkastelemaan ja muokkaamaan taustalla olevaa koodia sekä määrittelemään käyttöoikeuksia.

Kuva 17. Asiakas-lomakkeen puunäkymä



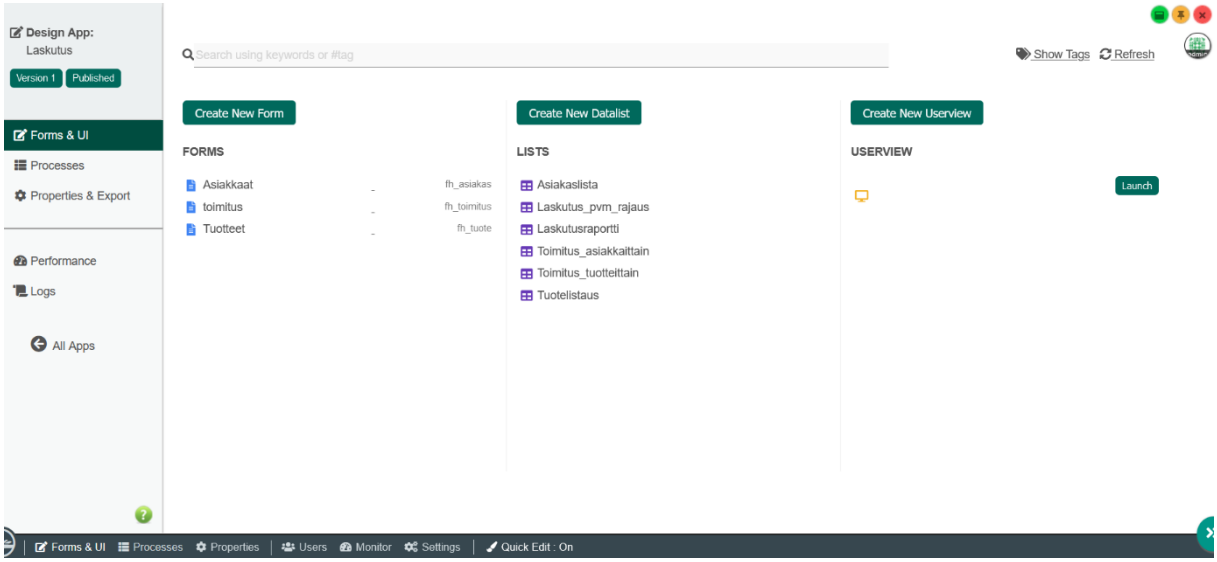
Viimeiseksi rakennettiin periaatteessa menu-muotoisena sivupalkkina toimiva käyttöliittymä, jonka kautta käyttäjät pääsevät käsiksi luotuihin lomakkeisiin ja listoihin. Elementtejä raahattiin ja pudotettiin kuvassa 18 näkyvään suunnittelunäkymään ja niitä oli mahdollista muokata melko monipuolisesti. Varsinkin tarjolla ollut CRUD-elementti nopeutti tietokantapohjaisessa sovelluksessa kehitystyötä melkoisesti ja teki tarpeettomaksi erilliset listoihin lisätyt tietokantatoiminnot. Käyttöliittymässä käytettiin pop-up -vahvistusikkunoita antamaan käyttäjälle lisätietoa, lisäämään käyttäjäystävällisyyttä ja pienentämään virheellisten toimien määrää.

Kuva 18. Userview-suunnittelunäkymä



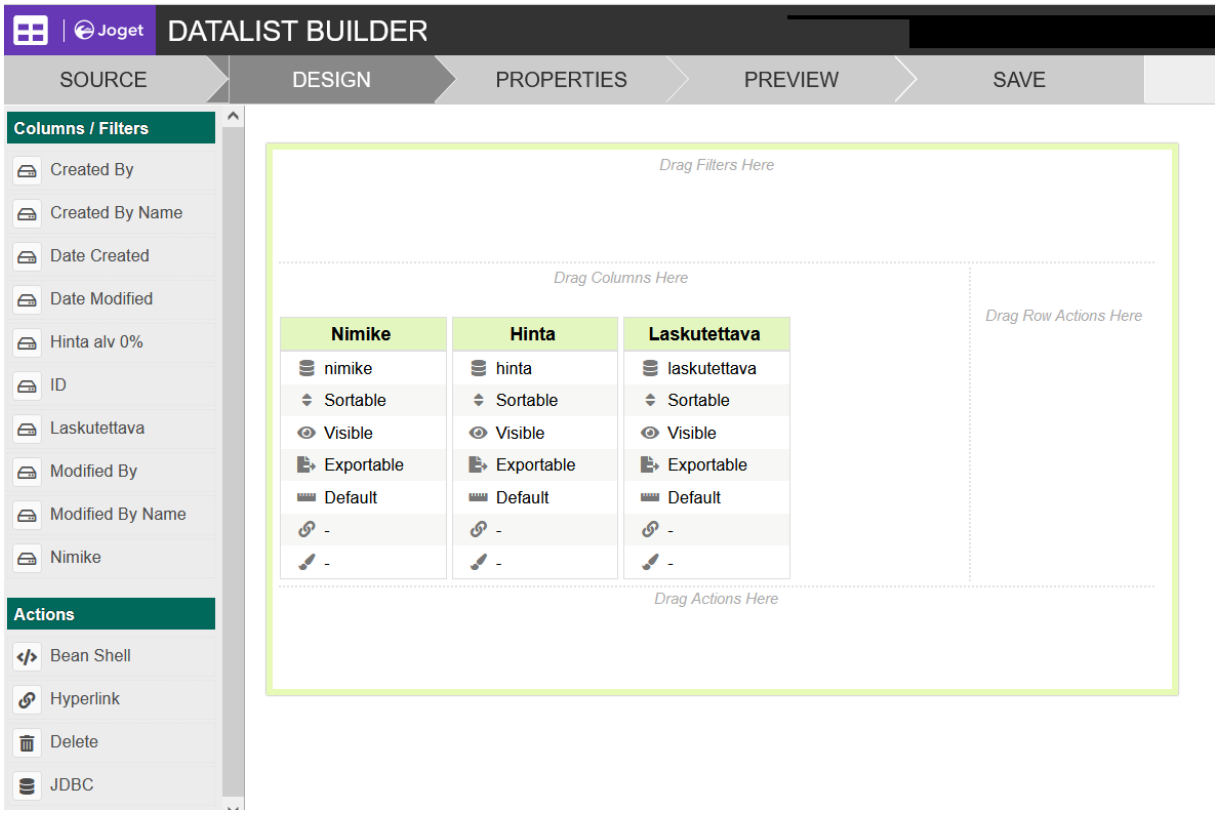
Valmis sovellus julkaistiin yhdellä napin painalluksella, jonka jälkeen sitä pystyi käyttämään selaimessa. Versiohistoria säilyy alustalla ja aiempiin versioihin on mahdollista tarvittaessa palata. Myös pikanäppäinkomento Ctrl + Z (Undo) toimii alustalla. Kaikki luodut sovelluksen osat näkyvät kootusti käyttäjän päänäkymässä (kuva 19) ja sivun alareunan Users-valikon kautta käyttäjien lisääminen ja hallinta on vaivatonta.

Kuva 19. Luodut lomakkeet, listat ja käyttäjänäkymät

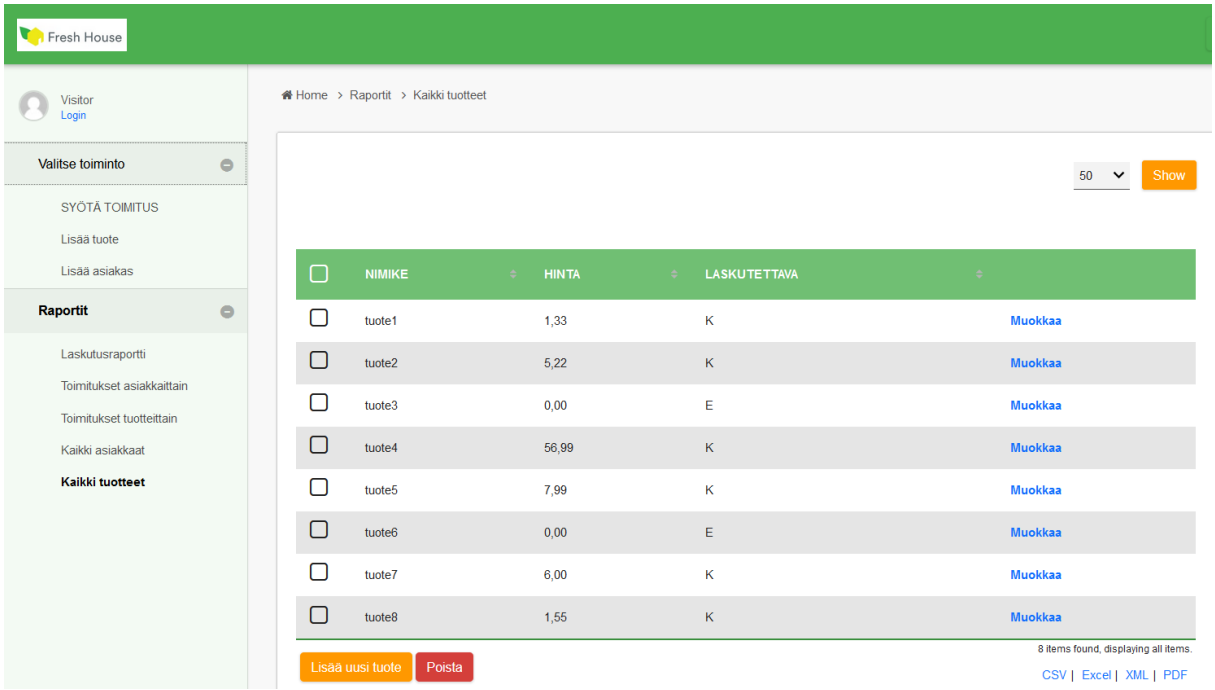


Joget DX -alustalla pystyy myös lisäämään prosesseja vuokaaviomuotoisena, mutta tätä toimintoa ei tässä sovelluksessa ollut tarvetta hyödyntää koska vakiomuotoiset toimintalogiikat olivat riittävät. Erinomaisena lisäominaisuutena alusta tarjosi mahdollisuuden tallentaa selaimessa näytetyt listat csv-, excel-, xml- tai pdf-muodossa, minkä ansiosta datan jatkokäsittely oli helppoa. Sen sijaan graafisia elementtejä ei käytännössä ollut tarjolla lainkaan. Suunnittelu- ja selainnäkymä eroavat jonkin verran toisistaan (kuvat 20 ja 21), koska osa määrittelyistä on tehty listanäkymään ja osa käyttöliittymään. Joget DX -alustalla toteutetun sovelluksen kaikki käyttöliittymäruudut ovat liitteessä 5.

Kuva 20. Tuotelista-sivun suunnittelunäkymä



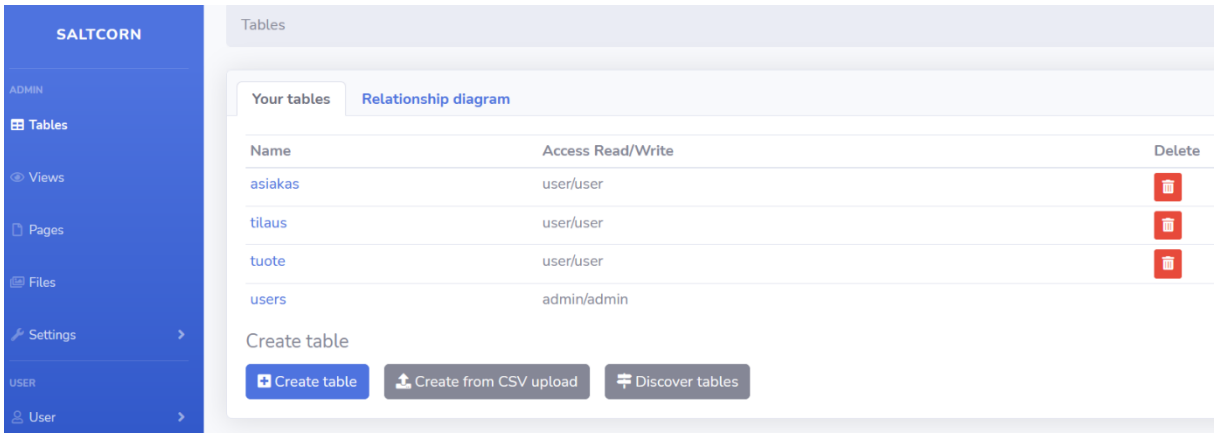
Kuva 21. Tuotelista-sivu selaimessa



4.2.3 Toteutus Saltcorn -alustalla

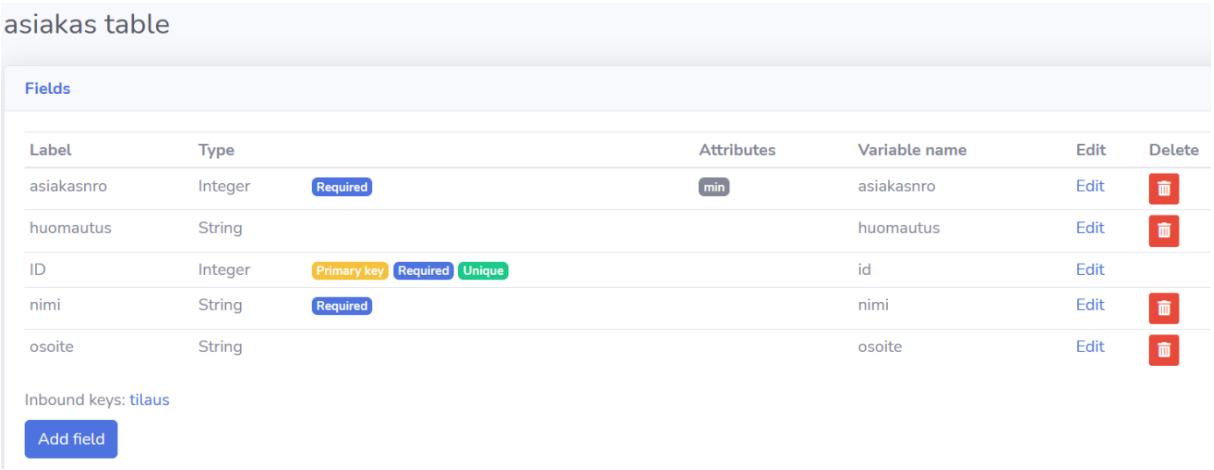
Käyttäjäksi rekisteröitymisen jälkeen sovellus luotiin alustan tarjoamaan ilmaiseen osoitteeseen xx.saltcorn.com, missä työkalua pystyttiin arvioimaan. Alustaa käytetään selaimessa ja oletuksena admin-roolilla luodaan taulut, näkymät ja hallinnoidaan muita käyttäjiä. Sovelluksen toteutus aloitettiin luomalla taulut kuvan 22 mukaisesti sivuvalikon kohdassa Tables ja määritettiin niille luku- ja kirjoitusoikeudet valmiiksi määriteltyjen käyttäjäroolien mukaisesti. Alustan oletuksena luomaa Users-taulua ei pystynyt muista tauluista poiketen poistamaan, mutta siihen oli mahdollista lisätä olemassa olevien admin-, user- ja public-roolien joukkoon tarpeen mukaan uusia.

