



Low-code-ohjelmistokehitys

Alustana Microsoft PowerApps

Tarmo Urrio

Opinnäytetyö, AMK

Heinäkuu 2022

Tekniikan ala

Tieto- ja viestintätekniikka

Urrio, Tarmo

**Low-code-ohjelmistokehitys:
Alustana Microsoft PowerApps**

Jyväskylä: Jyväskylän ammattikorkeakoulu. Heinäkuu 2022, 40 sivua.

Tekniikan ala. Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Työtapojen ja -ympäristöjen muuttuessa sekä etätyöskentelyn lisääntyessä, yritysten tarpeisiin tuotettujen ohjelmistojen tarve on jatkuvassa kasvussa. Tarve uusille sovelluksille on suuri, ja ohjelmistotalalla olevan osaajapulan hidastaessa yritysten tarvittavaa kasvua ja kehitystä, on tarve uusille osaajille myös lisääntyvä. Näihin haasteisiin vastaa osaltaan vuosi vuodelta suosiotaan kasvattava Low-code sovelluskehitys.

Opinnäytetyön tavoitteena oli tutkia Low-code sovelluskehitystä, syitä sen suosioon ja yleistymiseen sekä verrata oletuksia näillä alustoilla tapahtuvasta kehityksestä ja kehitystavan eduista verrattuna tavalliseen sovelluskehitykseen. Tarkemmin tutkittavana sovelluskehityksen alustana oli Microsoftin PowerApps.

Tutkimuksessa toteutettiin esimerkkisovellus, jota toteuttaessa seurattiin valitun alustan käyttöönottoa, ominaisuuksia sekä tällä tapahtuvan kehityksen eri osa-alueita. Vertailussa tarkasteltiin kehitykseen kuuluvia osa-alueita, ja sovelluksen tuottamiseen käytyä prosessia. Tutkimuksesta saatuja tietoja ja kokemuksia verrattiin kokemukseen tavallisesta sovelluskehityksestä ja kehityksen prosesseista.

Tutkimuksen pohjalta pystyttiin toteamaan Low-code alustoilla tapahtuvan kehityksen vastaavan pitkälti tutkimuksen alussa asetettuja olettamia, ja mahdollistavan toimivia sovellusratkaisuja erityisesti yritysten sisäisiin tarpeisiin hyvin kustannustehokkaasti. Ohjelmistokehityksestä taustaa omaavalle henkilölle siirtymä Low-code sovelluksiin ja kehityksen tavan omaksuminen tapahtuu hyvin vaivattomasti.

Low-code alustoilla tapahtuva sovelluskehitys tulee olemaan tulevaisuudessa merkittävässä roolissa osana uusia tuotettuja sovellusratkaisuja. Näillä ratkaisuilla voidaan erityisesti tehostaa yritysten sisäisiä prosesseja sekä tuottaa kustannussäästöjä että nopeuttaa monimutkaisia prosesseja, mahdollistamalla tietojen keräys, näyttö ja hallinta useammasta tietolähteestä, yhden sovelluksen avulla.

Avainsanat (asiasanat)

Low-code, Ohjelmistokehitys, Microsoft PowerApps

Muut tiedot (salassa pidettävät liitteet)

Urrio, Tarmo

**Low-code Software Development,
Microsoft PowerApps as a platform**

Jyväskylä: JAMK University of Applied Sciences, July 2022, 40 pages.

Information and Communications. Degree programme in Information and Communications technologies. Bachelor's thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

As working methods and environments change and working remotely increases, the need for software produced for business needs is growing. The need for new applications is significant, and as the shortage of experts in the software industry slows down the necessary growth and development of companies, the need for new experts is also growing. Low-code application development, which is increasing in popularity year by year, contributes to these challenges.

The aim of the thesis was to study Low-code application development, the reasons for its popularity and prevalence, and to compare assumptions about the development on these platforms and the advantages of the development method compared to ordinary application development. The application development platform that was studied in more detail was Microsoft's PowerApps.

An example application was implemented in the study, which was used to monitor the implementation and characteristics of the selected platform, as well as the various aspects of its development. The comparison focused on the development aspects and the development process. The data and experiences obtained from the study were compared with the experience of ordinary application development and its processes.

On the basis of the study, it was found that the development on Low-code platforms largely corresponds to the assumptions set at the beginning of the study, and enables efficient application solutions, especially for the internal needs of companies, in a very cost-effective manner. For a person with a background in software development, the methods and transition to Low-code applications are very effortless to learn.

Application development on low-code platforms will play a significant role in the future as part of the new application solutions produced. These solutions can be used to streamline companies' internal processes, both to generate cost savings and to speed up complex processes, by enabling data collection, display, and management from multiple data sources with a single application.

Keywords/tags (subjects)

Low-code, software development, Microsoft PowerApps

Miscellaneous (Confidential information)

Sisältö

1	Vähemmän koodia, enemmän arvoa.....	6
2	Tietoperusta	8
2.1	Tavallinen ohjelmistokehitys.....	8
2.2	Low-code-ohjelmistokehitys	9
2.3	No-code-ohjelmistokehitys	9
2.4	Historia	10
2.5	Ohjelmistokehityksen tulevaisuus	11
2.6	Markkinatilanne	12
2.7	Miten kehitys tapahtuu.....	13
2.8	Liitettävät järjestelmät ja tietolähteet.....	14
2.9	Varjo-IT - ongelma.....	14
2.10	Hyödyt	15
2.11	Haitat ja rajoitteet.....	17
3	Tutkimusasetelma	19
3.1	Tutkimuskohde.....	19
3.2	Tavoitekuvaus.....	20
3.3	Suunnitelma	21
3.4	Tutkimuksen rajaus	22
4	Tulokset.....	22
4.1	Tutkimuksen aloitus	22
4.2	Sovelluksen luonti	23
4.3	Tietojen yhdistäminen sovellukseen.....	25
4.4	Komponenttien lisäys.....	25
4.5	Näyttöjen lisäys	27
4.6	Näyttöjen ja komponenttien hallinta.....	28
4.7	Erilaiset komponentit.....	28
4.8	Virhetilanteet	30
4.9	Omien komponenttien määrittely	31
4.10	Sovelluksen kehittäminen	31
4.11	Tutkimuksen tulokset.....	32

5	Pohdinta.....	36
	Lähteet	41
	Liitteet	43
	Liite 1. Esimerkkiohjelman vaatimusmäärittely.....	43
	Liite 2. Valmiin ohjelman ulkoasu	45

Kuviot

Kuvio 1. Low-code kehitysalustojen ennustettu markkinatuotto (Vailshery, L. 2021)	12
Kuvio 2. PowerApps kehitysalustan aloitusikkuna.....	13
Kuvio 3. Low-code kehityksen nopeus verrattuna tavalliseen kehitykseen (Vailshery, L. 2022).....	16
Kuvio 4. Eri Low-code kehitysalustojen tarjonnan ja strategian tila (2021)	19
Kuvio 5. Uuden sovelluksen vaihtoehdot	23
Kuvio 6. Tyhjän sovelluksen luonnin aloitusvalinnat	24
Kuvio 7. Uuden sovelluksen aloitusnäkö.....	24
Kuvio 8. Tietolähteiden yhdistäminen	25
Kuvio 9. Uuden tekstikomponentin lisäys.....	26
Kuvio 10. Lisätty tekstikomponentti näytöllä	26
Kuvio 11. Komponentin tarkemmat asetukset	26
Kuvio 12. Käyttäjän tietojen näyttö usealla komponentilla.....	27
Kuvio 13. Uuden näytön lisäys	27
Kuvio 14. Tree view näkö.....	28
Kuvio 15. Tiedon visualisointi.....	29
Kuvio 16. Tietokortti - komponentti	29
Kuvio 17. Galleria – komponentti	29
Kuvio 18. Alustava etusivun näkö.....	30
Kuvio 19. Funktion virheilmoitus	30
Kuvio 20. Virheilmoitukset sovellusnäkö.....	31
Kuvio 21. Omien komponenttien luonti	31
Kuvio 22. Useamman funktion yhdistelmä	33

1 Vähemmän koodia, enemmän arvoa

Tavallinen ohjelmistokehityksen prosessi on usein pitkä, sitoo ison ryhmän kehittäjiä yhden projektin pariin ja voi kasvaa asiakkaalle hintavaksi. Tuotetun ohjelman mahdolliset ongelmat tai korjaustarpeet ovat usein välttämättömiä. Tyypilliseen ohjelmistokehityksen malliin voi kuulua jopa 16 eri vaihetta (The Low-Code Development Guide N.d.) ja sisältää huomattavan paljon mahdollisia muuttujia, joita ei voi ennalta arvioida. Näiden vaikutuksesta ohjelmiston tuotannon kulut voivat nousta alkuperäistä suunnitelmaa huomattavasti korkeammiksi.

Tästä hyvänä esimerkkinä on neljälle suomalaiselle sote-alueelle suunniteltu Aster-potilastietojärjestelmä, jonka toteutus kaatui lopulta osittain myös kohonneiden kehityskustannusten vuoksi. Alkuperäisestä sopimussummasta kolminkertaiseksi kohonnut loppulasku oli lopulta tilaajille liikaa ja näin iso syy projektin päättämiseen ennen tuotteen valmistumista. (Lindroos, K & Björklund, S 2021.)

Tavallisessa ohjelmistokehityksessä yhteistyö asiakkaan kanssa voi olla haasteellista, sillä projektiin kuuluvia sidosryhmiä voi olla useita, ja projektin senhetkinen tuotannon tila tulee tarkasti selväksi asiakkaalle vasta, kun tuote on testaus- tai esittelyvalmiudessa. Tätä ennen asiakkaan käsitys ohjelman kehityksen tilasta voi perustua esimerkiksi vain projektin asiakasvastaavien tilannekatsauksiin tai pieniin esittelyihin ohjelman tilasta, ja on riippuvainen pitkälti tuottajan kertomasta. Kehitys sitoo yrityksen resursseja projektiin, ja kehitysyhteistyö asiakkaan kanssa on usein rajallista, sillä usein asiakkaalla ei ole teknistä tuntemusta tarvittavista taustajärjestelmistä tai kehityksen eri prosesseista.

Entä jos ohjelmistojen kehitys olisikin helpompaa, nopeampaa ja kasvavat kulut johtuisivat vain uusien ominaisuuksien lisäämisestä valmiiseen tuotteeseen? Mitä jos asiakas voisi nähdä tuotettavan ohjelman senhetkisen tilan minä hetkenä hyvänsä ja parhaimmillaan osallistua jopa itse toteutukseen muutenkin kuin osallistumalla ohjelman vaatimusten määrittelyyn? Näihin ongelmiin vastaa low-code-ohjelmistokehitys, joka kasvattaa vuosi vuodelta yhä enemmän suosiotaan ja osuuttaan osana nykyaikaisia ohjelmistoratkaisuja.

Kasvaneeseen suosioon vaikuttavat useat eri tekijät, joista jopa tärkeimmässä roolissa on itse ohjelmistokehitykseen käytettävän työn määrän vähäisyys. Low-code-ohjelmistokehitys mahdollistaa nopeat toteutukset, tehokkaat integraatiot muihin järjestelmiin ja tietolähteisiin sekä matalamman kynnyksen tavan luoda ohjelmistoja eri käyttötarkoituksiin. Low-code alustojen käyttö on trendien perusteella iso osa tulevaisuutta ohjelmistokehityksessä (Vailshery, L. 2021). Nämä alustat antavat mahdollisuuden ohjelmointiin, ohjelmistojen suunnitteluun ja toteutukseen myös henkilöille, joilla ei ole välttämättä mitään aiempaa taustaa ohjelmistojen kehittämisestä.

Low-code alustat mahdollistavat ohjelman suunnittelun ja toteutuksen yksinkertaisen käyttöliittymän kautta niin, ettei kehittäjän tarvitse luoda ohjelmaa varten alusta alkaen kaikkea taustalla tarvittavaa ohjelmakoodia. Low-code alustojen hyödyntäminen vähentää ohjelman tuotantoon tarvittavaa kehitystyötä, kustannuksia sekä mahdollistaa suoran yhteistyön sovelluksen tilaajan kanssa, tuotteen kehitystason ollessa jatkuvasti nähtävillä. Tällä tavalla tuotetuilla ratkaisuilla madalletaan asiakkaan kynnystä valita tarvittavalle ohjelmistoratkaisulle juuri yrityksen tarpeisiin kohdennettu tuote. Tästä taas hyötyvät sekä ohjelman tilaaja että sen toteuttaja kumpikin omalta osaltaan eri tavoin. Nämä ratkaisut ovat myös helposti liitettävissä useisiin eri tietolähteisiin, joten valmiina olevaa tietoa pystytään hyödyntämään sovelluksen käytössä erittäin monipuolisesti ja tavalliseen ohjelmistokehitykseen verrattuna paljon helpommin, minimoimalla eri rajapintojen yhdistämisen tuomat haasteet.

Tämän työn tavoitteena oli tutkia low-code-ohjelmistokehitystä, sen hyötyjä tai haittoja sekä alati kasvavaa suosiota yleisellä tasolla. Työ sivuaa low-code kehitysratkaisuihin vaikuttavia tekijöitä, alustojen yleisiä ominaisuuksia, näillä toteutettavaa kehitystä sekä tarkastelee esimerkin kautta sovelluskehittämistä Microsoftin PowerApps-alustalla. Työ vertaa myös tämänhetkistä yleistä mielialidettä aihealueesta, ja vertaa sitä esimerkkisovelluksen tuottamisen kautta koettuun kokemukseen. Tutkimuksesta rajattiin pois eri kehitysalustojen vertailu ja näiden taustalla toimivien järjestelmien toiminta.

Työssä etsittiin vastauksia empiirisen tutkimuksen kautta low-coden suosioon, siihen kuinka nopeaa on sisäistää kehityksen tapa ja käytänteet sekä kuinka haastavaa se mahdollisesti on. Tutki-

mukksessa tarkasteltiin PowerApps – kehitysalustalla toteutetun esimerkkisovelluksen toteuttamista sekä tarkkaillaan tähän käytyä prosessia, vaikuttavia tekijöitä sekä saavutettuja tuloksia, verrattuna ennalta asetettuihin yleisiin tutkimuksen tietoperustasta johdettuihin mielipiteisiin.

2 Tietoperusta

2.1 Tavallinen ohjelmistokehitys

Tavallinen ohjelmistokehitys pohjautuu useaan eri vaiheeseen, jossa ohjelmistoa rakennetaan alkuideasta suunnittelun, toteutuksen ja testauksen kautta lopulliseksi tuotteeksi. Ohjelmiston toteuttamiseen voidaan käyttää yhtä tai useampaa ohjelmointikieltä, ja kehitys tapahtuu usein useammasta kehittäjästä koostuvan tiimin toimesta. Kaikki mitä ohjelmassa tapahtuu näkyvillä tai taustalla, on kirjoitettu ohjelman muotoon joko käsin tai lisätty ohjelman hyödynnettäväksi esimerkiksi valmiiksi toteutetusta avoimen lähdekoodin ratkaisusta. Myös jotkin osiot ohjelmasta voivat olla esimerkiksi ulkopuoliselta palveluntarjoajalta, tai hyödynnetty julkisesti saatavilla olevista ohjelmakirjastoista.

Ohjelman toteutukseen vaadittavien resurssien määrä on toteutettavan ohjelmiston laajuudesta tai kompleksisuudesta riippuvainen, mutta samalla mahdollistuvat myös optiot lukemattomille eri ratkaisuille, ohjelman hienosäädöille sekä täsmällisesti asiakastoiveiden mukaisesti räätälöidyille toteutuksille ohjelmaa luodessa. Mahdollisuudet eri toteutustavoille ja vapaus toteuttaa ohjelman koodi täsmällisesti kehittäjien mieltymysten mukaisesti tuovat omana osanaan myös edut ja mahdolliset haitat itse lopulliseen tuotteeseen.

Ohjelmiston laadun varmistamiseksi, tuotettavalle ohjelmalle toteutetaan paljon testausta, rasi-tusta sekä laaduntarkkailua, joka kuluttaa myös osaltaan resursseja ja tuottaa lopulta lisää kustannuksia asiakkaalle. Mahdolliset viat yritetään seuloa ohjelmasta jo alkuvaiheessa ja asiakkaalle lopulta toimitettava tuote on virallisesti valmis viimeisten hyväksyntäkierrosten sekä asiakkaalle toimituksen jälkeen. Tämän jälkeen aloitetaan ylläpitovaihe, jonka aikana ohjelmistotuotteen kuntoa ylläpidetään ja päivitetään, mahdollisiin vikatilanteisiin reagoidaan ja tuotetta kehitetään asiakkaan tarpeiden mukaisesti, kunnes tuote saavuttaa elinikänsä päädyn.

2.2 Low-code-ohjelmistokehitys

Low-code on nimensä mukaisesti ohjelmistokehitystä, jossa koodia tarvitaan joko mahdollisimman vähän tai ei ollenkaan. Se perustuu PaaS-palvelumetodilla tarjottavaan, usein pilvipohjaiseen sovelluskehityksen alustaan, jolla ohjelman kehittäjälle tarjotaan tämäntyyppisten ohjelmistojen kehitykseen tarvittava ympäristö, yleisimmin säännöllisesti laskutettavan lisenssimaksun hinnalla. Low-code kehitykseen tarkoitettuja alustoja on paljon, ja näiden palveluntarjoajat ja tuotteet vaihtelevat tunnetuista suurista yhtiöistä, pienempiin avoimeen lähdekoodiin perustuviin ratkaisuihin.

