Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2025

Riina Mattila

Verkkokehityksen modernit teknologiat ja niiden soveltaminen verkkosivuissa

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2025 | 38 sivua

Riina Mattila

Verkkokehityksen modernit teknologiat ja niiden soveltaminen verkkosivuissa

Opinnäytetyön tavoitteena oli tutkia modernien verkkosivuteknologioiden, erityisesti low-code-, no-code- ja React-ratkaisujen, ominaisuuksia sekä niiden soveltuvuutta erilaisten verkkosivuprojektien toteutukseen. Tarkoituksena on muodostaa ymmärrys siitä, millaisia hyötyjä ja rajoitteita teknologioihin liittyy.

Työssä hyödynnettiin käytettyjen lähteiden lisäksi käytännön esimerkkejä, joissa toteutettiin sama nettisivu sekä no-code-alustalla että Reactilla. Näiden kehitysprosessien nopeutta, haasteita ja kustannuksia verrattiin keskenään. Lisäksi analysoitiin teknologioiden ylläpidettävyyttä, skaalautuvuutta ja soveltuvuutta jatkokehitykseen.

Tulokseksi saatiin, että no-code- ja low-code-alustat sopivat parhaiten nopeasti toteutettaviin ja toiminnoiltaan yksinkertaisempiin projekteihin, kun taas React on parempi valinta, kun tarvitaan mahdollisuus räätälöintiin sekä laajennettavuuteen.

Tuloksista voidaan päätellä, että teknologian valinta verkkosivuprojekteissa kannattaa perustaa projektin monimutkaisuuteen, budjettiin sekä ylläpidollisiin seikkoihin, nopeissa ja kevyissä projekteissa no-code riittää, mutta laajoissa ratkaisuissa tarvitaan Reactin joustavuutta.

Asiasanat:

Verkkosivuteknologiat, no-code ja low-code-alustat, verkkokehitys.

Bachelor’s / Master’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communication Technology

2025 | 38 pages

Riina Mattila

Modern Web Development Technologies and Their Application in Websites

The aim of this thesis was to explore the characteristics of modern web technologies and their applicability to different development projects, while also providing a practical understanding of the benefits and challenges associated with these technologies.

In addition to a literature review, the study included practical examples, where the same website was implemented using both a no-code platform and React. The speed, challenges, and costs of these development processes were compared. Furthermore, the maintainability, scalability, and suitability of the technologies for further development were analyzed.

The results showed that no-code and low-code platforms are best suited for quickly implemented projects with simpler functionality, whereas React is a better choice when customization and scalability are required.

Based on the findings, it can be concluded that the choice of technology in web development projects should be based on the project's complexity, budget, and maintenance needs. No-code is sufficient for lightweight and fast projects, but more extensive solutions require the flexibility offered by React.

Keywords:

Web development technologies, no-code and low-code platforms, web development

Sisältö

Kuvat

Taulukot

1 Johdanto

Verkkokehityksessä käytettävät teknologiat kehittyvät nopeasti, ja uusien ratkaisujen, kuten no-code- ja low-code-alustojen sekä modernien JavaScript-kirjastojen käyttö verkkosivujen toteutuksessa on kasvanut huomattavasti viime vuosien aikana. Tässä opinnäytetyössä perehdytään verkkokehityksen työkaluihin sekä selvitetään, millaisiin verkkosivuprojekteihin kukin teknologia parhaiten soveltuu.

Aiheen valintaan vaikutti oma kiinnostukseni kehittyviin verkkokehitysteknologioihin ja miten teknologiavalinnat vaikuttavat kehitysprosessiin. Lisäksi oli kiinnostavaa selvittää, miten teknologiat pystyvät vastaamaan nykyaikaisen digiympäristön vaatimuksiin nopeudesta, skaalautuvuudesta sekä taloudellisesta tehokkuudesta. Teknologioiden nopea kehitys ja tarve kustannustehokkaille verkkoratkaisuille tekevät aiheesta ajankohtaisen ja tärkeän.

