Graafisen ohjelmoinnin alustat ja niiden erot

Low-code- ja No -code -alustat ohjelmistokehityksessä

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutus

Kevät, 2022

Niclas Koskinen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Niclas Koskinen Vuosi 2022
Työn nimi Graafisen ohjelmoinnin alustat ja niiden erot
Ohjaaja Lasse Seppänen

Low-code- ja no-code -alustat ovat graafiseen ohjelmointiin tarkoitettuja ohjelmia, joilla on mahdollista luoda ohjelmistoja parhaimmillaan kirjoittamatta riviäkään varsinaista koodia. Low-code -ohjelmistoissa on mahdollista muokata ohjelman koodia tai luoda toiminnallisuuksia jo valmiiden toiminnallisuuksien lisäksi tai niiden päälle. No-code-alustoilla ohjelmia luodaan esimerkiksi yhdistelemällä työnkulkuun valmiita toiminnallisuuksia.

Opinnäytetyön tavoitteena on tarjota ajantasainen katsaus graafisen ohjelmoinnin nykytilasta, antaa kattavaa dokumentaatiota eri alustojen eroavaisuuksista ja kertoa niistä yleisesti, sekä osoittaa lukijalle mihin nyansseihin kannattaa kiinnittää huomiota, kun alustaa lähdetään projektiin tai yrityksen tarpeita vastaamaan valitsemaan.

Toiminnallisessa osiossa tarkastellaan sekä low-code-, että no-code -alustojen toiminnallisuuksia ja komponentteja osoittamaan näiden erot ja auttamaan lukija alkuun graafisen ohjelmoinnin parissa. Työssä käydään myös läpi lisenssi ja tietoturva aspektit, jotka ovat tärkeitä huomioon otettavia asioita, kun graafisilla ohjelmointialustoilla lähdetään ohjelmistoja luomaan. Työ pyrkii vähentämään vaivaa sopivan alustan etsimisessä sekä auttaa lukijaa kartoittamaan tarpeensa, minkä pohjalta on helppo valita alusta ja aloittaa ohjelmiston kehittäminen. Lopussa autetaan lukijaa alustapäätöksen teossa, sekä projektimallien käytössä osana ohjelmistokehitystä.

Avainsanat Graafinen-ohjelmointi, ohjelmistotuotanto, GDPR, API-rajapinta, LCPD
Sivut 41 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Author Niclas Koskinen
Subject Platforms and differences of graphical programming
Supervisors Lasse Seppänen

Abstract
Year 2022

Low code- and no code -platforms are software types which are used to develop applications without writing a single line of code or only limited coding is needed. On the low-code -platforms one can customize source code and add own customizations or functionalities with own code. On the no code -platforms you can develop applications with adding functionalities to workflow.

The goal of this thesis is to offer a up to date review where graphical programming is and where it's going in future and tell, about differences of different platforms depending, is it either low-code- or no-code -based. There will also be information what are the main points that needs to be considered when choosing a platform for project or to fulfill the needs of company.

At the functional part there will be tested both low code- and no code -platforms to show reader differences, functionalities and frames of graphical programming platforms and help reader to start programming with those. In this thesis there will be a discussions about the licenses and security aspects, which are important things to consider when starting developing apps with graphical programming platforms. The thesis will decrease effort of choosing the platform and help the reader to map his/her needs and start developing. At the end there is deliberation which reader can use as a tool when making decision between platforms and how to manage and use different types of project models

Keywords Visual programming, software production, GDPR, API-interface, LCPD
Pages 41 pages

Sanasto

GDPR	EU:n yleinen tietosuoja-asetus.
Projektimalli	Ohjelmistokehityksessä käytettävä malli, jonka pohjalta ohjelmisto luodaan
API-rajapinta	Rajapinta, jonka kautta on mahdollista esim liittää eri ohjelmistojen moduuleita yhteen.
Tietokanta (Database)	Tietopankki, joka sijaitsee palvelimella tai tietokoneella, johon on mahdollista suorittaa hakuja ja tallettaa tietoa.
Työnkulku (Workflow)	Ohjelmistossa looginen jana, missä eri toiminnot suoritetaan tai, missä tieto liikkuu objektista toiseen.
Elementti (element)	Alustoissa käytettävät visuaaliset objektit esim. napit, joita on mahdollista lisätä applikaatioon.

Sisällys

1 Johdanto

Digitalisaatio on osa yritysten liiketoiminnan ydintä kasvavalla prosentilla jokapäiväisessä elämässä. Prosessien monimutkaistuessa ja kasvaessa yhä suurempiin mittakaavoihin on automaatiolla ja sovellusten käyttöönotolla suuri merkitys, jotta tehokkuus saadaan maksimoitua ja kulut minimoitua. Yritykset tarvitsevat yhä enemmän digitaalisia työnkulkuja ja sovelluksia käyttöönsä ja kohtaavatkin kysymyksen siitä kannattaako sovellusten kehittäminen ulkoistaa jollekin ulkoiselle kehittäjälle vai mahdollisesti alkaa tuottamaan ohjelmistokehitystä yrityksen sisässä. Graafisen ohjelmoinnin työkalut mahdollistavat ohjelmistokehityksen ja erilaisten työnkulkujen luomisen ilman, että kehittäjältä vaaditaan harjaantumista ohjelmoinnin saralla.

Tämä opinnäytetyö tulee käsittelemään graafista ohjelmointia ja sen kehyksiä, mahdollisia hyötyjä, haittoja sekä nyansseja mitä yrityksen tai yksittäisen kehittäjän tulisi ottaa huomioon, kun sovellusta lähtee graafisen ohjelmoinnin alustalla luomaan. Työ käsittelee graafista ohjelmointia jakamalla sen kahteen osaan, sekä low-code-, että no- code -alustoihin. Näihin kahteen alalohkoon tullaan pureutumaan tarkemmin sekä tarjoamaan käytännön esimerkkejä lukijalle, siitä kuinka nämä eroavat toisistaan ja kuinka nämä erot tarvitsevat valintaprosessissa ottaa huomioon. Opinnäytetyön tavoitteena on tarjota kattava selitys graafisesta ohjelmoinnista sekä toimia valintaa helpottavana dokumenttina, joko yritykselle tai yksityishenkilölle.

Työ on jaettu kahteen osaan. Teoriaosasssa kerrotaan kattavasti, mistä graafisessa ohjelmoinnissa on kyse sekä autetaan lukijaa erottamaan low-code- alustojen ja no code -alustojen erot. Teoriaosa tulee alustamaan käytännön osaa, jossa esitellään ohjelmointia kummankin alalohkon alustoilla. Työhön on valittu yksi low-coden- ja yksi no-coden -alusta ja näillä testataan alustojen elementtejä, integraatioita ja työnkulkujen luontia ja nämä vaiheet on dokumentoitu ja lukija voi käyttää tätä dokumentaatiota, joko apuna alkuun pääsemisessä tai apuna erottamaan näiden kahden sovelluksen erot ja viitekehitykset. Lopussa tullaan tarkastelemaan eroavaisuuksia, sekä vastaamaan kysymyksiin, siitä kumpi alusta soveltuu mihinkin tarkoitukseen ja millaiset ominaisuudet ne sovellukselle tarjoavat.

Opinnäytetyön tavoitteena on vähentää yrityksen resurssien käytön tarvetta ohjelmointialustan valitsemiseen, mikä mahdollistaa resurssien ohjaamisen sovelluksen perinpohjaiseen suunnitteluun ja testaamiseen sekä samalla havainnollistaa, miten alustojen skaalautuvuus ja ominaisuudet ilmenevät alustojen hinnoissa.

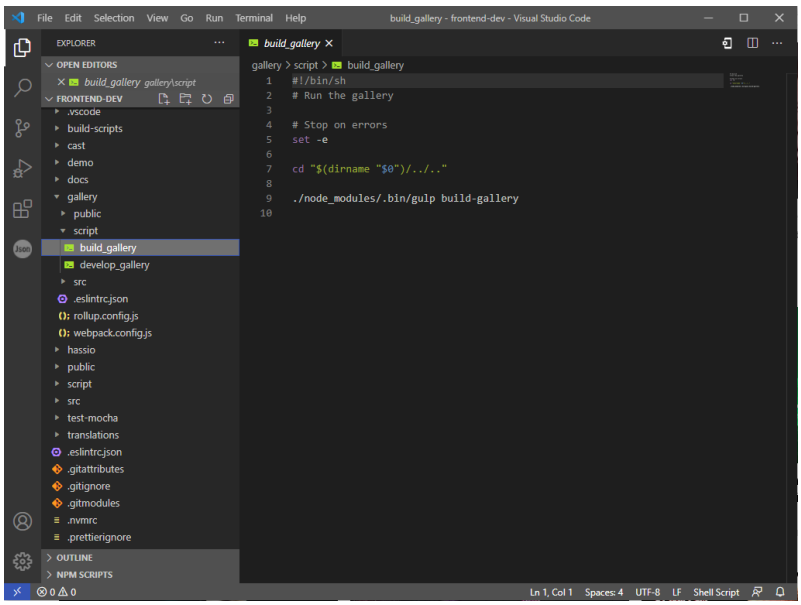
Tutkimuskysymykset:

- Mitä ovat low-code- ja no-code -alustat?
- Kumpi graafisen ohjelmoinnin alustatyypeistä sopii mihinkin tarkoitukseen?
- Miten prosessimallit toimivat graafisen ohjelmoinnin alustoilla?

2 Graafinen ohjelmointi

Graafinen ohjelmointi on tapa luoda ohjelmistoja alustoilla, joissa logiikoita ja toiminnallisuuksia yhdistelemällä voidaan saada aikaan applikaatio, työnkulku tai sovellus vähällä koodin kirjoittamisella tai kirjoittamatta varsinaista koodia riviäkään. Alla olevassa kuvassa (Kuva 1.) on vertailuksi otettu kuva perinteisestä editorista, jolla ohjelmoidaan. Matthew Revell (2021) sanoo visuaalisesta ohjelmasta artikkelissaan ”What is Visual Programming” seuraavalla tavalla: ”Whereas a typical text-based programming language makes the programmer think like a computer, a visual programming language lets the programmer describe the process in terms that make sense to human.” Eli graafinen ohelmointi on käyttäjäystävällisempi tapa myös henkilöille, jotka eivät omaa syvällisempää taustaa ohjelmoinnin saralla. Yleisimmin graafisen ohjelmoinnin alusta ilmenee peruskäyttäjille esimerkiksi kotisivukoneina, joilla voi luoda omat nettisivut ja joiden räätälöintiin ei tarvitse kirjoittaa koodia, mutta joiden kustomointi onnistuu graafisen käyttöliittymän kautta. Taustalla kuitenkin tapahtuu valintojen osalta esimerkiksi javascript, html ja CSS -elementtien vaihtumista, vaikka se ei loppukäyttäjälle näykään.

Kuva 1 Perinteinen ohjelmointiin käytettävä editori (Visual Studio Code).



2.1 Graafisen ohjelmoinnin historia

Matthew Revell kertoo artikkelissaan ”What is Visual Programming”, että graafisen ohjelmoinnin alustojen kehittämisen katsotaan alkaneen 1990-luvulla ja näillä alustoilla luotiin pääsääntöisesti pelejä, multimediatyökaluja ja tietokantoja. Rational Software kehitti UML (Unified Modeling-ohjelmointikielen) ohjelmointikielen, jolla oli mahdollista luoda kokonaisia järjestelmiä niin, ettei varsinaista koodia tarvinnut kirjoittaa riviäkään. Se käytti visuaalista ohjelmointia hyödyksi, mutta ei ollut varsinainen virallinen graafisen ohjelmoinnin alusta. Visuaalinen ohjelmointi jakautui lopulta kolmeen eri alalohkoon. Nämä lohkot keskittyivät peleihin ja opettamiseen, multimediatyökaluihin ja liiketoimintaan keskittyviin alustoihin. 2000- ja 2010-luvuilla alustojen määrä, kehitys ja on kasvanut räjähdysmäisesti ja tämä on mahdollistanut myös yrityksille mahdollisuuden ohjelmistokehitykseen yrityksen sisällä sen sijaan, että se ostettaisiin palveluna esimerkiksi ulkoiselta ohjelmistotalolta.

