LAB University of Applied Sciences

Low-coden ja no-coden hyödyntämien pien-yrittäjän tukena

Hääkuvaajan ajanvaraus

LAB-ammattikorkeakoulu
Tradenomi (AMK)
2025
Veeti Kukkonen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Kukkonen, Veeti	Opinnäytetyö, AMK	2025
	Sivumäärä 35	

Työn nimi

Low-coden ja no-coden hyödyntäminen pienyrittäjän tukena

Hääkuvaajan ajanvaraus

Tutkinto ja koulutusala

Tradenomi (AMK)

Toimeksiantajaorganisaatio (jos opinnäytetyöllä on toimeksiantaja)

Tiivistelmä

Opinnäytetyön päätavoite on selvittää voivatko yksinyrittäjät, jotka omaavat hyvät perus IT-taidot hyödyntää low-code-sekä no-code-työkaluja luodakseen liiketoimintaa tukevia sovelluksia omaan käyttöönsä.

Työn toiminnallisena osuutena toteutettiin hääkuvaajalle ajanvarausjärjestelmä, jossa kuvaaja pystyy hallinnoimaan varauksia sekä laskutusta käyttäen AppSheet-työkalua. Toteutus sisälsi suunnittelun, rakentamisen ja käyttökokemuksen arvioinnin.

Tulokset osoittavat, että on hyvinkin mahdollista luoda no-code-työkaluilla sovelluksia omaan käyttöön, vaikkei aikaisempaa sovelluskehityskokemusta olisikaan. Aikaisemmasta kokemuksesta on kuitenkin todella paljon hyötyä. Työ tukee näkemystä siitä, että no-code-työkaluilla on mahdollista rakentaa liiketoimintaa tukevia työkaluja ilman syvää teknillistä osaamista.

Asiasanat

low-code, no-code, yksinyrittäjä, hääkuvaus, varausjärjestelmä

Abstract

Author(s)	Type of Publication	Published
Kukkonen, Veeti	Thesis, UAS	2025
	Number of Pages 35	

Title of Publication

Usage of low-code and no-code for small businesses

Wedding photographers' reservation system

Degree, Field of Study

Bachelor of Business Administration (UAS)

Name, title and organization of the client

Abstract

The main goal of the thesis was to find out if small businesses and entrepreneurs with good basic IT-skills can use no-code and no-code platforms to create software that supports their daily operations. In this thesis a booking system was created using AppSheet to demonstrate the creation of so-called applications. The application allows the user to manage reservations and handle invoicing.

The results show that it is possible to create software with no prior experience in software development and that also it is very useful to have some experience or at least interest in the subject. The findings support the idea that no-code platforms really are a viable way for small businesses and entrepreneurs to create supportive software for their own use with minimal technical expertise.

Keywords

Low-code, no-code, entrepreneur, small business, reservation system

Sisällys

1 Johdanto 1
2 Low-code 3
2.1 Low-coden taustaa 3
2.2 Low-code vs no-code 3
2.2.1 Eroavaisuudet 4
2.2.2 Yhteenveto 5
2.3 Low-coden hyödyt 6
2.4 Low-coden ja no-coden haasteet 7
2.5 Visuaalinen sovelluskehitys ja pienyrittäjät 8
2.5.1 Hyödyt pienyrittäjän näkökulmasta 8
2.5.2 Haasteet pienyrittäjä näkökulmasta 8
2.5.3 Low-code vai no-code 9
3 Vertailu 10
3.1 Vertailun kriteerit 10
3.2 Vertailun kohteet 10
3.2.1 PowerApps 11
3.2.2 Zoho 11
3.2.3 AppSheets 11
3.3 Yhteenveto 12
4 Tutkimusmenetelmät 14
5 Tutkimuksen kulku 15
5.1 Vaatimusmäärittely 15
5.2 AppSheet-projektin aloitus 17
5.3 Rakenteen kuvaus 20
5.3.1 Näkymät 21
5.3.2 Taulut 23
5.3.3 Toiminnot 24
5.3.4 Automaatio 26
5.4 Haasteet ja opit 28
5.4.1 Teknisiä haasteita 28
5.4.2 Oppeja 29
5.5 Arvio työstä 29
5.5.1 Työn esittely 30
5.5.2 Varauksen elinkaari 31

5.5.3 Jatkuva kehitys 32
5.6 Tulosten yhteenveto 33
6 Yhteenveto ja pohdinta 34
Lähteet 36

Liite 1. Google Sheets-skripti sähköpostiautomaatioon

1 Johdanto

Teknologia kehittyy valtavalla vauhdilla ja on muokannut jo maailmaa ennennäkemättömällä nopeudella. Low-code-työkalut ovat vakiintuneet merkittäväksi osaksi ohjelmistokehitystä. Suuret ohjelmistotalot ovat jo usean vuoden ajan hyödyntäneet low-code- ja no-code-työkaluja suurissa ohjelmistoprojekteissa, ja niiden käytön tuomat edut ovat olleet tiedossa jo pitkään. Pienyrittäjien kohdalla työkalujen mahdollisuudet ovat kuitenkin vasta tulossa heidän tietoisuuteensa. Pienyrittäjillä ei ole käytössään erillistä IT-osastoa, josta löytyisi IT-osaamista moneen eri lähtöön, joten helposti käytettävät ja kustannustehokkaat low-code- ja no-code-ratkaisut voivat olla merkittävä hyöty heidän arjessaan.

Monella pienyrittäjällä on arjessaan paljon manuaalista työtä vaativia asioita sekä arjen askareita, joita voisi erilaisia työkaluja hyödyntäen vähentää. Esimerkiksi ajanvaraus tapahtuu usein vielä manuaalisesti sähköposteja lueskellen sekä lähettäen ja vie paljon aikaa arjen muista tehtävistä. Ajanvaraus on monelle pienyrittäjälle kuten kampaajille, valokuvaajille sekä hierojille keskeinen osa liiketoimintaa ja sen toimivuus on asiakastyytyväisyyden kannalta elintärkeää. Perinteisten ajanvarauskäytänteiden kuten sähköpostien sekä viestiketjujen perässä pysyminen on haastavaa sekä työlästä altistaen virheille ja nostaen yrittäjän stressitasoja. Low-code ja no-code-työkalujen avulla prosessista voisi tehdä yrittäjälle paljon huolivapaamman ja samalla vapauttaen aikaa muille liiketoiminnan tärkeille osa-alueille.

Tämän opinnäytetyön tavoitteena on selvittää, miten pienyrittäjä voi hyödyntää low-code-ja no-code-työkaluja oman liiketoimintansa tukena ilman merkittävää ohjelmointiosaamista tai suuria investointeja. Työssä tarkastellaan erityisesti sitä, miten mainittuja työkaluja voidaan hyödyntää arkisten prosessien, kuten ajanvarauksen tehostamisessa. Työ perustuu oletukseen, jossa yrittäjällä on hyvät perustason IT-taidot: hän osaa hakea tietoa verkosta, käyttää sähköpostia tehokkaasti ja hallitsee toimistosovellusten perustoiminnot.

Opinnäytetyön tutkimuskysymykset ovat

- Millaisia ominaisuuksia ja hyötyjä low-code- ja no-code-työkalut tarjoavat yksinyrittäjän arjessa?
- Miten low-code- ja no-code-työkalut soveltuvat erityisesti ajanvarausprosessien luomiseen ja kehitykseen pienyrityksissä?
- Miten hyvillä perustason IT-taidoilla varustettu yrittäjä pystyy hyödyntämään low-code- ja no-code-työkaluja oman arkensa helpottamiseksi?
- Onko mahdollista luoda toimiva ajanvarausjärjestelmä low-code- tai no-code-työkaluja käyttäen ilman ohjelmointiosaamista?

Opinnäytetyössä toiminnallisena osuutena toteutetaan ajanvarausjärjestelmä kuvitteelliselle hääkuvaajana toimivalle valokuvaajalle. Ajanvarausjärjestelmä on hyvä esimerkki toistuvasta manuaalisesta työstä, jossa no-code sekä low-code työkaluja voi hyödyntää.

Työssä alustetaan low-code-työkalujen mahdollisuuksia sekä vertaillaan eri alustoja soveltuvuuden, käytettävyyden ja kustannusten näkökulmasta. Low-coden sekä no-coden välinen raja on häilyvä ja osa työkaluista, jotka alkuun vaikuttavat tai mainostavat olevansa no-code työkaluja voivatkin olla low-coden määritelmään mahtuvia työkaluja. Tämän työn toiminnallisessa osuudessa hyödynnetään lähtökohtaisesti määriteltyyn tehtävään soveltuvinta työkalua jäämättä jumiin kyseisten termien väliseen terminologiaan, jolloin valittava työkalu voi olla myös no-code työkalu, joka sisältää low-code ominaisuuksia.

Termejä pienyrittäjä ja yksinyrittäjä käytetään osittain ristiin työn aikana. Tietoperustassa puhutaan pienyrittäjistä laajemmin ja toiminnallisessa osuudessa näkökanta keskittyy juuri yksinyrittäjän näkökulmaan. Molemmat viittaavat kuitenkin yrittäjään, jolla ei ole käytössä omaa IT-osastoa.

2 Low-code

2.1 Low-coden taustaa

Low-code ja no-code-kehitys alkoi 1970–1990-luvuilla termin ”low-code” ensimaininnan ollessa vuonna 2011 raportissa. Low-coden juuret ovat 2000-luvun alun RAD (rapid application platforms) noustessa pinnalle. Forrester käytti ensimmäisenä low-code-termiä vuonna 2014 viitaten alustoihin, jotka keskittyivät yksinkertaisuuteen sekä helppokäyttöisyyteen. Wordpressin lanseeraus vuonna 2003 ja Formstackin vuonna 2006 olivat ensimmäisiä no-code web-työkaluja. (Kissflow 2025a; Ogilvy 2023.)

Low-code ohjelmistokehitys on lähestymistapa, jolla minimoidaan perinteisen ohjelmoinnin osaamisen sekä eri ohjelmointikielien osaamisen tarvetta (PCMAG 2018). Low-code-työkalut mahdollistavat ohjelmistojen luomisen visuaalisten työkalujen kuten drag and drop-komponenttien, erilaisten pohjien sekä automaation avulla. Näitä työkaluja hyödyntäen ohjelmistokehitystä voidaan ketteröittää ja nopeuttaa ilman syvällistä teknistä osaamista. Low-code ratkaisut ovat erityisen toimivia organisaatioiden sisäisten sovellusten luomiseen sekä prototyyppien tekoon niiden nopean tuotantotahdin vuoksi. (Roivainen 2025.)

Näiden työkalujen avulla sekä erilaisia luotuja tai valmiita komponentteja yhdistämällä pystytään erittäin ketterään sovelluskehitykseen, joissa skaalautuvuus on myös erinomaista eli valmiin workflown pystyy valmiina jakamaan useille käyttäjille sellaisenaan suoraan. Esimerkiksi Microsoftin Power Appsia käyttämällä luodaan automaatioita, jotka nopeuttavat tiedostojen jakamista sekä hallintaa käyttäen apuna käyttäjän sähköpostia, jonka ympärille luodaan sääntöjä käyttäen apuna low-code ohjelmistokehitystä. (Microsoft.)

2.2 Low-code vs no-code

Low-code ja no-code ovat hyvin lähellä toisiaan olevia visuaalisuutta sekä käyttäjäystävällisyyttä korostavia työkaluja ohjelmistojen kehitykseen. muistuttavat toisiaan erittäin paljon ja näiden työkalujen kanssa työskentely ovat todella lähellä toisiaan. Molempien kehitysmallien tavoitteena on luoda ohjelmistokehityksestä tavoitettavampaa sekä tehokkaampaa. (Jednaszewski 2024.)

Usein termistö hieman sekoittuu yleiskielessä, minkä seurauksena low-code sekä no-code nivoutuu monesti saman käsitteen alle. Termit eivät kuitenkaan ole täysin sama asia vaan näiden väliltä löytyy eroavaisuuksia, vaikka ovatkin hyvin saman kaltaisia ohjelmistokehityksen tapoja.

2.2.1 Eroavaisuudet

Eroavaisuuksia löytyy jo kohdeyleisöstä. Low-coden suuntautuessa enemmän ammattimaiseen jo vakiintuneeseen kehitys ympäristöön, on no-code lähtökohtaisesti suunnattu loppukäyttäjälle pitäen kehitysympäristön helppokäyttöisenä mahdollistaen yksinkertaisten sovellusten rakentamisen paremman workflown saavuttamiseksi.

> *Ne ovat hyvä ratkaisu, jos tarvitset yksinkertaisten sovellusten kehittämistä, jotka vaativat vain vähän tai ei lainkaan räätälöintiä ja perustuvat yksinkertaisen työnkulun tehostamiseen* (Outsystems.)

No-code-alustat ovat siis suunnattu käyttäjille ilman teknistä osaamista. Niiden avulla voi luoda yksinkertaisia sovelluksia kuten lomakkeita sekä aikataulutukseen liittyviä sovelluksia. Monesti alustat ei sisällä mahdollisuuksia lisätä omaa koodia mikä tekee siitä helppokäyttöisen mutta samalla rajaa käyttömahdollisuuksia. (Snyder 2023.)

Low-coden käyttämisessä toivottavaa olisi, jos kehittäjä omaisi jonkinnäköistä ohjelmointitaustaa. Se onkin suunnattu enemmän yksittäisen käyttäjän sijaan organisaatioiden käyttöön vähentämään ohjelmistojen kehitykseen käytettyä aikaa sekä vähentämään riippuvuutta ohjelmistokehittäjiin sekä IT-tiimeihin. Low-code myös tukee vahvasti tiimien välistä yhteistyötä tarjoten joustavampaa sovelluskehitystä tehden siitä tavoitettavampaa laajemmalle osalle tiimejä. (Comidor 2024.)