Kehitys tapahtuu käyttöliittymäpohjaisesti, joko internet-selaimen tai tietokoneeseen asennettavan ohjelmiston kautta. Kehittäjällä on käytössään kehitysympäristö, missä graafisessa käyttöliittymässä tarjolla ovat kehitysalustan tarjoamat vaihtoehdot, kuten eri komponenttien tai funktioiden asetukset sekä muut tarvittavat työkalut, joilla hallinnoidaan loppukäyttäjälle näkyville tulevia näkymien asetteluja ja näihin tuotavia tietoja. Kehitysalustat pohjaavat toimintansa erilaisiin liittämiin, joilla tuotettava ohjelma yhdistetään hakemaan tietoa muista käytössä olevista järjestelmistä, kuten esimerkiksi Excel-taulukosta tai erillisestä tietokannasta. Näistä haettavaa tietoa käsitellään ja näytetään vapaasti valittavien komponenttien kautta erilaisissa luoduissa näkymissä, joista haettu tieto on helpommin saatavilla tai tarpeen mukaisesti käsiteltynä tai esitettynä.

Low-code sovelluksia toteutetaan usein yleisimmin yritysten erilaisiin sisäisiin tarpeisiin, kuten korvaamaan useampaa eri hallintatyökalua tai esimerkiksi keräämään tietoja useasta eri tietolähteestä yhteen näyttöön, jolloin tarvittavan tiedon hallinta helpottuu huomattavasti. Low-code mahdollistaa jatkuvan testauksen jo sovellusta tehdessä, jonka ansiosta mahdolliset virhekonfiguraatiot voidaan huomata ja ratkaista nopeammin jo ennen kuin ne päätyvät ohjelman testauksen vaiheeseen. Myös itse ohjelman toimintaan tarvittavan ohjelmakoodin vähäisellä määrällä, mahdollistetaan tapa luoda ohjelmia myös henkilöille, keillä ei ole aiempaa kokemusta ohjelmoinnista, yksinkertaistamalla uuden ohjelman luonnin prosessia sekä tarvittavaa osaamista ja työkaluja verrattuna tavalliseen ohjelmistokehitykseen.

2.3 No-code-ohjelmistokehitys

Low-codesta seuraava askel alaspäin tarvittavan koodin määrässä, on niin kutsuttu no-code. Näillä alustoilla mahdollisuus ohjelman toimintoja ohjaavan taustakoodin muokkaukseen on poistettu, ja

nämä kehitysalustat ovatkin huomattavasti rajatumpia verrattuna low-code kehitysalustoihin. Taustakoodin muokkauksen puuttuessa, käyttäjällä ei ole mahdollisuutta aiheuttaa tilannetta, jossa ohjelma ei toimisi enää käyttäjän tekemän taustakoodin muutoksen myötä. No-code alustoilla kehityksen rajoitukseksi nouseekin koodin lisäämisen puuttuessa, näiden alustojen joustavuus toteuttaa täsmällisemmän tarpeen ratkaisuja.

Kehitys näillä alustoilla on täysin alustan tarjoamien vaihtoehtojen tarjoamien rajojen varassa. Rajoitteena toimivatkin lopulta kyseisen alustan tuottajien toteuttamat kehitysalustan mahdollisuudet. Mikäli jotain ominaisuutta ei ole suunniteltu alustalle, on hyvin todennäköistä, että sitä ei pysty toteuttamaan edes tarpeeksi montaa erilaista alustan ominaisuutta yhdistämällä. No-code alustat ovatkin kohdennettu enemmän käyttäjille, keillä ei ole yhtään aiempaa ohjelmistokehityksen taustaa. Nämä ratkaisut sopivatkin siksi esimerkiksi yrityskäyttäjille, jotka tahtovat luoda nopeasti yksinkertaisen sovelluksen esimerkiksi nopeuttamaan yrityksen sisäisiä toimia (Forsyth 2021).

2.4 Historia

Ensimmäisenä low-coden kaltaisena alustana voidaan pitää Applen kehittämää Hypercard-järjestelmää vuodelta 1987. Tämä mahdollisti nykyisten kehitysalustojen kaltaisen ohjelmien luomisen ilman konkreettista ohjelmakoodin kirjoitusta, kuitenkin myös mahdollistaen tarvittaessa ohjelman taustakoodin muokkauksen. (Lasar 2019.) Jo tässä vaiheessa kohderyhmänä olivat uudet kehittäjät, keillä ei ollut tarpeen omata aiempaa kokemusta ohjelmoinnista. Tämän alustan kehitys lakkautettiin hieman yli kymmenen vuoden jälkeen, jonka jälkeen kehitystavan nykyisen suosion alkuna voidaan pitää low-code käsitteen nousemista ilmoille vuonna 2014. Tällöin tunnettu amerikkalainen Forrester-tutkimusyhtiö julkaisi artikkelin, joka käsitteli uuden suosiotaan kasvattavan kehitystavan yleistymistä ja käytti low-code termiä kyseisten kehityksen tavan ja alustojen nimeämiseen. (Tozzi 2021.)

Termin käyttö yleistyi alustojen jatkuvan suosion kasvaessa ja kehittyessä vuosien aikana ja on sitä kautta noussut käsitteenä nykyiselle tunnettavuuden tasolle. Kehitystavan suosion kasvaessa, eri toimijat suurimmista alkaen ovat alkaneet tuottaa omia kehitysalustoja, joilla vastata tähän kasvavaan kysyntään.

2.5 Ohjelmistokehityksen tulevaisuus

Nopean ohjelmistokehityksen ratkaisut ovat kohottaneet vuosi vuodelta suosiotaan, ja käyneet yhä tunnetummiksi myös ohjelmistoja tilaavien asiakkaiden keskuudessa. Ohjelmistoala on suunnattomasti kasvava ala, ja samalla tarve näiden ratkaisuiden toteuttajille on noususuunnassa. Schoettlen (2021) mukaan low-code kehitystä ei tule kuitenkaan väärinymmärtää tavallisen ohjelmistokehityksen syrjäyttäjäksi, vaan pikemminkin rinnalla toimivaksi vaihtoehdoksi. Molemmilla kehitystavoilla on omat puolensa, ja on itse ohjelmiston tilaajien ja kehittäjien ratkaistavissa, mikä tapa sopii parhaiten kyseiselle tuotettavalle ohjelmistolle.

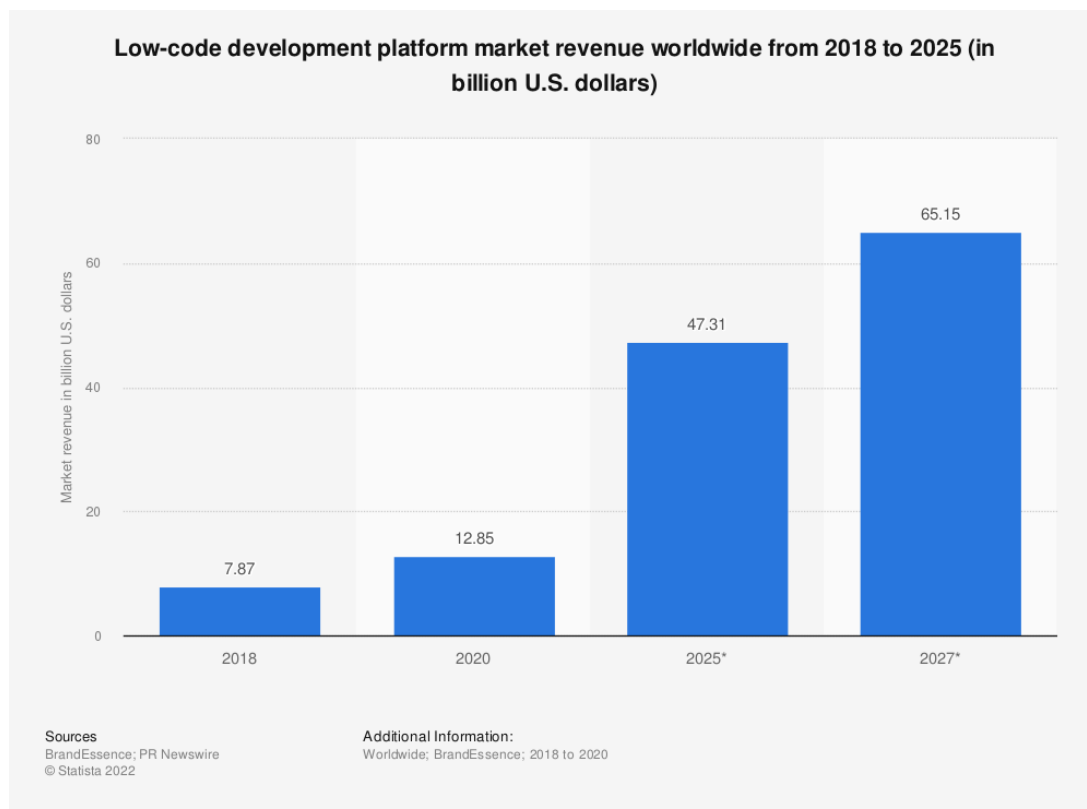
Low-code kehitystapana hyötyy kumminkin kyseisen alustan kehitystyötä tuntevan kehittäjän mukanaolosta toteutusprosessissa. Mikäli ohjelman toteutus tapahtuu esimerkiksi yrityksen sisäisesti, ja projektiin kiinnitettävillä henkilöillä ei ole ohjelmointitaitausta, eikä mahdollisuutta sitouttaa varsinaista ohjelmistokehittäjää mukaan projektiin, voi kehitysalustan valintana tällöin olla parhaiten sopivana no-codeen keskitetty alusta. Low-code kehityksessä, on ohjelmointiosaamisesta kuitenkin huomattavasti hyötyä, sillä alustalla luotavat ohjelmat tarvitsevat lisättyä ohjelmakoodia toimiakseen toivotuilla tavoilla tai rajatakseen haettavaa tietoa. Kehitysalusta itsessään nopeuttaa huomattavasti jo ohjelman tuottamisen prosessia, mutta ei kuitenkaan toteuta kaikkea valmiiksi käyttäjälle, vaan pikemminkin tarjoaa kehikon mille rakentaa tarvitsemansa ratkaisu. Tästä huolimatta kehitysalustat tarjoavat huomattavasti madalletun lähtökohdan ohjelman tuottamiseen, ja tarjolla olevien oppaiden sekä tutoriaalien johdosta, voi oman ohjelman kehittämiseen low-coden avulla päästä hyvinkin nopeasti kiinni.

Yrityksissä on usein henkilöitä, jotka tunnistavat liiketoiminnan tarpeet ja prosessit, mutta joutuvat käymään läpi työssään tarvitsemiaan tietoja joko käsin tai suurista Excel-taulukoista. Madaltamalla ohjelmistojen kehitykseen tarvittavaa tietotaitoa, ja mahdollistamalla itse ohjelman tuottaminen yhä useammalle, tuotetaan alan toimijoille sekä ohjelmistoja tarvitseville yrityksille kasvua ja kehitystä liiketoimintaa tukevien digitaalisten ratkaisujen kautta. Helpottamalla kynnystä ohjelmistokehitykseen, mahdollisesta myös näiden niin kutsuttujen kansalaiskehittäjien roolia ja osallistumista osana uusien ratkaisujen luontia, joilla nopeuttaa tai parantaa yrityksen eri liiketoimintojen osa-alueita. (Lindström 2021.)

Nopean kasvun, kehitystavan tuomien hyötyjen ja näistä seuraavan suosion vuoksi on oletettu, että vuoteen 2025 mennessä jopa 70 % uusista kehitettävistä sovelluksista voisi olla toteutettu käyttäen low-code alustoja, vastaavan luvun vuodelta 2020 ollessa 25 % (Stamford 2021). Mikäli nykyinen ohjelmistokehityksen suunta ja alustojen käyttöönotto jatkavat samalla tahdilla, on tämä korkealta kuulostava lukema hyvinkin mahdollinen.

2.6 Markkinatilanne

Low-code ratkaisujen kysyntä on yleistynyt nopeasti viime vuosien aikana, ja näin myös markkinoille on tullut useita eri kehitysalustaratkaisuja, joista kehittäjällä voi olla vaikeus valita juuri omiin tarpeisiinsa sopivaa ratkaisua. Alustojen kilpailuvaltteina toimivat tällä hetkellä yhdistämismahdollisuudet eri tietolähteisiin ja järjestelmiin, hinnoittelu, alustan helppokäyttöisyys sekä skaalautuvuus mahdollisten tarpeiden mukaan. Alustojen välinen kilpailu on kovaa, joka lisää osaltaan myös innovointia, ja tuo näin mukanaan uusia teknisiä ratkaisuja markkinoille. Kuten kuvio 1 osoittaa, markkinatilanne käy ennusteiden mukaisesti nopean nousun vaihetta läpi, low-coden nopean suosion vuoksi. Kehitysalustojen markkinoiden vuosittaisen tuoton ennustetaan kasvavan yli 50 miljardiin euroon vuoteen 2027 mennessä.



Kuvio 1. Low-code kehitysalustojen ennustettu markkinatuotto (Vailshery 2021)

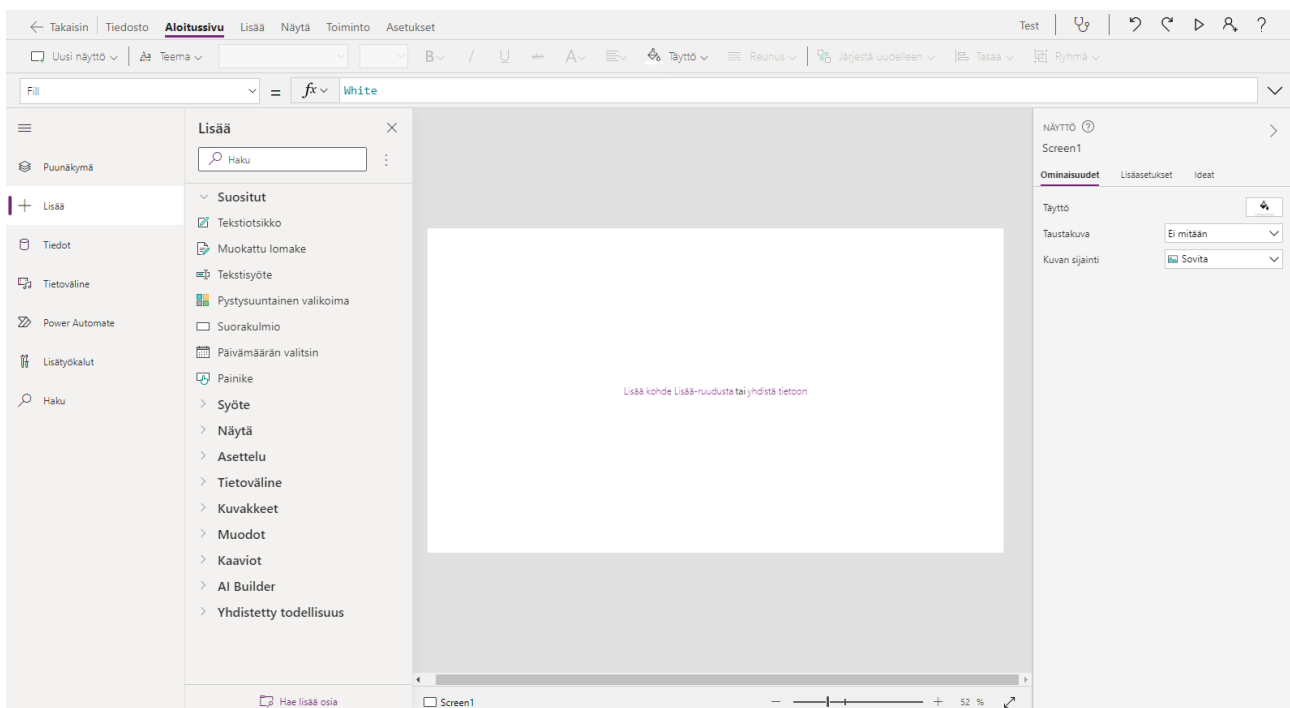
Tarve erilaisille sekä etenkin juuri tiettyyn tarkoitukseen toteutetuille ohjelmistoille on jatkuvassa kasvussa. Tähän tarpeeseen vastausta hidastaa osittain myös alaa vaivaava osaajapula. Tavallisessa ohjelmistokehityksessä on toivottavaa, että ohjelmistoa kehittäväällä työntekijällä on jo kokemusta itse kehityksen prosesseista sekä ohjelman toteutukseen valituista ohjelmointikielistä.

Osaajapulan vuoksi, projekteihin tarvittavia työntekijöitä voi olla vaikea löytää, josta johtuen yritykseltä voi jäädä jopa tarjous kokonaan tekemättä, ja projektin saa usein yritys miltä sopivan kokemuksen omaavia osaajia löytyy tarvittavasti.

Low-code on myös osaltaan vastaus markkinatarpeeseen, joka kaipaa kipeästi osaavia tekijöitä toteuttamaan uusia sovelluksia etenkin yritysten sisäisiin tarpeisiin. Yritysten digitalisaation edetessä ja vuonna 2019 alkaneen maailmanlaajuisen pandemian tuoman työtapojen muutoksen vauhdittaessa, yhä useampi yritys on tunnistanut tarpeen uusiin sisäisiin työkaluihin sekä muihin ratkaisuihin helpottamaan etätyöntekijöiden arkea.