2.2 Graafisen ohjelmoinnin alustat ja GDPR

Graafisen ohjelmoinnin alustat ovat nykyään suurimmaksi osaksi pilvipalveluita ja arkaluonteista tietoa käsittelevät ja liikuttelevat sovellukset voidaan joutua siirtämään näille alustoille mikäli yrityksen sovellukset ja työnkulut halutaan alustalla yhdistää. Palveluntarjoajien sivuilla kannattaakin perehtyä löytyykö alustalta GDPR-hyväksytty lisenssi.

Jotta GDPR:n täyttämät ehdot sovellusten mukaan täyttyisivät, pitää kehittäjän ottaa huomioon seuraavat seikat: Tuotannossa on käytettävä ainoastaan oikeaa dataa, muutoin on käytettävä keinotekoista dataa. Alustan toimittajan kontolle jää alustan ja infran tietoturvallisuus ja testaaminen. Mikäli graafisen ohjelmoinnin alustoilla luodaan uusia sovelluksia, joissa liikutetaan arkaluonteista dataa on sovelluksissa käytettävien valmiiden komponenttien testaus toimittajan vastuulla ja näiden komponenttien välisen työnkulun turvallisuus kehittäjän kontilla. Sekä alustan toimittajan, että sillä kehittävän henkilön tai organisaation on siis molempien luotava kattava tietoturvasuunnitelma, jotta kehitettävät sovellukset tulevat täyttämään GDPR:n täyttämät ehdot.

3 Low-code- ja no-code -alustat

Graafisen ohjelmoinni alustat voidaan jakaa alustoihin, joissa koodia pääsee muokkaamaan kehittäjän puolesta (Low-code -alusta) ja alustoihin, joissa lähde- ja muuta koodia ei pääse muokkaamaan vaan kehittäminen tapahtuu elementtien valmiiden ominaisuuksien määrittämässä kehyksessä. Molemmilla vaihtoehdoilla on mahdollisuus luoda yrityksen liiketoimintaa parantavia sovelluksia tai työnkulkuja, mutta toiminnallisuuksien laajuudessa tulee olemaan vaihtelevuutta. Alustoja on hinnaltaan vaihtelevasti ja osa on täysin ilmaisia, näissä kuitenkin on nyansseja jotka tulee ottaa huomioon ja tullaan myöhemmin työssä käymään läpi. Alla olevaan taulukkoon (Taulukko 1.) on dokumentoitu lyhyesti peruserot näiden kahden alustatyypin välillä.

Taulukko 1 Alustojen yleiset eroavaisuudet listattuna

	No-code-alusta	Low-code-alusta
Käyttäjät	Ei vaadi ohelmointi-osaamista, vaan loogisen ajattelun taitoa	Käyttö vaatii perus osaamista ohjelmoinnin saralta
Alusta	Alustalla ei tarvitse kirjoittaa riviäkään koodia, vaan graafisella käyttöliittymällä voidaan esimerkiksi yhdistellä ”toiminnallisuus-elementtejä” yhteen ja luoda näillä työnkulkuja ja applikaatioita.	Alustalla on mahdollisuus tehdä muutoksia graafisiin elementteihin, back-endiin ja API-rajapintoihin yhdistämiseen koodin avulla. Käyttöliittymä voi olla hyvin lähellä No-code alustan käyttöliittymää, mutta tarjoaa mahdollisuuden muokata koodia
Alustalla luotujen ohjelmien tyyppi	Yksinkertaisia sovelluksia	Mahdollisuus luoda toiminnallisuuksiltaan laajempia applikaatioita.
Tavoite	Mahdollistaa yksinkertainen sovelluskehitys, ilman ohjelmointi-osaamista	Nopeuttaa ohjelmistokehitystä, ohjelmoijille.

3.1 Low-code-alustat

Low-code-alustoilla tarkoitetaan ohjelmistokehitykseen luotuja ohjelmia, joilla varsinainen toimintojen ohjelmoiminen toteutetaan graafisen käyttöliittyymän kautta. Low-code-alustat on alun perin luotu nopeuttamaan ohjelmistokehittäjien työtä niin, ettei toistuvia perustoimintoja tarvitse ohjelmoida toistamiseen applikaatioon, näitä voivat esimerkiksi olla jotkin graafisen käyttöliittymän napit. Alustoilla on mahdollisuus muokata ohjelman back-end-koodia niin, että esimerkiksi eri API-rajapintoihin voidaan luoda yhteyksiä, sekä luoda dynaamisia applikaaatioita ja prosessiautomaatioita, jotka skaalautuvat mille tahansa päätelaitteelle . Digian selvityksen mukaan noin 85 prosenttia päättäjistä pitää ainakin jonkinasteisena ongelmana tietotyön rutineeja ja toistuvia työvaiheita. Low-code mahdollistaa irtipääsemisen osasta näistä toistuvista työvaiheista. (Digia, 2021). Alustat ovat nykyään pilvipalveluina, joten niiden hankkiminen ja kokeileminen on helppoa.

3.1.1 Low-code-alustojen historiaa

Ensimmäinen harppaus kohti graafisen ohjelmoinni alustoja oli Applen kehittämä HyperCard (Christopher Tozzi, The evolution of Low-code/no-code Development, 30.11.2021). Tämä alusta mahdollisti alustojen ja kehyksien luomisen ilman varsinaista ohjelmointia, mutta antoi kehittäjille mahdollisuuden koodin muokkaamiseen. Ensimmäiset alkeelliset low-code kehitysalustat alkoivat syntyä 1990-luvun ja 2000-luvun vaihteessa, vaikka ne eivät suoranaisesti kuitenkaan vastanneet nykykäsitystä low-code -alustasta. Ne kulkivat tuolloin nimellä LCPD (Low-Code Development Platform). Tuolloin ne pystyivät alkeelliseen koodin generoimiseen visuaalisen ohjelmoinnin pohjalta. Low-code -termin keksi Forrester Research 9.6.2014.

3.1.2 Tietoturva low-code-alustoilla

Low-coden salliessa räätälöinnin kehittäjälle, jakautuu tietoturvasta vastaaminen osittain alustan omistajan ja osittain kehittävän tahon harteille. Tuotteen omistaja on vastuussa siitä, että sovellus itsessään on tietoturvallinen ja GDPR:n mukainen sekä komponenttien turvallisuudesta. Riippuen siitä missä palvelu pyörii esim. palvelin on joko tuotteen omistaja tai palvelimen omistaja vastuussa sen tietoturvasta. Esimerkkinä voidaan mainita Microsoftin tuottama Powerapps-palvelu, joka pyörii Microsoftin omalla palvelimella, tämä laittaa Microsoftin vastuulliseksi palvelimen ja infran tietoturvasta.

Kehittäjän tehtäväksi jää sovelluksen itsensä tietoturvallisuus ja mikäli palvelu/sovellus pyörii kehittäjän tai yrityksen omalla palvelimella, niin kyseisen palvelimen tietoturvallisuuden ylläpito ja tarkistaminen. Alustoilla on mahdollisuus myös käyttää kolmannen osapuolen ”widgettejä”, eli ns. lisäosia tai lisätoiminnallisuuksia, näiden tietoturvallisuuden tarkistaminen jää kehittäjälle, mikäli hän näitä sovelluskehitykseen ottaa mukaan. Mitä tulee datan keräämiseen ja säilyttämiseen on kehittäjän vastuulla vastata tiedon turvallisesta liikuttamisesta applikaation/työnkulun sisällä, sekä turvallisesta säilyttämisestä.

Osassa alustoissa voi olla ikävä tilanne, että kehittäjä ei pääse tarkastamaan kaikkien komponenttien lähdekoodia, mikä voi johtaa siihen, että arkaluonteista tietoa kuljetetaan suojaamattomien komponettien läpi datana. Low-code-alustoilla tietoturvariskeille altistaa myös koodin muokkaamisen mahdollisuus, sillä kokemattomat kehittäjät voivat kopioida internetistä valmiita ohjelmanosia koodina ja leikkaa-liimaa-menetelmällä yhdistää ne omaan ohjelmaansa sen pahemmin sen tietoturvaa ajattelematta.

Koska low-code-alustan käyttö vapauttaa yrityksen resursseja voidaan esimerkiksi tietoturvan testaaminen aloittaa huomattavasti paremmilla resursseilla sekä aikaisemmin. Lisäksi valmiiden komponenttien testaaminen on nopeampaa eikä vaadin kuin yhden testauskerran. Myös suunnitteluun vapautuu enemmän resursseja ja aikaa. Tässä yrityksen tai kehittäjän kannattaa kiinnittää huomiota kattavaan ja turvalliseen suunnitteluun, uhkakuvien kartoittamiseen, sekä koodin tarkistamiseen. Koska koodia on suhteessa vähemmän normaaliin koodipainotteiseen ohjelmistokehittämiseen, käy sekin nopeammin.

3.1.3 Low-code-alustojen skaalautuvuus ja hyödyt.

Koska graafinen ohjelmointi tapahtuu valmiilla alustalla, on siinä useamman kehittäjän samanaikainen työskentely huomattavasti dynaamisempaa kuin perinteinen ei-graafinen ohjelmointi, jossa projektit tarvitsee ladata erilliseen palveluun josta ylläpito ja versionhallinta tapahtuu.

Alustat ovat lähtökohtaisesti hinnaltaankin skaalautuvia ja usein lähtöhinnat perus-ominaisuuksilla ovat hyvin halpoja eivätkä hinnat/kuukausi nouse kovin ylös kun erillisiä toimintoja aletaan ottamaan lisähinnasta käyttöön. Sovelluksia on myös maksullisuudeltaan erilaisia, on user per plan -tyyppisiä, missä jokaisesta käyttäjästä maksetaan kiinteää kuukausihintaa, sekä alustoja missä maksetaan projektikohtaisesti hintaa per kuukausi. Suurinpiirtein kaikki sovellukset tarjoavat ilmaisen kokeilujakson, jolloin juuri omaan projektiin sopivan alustan löytämiseen ei tarvitse varata pääomaa ja käyttöliittymää pääsee testaamaan.

Riippuen yrityksen koosta, voi parhaimmillaan yrityksellä olla käytössä useita eri sovelluksia ja työnkulkuja ja dataa voi sijaita useassa eri paikassa, ja näiden liikuttelu paikasta toiseen voi olla työlästä sekä resursseja sitovaa. Low-coden ansioista sovellukset voidaan luoda samalle alustalle/ samaan sovellukseen, mikä mahdollistaa todella dynaamisen ja prosessia nopeuttavan teknologian käyttöönoton yrityksessä. Tämän vaikutus on se, että prosessit paranevat, analysoitava data on helpommin saatavilla sekä helpommin analysoitavissa, sekä uusia toiminnallisuuksia, sovelluksia tai työnkulkuja on helpompi ottaa yrityksen käyttöön jatkossa.

”Low-code/no-code-alustat kehittävät ja laajentavat kyvykkyyksiään ja jatkuvasti, mikä mahdollistaa niiden hyödyntämisen yhä useammissa osissa organisaatiota” (Miika Kallasoja, Yritysjohdonkin olisi syytä kiinnostua low-code/no-code-alustoista, 06/04/2021). Eli siirtämällä sovellukset ja työnkulut valmiille alustalle jää varsinaisen alustan kehittäminen sitä ylläpitävälle taholle, mikä antaa käyttäjälle mahdollisuuden ottaa käyttöön uusimpia ominaisuuksia sekä korjauksia ajantasaisesti ilman, että niitä tarvitsee itse tehdä.

Koska alustoilla on mahdollisuus back-endin muokkaamiseen, on esimerkiksi API-rajapintoihin liittyminen mahdollista ja sovelluksia ja työnkulkuja on mahdollista integroida

muiden järjestelmien kanssa todella pitkälle. Koska sovelluksessa on valmiita toiminnallisuuksia ja elementtejä on ohjelmistojen muokkaaminen tai korjaaminen jälkikäteen kivuttomampaa ja alustojen kehittyessä on esimerkiksi niiden bugi-raportit kattavia ja raportin pohjalta helppo korjata.

Skaalautuvuus yrityksen sisällä on tehty helpoksi. Usein käyttäjämääriä ja oikeuksia projekteihin pystytään hallinnoimaan oman käyttöliittymän kautta, mikä tekee organisoimisesta dynaamisempaa.

