Jingwang Jiang

Erilaisten no-code-alustojen vertailu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

24.5.2024

Tiivistelmä

Tekijä:	Jingwang Jiang
Otsikko:	Erilaisten no-code-alustojen vertailu
Sivumäärä:	29 sivua
Aika:	24.5.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaajat:	Lehtori Toni Spännäri

Tämän opinnäytetyön tarkoituksena on selvittää, millainen no-code-kehitys on. Työssä tutustutaan sekä moderniin kehitykseen että no-code-kehitykseen ja niiden vaikutuksiin.

Työn teoriaosassa tutustutaan ensin modernin web-kehityksen perustaan ja siihen liittyviin työkaluihin, esimerkiksi kirjastoihin ja sovelluskehyksiin. Lopussa kerrotaan web-hostauksesta ja sen toteuttamisen prosessista ja keinoista.

Modernin web-kehityksen jälkeen on no-code-kehitys-luku, jossa kerrotaan no-code-kehityksestä ja sen tarpeen alkuperästä ja tulevaisuudesta.

Tutkimusosan tavoitteena on kokeilla ja vertailla eri no-code-alustoja, jotta tämän tutkimuksen tekijä ja lukijat saisivat kuvaa ja kokemusta ohjelmistokehityksestä.

Työssä on käytetty kolmea eri no-code-alustaa. Ne olivat AppSheet, Bubble ja monday.com. Tässä työssä täytyy luoda kolme versiota projektin seurantasovelluksesta käyttämällä näitä alustoja. Ohjelmistokehityksessä on esiintynyt ongelmia ja haasteita. Sovellusten toteutukset toimivat kuitenkin hyvin.

Kun sovellukset olivat valmiita, vertailtiin 3 alustaa ja annettiin arviointi niistä alustoista. Tutkimus toteutui suunnitelman mukaisesti. Työssä luotiin kolme erilaista projektin seurantasovellusta, joita testattiin ja vertailtiin. Tarkat tutkimustulokset esitetään luvussa 5.

Avainsanat: No-code, ohjelmointi, projektinseurantasovellus

Abstract

Author:	Jingwang Jiang
Title:	Comparison of different no-code platforms
Number of Pages:	29 pages
Date:	24 April 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technologies
Professional Major:	Media Technology
Supervisors:	Toni Spännäri, Senior Lecturer

The purpose of this thesis is to investigate what no-code development is. The thesis explores both modern development and no-code development and their effects.

The theoretical part of the thesis begins with an introduction to modern web development fundamentals and related tools, such as libraries and frameworks. It concludes with a discussion on web hosting and its implementation process and methods.

Following modern web development is a chapter of no-code development, detailing its origins, needs, and future.

The aim of the research section is to experiment with and compare different no-code platforms, providing the author of this research and its readers with an understanding and experience of software development.

Three different no-code platforms were used in this research: AppSheet, Bubble and monday.com. In this thesis, three versions of a project tracking application must be created using these platforms. While software development encountered some challenges and issues, the development of the applications still achieved the goal.

At the end, a comparison of the three platforms was conducted, and evaluations were provided for each. The research was conducted as planned. Three different project tracking applications were created, tested and compared. Detailed research findings are presented in Chapter 5.

Keywords: No-code, development, project tracking application

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoituksena on tutustua erilaisiin web-kehitystapoihin. Sovelluskehitys ja sen historia saattavat olla mielenkiintoista asiaa joillekin käyttäjille, tässä työssä kerrotaan sen kehitysprosessista.

Opinnäytetyön alussa kerrotaan selkeästi ja ytimekkäästi teorian pohjalta, mikä on moderni web-kehitys ja mitkä ovat siihen liittyvät teknologiat, kuten kirjastot ja ohjelmistokehykset. Työssä yritetään antaa lukijalle selkeä kuva modernista web-kehityksestä ja sen historiasta. Web-kehitys ei ole pelkästään koodin kirjoittamista, vaan se muodostuu monesta prosessista. Opinnäytetyössä kerrotaan sitten näistä prosesseista ja käytetyistä työkaluista. Tämän jälkeen selvitetään, miten ja miksi no-code-kehityksen tarve on syntynyt ja no-coden tulevaisuusnäkymiä. Onko se hyvä vai huono työkaluna, työn keskiosassa käsitellään tätä asiaa. Lisäksi työssä pohditaan ja vertaillaan no-code-kehityksen käytettävyyttä, yhteensopivuutta sekä sen hyviä ja huonoja puolia verrattuna moderniin kehitykseen.

Opinnäytetyön lopussa tutkitaan ja vertaillaan eri no-code-alustoja. Tutkimusta varten työssä luodaan kolme versiota projektin seuranta -sovelluksista käyttämällä kolmea eri no-code-alustaa. Työssä valitaan ja tutustutaan ensin kolmeen no-code-alustaan (AppSheet, Bubble ja monday.com) ja luodaan niiden avulla uusi sovellus. Lisäksi vertaillaan ja arvioidaan kolmen eri alustan vahvuutta ja heikkoutta. Tutkimuksen prosessista kerrotaan tarkemmin luvuissa 4 ja 5.

2 Moderni web-kehitys

Tässä luvussa kerrotaan web-kehityksessä olevia keskeisiä asioita.

World Wide Web eli web on englantilaisen tietojenkäsittelytieteilijä Tim Berners-Leen kehittämä hypertekstijärjestelmä, joka on julkaistu vuonna 1989 (Mozilla 2023).

Responsiivinen suunnittelu on yksi tärkeimmistä asioista modernissa web-kehityksessä, koska käyttäjillä on nykyään erilaisia laitteita esim. tabletti, puhelin, pöytäkone ja kannettava tietokone. Web-sovelluksen sopivuus kuuluu myös kehittäjän tehtävään. (Scott 2016.) Lähes kaikki suuret sivustot ovat responsiivisia nykyisenä aikana.

Responsiivinen suunnittelu on Ethan Marcotten keksimä termi, joka on keksitty vuonna 2010. Ennen responsiivista suunnittelua kehittäjän piti luoda eri kokoisia sivustoja. Sivusto tarkistaa ensi käyttäjän näytön koon ja näyttää sitten käyttäjälle sopivan sivuston version. Responsiivisen suunnittelun avulla kehittäjä tarvitsee luoda vain yksi kokoinen sivusto, ja se mukautuu automaattisesti käyttäjän näytön kokoon, suurentuen tai pienentyen tarpeen mukaan. (Romano 2022.)

HTML, CSS ja JavaScript ovat web-ohjelmoinnin aloittelijan kolme ensimmäistä opittavaa kieltä, jotka toimivat yhdessä muodostaen modernin web-kehityksen perustan. Alaluvussa 2.1 kerrotaan tarkemmin näistä kolmesta kielestä. Web-kehitys on tullut entistäkin tärkeämmäksi osaksi ihmisten arkea, sillä erilaiset verkkosivustot ja sovellukset ovat tärkeä osa ihmisten päivittäistä toimintaa. Niiden tarve on myös kasvanut paljon. Web-kehitys vaatii nykyään enemmän ohjelmointitaitoja kuin koskaan aikaisemmin.

Monet kehittäjät ja yritykset ovat kehittäneet erilaisia web-kehityksen työkaluja, kuten kirjastoja ja ohjelmistokehyksiä, joiden avulla kehittäjät voivat nopeuttaa ja helpottaa ohjelmointiprosessia. Myös monenlaiset alustat ovat saatavilla. Käyttäjä voi luoda alustan avulla oman web-sovelluksen jopa ilman

ohjelmointitaitoja. Tätä tekoa kutsutaan no-code developmentiksi eli ohjelmoinniksi ilman koodia. No-codella luotu web-sovellus sopii yksityiskäyttöön ja -tarpeisiin, mutta monimutkaisen sovelluksen ja sivuston kehitys vaatii kuitenkin perinteistä ohjelmointia ja syviä ohjelmointitaitoja.

2.1 Web-kehityksen perusta HTML, CSS ja Javascript

HTML (HyperText Markup Language) on merkintäkieli, jota käytetään web-sivujen rakentamiseen. HTML-koodilla määritellään sivujen rakenne ja sisältö, kuten otsikot, kappaleet, kuvat, linkit ja muut elementit (Mozilla 2023). HTML on perusta web-kehityksessä ja se on tärkeä perustaito web-kehittäjälle. Esimerkkikoodi 1 esittää perus-HTML-koodia.

```
<!DOCTYPE html>
     <html>
          <head>
               <title>Home</title>
          </head>
          <body>
               <h1>Hello World<h1>
          </body>
     </html>
```

Esimerkkikoodi 1. Perus HTML-koodi, joka tulostaa Hello World sivustolle.

CSS (Cascading style sheets) on toinen tärkeä ja osattava kieli web-kehityksessä. CSS-koodilla määritellään sivujen ulkoasu, kuten fontit, värit ja taustat (Mozilla 2023). CSS:n avulla voidaan luoda responsiivisia sivustoja, jotka sopivat erikokoisille näytöille.

```
body {
     background-color: black;
     }
     h1 {
     color: white;
     text-align: center; }
```

Esimerkkikoodi 2. CSS-koodi, jonka avulla taustaväri on musta, ja valkoinen h1-teksti pysyy keskellä.