2.7 Miten kehitys tapahtuu

Low-code kehitys eroaa tavallisesta kehityksestä koodin määrän lisäksi, myös sen visuaalisuuden kehitystavan vuoksi. Kehittäjä luo ohjelmistoa käyttöliittymästä, jossa hänellä on näkyvillä käytävissä olevat komponentit, liitettävät tietolähteet ja muut kehitykseen tarvittavat työkalut.



Kuvio 2. PowerApps kehitysalustan aloitusikkuna

Kehitys etenee lisäämällä tarvittavia näkymiä, joihin lisätään eri komponentteja, kuten tekstikenttiä tai taulukoita. Näihin komponentteihin voi lisätä tarvittavaa lisälogiikkaa, jolloin esimerkiksi käyttäjän syöttämä teksti saadaan asetettua käyttäjän käyttäjänimeksi tai taulukon tiedot saadaan syötettyä tietokantaan talteen. Sovellukseen voi luoda useita näkymiä, sisältäen eri toiminnallisuuksia ja näiden välisiä siirtymämahdollisuuksia, riippuen käyttäjän toiminnasta. Tarvittavat ulkoasun, kuten kuvan tai tekstin syöttökentän muokkaukset voi toteuttaa samalla käyttöliittymällä, ja valmis tuote on aina testausvalmiina kehittäjälle tai käyttäjälle.

2.8 Liitettävät järjestelmät ja tietolähteet

Low-code ratkaisujen päätoimena on kerätä tietoa eri lähteistä ja näyttää nämä yhden sovelluksen sisällä. Tämänkaltaisia ratkaisuja ovat esimerkiksi pääkäyttäjän työpöydät, joissa pääkäyttäjänä toimiva henkilö pystyy näkemään esimerkiksi hallinnoimansa palvelun tilan eri osa-alueet useasta eri lähteestä. Hyötynä tämänkaltaisessa ratkaisussa on tiedon saatavuuden kasvaminen ja työn nopeuttaminen, kun hallinta voi tapahtua vain yhden sovelluksen kautta. Liitosmahdollisuudet vaihtelevat kehitysalustoittain, mutta sisältävät pääsääntöisesti yleisimmät tietomuodot kuten käytetyimmät tietokannat, Excel-tilukot ja yleisimmin käytetyt tietolähteet.

Microsoft on julkaissut yhä kehityksessä olevan tietokantaratkaisunsa Dataversen, joka vastaa tietoturvalliseen tietojen säilyttämisen tarpeeseen PowerApps kehitysalustaa käyttäessä. Dataverse toimii Microsoftin tuoteperheen omana low-code tietokantana, ja hyödyntää pilvitallennustilaa tietojen säilyttämisessä. Dataverseen on myös mahdollista synkronoida säännöllisesti tietoa muista järjestelmistä ja näin hyödyntää yhtä alustaa kaiken tiedon lähteenä. (What is Microsoft Dataverse 2022.)

2.9 Varjo-IT - ongelma

Yksi tiedostettu ongelma IT-alalla on niin kutsuttu varjo-IT. Tämä kuvaa ilmiötä, missä työntekoon käytetään työkaluja kuten ohjelmistoja, joita yrityksen oma IT ei hallinnoi, tai joiden käytöstä yrityksen sisällä ei tiedetä yleisellä tasolla ollenkaan. (Person 2022.) Esimerkiksi jos yrityksen työntekijä luo asiakkaalle ohjelman alustalla, jonka käytöstä kukaan muu ei yrityksessä tiedä, tai jos työntekijä käyttää projektin hallintaan työkalua mitä ei ole otettu yrityksessä laaja-alaisesti käyttöön.

Syy näihin tilanteisiin löytyy oletettavimmin yrityksen virheestä tunnistaa työntekijöiden tarvetta tietylle ohjelmistolle, tai työntekijöiden ratkaisusta olla kertomatta tarpeestaan.

Tämänkaltaiset ohjelmat voivat tuoda ongelmia työntekijän vaihtaessa työpaikkaa tai olla suuri turvallisuusriski, mahdollistaessaan pääsyn ohjelman kautta jopa koko yrityksen sisäiseen verkkoon. Low-code ratkaisulla yritetään vastata tähän ongelmaan, tarjoamalla työkaluja, joilla nopeuttaa yksinkertaisia, aikaa vieviä, helposti toistettavassa olevia tehtäviä. Tuotetuilla ratkaisulla saadaan automatisoitua näitä tehtäviä, ja vapautettua resursseja tärkeämpiin työtehtäviin, ja näin myös edistettyä työntekijän työhyvinvointia. (Person 2022.)

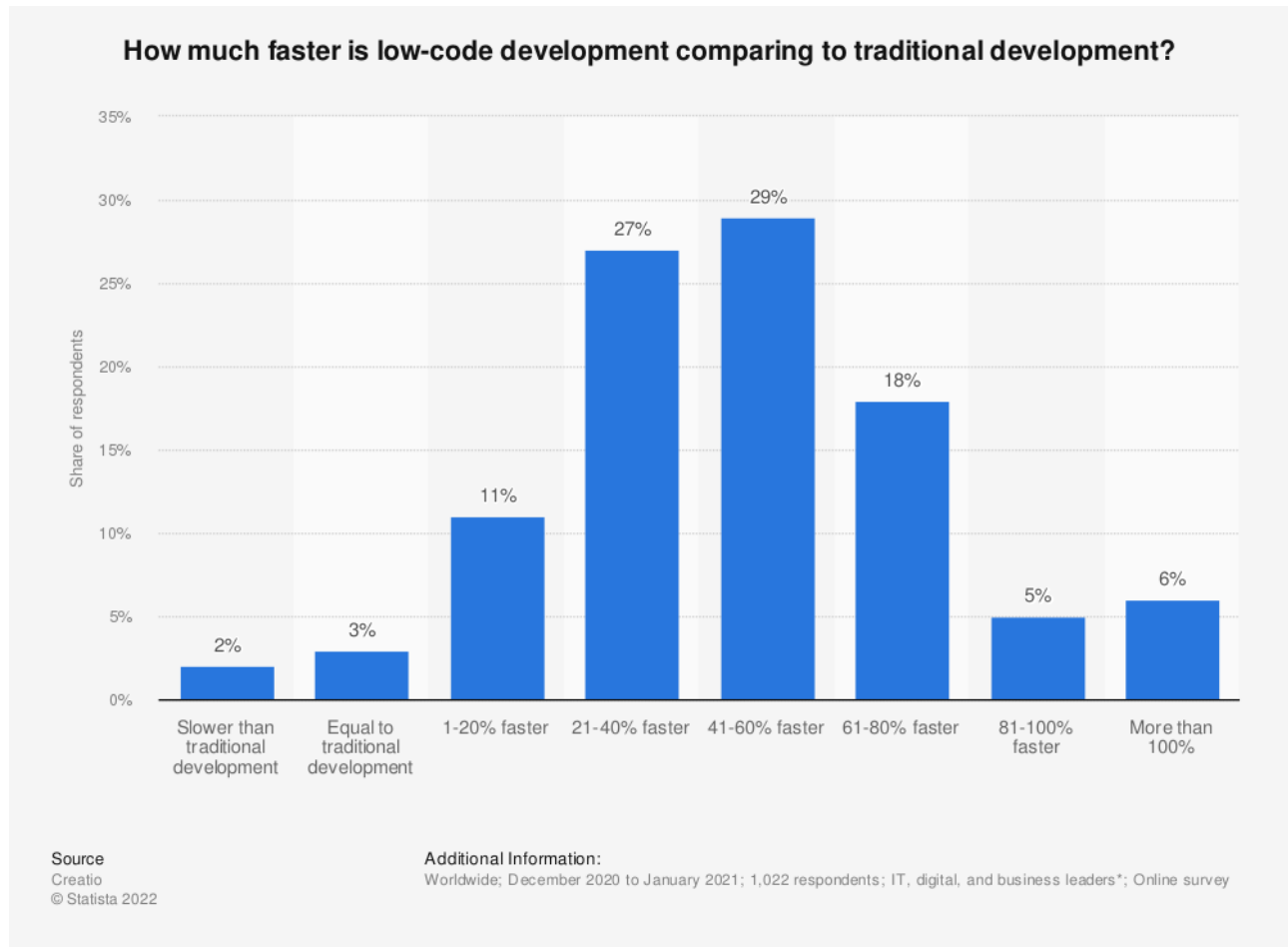
2.10 Hyödyt

Ohjelmistokehityksessä, ohjelmasta suunnitellaan aluksi ensiversio, johon kehitys alkaa tähtäämään. Tästä voidaan käyttää myös termiä MVP (Minium Viable Product) eli suoraan suomennettuna pienin kannattava tuote. (Citizen Development - The Handbook for Creators and Changemakers 2022.) Tällä pyritään saamaan mahdollisimman nopeasti tuotettua tuote, jota loppukäyttäjä pääsee testaamaan, ja jotta mahdolliset koekäytössä huomattavat asiat saadaan korjattua tai muokattua palautteen perusteella.

Low-code tarjoaa tähän nopean reitin, sillä kehitteellä oleva tuote on jatkuvasti testattavissa kehitysalustalla, vaikka kaikkia toiminnallisuuksia ei olisi vielä tuotteeseen toteutettukaan. Verrattuna tavalliseen kehitykseen, nopeus tuotannosta valmiiksi tuotteeksi kasvaa low-code ratkaisulla huomasti (ks. kuvio 3). Tästä seurauksena myös sovelluksen tilaaja saa itselleen toimivan tuotteen entistä nopeammin.

Nopeus onkin isossa roolissa low-code ratkaisujen ydintä. Vuonna 2021 tehdyn kyselyn mukaan (ks. kuvio 3) low-code kehityksen nopeus oli keskivertovastaajan mielestä selkeästi nopeampaa verrattuna perinteiseen ohjelmistokehitykseen. Mikäli esimerkiksi sisäiseen käyttöön tarkoitetun ohjelmiston saa tuotettua jopa puolet nopeammin, tarkoittaa se suoraan säästettyä työaikaa ja näin myös säästöä tuotteen toteuttavalle yritykselle. Mikäli lopputuloksena oleva tuote toimii kuten sille asetetut vaatimukset määrittävät, on selvää miksi low-code voi olla ensimmäisenä ratkai-

suvaihtoehtona tämänkaltaista sovellusta suunnitella. Tämä mahdollistaa esimerkiksi ohjelmistoyritykselle täsmälliset räätälöidyt sisäiset työkalut ja nopean mahdollisuuden esimerkkituotteen esittelyyn asiakkaalle.



Kuvio 3. Low-code kehityksen koettu nopeus verrattuna tavalliseen kehitykseen (Vailshery 2022)

Näistä seuraavat kustannussäästöt ovat sekä tuottavan yrityksen, että asiakkaan etuina. Matalien kustannusten ja kehityksen nopeutuvan mukautumismahdollisuuksien ansiosta, tuotannon kulut pysyvät matalina ja itse kehitys ketteränä. Low-code tuotteet skaalautuvatkin vaivattomasti tarkasti toteutettujen taustakoodigeneraattorien ansiosta. Esimerkiksi prosessoidun datan tai käyttäjäkunnan määrän kasvaessa, tavallisessa ohjelmistokehityksessä voidaan päätyä koodin optimointiin sovelluksen tehokkaan toiminnan ylläpitämiseksi, joka vie aikaa, tuottaa lisää kustannuksia ja voi pahimmassa tapauksessa jopa hetkellisesti haitata kriittisesti asiakasyrityksen toimintaa.

Koska alustat luovat tarvittavan taustakoodin sillä hetkellä, kun käyttäjä asettaa komponentin kehitysalustan käyttöliittymässä, on toteutettava ohjelma jatkuvasti testausvalmis, eikä siitä tarvitse esimerkiksi koota erikseen uusien ominaisuuksien testausta varten. Tämän ansiosta ohjelman kehittäjä voi jatkuvasti testata ohjelmaa, ja tehdä tarvittavat muutokset heti uuden ominaisuuden lisäämisen jälkeen. Tämä mahdollistaa myös jopa asiakkaan sitouttamisen kehitykseen vielä syvemmin, sillä kehitteillä olevan ohjelman sen hetkinen tila on jatkuvasti nähtävissä ja testattavissa. Tätä kautta myös mahdollisiin ohjelman muutostarpeisiin voidaan reagoida heti, eikä näistä aiheudu tavallisen ohjelmistokehityksen kaltaista suurta ohjelmakoodin muokkauksen tarvetta.

Low-code alustat mahdollistavat paljon samoja asioita mitä tavallinen ohjelmistokehitys tarjoaa, vaikka näillä tuotetut ohjelmat ovatkin alustan rajoitteiden vaikutusten alla. Aiempaa kokemusta omaavalle ohjelmistokehittäjälle, low-code alustat tarjoavatkin hyvät toteutusmahdollisuudet toteuttaa ohjelmistoja entistä nopeammin ja tehokkaammin. Toimintakoodin muokkauksella kehitysalustasta riippuen, voidaan saavuttaa jopa erittäin komplekseja ohjelmia, jotka pystyvät samaan mitä suuremman kokoluokan projekti voisi toteuttaa paljon hitaammalla aikataululla.

2.11 Haitat ja rajoitteet

Low-code alustat kuten mitkä tahansa muutkin vaihtoehdot, tuovat tullessaan rajoituksia tai haittoja kehitykseen, valmiiseen tuotteeseen, tai sen ylläpitoon. Alustan valitsemisessa tulee ottaa huomioon esimerkiksi, mihin palveluntarjoajan ekosysteemiin on suostuvainen sitoutumaan, mitä mahdollisia rajoitteita alustalla voi olla sekä mihin muihin sovelluksiin tai palveluihin tuotettava ohjelma tarvitsee yhteyksiä. Alustojen luomat taustakoodit, mitkä hallitsevat ohjelman toimintaa, eivät nimittäin ole useinkaan yhteensopivia keskenään esimerkiksi kehitysalustan vaihdon tilanteessa.

Koska kehitys on yleisimmin rajoittunut yhdelle kehitysalustalle, tuo tämä rajoitteena kyseisen alustan tarjoajan tuottamat vaihtoehdot. Kehitysalustoille ei esimerkiksi pysty lisäämään ylimääräisiä itse toteutettuja komponentteja, josta johtuen ohjelmiston tarvitsevat täsmälliset räätälöinnit voivat jäädä näistä kiinni. Alustaa käyttävä kehittäjä onkin tästä syystä kehitysalustaa tuottavan yrityksen innovointiin ja ratkaisuihin rajattuna. Näissä tilanteissa nouseekin esille kysymys, tahdotaanko välttämättä täsmällisesti tarpeita noudattava ratkaisu, vai voidaanko tyytyä mahdollisesti

pieneen eroavaisuuteen toivotussa toiminnossa tai käyttöliittymässä. Alustaa valittaessa on kannattavaa ottaa selville mihin muihin järjestelmiin, tietokantoihin tai tietolähteisiin kyseisen alustan pystyy yhdistämään ja verrata näitä tuotettavan ohjelman tarpeisiin.

Tietoturva on noussut yhtenä osa-alueena koko ajan suuremmaksi tekijäksi ohjelmistojen tuotannossa. Etenkin arkaluonteista tietoa käsitellessä, tulee ohjelmistokehittäjän olla tietoinen mahdollisista haavoittuvuuskohdista tai tiedon vuotoriskeistä. Valittaessa low-code alustaa, tullaan samalla valinneeksi myös sovelluksen tietoturvan arkkitehti, valmiiden sovellusten taustakoodin ollessa kehitysalustan tuottamaa koodia. Esimerkiksi jos jokin taustalla generoitu funktio ei toteuta käyttäjän syötteen validointia oikein, ei itse valmiin sovelluksen kehittäjä välttämättä ole tietoinen riskitekijästä ilman tarkempaa testausta. Kehitysalustan valinnassa kehittäjä oletetusti luottaa, että kehitysympäristön taustakoodin generaattori on suunniteltu toimimaan tarpeeksi tietoturvallisesti taatakseen hänen luomansa ohjelman tietoturvan.