3.2 No-code-alustat

No-code-alustoilla tarkoitetaan ohjelmistokehitykseen luotuja alustoja, joissa ei tarvitse kirjoittaa riviäkään koodia. Yleisimmin no-code-alustoihin törmää verkkosivujen luontiin tarkoitetuissa kotisivukoneissa tai pelimoottoreissa, joissa luodaan uusia pelejä. Näillä alustoilla on mahdollista luoda nopeasti yksinkertaisia sovelluksia tai työnkulkuja yrityksen käyttöön ilman, että vaaditaan osaamista ohjelmoinnista.

3.2.1 No-code-alustojen historiaa

No-code-alustojen esi-isänä voisi pitää Wordpressiä, joka mahdollisti ensimmäisenä alustana nettisivujen rakentamisen ilman koodin kirjoittamista. Wordpress ei kuitenkaan ole puhtaasti no-code alusta, sillä siinä on mahdollista muokata nykyisin koodiakin. Google sheets, joka on listattu Zapierissa 2012 ja Integromatissa vuonna 2016 parhaimmaksi no-code-alustaksi kehitettiin vuonna 2006 ja Arpit Choudhuryn artikkelissa ”A Brief History of No-code” kerrotaankin, että suurin osa nykyisistä no-code-alustoista onkin rakennettu Google Sheetsin päälle ja Google jatkaa suunnannäyttäjänä no-code-ohjelmistokehityksessä.

3.2.2 Tietoturva no-code-alustoilla

Lähtökohtaisesti, kun no-code alustaa valitaan kannattaa kiinnittää huomiota sen dokumentointiin tietoturvasta sekä lisenssoinnista, sillä jos kehittäjillä ei ole mahdollisuutta tarkastella alustan komponenttien ja alustan koodia joudutaan tilanteeseen, missä luotetaan sokeasti tietoturvaan. Läpinäkyvyys on siis nyanssi, jota tulee pitää arvossaan kun alustaa lähdetään valitsemaan. Komponenttien välisen työnkulun tietoturvan suunnittelu kuuluu kehittäjälle ja tietoturvatestausta tulee harjoittaa alusta asti kun sovellusta lähdetään alustalla kehittämään. No-code-alustoille luodut sovellukset pyörivät pääsääntöisesti palveluntarjoajan omalla palvelimella, jolloin palvelun tietoturva kuuluu alustan omistajalle. Jo ennen alustan käyttöönottoa tarvitsee yrityksen tai kehittäjän laatia kattava tietoturva-sunnitelma/kartoitus ja aloittaa testaus sen pohjalta.

Siinä missä no-code-alustan rajoittuneisuus toiminnallisuuksien osalta tekee siitä köyhemmän verrattuna low-code-alustoihin, antaa se kehittäjälle vähemmän mahdollisuuksia tehdä aukkoja tietoturvaan, eli mikäli alusta on GDPR:n mukainen ja kattavasti tietoturvan osalta testattu ja dokumentoitu on se parempi vaihtoehto kuin low-code alusta.

3.2.3 No-code-alustojen skaalautuvuus ja hyödyt

Silicon Valley Cloud IT sivuston blogipostauksessa ” Is No-Code Platform scalable enough for enterprise?” pureudutaan no-code-alustojen skaalautuvuuten yrityksen tarpeissa. ”Alusta itsessään on merkittävin tekijä kun kyseessä on skaalautuvuus, sillä toiminnallisuudet, ominaisuudet sekä toimittajan päivitysaktiivisuus määrittää pitkälti sen onko alustalla luotavat sovellukset skaalautuvia yrityksen tarpeisiin. Yrityksellä/ kehittäjällä toki on myös vastuu luoda sovellus niin, että uusien ominaisuuksien liittäminen tai sovelluksen päivittäminen käy kivuttomasti. Tämä on kuitenkin ominaisuus joka käy perinteistä ohjelmointia kivuttomammin ja nopeammin ja on nopeammin testauksessa ja tuotannossa.”

No-code-alustoilla luotavat sovellukset ovat myös helpommin testattavissa, sillä valmiit komponentit ovat yleensä valmiiksi testattavia, joten kehittäjälle jää testattavaksi ainoastaan tiettyjen työnkulkujen tai yleistoimivuuden testattavuus. Suorituskyky on myös skaalautuvuutta määrittävä tekijä ja graafisen ohjelmoinnin alustoilla luotavat sovellukset

ovat lähtökohtaisesti suorituskykyisempiä valmiiden ratkaisujen ansiosta. Graafinen käyttöliittymä sovelluksille on myöskin nopeasti rakennettavissa ja tämä mahdollistaa paremman käyttökokemuksen.

Parhaimmat no-code-alustat tarjoavat laajat mahdollisuudet API-rajapintojen käyttöön sovelluksissa joten sovellus on skaalattavissa todella nopeasti ja useita ominaisuuksia pystytään tämän ansiosta integroimaan yhteen. Koska lähtökohtaisesti ohjelmointiosaamista ei no-code-alustoissa tarvita, on API-rajapintoihin liittyminenkin tehty kivuttomaksi.

Ehdottomia hyötyjä no-code-alustan käyttöönotossa on nimenomaan sen skaalautuvuus omassa kehyksessään ja, että loogisella ajattelukyvyllä kuka vain voi ohjelmointitaidoista riippumatta luoda suorituskykyisiä simppeleitä sovelluksia ja työnkulkuja yrityksen käyttöön.

Hinnoiltaan no-code-alustat vaihtelevat kuukausimaksuiltaan 1 dollarista 5000 dollariin ja tässä kannattaakin käydä pohdintaa, mitä toiminnallisuuksia ja ominaisuuksia sovellukselta ja alustalta tullaan vaatimaan, mihin hintaan tämän saisi ja miltäkin palvelun tarjoajalta. Hinnat skaalautuvat myös per user -periaatteella, eli jokaisesta käyttäjästä voi joutua maksamaan oman hintansa, mikä tietenkin parhaimmillaan voi tippua käyttäjämäärän kasvaessa tiettyyn pisteeseen.

4 Ohjelmointi graafisen ohjelmoinnin alustoilla

Tässä luvussa tullaan käymään läpi kaksi graafisen ohjelmoinnin alustaa ja pureudutaan niiden ominaisuuksiin. Käytännönosaan on valittu yksi no-code- ja yksi low-code -alusta. Molemmista alustoista on dokumentoitu käyttöliittymän, elementtien, työnkulkujen sekä integraatioiden käyttäminen käytännössä.

Dokumentaatio on pyritty rakentamaan niin, että sen avulla pääsee molempien alustojen käytössä alkuun ja jo pelkästään lukemalla sen saa käsityksen, miten alustat toisistaan eroavat. Low-code-alustaksi valikoitui Retool ja no-code-alustaksi Bubble.

4.1 Retool

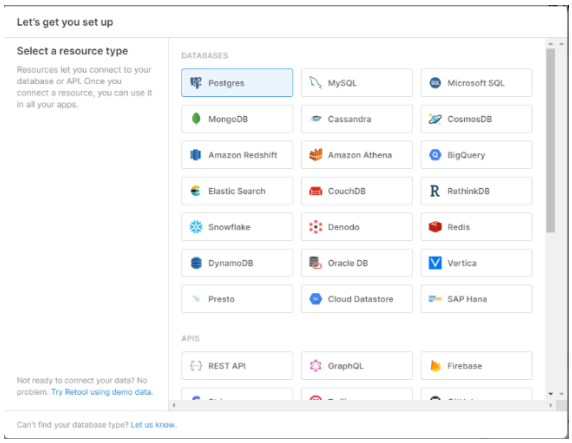
Low-code-alustaksi valitsin Retool nimisen alustan joka oli arvosteltu toiseksi parhaimmaksi softwaretestinghelp.com toteuttamassa arvostelussa. Alustan on luvattu skaalautuvan sekä pienten ja suurten yritysten käyttöön.

Alusta on pilvipohjainen ja sitä pystytään käyttämään seuraavilla alustoilla:

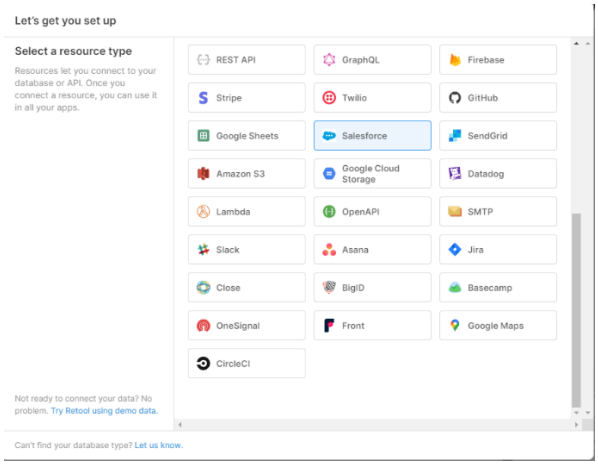
- Google chrome
- Mozilla firefox
- MacOS
- Windows
- Safari
- Internet Explorer

Alusta tarjoaa mahdollisuuden yhteen ilmaiseen sovelluksen tekoon ja tämän jälkeen 10 dollaria/kuukausi/käyttäjä ja kaupallisille sovelluksille 50 dollaria/kuukausi/käyttäjä. Retool antaa mahdollisuuden liittyä useisiin tietokantoihin (Kuva 2), sekä API-rajapintoihin (Kuva 3)

Kuva 2 Retoolin tarjoamat mahdollisuudet tietokantojen yhdistämiseen sovelluksessa.



Kuva 3 Retoolin tarjoamat API-rajapinta-yhdistämismahdollisuudet sovelluksessa

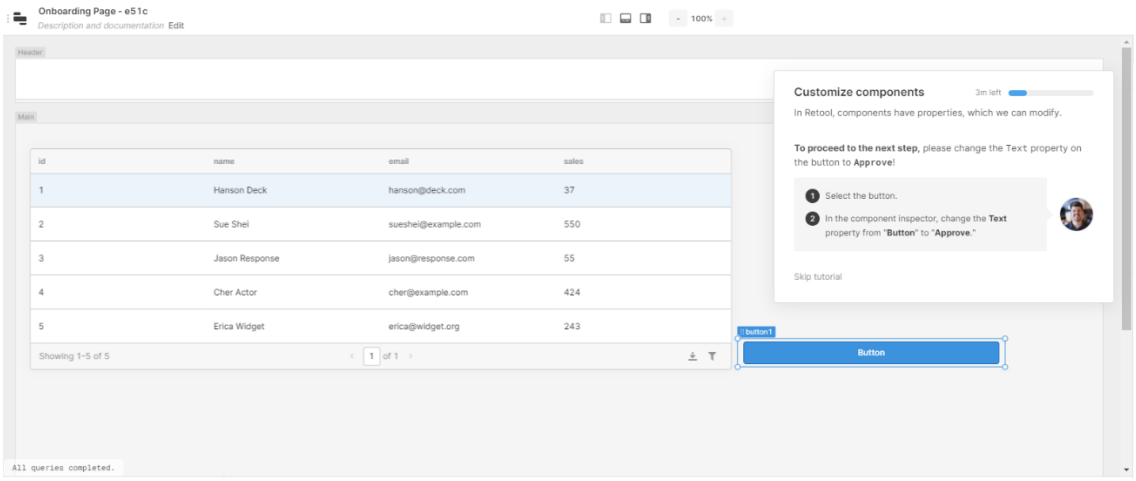


4.1.1 Retoolin käyttöönotto

Retoolin käyttöönotto tapahtuu helposti heidän omalta verkkosivultaan, jossa tarvitsee vain luoda tunnukset ja ensimmäistä sovellusta pääsee luomaan syöttämättä maksutietojaan, mikä on kokeilun kannalta todella luonteva toteutus.

Sovellus tarjoaa ensimmäisellä avauskerralla tutustumiskierroksen, jossa rakennetaan yksinkertainen tietokantataulukosta tiedon hakeva sovellus ja opetetaan myös miten API-rajapintaan otetaan yhteys (Kuva 4). Retool on myöskin kasannut kirjaston yksinkertaisimmista valmiista luomuksista, mistä osan on luonut Retoolin tiimi ja osan muut kehittäjät. Tämä antaa mahdollisuuden yritykselle poimia valmis pohja sovelluksellensa, jota voi lähteä nopealla aikataululla räätälöimään omaa käyttökokemusta vastaavaksi.