Kuva 22. Saltcorn -alustalla luodut tietokantataulut



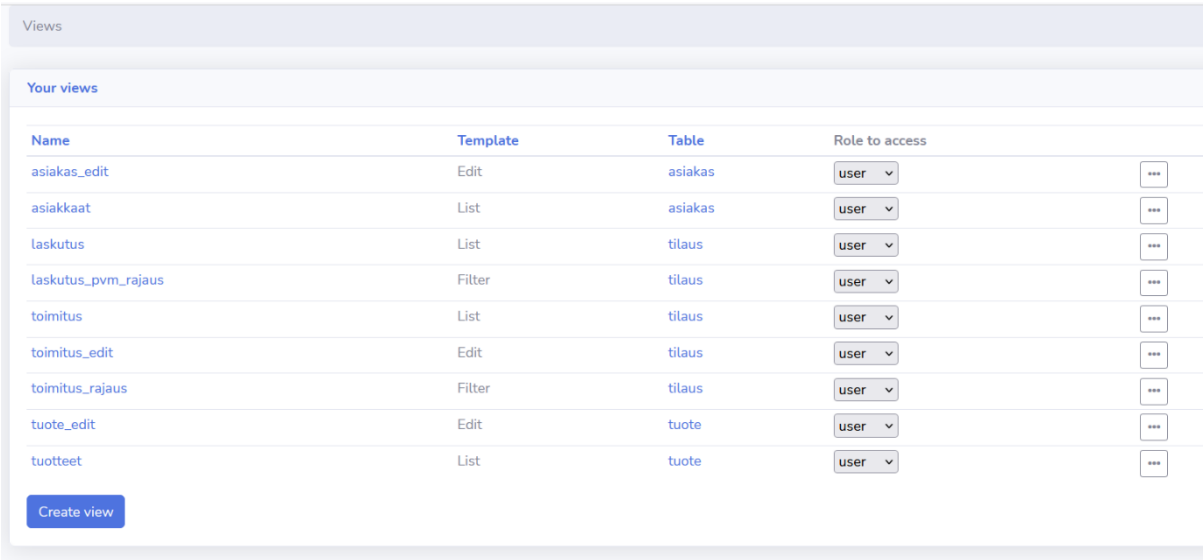
Alusta loi automaattisesti jokaiselle taululle juoksevasti numeroidun perusavainkentän ID, joka kuvassa 23 on merkitty keltaisella tunnisteella Primary key. Sinisellä tunnisteella Required näkyvät kentät on määritetty pakollisiksi ja vihreällä tunnisteella Unique yksilöllisiksi. Taulun kenttiä perustettaessa niille annettiin tyypit, jotka määrittävät millaista tietoa kenttään voidaan tallentaa. Perustietotyyppien lisäksi pystyttiin valitsemaan viiteavain (key to), jolla taulu yhdistettiin toiseen tauluun.

Kuva 23. Asiakas-taulun kentät



Sivuvalikon Views-kohdassa luotiin kuvan 24 mukaiset näkymät, jotka esittävät tauluihin tallennettuja tietoja valittujen kriteerien mukaisesti. Jokaiselle näkymälle määriteltiin rooli, joka näkymää voi käyttää. Näkymät konfiguroitiin käyttäen valmiita pohjia (template), joista edit- ja filter-pohjia oli mahdollista muokata suunnittelunäkymässä ja lisätä niihin elementtejä kun taas list-pohjassa käsiteltiin vain sarakkeita. List-valinta näytti valittujen sarakkeiden kaikki rivit, filter-valinnalla voitiin rajata tietoja ja edit-valinta loi lomakkeen, jossa taulun rivejä luotiin tai niiden sisältämää tietoa päivitettiin.

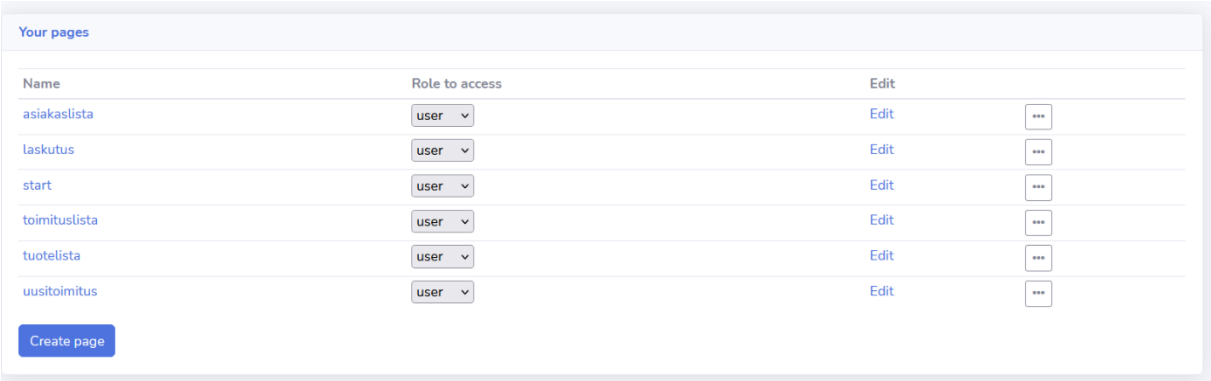
Kuva 24. Luodut näkymät



Sivuvalikon Pages-kohdassa rakennettiin varsinaiset sovelluksen sivut, joilla luotuja näkymiä näytetään käyttäjille. Tosin koska suoraan näkymiinkin pystyi linkittämään, ei kaikille näkymille ollut tarpeen tehdä omaa sivua. Luotuja sivuja oli mahdollista myös kopioida ja sen jälkeen muokata halutuilta osin, mikä nopeutti työtä. Suunnittelunäkymään raahattiin ja pudotettiin sivupalkista elementtejä, joiden ulkonäköä pystyi vain hyvin rajoitetusti muokkaamaan. Sivuja oli heti rakentamisen jälkeen mahdollista tarkastella selaimessa ilman erillistä julkaisemista.

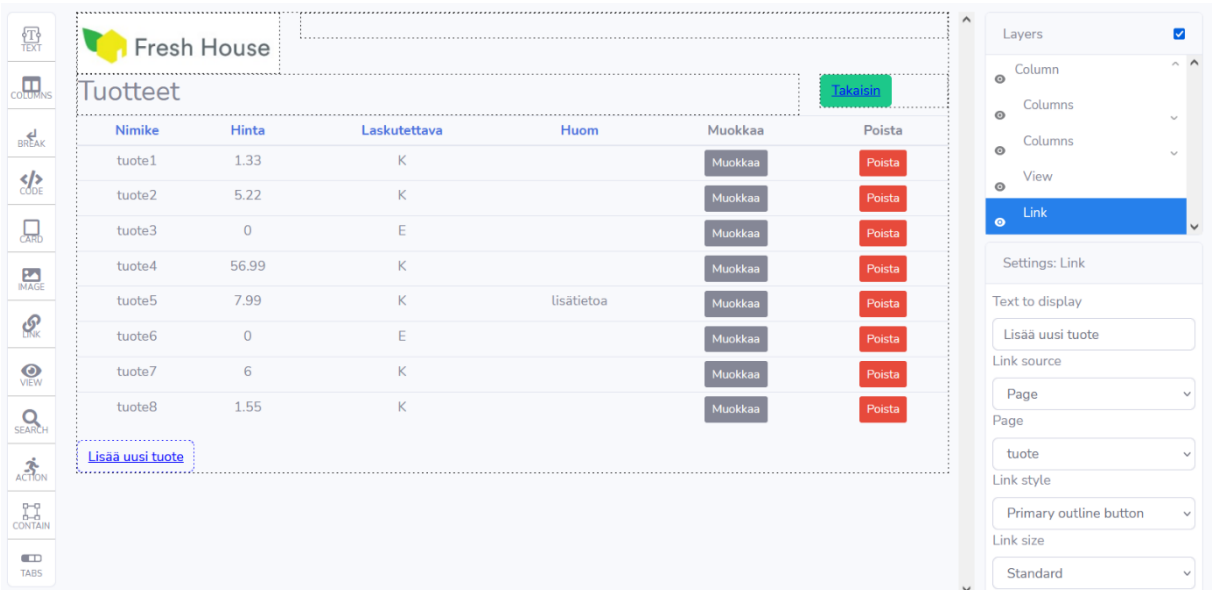
Valittu teema määritti esimerkiksi linkkien ja nappien ulkonäön, johon ei pystynyt itse vaikuttamaan. Myöskään toimintalogiikoita ei pystynyt omatoimisesti määrittelemään vaan ne oli sisäänrakennettu lomakkeen nappeihin. Rakennettuihin sivuihin annettiin käyttöoikeus kirjautuneelle käyttäjälle (user) kuvan 25 mukaisesti.

Kuva 25. Saltcorn-alustalla rakennetut sivut

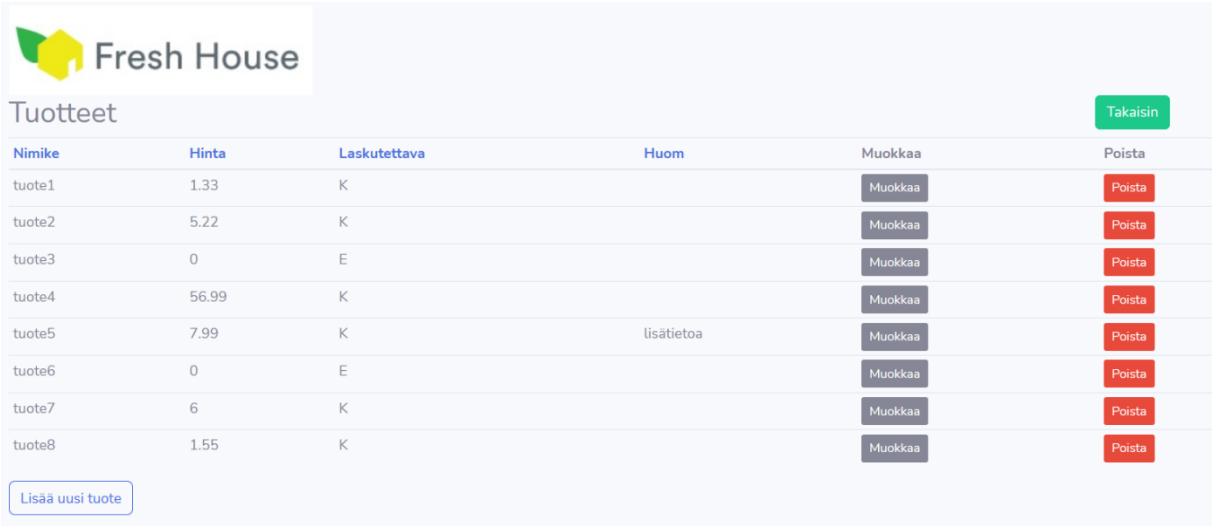


Saltcorn-alustalla päivämäärähaun toteutus laskutusraportissa ei onnistunut, joten ongelma kierrettiin lisäämällä kuukausi/vuosi-kenttä tilauksen tietoihin. Sen perusteella tietoja pystyi lajittelemaan kuukausitasolla. Alusta tarjosi vain rajoitetun määrän elementtejä, mutta oli tietokantaominaisuuksiltaan erittäin toimiva. Sillä luotu sivu näyttää selaimessa käytännössä samalta kuin suunnittelunäkymässä mikä voidaan havaita vertailemalla kuvia 26 ja 27. Saltcorn-alustalla toteutetun sovelluksen kaikki käyttöliittymäruudut ovat liitteessä 6

Kuva 26. Tuotelista-sivun suunnittelunäkymä



Kuva 27. Tuotelista-sivu selaimessa



4.3 Low-code -alustojen vertailu

Kaikilla alustoilla sovellus koodattiin tyhjästä eikä alustojen tarjoamia valmiita pohjia tai teemoja hyödynnetty. Alustojen käyttö oli pääsääntöisesti nopea oppia ja sovelluksen rakentamisen pystyi aloittamaan lähestulkoon heti alustalle kirjauduttuaan. Eniten perehtymistä vaati Outsystemsin käyttö, vaikka silläkin sovelluksen olisi halutessaan pystynyt rakentamaan varsin sukkelaan jos olisi jättänyt hyödyntämättä alustan tarjoamat

ainutlaatuiset ominaisuudet. Kun alustan peruskäyttö ja -toiminnot olivat tuttuja, ei alustojen välillä havaittu sovelluksen rakentamiseen kuluneessa ajassa juurikaan eroa.

Kaikkien alustojen käyttöliittymä oli helppokäyttöinen ja ymmärrettävä. Suurimmat erot olivat valmiiden elementtien määrässä ja tietokannan sekä taulukoiden käsittelymahdollisuuksissa. Outsystems oli odotetusti jo liiankin monipuolinen yksinkertaisen sovelluksen rakentamiseen, mutta sen loppuun asti mietittyjä toimintalogiikoita ja elementtejä jäin muita alustoja käyttäessäni kaipaamaan. Tiettyjen ominaisuuksien toteuttaminen vaati enemmän työtä muilla alustoilla kuin Outsystemsillä, joka tarjosi rakentamiseen valmiit komponentit.

Toisaalta sekä Joget DX että Saltcorn tarjosivat Oytsystemsiä paremmat ja yksinkertaisemmat CRUD-toiminnot tietokannan käsittelyyn, johon molemmat sopivat erinomaisesti. Elementtivalikoima oli kuitenkin melko rajoitettu ja koin suurena miinuksena varsinkin Joget DX -alustalta täysin puuttuvat graafiset komponentit.

4.4 Low-code -julkaisalustan valinta

Käytetyistä alustoista tehtiin vertailevaa arviointia varten omien havaintojen ja toimeksiantajan kommenttien perusteella SWOT-analyysi, joka on esitelty taulukossa 2. Se selkeytti toivotusti järjestelmien vertailua ja edesauttoi julkaisualustan valintaa, kun asiaa pohdittiin yhdessä toimeksiantajan kanssa. Valinnassa painotettiin toiminnallisuutta, käytettävyyttä, ylläpidettävyyttä ja kustannuksia. Toimeksiantajan tarpeisiin vastasi parhaiten Joget DX -alusta, jolla sovellus päätettiin julkaista. Sovellus viimeisteltiin käyttövalmiiksi poistamalla testidata ja korvaamalla se oikeilla, asianmukaisilla tuote- ja asiakastiedoilla.

Taulukko 2. Alustoja vertaileva SWOT-analyysi

	VAHVUUDET	HEIKKOUDET	MAHDOLLISUUDET	UHAT
Outsystems	▪ *laaja tuki ja käyttäjämäärä* ▪ *elementit* ▪ *tehokkuus* ▪ *tekninen kyvykkyys* ▪ *selkeys* ▪ *helppo-käyttöisyys*	▪ *käyttö vaatii perehtymistä* ▪ *suunnattu suurille yrityksille* ▪ *asennettava koneelle* ▪ *maksullisen version hinta*	▪ *skaalautuvuus* ▪ *laajennettavuus* ▪ *alustan jatkuva kehitys* ▪ *2 GB maksuton tila*	▪ *hinnoittelu datamäärän tai sovellus-tarpeen kasvaessa* ▪ *ilmaisversion lakkautus*
Joget DX	▪ *ylläpito* ▪ *helppo-käyttöisyys* ▪ *CRUD-elementit* ▪ *selainpohjaisuus* ▪ *hinnoittelu* ▪ *raportit*	▪ *data syötettävä käsin, ei sisäänlukumah dollisuutta* ▪ *graafisten elementtien puute*	▪ *laajennettavuus* ▪ *tietokannan tallennus omalle palvelimelle*	▪ *visuaalisten elementtien tarve käyttö-liittymässä* ▪ *rajallinen maksuton tila* ▪ *ilmaisversion lakkautus*
Saltcorn	▪ *tietokanta-pohjaisuus* ▪ *ylläpito* ▪ *selainpohjaisuus* ▪ *hinnoittelu*	▪ *pelkistetty elementti-valikoima* ▪ *rajoitetut muokkaus-mahdollisuudet* ▪ *lajittelu ei kaikilla kentillä onnistu*	▪ *tarpeellisia ominaisuuksia kehitteillä* ▪ *nuori projekti, jossa paljon potentiaalia*	▪ *alustan kehittämisen päättyminen* ▪ *serverin hinnoittelu*

4.5 Sovelluksen testaus

Kaikkia eri alustoilla luotuja sovelluksia testattiin tekijän toimesta jatkuvasti niitä rakennettaessa ja näin saatiin karsittua suurimmat virheet ja toimimattomuudet. Myös toimeksiantajan työntekijät testikäyttivät kaikkia kolmea sovellusta ja antoivat niistä palautetta, jota hyödynnettiin alustan valinnassa. Lisäksi palautteen perusteella käyttöönotettavan sovelluksen käytettävyyttä viilailtiin muun muassa lisäämällä vahvistusikkunoita ja rajausperusteita. Ennen käyttöönottoa sovelluksen toimintaa testattiin vielä eri päätelaitteilla ja selaimilla sekä mobiilissa. Sovelluksen pienikokoisuuden ja yksinkertaisuuden takia sitä ei erikseen pilotoitu vaan se otettiin testikäytön ja tehtyjen viimeistelyjen jälkeen suoraan toimeksiantajalle käyttöön.