2.2.2 Yhteenveto

Yhteenvetona low-coden ja no-coden eroavaisuuksista voidaan huomata, että merkittäviä eroja löytyy kohderyhmästä ja kehitettävän sovelluksen muokattavuudesta sekä ominaisuuksien laajuudesta. Alle taulukkomuodossa (Taulukko 1) on esitettynä Low- sekä no-coden suurimmat eroavaisuudet:

	Low-code	**No-code**
Kohderyhmä	Kehittäjät	Liiketoimintakäyttäjät
Päätavoite	Kehityksen nopeuttaminen	Helppokäyttöisyys
Koodaamisen tarve	Matala, mutta olemassa	Ei vaadi
Muokattavuus	Täysi muokattavuus mahdollista	Valmiita mallipohjia voi muokata
Alustasidonnaisuus	Vapaa siirtyminen eri alustojen välillä	Usein sidottu tiettyyn alustaan
Päätepisteestä toiseen kehitys	Kaikki alustat tarjoavat täyden kehitysketjun	Osa alustoista tarjoaa vain rajalliset ominaisuudet
Tarkoitus	Uuden sukupolven sovelluskehitystyökalu ammattilaiskehittäjille	Itsepalvelutyökalu liiketoiminnan käyttäjille
Sovellusten monimutkaisuus	Mahdollistaa monimutkaiset sovellukset	Mahdollistaa yksinkertaiset sovellukset
Kustannustehokkuus	Kustannustehokas yrityksille. Joilla on oma kehitystiimi	Kustannustehokas yrityksille. Joilla on IT-ruuhkaa ja paljon tarpeita

Taulukko 1. low-code ja no-coden keskeisimmät eroavaisuudet (Kissflow 2025b)

Eroavaisuudet siis ovat selkeät erityisesti käyttötarkoituksen osalta. No-coden avulla yksittäinen käyttäjä voi kehittää omaa arkeaan helpottavia sovelluksia sekä vähentää manuaalisiin työvaiheisiin käytettävää aikaa. Low-code puolestaan toimii otollisimmin enemmän tiimi- sekä organisaatiotasolla parantaen tuottavuutta sekä tiimien välistä yhteistyötä.

Molemmat voivat olla yksinyrittäjälle kelvollisia ratkaisuja mutta tämän vertailun perusteella no-code-ratkaisut sopivat todennäköisesti suurimmalle osalle low-code-ratkaisuja paremmin sen aloittelijaystävällisemmän oppimiskäyrän takia. Myös haasteiden vastaan tullessa no-code-työkalujen kanssa on helpompi toimia koodittoman kehityksen ansiosta, sillä virheiden etsintä koodin seasta voi olla haastavaa ammattilaisellekin. Myös no-coden nopea kehitystahti tukee yksinyrittäjän tarpeita paremmin.

2.3 Low-coden hyödyt

Low-coden hyötyjen ymmärtämiseksi kokonaisuutena on hyvä käydä läpi hyötyjä suurempienkin organisaatioiden näkökannalta ennen kuin syvennytään pienyrittäjiin.

Decrue (Decrue 2025) käy artikkelissan läpi seitsemän low-coden hyötyä, jotka keskittyvät enemmän suurempien organisaatioiden “painopisteisiin”.

- Tuote nopeammin markkinoille tuominen
    - Mahdollistaa prototyyppimaisen tuotannon, jolloin tuote voidaan saada käyttöön jo viikoissa tai päivissä.
- Tuotantokustannusten alentuminen
    - Vähentää käsin tehtävää koodausta, joka on kallista ja työlästä.
- Tiimien välisen yhteistyön paraneminen
    - Muut tiimit pystyvät paremmin olemaan mukana kehitystyössä ja jopa osallistumaan siihen aktiivisesti.
- Joustavuuden ja mukautettavuuden paraneminen
    - Vaikka työkalut sisältävät valmiita komponentteja, voi niitä muokata ja hienosäätää omaan tarkoitukseen tai lisätä omaa koodia sekaan mahdollistaen juuri sopivan sovelluksen luomisen.
- Skaalautuvuuden paraneminen
    - Sovellukset voivat kasvaa automaattisesti yrityksen mukana ilman uudelleen kehitystä alusta-alkaen.
- Jatkuvan päivittämisen tulevaisuustakuun paraneminen

    - Alustoja päivitetään säännöllisesti ja uusia ominaisuuksia lisätään aktiivisesti. Näin pysyy uusimmissa trendeissä helposti mukana.

- Innovaation sekä kokeilujen määrän kasvattaminen
    - Rohkaisee kokeilemaan sekä innovaatioon sillä nopeasti toteutettavilla testisovelluksilla uuden luomisen testaaminen on kustannustehokasta.

2.4 Low-coden ja no-coden haasteet

Low-coden ja no-coden käyttämisellä ei vältytä myöskään haasteilta, jotka koskevat jokaista visuaalista ohjelmistokehitystä hyödyntävää organisaatiota. Vaikka low-code skaalautuukin alkuun hyvin, voi ongelmia tulla skaalautuvuuden kanssa pitkällä aikavälillä, kun projektien laajuus kasvaa ja erilaisia järjestelmiä tulee paljon yhteiskuvaan. Sama haaste voidaan havaita myös integraation osalta. Varsinkin legacy-järjestelmien kanssa työskentely low-code-työkaluilla voi olla hyvinkin haasteellista. Muita haasteita low-code-kehityksessä on rajoittunut kustomointi, joka muuttuu ongelmaksi sovelluksen vaatimusten ollessa todella yksityiskohtaisia sekä uniikkeja. Myös turvallisuushuolet nousevat keskusteluun alustojen ollessa kolmansien osapuolten toimittamia eikä kaikkien alustojen turvallisuusmäärittelyt välttämättä täytä alan standardeja. Pitkäjänteisen kehityksen osalta myös alustariippuvuus on suuri huoli kehityksen kannalta. Siirtymien alustalta toiselle tai oman alustan luominen low-code-alustan käytön jälkeen voi olla haasteellista rajoittaen joustavuutta ohjelmistokehityksessä. (Roivainen 2025.)

Vaikkakin low-code-työkaluja markkinoidaan helppokäyttöisinä työkaluina, joiden avulla sovelluskehitys onnistuu keneltä vain, on todellisuus kuitenkin se, että näidenkin työkalujen käyttö vaatii jonkinnäköistä teknillistä osaamista, loogista päättelykykyä sekä liiketoimintamallin ymmärtämistä. Kokenut kehittäjä voi saada käsityksen hyvinkin nopeasti, miten alustat toimivat mutta ensikertalaiselle low-code-työkalujen käyttäjälle alustojen opiskelu voi viedä kriittistä aikaa. Tällöin voidaan miettiä jo olemassa olevan IT-tiimin kannalta, onko järkevää kouluttaa henkilöstöä vai jatkaa jo olemassa olevaa kehitysmallia.

Kustannustehokkuus on yksi low-coden ehdottomista hyödyistä mutta voi olla myös kompastuskivenä, jos koodausosaamista ei tarvittaessa löydy organisaatiosta ollenkaan. Tällöin kustannukset voivat kasvaa korkeammiksi verrattaessa no-code-kehitysmallin kustannuksiin. (Microsoft.)

2.5 Visuaalinen sovelluskehitys ja pienyrittäjät

Pienyrittäjän näkökulmasta ei ole niin merkitystä onko työkalu low-code vai no-code vaan tärkeintä on sen soveltuvuus omaan käyttötarkoitukseen. Työkalun täytyy myös pystyä saamaan käyttäjä niin sanotusti sisään työympäristöön, jolloin alustan käyttö ei vaikuta kokemattomasta käyttäjästä liian kuormittavalta. Visuaalisessa sovelluskehityksessä päästään eroon pelottavista termeistä kuten frontend sekä backend, jotka helposti karkottavat uudet käyttäjät kokeilemasta sovelluskehitystä. Visuaalisen UI:n, valmiiden pohjien sekä pitkälle valmiiksi tehtyjen workflowien tarkoitus on laskea juuri tätä rimaa ja pitää kehitysvaihe mahdollisimman käyttäjäystävällisenä visuaalisuuden avulla. Alustan tulisi myös tukea jo tuttuja IT-ratkaisuja kuten Microsoft Office ympäristöä tai Google Workspacea, sillä tämä helpottaa käyttöönotossa valtavasti.

2.5.1 Hyödyt pienyrittäjän näkökulmasta

Snyderin (Snyder 2023) mukaan pienyrittäjän näkökulmasta low-code tarjoaa erityisesti riippumattomuutta sekä hallintaa omista järjestelmistään. Low-code sekä no-code mahdollistavat yrittäjän itsenäisen sovelluskehityksen, jolloin kolmansia osapuolia ei tarvita ja yrittäjä pystyy ketterästi mukauttamaan tarvittaessa omaa ympäristöään. Vaikka ohjelmointitaustaa ei vaadita, tuovat silti hyvät perustason IT-taidot paljon helpotusta low-code-ympäristöjen käyttöönotossa.

Monet low-code työkalut myös mahdollistavat mobiiliystävällisten sovellusten kehityksen automaattisesti. Erityisen tärkeää tämä on aloilla, joissa yrittävä liikkuu paljon tai käyttää puhelimellaan sovellusta. (PCMAG 2018.) Monet alustat ovat suunniteltu juuri siten, että eitekniset käyttäjät pystyvät hyödyntämään niitä, joka madaltaa myös kynnystä kokeilla itse tekemistä ulkoisen toimittajan sijaan samalla lisäten omistajuuden tunnetta yrittäjällä. (Quixy 2025.)

2.5.2 Haasteet pienyrittäjä näkökulmasta

Vaikka Low-code-työkalujen jatkuva kehitys ja työkalujen parantuminen on ohjelmistoalaa mullistava voima voi sen valjastaminen yksinyrittäjän tarpeisiin olla haastavaa, jos yrittäjä ei omaa mitään taustaa ohjelmoinnista tai ohjelmistotuotannosta. Ohjelmistotuotanto on muutakin kuin koodaamista, tarvitaan ymmärrystä käyttöliittymän kehittämisestä, käytettävyydestä, testauksesta, versionhallinnasta, dokumentaatiosta, eri järjestelmien

vaatimuksista sekä puutteista ja useista muista pienistä työkuluista, jotka vaikuttavat kuitenkin kriittisesti sovelluksen toimintaan.

Myöskään tämä asia ei tosin ole täysin mustavalkoinen vaan sisältää paljon harmaan eri sävyjä. Eri työkalut vaativat täysin erilaisia aloituslähtökohtia ja jotkin työkalut ovat huomattavasti toisia työkaluja helpommin lähestyttäviä uuden käyttäjän näkökulmasta.

Vaikka low-code-ohjelmistotuotannossa on paljon selkeitä vahvuuksia, on haasteet luonnollisesti osana tässäkin tuotantotyylissä. Pienyrittäjän arjessa haasteet ovat lähinnä uusien asioiden opetteluun sekä tietoruvaan/ datan hallintaan liittyviä, jolloin ongelmaksi voi nousta myös lainsäädäntö ohjelmistotuotannon parissa.

2.5.3 Low-code vai no-code

Kumpi näistä visuaalisista ohjelmistokehityksen tuotantotavoista voisi sopia paremmin yksiyrittäjä arkeen. Low-code tarjoaa parempia mukautusmahdollisuuksia sekä potentiaalisesti tukee yrityksen kasvua paremmin liiketoiminnan kehittyessä. Kuitenkin Suomessa yli 90 % pienyrittäjistä työllistää itsensä lisäksi vain muutaman työntekijän (Tilastokeskus 2023). Low-coden suurimmat hyödyt näkyvät selkeästi vasta, kun yrityksellä on useita tiimejä, jotka työskentelevät itsenäisesti kehittämällä näiden välistä yhteistyötä. Voidaan siis sanoa, että suurimmalle osalle yksiyrittäjistä low-code-työkalut eivät välttämättä toimi parhaimmalla mahdollisella tavalla, vaan juuri no-code-työkalut nousevat vahvuuksillaan esiin.

Vaikka no-code-kehitys tukeekin hyvin sovellusten kehitystä ilman teknistä taustaa, silti lähes 90 % käyttäjistä omaa jonkin asteen teknistä osaamista. Tämä kuvastaa alalla olevan vielä suurta kasvupotentiaalia edessään no-code-työkalujen käytössä. (Zapier 2022.) Tämä kyselytulos ohjaa myös siihen suuntaan, että pienyrittäjälle joka omaa hyvät IT-taidot olisi no-code-alusta suotavampi vaihtoehto sen ollessa helpommin lähestyttävä ympäristö verrattuna low-code-työkaluihin.

3 Vertailu

Eri low-code-työkalujen vertailussa keskitytään olennaisiin ominaisuuksin ja rajoituksiin toiminnallisen kehitysosuuden näkökulma huomioiden eli miten työkalu soveltuisi pienyrittäjän käyttöön. Vertailun kohteena ovat kolmea suosittua työkalua low-code ohjelmistokehitykselle, jotka ovat Microsoft PowerApps, Zoho creator ja Google AppSheet.

Työkalut valikoituivat siksi, koska Microsoftin ja Googlen palveluiden tunnettavuus tuo turvaa uudelle käyttäjälle, joka ei ole ennen näiden tyylisiä työkaluja käyttänyt sekä Zoho, joka on näistä kolmesta tuntemattomin tavalliselle tallaajalle mutta low-code-kehityksessä hyvin suosittu työkalu. Muita vaihtoehtoja olisi ollut muun muassa Mendix sekä Appian mutta ne karsiutuivat vertailun ulkopuolelle.

3.1 Vertailun kriteerit

Vertaillessa aiemmin mainittuja työkaluja keskiössä ovat käytettävyys, kustannukset, integraatio, ominaisuudet sekä tietoturva. Käytettävyydellä viitataan erityisesti käyttöliittymän selkeyteen, oppimiskäyrään sekä arvioon kuinka nopeasti sen käytön oppii. Ominaisuuksilla arvioidaan mitä valmiita ”moduuleita” työkalu sisältää. Integraation suhteen arvioidaan muiden laajasti käytössä olevien järjestelmien, kuten Google Workspace sekä M365 tuki.

Tässä työssä sivutaan myös eri työkalujen skaalautuvuutta mutta toiminnallista osiota silmällä pitäen se ei ole niin olennainen osa. Myös muita kriteerejä kuten tietoturvaa sivutaan niiden olematta keskiössä arvioinnin suhteen.

Kriteerit valikoituivat erityisesti pienyrittäjän näkökulmasta painottaen käytännön hyötyjä sekä helppokäyttöisyyttä. Tavoitteena löytää työkalu, jolla ajanvarausprosessin digitalisointi onnistuisi ilman laajaa teknistä osaamista.

3.2 Vertailun kohteet

Vertaillaan tarkemmin kolmea tarkastelun kohteeksi valikoituneita alustoja. Näistä kolmesta Microsoftin PowerApps sekä GoogleSheets sisältää sekä low-code, että no-code-ominaisuuksia riippuen siitä, mitä pyritään luomaan. Näiden työkalujen osalta voi olla hankala rajata kumpaan ne kuuluvat. Jokainen alusta pystyy toteuttamaan pienyrittäjän tarpeet ajanvarausjärjestelmän osalta. Tärkeäksi ominaisuudeksi muodostuukin käytön helppous, sillä helposti alun hankaluudet tai liian jyrkkä oppimiskäyrä pysäyttävät itsenäisen ohjelmistokehityksen alkutekijöihin.