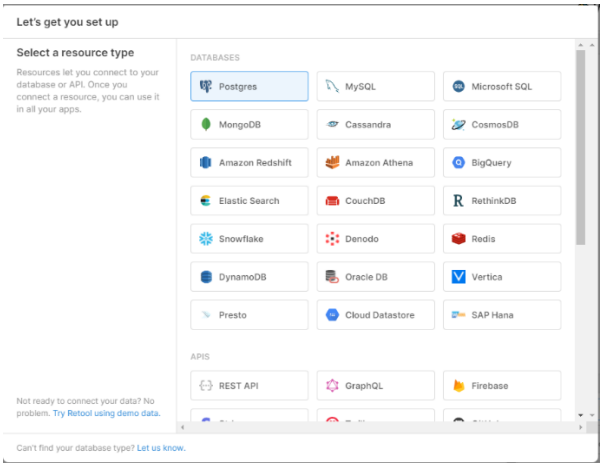
Kuva 4 Perehdyttäminen sovelluksen ominaisuuksiin tapahtuu simppelin sovelluksen ohjatulla rakentamisella.



4.1.2 Tietokantaan yhdistäminen Retoolilla

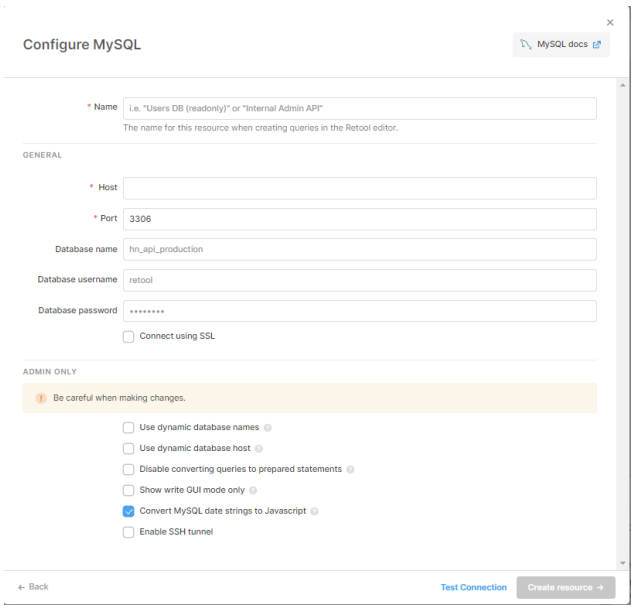
Kun Retoolilla avaa uuden tyhjän projektin, pyytää se ensimmäisessä vaiheessa yhdistämään johonkin olemassa olevaan tietokantaan (Kuva 5) Tämä on kuitenkin mahdollista ohittaa.

Kuva 5 Uuden projektin avaaminen Retoolissa antaa mahdollisuuden luoda tietokantayhteyden heti ensimmäisessä vaiheessa.



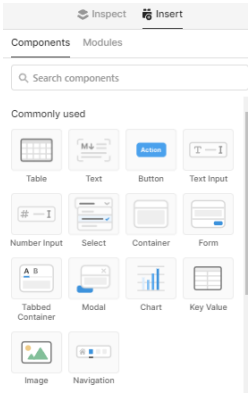
Esimerkiksi valitsemalla MySQL-tietokannan valikosta aukaisee se ikkunan missä on mahdollista syöttää tietokannan tiedot. Tietoturvan kannalta on hyvä, jos se on suojattu sillä Retoolilla on mahdollista ottaa yhteys myös suojattuu tietokantaan ja sekä tunnuksen, että salasanan pääsee syöttämään käyttöliittymässä.

Kuva 6 SQL-tietokannan määrittely.



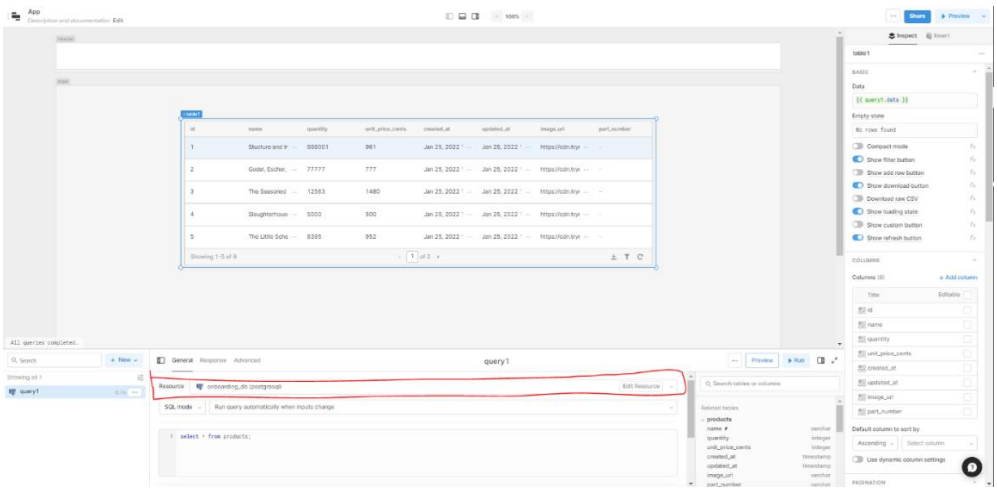
Kun haluttu tietokantayhteys on luotu voi applikaatioonsa valita haluamansa elementin mihin tietokannasta voi tuoda tiedon (Kuva 7). Mikäli esimerkiksi tietokannasta haluttaisiin hakea jotain tiettyjä tietoja kannattaisi elementiksi valita Table.

Kuva 7 Retoolin tarjoama perusvalikoima graafisia elementtejä.



Sovelluksen alustapohjalle raahattu ”Table”-elementti aktivoituu automaattisesti ja testi-applikaatiossani toimii tietolähteenä Retoolin tarjoama esimerkkitietokanta johon on syötetty jonkin verran testaukseen tarkoitettua dataa. Kuvassa (Kuva 8) punaisella ympyröidystä kohdasta voi valita tietolähteeksi haluamansa API-rajapinnan tai tietokannan ja tämän alla on koodieditori johon voi halutessaan luoda SQL-komennon. Se suodattaa Table1-elementin datan.

Kuva 8 Taulukon tietolähteen määrittely

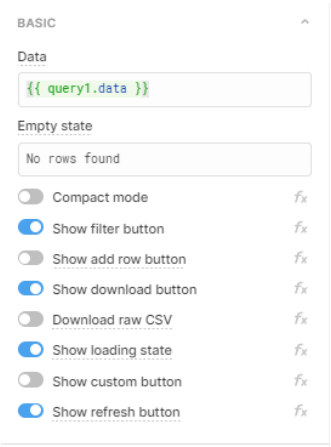


4.1.3 Toiminnallisuuksien lisääminen applikaatioon Retoolilla

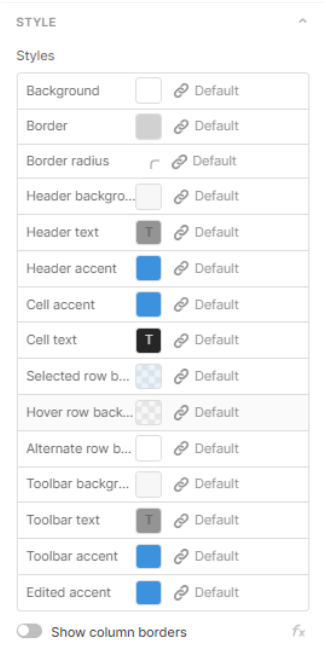
Nyt kun testiapplikaatioon on luotu tietokantayhteys voidaan alkaa luomaan toiminnallisuuksia, joilla voidaan esimerkiksi suodattaa taulukon ”Table 1” dataa.

Retool antaa mahdollisuuden muokata table-elementin tietoja ilman, että se vaikuttaa tietolähteenä toimivaan tietokantaan. Taulukon ulkoasua voi muokata kahdessa eri vaiheessa. Ensimmäinen muokkauskeino on sivupaneelissa sijaitsevasta ”Basic”-kohdasta (Kuva 9) Toinen tapa muokata enemmän visuaalista ilmettä on myös sivupaneelista, kun table-elementti on aktiivisena selataan kohtaan ”Style”. Taulukko on myös mahdollista piilottaa kokonaan näkyvistä tai sille on mahdollista luoda niin sanottu ”event handler”, joka laittaa sen näkyviin esimerkiksi, kun nappia painetaan. Tämä mahdollistaa sen, että elementtejä voi laittaa alustalle todella paljon ja niiden näkyvyyttä pystytään säätelemään manuaalisesti.

Kuva 9 Table-elementin muokkausvaihtoehdot kohdasta Basic.

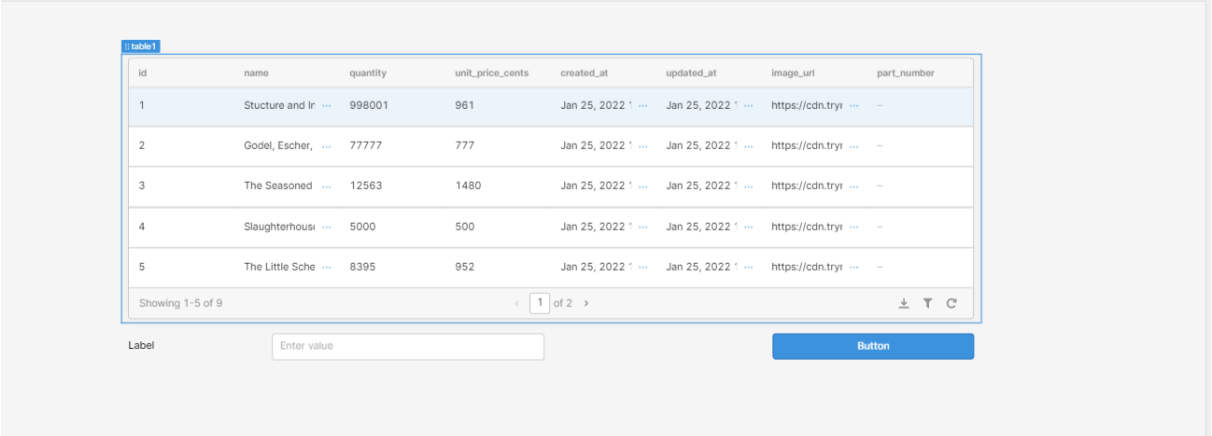


Kuva 10 Table-elementin muokkausvaihtoehdot kohdasta Style.



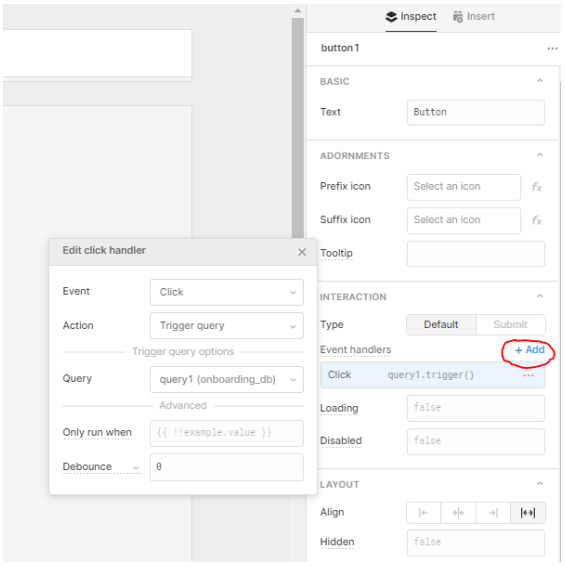
Seuraavaksi testiapplikaatioon luon aiemmin mainitun ”event handlerin”, joka liitetään tekstikentän ja nappulan muodostamaan kokonaisuuteen, jolla voidaan hakea tietoa taulusta. Prosessi aloitetaan raahaamalla alustalle ”Button”-elementti, sekä ”Text input”-elementit (Kuva 11). Näille elementeille on myös mahdollista valita tietolähde sovellukseen liitetyistä API-rajapinnoista tai tietokannoista. Tässä tapauksessa lähteeksi valitaan sama esimerkki-tietokanta, joka on table1-elementissäkin.

Kuva 11 Sovelluspohjalle lisätty text input- ja button -elementit.