5 Johtopäätökset ja pohdinta

Opinnäytetyön tekeminen sujui kokonaisuudessaan ongelmitta, sen toteutus onnistui tavoitteiden mukaisesti ja tulokset olivat jopa odotettua paremmat. Tietokantapohjainen web-sovellus osoittautui toimeksiantajalle sopivaksi ratkaisuksi ja sen toteuttaminen low-code -työkaluilla oli äärimmäisen mielenkiintoista. Alustat toimivat käytössä moitteettomasti ja koodittomuuden ansiosta sovelluksen rakentaminen oli nopeaa. Toki halutessaan säätöihin ja määrittelyihin olisi saanut aikaa tuhraantumaan vaikka kuinka paljon, mutta käytännössä toimivan sovelluksen sai syntymään melkeinpä käden käänteessä muutamassa tunnissa. Toisaalta valmiit elementit etukäteen määriteltyine raameineen myös asettivat rajoituksia, eikä sovelluksista olisi edes omilla css-määrittelyillä saanut ulkonäöltään identtisiä.

Projektin aikataulu piti hyvin, vaikka lomakausi hieman sotki käyttöönottoa ja loi painetta viimeistely- ja käyttöönottovaiheeseen. Sain kuitenkin sovelluksen toteutusvaiheen valmiiksi ennakoitua aiemmin, joten raportin kirjoittamiseen jäi hyvin aikaa. Käyttämäni tehtävienhallintatyökalu Plannerin visuaalisuus auttoi osaltaan pysymään aikataulussa ja saavuttamaan asettamani tavoitteet.

5.1 Toimeksiantajan testaushavainnot ja sovelluksen käytettävyys

Toimeksiantajan antaman palautteen mukaan sovellus täytti kaikki sille asetetut odotukset ja osin ylittikin ne. Sovelluksen tuottama laskutusraportti tehostaa huomattavasti laskutusprosessia, kun tietoa ei tarvitse etsiä erillisiltä lipuilta ja lapuilta vaan kaikki tarvittava on yhdessä näkymässä. Toimitusten syöttö on sujuvaa ja niitä on nopea lisätä esitäytettyjen kenttien ansiosta. Manuaalisen työn vähentymisen myötä syntynyt ajansäästö on useita tunteja kuussa ja vuositasolla arviolta kymmenkunta työpäivää.

Sovelluksen ansiosta laskutustieto on koko ajan ajantasaista eikä esimerkiksi hinta jää epähuomiossa jonkun asiakkaan kohdalla päivittämättä. Inhimillisten virheiden määrä pienenee, kun tiedot haetaan yhteisestä tietokannasta. Virheellisten kirjausten muokkaus on vaivatonta ja poista-napin painamisen jälkeen näkyvä varmistusviesti takaa, ettei dataa poisteta epähuomiossa.

Lisäksi toimeksiantaja yllättyi positiivisesti siitä, että sovelluksen avulla saadaan jatkossa seurattua myös tuotteiden ja tarvikkeiden menekkiä sekä ylläpidettyä kuukausi-inventaaria. Raporteista pystyy yhdestä näkymästä katsomaan kaikki tarvittavat tiedot, kun aiemmin aikaa kului runsaasti excel-välilehtien läpikäymiseen ja tarkastamiseen. Kiitosta saivat myös mahdollisuudet valita raportin aikaväli vapaasti ja määrittää alasvetovalikoiden avulla lisäehtoja ennen raportin tulostamista. Lisäksi raporttien ulkonäkö ja selkeys olivat toimeksiantajan mieleen.

5.2 Sovelluksen käyttöönotto ja jatkokehitys

Koska web-sovelluksia ei tarvitse erikseen asentaa, sujui käyttöönotto joutuisasti. Sovellus oli periaatteessa heti käyttäjätunnusten perustamisen jälkeen käyttövalmis. Ennen käytön aloittamista käytiin toimeksiantajan työntekijöiden kanssa läpi sovelluksen toiminnot ja suositellut, yhdenmukaiset tuotteiden ja asiakkaiden nimeämiskäytännöt mitkä helpottavat sovelluksen käyttöä. Näistä laadittiin lisäksi selkeä sähköpostiohjeistus, jonka avulla uusien työntekijöiden perehdyttäminen on jatkossa mahdollisimman vaivatonta.

Käyttöoikeuksien ylläpitoa varten pääkäyttäjän on kirjauduttava alustalle, mutta vaadittavat toimenpiteet ovat itsessään yksinkertaisia. Jos oikeuksien ylläpitotarve on jatkuvaa, on sovellukseen mahdollista myöhemmin lisätä erillinen käyttöoikeuksienpäivityslomake. Mikäli sovellusta on tarpeen päivittää esimerkiksi lisäämällä raporteissa näytettäviä kenttiä tai muokkaamalla lomakkeita, onnistuu se toimeksiantajan pääkäyttäjältä melko helposti ilman koodaustaitojakin.

Sovelluksen ulkoasuun ei varsinaisesti sitä rakentaessa panostettu vaan toteutus tehtiin toiminnallisuus edellä. Visuaalista ilmettä voitaisiin haluttaessa räätälöidä toimeksiantajan brändiin sopivammaksi, nyt siihen viittaa vain sovelluksen vihreä värimaailma.

Luotua sovellusta on mahdollista päivittää ja laajentaa tarpeiden mukaan tulevaisuudessa esimerkiksi ottamaan vastaan tuotetilauksia suoraan toimeksiantajan asiakkailta. Lisäksi laskutusraportin tiedot voisi jatkossa lukea automaattisesti sisään laskutukseen, mikä vähentäisi sekä virheiden että käsin syötettävien rivien määrää ja luonnollisesti nopeuttaisi laskutuksen tekemistä.

5.3 Kehittämistyön haasteita ja ratkaisuja

Alun perin sovellus oli kustannuslähtöisesti ajateltu koodattavaksi Outsystemsin lisäksi Appsheet- ja PowerApps-alustoilla, mutta niiden toimintoihin tutustumisen jälkeen molemmat päätettiin korvata toisilla. Appsheets on selkeästi suunnattu mobiilisovellusten tekemiseen eikä sellaisenaan soveltunut tämän web-sovelluksen toteuttamiseen. Lisäksi sen muokattavuus no-code -alustana oli käytännössä olematon eikä näkymiä juuri pystynyt muuttamaan tai omaa koodia lisäämään. PowerApps ei taipunut suunniteltuihin toimintoihin ja esimerkiksi eri tauluja yhdistäviin raportteihin toivotulla tavalla, joten siitä luovuttiin aikaisessa vaiheessa muutaman kokeilun jälkeen. On siis ensiarvoisen tärkeää miettiä millaista sovellusta kehittää ennen kuin valitsee siihen käytettävät low-code -työkalut, jotta niistä saa parhaan hyödyn irti.

Eniten päänvaivaa aiheutti laskutuslistauksen rakentaminen, koska siihen piti poimia tietoja kaikista kolmesta tietokannan taulusta. Lopullinen ratkaisu syntyi jokaisella alustalla täysin toisista poikkeavalla tavalla, mutta onnistui kuitenkin lopulta jokaisella. Tosin tehtyjä listauksia ei Saltcorn-alustalla pystynyt lajittelemaan linkitettyjen kenttien perusteella lainkaan, mikä oli selkeä puute.

Toinen puutteellisuus ilmeni Joget DX -alustalla, jossa asiakkaalle tehtyjä aiempia toimituksia ei saanut samaan näkymään uuden toimituksen syötön kanssa vaan niiden näkeminen vaatii ylimääräisen klikkauksen ja siirtymisen toiseen ikkunaan. Yhdelläkään alustalla pelkät valmiit elementit eivät olleet täysin riittäviä vaan tiettyjen toimintojen toteuttamiseen vaadittiin peruskoodaustaitoja javasript- ja html-kielillä. Lisäksi sovelluksiin tehtiin muutamia css-tyylimäärittelyjä, joilla ulkonäköä saatiin hieman siistittyä.

Jälkikäteen ajateltuna sovellus olisi kannattanut rakentaa ensimmäiseksi jollain toisella kuin Outsystems-alustalla. Sen lähes rajattomat mahdollisuudet asettivat muille alustoille odotukset, joita ne eivät täysin kyenneet lunastamaan. Myös alustalta toiseen vaihtaminen osoittautui ennakoitua hankalammaksi, koska työkalujen toiminnot oli toteutettu logiikaltaan niin toisistaan poikkeavasti. Identtisten toimintojen koodaaminen vei näin ollen todennäköisesti normaalia enemmän aikaa, kuin niin sanotusti puhtaalta pöydältä lähdettäessä.

Joidenkin toteutusprosessin aikana toimeksiantajalta tulleiden toiveiden lisääminen sovellukseen osoittautui melko hankalaksi, mikä korostaa perusteellisen suunnittelun tarvetta ja varsinkin alkuvaiheessa herkkää korvaa saadulle palautteelle sekä tiivistä yhteistyötä toimeksiantajan kanssa. Varsinkin tietokannan kenttiin jälkeenpäin tehtävät muutokset vaativat yllättävän paljon työtä. Jos kyseessä olisi ollut monimutkaisempi sovellus, olisi todennäköisesti sekä toiminnallisesta että ajallisesta näkökulmasta kannattanut aloittaa käyttöönotettavan sovelluksen toteuttaminen nollasta ennemmin kuin pyrkiä muuttamaan testisovellusta haluttuun suuntaan.

6 Yhteenveto

Opinnäytetyön teoreettisessa osassa onnistuin vastaamaan tutkimuskysymyksiini kattavasti ja samalla kerrytin huimasti omaa tietomääräni low-code -maailmasta, josta minulla ei ollut etukäteen kokemusta. Toiminnallisessa osuudessa testasin runsaan määrän erilaisia low-code -alustoja, mikä antoi syvempää näkemystä tarjolla oleviin ratkaisumalleihin. Low-code -työkaluilla toteuttamani web-sovelluksen avulla pystyttiin tavoitteiden mukaisesti merkittävästi tehostamaan toimeksiantajan laskutusprosessia ja samalla ratkaisemaan muitakin käytännössä havaittuja ongelmakohtia. Opinnäytetyön aikana hankittu tietotaito low-code -sovelluskehityksestä täydensi aiempaa osaamistani ja toi motivaatiota oppia laajasta aiheesta vielä lisää. Uskon tästä olevan tavattomasti hyötyä työelämässä erilaisten low-code -tekniikoiden hiljalleen valloittaessa myös pk-yrityksiä.

Opettavaisinta oli se, ettei low-code -työkaluja todellakaan voi niputtaa samaan kastiin vaan erot niiden välillä ovat merkittäviä. Alustoihin tutustumiseen on pk-yrityksissä panostettava jopa enemmän kuin suuryrityksissä, koska väärä valinta voi aiheuttaa turhia aika- ja kustannustappioita. Yrityksen tarpeet on kartoitettava tarkasti ja huomioitava, että se mikä sopii toiselle, ei välttämättä sovikaan itselle.

Low-code -sovelluskehitys tarjoaa pk-yrityksien liiketoiminnalle kasvupotentiaalin ja useita etuja, joita suuremmat yritykset eivät samalla tavalla pysty hyödyntämään. Se on luonut pk-yrityksille selvän markkinaraon ja kilpailuedun, jonka saavuttamiseen tarvitaan käytännössä vain uskallusta ja kokeilunhalua.

On nähtävissä, että low-code tavalla tai toisella on vääjäämättä osa tulevaisuuden yritysmaailmaa Suomessakin. Nykyaikana muutos on jatkuvaa ja vastaisuudessa laajakatseisuus sekä aito halu pysyä mukana kehityksen rattailla samalla aiemmin opittua soveltaen ovat välttämättömyys it-ammattilaisille. Into uuden oppimiseen, hyödyntämiseen ja kehittämiseen voi jatkossa osoittautua ammattitaidon veroiseksi valttikortiksi, jolle riittää kysyntää.

Lähteet

a.i.mater Oy. (2018). *Mitä on low-code?* https://aimater.com/mita-on-low-code/

Arrow ECS Finland Oy. (2020). *Arrow ja Microsoft rakentavat suomalaista low-code-ekosysteemiä | Arrow ECS FI*. Tutkimus. https://www.arrow.com/ecs/fi/ajankohtaista/news-articles/arrow-ja-microsoft-rakentavat-suomalaista-low-code-ekosysteemiae/

Ashbaugh, C. (2020). *Closing the App Gap with Microsoft Power Platform*. Microsoft. https://compass365.com/closing-the-app-gap-with-microsoft-power-platform/

Carroll, M. (2021). *Low code makes app development easy for small businesses*. IT Supply Chain. https://itsupplychain.com/low-code-makes-app-development-easy-for-small-businesses/

Enright, C. (n.d.). *How to choose a Low-Code platform for your digital transformation*. PhixFlow Blogi. https://www.phixflow.com/how-choose-low-code-platform-your-digital-transfor/

Forrester. (2014). *New Development Platforms Emerge For Customer-Facing Applications*. Report. https://www.forrester.com/report/New+Development+Platforms+Emerge+For+CustomerFacing+Applications/-/E-RES113411#

Forrester. (2020). *Predictions 2021: Software Development*.

Forsyth, A. (2021). *Low-Code and No-Code: What's the Difference and When to Use What?* Outsystems. https://www.outsystems.com/blog/posts/low-code-vs-no-code/

Garms, C. (2021). *Low Code for Small Businesses helps any IT*. Neptune Software Blogi. https://www.neptune-software.com/resource/low-code-for-small-businesses/

Gartner. (2021). *Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 23% in 2021*. https://www.gartner.com/en/newsroom/press-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021

GitHub. (2021). *Saltcorn: Free and open source no-code application builder*. https://github.com/saltcorn/saltcorn

Guta, M. (2021). *Low-Code Development Tools Offer New Options for Small Businesses*. https://smallbiztrends.com/2019/10/low-code-development-tools-small-business.html

Huff, T. (2019). *Adapting Agile to Build Products with Low-Code: Tips and Tricks*. Outsystems Blogi. https://www.outsystems.com/blog/posts/adapting-agile-to-low-code/

Huff, T. (2020). *Kanban vs. Scrum: Which Is Best for Your Low-Code Team?* Outsystems Blogi. https://www.outsystems.com/blog/posts/scrum-vs-kanban/

Ikonen, M. (2011). *Lean Thinking in Software Development : Impacts of Kanban on Projects*. [väitöskirja,Helsingin yliopisto]. http://www.cs.helsinki.fi/

Joget. (n.d.). *OPEN SOURCE WORKFLOW AND LOW CODE PLATFORM FOR DIGITAL TRANSFORMATION*. https://www.joget.com/

Juvonen, R. (2018). *Ohjelmistoprojektin sudenkuopat ja miten ne vältetään*. Books on Demand.

Kachot, D. (2020). *How Low-Code Development Helps Small Businesses*. Clarion Technologies Blogi. https://www.clariontech.com/blog/how-low-code-development-helps-small-businesses

Kailio, A. (2020). Low-code vähentää koodin kirjoittamista. *Tivi*.

Kissflow. (2018). *History of Low-Code Platforms*. https://kissflow.com/low-code/history-of-low-code-development-platforms/

Korhonen, R. (2019). Steppi nextille levelille – englanti suomessa. *Kielikello*. https://www.kielikello.fi/-/steppi-nextille-levelille-englanti-suomessa

Kotilainen, S. (2018). Nyt loppuu ohjelmointi? *Tivi*.

Kotilainen, S. (2019). Low-coden nopeus yllätti. *Tivi*.

Kuivainen, J., & Laukkanen, V. (2020). *Miten low-code ratkaisut liittyvät ERPin kehittämiseen?* . Attido Webinaari. https://www.attido.com/fi/webinaarit/low-code/webinaaritallenne-10-6-2020-miten-low-code-ratkaisut-liittyvat-erpin-kehittamiseen/

Laitila, T. (2021). Tällaista on low-code -sovelluskehitys - testissä Outsystems. *Tivi*.

Leskinen, M. (2017). *Mitä low-code-sovelluskehitys tarkoittaa?* Biit Oy Blogi. https://www.biit.fi/hub/blogi/mita-low-code-sovelluskehitys-tarkoittaa/

Lindroos, J.-E., & Lohivesi, K. (2010). *Onnistu strategiassa* (3. uudiste). Talentum.

Linna, J. (2021). *Kliksuttelemalla oivalluksiin – Näkemyksiä Power Platform -työkaluista*. Tampereen Korkeakouluyhteisön Digitalisaatio-Kehityshankkeen Blogi. https://blogs.tuni.fi/digitalk/suomeksi/kliksuttelemalla-oivalluksiin/

Lyytinen, K. (2020). *IT:ssä herätys – kansalaiskehittäjät tulevat!* Efima Blogi. https://www.efima.com/blogi/it-ssa-heratys-kansalaiskehittajat-tulevat/

Nyholm, J., Mäntyomena, H., & Niiranen, J. (2021). *Low-code -alustojen mahdollisuudet PK-yrityksille*. Forever Forward Webinaari.

https://www.youtube.com/watch?v=Fcu6aBCKUlg

Ojasalo, K., Moilanen, T., & Ritalahti, J. (2015). *Kehittämistyön menetelmät : uudenlaista osaamista liiketoimintaan*.