JavaScript on dynaaminen ohjelmointikieli, jota käytetään web-kehityksessä. JavaScriptin avulla kehittäjä voi toteuttaa monimutkaisia toimintoja

verkkosivustoissa ja sovelluksissa, kuten tietojen käsittelyä ja muokkaamista, AJAX-tietojen lataamista ja interaktiivisen käyttöliittymän luomista. Dynaaminen ohjelmointikieli tarkoittaa sitä, että koodin muuttujien tyypit ja arvot voivat muuttua ohjelman suorituksen aikana (Mozilla 2024).

Javascript on julkaistu vuonna 1995, ja sen entinen nimi oli Mocha. Alun perin sen kehittäjän tarkoituksena oli kehittää HTML:lle skriptikieli, jolla sivusto toimii paremmin ja nopeammin selaimella. JavaScript on muuttanut merkittävästi käyttäjien netin selailun kokemusta. Ennen Javascriptin julkaisemista sivuston koodit olivat pääasiassa staattisia, ja käyttäjien täytyi itse päivittää sivua nähdäkseen sivuston muutos. JavaScript pystyy verifioida käyttäjän lähettämää dataa selaimella ennen lähettämistä palvelimelle, mikä on auttanut kehittäjiä säästämään palvelimen tilaa (Frisbie 2023). JavaScript on jatkuvasti kehittyvä kieli, ja siksi sillä on vielä muita mahdollisuuksia tulevaisuudessa.

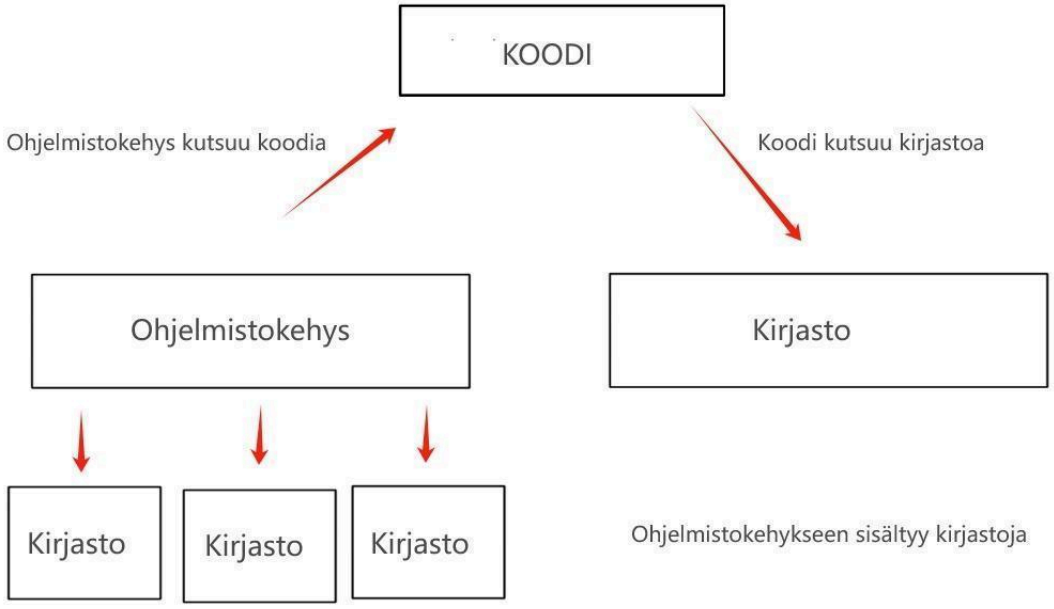
2.2 Kirjastojen ja ohjelmistokehysten käyttö

Kirjastot ja ohjelmistokehykset ovat suosittuja web-kehityksen työkaluja, joiden avulla kehittäjät voivat toteuttaa monimutkaisia projekteja nopeammin ja tehokkaammin (Sandip Roy 2022.).

Kirjastot sisältävät valmiita ja testattuja aliohjelmia, luokkia ja kokoelmia. Ne voivat vähentää kehittäjän työmäärää ja parantaa koodin laadukkuutta, koska kehittäjä ei tarvitse kirjoittaa koodia uudelleen. Framework eli ohjelmistokehys on samankaltainen työkalu kuin kirjasto, mutta ohjelmistokehyksellä on enemmän käyttörajoituksia, koska käyttäjät eivät voi muokata ohjelmistokehyksen koodia, mutta saavat laajentaa sitä.

Ohjelmistokehyksillä ja kirjastoilla on omia vahvuuksia ja heikkouksia web-kehityksessä. Kun ohjelmistokehystä käytetään kehityksessä, kehittäjä saa helposti kehittää oman sovelluksen, mutta ohjelmistokehyksen vaihtaminen jälkikäteen on vaikea.

Kirjastot ovat joustavampia kuin ohjelmistokehykset, mikä vaatii enemmän tarkkaavaisuutta ja ymmärrystä ohjelmoinnista. Kuvassa 1 esitetään suhteen, koodin, ohjelmistokehysten ja kirjastojen välillä.



Kuva 1. Yksinkertainen kuva, joka havainnollistaa ohjelmistokehysten ja kirjastojen suhdetta koodiin.

Esimerkkejä suosituista kirjastoista ja ohjelmistokehyksistä: React, Angular, Vue.js, jQuery, ja Express.js. React on suosituin kirjasto, joka on julkaistu 29.5.2013. React on Metan kehittämä vapaa ja avoimen lähdekoodin ohjelmisto (Deshpande 2024). Metan kehittäjillä oli suuri projekti, jossa käytetään JavaScriptMVC:ta, mutta he eivät olleet tyytyväisiä markkinoilla oleviin JavaScript-ohjelmistokehyksiin. Tämän vuoksi he kehittivät sitten oman kirjaston eli Reactin, joka oli entistä tehokkaampi ja sujuvampi käyttää. Kyseinen suuri projekti oli Instagram. (Deshpande 2024.)

Vue.js on avoimen lähdekoodin JavaScript-ohjelmistokehys, joka on erittäin suosittu GitHubissa. Vue.jsin on alkuperäisin Angularista, ja sen kehittäjä oli Angularin entinen työntekijä Evan You. Youn mukaan hän oli valinnut Angularista ne osat, joista hän piti, ja rakensi niiden pohjalta Vue.js:n. (Cromwell

2016.) Sekä Reactin että Vue.js:n tavoitteena on edistää käyttäjän kokemusta ja rakentaa ohjelmisto selkeämmin ja kätevämmin.

2.3 Muut liitännäiset (sivujen hostaus, pilvipalvelut, backend)

Web-sivujen hostaus on tärkeä osa web-kehityksessä, ja netti on sama kuin tori ja hostaus on sama kuin myyntipaikan vuokraaminen. Kehittäjän täytyy vuokrata itselleen myyntipaikka, jotta oma tuote on esillä torissa eli netissä. Perinteistä sivuhostausta on ostaa domain ja palvelin tarjoajalta. Kehittäjä lataa sitten toimivat tiedostot palvelimelle, minkä jälkeen muut ihmiset voivat selata verkkosivua selaimella. (HP 2023.)

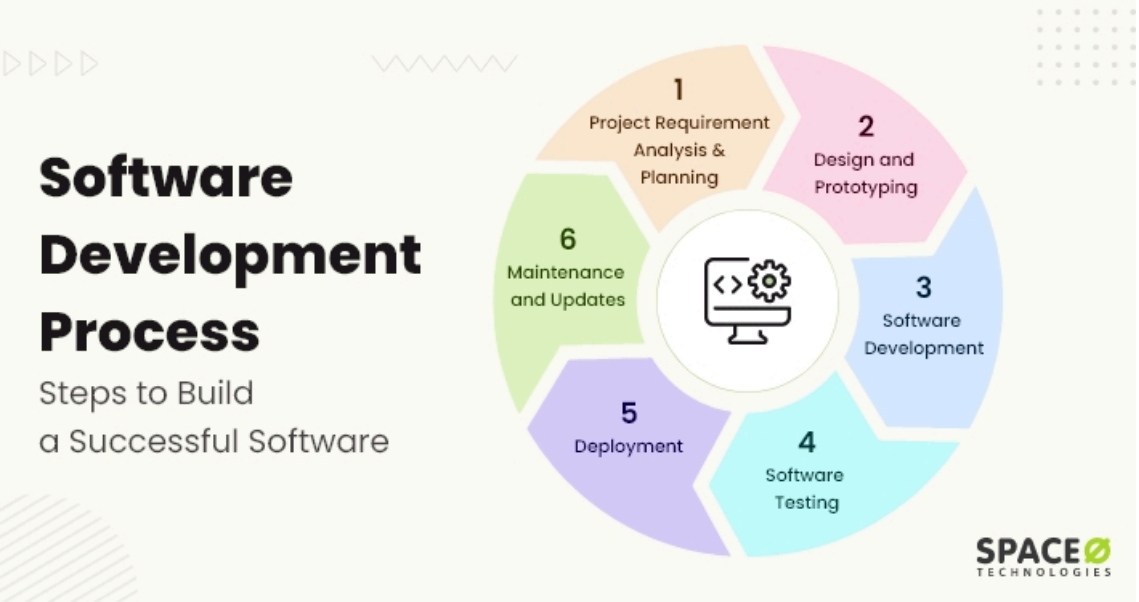
Domain on osoite ja palvelin on paikka. Domain on helppo ostaa, mutta palvelimen ostossa ostajan täytyy olla tarkka ja valita huolellisesti, jotta hän saa kohtuulliseen hintaan sopivan palvelun. Pilvipalvelu on suosittu tapa hostata sivun. Sivujen hostaus pilvipalvelulla on edullisempaa ja turvallisempaa kuin perinteisellä palvelimella, koska pilvipalvelun maksu riippuu käyttäjämäärästä ja kehittäjät eivät tarvitse huolehtia tietoturva-asiasta sekä varmuuskopiosta, ne kuuluvat palveluntarjoajan tehtäviin (Google 2024). Esimerkkejä suosituista pilvipalvelun tarjoajista ovat Amazon Web Services, Microsoft Azure ja Google Cloud.