Kun lähdetään luomaan ”event handleria” eli toiminnallisuutta sovellukseen valitaan aktiiviseksi button-elementti ja sivupaneelista valitaan event handler kohdan vierestä ”+ Add” (Kuva 12). Tämän jälkeen on mahdollista muokata Eventtiä eli tapahtumaa haluamakseen. Valintoina on muunmuassa verkkosivun avaaminen, scriptin ajaminen tai tässä esimerkissä käytettävä taulukon käyttö. Tässä kohdassa on mahdollista myöskin valita käytettävä tietokanta tai API-rajapinta. Nappulaan voi halutessaa luoda myös javascriptillä haluttuja toimintoja.

Kuva 12 Event handlerin luonti button -elementtiin tapahtuu sivupaneelista elementin ollessa valittuna.



4.1.4 Retoolin ominaisuudet

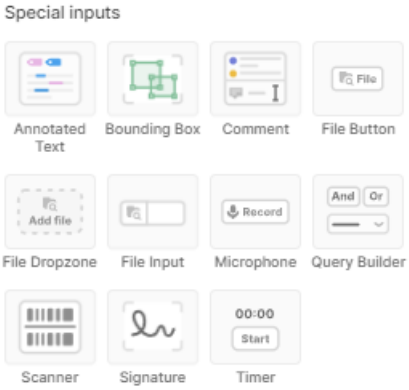
Retool soveltuu yritysten käyttöön, jotka haluavat analysoida, pilkkoa ja kuljettaa dataa ja luoda tähän simppelin käyttöliittymän. Retool käyttää ohjelmointi kielinään SQL:ää ja Javascriptiä. Yritykseltä tarvitsee siis valmiiksi löytyä API-rajapintoja tai tietokantoja, jotta tästä low-code-alustasta saadaan kaikki hyöty irti. Peruspaketista sovellus voidaan integroida taulukon 2 elementteihin.

Taulukko 2 Retoolin peruspaketin elementit.

Elementin nimi	Elementin toiminnallisuudet
Auth Login	Auth loginilla voidaan piilottaa API-rajapintaan liittyminen kirjautumisen taakse.
BrainTree Card Form	BrainTree Card-Formilla voidaan syöttää valmiin formin tekstikenttiin tietoa ja siirtää ne siittä automaattisesti esim tietokantaan.
Looker	Looker on elementti jonka sisässä voidaan käyttää selainta määritetyssä verkkosivussa. Elementtiin voisi esimerkiksi laittaa HAMK:n verkkosivun ja käyttää sitä verkkosivua applikaation sisässä.
Mapbox Map	Mapbox Map-elementillä voidaan ottaa yhteys kartta-API-rajapintaan ja tuoda sovellukseen esim paikka ja karttatietoja.
S3 Uploader	Tämä elementti antaa mahdollisuuden siirtää sovelluksesta tietoa Amazonin AWS:n tarjoamaan S3 tallennusinfrastruktuuriin.
Tableau	Tableau-elementti helpottaa tietokannan datan kuvaamista eri graafeilla ja tähän on sisään rakennettu automaatio joka tunnistaa datan ja kuvaajia saa muokattua sivupaneelista haluamaksensa.

Peruspaketista löytyy kattava valikoima valmiiksi kattava kirjasto myös perus elementtejä, joilla on mahdollista rakentaa monimutkaisempiakin sovelluksia. Riippuen alustasta mitä käyttää on tarjolla usein palveluntarjoajan kehittämiä omia ”erikoisempia” widgettejä. Retool antaa peruspakettissaan käyttöön 11 eri erikoisvimpainta (Kuva 8)

Kuva 13 Retooliin tarjoamat erikoisvimpaimet.



Retoolin elementtikirjastosta löytyy kattavasti erilaisten tiedostojen sekä datan presentointiin tarvittavia valmiita komponentteja. Alustalle on myös luotu ”Container” ja ”Form” elementtien kirjasto, joka mahdollistaa vaivattomamman UI:n (User Interface) rakentamisen. Tämän mahdollistaa se, että containerien ja formien sisään on mahdollista sisällyttää muita elementtejä niin, että työnkulkuja ja toiminnallisuuksia saadaan yksittäisten elementtien sijaan liitettyä suoraa isompiin kokonaisuuksiin kuten formeihin.

Retoolilla on mahdollista luoda todella kattavia ja dynaamisia sovelluksia. Nimenomaan liiketoimintaa tukevien prosessien automatisointiin ja digitalisointiin alusta toimii hyvin. Kattava valikoima integrointimahdollisuuksia, sekä yleisenä ohjelmointikielenä toimiva javascript ja SQL antavat myös kehittäjille suuret raamit joiden sisällä kehittää sovelluksia.

4.1.5 Retoolin käytettävyys.

Käytettävyydeltään Retool on helposti omaksuttava ja koko alustan käyttölittymää pystyy vaivatta muokkaamaan haluamakseen. Alustalle on rakennettu myös oma ”testauslabra” ominaisuus, jolla on mahdollista testata sovelluksen toimintaa ilman, että sitä tarvitsee vielä julkaista esim muille samassa projektissa työskenteleville. Lisäksi sisään rakennettu tutustumiskierros auttaa uudet käyttäjät kattavasti alkuun sovelluksen kehittämisessä.

Komponenttien muokkaaminen ja integrointi applikaatioon on tehty helposti ja yksittäistä komponenttia on helppo muokata sekä ulkoasun, että koodin osalta. Hinnaltaan ja käytettävyydeltään Retool on oiva valinta yritykselle, jolla on käytettävissään jonkinlaista osaamista ohjelmoinnista ja joilla on tarve siirtää prosesseja, työnkulkuja ja sovelluksia saman alustan alaisuuteen.

4.2 Bubble

Bubble on selaimessa käytettävä graafisenohjelmoinnin alusta, joka perustuu no-code pohjaiseen ohjelmoitiin, missä sovelluksia voidaan rakentaa kirjoittamatta ainuttakaan riviä koodia. Ohjelmointi tapahtuu yhdistelemällä valmiita komponentteja työnkulkuun ja näin saadaan aikaan dynaamisia helposti käytettäviä ohjelmistoja. Bubble:sta on saatavilla ilmainen Core-versio, joka sisältää perusominaisuudet, sekä kattavan kirjaston ohjeita sovellusten rakentamiseen. Maksulliset versiot ovat kuukausihinnoiltaan 25 dollarista 475 dollariin. Näistä kallein on yrityskäyttöön tarkoitettu.

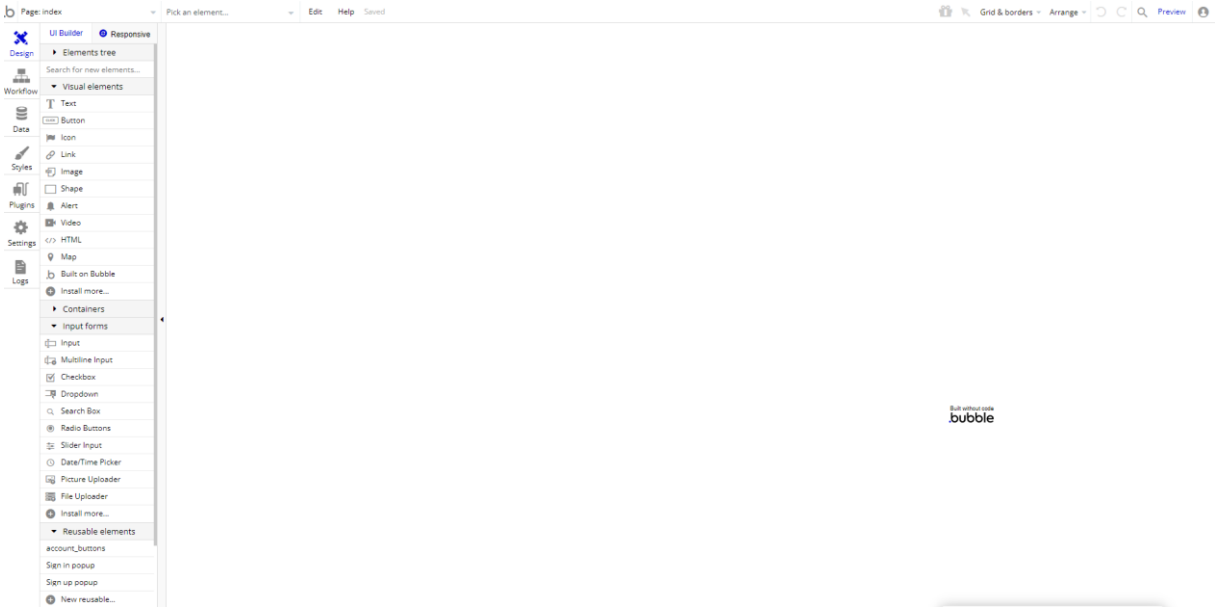
4.2.1 Bubblen käyttöönotto

Bubblen käyttöönotto tapahtuu bubble.io-osoitteesta ja perusominaisuuksia pääsee kokeilemaan täysin maksutta. Palveluun voi kirjautua joko luoden omat bubble-tunnukset tai kirjautumalla google-tilillä.

Kirjautumisen tai rekisteröitymisen jälkeen bubble tarjoaa tutustumiskierroksen alustaan pienellä sovelluksenluonti-tehtävällä, jossa luodaan yksinkertainen google mapsin rajapintaa

hyväksikäyttävä sovellus ohjeita noudattaen. Tässä päästään tutustumaan eri elementtien käyttöön ja räätälöintiin sekä työnkulun luomiseen. Käyttöliittymältään alusta on todella yksinkertainen navigoida ja käyttää (Kuva 14).

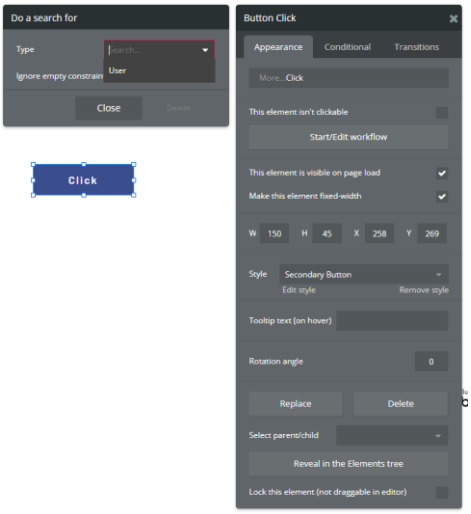
Kuva 14 Bubble:n käyttöliittymä.



4.2.2 Bubblen elementit ja niiden käyttäminen

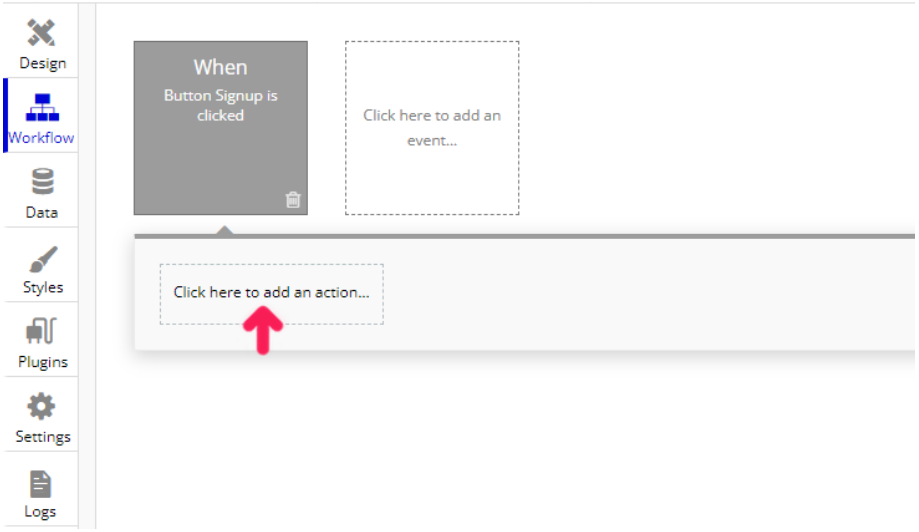
Bubblessa on peruspaketissa kattava määrä erilaisia elementtejä ja niiden kustomointi on tehty helpoksi. Esimerkiksi kuvassa 15 (Kuva 15) on kuvattu ”button”-elementin kustomointipaneeli ja miten ”event handler” luodaan ja sille määritetään toiminnallisuudet.

Kuva 15 ”Button” -elementin kustomointipaneeli ja ”event handlerin” luonti.