OutSystems. (n.d.). *OutSystems*. https://www.outsystems.com/

Paetsch, F., Eberlein, A., & Maurer, F. (2003). *Requirements engineering and agile software development*. Conference Paper. https://www.researchgate.net/publication/4034541_Requirements_engineering_and_agile_software_development

Rakshit, B. (2020). *How To Choose The Right Low-Code Development Platform*. TMCnet Feature. https://www.tmcnet.com/topics/articles/2020/07/16/446018-how-choose-right-low-code-development-platform.htm

Reinikainen, P. (2021). *It-talo ei pihtaile löytääkseen työntekijöitä*. https://www.yrittajat.fi/uutiset/654326-it-talo-ei-pihtaile-loytaakseen-tyontekijoita-6000-euron-rekrytointibonus-ja-yli-7000

Rousku, K. (2013). *Pienennä riskejä, etsi samalla mahdollisuuksia!* Tivi Blogi. https://www-tivi-fi.ezproxy.hamk.fi/blogit/pienenna-riskeja-etsi-samalla-mahdollisuuksia/a5475e30-ea86-39a0-ab1c-eaf7c66e25ec

Rymer, J., & Seguin, B. (2019). *Watch Your Language! "Low-Code" And "No-Code" Are Not The Same*. Forrester. https://go.forrester.com/blogs/watch-your-language-low-code-and-no-code-are-not-the-same/

Sacolick, I. (2020). *7 low-code platforms developers should know*. InfoWorld. https://www.infoworld.com/article/3583576/7-low-code-platforms-developers-should-know.html

Salesforce. (n.d.). *How to Select Low Code Application Development Frameworks*. https://www.salesforce.com/products/platform/best-practices/application-development-framework/

Saltcorn. (n.d.). *Saltcorn - open-source no-code*. https://saltcorn.com/

Standish Group. (2015). *Chaos Report*. https://www.infoq.com/articles/standish-chaos-2015/

Stellman, A., & Greene, J. (2014). *Learning agile : understanding Scrum, XP, Lean, and Kanban*.

Tapanainen, T. (2018). *Mitä on low-code ja kuka sitä tarvitsee?* ECraft Oy Ab Blogi. https://www.ecraft.com/fin/blog/2018/2/15/mita-on-low-code-ketteraa-

sovelluskehitysta

TechRepublic. (2020). *Low-code platforms: A cheat sheet*. https://www.techrepublic.com/article/low-code-platforms-a-cheat-sheet/

Viestintätoimisto Manifesto. (2020). *Kyselytutkimus: Onko koodaaminen osa jokaisen työnkuvaa tulevaisuudessa?* https://news.cision.com/fi/viestintatoimisto-manifesto/r/kyselytutkimus--onko-koodaaminen-osa-jokaisen-tyonkuvaa-tulevaisuudessa----lahes-viidennes-suomalais,c3120218

Vincent, P., Iijima, K., Driver, M., Wong, J., & Natis, Y. (2019). *Gartner: Magic Quadrant for Enterprise Low-Code Application Platforms*. https://www.gartner.com/en/documents/3956079/magic-quadrant-for-enterprise-low-code-application-platf

Visma. (2017). *Miten yritys tekee SWOT-analyysin?* Blogi. https://www.visma.fi/blog/miten-yritys-tekee-swot-analyysin/

Wennström, C.-G. (2021). *Nopeampaa ohjelmistokehitystä automaation avulla*. Solita Blogi. https://www.solita.fi/blogit/nopeampaa-ohjelmistokehitysta-automaation-avulla/

Westfall, B. (2020). *Your Guide to Understanding the True Cost of Business Software*. Capterra Blogi. https://blog.capterra.com/business-software-cost/

Wong, J., Vincent, P., & Jain, A. (2021). *Gartner: What Is the Difference Between No-Code and Low-Code Development Tools?*

Åkerman, H. (2021). *Mikä ihmeen low-code/no-code?* https://www.linkedin.com/pulse/mikä-ihmeen-low-codeno-code-henriikka-åkerman

Liite 1: Aineistonhallintasuunnitelma

Kehitysprojekti:

Kehitysprojektin aikana tehdään muistiinpanoja ja pidetään työskentelypäiväkirjaa, johon kerätään tietoa projektin etenemisestä. Lisäksi kehitystyön ja aikataulun hallinnassa hyödynnetään projektinhallintaohjelmaa. Edellä mainittuja tietoja analysoidaan opinnäytetyötä varten. Osana kehitystyötä piirretään mallinnoksia ja luonnoksia suunnittelutyökalulla sekä otetaan työvaiheista kuvakaappauksia, mitkä julkaistaan opinnäytetyön osana. Toteutusvaiheessa rakennetaan kolmella eri alustalla web-sovellus, joka ei ole julkinen vaan vaatii näkyäkseen kirjautumisen.

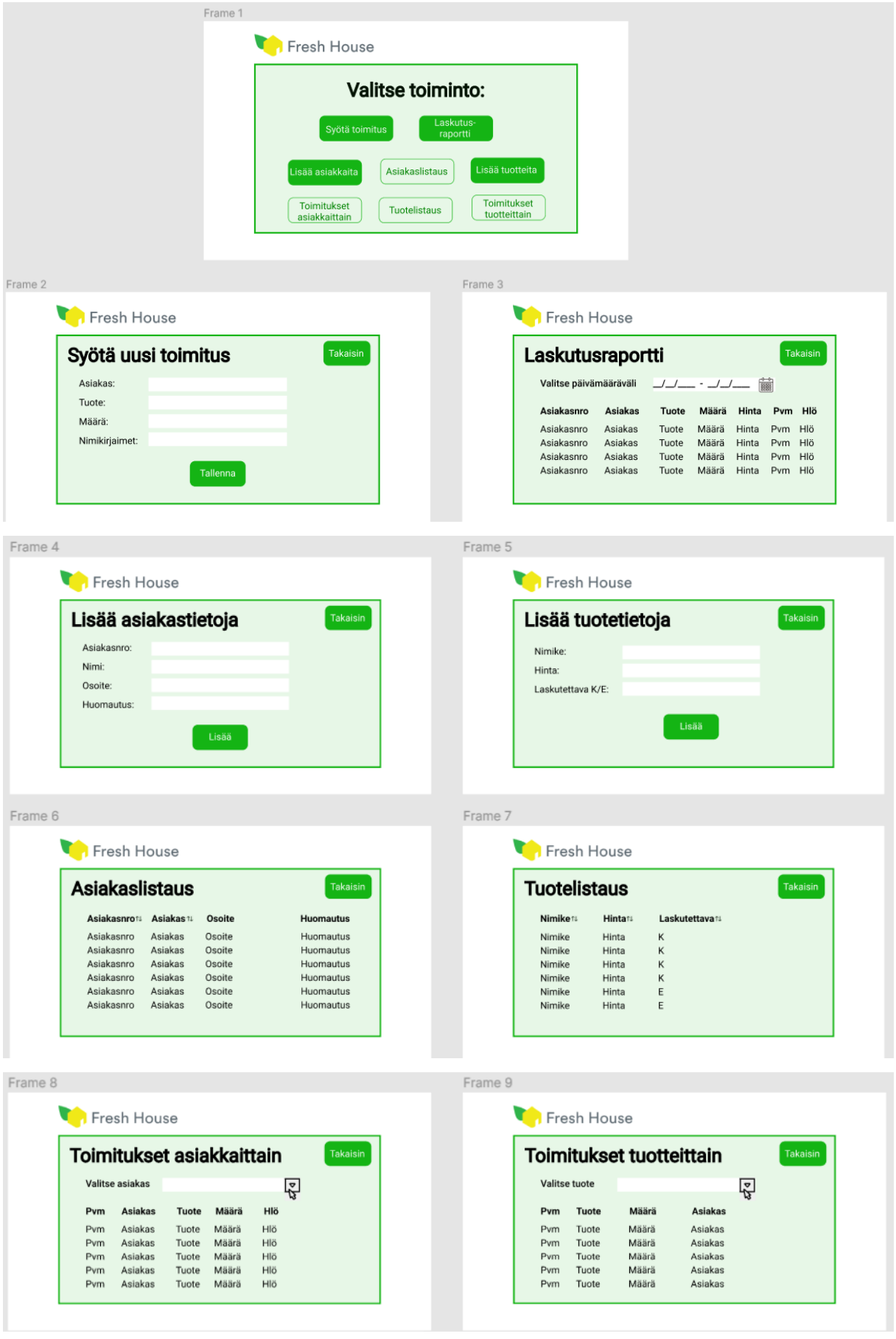
Toimeksiantajan kanssa pidetään yhteyttä sähköpostitse ja puhelimitse. Sähköpostit ja keskusteluista tehdyt muistiinpanot säilytetään alla kuvatun mukaisesti. Valmiin projektin onnistumisesta kerätään tietoa excel- ja word-tiedostoihin.

Kaikkea aineistoa säilytetään tekijän tietokoneen C-asemalla, josta tehdään säännöllisesti varmuuskopioita ulkoiselle USB-muistitikulle. Aineistoa säilytetään C-asemalla vuoden ajan opinnäytetyön valmistumisesta, jonka jälkeen aineisto ja alustoilla rakennetut sovellukset asianmukaisesti tuhotaan.

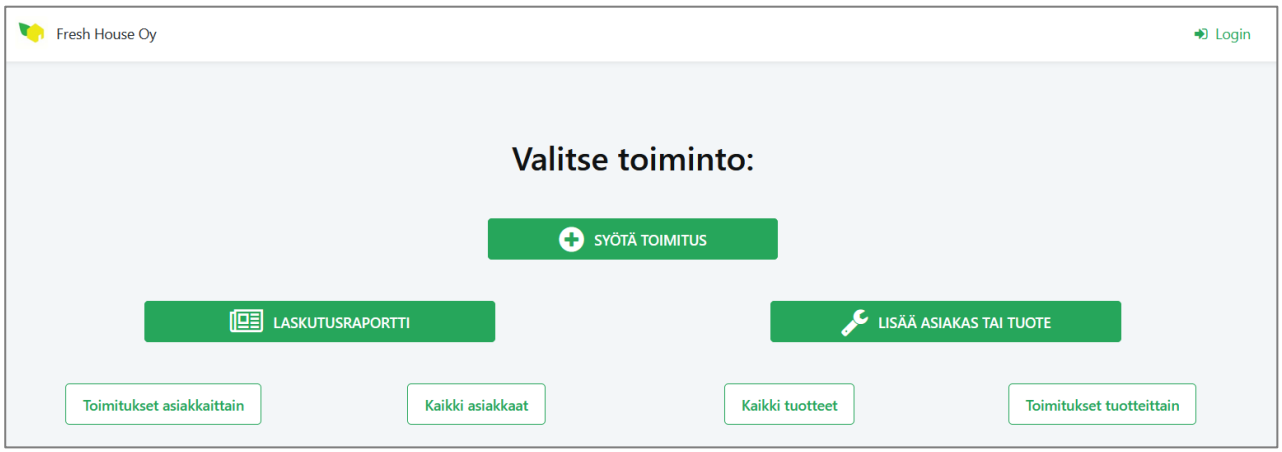
Liite 2: Käyttäjätarinat priorisoinnin mukaan järjestettynä

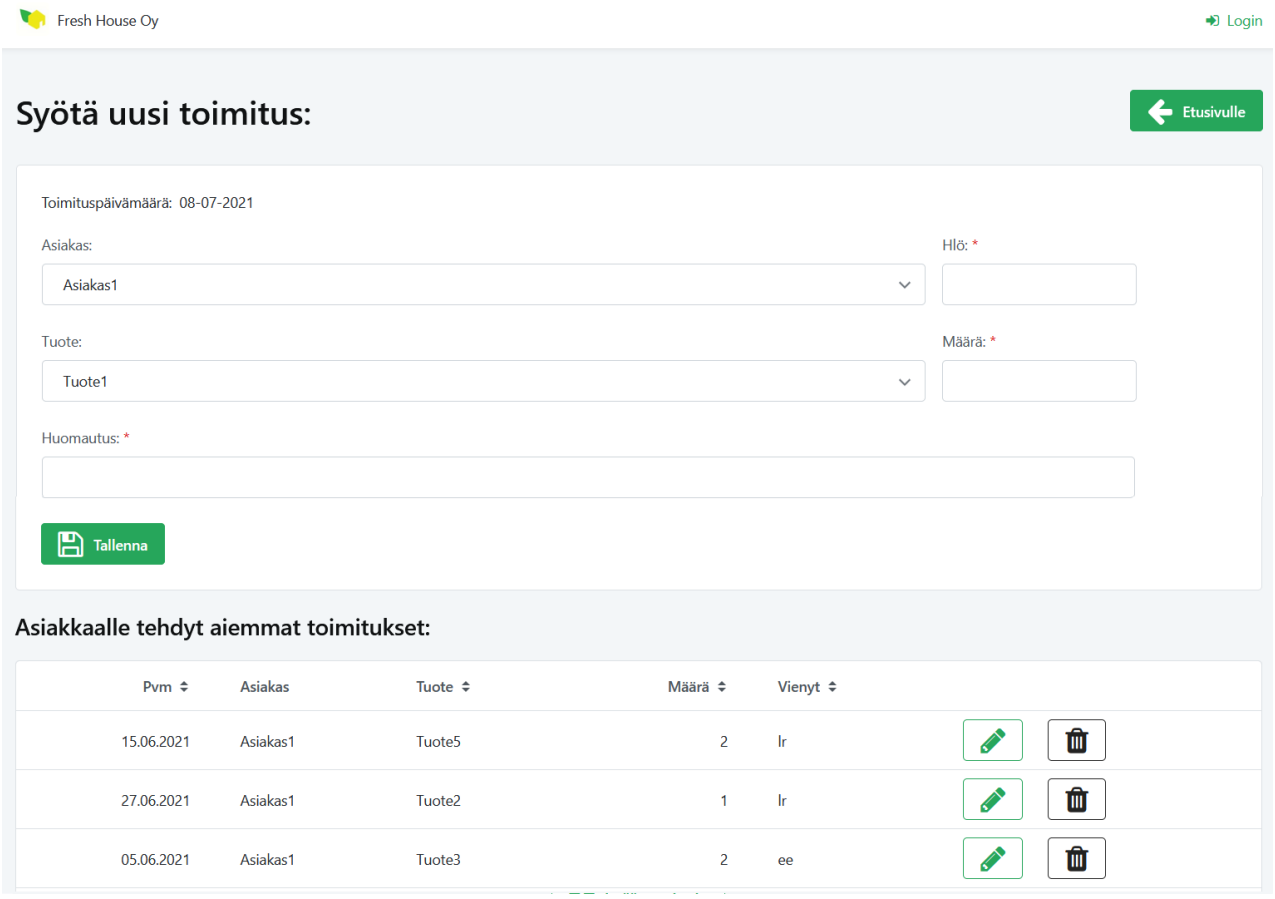
Käyttäjätarina	**Priorisointi**
Varastomiehenä haluan kirjata uuden toimituksen asiakkaalle.	1
Laskuttajana haluan tulostaa laskutusraportin päivämäärärajauksella ja lajitella tiedot asiakkaiden mukaan.	1
Toimistotyöntekijänä haluan päivittää asiakkaiden tietoja.	1
Toimistotyöntekijänä haluan päivittää tuotteiden tietoja.	1
Toimistotyöntekijänä haluan lisätä uusia tuotteita.	1
Toimistotyöntekijänä haluan lisätä uusia asiakkaita.	1
Toimistotyöntekijänä haluan tulostaa asiakaslistauksen.	2
Toimistotyöntekijänä haluan tulostaa tuotelistauksen.	2
Toimistotyöntekijänä haluan seurata tuotteiden menekkiä.	3
Toimistotyöntekijänä haluan listata toimitukset tuotteittain.	3
Toimistotyöntekijänä haluan listata toimitukset asiakkaittain.	3
Toimistotyöntekijänä haluan lajitella toimituslistausta tietojen perusteella.	3
Varastomiehenä haluan nähdä asiakkaalle tehdyt aiemmat toimitukset.	3