Opinnäytetyössä tietoperustana käsitellään verkkokehityksen historiaa, teknologisia kehitysaskeleita sekä nykyisiä työkaluja ja alustoja. Vaikka lähteiden anti on kattava, useimmat niistä ovat blogikirjoituksia tai yrityssivustoja, joiden ensisijainen tarkoitus on esitellä teknologiaa myönteisessä valossa. Tämän takia voidaan kyseenalaistaa tiedon objektiivisuutta ja soveltuvuutta kriittiseen vertailuun. Lisäksi useat artikkelit keskittyvät ennemmin vain yksittäisten teknologioiden esittelyyn ilman analysointia teknologian vaikutuksesta käytännön toteutukseen. Tämä opinnäytetyö pyrkiikin tuottamaan vastauksen tähän puutteeseen.

Työn tavoitteena on analysoida ja vertailla eri verkkosivuteknologioita sekä laatia suosituksia erityisesti pk-yrityksille, joilla on tavoite kehittää verkkosivujaan moderniksi resurssitehokkailla ratkaisuilla. Työn keskeiset sisältöalueet ovat verkkokehityksen aikajanaan sekä nykytrendeihin perehtyminen, eri teknologioilla toteutettujen verkkosivujen rakentaminen sekä prosessin arviointi ja vertailu. Työn rakenne etenee teoriasta toteutukseen ja lopuksi vertailuun ja suosituksiin.

2 Verkkokehityksen modernit teknologiat

Tässä luvussa käydään läpi verkkokehityksen nykyisiä teknologioita sekä niiden tulevaisuutta. Tarkastellaan low-code- ja no-code-alustoja ja niiden ominaisuuksia sekä perinteisempää web-kehityskehystä Reactia.

2.1 Verkkokehityksen muutos ja trendit

2.1.1 Verkkokehityksen varhaiset vuodet

Vuonna 1991 Tim Berners-Lee julkaisi ensimmäisen nettisivun, joka oli tehty käyttäen julkaisijan kehittämää HTML 1.0 -merkintäkieltä. Kieli sisälsi silloin 18 elementtiä, joiden avulla tekstin rakennetta pystyi muotoilemaan. (Berners-Lee, 1991.) Kuvia, värejä, fonttien vaihtomahdollisuutta tai muita nettisivuja toisistaan erottavia elementtejä ei ollut, joten nettisivut näyttivät 1990-luvun alussa pitkälti toistensa kopioilta. (Raj, 2024)

Kysyntä visuaalisesti miellyttäviin nettisivuihin kasvoi, joten vuonna 1994 norjalainen Hakon Wium Lie ja saksalainen koodaaja Bert Bos kehittivät CSS-tyylikielen ensimmäisen version. Sen avulla saatiin määriteltyä HTML:n avulla luodun tekstin ulkoasu ja lisättyä täten yksilöllisyyttä nettisivuihin värien, fonttien ja taustojen avulla. (Raj, 2024)

Kun staattisille nettisivuille tuli väriä, niille alettiin kaipaamaan liikettä. Tähän Netscape-niminen yritys vastasi kehittämällä komentosarjakielen nimeltä Mocha, joka vaihtui myöhemmin JavaScriptiksi. JavaScriptin avulla saatiin liikkuvuutta nettisivuille kuten, animoituja grafiikoita, kuvaesityksiä, tekstiehdotuksia ja käyttäjän kanssa vuorovaikutuksessa olevia lomakkeita. Näistä kolmesta kielestä tuli lopulta verkkokehityksen kolme kulmakiveä: rakenne, ulkonäkö sekä toiminnot. (Creek, 2022)

2.1.2 Dynaamisuus ja sisällönhallinta

Vähitellen verkkosivuista haluttiin tehdä tietokantapohjaisia, sillä staattisten sivujen ominaisuudet olivat ongelmallisia käyttäjäkohtaisen sisällön, päivitysten ja skaalautuvuuden suhteen. PHP kehitettiin vuonna 1995 ensin kehittäjä Rasmus Lerdorfin omaan käyttöön verkkosivujen kävijämäärän seurantaan, mutta myöhemmin laajempaan käyttöön, jolloin siitä oli apua lomakkeiden käsittelyssä sekä dynaamisen sisällön hallinnassa. (Dirox, 2024) Samoihin aikoihin Michael Widenius ja David Axmak kehittivät MySQLin tavoitteenaan kehittää ratkaisu isojen datamäärien nopeaan ja tehokkaaseen käsittelyyn. MySQL sai nopeasti paljon huomiota sen ollessa kevyt ratkaisu suurten tietomassojen käsittelyyn, pysyessään samalla nopeana, luotettavana ja skaalautuvana. (30 Days Coding, n.d.)