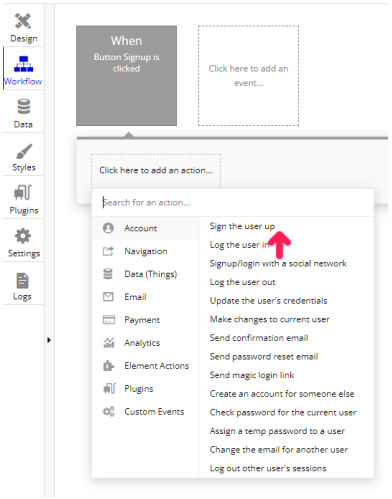


Elementtien työnkulkua eli workflowta pääsee muokkaamaan kustomointipaneelin kohdasta ”Start/Edit workflow”. Esimerkkikuvassa 16 (Kuva 16) luodaan ”button” elementille työnkuva ja kuvassa näytetään myös miltä työnkulun muokkausvälilehti näyttää. Esimerkissä on luotu sign in näppäin johon on liitetty kaksi tekstikenttää ”Email” ja ”Password”. Työnkulun luomiseen on bubblessa luotu valmiita toiminnallisuuksia (Kuva 17). Tässä esimerkissä käytetään valmiiksi luotua ”Sign the user up” kohtaa, joka nopeuttaa sovelluksen kehittämistä huomattavasti. Valitaan työnkulkuun elementtien käyttämät syötteet (Inputit) ja valitaan mitä näiden tietojen tulisi olla (Kuva 18). Työnkulkuun pystyy määrittämään myös paljon muita ominaisuuksia, esimerkiksi varmistussähköpostin lähettämisen kirjautumisen yhteydessä käyttäjän sähköpostiin.

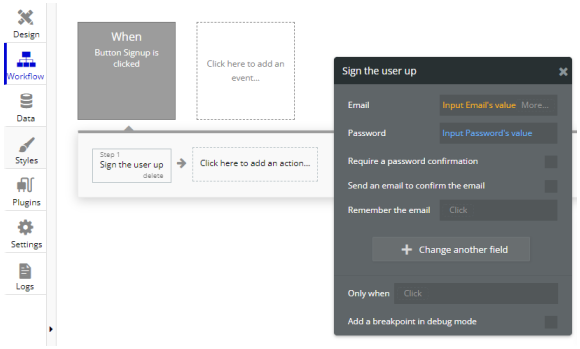
Kuva 16 ”button”-elementille luodaan workflow-välilehdellä työnkulku.



Kuva 17 Työnkulun luontiin kehittäjän valmiita toiminnallisuuksia.

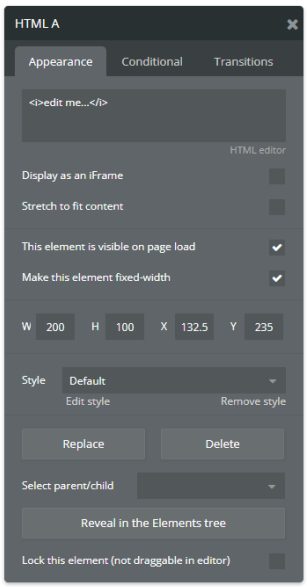


Kuva 18 Työnkulun toiminnallisuuksista valitun ”Sign the user up”-kohdan kustomointi.



Kaikille elementeille on olemassa vastaavanlaiset kustomointipaneelit kuin tässä käytyyn ”button”-elementtiin ja näihin on luotavissa samalla tavalla työnkulkuja. Vaikka kyseessä on no-code-alusta, on elementtikirjastossa kuitenkin ”HTML”-elementti, joka mahdollistaa HTML-koodin kirjoittamisen ja tätä kautta kustomoidun vapaan elementin luomisen, mikäli osaamista ja halua on (Kuva 19)

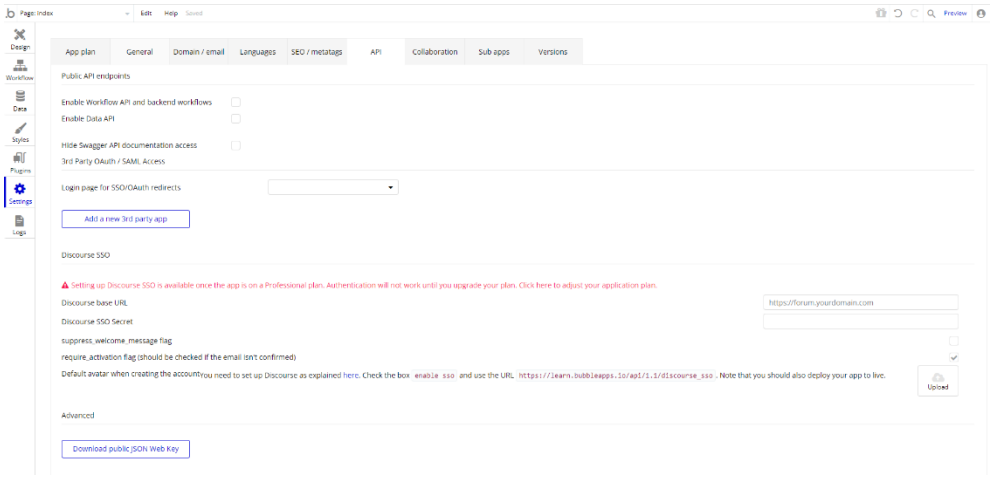
Kuva 19 HTML -elementti bubble:ssa.



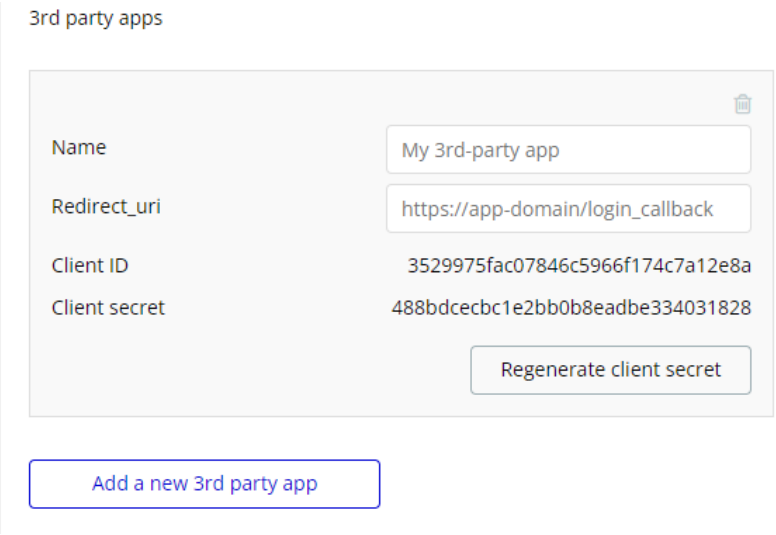
4.2.3 API-rajapintaan liittyminen Bubble:ssa

API-rajapintaan liittyminen tapahtuu Bubble:ssa ”Settings” välilehdeltä. API-rajapinta voidaan helposti liittää sovellukseen ja sen päälle- ja poiskytkentä tapahtuu yksinkertaisesti, joko laittamalla täpän kohtaa ”Enable Data API” tai ottamalla sen pois (Kuva 20). Sovellukseen voi liittää myös kolmannen osapuolen applikaatioita, joten mikäli yrityksellä olisi käytössä jo muita vastaavia sovelluksia, jotka haluttaisiin liittää yhteen samalle alustalle tai ottaa uuden sovelluksen kanssa käyttöön onnistuisi se tällä alustalla ”Add a new 3rd party app” -kohdasta ja syöttämällä sovelluksen nimi ja uri (Kuva 21). Settings-välilehden kohdasta ”General” voidaan tarkastella jo valmiiksi liitettyjä API-rajapintojen avaimia, tässä kohdassa käyttäjälle kerrotaan myös hyvin, että API-avaimet ovat välttämättömiä kun sovellusta aletaan pyörittämään omassa domainissa (Kuva 22). Bubblessa on mahdollista integroida myös ulkoisiin SQL-kantoihin. Näitä ovat Postgres, MySQL ja Microsoft SQL.

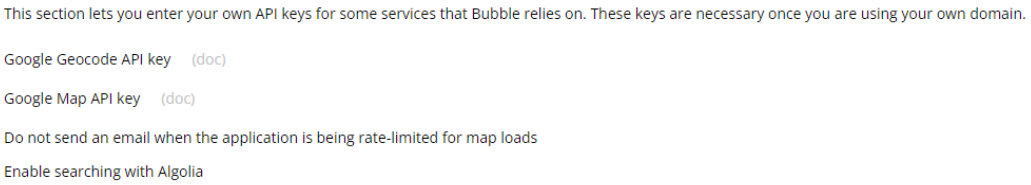
Kuva 20 API-rajapintoihin liittyminen ”Settings”-välilehdeltä.



Kuva 21 Kolmannen osapuolen applikaation lisääminen osaksi uutta sovellusta.



Kuva 22 API-rajapintojen määrittäminen General-välilehdellä asetuksissa.



4.2.4 Bubblen ominaisuudet

Bubble lupaa sivuillaan, että alustalla on mahdollista luoda esimerkiksi täysin twitteriä tai facebookkia vastaava sovellus, joten lähtökohtaisesti ominaisuuksia löytyy todella kattavasti. Sovellusten rakentaminen tapahtuu ”Drag and drop”-systeemillä, jossa valmiita elementtejä asetellaan tyhjälle pohjalle näin luoden suoraa graafinen käyttöliittymä. Alustalla on myös mahdollista luoda monikielisiä sovelluksia, sillä kaikki tekstit on mahdollista kääntää halutuille kielille nopeasti ja vaivattomasti.

Alustalla on mahdollista myös tarkkailla mitkä ominaisuudet ovat käytetyimpiä sovelluksen käyttäjien osalta, tämä antaa tärkeää dataa kun sovellusta lähdetään kehittämän parempaan suuntaan, koska muutokset saadaan pohjattua oikealle datalle. Tätä analytiikkaa pystytään siirtämään myös kolmannen osapuolen tahoille, sillä bubble on mahdollsita integroida

esimerkiksi Mixpanelin ja Google analyticsin kanssa. Myös applikaation käyttämää CPU:ta voidaan tarkkailla realiajassa.

Alustalla on mahdollista luoda kustomoituja sähköposteja valmiiksi määritetyillä pohjilla ja mikäli sovelluksessa tarvitaan maksuliikennettä on sovellus integroitavissa myös Stripe- ja Braintree -alustoihin, joilla voidaan luoda muunmuassa kuukausimaksujen maksaminen mahdolliseksi suoraa sovelluksessa, sillä kolmannen osapuolen maksuliikenteiden integroiminen on mahdollistettu.

Versionhallinta on tehty helpoksi ja testipuolen muutokset on mahdollista viedä tuotantoon yhden klikkauksen avulla. Kaikista muutoksista jää muistijälki alustan välimuistiin ja aiempien versioiden palauttaminen on mahdollista koska vain.

Bubblessa on mahdollista luoda omia Plug-in-lisäosia ja luoda omia elementtejä käyttämällä HTML, CSS ja Javascript -ohjelmointikieliä.

4.2.5 Bubble:n skaalautuvuus

Bublella on mahdollista luoda projekteja, joissa voi saman aikaisesti työskennellä jopa 40 käyttäjää. Käyttäjien oikeuksia voi muokata suoraa asetuksista. Projektit on mahdollista myös salasana-suojata, joka on iso tietoturvaan liittyvä hyöty.

Alustalla palvelimen tehokyky skaalautuu sitä mukaa kun applikaatio sitä tarvitsee ja testiapplikaatiosta siirtyminen julkaistuun oikeaan applikaation tapahtuu kivuttomasti tekemättä teknisiä muutoksia lainkaan. Alustalla maksetaan ainoastaan tarvittavista resursseista ja tarkka seuranta mahdollistaa sen, että yritys tai kehittäjä ei maksa muuta kuin käyttämästään resurssin määrästä. Tämä resurssi voi olla esimerkiksi CPU:n käyttöaste. Testiapplikaatiosta taasen ei vielä tarvitse maksaa mitään, joka vie alkukustannukset minimiin.