3.2.1 PowerApps

PowerApps on osa Microsoftin Power Platform-ekosysteemiä, joka auttaa käyttäjää low-code-työkaluja hyödyntäen luomaan sovelluksia liiketoiminnan tarpeisiin. PowerAppsin ydin on Microsoftin ekosysteemi ja integraatio Microsoft 365 ympäristön, Sharepointin sekä Azuren palveluiden kanssa. Suomessa moni yritys käyttää jo O365 ympäristöä tai Sharepointia yrityksen intrana, jolloin PowerAppsin käyttö tällaisessa tilanteessa on ideaalia kaiken datan olevan jo samassa paikassa Microsoftin Dataversessä. (Alosevičius 2024; Microsoft 2024).

Käytettävyydeltään PowerApps tarjoaa Alosevičiusen (Alosevičius 2024) mukaan monipuolisia kehitysvaihtoehtoja. Kokemattomille kehittäjille drag-and-drop-työkalut auttavat alkuun pääsyssä Copilot AI-assistentin tukemana, joka auttaa kehittämään sovelluksia käyttäjän kanssa. Vaativampaa käyttöä varten voi myös hyödyntää Power FX-kieltä ja mukauttaa sovelluksia entistä enemmän. Tämä kuitenkin vaatii jo teknisempää osaamista.

3.2.2 Zoho

Zoho Creator on Zoho Corporationin kehittämä all-in-one sovelluskehitysalusta, joka on luotu auttamaan yrityksiä liiketoimintansa digitalisoinnissa. Yksi Zohon pääominaisuuksista on auttaa normalisoimaan sekä analysoimaan dataa ja luomaan erilaisia näkymiä datan visualisointia varten. Zoho tarjoaa pitkälti samoja ominaisuuksia kuin PowerApps sekä GoogleSheet tarjoten hyvät integraatiomahdollisuudet datalähteisiin ja hyvän käyttöliittymän sekä AI-assistantin avun sovelluskehitykseen. Käytettävyyden kannalta se on hieman PowerAppsiä helpompi mutta AppSheetsiä vaikeampi tarjoten paremmat mukautusvaihtoehdot.

Zohon hinnoittelu on selkeä ja hyvin kilpailukykyinen verrattuna kilpailijoihin ja samoin, kuin muillakin on vaativammat ominaisuudet maksumuurin takana. Muiden ominaisuuksien kannalta tarjoaa Zoho pitkälti PowerAppsiin verratut palvelut sisältäen oman kielen, jolla osavampi käyttäjä pystyy mukauttamaan sovelluksia tarpeiden mukaan. (Zoho.)

3.2.3 AppSheets

AppSheet on Googlen kehittämä no-code-työkalu, joka mahdollistaa sovellusten sekä automaatioiden nopean rakentamisen ilman koodausta. Automaattinen vahva integraatio Goole Workspacen kanssa auttaa käyttäjää yhdistämään datansa helposti. AppSheet on helppokäyttöinen ja sisältää hyvät automaatiovaihtoehdot. Se tunnetaan nopeudestaan ja tehokkuudestaan ja toimivuus Android laitteilla on erinomainen. (Alosevičius 2024.)

Myös Appsheetsin käyttäjän avuksi on tekoäly assistentti vahvasti osana alustaa, Googlen tapauksessa Gemini AI. Alusta on selvästi suunnattu nopeaa kehitystä varten, jonka ideana on pystyä rakentamaan automaatioita ja pieniä sovelluksia sekä julkaista niitä salamannopeasti yhtä nappia painamalla. AppSheet sisältää lukuisia mallipohjia, joita pystyy muokkaamaan omaan käyttöönsä sopivaksi sekä käyttöönottaa hyvin nopeasti.

Selvästi AppsSheet on suunniteltu pienempää yritystä silmällä pitäen kehityksen helppouden ja pienten kustannusten takia mutta myös suurempia organisaatioita varten on Enterprice Plus-paketti, joka sisältää paljon edistyksellisempiä ominaisuuksia tarjoamalla esimerkiksi tuen REST rajapintaan sekä koneoppimisen tuen. Parhaiten AppSheet kuitenkin tukee kevyitä sekä pienempiä kokonaisuuksia. (AppSheet.)

3.3 Yhteenveto

Taulukkoon kerättynä pienyrittäjän näkökulmasta tärkeimpiä kriteereitä, joita tarkastellaan uuden alustan käyttöönotossa (Taulukko 2).

Alusta	PowerApps	AppSheet	Zoho
Käytettävyys	Vaativampi	Erittäin helppo	Keskitaso
Hinta	20 USD/kk/käyttäjä	5 USD/kk/käyttäjä	8 USD/kk/käyttäjä
Ominaisuudet	Monipuoliset sovellukset, AI työkalut, laajat mukautusvaihtoehdot, skriptaus	Yksinkertaiset workflowt, automaatio, AI-assistant (maksullinen versio)	Skriptaus, automaatio, lomakkeet, raportointi
Integraatio	Microsoftin O365 ja Azure	Google workspace	Zoho-ekosysteemi, Zapier, rajapinnat sekä myös
Skaalautuvuus	Suuret yritykset	Pienet yritykset	Sopii sekä pienille, että suurille
Soveltuvuus pienyrittäjälle	Hyvä jos kokemusta	Erinomainen	Hyvä

Taulukko 2. Kooste vertailun kohteista (Alosevičius 2024; AppSheet; Microsoft 2024; Zoho)

Jokaisella alustalla on omat vahvuutensa ja kohderyhmä eroaa hieman toisistaan. Suurimmat erot löytyvät skaalautuvuuden ja ominaisuuksien, kuten muokattavuuden parista. Paras valinta riippuu yrityksen koosta ja tulevaisuuden suunnitelmista, jo olemassa olevasta teknologiaympäristöstä sekä käyttäjän omasta teknisestä osaamisesta.

Vertailun perusteella voidaan todeta, että kaikki tarkastellut työkalut tarjoavat hyvät mahdollisuudet hääkuvaajan ajanvarausjärjestelmän toteuttamiseen ilman syvällistä teknillistä osaamista. Jokaisella alustalla on toimiva integraatio sekä automaatiomahdollisuudet sähköpostin kuten G-mailin tai Outlookin kanssa, joka toimii pienyrittäjillä yhtenä tärkeimmistä asiakaskontaktoinnin tavoista.

Yksinyrittäjän näkökulmasta parhaiten soveltuva työkalu olisi ehdottomasti Google AppSheet, erityisesti, jos käytössä on jo valmiiksi Gmail tai Google Workspace. AppSheet mahdollistaa nopean kehityksen ilman koodaustaitoja ja integroituu erinomaisesti jo olemassa oleviin Googlen palveluihin. Se on nimenomaan suunniteltu pienille organisaatioille, johon yksinyrittäjä tässä tapauksessa määritellään. Jos kuitenkin yrittäjällä on entuudestaan kokemusta Microsoftin tuotteista tai tavoitteena on kasvattaa yritystä suuremmaksi kokonaisuudeksi Microsoft PowerApps voi pitkällä tähtäimellä tarjota paremmat mahdollisuudet jatkuvuuden kannalta.

4 Tutkimusmenetelmät

Tässä opinnäytetyössä käytetään toiminnallista tutkimusstrategiaa, silla tavoitteena on selvittää konkreettisesti, miten low-code sekä no-code työkaluja voi hyödyntää liiketoiminnan arjessa yrittäjän tukena. Toiminnallisessa osuudessa pyritään luomaan low-code- tai no-code-työkaluja käyttäen ajanvarausjärjestelmä kuvitteelliselle hääkuvauksia tekevälle valokuvaajalle. Näin saadaan käytännön kokemuksen avulla realistinen kuva siitä, mitä vaaditaan nimenomaisen järjestelmän luontiin tai onko sellaisen rakentaminen itse millään tavalla järkevää.

Kehitysprosessin aikana arvioidaan valitun työkalun soveltuvuutta uuden käyttäjän näkökulmasta ajanvarausjärjestelmän kehitykseen. Käytännön kokeilun

Analyysimenetelmänä opinnäytetyössä käytetään tavoiteperustaista arviointia suhteessa työn alussa määritettyihin tavoitteisiin. Arvioinnin keskiössä ovat erityisesti:

- käytettävyyden ja käyttäjäkokemuksen onnistuminen
- vaatimusmäärittelyn toteutuminen
- AppSheetin käyttökokemuksen arvioiminen
- Jatkokehitysmahdollisuuksien arvioiminen

Sovelluksen arviointi tapahtuu sen toiminnallisuuden ja vaatimusten toteutumisen perusteella. AppSheetin käyttökokemusta arvioidaan uuden käyttäjän näkökulmasta.

Työkaluksi valikoitui Google AppSheet sen matalan oppimiskynnyksen sekä helppokäyttöisyyden takia. AppSheet tarjoaa parhaimmat mahdollisuudet yksiyrittäjän lähtökohdat huomioiden toteuttaa kevyt sekä toimiva ajanvarausjärjestelmä hyödyntäen no-code-työkaluja.

5 Tutkimuksen kulku

Toiminnallisessa osiossa tavoitteena on luoda ajanvarausjärjestelmä pääasiassa hääkuvauksia tekevälle valokuvaajalle hyödyntäen Googlen AppSheet-työkalua. Kappaleessa käsitellään vaiheittain työnkulkua sekä sovelluksen kehittämisen vaiheita vaatimusmäärittelystä lähtien. Järjestelmää ei kuvata yksityiskohtaisesti vaan käydään yleisellä tasolla läpi, mitä vaiheita projektissa on pyrkien antamaan kokonaisvaltaisen kuvan sovelluksen kehityksestä Appsheetillä.

Odotuksena on kehittää varausjärjestelmä, jota hyödyntämällä valokuvaaja pystyy organisoimaan, tarkastelemaan sekä hallitsemaan varauksiaan helposti sekä vaivattomasti. Vaatimusmäärittelyssä määritellään ominaisuudet, joiden tulisi toimia lopullisessa työssä sekä tuodaan ilmi mahdollisia lisäkehityksen kohteita. AppSheet mahdollistaa jatkuvan kehityksen uusien ominaisuuksien nopealla kehittämisellä, joten opinnäytetyössä oleva järjestelmä rajataan pakollisiin perusominaisuuksiin. Työn tyylisen järjestelmän kehitystä voi jatkaa käytännössä loputtomiin parantaen jotain ominaisuutta vähän tai luomalla kokonaan uusia ominaisuuksia, jolloin rajaamatta jättäminen voi johtaa rönsyilevään lopputulokseen.

Lähtöasetelmana kuviteltu hääkuvaaja omistaa internet sivut, joiden kautta asiakkaat tavoittelevat kuvaajaa yhteydenottolomakkeen kautta. Tässä työssä asiakkaalla ei ole pääsyä järjestelmään vaan vain valokuvaajalla (myöhemmin käyttäjä) on pääsy sovellukseen ja muokata sen tietoja. Tämä sen takia, ettei sovelluksen tietototurvan sekä tietosuojan takia ole järkevää päästää asiakkaita järjestelmään käsiksi. Myös sovelluksen kehittämiseen menevä aika kasvaisi huomattavasti, mikä veisi osittain työn tarkoituksen, joka on olla helpottava työkalu, eikä suurta ylläpitoa vaativa järjestelmä. Sovelluksen kehityksessä ei huomioida lainsäädäntöä tietosuojan tai tietoturvan näkökulmasta. Tietoturvaa tai tietosuojaa ei myöskään erityisemmin huomioida sovelluksen kehitysvaiheessa.

5.1 Vaatimusmäärittely

Sovelluskehittimessäkin lopullisen työn laadun varmistamiseksi on hyvä tehdä vaatimusmäärittely halutuista ominaisuuksista. Tämä antaa sovellukselle suunnan jo heti kehityksen alkuvaiheessa, mikä helpottaa työskentelyvaiheita. Lisäksi myös näin saadaan helposti luotua dokumentaatiota tehdyistä ominaisuuksista. Vaatimusmäärittely jaetaan tässä työssä kahteen osaan, toiminnallinen ja ei-toiminnallinen.

Ei-toiminnallisia vaatimuksia työlle:

- helppokäyttöinen

- vain käyttäjällä pääsy sovellukseen
- kustannustehokkuus: ilmainen tai edullinen
- laajennettavuus/ jatkuva kehitys: sovellusta pystyy jatkokehittämään sekä lisäämään ominaisuuksia helposti
- ylläpidettävyys: rakenne ja logiikka yksinkertaisia eikä vaadi ohjelmointia
- responsiivisuus: toimii sekä tietokoneella että mobiilisovelluksena

Toiminnalliset vaatimukset (Taulukko 3) kuvaavat konkreettisesti, mitä sovelluksen tulee tehdä sekä käyttäjän käyttämät toiminnot.

Vaatimus	**Tyyppi**
Uuden varauksen luominen ja muokkaaminen	Pakollinen
Varauksen tilan muuttaminen	Pakollinen
Varaus näkyy kalenterissa	Pakollinen
Laskun luominen varauksesta	Pakollinen
Laskun kirjaaminen maksetuksi	Pakollinen
Automaattinen varauksen luominen sähköpostiautomaatiolla	Pakollinen
Sovellusmuistutukset	Pakollinen
Varausvahvistus asiakkaalle	Suositeltava (ilmaisversiossa vain käyttäjän sähköpostiin voi lähettää automaattisia viestejä)
Google kalenteri -integraatio	Mahdollinen
Tiedostoliitteet, inspiraatiokuvia	Mahdollinen
Raporttinäkymä	Mahdollinen
Laskun lähettäminen	Suositeltava
Automaattinen laskun lähetys	Mahdollinen