PHP ja MySQL olivat keskeiset teknologiat WordPressin kehityksessä. WordPress on yksi ensimmäisistä sisällönhallintajärjestelmistä, joka teki sisällönhallinnan helpoksi sekä avoimen lähdekoodin alustan saataville suurelle käyttäjäkunnalle. WordPressin perustuksen takana oli idea käyttäjäystävällisestä blogialustasta, mutta se laajeni nopeasti myös blogimaailman ulkopuolelle. (WP Marmite, n.d.) Ensimmäisen version julkaisun jälkeen WordPress kehittyi nopeasti saaden työkaluja sivuston ulkoasun muuttamiseen, luokkien lisäämiseen blogisivustoille, sekä nopeaan kommenttien poistamiseen. Se siis kehittyi niin sisällönhallinnallisesti, että mukauttamismahdollisuuksien osalta, mikä mahdollisti entistä laajemman käytön ja erilaisten verkkosivustojen luomisen ilman syvällistä teknistä osaamista. (Editoral Staff of Wpbeginner, 2025)

Vuosituhannen vaihteen jälkeen nettisivujen määrän kasvu kiihtyi, joten kiihtyi myös tarve tehdä nettisivujen koodauksesta mutkattomampaa. jQuery julkaistiin auttamaan tätä ongelmaa. Kehittäjä John Resigin mukaan hän loi kyseisen JavaScript-kirjaston, jotta JavaScriptin kirjoittamisesta tulisi tehokkaampaa ja vaivattomampaa. JQuery saavutti nämä tavoitteet yksinkertaistamalla yleisiä, toistuvia tehtäviä poistamalla turhaa koodia sekä tarjoamalla selkeitä, lyhyitä ja

ymmärrettäviä ratkaisuja. JQueryn suosituimmat ominaisuudet olivat ergonominen API-verkkosivujen elementtien käsittelyyn sekä se, että selainten välisten eroavaisuuksien poistamisen avulla saatiin koodia, joka toimisi luontevasti kaikissa selaimissa, mikä oli harvinaista 2000-luvun alussa. (Gyo, 2019)

2.1.3 Modernin verkkokehityksen murros

HTML5 kehitettiin vastaamaan nykyaikaisten verkkosovellusten tarpeita, sillä aiempi versio 4.01 oli vanhentunut eikä sopinut yhteen sen tilalle kehitetyn XHTML 2.0:n kanssa. Merkittävimmät uudistukset HTML:n uusimmassa versiossa oli semanttiset elementit, kuten <header>, <section>, <footer>, jotka selkeyttivät nettisivujen rakennetta, kun käytössä ei ollut pelkästään -tagit. HTML5 tarjosi myös tuen äänelle ja videolle, jolloin kolmannen osapuolen lisäosien tarve vähentyi. Lomakkeet saivat myös uusia kenttätyyppejä ja yhteensopivuutta selainten välillä lisättiin sekä sovellusten toiminta taattiin myös ilman internetyhteyttä, kun tietoja pystyttiin tallentamaan paikallisesti uusien localStorage- ja sessionStorage-ominaisuuksien avulla. (W3C, 2014)

Niin kuin HTML5 ja jQuery, myös CSS3 toi mukanaan verkkokehittäjille mahdollisuudet selaintenväliseen yhteensopivuuteen. CSS3 toi mukanaan vuonna 2011 ominaisuuksia, jotka auttoivat silmää miellyttävän verkkosuunnittelun toteuttamista. Uusia ominaisuuksia oli esimerkiksi pyöristettyjen kulmien tekemistä elementeille, ilman kuvia tai monimutkaisia HTML-rakenteita, varjojen asettaminen elementeille sekä tekstille, animaatioiden tekeminen, ilman että joutuu käyttämään JavaScriptiä sekä kustomoitujen fonttien käyttäminen. Nämä ominaisuudet saivat nettisivuista nopeammin ladattavia sekä tyylittely oli edelleen mahdollista tehdä uniikimmaksi. (Vadita, n.d.)