4.2.6 Bubble:n tietoturva

Bubblen sivuilta käy ilmi, että se on rakennettu Amazonin AWS:n pohjalle, joka antaa sille muun muassa SOC 2 CSA ja ISO 27001 -sertifikaattien hyväksynnän. Alustaa itseään testataan automatisoidulla kooditestauksella ja jatkuvalla monitoroinnilla. Kaikki käytetty data on mahdollista palauttaa ”Point-in-time” -palautuksen avulla. Aiemmin esitetyllä bubble:n käyttöliittymällä on myös ”logs”-välilehti, eli alustalla on mahdollista seurata ja hallita lokeja vaivattomasti, eli kehittäjät ja yritykset voivat vapaasti seurata applikaation lokeja mikä lisää läpinäkyvyyttä ja mahdollistaa tietoturvan ja bugien analysoinnin. Sovelluksessa on mahdollista ottaa käyttöön myös kaksivaiheinen tunnistautuminen, joka lisää tietoturvaa rutkasti.

Kaikkea dataa, mitä bubblessa käytetään ja liikutetaan suojataan RDS EAS-256-kryptauksella. Tämä on Amazonin kehittämä kryptausmuoto, jossa data kryptataan ennen kuin se viedään tietokantaan ja dekryptataan ennen kuin se luetaan sieltä. Kaikki data on myös suojattu yleisen kuluttajakäytännön mukaan.

5 Vaatimusmäärittely sovellukselle

Kun yritys harkitsee sovelluksen käyttöönottoa tai siirtämistä uudelle alustalle on tärkeää luoda vaatimusmäärittely sovellukselle. Tämä toimenpide auttaa myös päätöksentekoa alustan suhteen sekä tarjoaa avaimet päätökseen annetaanko sovelluksen toteutus ulkoisen tahon haltuun vai luodaanko sovellus mahdollisesti yrityksen itsensä toimesta. Jukka Paakki (Ohjelmistojen vaatimusmäärittely, Syksy 2011, Helsingin Yliopisto) on määrittänyt vaatimusmäärittelyn presentaatiossaan, että täytyy määritellä, muotoilla ja analysoida kolme tärkeää nyanssia:

- Mikä on ratkaistava ongelma?
- Miksi ongelma pitää ratkaista?
- Kenen vastuulla on ongelman ratkaisu?

5.1 Sovelluksen tarkoitus

Yrityksen tulisi kartoittaa tarkkaan mihin sovellusta tullaan käyttämään, eli mikä sen tarkoitus on ja mitkä ovat prosessit tai työnvaiheet mitkä sillä pystytään automatisoimaan tai parantamaan. Tämä tulee vastaamaan aiemmin esitettyyn kysymykseen, siitä mikä on ratkaistava ongelma? Eli sovellus tulee olemaan ratkaisu tai osaratkaisu tämän ongelman korjaamiseen. Eri työnvaiheet kannattaa dokumentoida mahdollisimman tarkkaan ja samalla myös kirjata vaiheet mitkä esimerkiksi tehdään manuaalisesti. Tärkeää on myös miettiä mille loppukäyttäjille sovellus tulee käyttöön ja kuinka esimerkiksi teknologia-orientoituneita he ovat, tämä on nyanssi joka kannattaa ennenkaikkea käyttöliittymän suunnittelussa ottaa huomioon. Suositeltavaa on tehdä tutkimusta tulevien loppupäänkäyttäjiltä ja kartoittaa heidän tarpeitaan ja toiveitaan sovelluksen suhteen.

Mikäli yrityksillä on käytössään jo muita sovelluksia kannattaa niidenkin käyttötarkoitukset ja hyödyt ottaa huomioon ja mahdollisesti miettiä olisiko nämä integroitavissa tai korvattavissa tulevalla uudella sovelluksella.

5.2 Resursointi

Sovelluksen kehittäminen vaatii resursseja koko elinkaarensa ajan. Yrityksen kannattaa miettiä tarkkaan kuinka paljon se esimerkiksi tarvitsee kehittäjiä kehittämis- ja suunnitteluvaiheeseen. Tämä aspekti on otettava myös alusta valitsemisessa huomioon, sillä samanaikaisten käyttäjien määrä per projekti on alustoissa rajoittava tekijä. Aiemmin esitettyyn kysymykseen siitä, kuka on taho jonka ongelma tulee ratkaista kannattaa miettiä resursointia. On suositeltavaa, että nämä tahot sidotaan resursseihin, sillä heillä on paikkaansapitävin näkökulma ongelmaan tai prosessiin jota halutaan parantaa.

Kun siirrytään kehittämisvaiheesta tuotantoon, vaati sovellus ylläpitoa. Yrityksen kannattaa kartoittaa kuinka paljon sovelluksen ylläpito vaatii rahaa ja työntekijöitä, tässä huomioon on otettava myös sovelluksen ylläpitoon kuuluvat kuukausimaksut alustan palveluntarjoajan osalta. Testausta ja ylläpitoa voidaan ostaa ulkoisilta tahoilta ja tämäkin kannattaa ottaa huomioon, mutta mikäli alusta tarjoaa valmiit ja helpot tavat testaukseen ja ylläpitoon kannattaa yrityksen harkita näiden hoitamista sisältäpäin. Budjetointi kannattaa tehdä

alustavasti jo etukäteen, sillä alustan valinta helpottuu kun vaihtoehdot saadaan supistettua esimerkiksi kuukausimaksun mukaan.

5.3 Datan varastointi ja liikuttaminen

Vaatimusmäärittelyä tehdessä yrityksen tulisi miettiä ja dokumentoida sovelluksessa käytettävä data ja hahmotella suunnitelmaa miten ja mihin data varastoidaan ja mihin formaattiin, sekä listata tähän erilaisia vaihtoehtoja siltä varalta, että alustoissa datan varastointiin on olemassa eri vaihtoehtoja. Tulee myös miettiä onko data säilössä kolmannen osapuolen tietokannoissa vai tuleeko tietokannat mahdollisesti jollekkin yrityksen omalle on-prem tai pilvipalvelimelle. Mikäli näin onko se mahdollisesti integroitavissa alustalle, jota aiotaan sovelluksen kehittämiseen käyttää?

Tietoturvan osalta data tulee säilyttää turvassa yleisten käytäntöjen mukaan. Vaikka ohjelmointi tulee tapahtumaan valmiilla alustalla, on kehittäjän vastuulla vastata datan turvallisesta liikuttelusta komponenttien välillä työnkuluissa ja varastoinnissa.

5.4 Sovelluksen toiminnallisuudet

Sovellukselta vaaditut toiminnallisuudet kannattaa kasata listaksi. Tällaisia toiminnallisuuksia ja ominaisuuksia ovat muun muassa käyttöliittymän rakenne (sivupohjat, elementtien paikat yms.), toiminnallisuudet (sähköpostien mallipohjat, lomakkeet kyselyt, laskurit, kaaviot yms.), mahdolliset integraatiot (esimerkiksi ulkoisiin tai sisäisiin API-rajapintoihin tarvittavat yhteydet). Nämä mainitut nyanssit on valitusta projektimallista riippuen helppo jakaa omiksi tehtävikseen kehitystiimille tai kehittäjälle. Graafisen ohjelmoinnin alustojen sisältäessä mahdollisuudet ulkoasun ja toiminnallisuuksien nopeaan käyttöönottoon ja testaamiseen on näiden tehtävien toteutus dynaamista.

Toiminnallisuuksien listaaminen auttaa myös alustan valinnassa sekä budjetin suunnittelussa. Sillä mitä aikaisemmin tiedetään, mitä sovellukselta vaaditaan sitä helpompi on alustojen hintoja punnita ja sen pohjalta rakentaa budjetti.

5.5 Projektimallit ja graafinen ohjelmointi

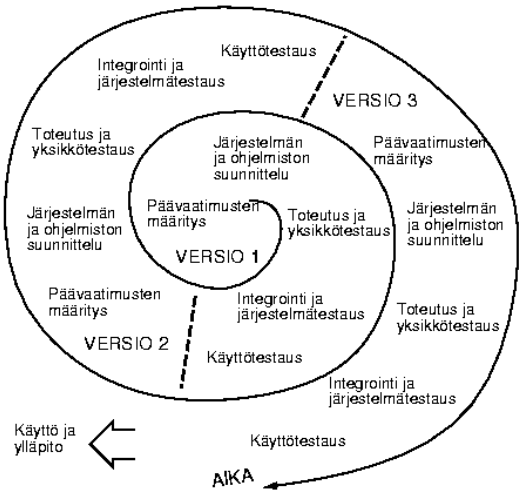
Kehittämisvaiheessa yrityksen suositellaan valitsemaan tuotantoprosessi, miten sovellus tullaan luomaan. Perinteisestä ohjelmistokehityksestä poiketen graafisen ohjelmoinni alustat muokkaavat toimintamalleja, sillä esimerkiksi testausta voidaan harjoittaa ketterästi kehitysvaiheessa valmiiden testaustyökalujen avulla. Seuraavaksi tullaan käymään läpi kolme perinteistä ohjelmistotuotannon projektimallia sekä sitä miten ne taipuvat ohjelmistokehitykseen graafisen ohjelmoinnin alustoilla.

5.5.1 Spiraalimalli

Yksi perinteisistä ohjelmistokehityksen projektimalleista on spiraalimalli (Kuva 22). Spiraalimallissa toistetaan neljää eri vaihetta joita toistetaan kerta toisensa jälkeen samassa järjestyksessä. Nämä vaiheet ovat vaatimusten määrittely, suunnittelu, toteutus ja testaus. Graafisen ohjelmoinnin alustoilla testaus ja versionhallinta on tehty dynaamiseksi, joka tekee tästä projektimallista liian kankean taipuvaksi ohjelmistokehitykseen, kun ohjelmaa luodaan graafisen ohjelmoinnin alustalla.

Eri versioita voidaan testata jo kehittäessä ja resursseja menisi turhaan hukkaan, kun testaus jätettäisiin aina jokaisen sprintin viimeiseksi vaiheeksi. Mallin saisi toimimaan muokkaamalla sitä niin, että testaus tapahtuisi samanaikaisesti kehitystyön kanssa ja uudelle spiraalin kehälle lähdettäisiin aina, kun sovellus todettaisiin vaatimuksia vastaavaksi.

Kuva 22 Spiraalimalli



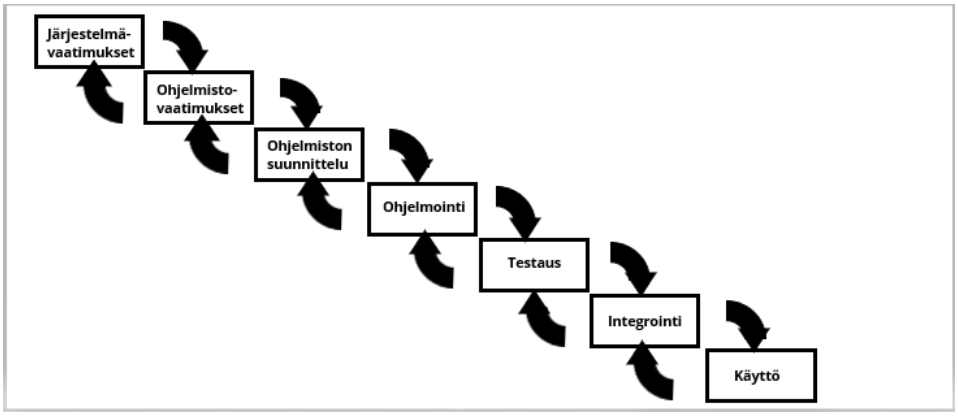
5.5.2 Vesiputousmalli

Vesiputousmalli on yksi suosituimmista ohjelmisokehityksen projektimalleista (Kuva 23). Vesiputousmallisa projekti aloitetaan ensimmäiseksi listaamalla järjestelmän vaatimukset, mistä siirrytään ohjelmiston vaatimuksiin ja ohjelmiston määrittelyyn. Seuraava työvaihe on ohjelmiston suunnittelu jota seuraa varsinainen ohjelmointi. Viimeiset vaiheet ovat testaus ja käyttö.