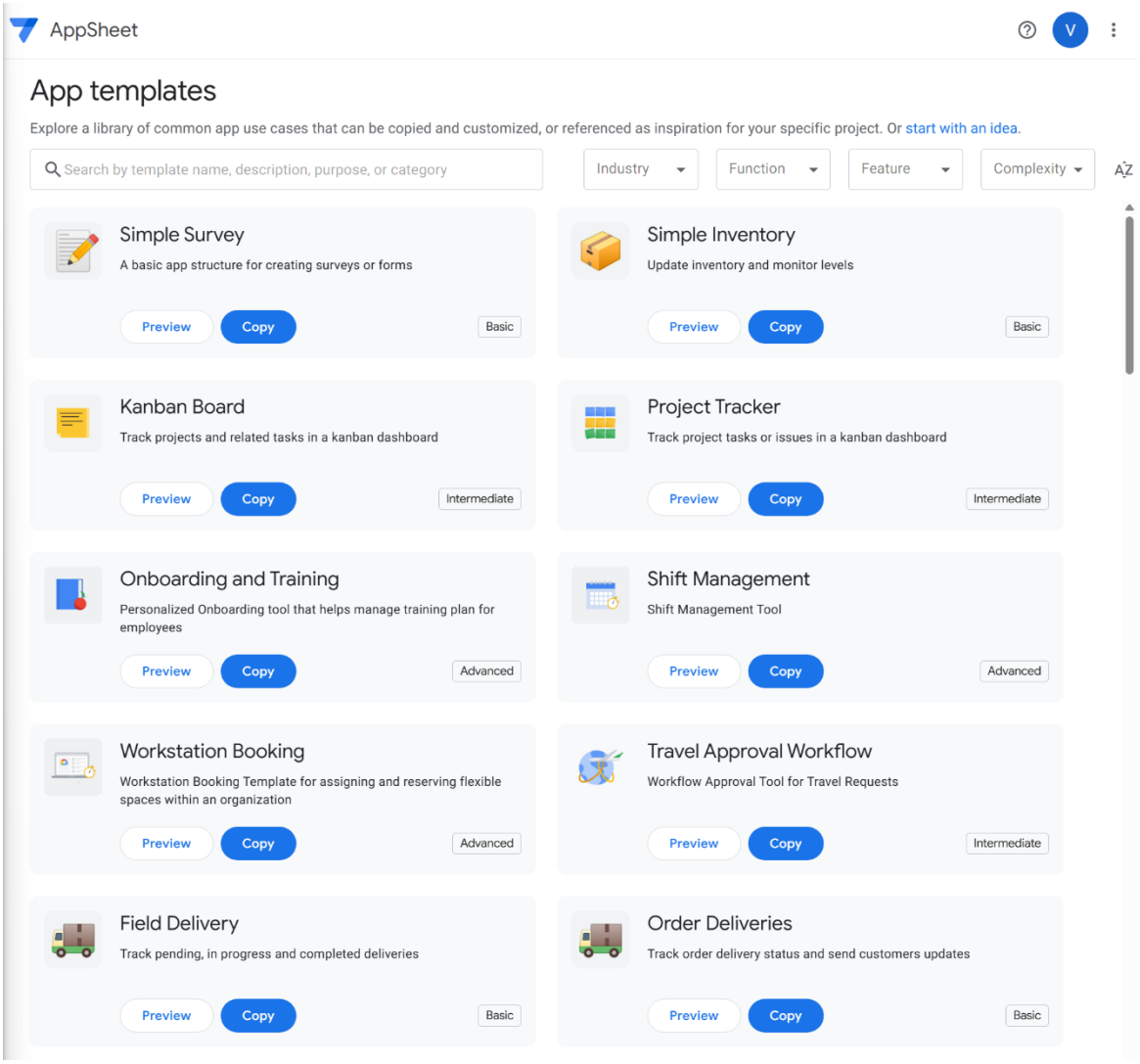
Taulukko 3. Toiminnalliset vaatimukset

Vaatimusmäärittelystä selviää sovelluksen ydintoiminnot sekä käyttäjän hallinnoimat ominaisuudet. Nämä varmistavat, että sovellus tarjoaa vaadittavat ominaisuudet toimiakseen käyttäjän tarpeiden mukaan. Lisäksi etenkin toiminnalliset vaatimukset luovat hyvän pohjan testauksen ja kehityksen seurannalle.

5.2 AppSheet-projektin aloitus

Ennen projektin aloittamista tutustuin AppSheet-työkaluun kokeilemalla valmiita mallipohjia, tutkimalla käyttöliittymää sekä katsomalla aiheeseen liittyviä opetusvideoita YouTubesta. Nämä toimet vastaavat hyvin pitkälti prosessia, jonka läpi kuka tahansa ensikertalainen AppSheetin käyttäjä käy. Vaikka koulutustaustani antaa minulle jonkin verran etulyöntiasemaa teknisten rakenteiden ymmärtämisessä, pyrin huomioimaan tämän arvioinnissa ja reflektoinnissa. Tavoitteena on ollut tarkastella AppSheetin käyttöönottoa ja sovelluskehityksen vaiheita erityisesti sellaisesta näkökulmasta, josta yksinyrittäjä voisi lähteä liikkeelle ilman syvempää ohjelmointiosaamista.

AppSheetin nopean sovelluskehityksen filosofian huomaa heti uuden projektin alkuvaiheilla. Uutta projektia luodessa käyttäjältä päättää, haluaako hän luoda uuden sovelluksen vai tietokannan. Tietokannan pystyy luomaan suoraan tyhjästä AppSheetsiin tai tuoda tietokannan Google Sheetsistä. Valitessa sovelluksen voi käyttäjä aloittaa tyhjästä tai käyttää valmiita pohjia (ks. Kuva 1), joita löytyy jo valmiiksi paljon esimerkiksi yksinkertaiseen kyselylomakkeeseen tai vuorojen hallintaan. Valmiit pohjat on vielä jaoteltu haastavuuden perusteella kolmeen ryhmään, jolloin uusi käyttäjä voi helposti tutustua erilaisiin toteutusvaihtoehtoihin ja hakea niistä inspiraatiota tai muokata omaan käyttöön. Lisäksi maksullisissa versioissa Gemini AI:n avulla pystyy luomaan nopeasti erilaisia pohjia, joita sitten käyttäjä voi itse muokata. Ilmasessa versiossa, jolla tätä teen, ei kyseistä vaihtoehtoa ollut.

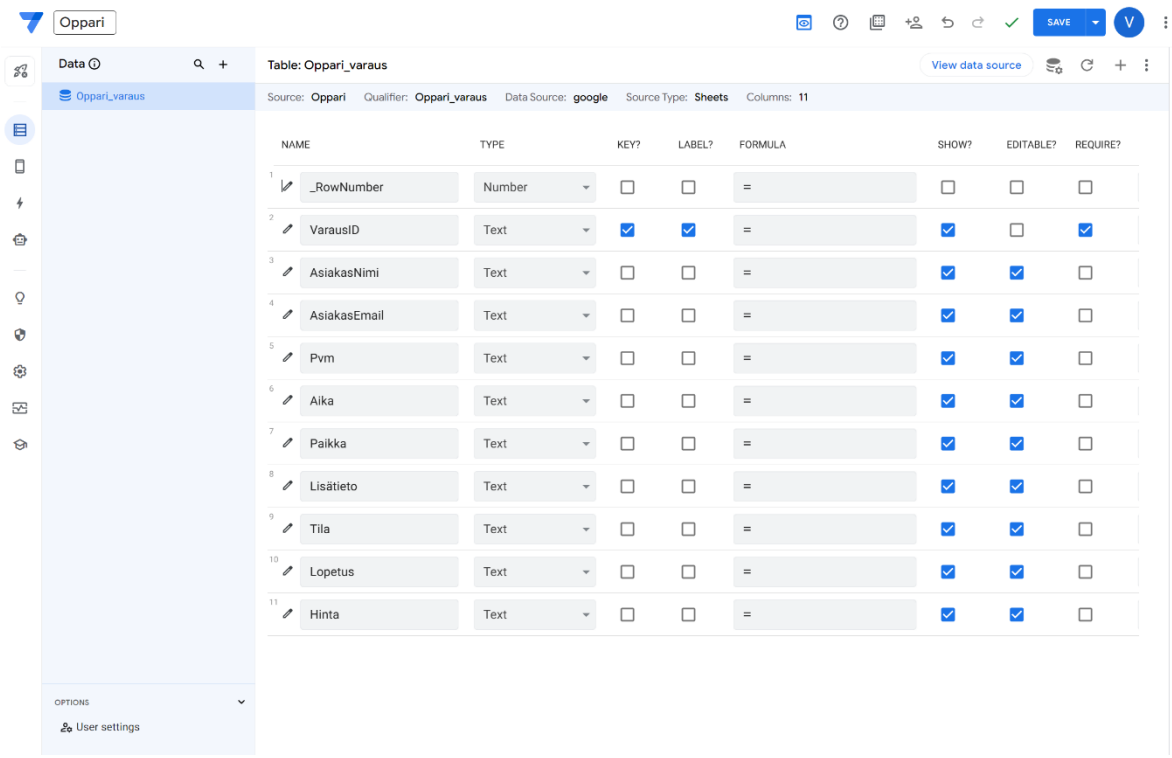


Kuva 1. AppSeet mallipohjia

Käyttäjä voi halutessaan tarkastella, kopioida tai muokata valmiita mallipohjia. Alkuun on hyvä tutustua malleihin, jotta työkalun mahdollisuudet tulisivat paremmin esille.

Varausjärjestelmän kehittäminen

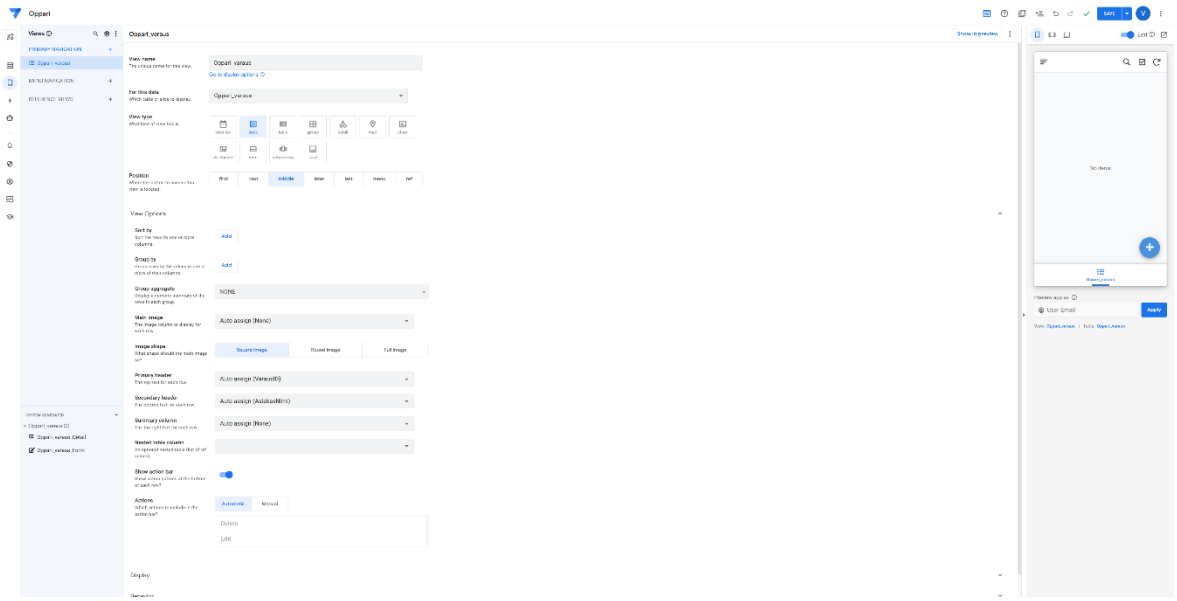
Lähdin itse aluksi liikkeelle tekemällä Google Sheetsiin taulukon, johon olin kirjoittanut sarakkeet valmiiksi varausta koskevia tietoja varten. Tämän jälkeen AppSheetin puolella alustin uuden sovelluksen valitsemalla ”aloita omalla datalla” -vaihtoehdolla, josta liitin juuri luomani taulukon AppSheetin tietokannan pohjaksi. AppSheet loi jakamani taulukon pohjalta automaattisesti seuraavanlaisen taulun (ks. Kuva 2).



Kuva 2. AppSheetin luoma taulu

Tästä taulusta yksittäisiä rivejä ja sarakkeita pystyy muokkaamaan käyttäjän tarpeiden mukaan esimerkiksi pakollisiksi kentiksi. Jokaista riviä pystyy myös määrittämään entistä tarkemmin kynäemojia painamalla.

Aloitusvaiheessa toinen tärkeä osa käyttöliittymää on näkymät. Näkymät ovat se osa työtä, joka näkyy käyttäjälle sovellusta käyttäessä ja toimii ikään kuin front-endinä järjestelmälle. Uusi näkymä tekee aina uuden ns. sivun, johon voidaan lisätä käyttäjän haluamaa dataa sekä muokata datan esittämistyyliä (ks. kuva 3). Etenkin tässä kohtaa huomaa no-code-alustojen rajoittuneisuuden muokkaamisen osalta sillä näkymiä pystyy muokkaamaan hyvin rajallisesti. Tämä voi myös helpottaa uutta käyttäjää sillä hänen ei silloin tarvitse huolehtia ulkoasun luomisesta tai muokkaamisesta. Responsiivisuus on toteutettu myös erinomaisesti sillä käyttäjän ei tarvitse huomioida tätä lainkaan. Sovellus toimii automaattisesti mobiilissa sekä työpöytäversiona automaattisesti. AppSheet sisältää valmiiksi sovelluksen esinäytön mobiilissa, tabletilla sekä tietokoneella. Uudelle kehittäjälle tämä on valtavan suuri etu sillä responsiivisuuden määrittäminen käsin voi olla todella aikaa vievä prosessi.



Kuva 3. AppSheetin näkymät

Projektin aloitus onnistuu uudellekin käyttäjälle hyvin helposti ja AppSheetin käyttöliittymä on todella intuitiivinen. Alussa uuden käyttöliittymän oppimiseen menee aina hetki mutta pienen opettelun jälkeen on se hyvin selkeä. Perus IT-taidot omaava henkilö varmasti pääsee omin avuin projektissa alkuun viimeistään apuvideoiden tai erilaisten tekoälyavustajien avulla.

5.3 Rakenteen kuvaus

AppSheetin sovellusten rakenne on hyvin suoraviivainen ja jakautuu neljään eri osioon: data, toiminnot, näkymät sekä automaatio. Nämä jaotukset voivat erota hieman käyttääkö editorin vanhaa vai uutta näkymää, mutta periaate on molemmissa sama. Perustoiminnallisuuden osalta tärkeimmät osiot ovat data sekä näkymät, joita luomalla ja muokkaamalla pystyy kehittämään yksinkertaisia ominaisuuksia sisältäviä sovelluksia. Järjestelmä on jaettu datan osalta kahteen taulukkoon, jotka ovat varaukset sekä laskut. Laskujen kerääminen omaan taulukkoon yksinkertaistaa ja helpottaa talousdatan keräämistä sekä ylläpitämistä.

Toiminnot ja automaatio taas mahdollistavat kehittyneempien sovellusten sekä toiminnallisuuksien luonnin. Erilaiset automaattiset vastausviestit sekä laskun luominen toimivat esimerkkinä, mitä toiminnallisuuksia näillä välilehdillä voi toteuttaa.