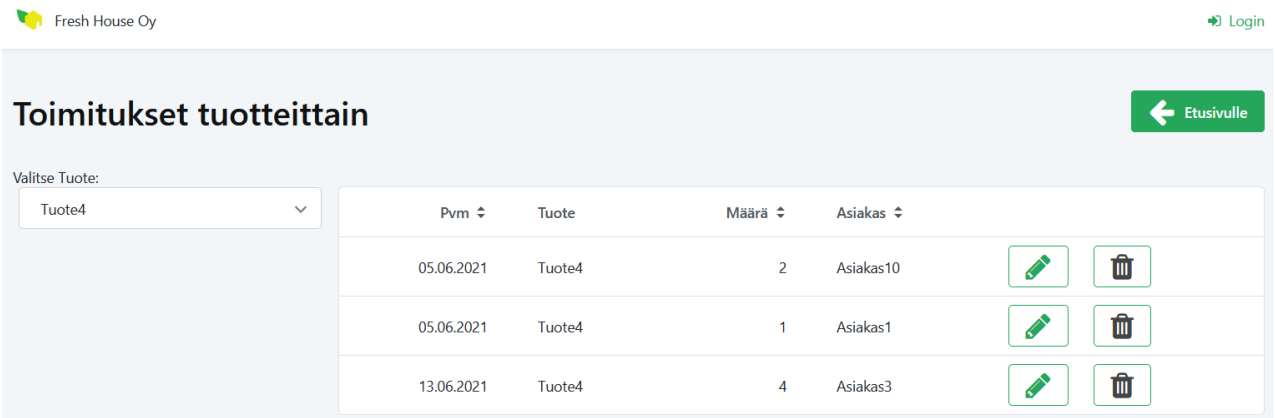
Liite 3: Luonnokset sovelluksen käyttöliittymäruuduista

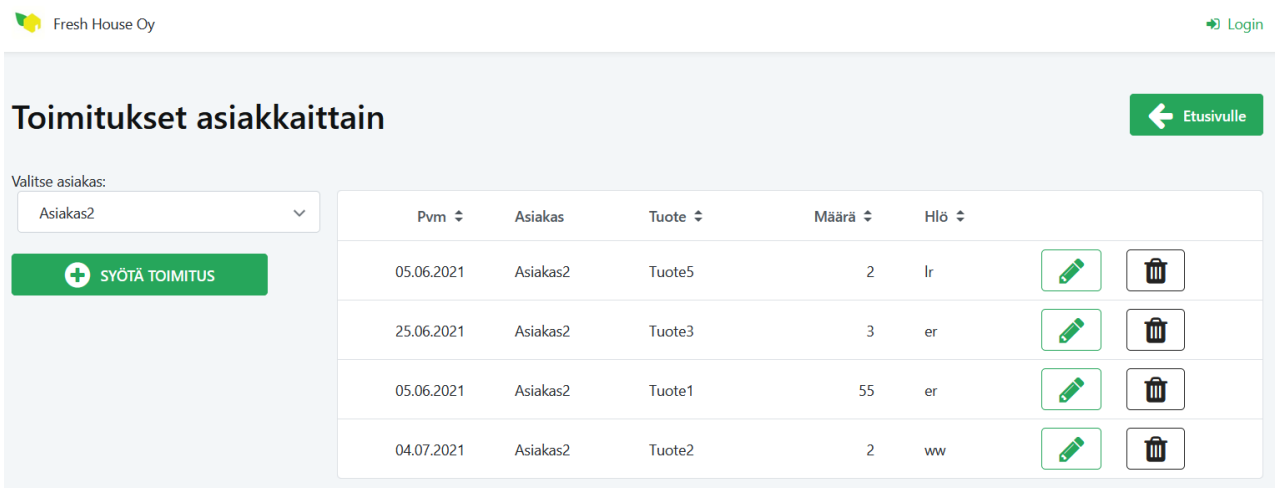


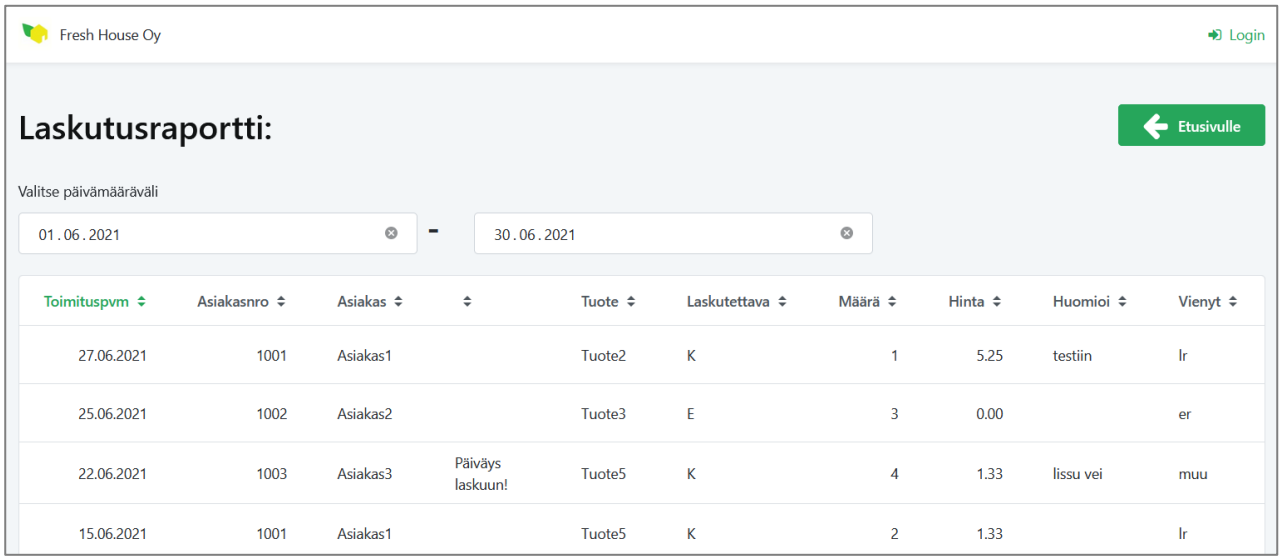
Liite 4: Outsystems-alustalla toteutetut sovelluksen käyttöliittymäruudut

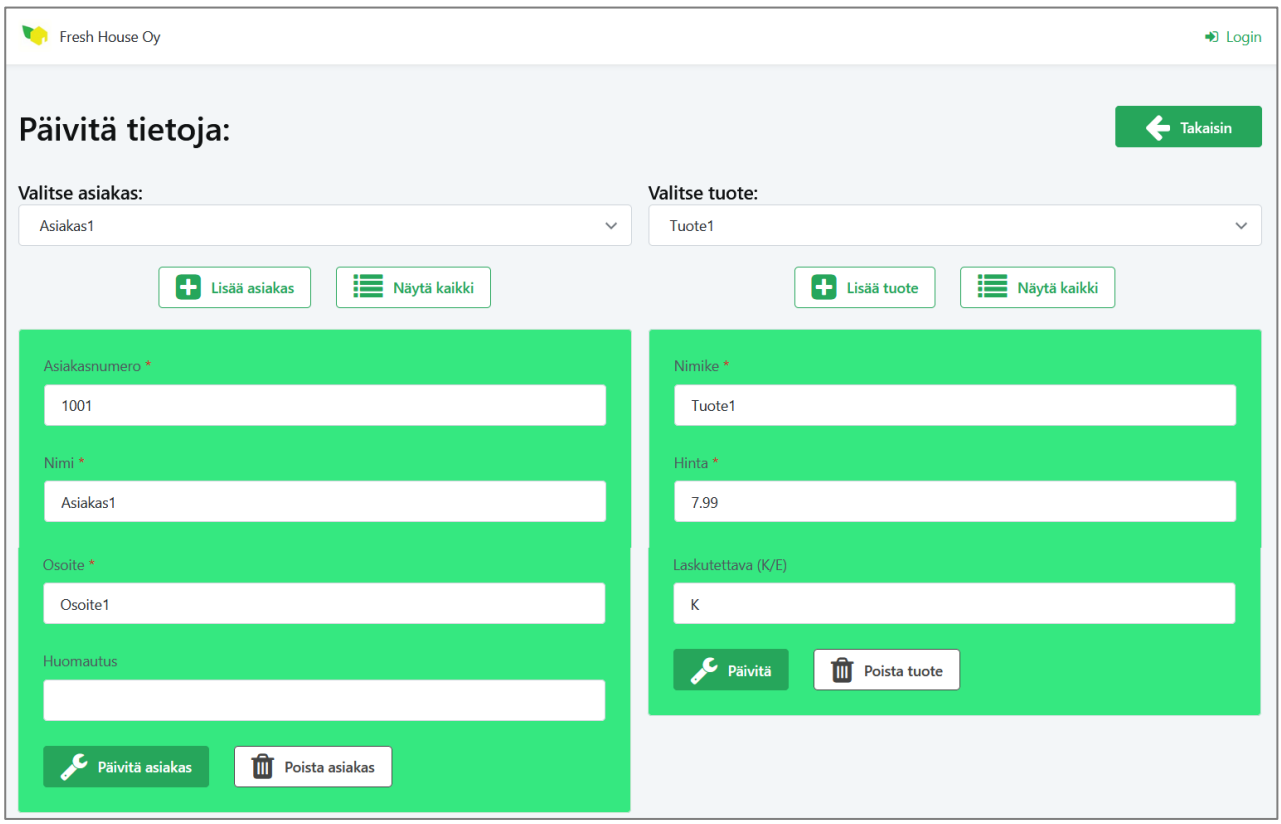


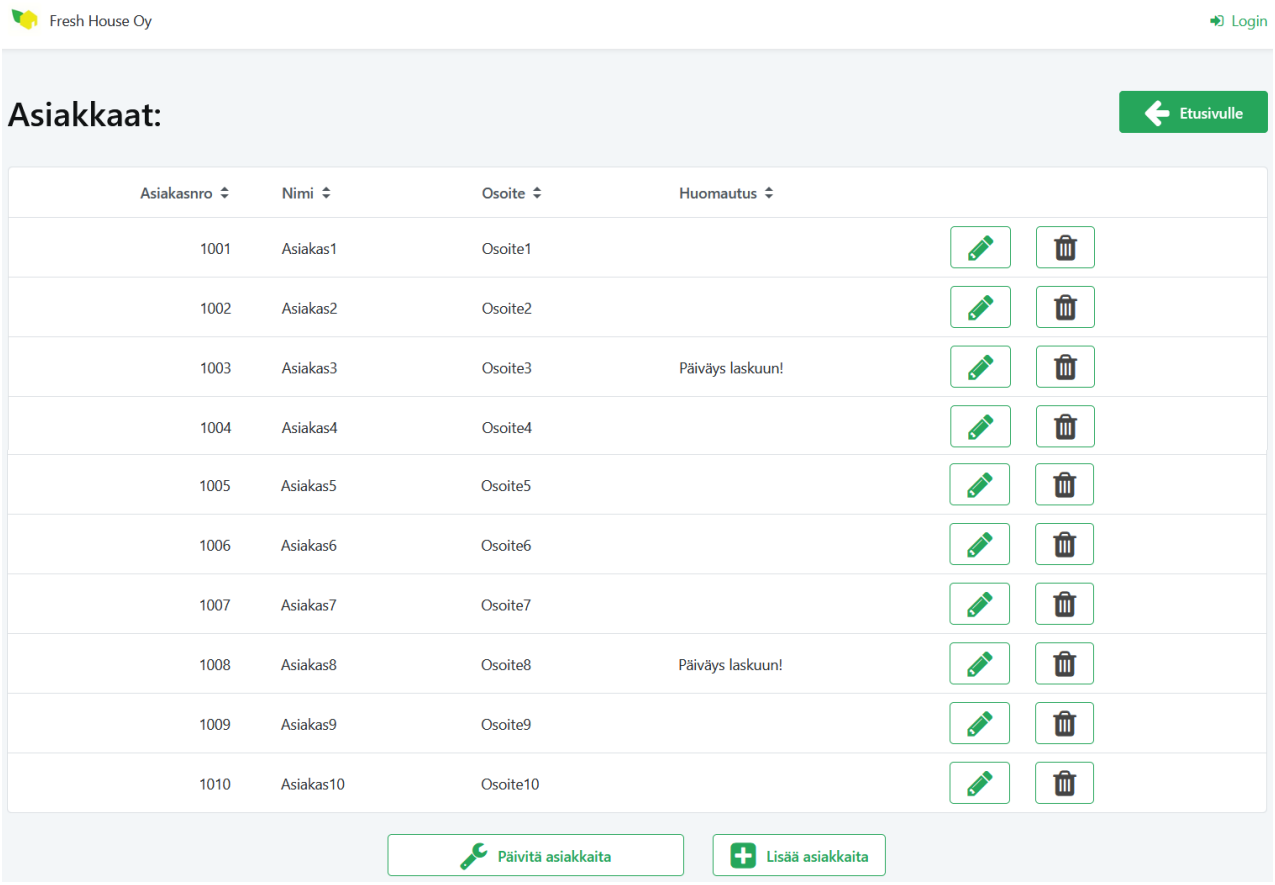


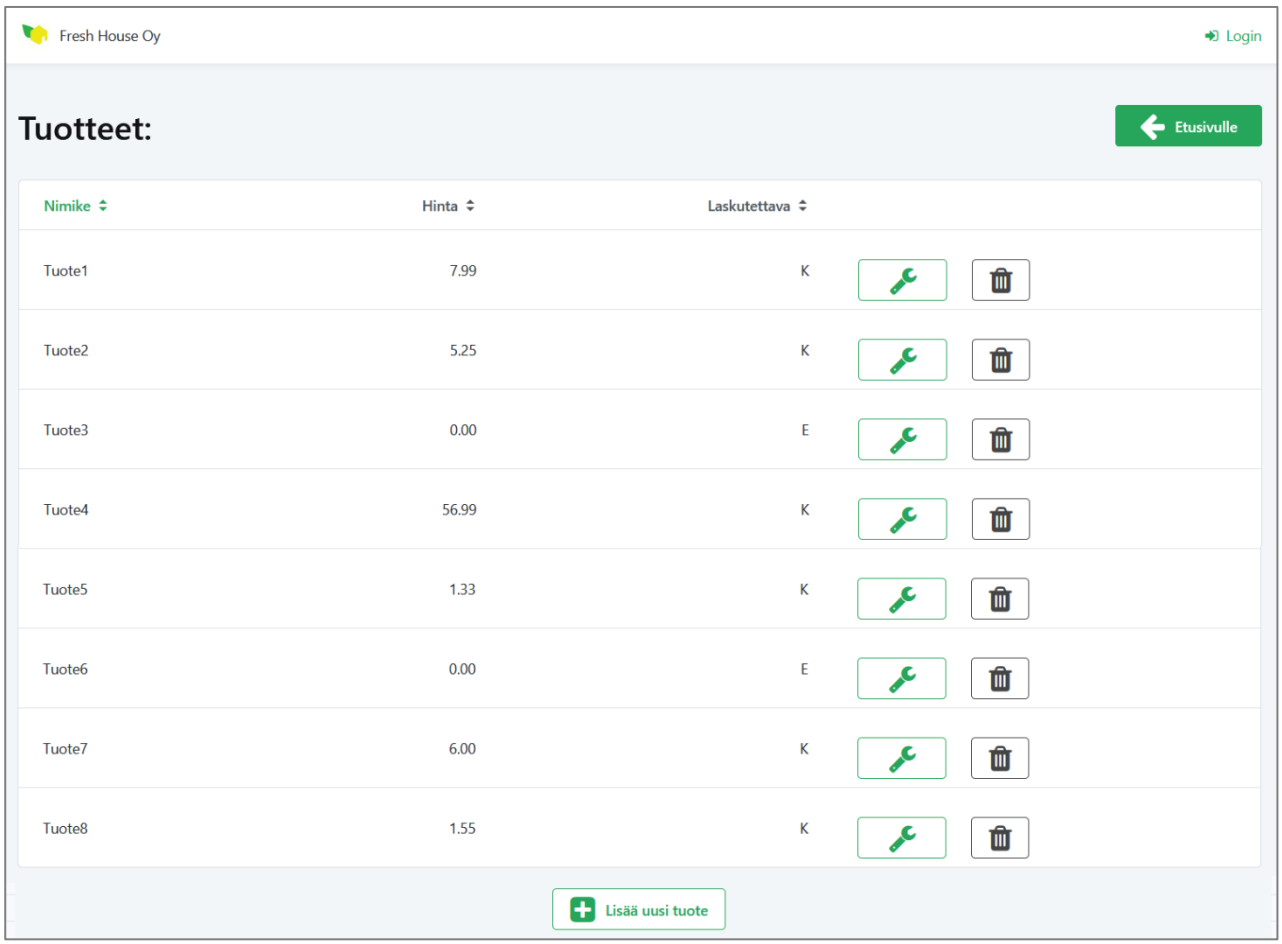



Fresh House Oy
Login
Toimitukset asiakkaittain
Etusivulle
Valitse asiakas:
Asiakas2
SYÖTÄ TOIMITUS
Pvm
Asiakas
Tuote
Määrä
Hlö
05.06.2021 Asiakas2 Tuote5 2 lr
25.06.2021 Asiakas2 Tuote3 3 er
05.06.2021 Asiakas2 Tuote1 55 er
04.07.2021 Asiakas2 Tuote2 2 ww