Backend on web-kehityksen palvelimen puoli, jonka sisältö ei ole suoraan näkyvissä käyttäjille. Backend käsittelee erilaisia asioita, kuten tietokantoja, sovelluksen logiikkaa, tiedonhallintaa ja autentikointia (GeeksforGeeks 2023). Backendin vastakohta on frontend, joka on selainpuoli. Frontendin tehtävät keskittyvät käyttöliittymän suunnitteluun ja käyttäjäkokemukseen.

3 No-code-kehitys

No-code-kehitys eli kooditon kehitys on nopea ja edullinen sovelluskehitys. No-code-kehitys on yksinkertainen, sillä kehittäjä tarvitsee vain suunnitella oma sovellus. Kun suunnittelu on valmis, kehittäjä käyttää jotakin no-code-alustaa esimerkiksi WordPress-alusta tarjoaa muutamia vaihtoehtoja käyttäjille, käyttäjä valitsee haluamansa ominaisuudet ja asettaa ne paikalleen, ja näin sovellus on jo valmis.

Perinteisessä kehityksessä on yleensä 6 vaihetta. Ne ovat sovelluksen suunnittelu, muotoilu ja prototyyppi, koodin kirjoittaminen, testaus, käyttöönotto, ylläpito ja päivitys. Koodin kirjoittaminen ja testaus vie kehittäjältä eniten aikaa, mutta no-code-kehityksessä kehittäjä voisi ohittaa koodin kirjoittamisen (Bhaval Patel 2024). Kuvassa 2 on kuvattu perinteisen ohjelmistokehityksen vaiheet.



Kuva 2. Perinteisessä ohjelmistokehityksessä esiintyvät vaiheet (medium 2023).

3.1 No-code-tarpeen synty

Internetin käyttö on yleistynyt huomattavasti viimeisten kymmenen vuoden aikana. ITU:n tilastojen mukaan vuoden 2012 Internetin käyttäjien määrä oli 2,4 miljardia ja vuoden 2022 käyttäjien määrä oli 5,3 miljardia eli käyttäjien määrä on kaksinkertaistunut kymmenessä vuodessa(FiCom 2022). Käyttäjien määrän kasvu tarkoittaa myös internetin käytön tarpeen kasvamista, kun perinteinen web-kehitys ei enää riitä käyttäjien tarpeeseen, ja no-code -kehitys on syntynyt.

Low-code- ja no-code-alustat ovat myös lisänneet kehittäjien joustavuutta, koska eri projekti saattaa vaatia eri ohjelmointikieltä ja taitoa, eikä kehittäjä välttämättä osata kaikkea. Low-code ja no-code-kehitystä hyödynnetään merkittävästi tässä työssä. (IBM 2024.)

3.2 Käytettävyys, yhteensopivuus, laatu ja kustannus

Kun käyttäjät valitsevat no-code-kehityksen, sen käytettävyys, yhteensopivuus, laatu ja kustannus tulevat huomioon otetuksi.

Käytettävyys on ensimmäinen asia, jota otetaan huomioon. No-code-kehityksen tarkoitus on helpottaa ohjelmiston kehittämistä ilman syvällistä koodaustaitoa. Tämän takia no-code-kehityksen työkalua eli alustaa on yleensä helppo käyttää. Internetissä on yli 30 erilaista alustaa tarjolla, ja niiden käyttötapa on lähes sama. Alusta tarjoaa visuaalisia käyttöliittymiä ja helppokäyttöisiä työkaluja, kuten lohkojen ja komponenttien vetämistä ja pudottamista, jotta käyttäjät voivat nähdä heti sovelluksen muutosta tehtäessä. Käytettävyys voi kuitenkin vaihdella alustojen välillä, joten on tärkeä valita alusta, joka sopii käyttäjien taitoihin ja tarpeisiin(Kiguolis 2023).

No-code-alustat tarjoavat usein valmiita integraatioita ja liittymiä muihin järjestelmiin ja palveluihin (esim. tietokantoihin ja pilvipalveluihin). Tämä helpottaa sovelluksen yhteensopivuutta muiden järjestelmien kanssa. (OutSystems 2024.)

No-code-kehityksen laatu voi vaihdella riippuen alustasta ja sovelluksen vaatimuksista. No-code-kehitys voi olla hyvä vaihtoehto yksinkertaisiin ja pienimuotoisiin sovelluksiin. Monimutkaisissa sovellusten kehittämisessä voi olla haasteita, koska no-code-työkalut eivät välttämättä pysty täysin tukemaan niitä (Quixy 2023).

Kustannus voi usein olla syy siihen, miksi käyttäjät valitsevat no-code-kehityksen. No-code-kehitys säästää sekä aikaa että kustannuksia. Kehittäjä ei tarvitse kirjoittaa koodia alusta loppuun. Vaikka eri alustoilla on erilainen hinnoittelu, mutta alustojen peruskäyttö on yleensä maksutonta. (Quixy 2023.)

3.3 Tietoturvallisuus

No-code-kehityksen tietoturvallisuus on tärkeä näkökohta, kun käyttäjä harkitsee no-code-alustan käyttöä. Tietoturvallisuus voi vaihdella eri alustojen välillä. Tietoturvallisuuteen yleisesti vaikuttavat seikat ovat alustan turvallisuus, tietosuoja, haavoittuvuus, kolmansien osapuolien integraatiot, testaus ja virheenhallinta. Sopivan ja luotettavan alustan valinta on tärkeää, jotta vältetään turvaton alusta. Turvallinen alusta tarjoaa asianmukaiset suojausmekanismit, kuten käyttäjien tunnistamisen, salatun tiedonsiirron ja tietojen suojaamisen.

Tietosuoja on herkkä aihe sekä kehittäjille että käyttäjille. Alustat, jotka noudattavat tietosuojasääntelyä kuten GDPR (General Data Protection Regulation) ovat tärkeitä. Käyttäjän kannattaa tarkistaa, että alusta noudattaa GDPR-sopimusta (Fleming 2022). Sovelluksen haavoittuvuus on väistämätön riski sen käytön aikana. Alustan säännöllinen päivitys on paras tapa välttää haavoittuvuutta. Kun no-code-kehityksessä käytetään kolmansien osapuolien integraatioita kuten tietokantoja ja pilvipalveluita, on tärkeää tarkistaa, että nämä palvelut tarjoavat asianmukaiset tietoturvaominaisuudet, jotta integraatiot tapahtuvat turvallisesti ja suojattujen yhteyksien kautta. Alustan tarjoama debug-ominaisuutta on suositeltavaa käyttää säännöllisesti, jotta sovellus toimii tietoturvallisesti.

3.4 Ero no-coden, low-coden ja full-coden välillä

No-coden, low-coden ja full-coden ero näkyy suoraan jo sanoissa.
No-code-kehitys eli kooditon kehitys tarkoittaa ohjelmistokehitystä ilman koodin kirjoittamista. Tämä tarkoittaa myös sitä, että kehittäjät voivat luoda sovelluksia käyttämällä alustan visuaalisia käyttöliittymiä. No-code-alustat tarjoavat valmiita lohkoja ja komponentteja, jotka voidaan yhdistää toisiinsa käyttöliittymän avulla. No-coden helppous tuo myös rajoituksia kehitykseen, koska tietyt monimutkaisemmat toiminnot saattavat olla vaikeampia tai jopa mahdottomia toteuttaa ilman koodin kirjoittamista. (OutSystems 2024.)

Low-code on ohjelmistokehityksen lähestymistapa, jossa pyritään luomaan koko sovellus vähemmällä koodin kirjoittamisella. Low-coden ja no-coden tarkoitus on samanlainen, mutta koodin avulla low-codessa on enemmän joustavuutta kuin no-codella. (OutSystems 2024)
Full-code on perinteinen ohjelmistokehitysmuoto, jossa sovelluksia kehitetään kirjoittamalla koodia alusta loppuun. Kehityksessä käytetään erilaisia ohjelmointikieliä, kuten Java, Python, JavaScript ja C++.

No-coden ja low-coden hyvä puoli on se, että ne ovat nopeita ja helppoja. Niiden huono puoli on rajoitettu joustavuus. Jos kehittäjällä on riittävä ohjelmointitaito ja aikaa, täysin koodin kirjoittaminen (full-code) on paras vaihtoehto, koska kehittäjä voi itse valita, mitä ja miten sovellusta kehitetään. (OutSystems 2024.)