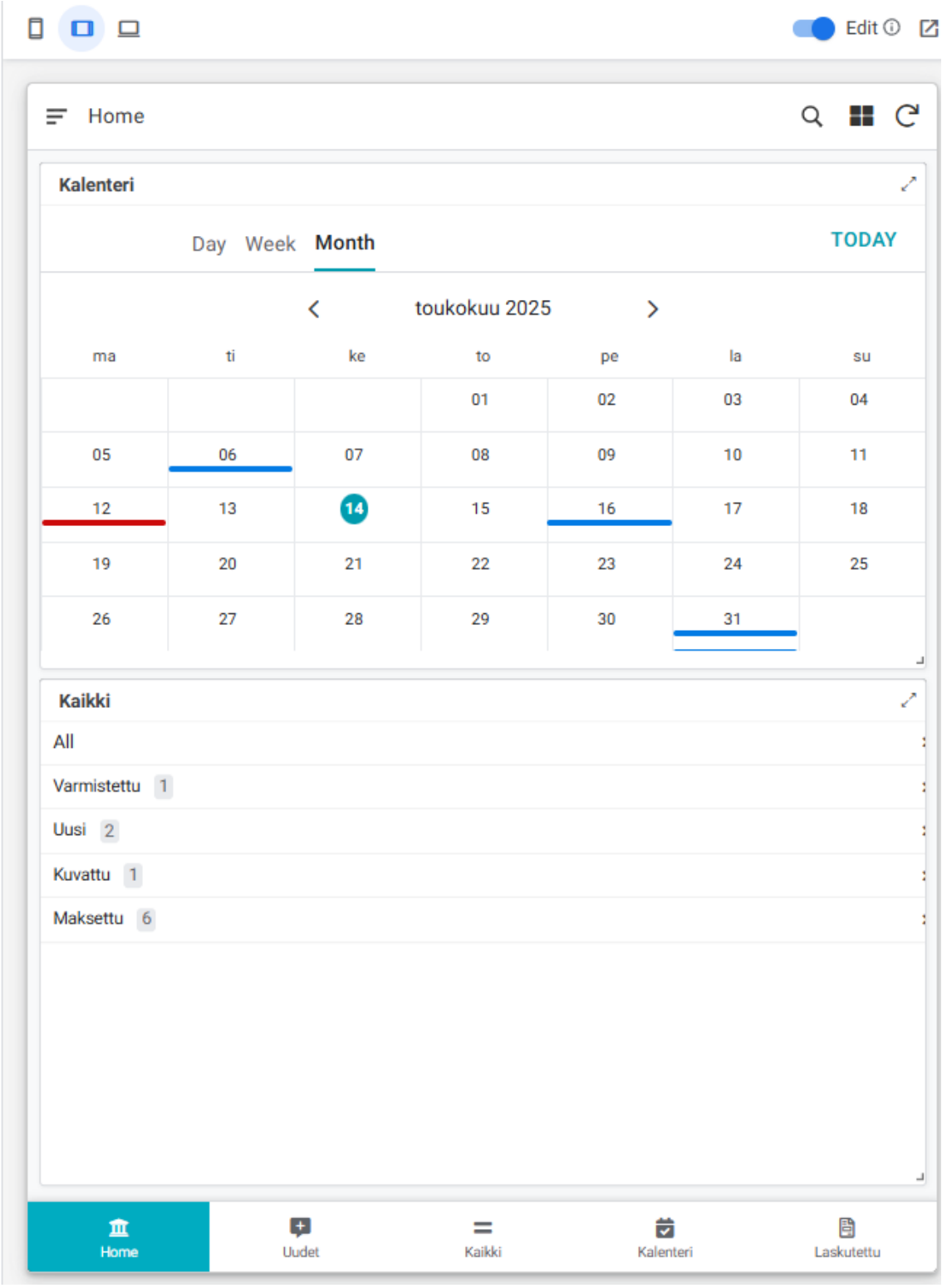
Olennaisena osana työssä on Google Apps Scriptiin luotu skripti (Liite 1.), joka määritettyjen parametrien perusteella hakee tietoa Gmailiin saapuneista sähköposteista ja luo niistä uuden rivin Sheetsiin josta AppSheet hakee dataa ja luo näin uuden varauksen tietokantaan. Skriptin käyttö ei ole ideaalia sillä tavoitteena oli välttää teknistä työtä tai koodaamista

sovellusta tehdessä. AppSheet ei tällä hetkellä kuitenkaan pysty hakemaan dataa suoraan Gmailista vaan siihen vaaditaan välikäsi. Tähän on kuitenkin tulossa muutos, joten tulevaisuudessa automaatiolla datan haku Gmailista onnistuu. Tässä tapauksessa Googlen tuotekatalogista löytyvä Apps Script valikoitui sillä sen saa helposti toimimaan yhteen Google Sheetsin kanssa eikä tarvita muita uusia sovelluksia tämän tekoon. Omasta opiskelutaustasta on hyötyä skriptin tekemisessä mutta internetin hakukoneita käyttämällä sekä esimerkiksi tekoälytyökalujen avulla skriptin tekoprosessi on todella helppo ja yksinkertainen. Skriptin käyttöä voidaan perustella datan keräämiseen suoraan Gmailista olevan hyvin olennainen osa työn rakennetta, mikä helpottaa jälkikäteen tehtyjä ominaisuuksia tärkeimmän datan automaattisen keräämisen takia.

Voidaan todeta, että sovelluksen rakenne on selkeästi jaoteltu, jossa näkymät, data, toiminnot ja automaatiot tukevat loogisesti toisiaan. Rakenne on pidetty yksinkertaisena ja sen kehittäminen onnistuu myös kokemattomalta käyttäjältä ovatkin hyvin saman tyylisiä ohjelmointikehityksen tapoja. Ratkaisu mahdollistaa helpompien, että vaativampien ominaisuuksien toteuttamista nopealla tahdilla.

5.3.1 Näkymät

Näkymät toimivat AppSheetissä ikään kuin sovelluksen frontendinä. Näkymistä siis muokataan sitä sovelluksen osaa, joka näkyy käyttäjälle ja jonka kanssa käyttäjä vuorovaikuttaa. Uusi näkymä luo aina uuden sivun, jossa dataa esitetään käyttäjän haluamalla tavalla esimerkiksi kalenterinäkymällä tai erilaisilla taulukoilla. Kotinäkymään voi toteutetun sovelluksen tavoin (ks. Kuva 4) upottaa esimerkiksi kalenterinäkymän sekä kaikki varaukset allekkain, josta käyttäjä nopeasti näkee kaikkien varausten tilan sekä kuluvan kuukauden varaukset kalenterinäkymästä. Muita näkymätyyppejä on useita kuten taulukko-, kortti- sekä luettelonäkymä, jotka kaikki esittävät dataa käyttäjälle hieman eri tavalla.



Kuva 4. kotinäkymä tabletilla

Sovellusta käytettäessä voi näkymään upotettuja eri elementtejä myös liikutella sekä vaihdella niiden paikkaa. Samalle näkymälle voi saada upotettua esimerkiksi neljä eri näkymää.

Huonona puolena AppSheetin näkymissä voidaan pitää niiden suhteellisen paljon rajoitettua visuaalista muokattavuutta. Samalla se kuitenkin korostaa olemassa olevien

muokkaamisvaihtoehtojen tärkeyttä ja pakottaa käyttäjän opettelemaan niiden käytön. AppSheetin helppo käyttöliittymä tekee näkymien muokkaamisesta kuitenkin vaivatonta, joten se ei sinänsä luo ongelmaa. AppSheetin slice-toiminnolla pystyy rajaamaan näkymässä näytettävää tietoa. Esimerkiksi sovelluksen ”NäytäUudet”-slice suodattaa näkymän näyttämään ainoastaan varaukset, joiden tila on uusi, kun taas ”NäytäVarmat” näyttää ainoastaan varmistetut kuvauspäivät. Suodattimien ansiosta käyttäjä voi kohdentaa helposti haluamaansa dataa näkyviin.

Näkymien kautta käyttäjälle esitetään oleellinen tieto selkeässä ja yksinkertaisessa muodossa. Näkymät mukailevat Googlen muiden tuotteiden minimalistista ulkoasua samalla pitäen sovelluksen käyttöliittymän intuitiivisena ja helposti käytettävältä. Myös erilaisten näkymien luominen on todella nopeaa ja helppoa, joka rohkaisee kokeilemaan erilaisia ratkaisuvaihtoehtoja. Tämä on erittäin hyvä konkreettinen sekä selkeä vahvuus no-code-sovelluskehityksen näkökulmasta.

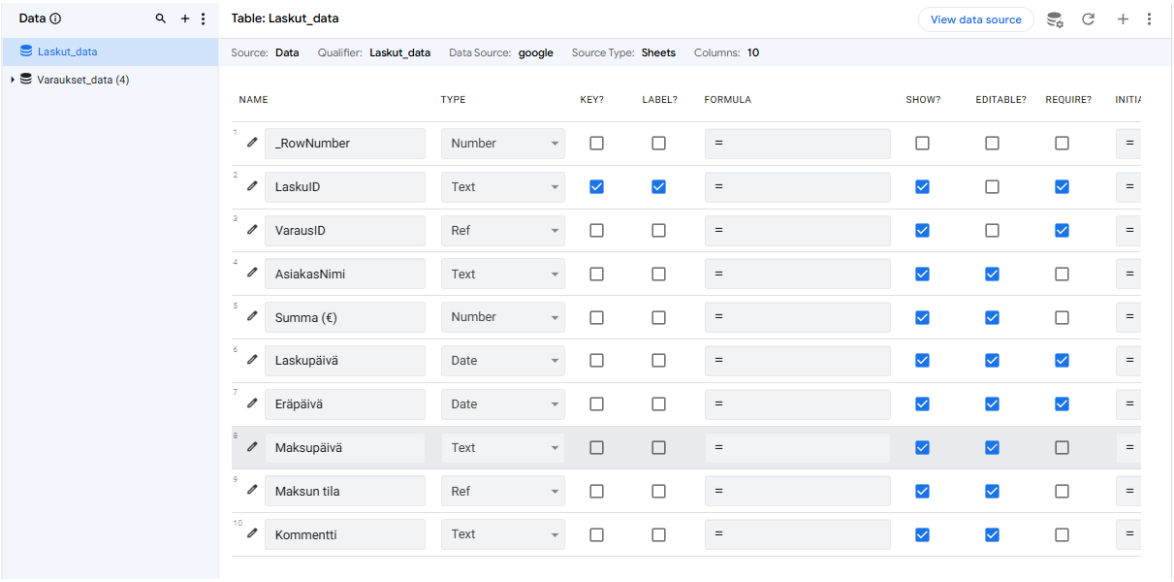
5.3.2 Taulut

Taulut toimivat AppSheet sovellusten keskeisenä rakennuspalana vastaten sovelluksen backendiä. Tässä työssä taulut saavat datansa ulkoisesta lähteestä, jona toimii tässä tapauksessa Google Sheets. Myös AppSheetin sisällä pystyy halutessaan rakentamaan suoraan oman tietokannan mutta datan automaattisen keräämisen takia toteutuksessa käytetty Google Sheetsiä. Opinnäytetyössä suoritetun esimerkkisovelluksen tietolähteinä toimivat kaksi Google Sheets-tiedostoa: Varaukset ja Laskut. Näihin tauluihin tallentuu kaikki sovelluksen käyttämä olennainen tieto varauksista sekä laskutuksesta. Datan jakaminen kahteen eri tauluun mahdollistaa myöhemmin toimintojen, erillisten näkymien ja automaatioiden rakentamisen ilman tietojen sekoittumista parantaen sovelluksen jatkokehitysmahdollisuuksia.

Tietorakenteet ovat yksinkertaisia ja helposti muokattavissa. Sovellus lukee ja pystyy myös päivittämään tauluja automaattisesti käyttäjän, toimintojen tai automaatioiden muutosten perusteella. Tämä mahdollistaa sovelluksen todella helpon ylläpidon myös käyttäjälle, jolla ei ole aikaisempaa osaamista tietokannoista.

Taulujen perustoiminnot ovat helppokäyttöisiä, mutta myös hieman edistyneempi logiikan kehitys on mahdollista. Esimerkiksi Laskut-taulussa oleva VarausID-kenttä viittaa Varaukset-tauluun, mikä mahdollistaa sen, että uutta laskua luodessa voidaan kohdistaa lasku oikeaan varaukseen. Yleisesti on myös suositeltavaa käyttää yksilöllisiä tunnisteita, kuten numerosarjoja datan yhdistämisessä eikä esimerkiksi asiakkaan nimeä tai henkilötietoja.

Taulukkonäkymän muokkausvaihtoehdot ovat todella käyttäjäystävälliset. Taulua on helppo ymmärtää ja sen muokkaaminen nopeaa. Käyttäjä voi esimerkiksi valita sarakkeiden datatyypin haluamakseen, kuten numeroksi, päivämääräksi tai tekstiksi. Sarakkeita voi myös asettaa pakollisiksi tai näkymättömäksi sekä lisätä oletusarvoja tai erilaisia tarkistuksia ilman ohjelmointia. (ks. Kuva 5).



Kuva 5. Laskut -taulu

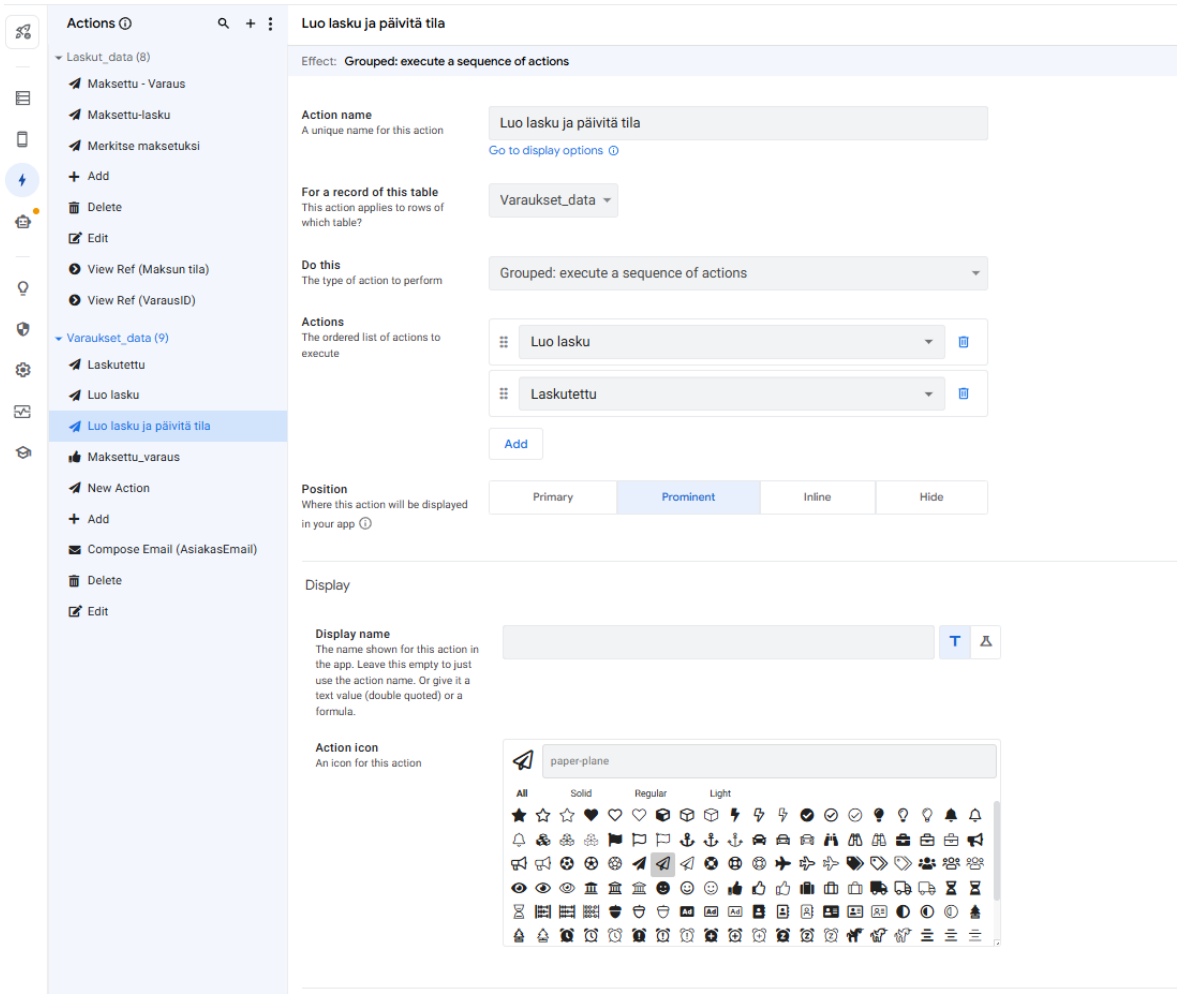
Taulukoiden käsittely AppSheetissä on käyttäjäystävällistä sillä graafinen editori piilottaa alleen monimutkaisemman teknisen osion. Käyttäjä pystyy helposti ilman koodaamista lisäämään, muokkaamaan ja poistamaan dataa. AppSheet ehdottaa myös automaattisesti sopivia tyyppejä sarakkeille datan perusteella. Muun muassa näiden ominaisuuksien ansiosta datan käsittely on mahdollisimman helppoa uudelle käyttäjälle.