2010-luvun alussa alkoi tulla yhä enemmän tarvetta nopeuttaa verkkosivujen tekemistä, sillä nettisivuja oli olemassa jo yli 200 000 000 ja määrä vain lisääntyi. (Cardillo, 2024) Yksi 2010-luvun alkupuolen tärkeimmistä front-end-

kehyksistä oli Bootstrap. Twitterin työntekijät Mark Otto ja Jacob Thornton kehittivät sen, ensiksi nimellä Twitter Blueprint, tarkoituksenaan saada yrityksen sisäisten työkalujen käyttöliittymät yhtenäisemmiksi sekä niiden ylläpitokustannukset vähenemään. Bootstrap vaikutti verkkosivujen ja sovellusten luomiseen tarjoamalla valmiita HTML-, CSS- ja JavaScript-komponentteja. Sen sisältämän responsiivisuuden ansiosta verkkosivu tai sovellus voi mukautua automaattisesti käyttäjän näytön kokoon ja käyttäjän käyttökokemus tulee täten mukavammaksi ja vaivattomammaksi. (Dev, 2021)

Samoihin aikoihin lisääntyviin käyttöliittymäkehityksen vaatimuksiin ja monimutkaistuneiden verkkosovellusten vaatimuksiin tuli vastaamaan kaksi merkittävää teknologiaa: React ja Angular. (Shaduk, 2023)

Facebook julkaisi vuonna 2013 JavaScript-kirjaston nimeltä React, joka keskittyi käyttöliittymäkomponenttien rakentamiseen. Se mahdollisti koodin uudelleenkäytön sekä paremman hallittavuuden. Sen sisältämä virtuaalinen DOM päivittää vain sivuston muuttuneet osat, eikä koko sivua, mikä tekee siitä tehokkaan dynaamisissa sovelluksissa. (Shaduk, 2023)

Angular on sen sijaan Googlen vuonna 2010 julkaisema kehitysalusta ja sovelluskehys, joka on rakennettu TypeScriptillä. Sen sisältämät kirjastot ja ominaisuudet helpottavat front-end-kehitystä saaden sitä nopeammaksi mutta myös skaalaamaan projekteja yhden kehittäjän sovelluksista, sovelluksiin, joita kehittävät useat koodarit. (Shaduk, 2023)

2.1.4 Nykypäivän kehitystyökalut ja trendit

Viimeisen kymmenen vuoden aikana on luotu yhä enemmän ratkaisuja, jotka edistävät verkkokehitystä, samalla kun ratkaisut siirtyvät monimutkaisemmiksi. Näitä ratkaisuja ovat esimerkiksi päätön CMS, pilvipalvelut ja low-code- ja no-code-kehitystyökalut.

Päätön CMS eli päätön sisällönhallintajärjestelmä keskittyy sisällönhallintaan ilman esitysmuotoon liittyviä rajoituksia, toisin kuin esimerkiksi WordPress, joka

hallitsee sekä sisällön että käyttöliittymän. Päätöntä CMS:ää hallitaan yhdestä paikasta, mutta sitä voi jakaa useille kanaville, kuten verkkosivuille, sovelluksiin ja älylaitteisiin. Päättömän CMS:n etuja ovat nopeus ja tietoturva, sillä sisältö hallitaan erillisessä järjestelmässä ilman suoraa yhteyttä tietokantoihin. Lisäksi sen yhdistäminen muihin järjestelmiin on vaivatonta, ja sisältöä voidaan jakaa samanaikaisesti useisiin kanaviin, mikä helpottaa sisällönhallintaa. (Arola, 2022)

Pilvipalveluita on neljään tarkoitukseen. PaaS-palvelut tarjoavat kehitystyökaluja, tietokantoja ja palvelinympäristöjä. Se nopeuttaa sovelluskehitystä, sillä turhaa huolta palvelinympäristöjen hallinnasta ei ole. IaaS tarjoaa sen sijaan IT-infrastruktuuria, kuten virtuaalikoneita ja tallennustilaa. Ratkaisu on kehittäjäyritykselle taloudellinen, sillä investointitarve omiin laitteistoihin pienenee ja koko IT-infrastruktuuri on nopeasti tavoitettavissa. BaaS vaikuttaa mobiili- ja verkkosovellusten kehittämiseen tarjoamalla tietokannan hallinnan, käyttäjien todennukset ja push-ilmoitukset, jolloin sovellusten kehittäjät pystyvät keskittymään front-endin kehittämiseen ja säästävät kustannuksia ja aikaa, kun taustajärjestelmä tulee palveluntoimittajan kautta. SaaS-mallissa ohjelmistot toimitetaan kokonaan pilvipalveluna, ilman asentamista tai ylläpitämistä omille laitteille. Näitä voi olla esimerkiksi sähköpostipalvelut tai liiketoimintasovellukset. (Hurja, 2024)