3.5 Tulevaisuudennäkymät

No-code-kehityksen tulevaisuudennäkymät näyttävät lupaavilta, ja no-code-kehityksen suosion odotetaan jatkavan kasvuaan ohjelmistokehityksen alalla kasvavan tarpeen vuoksi.

No-code-alustat mahdollistavat ihmiselle pääsyn ohjelmistokehitykseen. Ohjelmointitaidon puute ei ole enää este, kun käytettävissä on visuaalisia ja intuitiivisia työkaluja, joiden avulla ihmiset voivat kehittää sovelluksia helposti ja

nopeasti. Uusia no-code-työkaluja, kirjastoja ja ohjelmistokehyksiä syntyy jatkuvasti, ja niiden avulla no-code-kehittäjät voivat laajentaa taitoaan ja kehittää monimutkaisempia sovelluksia (Kochar 2023). No-code-työkaluja voi nyt käyttää myös muuhun kuin sovelluksen kehittämiseen, kuten data-analysointiin, chatbottien luomiseen ja koneoppimisen hyödyntämiseen. No-code-kehityksellä on siis monipuolinen tulevaisuus.

Tekoälyn kehitys on parantanut ja helpottanut no-code-kehityksen käyttöönottoa, koska se voi opastaa ihmiset käyttämään no-code-alustaa paremmin. Ihmiset eivät tarvitse enää lukea pitkää käyttöohjetta ennen käyttöönottoa, koska tekoäly voi vastata kysymyksiin.

Ei voi varmuudella sanoa, että no-code korvaa kokonaan full-coden tulevaisuudessa, mutta se korvaa full-coden jo nyt pienissä projekteissa ja yksinkertaisissa sovelluskehityksissä.

4 Sovelluksen toteutus no-code-alustoilla

Tässä luvussa kerrotaan sovelluksen toteutuksesta kolmella eri no-code-alustalla.

Projektin alussa valitaan ensin kolme no-code-alustaa ja luodaan kolme eri versiota projektiseurantasovelluksesta käyttämällä näitä alustoja. Projektinseurantasovellus on helppokäyttöinen ja tuttu sovellus kaikille, joten se on sopiva valinta. Tarkoituksena on luoda sovellus, johon käyttäjä voi lisätä uuden projektin ja muokata olemassa olevia projekteja. Jos kehittäjä pystyy, kehittäjä lisää vielä käyttäjän todennuksen ja muistiinpano-toiminnallisuuden sovellukseen.

Kolme alustaa oli vaikea valita, koska vaihtoehtoja oli paljon. Loppujen lopuksi AppSheet, Bubble ja monday.com tulivat kolmeksi valituksi no-code-alustaksi. WordPress oli kolmantena alustana vaihtoehtona, koska se on suosittu ja tunnettu no-code-alusta, mutta Wordpress sopii enemmän sivuston luontiin kuin muuhun. Lisäksi sovelluksen toteuttaminen WordPressin avulla edellyttää laajennuksen eli lisäosan asentamista, mikä puolestaan vaatii palvelun hankkimista tai oman verkkotunnuksen. Tämän vuoksi no-code-alusta monday.com valittiin Wordpressin tilalle.

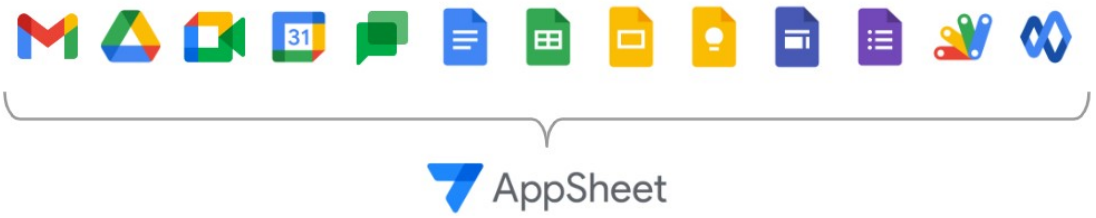
4.1 AppSheet

AppSheet on Google Cloudin tarjoama no-code-alusta, joka tuo käyttäjälle mahdollisuuden luoda erilaisia sovelluksia ilman ohjelmointitaitoja.

Työssä valittiin AppSheet tutkimusalustaksi, koska se on hyvin tunnettu ja luotettava no-code-alusta ja sen emoyhtiö on Google.

AppSheet tarjoaa perusominaisuuksia kuten sovelluksen luomisen maksutta. Lisäksi se tarjoaa myös erilaisia maksullisia palveluita kuten automaatiota, asiakastukea, entisten sovellusten hallintaa ja turvallisuustukea. Maksullisen version palveluhinta riippuu asiakkaan tarpeesta.

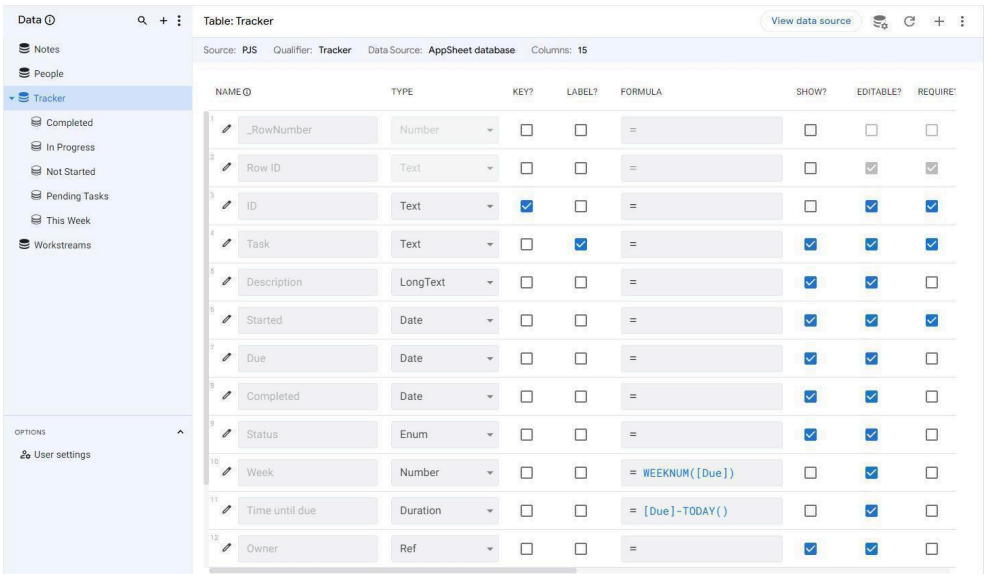
AppSheet on yhteensopiva monien Google-palveluiden kanssa, kuten Google Sheets, Excel, Dropbox, Google Maps ja Google Workspace. Tämä tekee siitä monipuolisen ja joustavan ratkaisun sovelluskehityksessä. Kuvassa 3 on erilaisia Googlen tarjoamia palveluita.



Kuva 3. Googlen Workspacen palvelut, joita voi käyttää yhdessä AppSheetin kanssa.

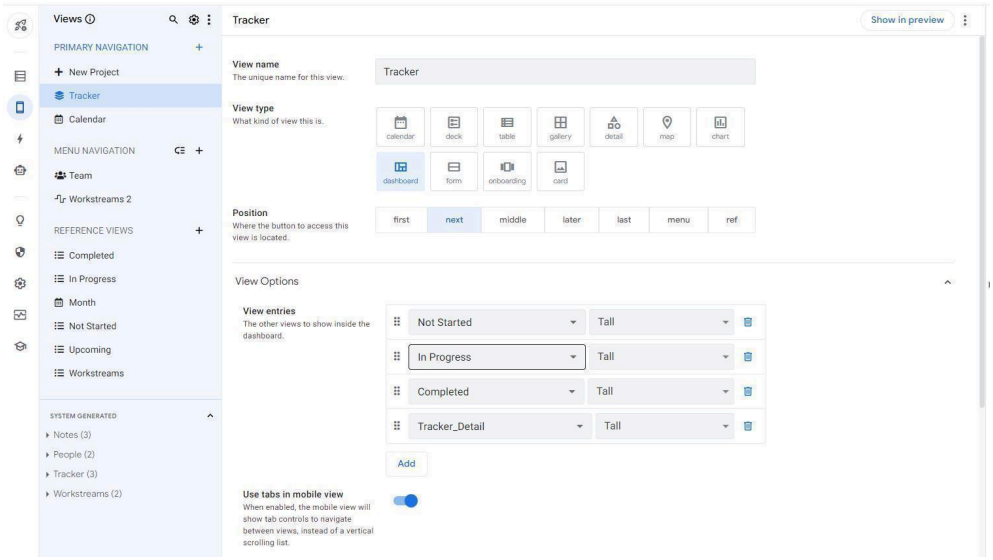
Sovelluksen luominen AppSheetillä on täysin kooditonta, mutta tämä ei tarkoita että se olisi helppoa. Työssä luotiin ensimmäinen versio projektinseurantasovelluksesta. Tämän tutkimuksen tekijän oman kokemuksen ja tuntemuksen mukaan AppSheetilla sovelluksen luominen on hieman samanlaista kuin modernia ohjelmointia. Jos käyttäjällä on koodaustaitoa tai kokemusta, se voi olla hyödyksi AppSheetiä käyttäessä.