5.3.3 Toiminnot

Toiminnot (Actions) ovat AppSheetin ominaisuus, jonka avulla käyttäjä voi luoda erityylisiä toiminnallisuuksia tai tehtäviä sovellukselle. Jokin käyttöliittymän elementti tai painike voi käynnistää toiminnon, joka esimerkiksi luo uuden laskun, lähettää automaattiviestin, muuttaa taulujen tietoja tai siirtyy uuteen näkymään.

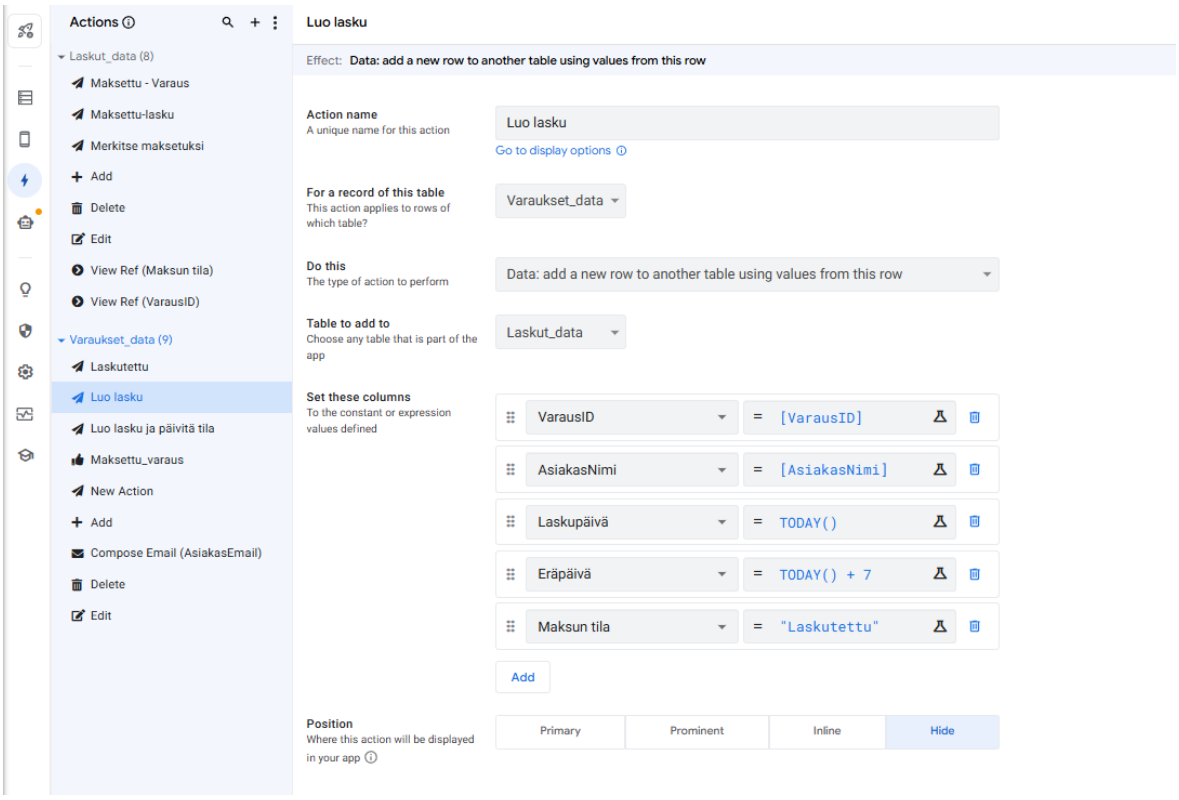
Tässä työssä yksi tärkeimmistä luoduista toiminnoista on ”Luo lasku ja päivitä tila” -komento (ks. Kuva 6). Toiminto on yhdistelmä kahdesta eri toiminnosta, jotka yhdistettynä luo varauksen tietojen pohjalta uuden rivin ”Laskut_data” -tauluun (ks. Kuva 7). Tämä on erittäin tärkeä ominaisuus sovelluksessa sillä se mahdollistaa laskun luomisen varauksen tietojen pohjalta yhtä nappia painamalla. Tämä varmistaa laskujen tulevan oikealle varaukselle asettaen VarausID:n, asiakkaan nimen sekä hinnan suoraan Varaukset_data -taulussa

olevan tiedon perusteella. ”Merkitse maksetuksi” niminen toiminto toteuttaa käytännössä saman työnkulun mutta se muuttaa vain varauksen sekä laskun tilan maksetuksi.



Kuva 6. Luo lasku ja päivitä tila -toiminto

Vasemmassa reunassa näkyy valmiiksi tehtyjä toimintoja. Toiminnot jakautuvat aina taulun perusteella. Toiminnoille voi myös valita haluamansa kuvakkeen.



Kuva 7. Luo lasku -toiminto

Toimintoja siis luodaan AppSheetin Action-editorissa valitsemalla tietotaulu, jonka jälkeen määritellään mitä tehdään ja mitä siitä seuraa. Automaatioiden avulla toiminnoista pystyy tekemään paljon monimutkaisempia ja tekevän kokonaisia työkulkuja napin painalluksella.

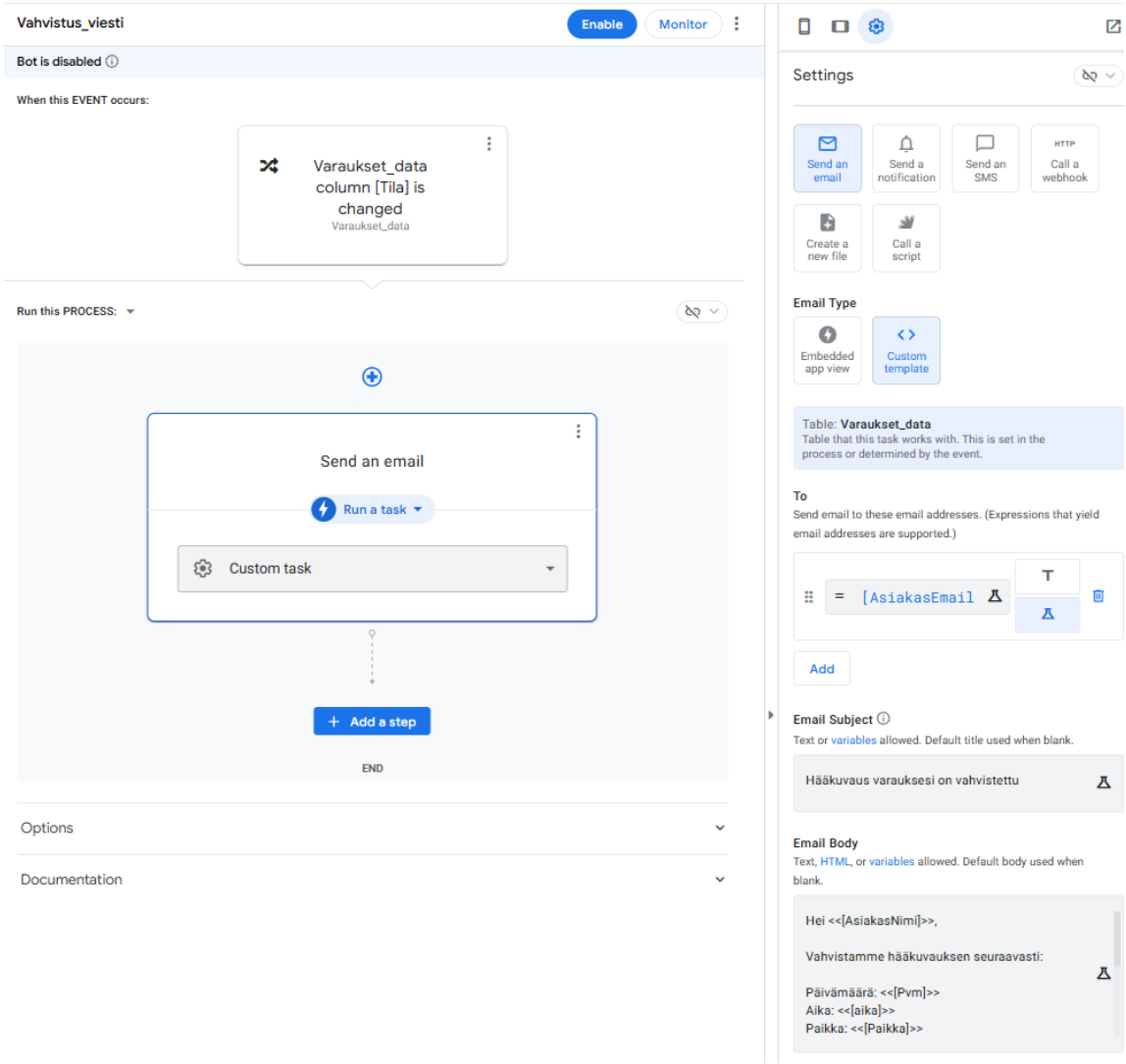
Toimintojen tärkein tehtävä on vähentää manuaalista työtä sekä varmistaa, että tiedon liikkuminen ja muuttuminen tauluissa sujuu mahdollisimman hyvin ja luotettavasti. Näin minimoidaan inhimillinen riski eli ihmisen tekemät virheet, kuten laskujen kohdistaminen väärälle varaukselle tai päällekkäisten laskujen luominen. Toimintojen yhdistely myös mahdollistaa entistä monimutkaisempien prosessien luonnin ja hallinnan ilman ohjelmointia.

5.3.4 Automaatio

Automaatioita käytetään AppSheetissä automatisoimaan toistuvia tehtäviä, jotka tapahtuvat aina samalla tavalla samoin säännöin. Hyviä käyttökohteita automaatiolle ovat esimerkiksi muistutusten lähettäminen, automaattisten sähköpostien lähettäminen ja tiedostojen luonti.

Automaatioita tehdessä ongelmaksi ilmeni automaatioiden vaativan testausta tai käyttöönottoa varten sovelluksen julkaisun, joka taas vaatii maksullisen tilauksen. Näitä ominaisuuksia en sen seurauksena täydessä potentiaalissa päässyt kokeilemaan mutta automaatioiden mahdollisuudet tuli kuitenkin hyvin selville. Yksi toimivaksi testattu automaatio oli vahvistusviestin lähettäminen asiakkaalle (ks. Kuva 8). Rajoituksena automaatiossa oli, että

viestin pystyi lähettämään ilmaisversiossa vain käyttäjän omaan sähköpostiin. Automaatio on testattu ja julkaistussa versiossa se toimisi odotetusti ja ennalta määritelty pohja varauksen tietojen perusteella lähetettäisiin asiakkaalle varauksen vahvistuksen jälkeen.



Kuva 8. Vahvistusviestin automaatio

Mielenkiintoisin automaatioon liittyvä asia työhön liittyen oli beetatestauksessa oleva Gmail-integraatio. Tämän jälkeen dataa voisi noutaa suoraan tietyn tunnisteen sähköposteista ja luoda uusia varauksia suoraan AppSheetin omaan tietokantaan. Seurauksena Google Sheetsin käyttö datan keruuta varten skriptillä ei välttämättä ole pakollista, joka entistä enemmän painottaisi no-code-näkökulmaa.

Automaatiot voivat olla hieman haasteellisempia toteuttaa uudelle käyttäjälle ja esimerkiksi erilaisia skriptejä pystyy käyttämään automaatioiden kautta hyvin laajasti. Automaatioiden rakentamisessa ohjelmointiosaamisesta voidaan siis sanoa olevan hyötyä. Vaikka perus-automaatioita pystyy tekemään parilla hiiren klikkauksella, vaatii ehtojen rakentaminen

pientä teknistä ymmärrystä. Ehtojen rakentelua voi kuitenkin verrata Excelin skriptien tekemiseen, joten syvällistä teknistä taitoa ei vaadita.

Automaatioiden toinen erinomainen käyttökohde on erilaisten tiedostojen automaattinen luonti esimerkiksi tila-arvon muututtua. Esimerkkinä toteutetussa työssä automaatio luo PDF-laskun automaattisesti varauksen tietojen pohjalta, kun laskun tila muutetaan laskutetuksi. Samalla automaatio pystyy myös lähettämään laskun asiakkaalle, jolloin yrittäjän ei tarvitse huolehtia erikseen laskujen luomisesta tai lähettämisestä.

Automaatiot ovat erityisen hyödyllisiä prosessien sekä prosessien kehittämisessä sillä ne vähentävät manuaalista työtä ja vähentävät inhimillisten virheiden riskiä. Etenkin pienyrittäjä hyötyy tästä suuresti ja esimerkiksi valokuvaaja pystyy keskittämään aikaansa luovaan työhön laskujen kirjoittamisen sijasta. Samalla myös dokumentaatio varauksista sekä laskuista kehittyy ikään kuin automaattisesti, jota voidaan pitää myös lisähyötynä pienyrittäjälle.