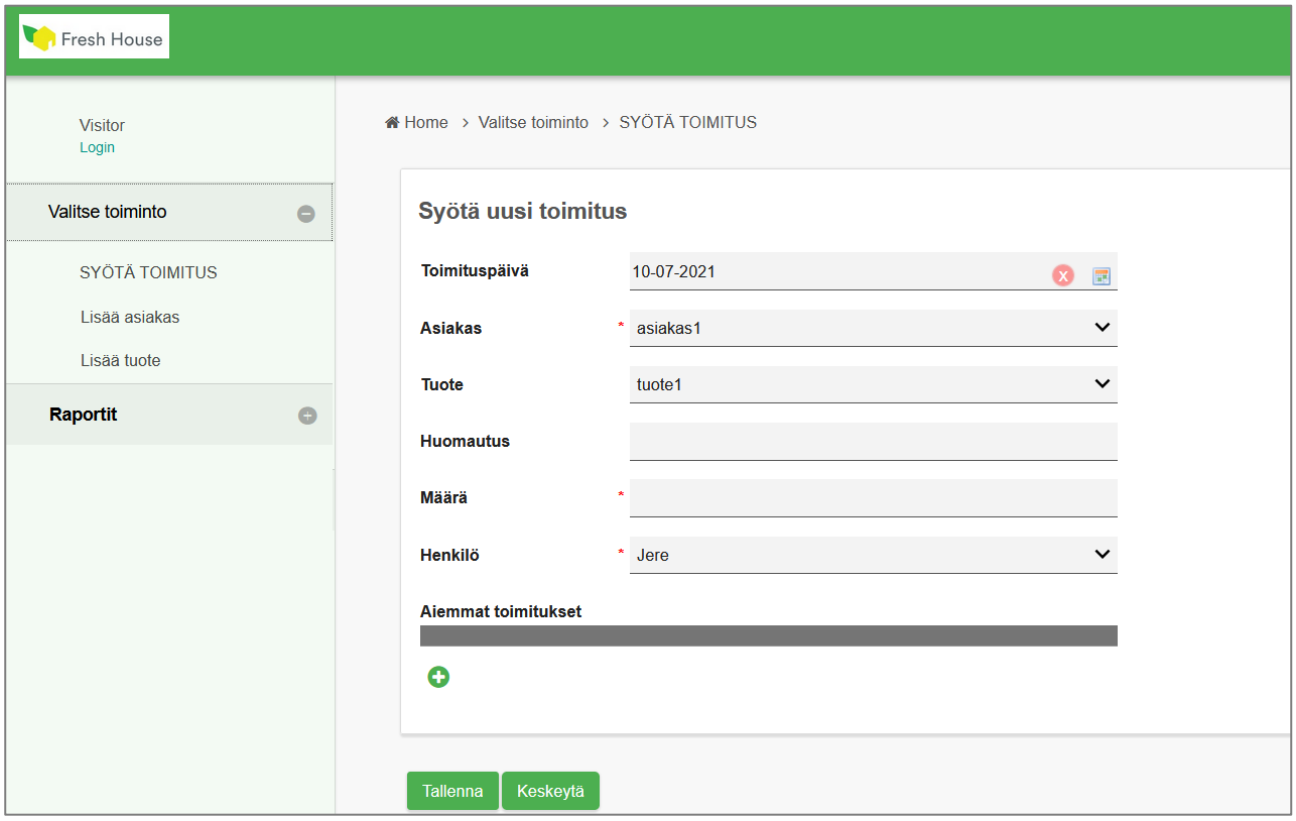

Fresh House Oy
Login
Laskutusraportti:
Etusivulle
Valitse päivämääräväli
01.06.2021
-
30.06.2021
Toimituspvm
Asiakasnro
Asiakas
Tuote
Laskutettava
Määrä
Hinta
Huomioi
Vienyt
27.06.2021 1001 Asiakas1 Tuote2 K 1 5.25 testiin lr
25.06.2021 1002 Asiakas2 Tuote3 E 3 0.00 er
22.06.2021 1003 Asiakas3 Päiväys laskuun! Tuote5 K 4 1.33 lissu vei muu
15.06.2021 1001 Asiakas1 Tuote5 K 2 1.33 lr

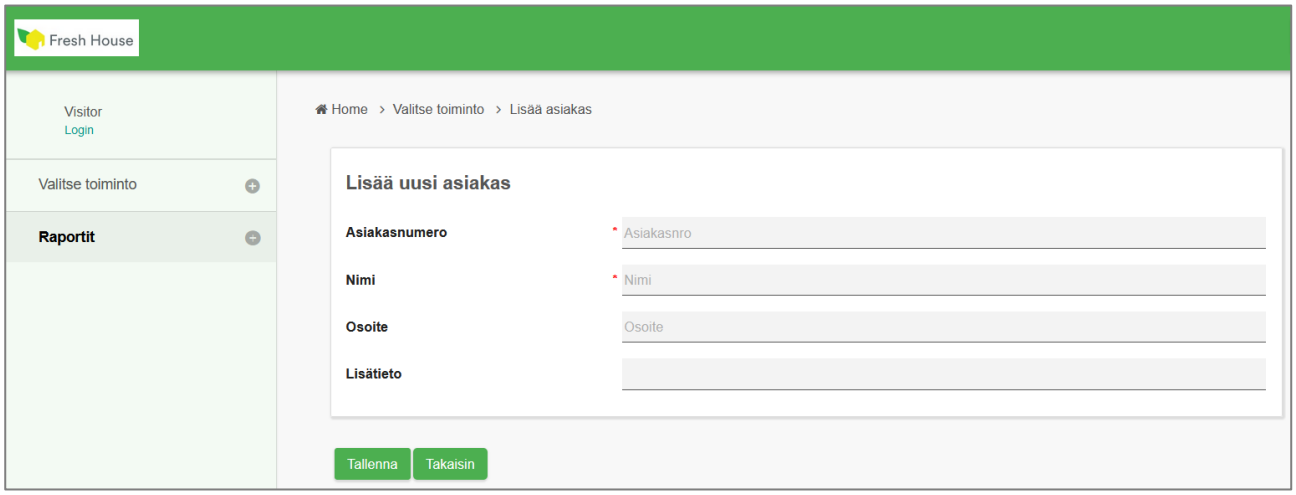

Fresh House Oy
Login
Päivitä tietoja:
Takaisin
Valitse asiakas:
Asiakas1
Lisää asiakas
Näytä kaikki
Asiakasnumero *
1001
Nimi *
Asiakas1
Osoite *
Osoite1
Huomautus
Päivitä asiakas
Poista asiakas
Valitse tuote:
Tuote1
Lisää tuote
Näytä kaikki
Nimike *
Tuote1
Hinta *
7.99
Laskutettava (K/E)
K
Päivitä
Poista tuote

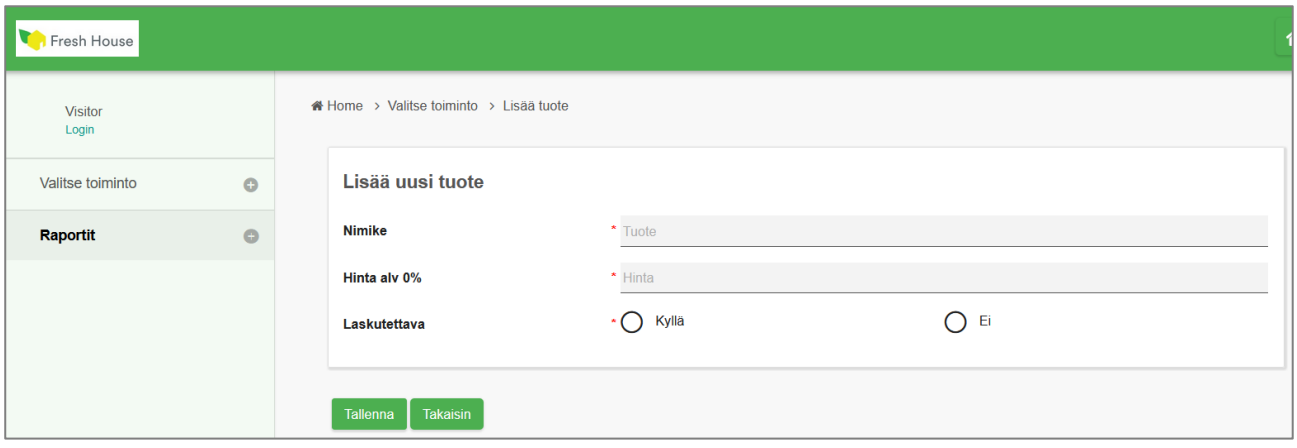

Fresh House Oy
Login
Asiakkaat:
Etusivulle
Asiakasnro
Nimi
Osoite
Huomautus
1001 Asiakas1 Osoite1
1002 Asiakas2 Osoite2
1003 Asiakas3 Osoite3 Päiväys laskuun!
1004 Asiakas4 Osoite4
1005 Asiakas5 Osoite5
1006 Asiakas6 Osoite6
1007 Asiakas7 Osoite7
1008 Asiakas8 Osoite8 Päiväys laskuun!
1009 Asiakas9 Osoite9
1010 Asiakas10 Osoite10
Päivitä asiakkaita
Lisää asiakkaita

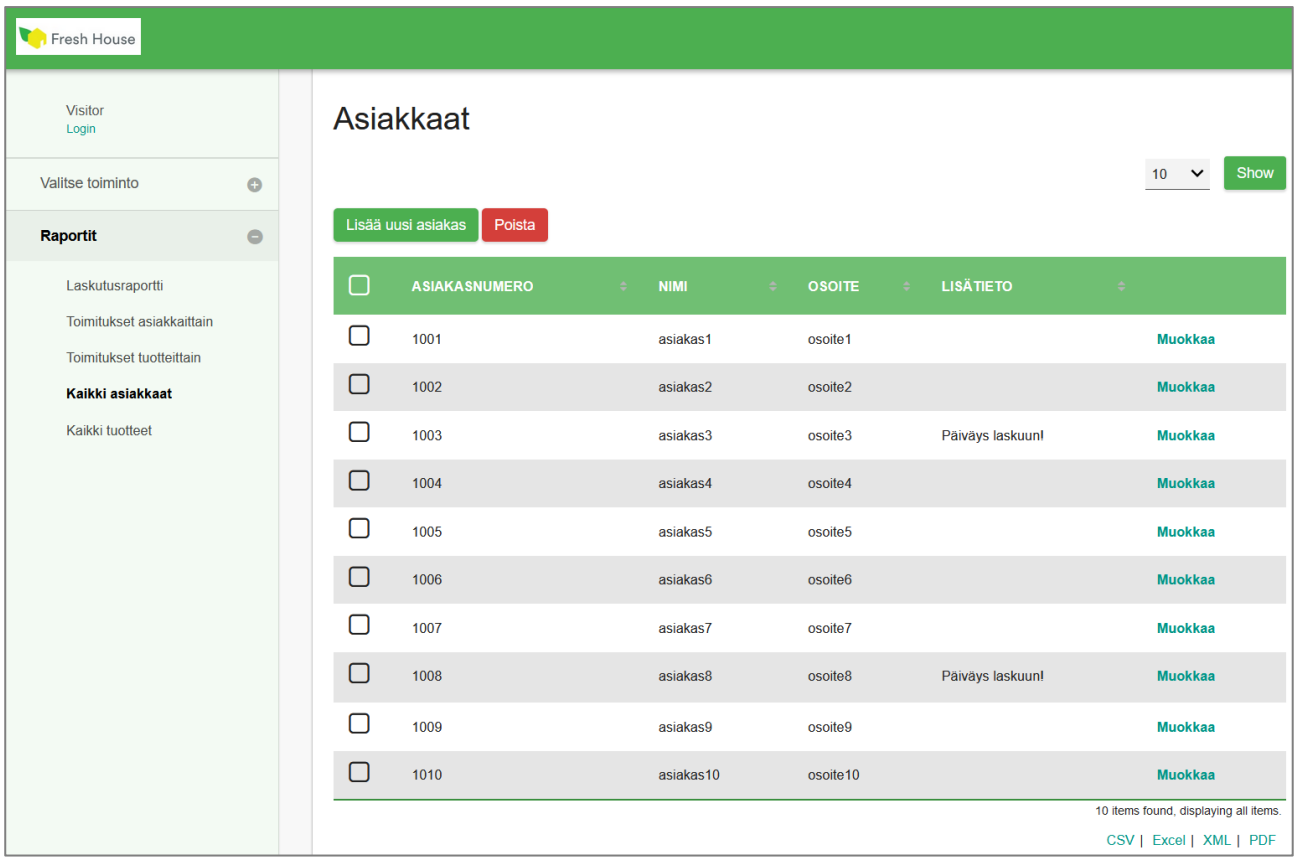

Fresh House Oy
Login
Tuotteet:
Etusivulle
Nimike
Hinta
Laskutettava
Tuote1 7.99 K
Tuote2 5.25 K
Tuote3 0.00 E
Tuote4 56.99 K
Tuote5 1.33 K
Tuote6 0.00 E
Tuote7 6.00 K
Tuote8 1.55 K
Lisää uusi tuote

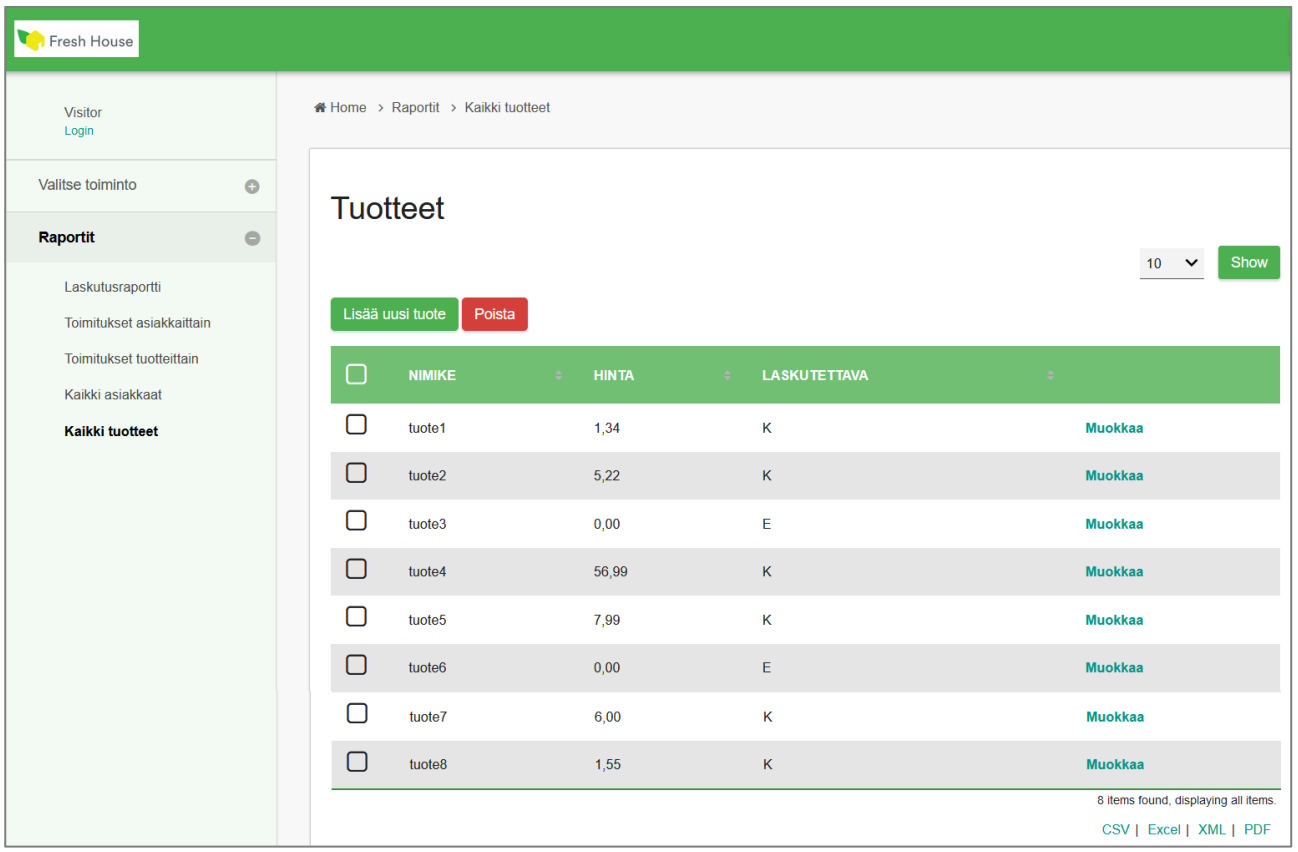
Liite 5: Joget DX -alustalla toteutetut sovelluksen käyttöliittymäruudut