Low-code ja no-code on sovelluskehitysalustoja, joilla sovelluksia pystytään rakentamaan siten, että koodia käytetään joko hieman tai ei ollenkaan. Alustat toimivat vedä-pudota-toiminnolla ja muotoilu tapahtuu visuaalisessa näkymässä. Nämä alustat tuovat haluttua nopeutta ja ketteryyttä sovellusten luomiseen. (Simplified, n.d.) Lisää low-code- ja no-code-alustoihin perehdytään seuraavassa luvussa.

2.2 Low-code ja no-code-alustat

Low-code ja no-code-alustat tarjoavat ennalta rakennettuja komponentteja ja käyttövalmiita malleja sovelluskehitykseen sen nopeuttamiseksi ja sujuvoittamiseksi. Alustat perustuvat mallipohjaiseen suunnitteluun,

automaattiseen koodin tuottamiseen ja visuaaliseen ohjelmointiin. (SAP, n.d.) Koodi siis syntyy käyttäjän tekemien visuaalisten valintojen pohjalta automaattisesti, eikä käyttäjän tarvitse kiinnittää siihen erikseen huomiota, ellei low-code-alustalla halua tehdä omia muutoksia.

2.2.1 No-code-alustat

No-code-alustat eivät tarvitse minkäänlaista kokemusta koodaamisesta. Nettisivut voidaan rakentaa valmiista mallista, vain muuttamalla tekstit ja kuvat, joten edes tietoa nettisivujen rakenteesta ei välttämättä tarvitse omata. Heikon mukautettavuuden puolesta niitä kuitenkin suositellaan pienempiin projekteihin, jotka saattavat olla vain yrityksen sisäisessä käytössä, kuten kalenterit, lomien hyväksynnän applikaatiot tai yksinkertaiset nettisivut, joissa ei tarvita eritysominaisuuksia. (Outsystems, n.d.) Käyttöystävällisyytensä ja nopean käyttöönottonsa vuoksi no-code-alustoilla tehdyt ratkaisut sopivat yrityksille, jotka tähtäävät esimerkiksi paperittomampaan toimistoon tai prosessien automatisointiin, sillä no-code-ratkaisut ovat kustannustehokkaita ja nopeasti toteutettavissa.

Alustat, joilla koodausta ei tarvita, tuovat mukanaan hyötyjä, joita yritykset nykymaailmassa osaavat arvostaa ja kaivata.

Kustannustehokkuus on no-code-alustojen suuri etu, koska ne ovat nopeita käyttää, myös työtunteja yhtä projektia kohtaan kuluu vähemmän. Yritykset voivat hyödyntää nykyistä henkilöstöään verkkosivustojen ja sovellusten luomiseen ilman, että heidän tarvitsee kouluttaa työntekijöitä perinteiseen ohjelmointiin. Tämä vähentää riippuvuutta kehitystiimeistä ja ulkopuolisista asiantuntijoista, mikä tuo säästöjä ja lisää yrityksen sisäistä osaamislaajuutta.

Verkkokehitys tehostuu huomattavasti no-code-alustojen avulla, sillä ne mahdollistavat ideoiden nopean muuttamisen toimiviksi tuotteiksi. Siinä missä perinteinen koodausprojekti voi kestää kuukausia, no-code-ratkaisujen avulla sovelluksia voidaan rakentaa ja ottaa käyttöön jopa muutamassa päivässä. Tämä nopeus mahdollistaa pikaisen reagoinnin markkinatarpeisiin sekä

asiakkaan vaatimuksiin ja se on suuri kilpailuetu nykypäivän digitaalisessa ympäristössä.