5.4 Haasteet ja opit

Varaussovelluksen toteuttaminen tarjosi monia oppimiskokemuksia sekä näkökulmia sovelluskehitykseen no-code-työkaluilla. Kehitysprosessin aikana nousi jatkuvasti esiin uusia hidasteita mutta kehitystyökalu osasi monissa paikoin itsessään ohjata oikeaan suuntaan, jonka ansiosta ne eivät tuntuneet niinkään haasteilta, vain uusilta asioilta. Työ siis osoitti, että no-code-työkalujen ketterä tuotantotyyli todella toimii.

Vaikka AppSheet mahdollistaa nopean kehitystyön, vaatii kehittäminen silti loogista ajattelua ja kykyä hahmottaa tietorakenteita. Yksinyrittäjälle voi olla alkuun työlästä uuden opettelua, mutta opetusvideot ja muut internetin lähteet helpottavat prosessia.

5.4.1 Teknisiä haasteita

Haastavimmat osiot esimerkkisovelluksen kehityksessä liittyivät datan automaattiseen keräämiseen sähköpostista sekä automaation rakentamiseen yleisesti. Tässä vaiheessa Gmail-integraatio on vielä beta-testauksessa, joten sen mahdollisuuksia ei vielä päästy kokeilemaan datan keräämisen suhteen. Tämän seurauksena automaation toteuttamiseksi jouduttiin käyttämään Apps Script-ratkaisua Google Sheetin kautta. Vaikka skriptin tekeminen vaatii jo enemmän teknistä osaamista, sen toteuttaminen internetin avustuksella onnistuu helposti ja mahdollista myös uudelle käyttäjälle. Tässä työn vaiheessa omasta koulutustausta kuitenkin oli ehdottomasti hyötyä, varsinkin ajallisesti sillä tiesin, miten ongelmaa

lähtisi ratkomaan ja aikaa ei mennyt ylimääräiseen tiedon keräämiseen. Uusi käyttäjä olisi voinut myös ratkaista ongelman toisella tavalla ja välttää näin skriptin tekemisen.

Toinen selkeä haaste nousi esiin automaatioita tehdessä. Monia automaation ominaisuuksia ei pystynyt käyttämään tai kokeilemaan ilman sovelluksen tuotantoa, joka olisi vaatinut maksullisen version. Toimintaperiaatteita pääsi kuitenkin kokeilemaan tarpeeksi hyvin, jotta niiden toimivuudesta sai selkeän käsityksen. Uusi käyttäjä voi kokea haastavammat automaatiot vaikeina mutta tehdyssä sovelluksessa automaatiot olivat yksinkertaisia eivätkä vaatineet pitkiä logiikkaketjuja.

5.4.2 Oppeja

AppSheetin käytössä mielenkiintoinen huomio oli se, että sen käyttäminen todella innostaa jatkokehitykseen ja uusien ominaisuuksien testaamiseen. Ominaisuuksia pystyy luomaan niin nopeasti, että yhden valmistuttua voi epäonnistua lukuisten ominaisuuksien teossa ja silti kokea saavansa paljonkin hyödyllistä aikaan.

Jatkuvaa kehitystä tukeva ajattelumalli myös iskostautui itseeni siinä mielessä, että uusia ominaisuuksia tai parempia tapoja tehdä jokin toiminto tulee jatkuvasti mieleen. Myös olemassa olevia toimintoja sekä ominaisuuksia pystyy jatkokehittämään todella helposti aina vähän ns. valmiimmaksi tai varmemmin toimivaksi.

Kokonaisuutena AppSheetin kanssa työskentely oli erittäin mielekästä ja voin kuvitella sen innostavan uusia käyttäjiä kokeilemaan sovellusten luontia. Aikaisempi kokemus ei ole välttämätöntä mutta kokemus ehdottomasti auttaa logiikan kanssa sekä tietokantojen ymmärtämisessä. Alustan peruskäytön kuitenkin oppii todella nopeasti ja AppSheet myös osaa ohjata uutta käyttäjää alustana hyvin.

Yksinyrittäjän näkökulmasta on esimerkkisovelluksen tyylinen ajanvarausjärjestelmä todellakin mahdollista tehdä mutta perehtyminen aiheeseen avittaa alun kompastuskivissä erittäin paljon. Huomionarvoista on myös se, että ominaisuuksien ollessa nopeita kehittää ja testata, on sovelluksen kehittäminen pienissä erissä myös mahdollista. Tämä tukee yksinyrittäjän kehitystyötä sillä projektin edistämiseen ei tarvitse varata useita tunteja aikaa ennakkoon.

5.5 Arvio työstä

Vaatimusmäärittelyssä määriteltiin pakollisia sekä mahdollisia ominaisuuksia valmiille työlle. Kaikki pakollisiksi määritetyt ominaisuudet tehtiin, jotka toimivat hyvin ja useita mahdollisiksi määritettyjä ominaisuuksia on toteutettu pakollisten lisäksi. Samalla myös

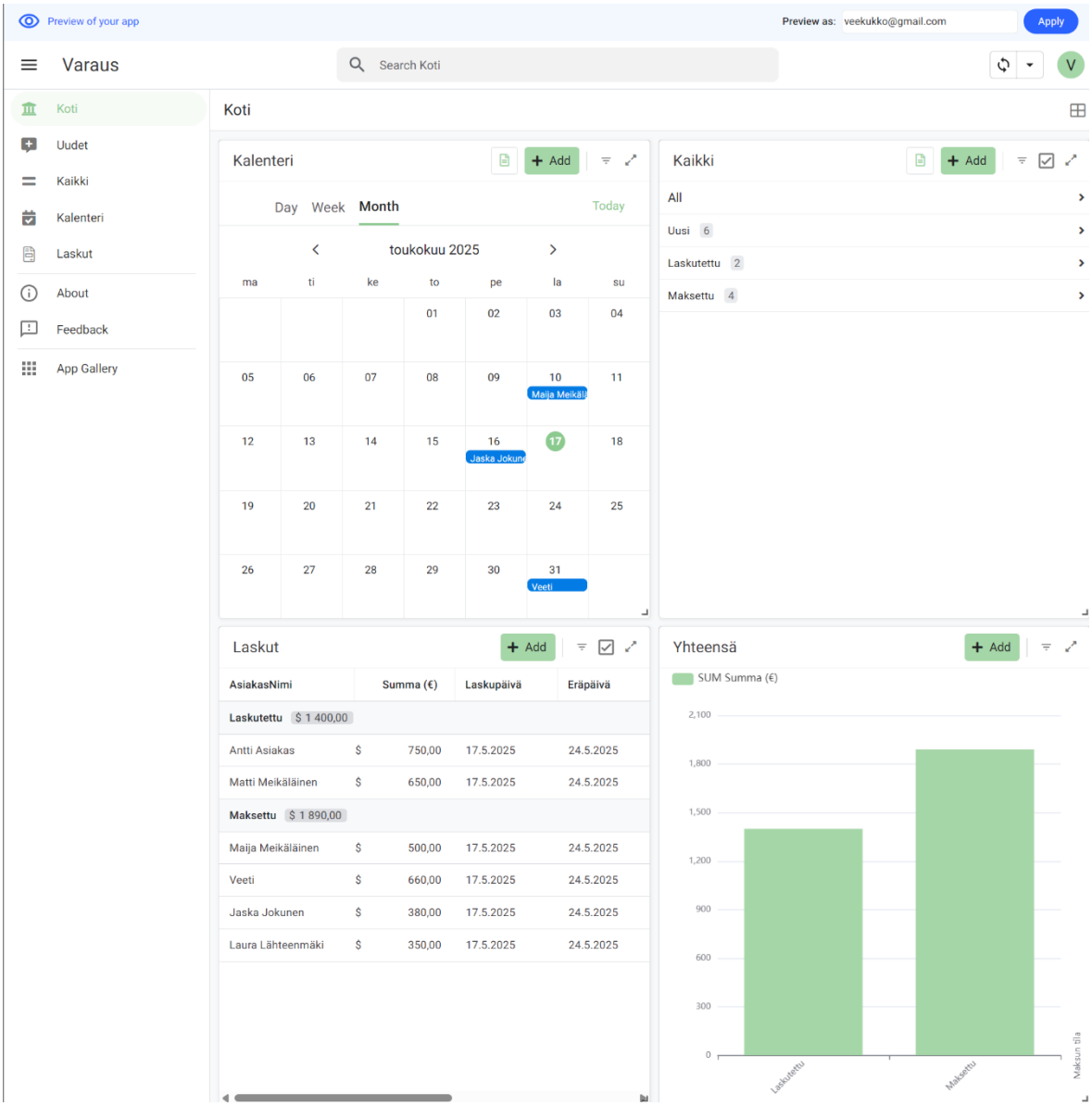
jatkokehitykseen jää joitain ominaisuuksia jo vaatimusmäärittelyn pohjalta. Ajanvarausjärjestelmänä työ toimii jo omassa käytössä, sillä tasolla, että hääkuvaaja pystyy hyödyntämään sitä liiketoiminnassaan.

Voidaan siis todeta sovelluksen toteuttavan tarkoituksensa ja onnistuneen sen kannalta hyvin. Kehityksen aikana ja jälkikäteen tehdyn tarkastelun pohjalta joitain ominaisuuksia olisi voinut toteuttaa eri tavalla, mutta sovelluksen elinkaaren tässä vaiheessa ne jätetään jatkokehitykseen.

5.5.1 Työn esittely

Rakenteen kuvauksessa on tuotu jo esitelty valmiin sovelluksen eri ominaisuuksia ja toimintoja, joten niitä ei erikseen käydä uudelleen läpi. Esitellään ennemmin tyypillistä käyttäjäkokemusta ajanvarausjärjestelmän käytöstä. Olen generoinut Google Sheetsiin esimerkkidataa, jota hyödynnetään esittelyssä. Henkilöiden tiedot eivät pohjaudu reaalimaailmaan.

Sovelluksen käynnistyessä kotinäkymä avautuu käyttäjälle. Siitä voi nähdä kokonaisvaltaisen tilanteen varauksista, niiden tiloista sekä laskutuksen tilasta. Mobiiliversiossa sekä tabletilla näytön alareunasta löytyy myös navigointipainikkeet eri näkymiin ja työpöytäversiossa navigointi on sijoitettu sivupalkin taakse. (ks. Kuva 9). Tabletilla ja työpöydällä sovellus pääsee vahvuuksiinsa näkymien kanssa mobiilinäkymän ollessa turhan ahdas niille.



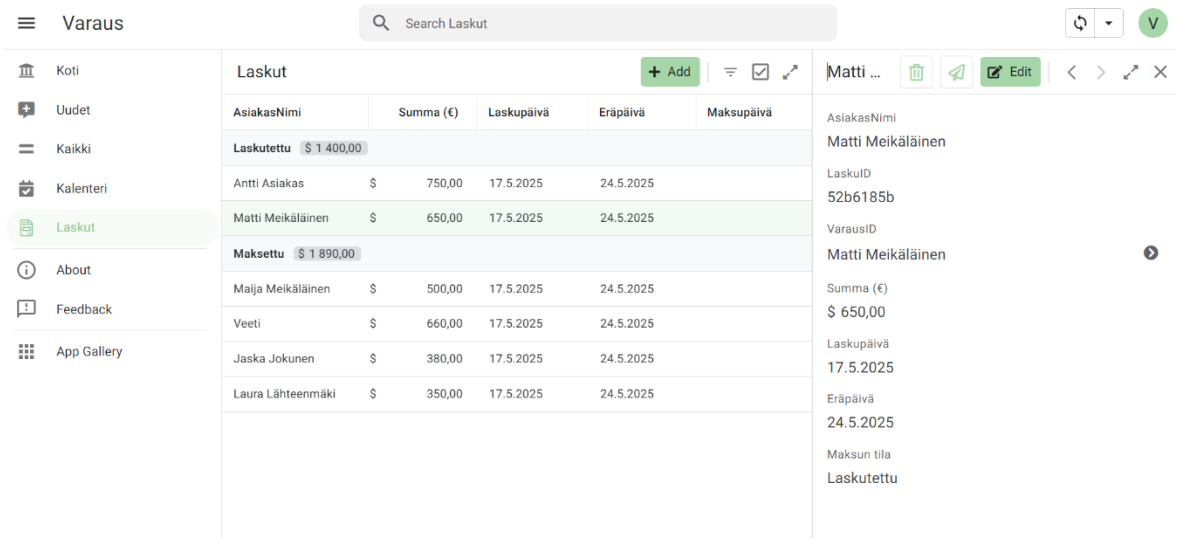
Kuva 9. Kotinäkymä työpöytäsovelluksessa

Työpöytäversiossa navigointi sijoittuu sivupalkkiin. Kuvassa myös esimerkki miltä neljän näkymän upottaminen yhteen näkymään voi näyttää. Dataa voi esittää myös useasta eri taulusta, jolloin yhdellä silmäyksellä voi saada kattavan kuvan kokonaisuudesta.

5.5.2 Varauksen elinkaari

Varauksen elinkaari tässä sovelluksessa kulkee kutakuinkin seuraavasti: Asiakas lähettää kyselyn kuvaajalle verkkosivujen ajanvarauslomakkeen kautta. Skriptin avulla automaatiota hyödyntäen varauksen tiedot kerätään Goosge Sheet-taulukkoon, josta AppSheet luo uuden varauksen sovellukseen. Tämän jälkeen käyttäjä näkee uudet varaukset ”Uudet”-näkymän takaa, josta käyttäjä voi muokata varauksen tilaa tilanteeseen sopivaksi. Kun keikka on tehty ja käyttäjä haluaa laskuttaa asiakasta, pystyy käyttäjä ”Luo lasku ja päivitä tila”-

painiketta painamalla luomaan asiakkaalle automaattisesti varauksen tietojen pohjalta laskun, lähettää sen ja päivittää tilaksi ”Laskutettu”. Samalla myös toiminnallisuus lisää ”Laskut_data” tauluun uuden rivin, joka pohjautuu varauksen tietoihin. Samaa varausta ei voi laskuttaa uutta kertaa vaan jokaisen VarausID:n pystyy laskuttamaan vain kerran. Tässä vaiheessa varauksen elinkaari siirtyy ”Varaukset”-taulusta ”Laskut”-tauluun (ks. Kuva 10).



Kuva 10. Laskut-näkymä

Laskut-näkymässä käyttäjä pystyy tarkastelemaan laskutettuja varauksia sekä maksettuja varauksia. Kun käyttäjä vastaanottaa asiakkaan maksusuorituksen, merkitään varaus maksetuksi painamalla paperilennokin kuvaketta. Myöskään samaa laskua ei pysty merkitsemään useaan kertaan maksetuksi. Ajanvarausjärjestelmän nykyisessä tilassa varauksen elinkaari päättyy tähän.