Projektinseurantasovelluksen no-code-versio tarvitsee myös oma tietokanta, koska siihen syötetään ja tallennetaan tietoja. Tämän vuoksi luotiin ensin tietokanta sovellukseen. AppSheetillä on oma tietokantapalvelu, joten alustaa ei tarvitse yhdistää ulkopuoliseen tietokantaan. Kuvassa 4 on kuvakaappaus sovelluksen tietokannasta.



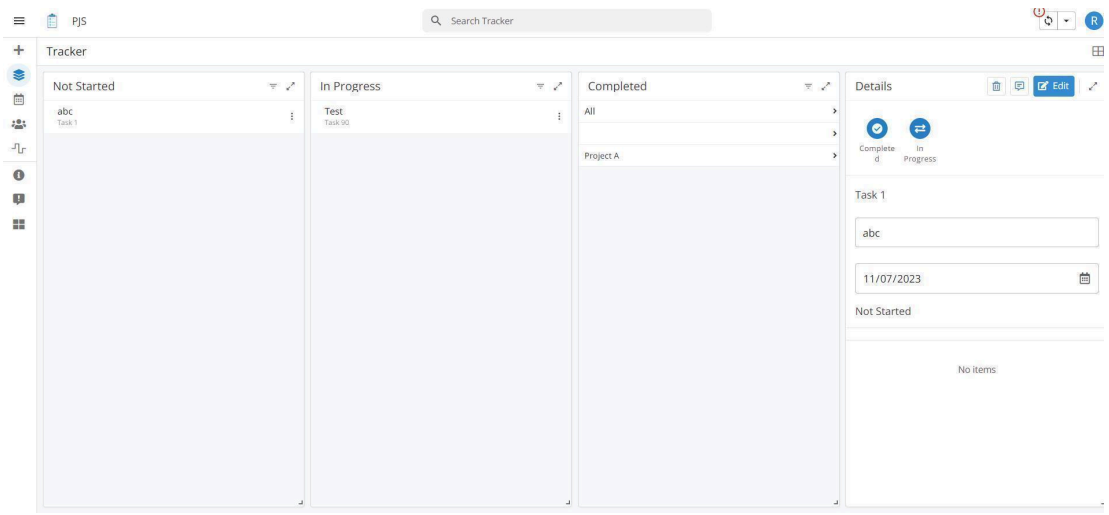
Kuva 4. Kuva sovelluksen tietokannasta.

Kun tietokannan osuus oli valmis, työssä ryhdyttiin luomaan käyttöliittymä, mutta tämän tutkimuksen tekijä ei pystynyt luoda käyttöliittymä oman tavoitteen mukaisesti. Tämä on siis no-code-alustan rajoitus. Hyvä puoli oli se, että AppSheet tarjoaa käyttäjälle muutamaa vaihtoehtoa jokaiseen komponenttiin. Kuvassa 5 esitetään AppShetin alustan käyttöliittymän asettelua.



Kuva 5. AppSheetin komponenttien vaihtoehdot.

Kun molemmat osuudet olivat valmiita, lopputuote on nimeltään PJS eli projektiseuranta. Kuvassa 6 on kuvakaappaus PJS:n käyttöliittymästä.



Kuva 6. PJS:n käyttöliittymä.

Kehityksen alkuosa eli tietokannan luominen oli kehittäjälle vaikea, koska hänen piti miettiä, millaisia tietoja syötetään ja tallennetaan sovellukseen sekä mitkä tiedot olivat pakollisia.

Käyttöliittymän luomisessa ei ollut haasteita. Tässä vaiheessa oli helppo muokata ja vaihtaa komponentteja.

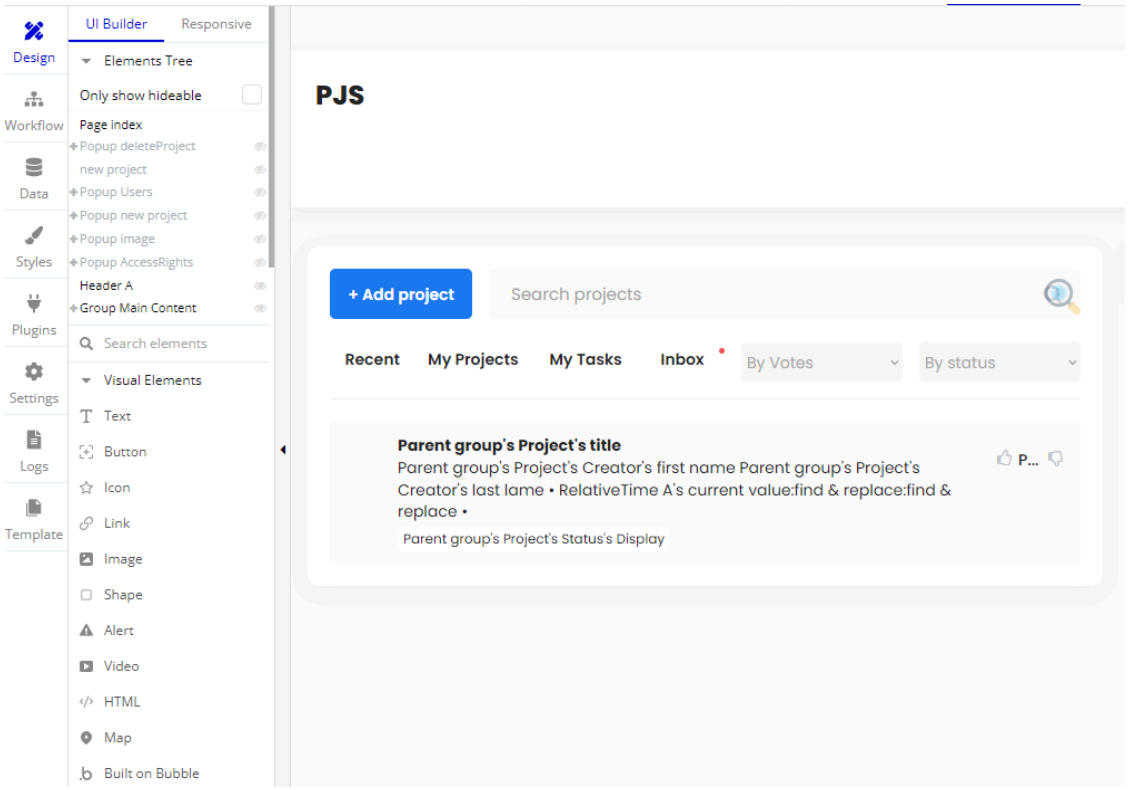
Sovellus kehitettiin AppSheetin verkkosivustolla, eikä käyttäjä tarvinnut asentaa mitään ohjelmistoa tietokoneelle. Sovellus toimii mobiililaitteella, tabletilla ja tietokoneella.

4.2 Bubble

Bubble on visuaalinen web-sovelluskehitysalusta, joka on julkaistu vuonna 2012. Bubble tarjoaa sekä no-code- että low-code-kehitystapoja; käyttäjä saa valita jommankumman Bubblesta käyttötarpeensa mukaan. Bubble on ehkä vaikeampi käyttää kuin AppSheet, mutta kaikkien sovellusten rakentaminen onnistuu myös Bubblessa ilman ohjelmointitaitoja.

Bubble on tunnettu ja suosittu alusta no-code-alustan käyttäjien keskuudessa, koska se tuo käyttäjälle erilaisen kokemuksen kuin monet muut alustat.

Bubblen käyttö on hieman samanlaista kuin palapelin ratkaisemista ja Photoshopin käyttöä. Käyttäjä saa valita tarvittavan komponentin ja vetää sen haluamaansa paikkaan. Komponentin kokoa, väriä ja sisältöä voi muokata vapaasti ja joustavasti. Tämä on ehkä syy siihen, miksi Bubble on niin suosittu, koska se on erittäin joustava käyttää, ja käyttäjä voi nähdä suoraan, mitä hän on tekemässä. Kuvassa 7 esitetään Bubblen käyttöliittymän ja PJS:n toisen version prototyyppiä.

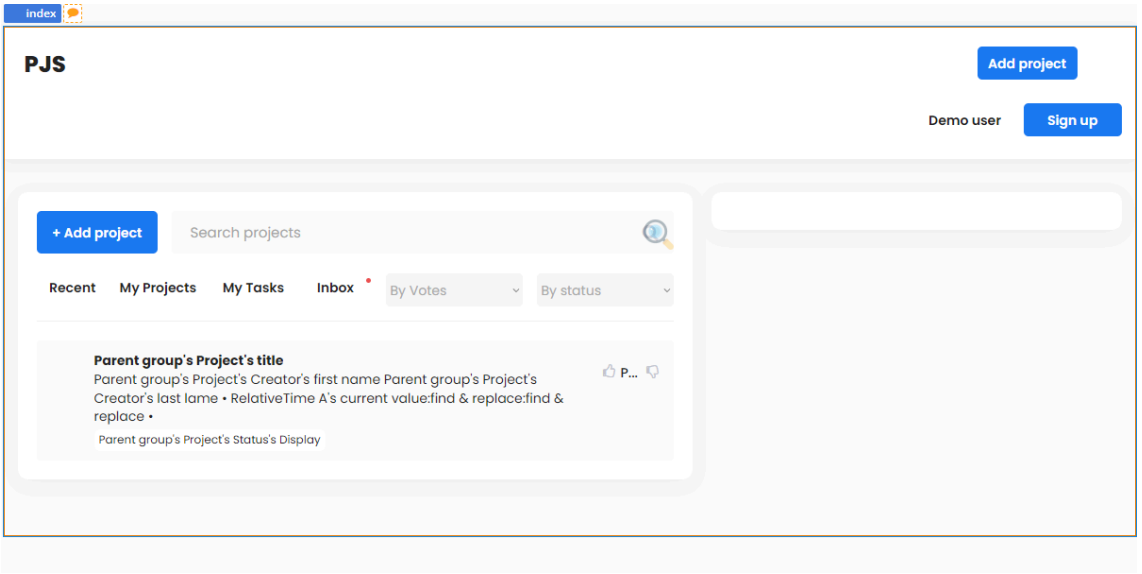


Kuva 7. No-code-alusta Bubblen käyttöliittymä.