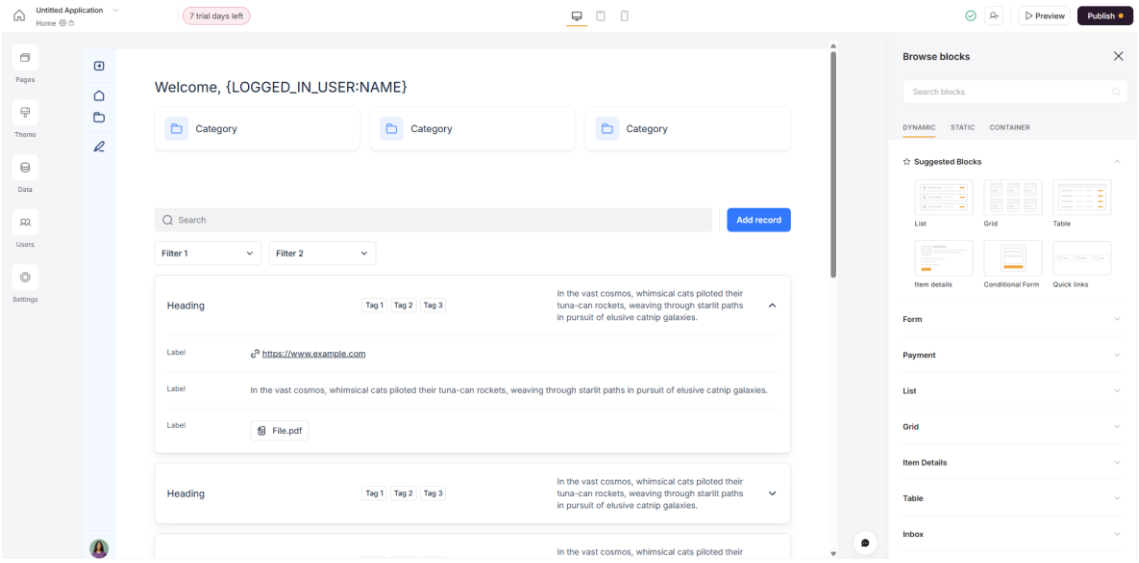
Vaikka no-code-alustat mahdollistavat nopean ja helpon sovelluskehityksen ilman ohjelmointiosaamista, niiden käyttöön liittyy tiettyjä rajoituksia. Erityisesti suurille ja monimutkaisille järjestelmille no-code voi olla liian yksinkertainen ratkaisu, sillä se ei aina tarjoa tarvittavaa mukautettavuutta monimutkaisten prosessien toteuttamiseen valmiilla komponenteillaan ja malleillaan, jotka voitaisiin saavuttaa perinteisellä koodauksella tai low-code-alustalla. (Khan & Downie, 2024)

No-code-alustat toimivat myös itsenäisesti suljetuissa ympäristöissä, joten niitä ei pysty välttämättä yhdistämään vanhoihin järjestelmiin ja kolmannen osapuolen palveluihin, joka voi rajoittaa tiedon liikkumista tai käyttöä. (Jednaszewski, 2024)

Koska sovelluksen tai nettisivun rakentaja ei itse kehitä ja ylläpidä koodia, koodin laatua ei pysty tarkastamaan. Tämä voi johtaa tilanteeseen, jossa sovellus on alttiina haavoittuvuuksille, joita kehitysalustassa saattaa olla. Siksi yrityksen tulee valitessaan alustaa kiinnittää huomiota alustan tietoturvaominaisuuksiin ja päivityskäytäntöihin, jotta kehitetyillä sovelluksilla on riittävä suoja kyberhyökkäyksiä vastaan. (Khan & Downie, 2024)

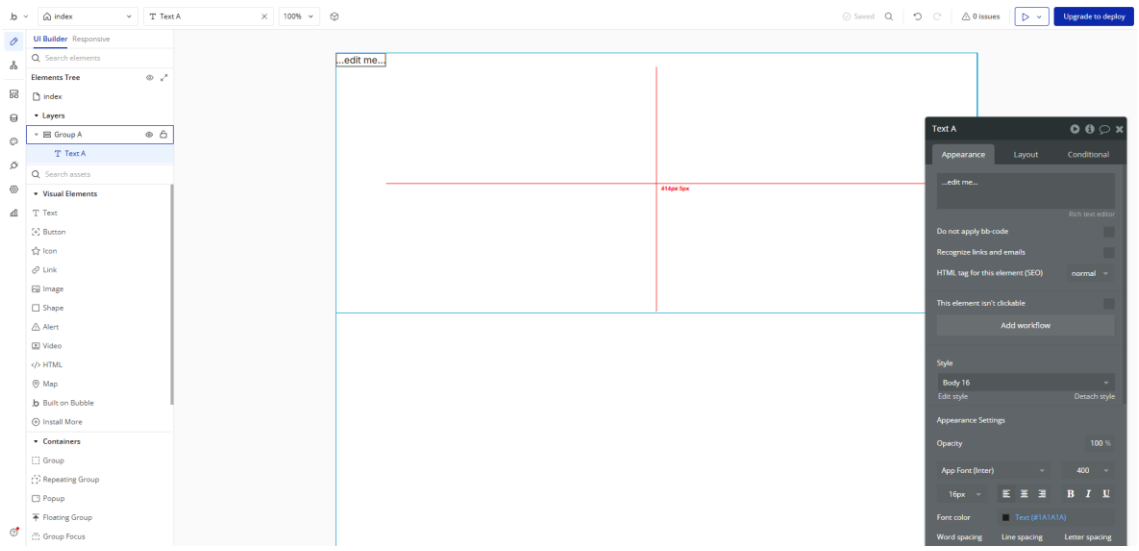
Seuraavaksi käydään läpi suosituimpia no-code-alustoja ja vertaillaan niiden eroja.