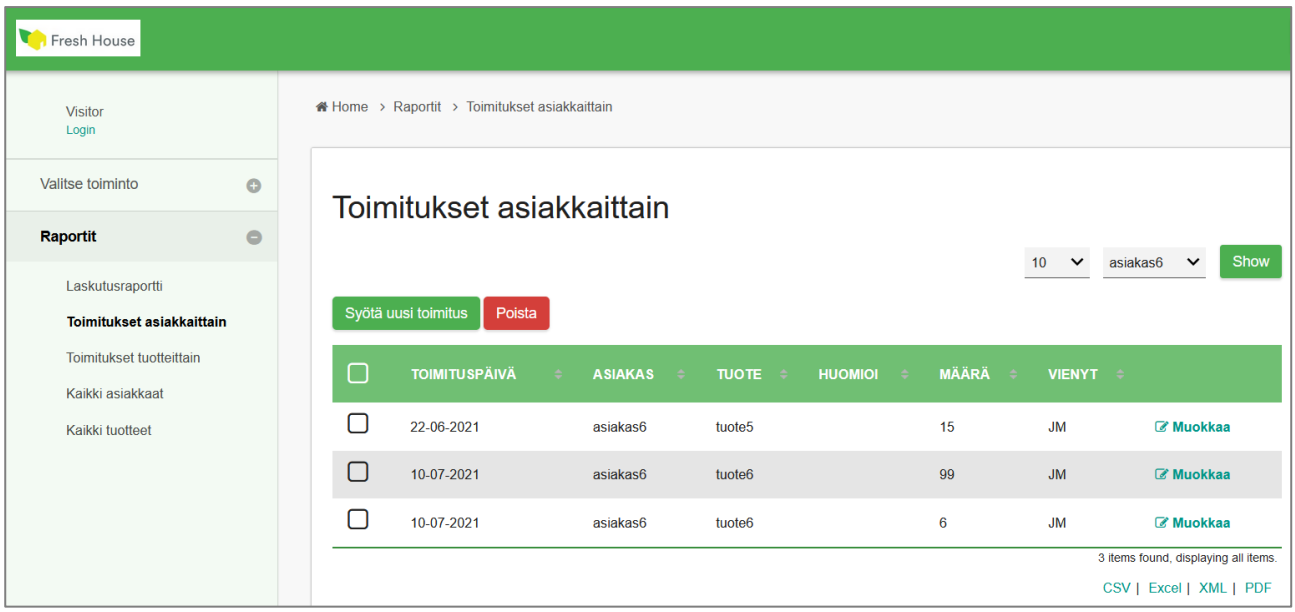


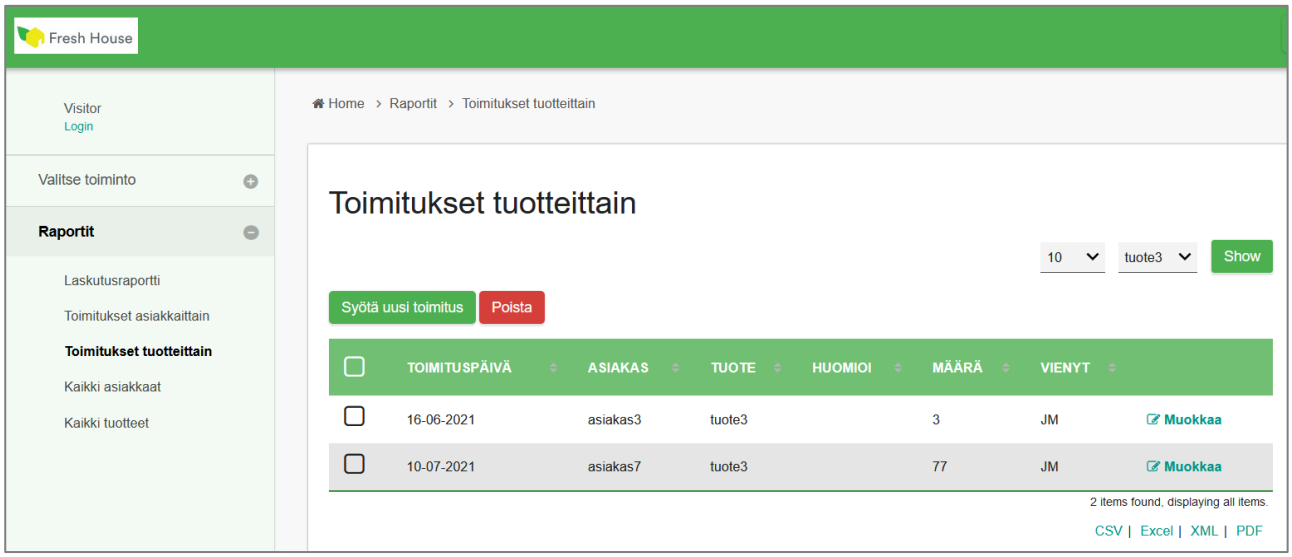


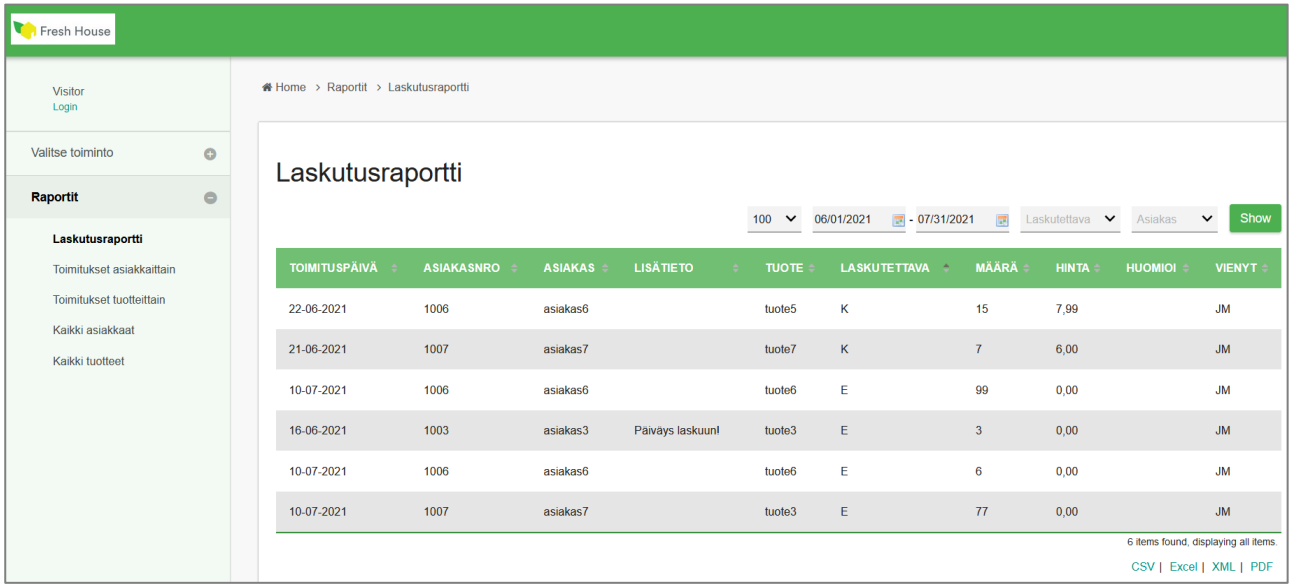



Fresh House
Visitor
Login
Valitse toiminto
Raportit
Laskutusraportti
Toimitukset asiakkaittain
Toimitukset tuotteittain
Kaikki asiakkaat
Kaikki tuotteet
Asiakkaat
10
Show
Lisää uusi asiakas
Poista
ASIAKASNUMERO
NIMI
OSOITE
LISÄTIETO
1001 asiakas1 osoite1 Muokkaa
1002 asiakas2 osoite2 Muokkaa
1003 asiakas3 osoite3 Päiväys laskuun! Muokkaa
1004 asiakas4 osoite4 Muokkaa
1005 asiakas5 osoite5 Muokkaa
1006 asiakas6 osoite6 Muokkaa
1007 asiakas7 osoite7 Muokkaa
1008 asiakas8 osoite8 Päiväys laskuun! Muokkaa
1009 asiakas9 osoite9 Muokkaa
1010 asiakas10 osoite10 Muokkaa
10 items found, displaying all items.
CSV | Excel | XML | PDF

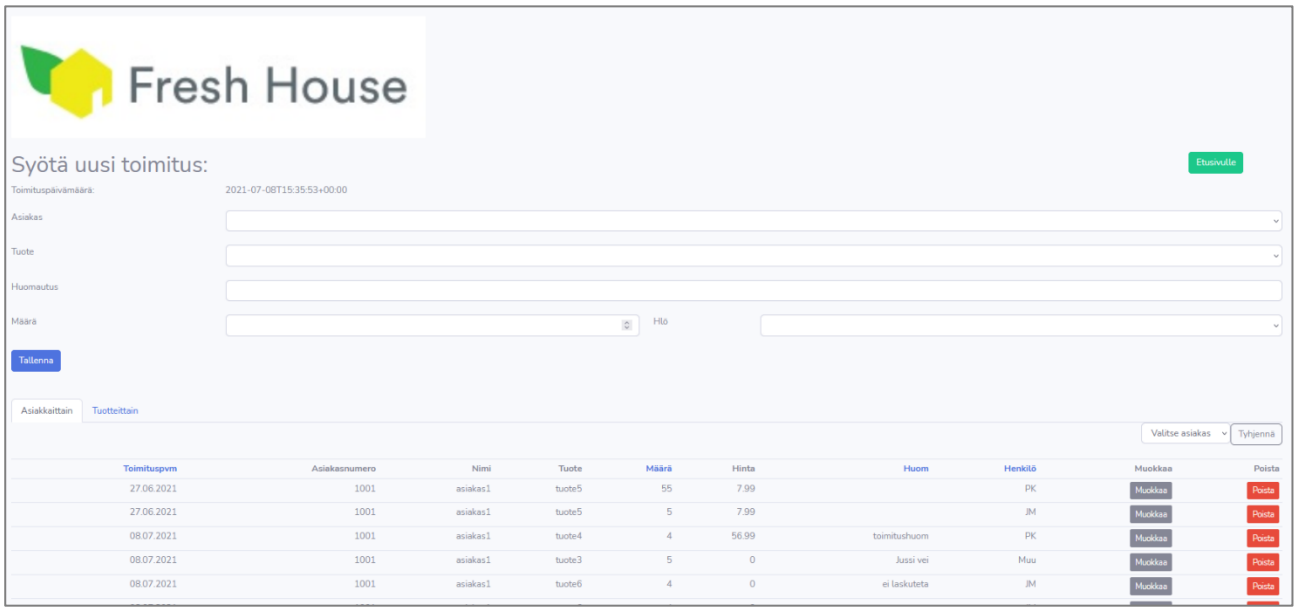

Fresh House
Visitor
Login
Valitse toiminto
Raportit
Laskutusraportti
Toimitukset asiakkaittain
Toimitukset tuotteittain
Kaikki asiakkaat
Kaikki tuotteet
Home › Raportit › Kaikki tuotteet
Tuotteet
10
Show
Lisää uusi tuote
Poista
NIMIKE
HINTA
LASKUTETTAVA
tuote1 1,34 K Muokkaa
tuote2 5,22 K Muokkaa
tuote3 0,00 E Muokkaa
tuote4 56,99 K Muokkaa
tuote5 7,99 K Muokkaa
tuote6 0,00 E Muokkaa
tuote7 6,00 K Muokkaa
tuote8 1,55 K Muokkaa
8 items found, displaying all items.
CSV | Excel | XML | PDF

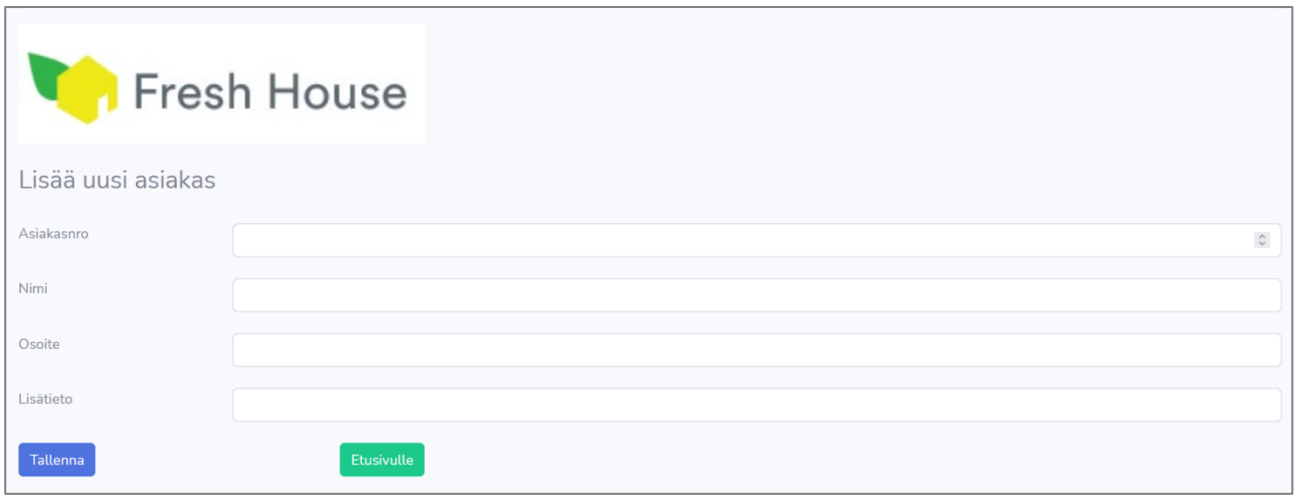

Fresh House
Visitor
Login
Valitse toiminto
Raportit
Laskutusraportti
Toimitukset asiakkaittain
Toimitukset tuotteittain
Kaikki asiakkaat
Kaikki tuotteet
Home › Raportit › Toimitukset asiakkaittain
Toimitukset asiakkaittain
10
asiakas6
Show
Syötä uusi toimitus
Poista
TOIMITUSPÄIVÄ
ASIAKAS
TUOTE
HUOMIOI
MÄÄRÄ
VIENYT
22-06-2021 asiakas6 tuote5 15 JM Muokkaa
10-07-2021 asiakas6 tuote6 99 JM Muokkaa
10-07-2021 asiakas6 tuote6 6 JM Muokkaa
3 items found, displaying all items.
CSV | Excel | XML | PDF