Bubblen peruskäyttö on maksutonta, ja se tarjoaa myös maksullisia palveluja, esim. oman domainin, monen kehittäjän käytön (ilmaisversiossa saa olla vain yksi kehittäjä) ja sovellusten seurannan. Palvelun hinta riippuu asiakkaan tarpeesta.

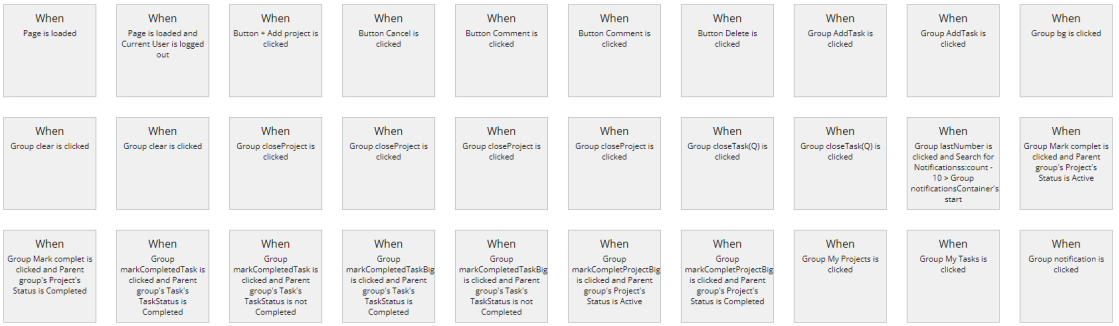
Projektinseurantasovellus oli vaikea toteuttaa Bubblella, vaikka tämän tutkimuksen tekijä oli jo no-code-ohjelmistokehittämiskokemus AppSheetista. Bubblen käyttötapa oli kuitenkin ihan erilainen kuin AppSheetin.

Tämän tutkimuksen tekijä suunnitteli ja loi ensin sovelluksen käyttöliittymän. Tämä vaihe oli helppo, koska se ei vaatinut vielä mitään vaikeaa suunnittelua, kuten tietokannan luomista. Kuvassa 8 on PJS:n toisen version käyttöliittymä.



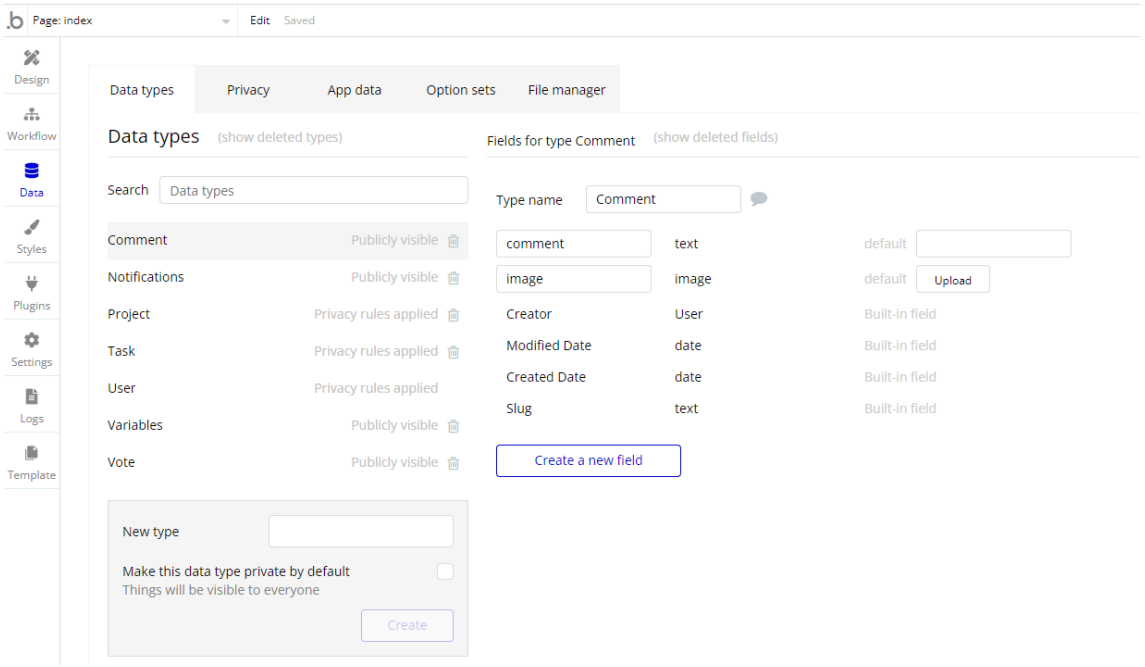
Kuva 8. Käyttöliittymäprototyyppi sovelluksen Bubble-versiosta.

Kun käyttöliittymä oli valmis, työhön luotiin painikkeen ominaisuudet, kuten mitä tapahtuu, kun painiketta napsautetaan. Tämä osa vaatii vähän logiikkaa ja kärsivällisyyttä. Tämän tutkimuksen tekijä on testannut monta kertaa, sen että painikkeet toimivat kunnolla. Kuvassa 9 esitetään erilaisia painikkeen toimintojen logiikoita.



Kuva 9. Erilaiset painokytkimet ja niiden toiminnallisuudet.

Kuvassa 10 näkyy sovelluksen tietokanta. Koska Bubblen käyttö on niin joustavaa, tämän tutkimuksen tekijän piti itse luoda tietokanta, ja sen valinta oli enemmän kuin AppSheetissa. Joustavuuden huono puoli on se että, käyttöä on vaikea osata ja hyvä puoli on monipuolisuus.



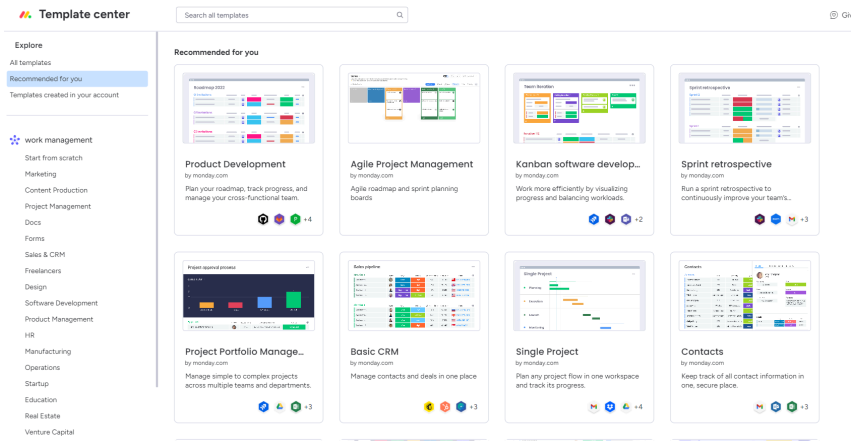
Kuva 10. Kuvakaappaus Bubble-version sovelluksen tietokannasta.

4.3 Monday.com

Monday.com on no-code- ja low-code alusta, joka on julkaistu vuonna 2014. Monday.comin entinen nimi oli dapulse, mutta koska käyttäjät eivät ymmärtäneet, mitä se tarkoittaa ja se oli vaikea muistaa, dapulsen tilalle tuli uusi nimi eli monday.com. Alustan käyttö oli todella helppoa, koska alustalla on erilaisia sovellusmalleja tarjolla. Käyttäjä tarvitsee valita itselleen sopiva malli ja luoda sen päälle oma sovelluksensa.

Monday.com tarjoaa sekä maksutonta että maksullista palvelua. Ilmaisversiossa on useita esteitä kuten kalenterin käytön ja sovelluksen kehittäjämäärän osalta. Maksullinen palvelu tarjoaa neljää hintatasoa; ensimmäisestä kolmanteen hinta on kiinteä ja neljännen hinta riippuu käyttäjän tarpeesta. Monday.comia käyttävät enemmän yritykset kuin yksilöt, koska maksullinen palveluosto vaatii vähintään kolmea jäsenyyttä.

Projektinseurantasovelluksen toteutukseen monday.comilla ei käytetty paljon aikaa. Aluksi luotiin ensin oma työtila alustan verkkosivustolla. Sen jälkeen napsautettiiin Add from templates' -painiketta, ja lopuksi valittiin sopiva malli. Kuvassa 11 on kuvakaappaus monday.comin työtilasta.