Softr-alustan avulla pystyy luomaan no-code-verkkosivuja kokonaan tyhjästä, yli 90 valmiin pohjan avulla tai tekoälyn avulla, jolloin kirjoitetaan vaan komentoja ja tekoäly rakentaa verkkosivut niiden mukaan. Käyttöliittymä on selkeä ja visuaalinen, mikä auttaa sujuvassa nettisivujen rakentamisessa, vaikka ympäristö olisikin uusi (Kuva 1.). Se sopii parhaiten verkkosivujen ja yksinkertaisten verkkosovellusten rakentamiseen.



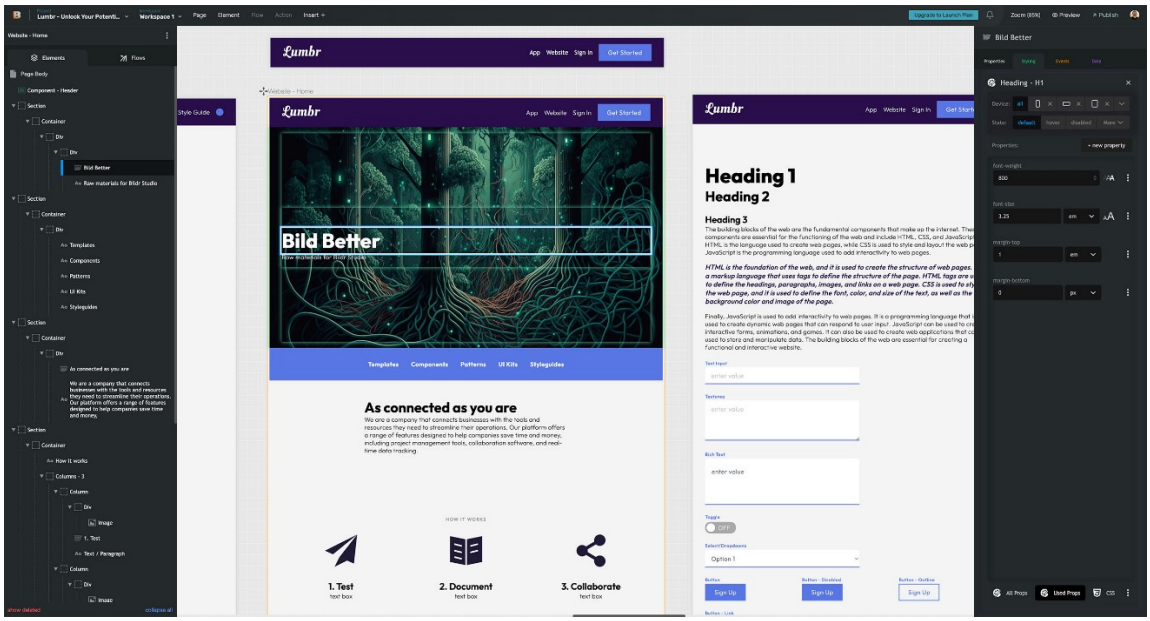
Kuva 1. Softr-alustan käyttöliittymä.

Bubble on yksi tunnetuimmista no-code-alustoista. Sen avulla pystytään rakentamaan laajojakin verkkosovelluksia, sillä suunnittelu sekä datan ja logiikan hallinta pystytään hoitamaan samalla alustalla. Koska Bubble on niin laaja kokonaisuus, se vaatii hieman opettelua alkuun. Sen sisältämiin ominaisuuksiin kuuluu esimerkiksi tekoälyn avulla nettisivujen rakennus. (Bubble, 2025) Käyttöliittymä on hieman yksinkertaisempi kuin muissa esimerkeissä, mutta kuitenkin selkeä. (Kuva 2.)