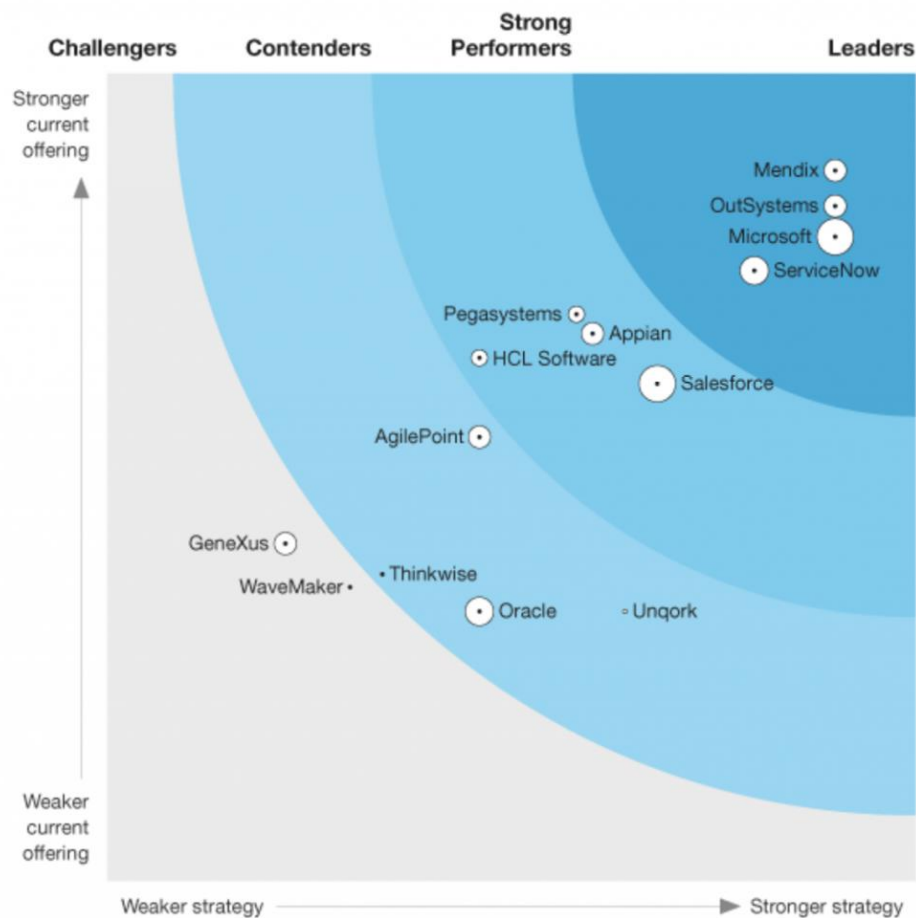
Matalana haittana low-code ohjelmistokehityksessä on koodin muokattavuuden vuoksi myös mahdollisuus ohjelman rikkoutumiseen. Mikäli kehitysalustan taustakoodiin tulee esimerkiksi muutos kohdassa, mihin kehittäjä on lisännyt omaa koodia. Tämä on kuitenkin matalatasoinen haitta, sillä tulevien muutoksien sisällöstä ja mahdollisista vaikutuksista tuotteisiin yleensä tiedotetaan alustaa käyttäviä kehittäjiä. Tästä johtuen tarvittavat muokkauksetkin on mahdollista toteuttaa nopeasti, ilman isompaa ongelman tutkintaa.

Ominaisuuksien puuttuminen tai näiden rajallisuus ollessa lukittautuneena tiettyyn kehitysalustan tuottajaan on yksi isoimmista low-code kehityksen tuomista haitoista. Tähän ongelmaan vastauksena esimerkiksi Microsoft kerää foorumityyppisellä ratkaisulla kokoelmaa käyttäjien ideoimista uusista ominaisuuksista tai muokkauksista. Tässä käyttäjät pystyvät äänestämään eri ideoita ja näistä pyydettyimmät lopulta toteutetaan ja implementoidaan osaksi PowerApps:in ominaisuuksia.

3 Tutkimusasetelma

3.1 Tutkimuskohde

Tutkimuksen kohdealustaksi valikoitui Microsoftin tuottama low-code alustaratkaisu PowerApps. Ensimmäinen julkinen ilmoitus tästä tapahtui Microsoftin blogissa vuonna 2015, jossa sen kerrottiin olevan yrityksen ratkaisu yritysten sisäisten työkalujen ratkaisujen haasteisiin ja taitavien mobiilisovellusten kehittäjien määrän ongelmiin (Staples 2015). Se julkaistiin ensimmäisen kerran julkiseen koekäyttöön huhtikuussa vuonna 2016 (Desai 2016) ja lopulta virallinen julkaisu tapahtui saman vuoden lokakuussa (Phillips 2016). PowerAppsin sovellukset ovatkin toteutettu mobiili ensin -periaatteella, jossa otetaan huomioon ensin mobiilikäyttäjien näkymä sovellusta käyttäessä. Tällä tuotetut ratkaisut ovat myös integroitavissa osaksi Microsoftin omaa ekosysteemiä, esimerkiksi SharePoint-sivulle tai erilliseksi Teams-sovellukseksi. PowerApps valikoitui tutkimuksen kohteeksi ollessaan yksi hallitsevista low-code alustojen tuottajista (ks. kuvio 4), ja Microsoftin brändin tunnettavuuden vuoksi.



Kuvio 4. Eri Low-code kehitysalustojen tarjonnan ja strategian tila (Ramel 2021)

PowerApps perustuu ympäristöpohjaiseen käyttöön, jossa käyttäjä pystyy luomaan toisistaan irrallisia ympäristöjä, joihin tuottaa erilaisia sovellusratkaisuja, niin että eri ympäristöissä olevat sovellukset voivat käyttää esimerkiksi vain kyseiseen ympäristöön määriteltyjä yhteyksiä. Tämä mahdollistaa laajempien kokonaisuuksien helpomman hallinnan sekä rajaa tarvittavia yhteyksiä ja taulujen käyttöä ympäristöjen tarpeiden mukaan.

PowerAppsin muokattavassa koodissa käyttämä PowerFX-ohjelmointikieli on saanut inspiraationsa Excel-ohjelman käyttämisestä funktioista sekä tiedon hallintaan tarkoitettuista komennoista. Microsoft kertookin tämän olevan juuri tarkoitettuna auttamaan siirtymää etenkin bisnestaustaisille käyttäjille, keillä ei välttämättä ole enempää kokemusta ohjelmoinnista, kuin Excel-taulukkojen eri funktioiden osalta. Tämä helpottaa omalta osaltaan kyseisen ryhmän valmiutta toteuttaa eri sovellustoteutuksia, tarjoten tuttuja kaavoja, joilla suodattaa, järjestellä tai laskea tietoa. Mikäli vastaavaa toimintoa toteuttavaa komentoa ei jotain toimintoa varten löytynyt, otettiin siihen mallia seuraavaksi yleisestä tiedonkäsittelykielestä eli SQL-tietokantakielestä. (Microsoft Power Fxin yleiskatsaus 2022.)

3.2 Tavoitekuvaus

Tutkimuksen tavoitteena oli tutkia low-code ohjelmistokehityksessä käytävää prosessia kokonaisuutena sekä tutkia sen kasvavan suosion syitä käytännön tasolla. Tutkimusta tehdessä tarkkailtiin myös itse kehitykseen liittyviä osa-alueita, käyttäessä valittuna alustana Microsoftin PowerApps kehitysalustaa. Sovellusta kehittäessä pyrittiin huomioimaan tietoperustassa mainittuja etuja sekä haittoja sekä tarkasteltiin vielä tarkemmin low-code ratkaisun pohjalta toimivan sovelluksen luonnin ominaisuuksia, verrattuna tavalliseen ohjelmistokehitykseen. Low-code kehityksestä ja etenkin sen tulevaisuudesta on tuotettu paljon sisältöä, jonka perusteella tutkimukselle voidaan asettaa pohjaoletuksia, joita vastaan asettaa vastakkain esimerkkisovelluksen tuottamisessa koettu.

Tutkimuksen oletukset:

1. Low-code-ohjelmistokehitys on nopeampaa kuin tavallinen ohjelmistokehitys
2. Low-code alustalla on helppo toteuttaa toimiva ohjelma
3. Low-code mahdollistaa matalan kynnyksen reitin ohjelmointiin
4. Low-code alustalla tuotettu tuote on yhtä toimiva, kuin tavallisen ohjelmistokehityksen tuote

Jotta näihin oletuksiin saadaan käytännön toiminnan kautta verrattava kokemuspohja, määritellään ohjelman tuottamisessa huomioon otettavat ja tarkkailtavat ohjelmistokehityksen osa-alueet seuraavilla kysymyksillä.

Tuotannon seurantakysymykset:

1. Kuinka nopeasti kehityksessä pääsee alkuun?
2. Millainen kokemus sovelluksen tuottamisesta syntyi?
3. Mitkä olivat suurimmat hidasteet kehityksen etenemisessä?
4. Mitkä olivat suurimmat rajoitteet kehityksessä?
5. Mitä koettuja hyötyjä ilmeni kehityksessä?
6. Esiintyikö sovellusta kehittäessä haasteita tai ongelmia?
7. Vastasivatko tämänhetkinen suosio ja mittaukset ohjelman kehityksessä koettua?
8. Päästiinkö suunniteltuihin toiminnallisuuksiin?

3.3 Suunnitelma

Toteutettavaksi sovellukseksi valittiin projektinhallinnan työkalu, joka mahdollistaa projektiin liittyvien tehtävien hallinnan, jakamisen projektin jäsenille sekä projektin yleistietojen täyttämisen ja seurannan kaikille projektin jäsenille yhteisestä sovelluksesta. Sovellus toteutettiin kuvitteelliselle yritykselle. Sovelluksen tarkemmat tiedot löytyvät sovelluksen vaatimusmäärittelystä liitteestä 1. Sovelluksen käyttötarkoituksen valinnan taustalla oli tuottaa käytännöllinen esimerkki, jonka voisi ottaa käyttöön missä vain yrityksessä, ja tuottaa samalla mahdollisimman oikeanmukaista tietoa kehityksen eri osa-alueista.

Koska tutkimus tapahtui, ja tutkimuksen tulokset ovat johdettuja yhden henkilön kokemuksen perusteella, ovat tutkimuksen tulokset vain suuntaa antavia. Tuloksia arvioidessa tulee ottaa huomioon, että vastaavaa tutkimusta toteuttaessa, saadut tulokset voivat vaihdella huomattavasti riippuen tutkimuksen toteuttajan mahdollisista eri oppimisen tyyleistä, keskinäisten asioiden sisäistämisestä sekä aiemmin oppiman soveltamisesta käytäntöön. Tutkimuksen vastakkainasetteluna ovat vahvasti julkisesti saatavilla olevien kyselyiden ja aiheen ympärillä olevan suosion ja nosteen lähtökohdat ja näistä johdetut oletusarvot.

3.4 Tutkimuksen rajaus

Tutkimuksesta rajattiin pois eri kehitysalustojen vertailu, vertailukohteiden ominaisuuksista johtuen. Jokainen kehitysalusta on itsessään hyvin laaja, kattaen usein eri toimintoja, ja erilaisia toteutustapoja ohjelmistoratkaisuille. Tarkempi perehtyminen eri kehitysalustoihin vaatisi syvällisempää tutustumista näihin, ja tämä ohjaisi resursseja pois itse sovelluksen kehityksen tutkinnasta ja pääaiheesta. Itse sovelluksen kehittäminen tapahtuu jokaisella alustalla pääosin samalla tapaa, joten saatu lisäarvo tutkimukselle olisi pieni verrattuna tutkimuksen määrään.

Tutkimus ei perehdy tarkemmin kehitysalustan eri funktioihin tai taustalla toimivaan järjestelmään, joka luo tarvittavan ohjelmakoodin, kyseisten aihealueiden mittavuuden vuoksi. Eri kehitysalustoilla on samankaltaisten ominaisuuksien lisäksi myös erilaisia alustakohtaisia toimintoja ja omat taustatoiminnallisuudet koodin generointiin. Nämä ovat mittavia itsessään, ja näihin toimintoihin perustuvat alustojen toiminta, jonka vuoksi myös näihin pääsy on estetty pääosin muissa alustoissa paitsi vapaan lähdekoodin alustoilla. Tutkimuksen keskittyessä käyttäjäkokemukseen tietyn kehitysalustan käytöstä, on rajaus tarpeellinen alueen tuoman lisäarvon puuttuessa.

Tutkimuksen tulosten näkökulma rajoittuu ohjelmistoalalla jo hieman kokemusta saaneen henkilön näkökulmaan, eikä ota huomioon muita näkökulmia, kuten kokemattoman tai enemmän kokeneen mahdollisia kokemuksia ohjelman kehittämisestä. Tarkempaa käyttäjäkokemuksen näkökulman tarkastelua varten, tarvittaisiin otos käyttäjäkokemuksia eri tasoilta henkilöiltä, ja tutkimus olisi haasteellisesti mitattavissa erilaisten oppimistapojen ja kykyjen muuttujien vuoksi, eikä tämä toisi lisäarvoa tutkimukselle, joka keskittyy pääasiassa itse alustalla toteutettavaan kehitystyöhön.

4 Tulokset

4.1 Tutkimuksen aloitus

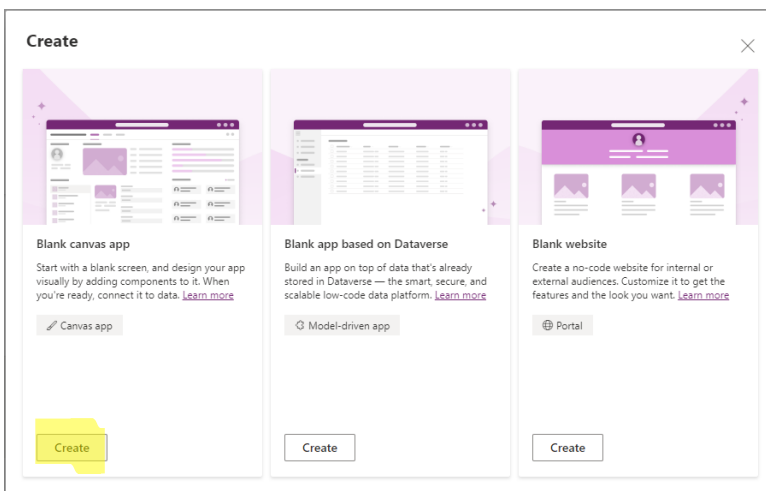
Sovellukselle luotiin aluksi suppea vaatimusten määrittely ja tarkoitusperä (ks. liite 1). Tälle kuvailtiin kohderyhmä, tarkoitus sekä tavoitellut vaatimukset eri toiminnallisuuksille. Varsinainen ohjelman kehitys aloitettiin luomalla tietokanta ennalta suunnitellun tietokantakuvauksen pohjalta, Microsoftin Dataverseä käyttäen.

Dataverseen luotiin ohjelmalle suunnitellut taulut sekä näiden välille lisättiin suhteet tietojen välistä yhteyksiä varten. Sovellus määriteltiin hakemaan Users - taulusta tietoja käyttäjistä, joilla ovat vaadittavat lisenssiedellytykset PowerAppsin käyttöön. Näin mahdollistettiin sovelluksen käyttö, käyttäjän ollessa kirjautuneena selaimella Microsoft – tililleen. Näin sovellusta varten ei tarvinnut luoda erillisiä käyttäjätunnuksia. Samalla myös kuvitteellisen yrityksen käyttäjätietokantaan lisätty uusi käyttäjä olisi suoraan valmis lisättäväksi projekteille, käyttäjän lisenssien ollessa kunnossa.

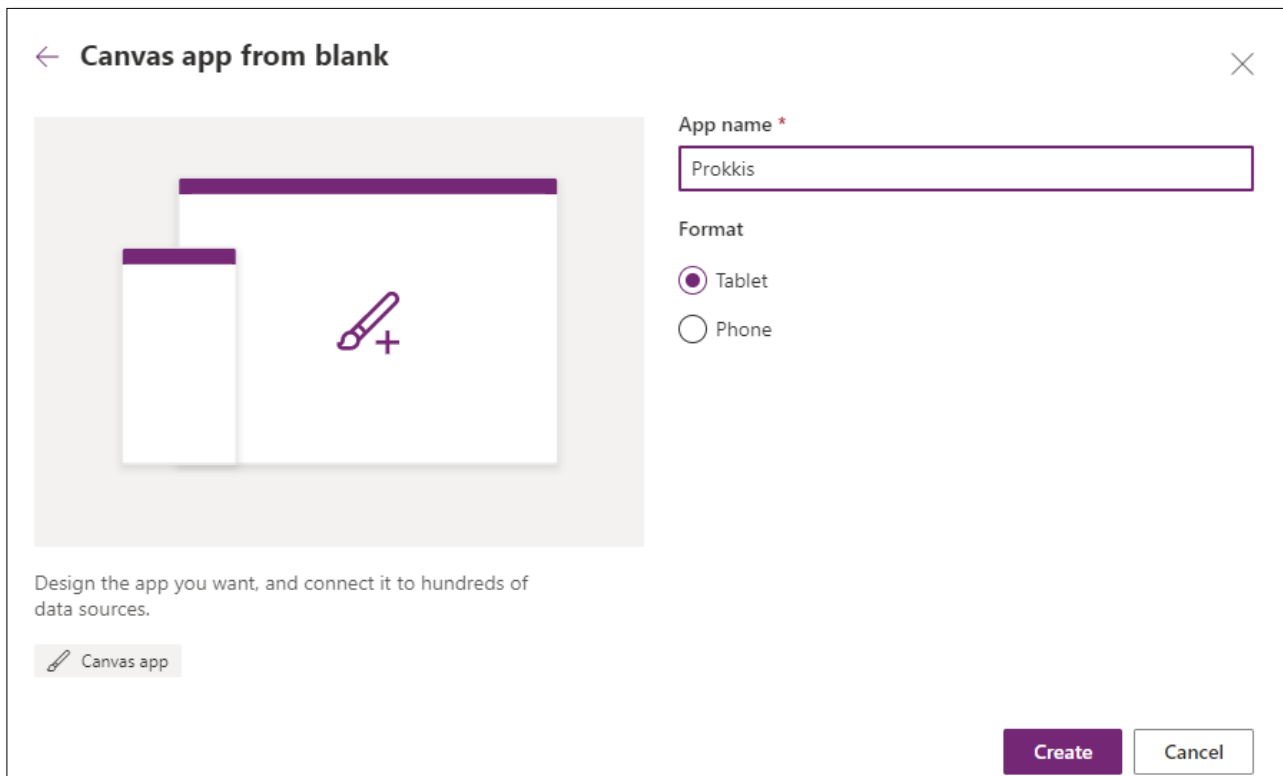
Sovellusta aloitettiin toteuttaa mahdollisimman vähäisellä ohjeistuksen seurauksella, mahdollistaen mahdollisimman autenttisen kokemuksen siitä, kuinka nopeasti uuden sovelluksen pystyy toteuttamaan ilman aiempaa kokemusta.