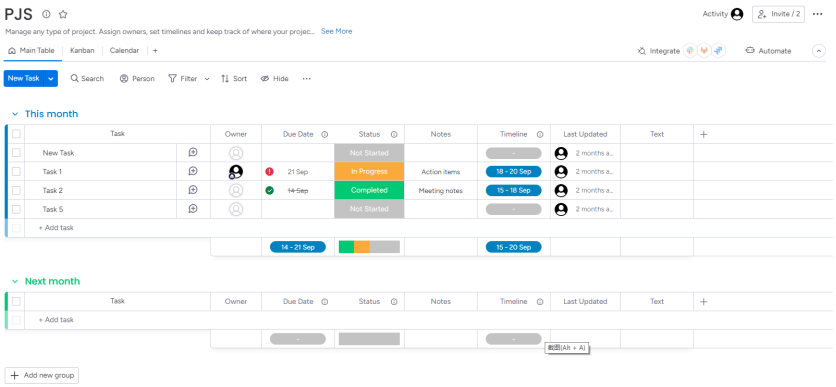


Kuva 11. Kuva monday.comin työtilasta.

Alusta loi automaattisesti valmiin mallin tämän tutkimuksen tekijälle, hän tarvitsi vain tehdä sovelluksen viimeistely, esim. komponenttien lisääminen, poisto, nimen muokkaus ja värinvaihto.

Lopputulos oli helppo ja yksinkertainen projektinseurantasovellus. Sovelluksen kehityksessä ei ollut haastetta eikä vaikeutta, koska tämän tutkimuksen tekijä ei tarvinnut luoda tietokantaa.

Monday.comin yksinkertaisuus oli ehkä yksi syy siihen, miksi se menestyi no-code-alustan käyttäjien keskuudessa. Kuvassa 12 on monday.com-version käyttöliittymä.

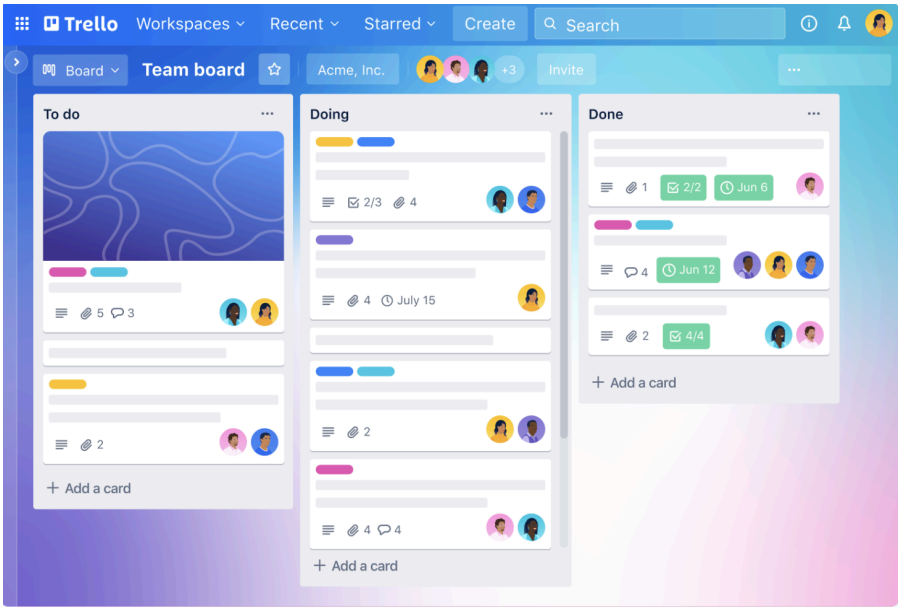


Kuva 12. Sovelluksen käyttöliittymä monday.com-versiosta.

5 Erot eri alustoilla luodun sovelluksen välillä

Työssä käytettiin kolmea eri no-code-alustaa, ne olivat AppSheet, Bubble ja monday.com.

Työn tarkoituksena oli luoda kolme eri versiota projektinseurantasovelluksesta, jossa käyttäjä saa sovelluksen kautta lisätä, poistaa ja seurata projektejaan. Sovelluksen piti olla monelle ihmiselle tuttu ja helppokäyttöinen, jotta käyttäjä pystyy erottamaan, mikä ero on modernisti luodun projektinseurantasovelluksen välillä. Toiminnallisuuksia voi laajentaa, mutta projektin muokkaus ja lisääminen olivat perustarpeita. Trellon kaltainen projektinseurantasovellus olisi hyvä tavoite.



Kuva 12. Tunnetun ja suositun projektinseurantasovelluksen Trellon käyttöliittymä (Trello 2024).

5.1 Alustan käyttöönotto, käytettävyys ja sovelluksen luonti

AppSheet oli ensimmäinen no-code-alusta, jota tämän tutkimuksen tekijä käytti tässä työssä. Alustan käyttöönotto ei ollut helppoa, koska tämän tutkimuksen tekijällä ei ollut aikaisempaa kokemusta tällaisesta alustasta. Alussa oli vaikeaa, mutta kun tämän tutkimuksen tekijä tutustui alustan kaikkiin ominaisuuksiin, hän

huomasi sen rajoitukset. Tämä ei ollut huono asia uudelle käyttäjälle, koska liiallinen joustavuus lisää sovelluksen käyttövaikeuksia. Esimerkiksi modernin web-kehityksen kehittäjän pitää oppia jatkuvasti uutta, mutta no-code-alustan käyttäjä ei tarvitse oppia mitään muuta kuin alustan tarjoamia asioita.

Kun tämän tutkimuksen tekijä on perehtynyt AppSheetiin, sen käyttö ei ollut enää vaikeaa hänelle. AppSheetin käytön alussa projektin luonnissa oli kyselylomake siitä, minkä sovelluksen käyttäjä haluaa luoda. Vastauksen avulla luodaan automaattisesti projektin pohja. Tämän tutkimuksen tekijän mielestä se on käyttäjäystävällistä.

Bubble oli vapaamuotoisempi kuin AppSheet, koska alussa se ei kysynyt käyttäjältä mitään. Projektin pohja oli täysin tyhjä, joten käyttäjällä oli enemmän opittavia asioita kuin muilla alustoilla.

Jos käyttäjä on tottunut käyttämään kuvankäsittelyohjelmia, esim. Photoshopia, hänellä olisi varmaan helppo käyttää Bubblea, koska tämä alusta on hyvin samankaltainen alusta.

Vaikka Bubble tarjoaa monipuolisia mahdollisuuksia sovellusten luomiseen ilman koodaustaitoja, sen käyttöön tutustuminen ja sovelluksen luominen ovat saattaneet vaatia enemmän aikaa verrattuna muihin alustoihin.

Monday.com oli taas erilainen no-code-alusta kuin aiemmin mainitut alustat. Monday.com luo käyttäjän vastauksen perusteella valmiin sovelluksen, joten käyttäjällä oli hyvin vähän muokattavia ja korjattavia asioita tämän alustan projekteissa. Käyttäjä saa vaihtaa esim. komponenttien väriä ja nimeä.

AppSheetiin ja Bubbleen tarvitsee käyttäjän rakentaa omaa tietokantaa, monday.comin ei tarvitse tehdä muuta kuin vastata kyselyyn ja muokata valmista sovellusta.

5.2 Sovelluksen käyttöliittymä ja toiminnallisuudet

AppSheetilla luodun sovelluksen käyttöliittymä on monipuolinen. Se toimii monilla laitteilla, esim. puhelimella, tabletilla ja tietokoneella.

Kun käyttäjä kirjautuu sisään sovellukseen, käyttäjä näkee ensin Trackerin eli sovelluksen kotisivun, jossa on esillä kaikkien lisättyjen projektien tilanteet. Trackerin vieressä on New Project-painike ja calendar-painike, joiden avulla käyttäjä voi lisätä uuden projektin ja tarkastella olemassa olevia projekteja kalenterin muotoisessa näkymässä.

AppSheetin version käyttöliittymän tyyli on yksinkertainen, joten kuka tahansa voi helposti käyttää tätä sovellusta ilman opastusta.

Bubblen versio toimii vain selaimella, ja se on hyvin samankaltainen kuin AppSheet-versio, mutta siinä on enemmän ominaisuuksia, esim. like-painike ja muistio.

Bubble-versiossa on myös käyttäjän tunnistaminen, ja käyttäjän täytyy kirjautua ensin sisään, jotta projektin lisääminen ja muokkaaminen onnistuvat.

Monday.comin versio on selkein ja yksinkertaisin näistä alustasta, ja sen tyyli on hieman samanlainen kuin Microsoft Excelin.

Kalenterit-ominaisuus oli maksullinen lisäosa Bubblessa ja monday.comissa. Tämän vuoksi kalenterit-ominaisuutta ei toteutunut kummassakaan versiossa.

Tämän tutkimuksen tekijän mielestä monday.com-versio on hyvä vaihtoehto pienelle projektille, koska sen käyttöliittymä on yksinkertainen ja selkeä. Jos käyttäjällä on vähän suurempi projekti ja käyttäjä tarvitsee täsmällisyyttä. AppSheet-versio on keskimäärin paras vaihtoehto kolmesta, koska se ei ole niin vaikea käyttää kuin Bubble eikä myöskään liian yksinkertainen kuin monday.com. Jos käyttäjällä on paljon projekteja ja tarvitsee seurata tarkkaan projekteja, Bubble-versio on absoluuttisesti sopivin vaihtoehto.