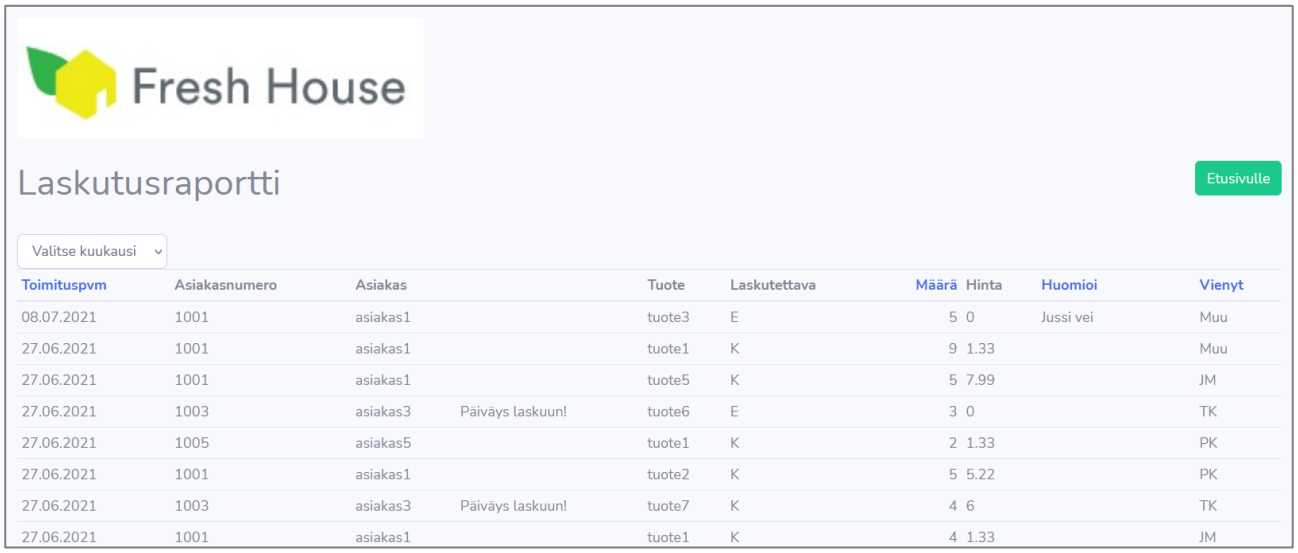

Fresh House
Visitor
Login
Valitse toiminto
Raportit
Laskutusraportti
Toimitukset asiakkaittain
Toimitukset tuotteittain
Kaikki asiakkaat
Kaikki tuotteet
Home › Raportit › Toimitukset tuotteittain
Toimitukset tuotteittain
10
tuote3
Show
Syötä uusi toimitus
Poista
TOIMITUSPÄIVÄ
ASIAKAS
TUOTE
HUOMIOI
MÄÄRÄ
VIENYT
16-06-2021 asiakas3 tuote3 3 JM Muokkaa
10-07-2021 asiakas7 tuote3 77 JM Muokkaa
2 items found, displaying all items.
CSV | Excel | XML | PDF


Fresh House
Visitor
Login
Valitse toiminto
Raportit
Laskutusraportti
Toimitukset asiakkaittain
Toimitukset tuotteittain
Kaikki asiakkaat
Kaikki tuotteet
Home › Raportit › Laskutusraportti
Laskutusraportti
100
06/01/2021 - 07/31/2021
Laskutettava
Asiakas
Show
TOIMITUSPÄIVÄ ASIAKASNRO ASIAKAS LISÄTIETO TUOTE LASKUTETTAVA MÄÄRÄ HINTA HUOMIOI VIENYT
22-06-2021 1006 asiakas6 tuote5 K 15 7,99 JM
21-06-2021 1007 asiakas7 tuote7 K 7 6,00 JM
10-07-2021 1006 asiakas6 tuote6 E 99 0,00 JM
16-06-2021 1003 asiakas3 Päiväys laskuun! tuote3 E 3 0,00 JM
10-07-2021 1006 asiakas6 tuote6 E 6 0,00 JM
10-07-2021 1007 asiakas7 tuote3 E 77 0,00 JM
6 items found, displaying all items.
CSV | Excel | XML | PDF

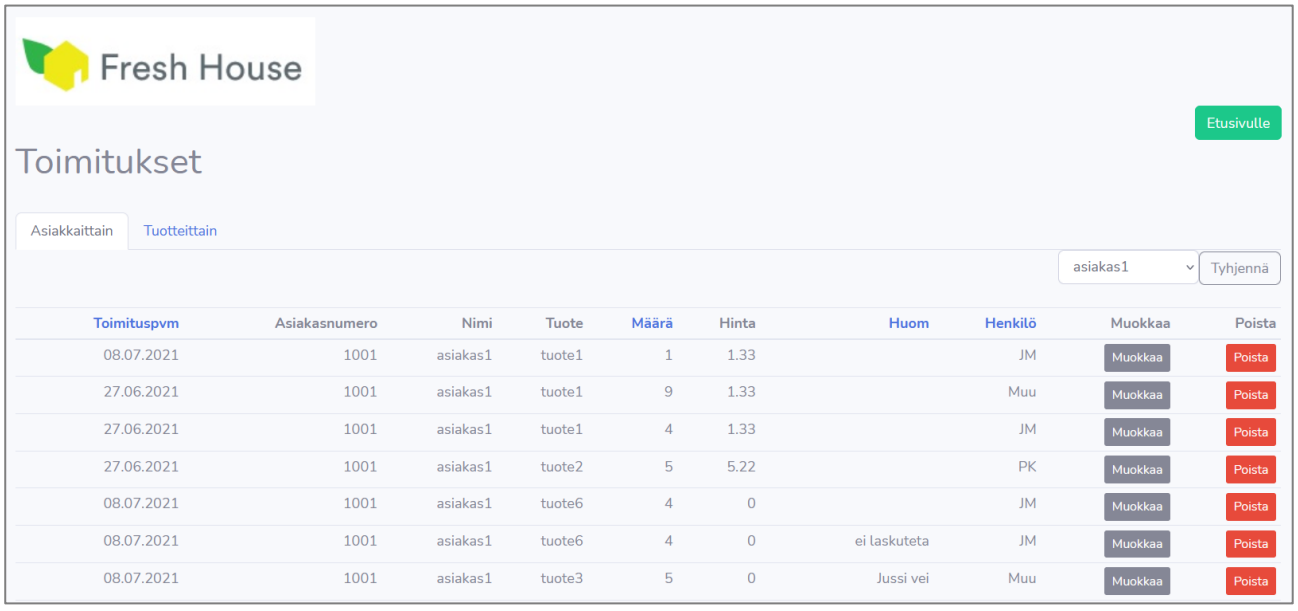
Liite 6: Saltcorn-alustalla toteutetut sovelluksen käyttöliittymäruudut



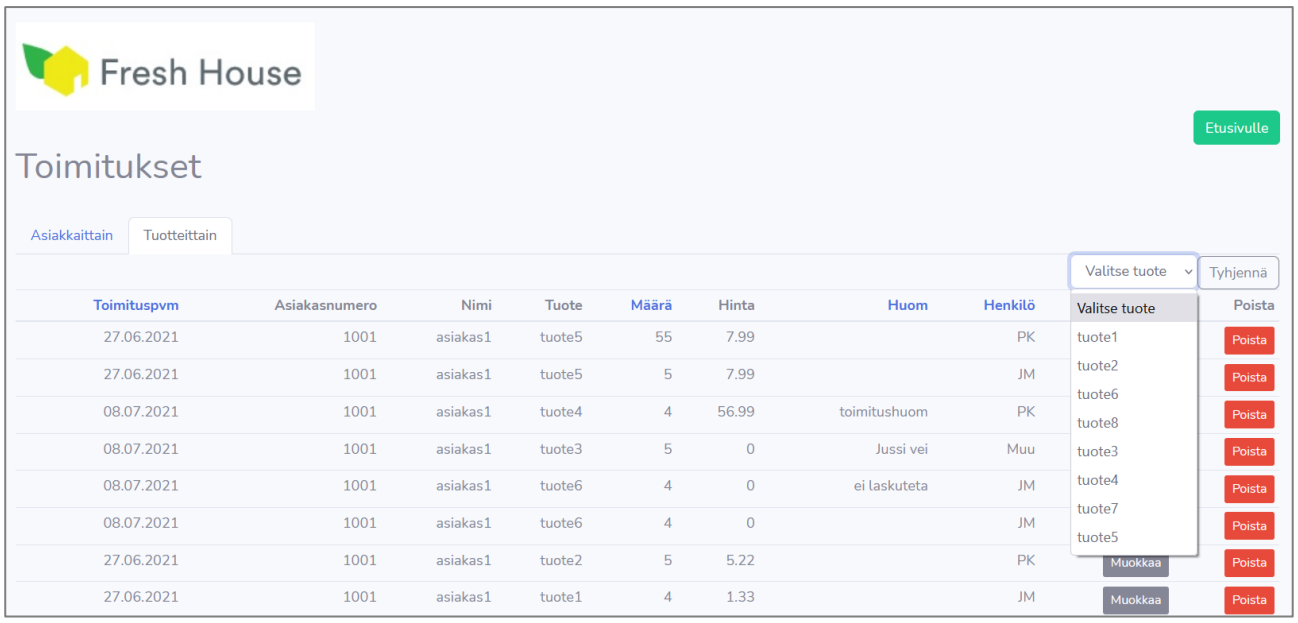


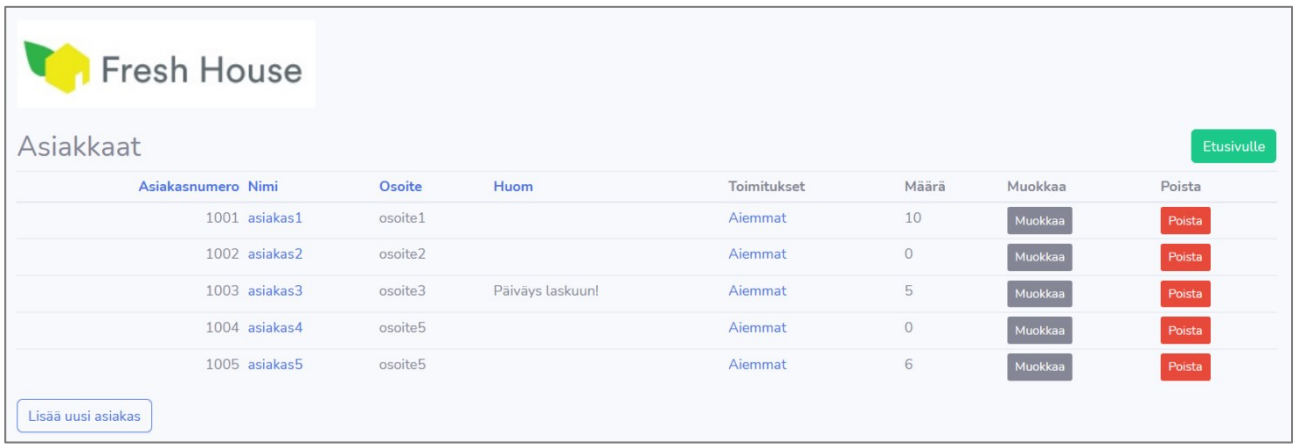


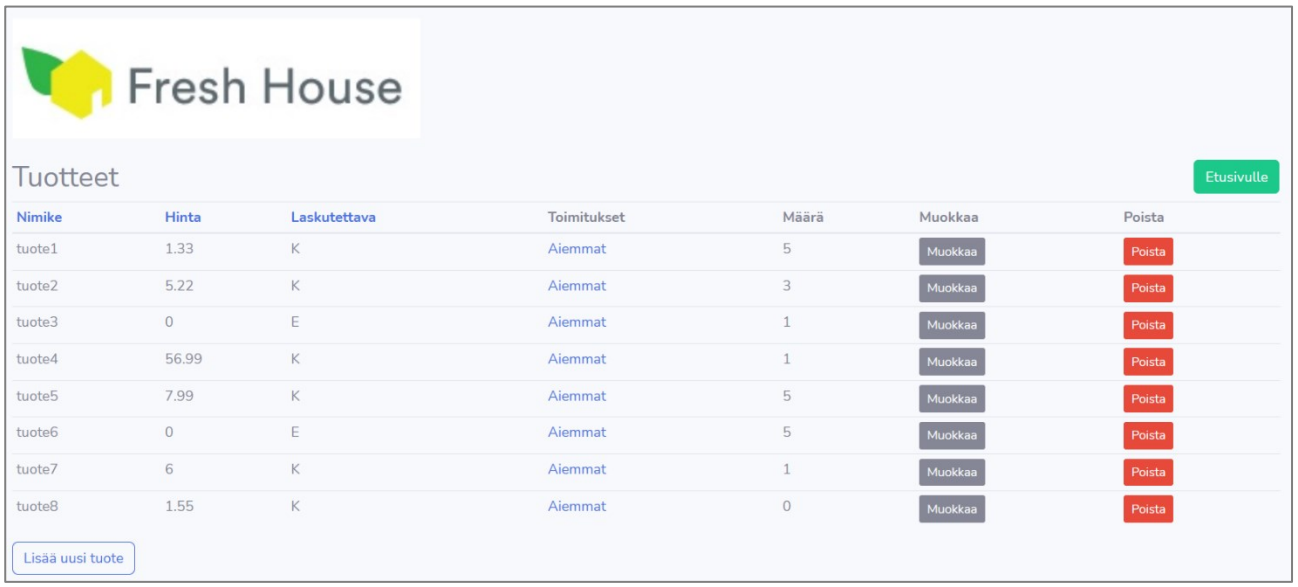
Toimituspvm	Asiakasnumero	Asiakas		Tuote	Laskutettava	Määrä	Hinta	Huomioi	Vienyt
08.07.2021	1001	asiakas1		tuote3	E	5	0	Jussi vei	Muu
27.06.2021	1001	asiakas1		tuote1	K	9	1.33		Muu
27.06.2021	1001	asiakas1		tuote5	K	5	7.99		JM
27.06.2021	1003	asiakas3	Päiväys laskuun!	tuote6	E	3	0		TK
27.06.2021	1005	asiakas5		tuote1	K	2	1.33		PK
27.06.2021	1001	asiakas1		tuote2	K	5	5.22		PK
27.06.2021	1003	asiakas3	Päiväys laskuun!	tuote7	K	4	6		TK
27.06.2021	1001	asiakas1		tuote1	K	4	1.33		JM



Toimituspvm	Asiakasnumero	Nimi	Tuote	Määrä	Hinta	Huom	Henkilö	Muokkaa	Poista
08.07.2021	1001	asiakas1	tuote1	1	1.33		JM	Muokkaa	Poista
27.06.2021	1001	asiakas1	tuote1	9	1.33		Muu	Muokkaa	Poista
27.06.2021	1001	asiakas1	tuote1	4	1.33		JM	Muokkaa	Poista
27.06.2021	1001	asiakas1	tuote2	5	5.22		PK	Muokkaa	Poista
08.07.2021	1001	asiakas1	tuote6	4	0		JM	Muokkaa	Poista
08.07.2021	1001	asiakas1	tuote6	4	0	ei laskuteta	JM	Muokkaa	Poista
08.07.2021	1001	asiakas1	tuote3	5	0	Jussi vei	Muu	Muokkaa	Poista


Fresh House
Etusivulle
Toimitukset
Asiakkaittain
Tuotteittain
Valitse tuote
Tyhjennä
Toimituspvm
Asiakasnumero
Nimi
Tuote
Määrä
Hinta
Huom
Henkilö
Poista
27.06.2021 1001 asiakas1 tuote5 55 7.99 PK
27.06.2021 1001 asiakas1 tuote5 5 7.99 JM
08.07.2021 1001 asiakas1 tuote4 4 56.99 toimitushuom PK
08.07.2021 1001 asiakas1 tuote3 5 0 Jussi vei Muu
08.07.2021 1001 asiakas1 tuote6 4 0 ei laskuteta JM
08.07.2021 1001 asiakas1 tuote6 4 0 JM
27.06.2021 1001 asiakas1 tuote2 5 5.22 PK
27.06.2021 1001 asiakas1 tuote1 4 1.33 JM
Valitse tuote
tuote1
tuote2
tuote6
tuote8
tuote3
tuote4
tuote7
tuote5
Muokkaa
Poista


Fresh House
Asiakkaat
Etusivulle
Asiakasnumero
Nimi
Osoite
Huom
Toimitukset
Määrä
Muokkaa
Poista
1001 asiakas1 osoite1 Aiemmat 10
1002 asiakas2 osoite2 Aiemmat 0
1003 asiakas3 osoite3 Päiväys laskuun! Aiemmat 5
1004 asiakas4 osoite5 Aiemmat 0
1005 asiakas5 osoite5 Aiemmat 6
Muokkaa
Poista
Lisää uusi asiakas


Fresh House
Tuotteet
Etusivulle
Nimike
Hinta
Laskutettava
Toimitukset
Määrä
Muokkaa
Poista
tuote1 1.33 K Aiemmat 5
tuote2 5.22 K Aiemmat 3
tuote3 0 E Aiemmat 1
tuote4 56.99 K Aiemmat 1
tuote5 7.99 K Aiemmat 5
tuote6 0 E Aiemmat 5
tuote7 6 K Aiemmat 1
tuote8 1.55 K Aiemmat 0
Muokkaa
Poista
Lisää uusi tuote