4.2 Sovelluksen luonti

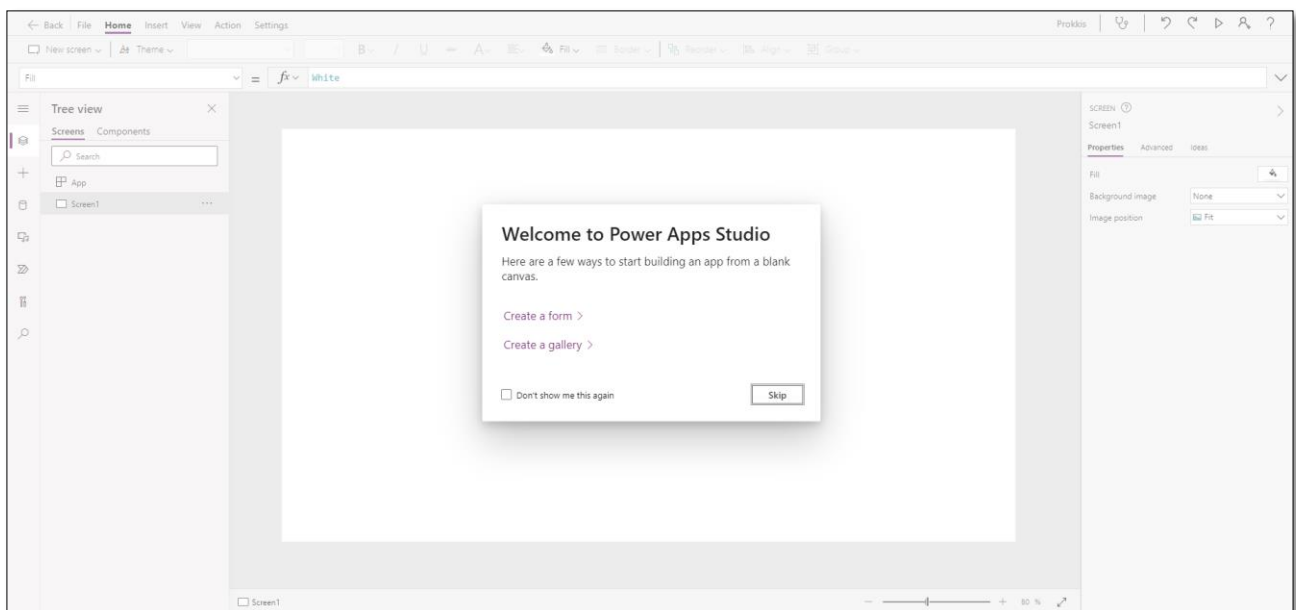
Sovellus luotiin valitsemalla sovelluksen pohjaksi tyhjä canvas-tyyppinen sovellus. Tämä luo uuden sovelluksen, ilman alustavia pohjia tai esiasetettuja tietoyhteyksiä sovellukseen.



Kuvio 5. Uuden sovelluksen vaihtoehdot



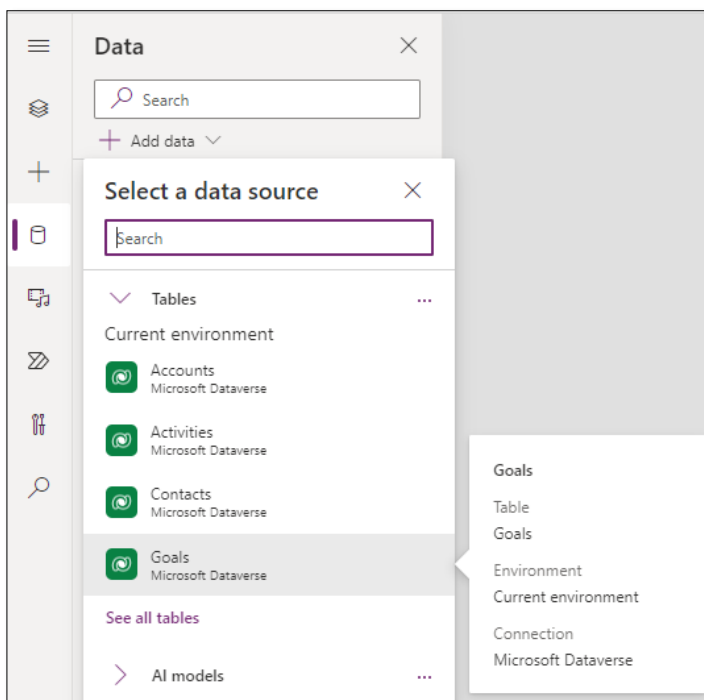
Kuvio 6. Tyhjän sovelluksen luonnin aloitusvalinnat



Kuvio 7. Uuden sovelluksen aloitusnäky

4.3 Tietojen yhdistäminen sovellukseen

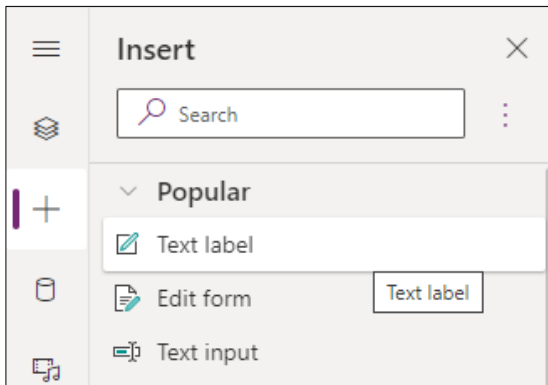
Aiemmin luodun Dataverse-tietokannan taulut yhdistettiin sovellukseen lisäämällä yhteys jokaiseen tauluun sovelluksen käytettäväksi. Näin tauluista haettavia tietoja pystyttiin yhdistämään luotaviin komponentteihin ja näyttämään näitä tietoja käyttäjälle. Sovellukseen lisättiin myös alustavasti yhteys Azure Active Directoryyn, joka hallinnoi aina kyseisen organisaation käyttäjiä, mutta tämä todettiin jälkikäteen tarpeettomaksi, sillä Dataversessä vakiona oleva User-taulu haki aina kyseisen organisaation lisensoidut käyttäjät automaattisesti.



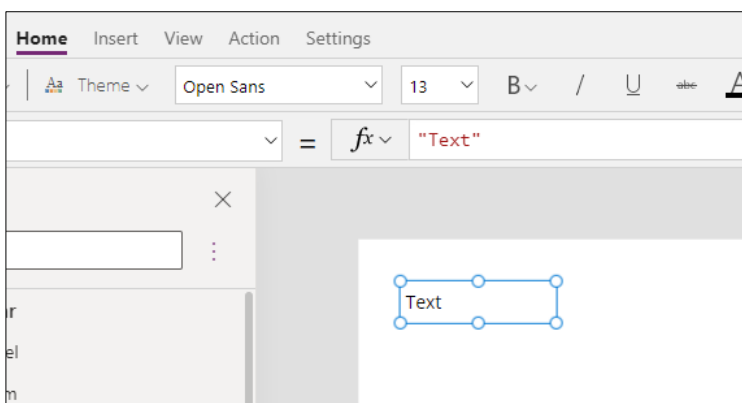
Kuvio 8. Tietolähteiden yhdistäminen

4.4 Komponenttien lisäys

Aloitussivun hahmottelu aloitettiin lisäämällä uusi tekstikomponentti, sovelluksen nimen näyttämistä varten.

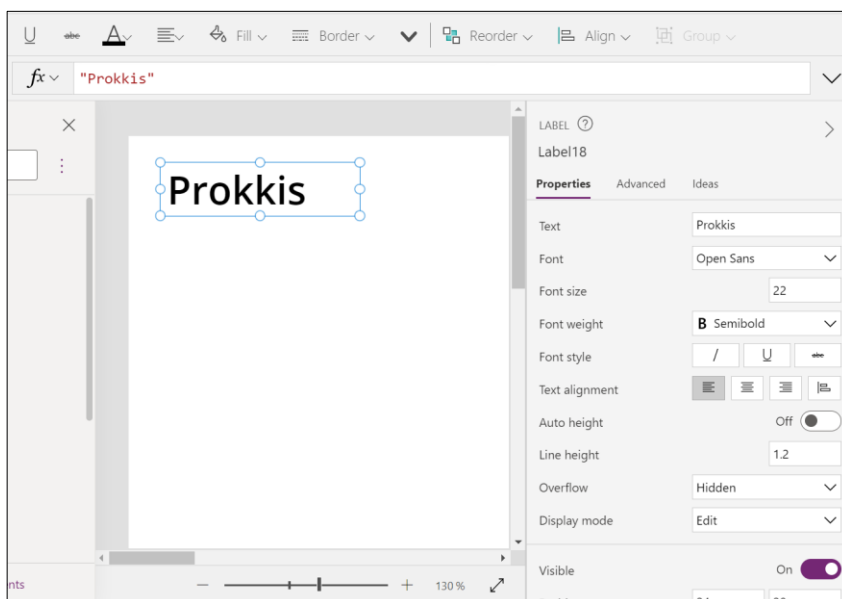


Kuvio 9. Uuden tekstikomponentin lisäys



Kuvio 10. Lisätty tekstikomponentti näytöllä

Teksti muutettiin sovelluksen nimeksi, ja aseteltiin oikealle kohdalleen. Komponentin eri ominaisuuksia, kuten fonttia, kokoa tai väriä pystyi säätelemään komponentin asetusten paneelista.



Kuvio 11. Komponentin tarkemmat asetukset

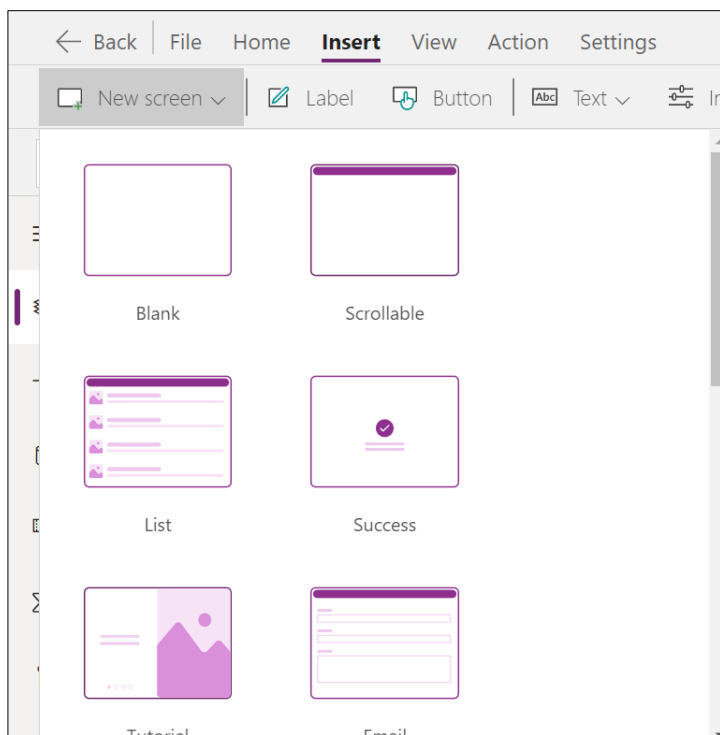
Eri komponentteja käytettiin näyttämään eri tietoja myös funktioiden avulla. Kirjautuneena olleen käyttäjän tietoja pystyi esimerkiksi hakemaan funktioilla `User().Image` ja `User().FullName`. Näitä yhdistämällä käyttämällä kuva- ja tekstikomponenttia, luotiin yhdistelmäkomponentti (ks. kuvio 12) mikä näytti käyttäjän profiilikuvan sekä nimen.



Kuvio 12. Käyttäjän tietojen näyttö usealla komponentilla

4.5 Näyttöjen lisäys

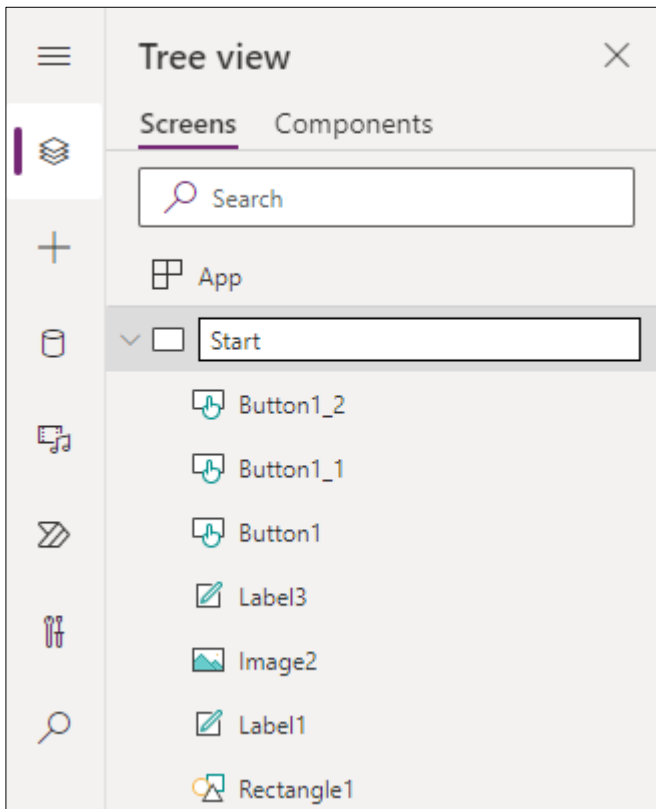
Sovellukseen lisättiin näyttöjä (ks. kuvio 13) eri ominaisuuksille, esimerkiksi projektin tai tehtävän yksityiskohtaisempaan tarkasteluun ja hallintaan. Näyttöjen välillä siirtyminen tapahtui painikkeiden ja eri siirtymien kuten uuden tehtävän luonnin kautta.



Kuvio 13. Uuden näytön lisäys

4.6 Näyttöjen ja komponenttien hallinta

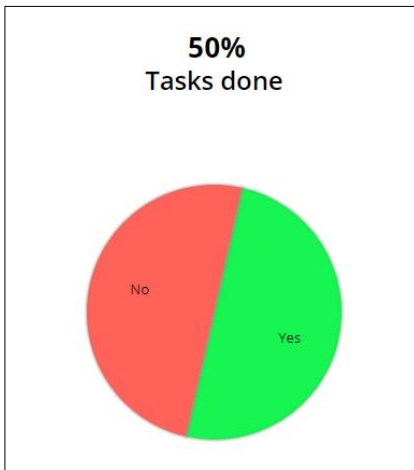
Sovelluksen osioiden hallinta tapahtui Tree view – paneelin (ks.kuvio 14) kautta, jonka listauksessa näkyivät sovelluksen kaikki eri näytöt ja komponentit.



Kuvio 14. Tree view näkymä

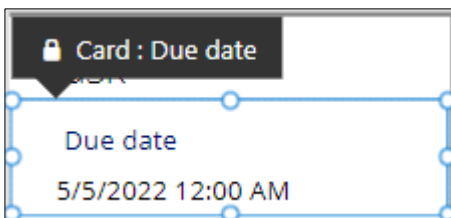
4.7 Erilaiset komponentit

Projekteista saatavaa tietoa hyödynnettiin valmiilla kaaviokomponentilla (ks. kuvio 15). Nämä li-sättiin projektin tavoitteiden osioon visualisoimaan jäljellä olevien tehtävien määrää projektilla, ja antamaan havainnollistavan kuvan projektin tehtävämäärän sen hetkisestä tilasta. Haettava tieto suodatettiin projektikohtaiseksi sekä piiraskaavion tieto-osiot määriteltiin näyttämään valmiina olevien tehtävien osuutta.