5.3 Alustojen vertailu

Monday.com vaikuttaa todella kätevältä, varsinkin jos käyttäjä haluaa nopeasti toimivan ratkaisun ilman suurta työtä. Tämä alusta on hyvä vaihtoehto, koska se tarjoaa käyttäjälle valmis sovellus. Huono puoli on se, että käyttäjällä on vaikea oppia uutta asiaa tässä alustassa, ja se tarjoaa monta maksullista palvelua. Tämän vuoksi käyttäjä tarvitsee vähän budjettia, jos käyttäjä haluaa lisää ominaisuuksia sovellukseen.

AppSheet sopii sellaiselle käyttäjälle, joka haluaa luoda riittävän monimuotoisen sovelluksen eikä halua käyttää paljon rahaa projektiin.

Jos käyttäjä on aloittelija ja haluaa luoda hyvän sovelluksen, AppSheet on suositeltava vaihtoehto, koska se on monimuotoinen ja tarjoaa käyttäjälle sopivan sovelluspohjan. Lisäksi tämän alustan takana on luotettava kehittäjäryhmä eli Google, sen yhteensopivuus muiden Google-palveluiden kanssa on myös lisäpiste.

Bubble on ehkä monimuotoisin alusta kolmesta alustasta, koska se tarjoaa käyttäjälle täysin tyhjän pohjan, jolle käyttäjä rakentaa sen pohjalta oman sovelluksensa. Bubblella on kuitenkin maksullinen palvelu, mutta se ei rajoita ilmaisversion käyttöä. Bubblen alustalla käyttäjä voi luoda tyylikkäämpiä sovelluksia kuin modernissa sovelluskehityksessä, mutta tämän alustan käyttö vaatii enemmän oppimista ja aikaa kuin muut alustat.

6 Yhteenveto

Tämän opinnäytetyön tarkoituksena oli tutustua no-code-kehitykseen ja vertailla kolmea no-code-alustaa. Työssä käytettiin kolmea eri no-code-alustaa: AppSheet, Bubble ja monday.com. Alussa tämän tutkimuksen tekijä harkitsi WordPressin käyttöä tutkimusalustana, koska se on suosittu ja tunnettu, mutta huomasi, että WordPressillä oli parempi kehittää web-sivustoja. Tämän tutkimuksen tekijä joutui luopumaan Wordpressistä ja asettamaan sen tilalle monday.com.

Työssä kehitettiin aluksi kolmella tutkimusalustalla projektinseurantasovelluksen kolme eri versiota. Sovellusten kehitysvaiheessa oli monta haastetta, koska kehittäjän piti tutustua kolmeen eri alustaan ja oppia niiden käyttöä. Osalle alustoista täytyi rakentaa vielä oma tietokanta, mikä lisäsi myös koko tutkimuksen työmäärää.

Seuraavaksi sovelluksia verrattiin keskenään, kun ne olivat valmiita. Vertailussa oli sovelluksen toteutuksen ja alustan käytön vaikeustaso, erot alustan ilmaisversion ja maksullisversion välillä sekä alustan yhteensopivuus ulkopuolisten palveluiden kanssa ja käytettävyys.

Tämän tutkimuksen tekijän saaman tuloksen mukaan monday.com oli helpoin käyttää kolmesta alustasta ja Bubble oli vaikein.

AppSheetillä oli hyvä yhteensopivuus, koska se on Googlen kehittämä alusta, jonka voi yhdistää muihin Googlen palveluihin.

AppSheet oli toiseksi vaikein käyttää kolmesta no-code-alustasta. Bubblessa oli enemmän joustavuutta kuin edellisellä mainitulla alustalla. Sen joustavuus lisäsi vaikeutta alustan käyttöön, koska alustalla oli paljon vaihtoehtoja komponenteista ja toiminnallisuuksista. Tämä johti siihen, että kehittäjän oli käytettävä enemmän aikaa alustan oppimiseen.

Tämän tutkimuksen tekijä käytti vain alustan ilmaisversiota, ja siinä ei ole paljon esteitä alustan ilmaiskäyttäjälle. Lisäksi AppSheet ei ollut liian vaikea käyttää tämän tutkimuksen tekijälle. Jos käyttäjä haluaa kehittää heti valmiin

sovelluksen, monday.com ei ole huono vaihtoehto. Bubble sopii sellaiselle käyttäjälle, joka tahtoisi mahdollisimman paljon joustavuuksia kehityksessään.

Tutkimustulokset perustuvat tämän tutkimuksen tekijän alan tuntemukseen ja kokemukseen. Työssä oli vaikeaa kehittää kolme täysin samanlaista sovellusta no-code-alustojen rajoitusten vuoksi.

Lähteet

Aastha, Kochar. 2024. The Future of No-Code. Verkkoaineisto. Shnoco. <https://www.shno.co/blog/future-of-no-code/>. 22.1.2024. Luettu 8.2.2024

Building Web Apps that Work Everywhere. Scott, Adam. 2016. E-kirja. O'Reilly Media, Inc. Publishing.

Chinmayee, Deshpande. 2023. What is React? Verkkoaineisto. Simplilearn. <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs/>. 23.10.2023. Luettu 11.3.2023.

CSS basics. 2023. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics/>. Päivitetty 2.10.2023. Luettu 24.10.2023.

Data privacy and regulatory compliance in low-code platforms. Fleming, Brian. 2022. Verkkoaineisto. PlanetCrust. <https://www.planetcrust.com/data-privacy-and-regulatory-compliance-in-low-code-platforms?utm_campaign=blog/>. 17.11.2022. Luettu 24.5.2023.

Exploring the Pros and Cons of Low-Code Development. 2023. Verkkoaineisto. Quixy. <https://quixy.com/blog/pros-and-cons-of-low-code-development/#1-limited-customization-options/>. 29.11.2023. Luettu 12.12.2023.

Evolution of HTTP. 2023. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP/>. Päivitetty 10.4.2023. Luettu 14.5.2023.

Frontend vs Backend. Verkkoaineisto. GeeksforGeeks. <https://www.geeksforgeeks.org/frontend-vs-backend/>. 18.4.2023. Luettu 12.5.2023.

HTML basics. 2023. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics/>. Päivitetty 1.3.2023. Luettu 11.3.2023.

Internetin käyttö. Muuttujina internetin käyttäjiä, mrd, prosenttia väestöstä 2005-2022. Verkkoaineisto. FiCom 2022. <https://ficom.fi/ict-ala/tietopankki/internetpalvelut/internetin-kaytto/internetin-kaytto/#internetin-kaytto-maailmanlaajuisesti/>. Päivitetty 2022. Luettu 20.4.2023.

Javascript basics. 2024. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics/>. Päivitetty 16.2.2024. Luettu 26.2.2024.

Kiguolis, Linas. 2023. Pro Code vs. No Code vs. Low Code in 2023. Verkkoaineisto. CodeorNoCode. <https://codeornocode.com/software-development/pro-code-vs-no-code-vs-low-code-2022/>. 4.3.2023. Luettu 20.5.2023.

Low-code no-code: Why low-code reigns supreme. Verkkoaineisto. OutSystems. <https://www.outsystems.com/tech-hub/low-code/no-code/#what-is-low-code-no-code/>. Luettu 10.3.2024.

medium. 2023.<https://medium.com/@santhoshguna0511/what-is-the-software-development-process-explaining-types-of-process-models-c1737c5cef62>. Viitattu 10.5.2024

Patel, Bhaval. 2023. Software Development Process. Verkkoaineisto. Space-O Technologies. <https://www.spaceotechnologies.com/blog/software-development-process/>. 15.1.2023. Luettu 12.3.2023.

Pavlovic, Dwight. 2020. What Is Web Hosting and How Does It Work? Verkkoaineisto. HP. <https://www.hp.com/us-en/shop/tech-takes/what-is-web-hosting?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp/>. 18.9.2020. Luettu 12.3.2023.

Professional JavaScript for Web Developers, 5th Edition. Frisbie, Matt. 2023. E-kirja. O'Reilly Media, Inc. Publishing.

Romano, Jenna. 2022. Responsive web design vs. adaptive: Which should you use? Verkkoaineisto. WIXBlog. <https://www.wix.com/blog/responsive-vs-adaptive-design/>. 2.6.2022. Luettu 6.6.2023.

Roy, Sandip. 2022. The Difference Between a Framework and a Library. Verkkoaineisto. Baeldung. <https://www.baeldung.com/cs/framework-vs-library/>. 23.12.2022. Luettu 5.10.2023.

Trello. 2024. <https://trello.com/>. Viitattu 10.5.2024

Understanding client-side JavaScript frameworks. 2024. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/>. Päivitetty 5.3.2024. Luettu 11.3.2024

What is cloud hosting? Verkkoaineisto. Google. <https://cloud.google.com/learn/what-is-cloud-hosting/>. Luettu 12.3.2024.

What is low-code? Verkkoaineisto. IBM. <https://www.ibm.com/topics/low-code/>. Luettu 14.3.2023.

What is no-code and when should you use it. Verkkoaineisto. OutSystems. <https://www.outsystems.com/tech-hub/no-code/>. Luettu 16.3.2023.

You, Evan. 2016. Creator of Vue.js. Cromwell, Vivian. Haastattelu 3.11.2016