5.5.3 Jatkuva kehitys

Jatkuvan kehityksen kannalta AppSheet soveltuu erinomaisesti kehitetyn sovelluksen tyylisten projektien toteuttamiseen. Aikaisemmin työssä on jo tullut esille, miten innostava AppSheet on alustana ja tämä onkin avainasemassa myös jatkokehityksen kannalta etenkin yksinyrittäjän käytössä. Nopean kehityksen ansiosta uuden ominaisuuden luomista voi kokeilla ilman huonoja puolia tai mahdollisia takaiskuja.

Myös sovelluksen aktiivinen käyttö nostattaa etenkin tässä vaiheessa paljon ominaisuuksia ja ideoita uusille näkymille ja käyttötarkoituksille. Sovellusta pystyy esimerkiksi hyvin pienillä resursseilla jatkokehittämään yrityksen tarpeiden mukaan kohti kokonaisvaltaisempaa työkalua liiketoiminnan tukena. joka voisi esimerkiksi sisältää erilaisia valmiita sopimuspohjia, laskutaulukkoja tai inspiraatiogallerioita.

5.6 Tulosten yhteenveto

Yhteenvetona voidaan todeta, että AppSheet tarjoaa yksinyrittäjälle erittäin potentiaalisen ympäristön sovellusten jatkuvaan kehittämiseen, laajentamiseen ja muokkaamiseen ilman ohjelmointia tai suuria kustannuksia. Myös pienyrittäjien näkökannalta työkalut toimivat erinomaisesti liiketoiminnan tukena.

Toteutettu ajanvarausjärjestelmä vastasi hyvin sille asetettuja ennakkotavoitteita. Sen avulla asiakkaiden varausten sekä laskujen hallinta helpottuu huomattavasti. Sovellus kykenee hallitsemaan lukuisia varauksia sekä päivittämään varausten tietoa automaattisesti. Myös varausten automaattinen vienti sähköpostista sovellukseen toimi onnistuneesti.

Ennalta määritetyt ominaisuudet saatiin myös toteutettua onnistuneesti, vaikka jouduttiin käyttämään skriptiä sähköpostiautomaation aikaansaamiseksi. AppSheetin ilmaisversion rajoitukset rajoittivat hieman kehitysvaihetta, mutta kokonaisuuden kannalta tärkeimmät ominaisuudet saatiin testattua toimiviksi ilman päivittämistä maksulliseen versioon.

Sovelluksen jatkokehitys on myös todella yksinkertaista ja AppSheetin käyttö kannustaa siihen itsessään epäsuorasti toimimalla sulavasti ja antamalla valmiita pohjia uusille ominaisuuksille. Sovellus toimii käytännössä sellaisenaan kuitenkin tällä hetkellä, joten jatkokehitys ei ole pakollista. Hyviä kohteita jatkokehitykselle on kuitenkin erilaisten laskutuspohjien luonti ja järjestelmän laajentaminen esimerkiksi inspiraatiogallerioilla.

Työ osoitti käytännössä, että yksinyrittäjän ja pienyrittäjän on mahdollista hyödyntää no-code-työkaluja liiketoimintaa tukevan sovelluksen luomisessa ilman teknistä osaamista. AppSheetin aloittelijaystävällinen käyttöliittymä yhdistettynä mahdollisuuteen toteuttaa kehittyneitä rakenteita tekee siitä erinomaisen vaihtoehdon myös jatkokehitystä ajatellen. Työn tavoitteet toteutuivat suunnitellusti ja käytännön kehitystyö vahvisti, että AppSheet soveltuu ajanvarausjärjestelmän luomiseen yksinyrittäjän tai pienyrittäjän resurssit ja valmiudet huomioiden.

6 Yhteenveto ja pohdinta

Tämän opinnäytetyön tavoitteena oli selvittää, voidaanko low-code- tai no-code-työkaluja hyödyntää pienyrittäjän liiketoimintaa tukevien sovellusten kehittämisessä ilman ohjelmointiosaamista. Tavoitteen pääsyn tueksi luotiin hääkuvaajalle esimerkkisovelluksena ajanvarausjärjestelmä, joka automatisoi ajanvaraamisprosessia ja vähentää käsin tehtävää työtä samalla helpottaen yrittäjän arkea.

Työn aikana AppSheetillä kehitetty ratkaisu täytti sille ennalta määritellyt tavoitteet. Sovellus mahdollistaa varauksen automaattisen luonnin suoraan sähköpostiin saapuneiden kyselyiden perusteella, niiden hallinnan, laskutuksen luomisen sekä hallinnan ja varausten sekä laskujen tilan seuraamisen. Sovellus toimii niin mobiililla, tabletilla kuin työpöytälaitteilla ja sen rakenne mahdollistaa sovelluksen skaalautuvuuden ja kehittämisen yrittäjän tarpeiden mukaan nopeasti, helposti ja ilman ohjelmointia.

Kehitysprosessin aikana rakennettiin ajanvarausjärjestelmä erityisesti hääkuvauksia tekevän kuvaajan käyttötarkoituksia varten. Prosessi oli suoraviivainen, sillä vaatimusmäärittely oli selkä ja antoi tarpeeksi liikkumavaraa sovelluksen muovautumiseen kehityksen aikana. Tarkoituksena oli tarkastella sovelluskehitystä uuden, ohjelmointikokemusta omaamattoman käyttäjän näkökulmasta. Tämä onnistui sangen hyvin sillä alusta ei ollut ennestään laisinkaan tuttu. AppSheettiin tutustuminen alkoi uuden käyttäjän tavoin valmiiden pohjien tutkimisella sekä muutamalla YouTube videolla. Hyvin nopeasti kuitenkin siirryttiin suoraan kehittämään ja opettelemaan alustan käyttöä itse työkalusta käsin.

Pakollisten vaatimusten jälkeen toteutettiin joitain mahdollisia ominaisuuksia, kuten laskun lähettäminen mutta sovelluksen syvempi jatkokehitys jätettiin myöhemmin toteutettavaksi. Sovelluksen jatkokehitys ja ominaisuuksien suuri määrä pakollisen toiminnallisuuden lisäksi ei olisi myöskään tukenut opinnäytetyön tavoitteita, jotka olivat low-code ja no-code-työkalujen mahdollisuuksia ja käyttötarkoituksia yrittäjän omaan käyttöön.

Kehitysprosessi sujui itsessään todella nopeasti ja sovelluksen toimintaan saamisessa ei mennyt kauaa. Pohdintaa aiheutti se, milloin sovellus on niin sanotusti valmis. Tämän tyyliset sovellukset eivät koskaan ole täysin valmiita, vaan niitä jatkokehitetään elinkaarensa aikana. Kirjoitustyön viedessä paljon aikaa en lähtenyt ylikehittämään sovellusta vaan tarkoituksen mukaisesti pidin sen yksinkertaisena. Toisaalta se myös tuki tavoitteita sillä yksinyrittäjäkään ei lähde ensimmäisinä sovelluksinaan kehittämään mitään erityisen monimutkaista.

Haasteellista opinnäytetyössä oli määrittely pienyrittäjän ja yksinyrittäjän välillä. Hääkuvaaja on monesti yksinyrittäjä mutta saattaa työllistää esimerkiksi freelancereita.

Tietoperusta perustuu myös pienyrittäjiin mutta on kuitenkin sovellettavissa suoraan yksinyrittäjiin tässä tapauksessa. Tämän kokonaisuuden rakenteen haaste saattaa näkyä työn eri vaiheissa ja keskittymiskohta painottui loppua kohden enemmän yksinyrittäjään. Lopullisessa rakenteessa pienyrittäjän näkökulma nousee enemmän tietoperustassa ja yksinyrittäjä toiminnallisessa osuudessa. Jatkokehitysmahdollisuuksia ajanvarausjärjestelmälle on paljon mutta seuraavaksi voisi lähteä rakentamaan siitä valokuvaajalle kokonaisvaltaista hallintajärjestelmää yritykselleen.

Myös valmiin työn dokumentoinnin ja raportoinnin kanssa oli haasteita määrittää, kuinka tarkkaan ja miten paljon sovellusta näytetään. Sovellusta ei voi suoraan lisätä liitteeksi niin valmis sovellus tuodaan esille kuvina työn eri vaiheissa. Vaikka ajanvarausjärjestelmä esiteltiin, saattaa sen reflektointi yksinyrittäjän sekä pienyrittäjän suhteen olla puutteellista. Tuli myös selväksi, että vaikka työ osoittaa no-code-työkalujen toimivuuden, on haastavaa arvioida työtä toisen henkilön silmin. Vääjäämättä asioita, jotka ei ole uudelle käyttäjälle selviä jää huomioimatta. Tähän olisi voinut kehittää paremman tavan vertailla tuloksia, esimerkiksi pyytää palautetta ammattivalokuvaajilta.

Opinnäytetyö oli kokemuksena opettavainen ja innostava. AppSheet herätti oman mielenkiinnon mahdollisena keinona tuottaa sovelluksia ja varausjärjestelmän kehittäminen tulee jatkumaan edelleen. Haasteitakin oli mutta kokonaisuutena huomioiden kehitetyn ajanvarausjärjestelmän toimivuus ja aikataulu voidaan pitää projektia onnistuneena. Kehitetty ajanvarausjärjestelmä tulee myös pienten muutosten jälkeen käyttöön valokuvaajalle.

Lähteet

Alosevičius, D. 2024. Appsheet vs PowerApps: Which Is The Better App Maker? Viitattu 26.4.2025. Saatavissa https://codeornocode.com/comparisons/appsheet-vs-powerapps/

AppSheet. 2025. Google AppSheet | Build apps with no code. Viitattu 26.4.2025. Saatavissa https://about.appsheet.com/home/

Comidor 2024. Top 10 Business Challenges Low-Code Platforms Solve | Comidor. Viitattu 25.4.2025. Saatavissa https://www.comidor.com/blog/low-code/challenges-low-code-platforms-solve/

Decrue, L. 2025. The top 7 benefits of low-code development and how to find the right partners. Viitattu 30.3.2025. Saatavissa https://www.holycode.com/blog/the-top-7-benefits-of-low-code-development/

Jednaszewski, M. 2024. No-Code vs. Low-Code Development. Viitattu 27.4.2025. Saatavissa https://www.mendix.com/blog/understand-no-code-vs-low-code-development-tools/

Kissflow 2025a. The History of Low-Code and No-Code Development | The Next Step Forward. Viitattu 26.4.2025. Saatavissa https://kissflow.com/low-code/history-of-low-code-development-platforms/

Kissflow 2025b. Low-Code Vs No-Code | What's the Difference? [Ultimate Guide of 2025]. Viitattu 25.4.2025. Saatavissa https://kissflow.com/low-code/low-code-vs-no-code/

Microsoft. Low-code vs. no-code app development. Viitattu 25.4.2025. Saatavissa https://www.microsoft.com/en-us/power-platform/products/power-apps/topics/low-code-no-code/low-code-no-code-development-platforms

Microsoft 2024. What is Power Apps? - Power Apps. Viitattu 26.4.2025. Saatavissa https://learn.microsoft.com/en-us/power-apps/powerapps-overview

Ogilvy, L. 2023. The Origins and Impact of Low Code / No Code Platforms - Develocity. Viitattu 24.3.2025. Saatavissa https://develocity.io/the-origins-and-impact-of-low-code-no-code-platforms/

PCMAG. 2018. The Best Low-Code Development Platforms. Viitattu 31.3.2025. Saatavissa https://www.pcmag.com/picks/the-best-low-code-development-platforms

Roivainen, M. 2025. Low Code vs High Code: Pros and Cons Explained. Viitattu 31.3.2025. Saatavissa https://www.esystems.fi/en/blog/low-code-vs-high-code-pros-and-cons-explained

Snyder, J. 2023. Are Low-Code and No-Code Tools the Answer for Small Businesses? Viitattu 21.3.2025. Saatavissa https://biztechmagazine.com/article/2023/02/are-low-code-and-no-code-tools-answer-small-businesses

Tilastokeskus 2023. Tilastokeskus. Tilastokeskus. Viitattu 26.4.2025. Saatavissa https://stat.fi/tup/suoluk/suoluk_yritykset.html

Zapier 2022. Zapier data report: The rise of no-code. Viitattu 26.4.2025. Saatavissa https://zapier.com/blog/no-code-report/

Zoho. 2025. What is Zoho Creator? - A Low-Code App product. Viitattu 26.4.2025. Saatavissa https://www.zoho.com/creator/product-overview.html

Outsystems. 2025. Low-code vs. no-code: Why low-code is the future. Viitattu 28.3.2025. Saatavissa https://www.outsystems.com/low-code/vs-no-code/

Liite 1. Google Sheets-skripti sähköpostiautomaatioon

```
function tuoHaakyselytSheetiin() {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Taulukko1");
  if (!sheet) {
    Logger.log("Taulukkoa 'Taulukko1' ei löytynyt!");
    return;
  }

  var threads = GmailApp.search('label:hääkysely is:unread');
  for (var i = 0; i < threads.length; i++) {
    var messages = threads[i].getMessages();
    for (var j = 0; j < messages.length; j++) {
      var message = messages[j];
      var body = message.getPlainBody();

      // Alustetaan tiedot
      var nimi = "";
      var email = "";
      var pvm = "";
      var aika = "";
      var paikka = "";
      var lisatieto = "";

      // Pilkotaan rivi riviltä ja haetaan tutut kentät
      var lines = body.split("\n");
      for (var k = 0; k < lines.length; k++) {
        var line = lines[k].trim();
        if (line.toLowerCase().startsWith("nimi:")) {
          nimi = line.substring(5).trim();
        } else if (line.toLowerCase().startsWith("sähköposti:")) {
          email = line.substring(11).trim();
        } else if (line.toLowerCase().startsWith("päivämäärä:")) {
          pvm = line.substring(11).trim();
        } else if (line.toLowerCase().startsWith("aika:")) {
          aika = line.substring(5).trim();
        } else if (line.toLowerCase().startsWith("paikka:")) {
          paikka = line.substring(7).trim();
        } else {
          lisatieto += line + "\n";
        }
      }

      // Luodaan yksilöllinen varausID
      var varausID = Utilities.getUuid();

      // Lisätään tieto riviin
      sheet.appendRow([
        varausID,
        nimi,
        email,
        pvm,
        aika,
        paikka,
        lisatieto.trim(),
        "Uusi", // oletustila
        "" // Lopetus (tyhjä toistaiseksi)
      ]);

      // Merkitään viesti luetuksi
      message.markRead();
    }
  }
}
```