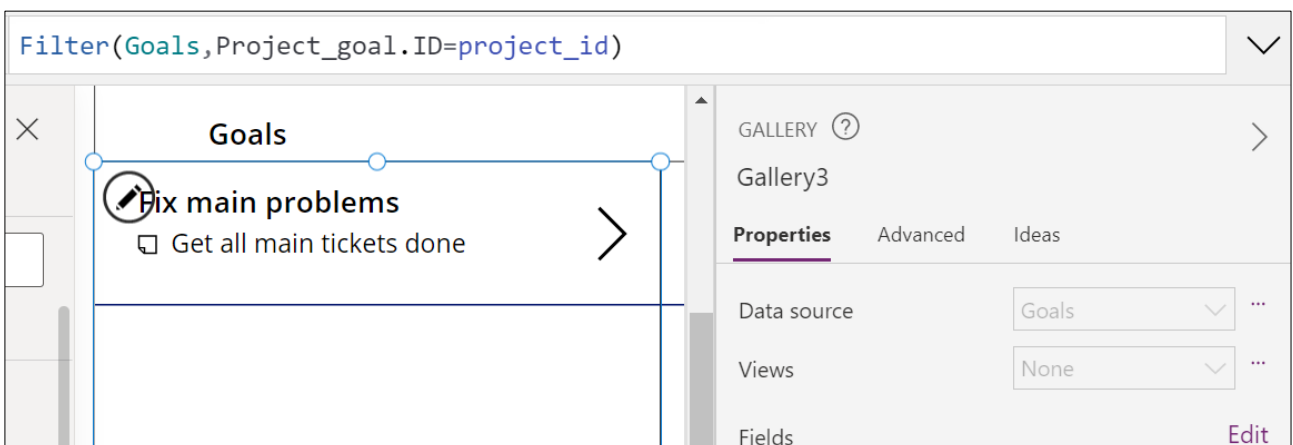


Kuvio 15. Tiedon visualisointi

Valmiita tietokortteja käytettiin suodatetun tiedon eri kenttien näyttämiseen yksittäisten kohteiden tieto-osioissa. Tietokortteihin määriteltiin oikeat tarvittavat tietolähteet ja näiden tietojen tarvittavat suodattukset lisättiin tiedonhaun funktioihin. Tietokortit (ks. kuvio 16) ja galleria-tyyppiset komponentit (ks. kuvio 17) hakivat ja näyttivät automaattisesti määritellyn tiedon tietolähteestä, antoivat hyvät määrittelymahdollisuudet tarvittavan tiedon monimutkaisempaankin rajaukseen.

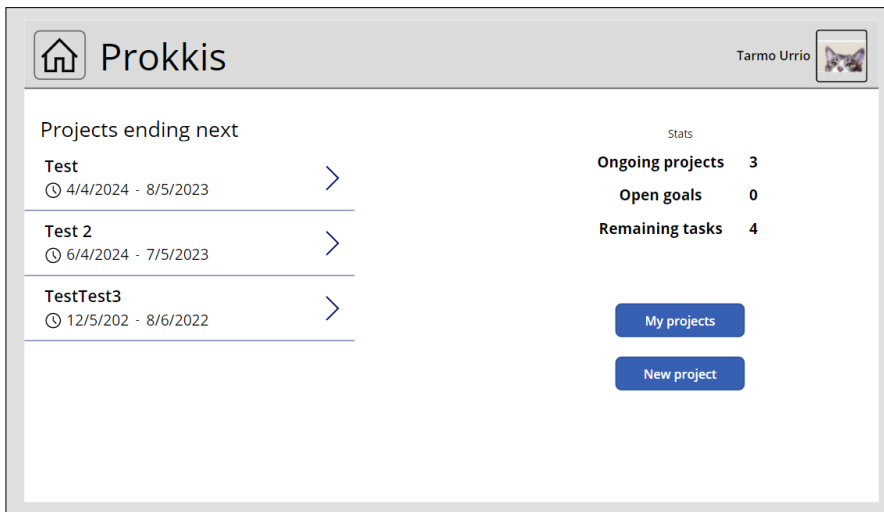


Kuvio 16. Tietokortti - komponentti



Kuvio 17. Galleria – komponentti

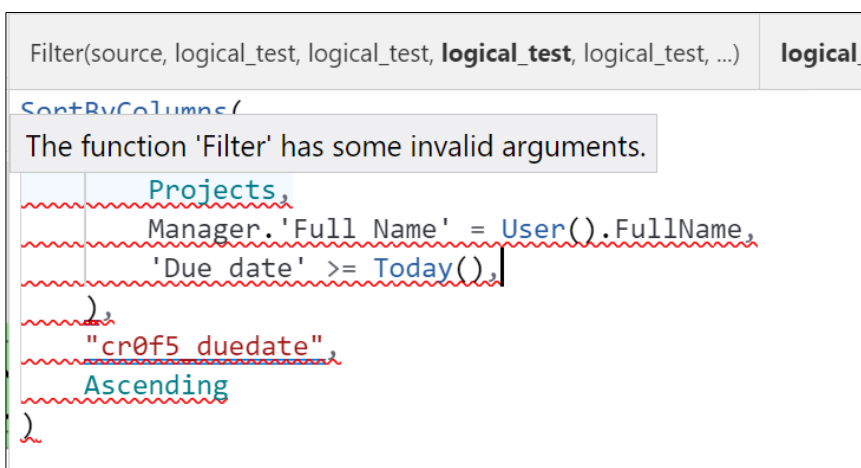
Tietolähteiden liittämisen ja useamman komponentin lisäyksen jälkeen, sovelluksen aloitussivu alkoi hahmottumaan.



Kuvio 18. Alustava etusivun näkymä

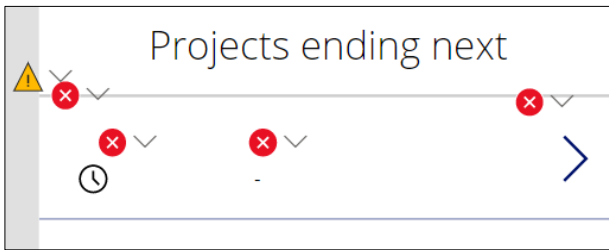
4.8 Virhetilanteet

Jos sovellusalusta havaitsi tiedonhaussa käytetyssä funktiossa virheen, näkyi ilmoitus selvästi muokkausosiossa punaisella värillä alleviivattuna sekä kertomalla missä osassa funktiota virhe sijaitti.



Kuvio 19. Funktion virheilmoitus

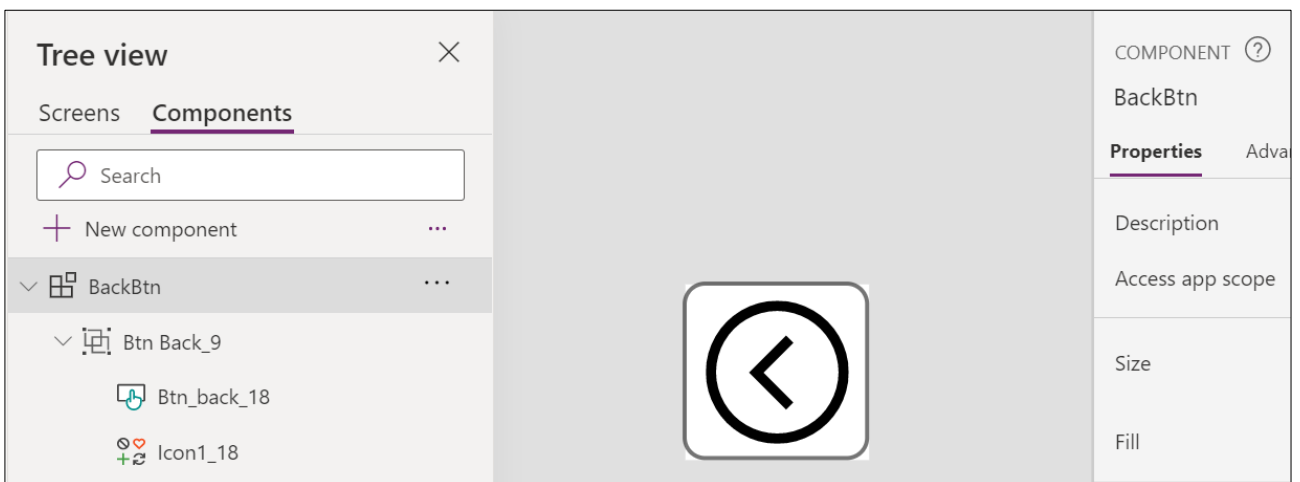
Sovellusnäytön puolella vastaava virhe ilmeni myös selkeästi havainnollistaen virhe- ja huomioikoneilla (ks. kuvio 20), ilmoittaen tarkasti missä osiossa vikoja sijaitsee virheestä johtuen.



Kuvio 20. Virheilmoitukset sovellusnäytymän puolella

4.9 Omien komponenttien määrittely

Viimeistelyjä toteuttaessa haasteelliseksi kohdaksi osoittautuivat eri näkymissä olevat samat komponentit, kuten ylänavigaatio. Vaikka komponentit olivat aina edellisenä työskentelystä näytöstä kopioituja, pienten liikuttelujen ja muokkausten vuoksi painikkeet eivät olleet suoraan täsmälleen samoilla kohdilla, ja näin näytöstä toiseen siirtyessä, näiden liikkumisen pystyi erottamaan. Tähän ratkaisuksi löytyivät omat komponentit (ks. kuvio 21). Komponentit voivat olla mitä tahansa käyttäjän luomia yhdistelmiä eri osista, ja toimivat helposti lisättävinä ja kopioitavina yhdistelminä tilanteissa, jossa jotain toiminnallisuutta tahdotaan toistaa useita kertoja. Näitä käyttäen toteutettiin ylänavigaatio ja sen toiminnallisuudet jokaisessa näytössä.



Kuvio 21. Omien komponenttien luonti

4.10 Sovelluksen kehittäminen

Sovelluksen toteutus eteni näyttö kerrallaan, samalla testaten toimintoja ja tietojen näkymistä aktiivisesti. Uuden näytön alustavan version valmistumisen jälkeen, testattiin valmistuneesta osiosta

siirtymät sekä tietojen siirto eri näkymien välillä. Näin mahdolliset virhekohdat pystyttiin huomaamaan ja korjaamaan tuoreeltaan osiota toteuttaessa. Sovellusta tehdessä huomio kiinnitettiin sovelluksen käytettävyyteen ja suoraviivaisuuteen, jotta sovellusta pystyisi käyttämään ilman ohjeiden lukua tai laajempaa perehtymistä. Tästä johtuen sovellus koitettiin pitää mahdollisimman helppolukuisena, käyttäen tarpeeksi isoja fonttikokoja, hyödyntäen selkeitä valintapainikkeita ja tiedonesitystä sekä käyttäen loogisia etenemispolkuja sovelluksessa.

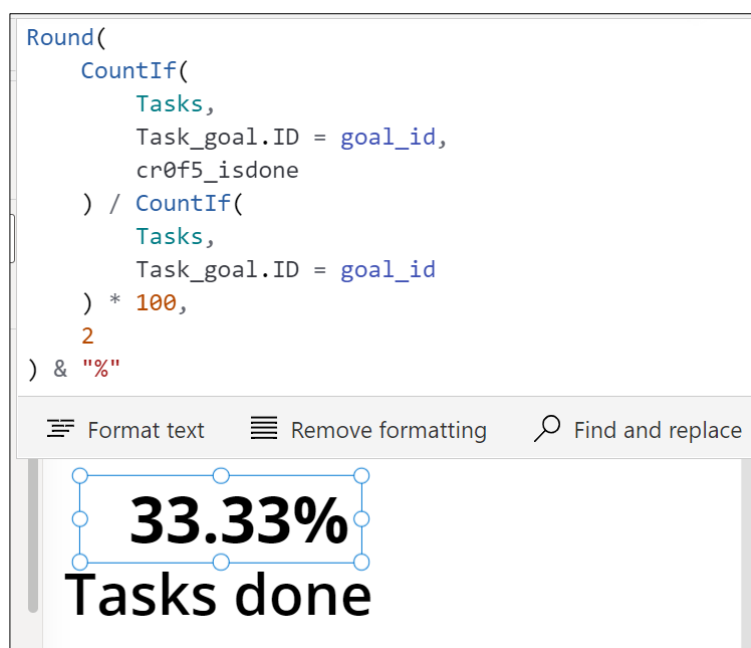
Kohdatuissa ongelmakohdissa, ratkaisu löytyi usein nopeasti käyttäen PowerAppsin käyttäjille yhteisöfoorumia, jossa alustan käyttäjät keskustelevat eri ratkaisuista sekä mahdollisista ongelmista mitä kohtaavat toteutuksia tehdessään. Tutkimusta tehdessä tarkasteltiin tyypillisten haastekohden yhteydessä tutkimukselle asetettuja seurantakysymyksiä, ja näitä verrattiin tilanteissa koettuun ja siihen saakka koettuun kehityksestä saatuun kokemukseen. Ongelmatilanteissa selvitettiin kuinka nopeasti ongelman pystyi ratkaisemaan itse sovellusalustan ohjeistuksilla, tai kuinka helpposti toteutettavan kohdan toteutusvaihtoehdot selvisivät.

4.11 Tutkimuksen tulokset

Ohjelman toteutus oli alkuun hapuilevaa, kehitysalustan ollessa vielä uusi ja tuntematon. Itse toteutukseen kuitenkin pääsi erittäin nopeasti kiinni, testailemalla eri komponentteja ja pohtimalla seuraavaa askelta ja sitä kuinka jokin tyypillinen toiminto toteutetaan kyseisellä kehitysalustalla. Kun tästä sai kiinni, ja kokonaiskuva alkoi hahmottumaan, oli itse toteuttaminen vain ratkaisun tavallinnasta kiinni ja toteuttaessa pystyi huomaamaan, miksi low-code ratkaisut ovat tutkimuksen alkuoletuksien perusteella huomattavasti nopeampia tuottaa verrattuna tavalliseen sovelluskehitykseen. Alkuvaiheen jälkeen nämä oletamat alkoivat muokkautua ilmiselviksi tosiasioiksi, jädessäni usein pohtimaan itse toteuttamisen yksinkertaisuutta ja intuitiivisuutta.

Isoimpia haasteita sovellusta toteuttaessa olivat tietokannasta tulevan tiedon näyttäminen, projekteille valittavien haettavien käyttäjätunnusten rajaaminen sekä eri lomakkeiden valintojen määrittelyt. Itse tietokannan taulujen sekä näiden välisten suhteiden luonti oli vaivaton kokemus, ja mikäli käyttäjällä on yhtään aiempaa kokemusta tietokannan luonnista, voi tämän toteuttaa ilman tarpeeksi dokumentaatioon perehtymistä. Täältä saatavaa tietoa suodattaessa ongelmaksi nousi usein vähäinen virhekuvauksen määrä. Kehitysalustan virheilmoitus kertoi tiedonhaun epäonnistumisesta, esimerkiksi että jotain tietoa ei pysty suodattamaan tai vain alleviivasi kohdan taustakoodista

kertoen, että kyseinen osio funktiosta on väärin, mutta ei pystynyt selvittämään tarkemmin mikä itse kohdassa on virheellistä. Näihin kohtiin kuitenkin aina lopulta löytyi ratkaisu, ja ongelmaksi ilmeni usein useamman funktion ketjutuksesta johtuva sekaannus. Koska PowerApps antaa aina komponentin toiminnolle yhden syöttökentän, johtaa se tilanteeseen, jossa eri funktioita ketjuteaan useampi sisäkkäin esimerkiksi haettavaa tietoa rajatessa. Kun vaikka halutaan hakea tietokannasta tulokset, joissa täyttyy useampi ehto ja joka tarkistaa useamman eri osa-alueen kyseisistä tiedoista, kasvaa tähän hakukenttään tuotettava funktio monen rivin mittaiseksi (ks. kuvio 22).



Kuvio 22. Useamman funktion yhdistelmä

Tästä huolimatta, funktioiden luonti on hyvin avustettua, ja kehitysalusta ohjeistaa jokaisessa funktion luonnin kohdassa kyseiseen parametriin tarvittavan tiedon määritelmästä kehitysympäristöille tyypilliseen tapaan. Tämä tukee uusien asioiden nopeasti oppimista, toistuvan opastuksen kautta, ja yleisimmin käytetyt funktiot sisäistyivätkin suhteellisen nopeasti.

Toteutuksen edetessä toistuvana ongelmana esiintyi alustan taustalla toimivan koodin näkemisen puute. Koska koodiin ei ollut suoraa näkymää, kaikki tieto tehdyistä funktioista ja valituista valinnoista jäi tietoiikkunoiden taakse. Tämä hidasti joissain kohdissa etenemistä turhaksi toistoksi koettuna toimintana, sillä esimerkiksi asetetun muuttujan tai värin arvon mieleen palautus oli useamman painalluksen ja näytöllä välillä siirtymisen takana, jonka jälkeen joutui palaamaan takaisin työn alla olevaan näyttöön samalla tavalla.

Low-code kehitysalustat ovat pitkälti rajattuja ympäristöjä, mutta silti tutkimuksen aikana ei nous-
sut ilmi isommin kokemusta rajatuksi tulemisesta, alustan rajoitteiden tai mahdollisuuksien osalta.
Haettavien tietojen ja toiminnallisuuksien näyttämiseen oli saatavilla muutama alustava vaihto-
ehto tai valmiista vaihtoehdosta pystyi luomaan aivan omannäköisensä. Sovelluksen ulkoasua poh-
tiessa ja toteuttaessa haasteeksi ilmenivät alustan rajatut ulkoasupuolen toteutuksen rajoitteet.
Alustalla pääasialliset vaihtoehdot ulkoasun muokkaukseen olivat fontin, taustan sekä reunojen
väriytykset. Lisäksi esimerkiksi ainoana vaihtoehtona muotojen tai ulkoasun tyyllittelyyn komponent-
teilla olivat suorakulmion muotoiset tyhjät graafiset elementit, joita ei pystynyt kiertämään. Käy-
tössä oli lisäksi myös Microsoftin tuottama ikonikirjasto, mutta nämäkin olivat pääasiallisesti tar-
koitettuna visuaalisina elementteinä eri komponenttien tietojen yhteyteen, eivätkä niinkään itse
ulkoasun toteutukseen.