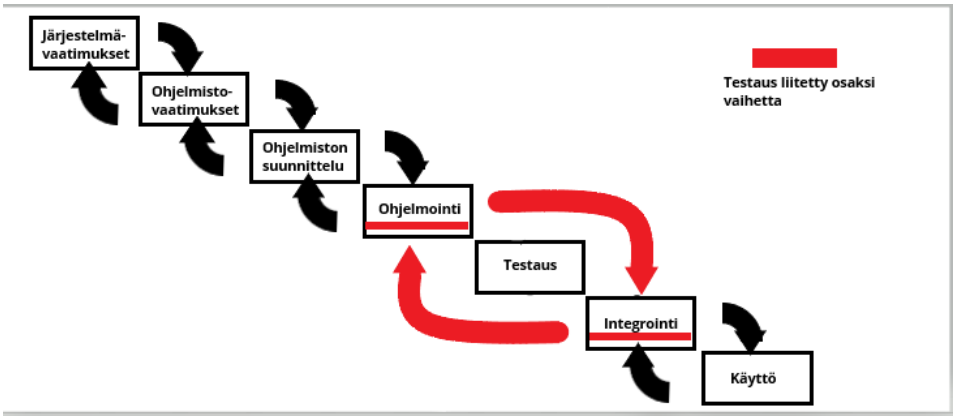
Tämä projektimalli ei vaadi käytettäviä tekniikoita, mikä tekee siitä mukautuvan kaikenlaisiin projekteihin. Graafisen ohjelmoinnin alustat sopivat tällaiseen malliin, sillä valmiit komponentit rajoittavat räätälöintiä, mutta kun projektimalli ei määrittele näitä viitekehyksiä, voidaan ohjelmiston kehittäminen mahdollistaa tämän osalta vesiputousmallilla.

Vesiputousmallissa testaus jää viimeiseksi vaiheeksi ja mikäli vajavaisuutta ominaisuuksissa tai bugeja löytyy, lähdetään uudelle kierrokselle. Vesiputousmallin kohtien mukaan jää sovelluksen käyttöönotto pahimmillaan todella myöhäiseksi. Tästä voidaan tehdä johtopäätös, että graafisen ohjelmoinnin alustalla projektia aloittaessa ei vesiputousmalli sellaisenaan taivu dynaamisimmaksi projektimallivaihtoehdoksi. Alustat kuitenkin mahdollistavat jatkuvan testaamisen kehittämisen ohessa, joten mikäli testaus liitetään osaksi toteutusta ja integrointia, on tämä projektimalli yksi parhaimmista vaihtoehdoista malliksi (Kuva 24).

Kuva 23 Vesiputousmalli



Kuva 24 Vesiputousmalli muokattuna sopivaksi graafisen ohjelmoinnin käyttöliittymällä toteutettavaan projektiin.



5.5.3 Agile

Agile on yksi ketterien menetelmien projektimalleista. Siinä koko projektia ei yritetä suunnitella ennakkoon ja malli on rakennettu niin, että muutoksia ja parannuksia on mahdollista tehdä kehittämisen ohella. Projekti jaetaan ikään kuin pieniin miniprojekteihin, joissa tapahtuu suunnittelua, kehittämistä ja testausta tässä järjestyksessä jatkuvalla tahdilla. Tämä mahdollistaa nopeiden muutosten ja korjausten tekemisen ja graafisen ohjelmoinnin alustoista suurella osalla on olemassa dynaaminen versionhallinta mikä soveltuisi loistavasti agile-projektimalliin.

Koska graafisen ohjelmoinnin alustat omaavat valmiita työnkulun rakentamiseen tarvittavia ominaisuuksia sekä valmiita elementtejä on se osaltaan rajoittava tekijä, mutta mahdollistaa nopean testauksen sekä sprinteille toteutettavien taskien luomisen vähäisen varsinaisen ”koodaamisen” osalta sopisi agile-projektimalliksi sellaisenaan yrityksen käyttöön.

6 Tulokset

Testaamani alustat olivat lähtökohtaisesti aika samanlaisia ja näitä vertaillessa, voidaan havaita selkeä pääero jo alustojen merkityksessä. Low-code-alustat ovat luotu jo ohjelmoinnin parissa työskennelleiden kehittäjien työn nopeuttamiseen, kun taas no-code alustat kehittäjille, joilta ohjelmointitaito uupuu.

Mikäli yritys päätyy kehittämään sovellusta sisäisesti ja oma kehittäjätiimi tälle löytyy, on enemmän kuin suositeltavaa valita alusta, jolla on mahdollista tehdä koodimuutoksia. Testaamani Retool tarjosi tähän erittäin loistavan käyttöliittymän ja koodin muokkaaminen komponenteissa ja tyuönkuluissa tapahtui helposti siirtymättä uuteen välilehteen. No-code-alustan opetteleminen vaatisi kehittäjiltä uuden opettelua, mikä itsessään hidastaisi prosessia huomattavasti.

No-code-alustan käyttöä suosittelen mille tahansa yritykselle, jonka tarkoituksena on digitalisoida työnkulkuja tai ottaa käyttöön applikaatioita työn jouhevoittamiseksi. Tässä Bubble tarjoaa jo laajan kirjaston valmiita applikaatioita, jotka step by step -menetelmällä luodaan ja, jotka ovat räätälöitävissä jälkikäteen todella kattavasti. Esittelemäni bubble tarjoaa hyvin tietoa kokenemattomallekkin kehittäjälle esimerkiksi API-rajapintoihin liittymisestä ja miten tämä käytännössä tehdään.

Graafisen ohjelmoinnin alustat ylipäätään tarjoavat kilpailukykyisen tavan luoda applikaatioita ja työnkulkuja yritysten käyttöön sitomatta pääomaa tai resursseja ihan yhtä paljon, kuin perinteinen ohjelmointi. Työn jälki näkyy konkreettisesti nopeammin ja jo pelkästään projektimalleista on graafisen ohjelmoinnin ansiosta muokattavissa dynaamisempia ja nopeampaa lopputulosta kiihdyttäviä.

Jatkuva ja nopea testaaminen sekä muutoksien tuotantoon saattaminen tarjoavat myös käyttäjäystävällisempien sovellusten tuottamisen, sillä esimerkiksi Bubblesta löytyi analytiikkatyökalu käyttäjädatan keräämiseen ja analysointiin. Eli kaikki muutokset, mitä

sovellukseen tullaan tekemään voidaan pohjata täysin käyttökokemusten perusteella tutkittuun tietoon. Projektimallit ylipäätään eivät sellaisenaan sovi graafisella ohjelmoinnilla toteutettaviin projekteihin, mutta yritykselle voi ylipäätään olla huono näkemys ohjelmistokehityksen projektimalleista, jolloin projektimallin joutuu kuitenkin opettelemaan ja tämä työ tarjosi mahdolliset vaihtoehtoiset tavat toimia projektimalleja käyttäessä.

Alustoja on markkinoilla satoja erilaisia, tällä kertaa työhön valikoidut sovellukset tarjoavat molemmat todella luotettavat ja ominaisuuksiltaan kattavat alustat, joiden kaikkiin ominaisuuksiin pureutuminen ei olisi työtä välttämättä palvellut. Tarkoituksena oli näyttää mistä graafisessa ohjelmoinnissa on kyse ja miksi sitä kannattaa harkita vaihtoehtona ohjelmistokehitykseen.

Etenkin bubble nousi alustana omaksi suosikikseni sen kattavalla dokumentoinnilla ja käyttäjäystävällisellä käyttöliittymällä. Myös alustan suuri tarjonta valmiiden sovelluspohjien luontiin saa minulta suuret pisteet ja mikäli yritys tarvitsee yksinkertaisia sovelluksia käyttöönsä voi nämä toteuttaa valmiiden ja kuukausihintaan sisältyvien ohjeiden perusteella.

7 Yhteenveto

Tutkimuskysymyksiin saatiin kattavasti vastaukset. Graafisen ohjelmoinnin alle lukeutuvat no-code- sekä low-code -alustat avattiin kattavasti ja näistä käytiin läpi historiaa, tietoturvaa ja skaalautuvuutta. Johtopäätöksen voi vetää, että edelleen no-code-alustat ovat enemmän muille kuin teknologiayrityksille optimaalisempia valintoja. Low-code-alustat puolestaan tarjoavat yhä enemmän työkaluja ohjelmointia työkseen tekevien ohjelmoinnin nopeuttamiseen. Esimerkiksi Retoolissa on mietitty elementtien muokkaamista ja koko alustan käyttöliittymää niin, että kehittäjällä on mahdollisimman hyvät puitteet luoda ohjelmaa. Projektimallit valittiin käytetyimmistä ohjelmistokehityksessä käytetyistä malleista ja näiden käyttöä pohdittiin graafisen ohjelmoinnin osalta. Lisäksi tarjottiin vaihtoehtoisia tapoja käyttää näitä malleja niin, että yritys tai kehittäjä voi esiteltyjä tapoja käyttää sellaisenaan projekteissaan.

Käytännönosasta voidaan vetää johtopäätös, että aluksi on luotava kattava suunnittelu ja vaatimismäärittely, jotta voidaan ylipäätään valita ominaisuuksiltaan ja tarpeita vastaava alusta. Tässä teknistä osaamista tarvitaan ja loppukäyttäjiltä tulee kerätä tietoa työnkulusta. Mikäli yrityksen sisältä ei löydy tarpeeksi teknologiaoritentoitunutta osaajaa, on suositeltavaa sovelluksen käyttötarkoituksen onnistumisen osalta hankkia konsultointia ulkoiselta taholta.

Lähteet

Amy Groden-Morrison, What To kKnow About Low Code/ no Code Platforms & Security. Haettu 14.1.2022 osoitteesta

https://www.alphasoftware.com/blog/what-you-need-to-know-about-low-code/no-code-platforms-and-security

Bubble,Bubble vs. other tools, 2022. Haettu 26.1.2021 osoitteesta

https://bubble.io/faq

Bubble, Capabilities, 2022. Haettu 26.1.2021 osoitteesta

https://bubble.io/faq

Bubble, Learning Bubble, 2022. Haettu 26.1.2021 osoitteesta

https://bubble.io/faq

Bubble, Integrations, 2022. Haettu 26.1.2021 osoitteesta

https://bubble.io/faq

Bubble, Scaling, 2022. Haettu 26.1.2021 osoitteesta

https://bubble.io/faq

Christopher Tozzi, The evolution of Low-Code/No-code Development, 2021. Haettu 24.1.2021 osoitteesta

https://www.itprotoday.com/no-codelow-code/evolution-low-codeno-code-development

Creamailer, GDPR eli EU:n tietosuojauudistus – asiakkaiden ja palveluntarjoajien vastuut, 2017. Haettu 25.1.2021 osoitteesta

https://www.creamailer.fi/blogi/eun-tietosuojauudistus-asiakkaiden-ja-palveluntarjoajien-toimet-ja-vastuut

Digia, Low-code-selvitys: Rutiinit Digityön Haasteina Suomalaisorganisaatioissa, 2021. Haettu 22.12.2021 osoitteesta

https://resources.digia.com/low-code-selvitys

Innominds, How Secure and scalable are Low-code Platforms, 2021. Haettu 20.1.2021 osoitteesta

https://www.innominds.com/blog/how-secure-and-scalable-are-low-code-platforms

Jukka Paakki, Ohjelmistojen vaatimusmäärittely, 2011. Haettu 7.2.2022 osoitteesta

https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-4.pdf

Matias Madou, Low code, no code: innovation or security catastrophe?, 2021. Haettu 22.12.2021 osoitteesta

https://thestack.technology/low-code-no-code-security-risks/

Michael Bargury, Addressing the Low-Code Security, 2018. Haettu 22.12.2021 osoitteesta

https://www.darkreading.com/edge-articles/addressing-the-low-code-security-elephant-in-the-room

Miika Kallasoja, Yritysjohdonkin olisi syytä kiinnostua low-code/no-code-alustoista, 2021 Haettu 23.12.2021 osoitteesta

https://www.sofigate.com/insight/lowcode-nocode-alustoilla-sovelluskehityksesta-nopeampaa/

P. Koikkalainen, Ohjelmistotekniikan prosessimallit, 2000. Haettu 7.2.2022 osoitteesta

http://www.mit.jyu.fi/ope/kurssit/TIE358/sivusto/johdanto/ohjelmistoprosessi.html

Retool, What is Retool?, 2022. Haettu 14.1.2021 osoitteesta

https://docs.retool.com/docs

SVCIT Editorial, Is No-Code Platform scalable enough for enterprise?, 2020. Haettu 24.1.2021 osoitteesta

https://www.siliconvalleycloudit.com/is-no-code-scalable-enough-for-enterprise/

Tigersheet, Low Code vs No Code. What's the Difference?, 2021. Haettu 24.1.2021 osoitteesta

https://tigersheet.com/blog/low-code-vs-no-code-whats-the-difference-infographic/