Graafisen puolen vähyydestä huolimatta, itse kehityksen edut tulivat hyvin esille kehityksen ede-
tessä. Ulkoasun ollessa pohdittuna, ja alkutuntumaan päässeenä, esimerkiksi uuden näytön toteu-
tus oli erittäin virtaviivainen ja miellyttävä kokemus. Tiedon yhdistämiset ja haut tapahtuivat auto-
maattisesti, kun vain saapuvan tiedon rajasi tarpeen mukaisesti sekä toiminnallisuudet toimivat
suoraan uuden komponentin lisäyksen jälkeen ilman useampia toistoja toiminnallisuuden varmis-
tamiseksi. Jatkuva mahdollisuus sovelluksen testaamiseen pelkällä alt-näppäimen pohjassa pitämi-
sellä kiihdytti entisestään jo nopeaa kehityksen tahtia sekä antoi mahdollisuuden keskittyä itse so-
velluksen tuottamiseen ilman, että toistuvasti joutui siirtymään kehitysympäristön ja
sovellusnäkymien välillä kuten esimerkiksi tyyppillisessä verkkosovelluskehityksessä uutta kompo-
nenttia testatessa.

Isoimmat vaikeudet tutkimusta toteuttaessa eivät kuitenkaan kohdistuneet itse Power Appsin toi-
mintoihin, vaan pikemminkin tietokantaratkaisuksi valikoituneen Dataversen taulujen kompleksien
tietorakenteiden hallintaan ja yhdistämiseen keskenään. Esimerkiksi tietokannan vakiotaulu User,
antaa tiedot kaikista kyseisen organisaation alla olevista käyttäjistä, keillä on tarvittavat lisenssioi-
keudet käyttää alustan ratkaisuja. Yhdeksi ongelmaksi muodostui useamman käyttäjän samaan ai-
kaan lisääminen projektille, sillä useampaa käyttäjä-tyyppistä tietuetta ei saanut lisättyä yhdelle
taulun tietoriville. Tämä haaste kierrettiin yksinkertaisesti antamalla käyttäjälle mahdollisuudeksi
vain lisätä projektille vain yhden käyttäjän kerrallaan, ja käyttämällä välitaulua, joka yhdistää käyt-

täjän projektiin, sillä kyseiseen haasteeseen paremman ratkaisun löytäminen olisi ohjannut huomiota pois itse kohteena olleen kehitysalustan tutkimuksen tavoitteesta. Tämä toi myös osaltaan huomion tutkimuksen alkuolettamuksiin itse kehittämisen oletetuista hyvistä osapuolista. Isompien haasteiden puuttuessa itse kehityksestä, voidaan todeta tutkimuksen perusteella, että kehitystavan tuomat edut kehityksen nopeudessa ja yksinkertaisuudessa ovat hyvinkin paikkaansa pitäviä. Artikkelit, mielipiteet ja julkinen asenne low-code ratkaisujen puolesta, kohottavat tätä hyvinkin korkealle sovellusratkaisujen vaihtoehtoisissa, ja tuntuvat pitävän paikkansa. Itse kehitys on erittäin suoraviivaista ja mikäli toteuttaja omaa yhtään aiempaa ohjelmistokehityksen taustaa, on toteutuksessa ainoana haasteena ympäristöön ja sen toimintatyyliin tutustuminen.

Vaikkakin sovellus toteuttaa perustoimintoja, eikä ole liitännöiltään tai logiikaltaan kompleksi, tuotti sovelluksen tuottaminen käytännössä paljon tietoa tähän tarvittavasta prosessista, eri vaiheista ja parhaista käytännöistä, joita tulisi noudattaa uutta sovellusta luodessa. Microsoft on myös luonut PowerApps-kehitykselle ohjekirjan, joka käsittelee parhaita käytäntöjä ja standardeja, kuten eri komponenttien tai näkymien nimeämiskäytänteitä sovellusta luodessa. Kyseistä ohjekirjaa ei kuitenkaan ole tarkoitettu luettavaksi välttämättä ennen sovelluskehityksen aloitusta, vaan pikemminkin kehittämisen ensikosketuksen jälkeen muovaamaan tulevien sovellusten tuottamista ohjeistamalla parhaimmaksi todettuja käytänteitä. (Baginski & Dunn 2018.)

Sovelluksen lopputuloksessa päästiin hyvin tarkoituksenmukaisuuteensa ja sovellukselle määriteltiin vaatimuksiin. Sovellus toimi suunnitellun tarkoituksen mukaisesti ja toimii tarkoituksensa mukaisesti yksinkertaisena projektin ja sen tavoitteiden ja tehtävien hallinnan välineenä. Tuotetun ratkaisun toiminnallisuuden tavoitteena oli toimia enemmän testikohteina, kuin tavoitteina, joilla mahdollistettiin tutustuminen alustaan esimerkkiratkaisun pohjalta. Nämä tuottivat hyvin kokemuksia itse kehityksestä ja sen eri osa-alueista, tarjoten kattavaa näkemystä kehitysalustan eri mahdollisuuksiin ja ominaisuuksiin. Tuotannon osa-alueet tarvittavan tiedon, toiminnallisuuden ja ulkoasun yhdistämisestä toimivat saumattomasti yhteen ja sovelluksen tuottaminen oli erittäin miellyttävä kokemus, joka kannusti jatkamaan aiheeseen perehtymistä jättämällä positiivisen mielikuvan sovelluskehityksestä low-code alustoilla.

Tutkimuksesta saadut kokemukset tukevat vahvasti muita aiheesta tehtyjä tutkimustuloksia, ollessa erittäin myönteisiä tutkimukselle asetettujen alkuolettamien kanssa. Kehityksen nopeus,

millä sovelluksen sai ensimmäiseen toimivaan versioon sekä mahdollisuus jatkuvaan testaukseen sovelluksen eri toimintojen valmistuessa, tukevat hyvin väitteitä kehitysmallin tuomista eduista sekä sovelluksen tuottajalle että asiakkaalle. Tulosten perusteella pystytään oletttamaan melko varmasti, että low-code kehitys tulee yleistymään huomattavasti seuraavien vuosien aikana, etenkin yritysten sisäisiin tarpeisiin tuotettujen sovellusten tarpeen kasvaessa yritysten digitalisaation ohessa.

Vaikkakin itse kehitystyö oli kokeman perustella suhteellisen helppoa ja yksinkertaista, tulee ottaa huomioon, että kokemukseen vaikuttavat paljon osaltaan aiempi kokemus ja osaaminen ohjelmistojen kehityksestä. Kansalaiskehittäjälle, kenellä ei välttämättä ole kokemusta sovelluksen luomisesta, tai perusteita yleisimmin käytettyjen funktioiden toimesta, voi kehittämiseen kiinni pääseminen vaatia varmasti enemmän perehtymistä. Alusta itsessään tarjoaa kuitenkin tavalliseen ohjelmistokehitykseen verrattuna erittäin helpon reitin sovelluksen kehittämiseen, yksinkertaistamalla monia yleisiä sovellusten toiminnan osa-alueita, kuten tarvittavan tiedon käsittelyä tai eri osioiden toiminnallisuuksia. Kuitenkin kattavan dokumentaation ja aktiivisen yhteisön omaavana, PowerApps on hyvä kohde low-code kehittämisen aloitukseen, tarjoten hyvät resurssit aiheen opetteluun ja ongelmatilanteiden ratkaisuun.

5 Pohdinta

Tämän työn alussa en tuntenut kovinkaan paljoa low-code kehitystä. Perustin tietoni kuulemaani ja näkemääni opintojen sekä työelämän puolelta. Ilman aiempaa kokemusta, pohjasin oman ensivaikutelmani kehityksestä hyvin pitkälti kuulupuheeseen ja näkemääni. Ottaen huomioon kehitystavan jatkuvan suosion kasvun, on aiheesta kuulematta oleminen ollut miltei mahdotonta ohjelmistoalaa opiskelevana. Usein aihealueen suosion kasvaessa vakiona pitkällä aikavälillä, on sen suosio myös ansaittua aiheen ominaisuuksien vuoksi. Tämä herättääkin usein kuulijassa kysymyksen, tarjoaako kohde sen mitä lupaa. Itselläni ensimmäisinä kysymyksinä nousivat pinnalle, mitkä ovat low-code kehityksellä tuotetun tuotteen mahdollisuudet sekä kuinka taipuisaa itse kehitys on.

Oletusarvoina käytetyt yleiset mielipiteet olivat hyvin positiivissävytteisiä, ja lupaukset nopeasta kehityksestä ja kattavista mahdollisuuksista, olivat omansa kasvattamaan mielenkiintoa kyseistä aihealuetta kohtaan. Jos jokin on niin hyvä, miksei isoa osaa sitten toteuteta tällä tavalla? Aihealue herätti myös osallaan skeptisyyttä mielessäni, sillä jokin tavassa tuottaa ohjelma niin, että hallinta

ei ole täysimittaisesti itsellään, tuottaa kokemuksen oikotiestä lopputulokseen ja epävarmuudesta tuotteen lopullista laatua kohden. Osittain tämä kokemus voi pohjautua uhkakuihin oman osaamisen tarvittavasta uudelleensuuntaamisesta kehitystrendin jatkuessa pitempään, tai siksi että olen opiskeluni ajan omaksunut tavallisen ohjelmistokehitystavan prosesseja. Syynä aihealueen tuottamiin mielikuihin olikin luultavimmin pelko oman koetun osaamisen puute ja totuttautuminen uuteen.

Tutkimuksen edetessä, mielikuvani aiheesta muuttui huomattavasti, sillä itse tuotteen toteutus on käytännössä tavallista ohjelmistokehitystä, mutta vain helpotettuna ja nopeutettuna huomattavasti. Isoimpien ongelmieni keskittyessä ainoastaan tiedon näyttämiseen, huomasin usein pohtivani kuinka vaivatonta itse idean toteuttaminen onkaan alustan avulla. Graafinen käyttöliittymä vastasi huomattavasti ulkoasuun suunnitteluun tarkoitettujen työkalujen toimintaa tai esimerkiksi C# - ohjelmointikielen käyttöliittymällisten ohjelmien kehitystä, ja näistä hieman kokemusta omaavana, oli vastaavan kaltainen graafinen käyttöliittymä hyvin suoraviivaista omaksua.

Low-code kehitys on ollut yleisessä tiedossa jo pitemmän aikaa, ja vaikkakin tästä on ennustettu useamman vuoden ajan ohjelmistokehityksen mullistajaa, tuntuu että varsinainen läpimurto on tapahtunut viimeisen kahden vuoden aikana, luultavimmin osittain globaalin pandemian seurauksena. Ison osan työnteosta siirtyessä verkkoon, nousi samalla esiin työnantajien tarve saada enemmän liiketoimintaa tukevia ohjelmistoja, sillä työntekijöiden ollessa verkossa, tuli myös työssä tarvittavien välineiden olla työskentelyä tukevia uusia sovellusratkaisuja. Näihin tarpeisiin low-code kehitys vastaa täydellisesti. Kun tarvittavan ohjelmiston kehitykseen ei tarvita välttämättä isoa toteutusta ja hinta sekä laatu kohtaavat tarpeet, voi ratkaisun toteutus low-codella olla miltei itsentäänselvyyys yritykselle. Omien kokemuksieni pohjalta nämä ovat erittäin todennäköisiä myyntivaltteja, ja kynnys ohjelman tilaukselle tai jopa itse tuottamiselle voi madaltua huomattavasti näiden etujen myötä. Tästä johtuen kehitystavasta hyötyvät sekä toimittaja että asiakas, ratkaisun tuodessa molemmille loistavia etuja.

Low-code kehitys tarjoaa juuri sitä mihin se on suunniteltu. Oma tietämykseni kehityksestä oli alkeellisella tasolla ennen tutkimuksen aloitusta, ja koen kehittyneeni nopeasti tutkimuksen aikana kyseisen alustan kehitystyössä. Kehitys on erittäin sukkela ja useasti jotain pohtiessani osuin ratkaisuun yllättävänkin nopeasti vain kokeilemalla, ilman suurempaa tarvittavaa selvitystyötä. Miltei

kaikki eri osioiden vaihtoehdot on suoraviivaistettu ja yksinkertaistettu, mutta tarjoavat silti mahdollisuuden tarkempaan asetusten määrittelyyn. Tällä mahdollistetaan esimerkiksi kehittäjälle laajempi työkalupakki, mutta tarjotaan myös samalla vähemmän osaavalle tekijälle sovelluksen toteuttamisen edellytykset. Vaikkakin low-code alustat ovat myös kohdennettu bisnestaustaisille käyttäjille, koin että ilman kokemusta ohjelmoinnista, tai samankaltaisesta kehitystyöstä, olisi projekti voinut ollut huomattavasti raskaampi toteuttaa, sillä ohjelmointiosaaminen antaa tehokkaan oikotien kehitykseen kyseisillä alustoilla. Tietämällä ennalta jo yleisten funktioiden ja tyyppillisten ohjelmien toimintojen toiminnan perusteista, opeteltavaksi alustoilla kehittämisestä jäävät alustan omat sekä käytetyn taustakoodin muokattavan osion kielen käytänteet.

Isoimmat kohdatut haasteet liittyivät datan hakuun, suodattamiseen ja näyttämiseen. Tämä tiivistää miltei itse low-coden pohjan, jonka saralla se toteuttaakin tarkoitustaan loistavasti. Yritysten käytössä oleva datan määrä kasvaa jatkuvasti, ja tästä seuraakin usein tarve tai idea kaiken tämän tiedon hyödyntämisestä. Esimerkiksi projekteista, virhetilanteista tai tuotannosta saatava data voi antaa kattavaa yleiskuvaa yrityksen tilasta ja tehostaa lopulta työntekijän työskentelyä, kun useasta lähteestä yhdistetään tietoa yhteen ja samaan kohteeseen. Tiedonhallinnan ja analytiikan haasteisiin keskitytään pääosin data-analytiikan puolella, ja näihin tarkoituksiin onkin kohdennettu omat sovelluksensa, mutta tyyppisiin yrityksen tarpeisiin kuten esimerkiksi tuotettujen tarjousten kirjaukseen tai vaikkapa useamman pääkäyttäjän hallintapaneelin korvaamiseen yhdellä ratkaisulla, on low-code omiaan tuottamaan toimivia ratkaisuja.

PowerApps alustana on loistava ratkaisu etenkin Microsoftin ekosysteemiin kohdentuneille yrityksille. Helppo yhdistäminen tiedon saamiseksi esimerkiksi SharePoint-listasta, kasvattaa itsessään myös Microsoftin omilla ratkaisuilla tuotettavien sovellutusten kyvykkyyttä tarjota monipuolisia ja ketteriä ratkaisuja eri tarpeisiin. Pääasiallisesti näkisin vastaavien ratkaisujen kohteena pienet ja keskisuuret bisnessovellukset, joilla nopeutetaan esimerkiksi työnkulkua tai käsitellään tietoja useasta lähteestä. Low-code toteutus mahdollistaa myös tuotteen skaalautuvuuden ja lisäominaisuuksien tuottamisen ilman suurempia haasteita, ja takaa näin myös hyvät lähtökohdat tuotteen jatkokehitykselle.

Tutkimuksen tulosten perustuessa yhden henkilön kokemuksiin, yhden kehitysalustan työskentelystä, tulee näiden tulosten validiudesta olla erittäin kriittinen. Kuitenkin verrattaessa konkreettisia

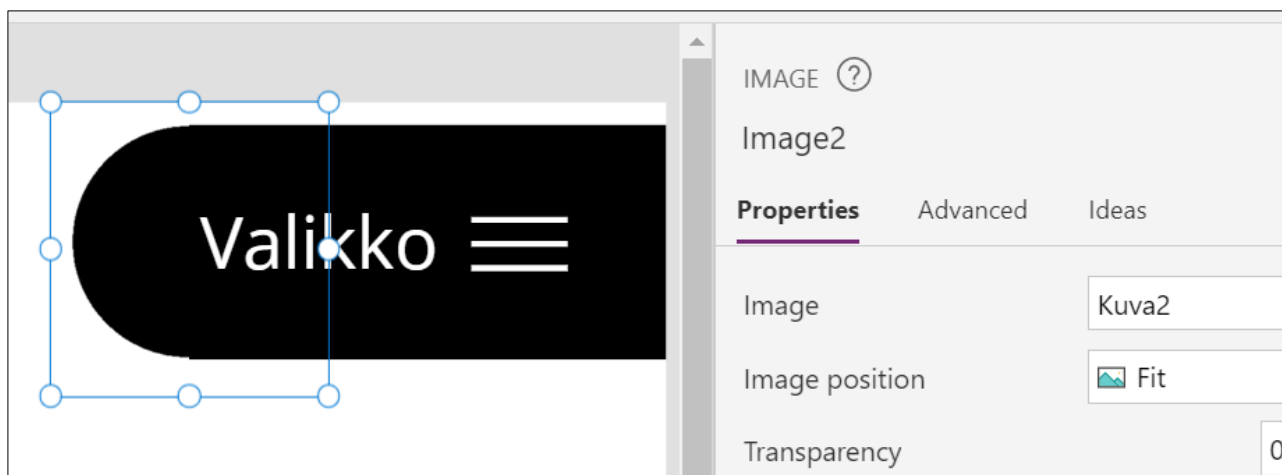
etuja kehitystavasta, pystytään joitakin osa-alueita todentamaan kuitenkin kohtalaisella varmuudella. Voidaan todeta tuotteen kehitykseen käytetyn ajan vähentyvän huomattavasti low-code alustan toteutuksessa useastakin syystä. Toteutuksen tapahtuessa verkossa, itse kehitysympäristölle ei ole pystytystarvetta sekä visuaalisen puolen ollessa jatkuvasti läsnä käyttöliittymä- ja näköympäristöön kehityksessä tuotteen sen hetkinen tila on jatkuvasti nähtävillä. Tämä mahdollistaa korjaukset esimerkiksi haetun datan näyttötavan muokkaamisessa, ja itse datan haun tapahtuessa välittömästi, varmistetaan myös koodin toimivuus sillä hetkellä, kun se on kirjoitettu.

Esimerkiksi yhdistäessä tietoja näyttävän osa haluttuun tietokantatauluun, haetun tiedon suodattamista tehdessä voi hakulausekkeen toimivuudesta varmistua heti, sillä virhetilanteessa kehitysalusta antaa välittömän palautteen kirjoitetun funktion toimimattomuudesta. Myös esimerkiksi juuri lisätyn painikkeen toiminnon testauksen voi suorittaa välittömästi painikkeen toiminnallisuuden määrittelyn jälkeen. PowerApps alustaa käyttäessä luotavaa ohjelmaa voi testata jopa vain pitämällä pelkästään Alt-näppäintä pohjassa. Sillä ohjelman toimintoja ylläpitävän taustakoodin ollessa jatkuvasti ajan tasalla, mahdollistetaan myös ohjelman jatkuva testaus- ja toimintavalmius. Näin ohjelman kehittäjä voi varmistaa juuri luomansa osion toiminnan heti, tuottamatta kehitystyöhön isompaa keskeytystä.

Jäin tutkimustyötä toteuttaessa kaipaamaan lisää visuaalisen puolen elementtejä sovellukseen. Vaikkakin isossa osassa komponentteja, valintamahdollisuudet eri visuaalisille osioille ja muokkauksille olivat hyviä, jäi kehitystä toteuttaessa tunne, että ulkoasu on hyvin pelkistetyn näköinen, sovelluksen käyttäessä samankaltaisia komponentteja ja tiedon näyttämisen tyyliä. Iso mahdollisuus on, että tarkemmat visuaaliset muokkauksen mahdollisuudet jäivät itseltäni myös huomaamatta, sillä usein erilaisten ominaisuuksien muokkauskohteita oli monia. Huomioni keskittyikin usein vain komponentin täyteväriin, ja käyttäjän eri toimintojen vaikutuksista tähän väriin. Ohjelman ulkoasu vaikuttaa osaltaan tuotteen käyttökokemukseen, ja vaikka ohjelma suorittaisi tarvittavat toiminnallisuudet, tuo hyvin suunniteltu ja soviteltu sovelluksen ulkoasu lisäarvoa sovelluksen käyttäjälle.

Myöhemmällä pohdinnalla ja testauksella, löysin tähän ratkaisuna itse tuotetut kuvatiedostot, joiden avulla tarvittavia erimuotoisia elementtejä pystyi lisäämään kuva-komponentteja hyödyntäen.

Esimerkiksi ympyränmuotoisen visuaalisen elementin sai luomalla muodon kuvankäsittelyohjelmalla ja poistamalla kuvasta taustan värityksen. Yhdistämällä tämän komponentin vakiona olevaan suorakulmiokomponenttiin (ks. kuvio 23), saa tuotettavaan ohjelmaan lisää mahdollisuuksia käyttöliittymän suunnitteluun ja toteutukseen.



Kuvio 23. Kuva-komponentin hyödyntäminen visuaalisesti

Jatkokehitysideana olisi mielenkiintoista tutkia eri kehitysalustoja ja näiden ominaisuuksia. Erilaisia kehitysalustoja on valtava määrä, ja kaikilla näistä on omat toimintatyylinsä ja eroavaisuutensa toisiinsa nähden. Esimerkiksi alustojen erilaiset liitettävyydet mahdollistavat miltei rajattoman määrän vaihtoehtoja toteutettavalle sovellukselle ja sen toiminnoille. Myös tarkempi tarkastelu esimerkiksi käyttäjäkokemuksesta ja low-codella tuotettujen tuotteiden verrattavuudesta tavallisen ohjelmistokehityksen tuotteisiin loppukäyttäjien näkökulmasta voisi olla erittäin mielenkiintoista tutkittavaa, juuri kehitystapojen lopputuotteiden eroavaisuuksien osalta.

Vaikkakin saadut tulokset tukivat täysin tutkimuksen alussa asetettua oletusasetelmaa, on lopullinen kuva itse kehityksestä aina kehittäjäkohtaista. Yksilölliset mielipiteet, asenteet sekä aiemmat kokemukset vaikuttavat kokemukseen kehityksestä ja kuten eroavaisuudet ja mieltymykset eri ohjelmointikielien välillä, kohtaa low-code varmastikin samankaltaista vastakkainasettelua myös omalta osaltaan. Varmaksi voidaan kuitenkin sanoa, että low-code kehitys tulee yleistymään tulevien vuosien aikana vielä suuremmin, ja kasvamaan osaksi yleisimmin käytettyjä sovellusratkaisuja tavallisen kehitystyylin rinnalle, tuoden tarvittavia uusia osaajia sekä uutta kapasiteettiä sovellusten tuotantoon.

Lähteet

Bock, A.C. & Frank, U. 2021. Low-Code Platform. Viitattu 26.4.2022.

<https://doi.org/10.1007/s12599-021-00726-8>.

Citizen Development - The Handbook for Creators and Changemakers. 2022. Project Management Institutien tuottama ohjekirja. Viitattu 26.4.2022. <https://app.knovel.com/hot-link/toc/id:kpCDTHCC01/citizen-development-handbook/citizen-development-handbook>.

Desai, D. 2016. Announcing Public Preview for PowerApps. Viitattu 27.4.2022. <https://powerapps.microsoft.com/en-us/blog/powerapps-public-preview>.

Forsyth, A. 2021. Low-Code and No-Code: What's the Difference and When to Use What?. Viitattu 25.4.2022. <https://www.outsystems.com/blog/posts/low-code-vs-no-code>.

Garcia, K. 2021. What is low code? A comprehensive guide. Viitattu 26.4.2022. <https://re-tool.com/blog/what-is-low-code>.

Lasar, M. 2019. 30-plus years of HyperCard, the missing link to the web. Viitattu 27.05.2022. <https://arstechnica.com/gadgets/2019/05/25-years-of-hypercard-the-missing-link-to-the-web>.

Lindroos, K. & Björklund, S. 2021. Onko parjatulle Aster-potilastietojärjestelmälle vaihtoehtoja? Kaksi kotimaista yritystä uskoo pystyvän samaan edullisemmin: "Toteutettavissa huomattavasti halvemmalla". Viitattu 23.4.2022. <https://yle.fi/uutiset/3-12107472>.

Lindström, S. 2021. Sovelluskehitys ei aina vaadi koodaustaitoja – Kansalaiskehittäjän valttikortti on liiketoiminnan tarpeiden omakohtainen ymmärrys. Viitattu 28.4.2022. <https://www.itewiki.fi/blog/2021/06/sovelluskehitys-ei-aina-vaadi-koodaustaitoja-kansalais-kehittajan-valttikortti>.

Microsoft Power Fxin yleiskatsaus. 2022. Artikkelin Microsoftin dokumentaationsivuilla. Viitattu 2.5.2022. <https://docs.microsoft.com/fi-fi/power-platform/power-fx/overview>.

Person, D. 2022. How Low-Code And No-Code Tools Can Help Stave Off Shadow IT And The Great Resignation. Viitattu 26.4.2022. <https://www.forbes.com/sites/forbestech-council/2022/02/09/how-low-code-and-no-code-tools-can-help-stave-off-shadow-it-and-the-great-resignation>.

Phillips, J. 2016. Microsoft PowerApps and Flow are generally available starting tomorrow. Viitattu 27.4.2022. <https://blogs.microsoft.com/blog/2016/10/31/microsoft-powerapps-flow-generally-available-starting-tomorrow>.

Ramel, D. 2021. Microsoft Power Apps Ranks High in Low-Code Research Report. Viitattu 26.4.2022. <https://visualstudiomagazine.com/articles/2021/05/13/forrester-lowcode.aspx>.

Schoettle, A. 2021. No coding skills? Increasingly, that's no problem. Indianapolis Business Journal, vol. 42. Viitattu 28.4.2022. <https://www-proquest-com.ezproxy.jamk.fi:2443/trade-journals/no-coding-skills-increasingly-thats-problem/docview/2506760598/se-2>.

Stamford, C. 2021. Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences. Viitattu 28.4.2022. <https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences>.

Staples, B. 2015. Introducing Microsoft PowerApps. Viitattu 27.4.2022. <https://blogs.microsoft.com/blog/2015/11/30/introducing-microsoft-powerapps>.

The Low-Code Development Guide. N.d. Esittely low-code kehityksestä Outsystems sivuilla. Viitattu 25.4.2022. <https://www.outsystems.com/guide/low-code>.

The state of low-code/no-code. 2021. Raportti Creation sivuilla. Viitattu 26.4.2022. <https://www.creatio.com/page/sites/default/files/2021-05/Report-May.pdf>.

Tozzi, C. 2021. Viitattu 4.5.2022 <https://www.itprotoday.com/no-codelow-code/evolution-low-codeno-code-development>.

Vailshery, L. 2022. How much faster is low-code development comparing to traditional development?. 2022. Viitattu 22.4.2022. <https://www-statista-com.ezproxy.jamk.fi:2443/statistics/1254662/low-code-development-speed-compared-traditional-it>.

Vailshery, L. 2021. Low-code development platform market revenue worldwide from 2018 to 2025 (in billion U.S. dollars). Viitattu 22.4.2022. <https://www-statista-com.ezproxy.jamk.fi:2443/statistics/1226179/low-code-development-platform-market-revenue-global>.

What is Microsoft Dataverse. 2022. Artikkelin Microsoftin dokumentaation sivuilla. Viitattu 4.5.2022. <https://docs.microsoft.com/en-us/power-apps/maker/data-platform/data-platform-intro>.

Liitteet

Liite 1. Esimerkkiohjelman vaatimusmäärittely

Vaatimusmäärittely – Projektinhallintatyökalu ”Prokkis”

Lyhyesti

Yrityksessä on tunnistettu haasteena projektien hallinnan vaikeus. Projektien jäsenillä ei ole usein tarkkaa käsitystä projektin tilasta projektin edetessä ja tästä johtuen omien työtehtävien aikataulutus voi olla haasteellista, tai näitä joudutaan käymään läpi projektin vetäjän kanssa. Tähän ratkaisuun toteutetaan projektinhallintasovellus helpottamaan projektinhallintaa, seuranta ja etenemistä. Työkalu toteutetaan käyttäen Microsoftin low-code alusta PowerAppsiä. Testausvaiheessa sovelluksen saavat käyttöönsä kaikki uusiin projekteihin osallistuvat henkilöt. Markkinoilla on saatavilla useita eri projektinhallinnan työkaluja, mutta yrityksen pienestä koosta johtuen päädyttiin tuottamaan projektinhallinnan työkalu sisäisesti, valmiina olevalla kehitysalustalla osana aiheeseen perehtymistä ja palveluvalikoiman lisäystä.

Käytettävät teknologiat

- PowerApps
- Dataverse
- Azure Active Directory

Käyttäjärühmät

1. Projektin vetäjä
 - Hallinnoi projektin tilaa
 - Määrittää projektille tehtäviä
 - Määrittää tehtäville henkilöitä
 - Määrittää projektille tavoitteita
2. Projektin jäsen
 - Hallinnoi hänelle määriteltyjä tehtäviä
 - Merkitsee tehtävien tilaa

Käyttäjätarinat

USER001 -Projektin jäsenenä haluan nähdä projektieni tilan yhdestä näkymästä
 USER002 -Projektin jäsenenä haluan nähdä minulle määritetyt tehtävät
 USER003 -Projektin jäsenenä haluan nähdä tehtäviin liittyvät tavoitteet
 USER004 -Projektin vetäjänä haluan hallinnoida projektejani
 USER005 -Projektin vetäjänä haluan määrittää tehtäviä käyttäjille
 USER006 -Projektin vetäjänä haluan määrittää projektille tavoitteita

Ominaisuudet

- FEAT001 - Projektin hallinnan työkalut (lisäys / muokkaus / poisto / tila)
- FEAT002 - Tehtävän hallinnan työkalut (lisäys / muokkaus / poisto / tila)
- FEAT003 - Tavoitteiden hallinnan työkalut (lisäys / muokkaus / poisto / tila)
- FEAT004 - Projektin jäsenten hallinta (lisäys / poisto)

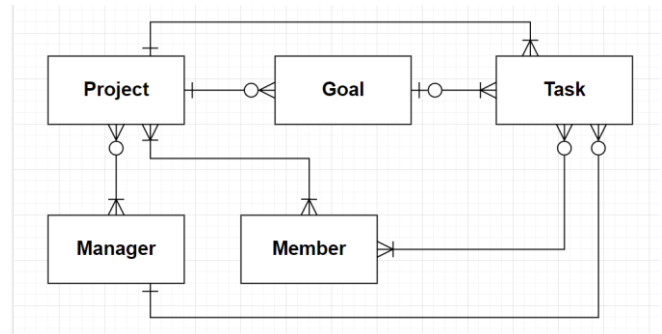
Näkymät

- VIEW001 - Projektien listaus
- VIEW002 - Projektin tila
 - ❖ Tehtävät
 - ❖ Tavoitteet
- VIEW003 - Uusi projekti
- VIEW004 - Uusi tehtävä
- VIEW005 - Uusi tavoite
- VIEW006 - Omat projektit
- VIEW007 - Omat tehtävät

Tietokanta

Luodaan käyttäen Dataverseä kuvauksen mukaisesti.


- Project
 - ID - int
 - Name - string
 - Start date - date
 - Due date - date
 - [Project->Tasks] - Task
 - [Project->Goals] - Goal
 - [Members] - Member
 - [Managers] - Manager
- Goal
 - ID - int
 - Name - string
 - Description - text
 - Due date - date
 - [Tasks] - Task
- Task
 - ID - int
 - Name - string
 - Description - text
 - Assigned to - Member/Manager
 - Due date - date
 - [Goals] - Goal
- Manager
 - Data from Active Directory
- Member
 - Data from Active Directory





Liite 2. Valmiin ohjelman ulkoasu



Aloitussivu

Prokkis

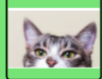
Tarmo Urrio

Projects ending next	Tasks due next	My statistics
Client Project 101 🕒 1/6/2022 - 1/8/2022 >	Remove popup 🕒 6/6/2024 >	Ongoing projects 1 Open goals 1 Open tasks 1
		<div> My projects</div> <div> New project</div>

Uuden projektin luonti



Prokkis

Tarmo Urrio

New Project

* Name

* Description


Manager

Tarmo Urrio

▼

* Start date

6/1/2022



00

▼


:

00

▼

Due date

8/1/2022



00

▼

:



00

▼

Save

Clear

Minun projektini


Tarmo Urrio


My projects




Company
Branding
Here

1/6/2022 - 1/8/2022
Client Project 101

Project for managing customer related tasks

ID: 1006

Uusi tavoite



Prokkis
Tarmo Urrio


New Goal

* Name

Fix main problems

Description

Get all main tickets done

Goal tasks

Goal manager

Tarmo Urrio

Due date

6/30/2022

00




:

00

Save

Reset

Uusi tehtävä



Prokkis
Tarmo Urrio


New Task

* Name

Description

* Due date




 :

Assigned to

Task Goal

Is Done ☐ No



Yksittäisen projektin näkymä



Prokkis
Tarmo Urrio



Project
Name
 Client Project 101
Description
 Project for managing customer related tasks
Start date
 1/6/2022
Due date
 1/8/2022
Members
 Jane Doe
 Rob Pete

Goals	Tasks
Fix main problems <input type="checkbox"/> Get all main tickets done	Fix issue 123 Jane Doe 6/21/2022 12:00 AM <input checked="" type="checkbox"/>
	Fix issue 343 Rob Pete 6/27/2022 12:00 AM
	Remove popup Tarmo Urrio 6/30/2022 12:00 AM

Yksittäisen tavoitteen näkymä



Prokkis

Tarmo Urrio

Goal

Name

Fix main problems

Description

Get all main tickets done

Due date

6/6/2024

Goal_respondee

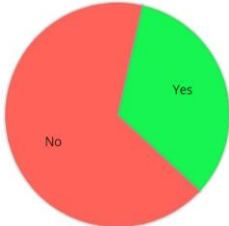
Tarmo Urrio

Edit goal

Remove goal



Name	Assigned to	Due date	Done?
Fix issue 123	Jane Doe	9/6/2023	Yes
Fix issue 343	Rob Pete	3/6/2024	No
Remove popup	Tarmo Urrio	6/6/2024	No

33.33%
Tasks done




Category	Count
Yes	1
No	2

Käyttäjien lisäys projektille



Prokkis

Tarmo Urrio

Add New Member

User

Tarmo Urrio

▼

Add Member

Remove Member

Selected: Jane Doe

Remove Member

Members

Jane Doe

Rob Pete