Kuva 2. Bubblen käyttöliittymä.

Bildr on no-code-alusta, joka tarjoaa visuaalisen käyttöliittymän suunnittelun lisäksi sovelluslogiikan hallinnan. Sen avulla pystyy rakentaman vaativampiakin verkkosovelluksia, joissa on toiminnallisuuksina esimerkiksi kirjautuminen, käyttäjähallinta ja integraatiot kolmannen osapuolen palveluihin. Käyttöliittymä Bildrissä on visuaalisesti moderni, mutta voi tuntua hieman raskaalta opetella alkuun, sillä sisältää paljon toimintoja kuten Bubblekin. (Kuva 3.)



Kuva 3. Bildrin käyttöliittymä. (Bildr.com, ei pvm)

2.2.2 Low-code-alustat

Low-code-alustat ovat no-code-alustojen ja perinteisen ohjelmoinnin keskitie. Koodauksen perusosaaminen on hyvä olla hallussa, mutta low-code-alustojen käyttäminen on kuitenkin kaukana perinteisestä koodauksesta ja samalla hyvin paljon pidemmällä kuin no-code-alustalla tehdyt sovellukset, sillä mukauttamismahdollisuuksia on paljon enemmän ja kolmannen osapuolen sovelluksia, kuten tekoälyä, koneoppimista ja lohkoketjua, on mahdollisuus käyttää. Alustoilla käytetään ohjelmistorajapintoja, vedä ja pudota-toimintoja, koodialustoja ja muita työkaluja. Jotkin alustat käyttävät vielä vähemmän koodia tekoälytyökalujen avulla, jotka voivat ehdottaa seuraavaa koodin pätkää tai ennustaa koodauksen seuraavat vaiheet. (Gillen, 2025)

Mukautettavuutensa, turvaominaisuuksiensa ja skaalautuvuutensa ansiosta low-code-alustoilla pystyy tekemään monimutkaisiakin sovelluksia, kuten asiakasportaaleja tai yrityksen keskeisiä järjestelmiä. (Outsystems, n.d.)

Low-code-alustat tarjoavat kehittäjille paljon hyötyä ketteryytensä ansiosta. Low-code-alustojen visuaaliset työkalut ja valmiit komponentit helpottavat sovellusten päivittämistä ja ylläpitoa. Koska suurinta osaa koodista ei tarvitse itse kirjoittaa, muutosten tekeminen ja ominaisuuksien lisääminen on nopeampaa kuin perinteisessä ohjelmoinnissa.

Low-code-ratkaisut mahdollistavat sovellusten rakentamisen huomattavasti nopeammin kuin koodaamalla sovelluksen täysin alusta. Kehittäjät voivat hyödyntää valmiita komponentteja ja automaatiota, mikä lyhentää kehitysprojektien kestoa ja sovellus saadaan nopeammin käyttöön.

Laajennusten ja koodinlisäysmahdollisuuksien avulla low-code-alustoilla saa joustavamman lopputuloksen kuin koodittomalla alustalla, joka on etu etenkin yrityksille, jotka tarvitsevat ratkaisussaan erityisiä toiminnallisuuksia tai integraatioita. (Outsystems, n.d.)

Low-code-alustat tarjoavat merkittäviä etuja, mutta niihin liittyy samalla myös kääntöpuolia. Seuraavaksi käydään läpi keskeisimpiä haittoja ja rajoitteita, joita low-code-alustojen käyttö ilmentää. (Microsoft, n. d.)

Vaikka low-code-alustat vähentävät manuaalisen koodauksen tarvetta huomattavasti, ne eivät poista sitä kokonaan. Ohjelmistokehityksen periaatteet ja peruskoodaustaidot on osattava, jotta pystyy hyödyntämään alustojen koko potentiaalin. (Microsoft, n. d.)

Low-code-alustoissa on yleisesti korkeammat kustannukset kuin täysin koodittomissa alustoissa. Alustan käyttöhinnan lisäksi kustannuksia saattaa koitua työntekijöiden kouluttamisesta koodauksen perusteisiin tai osaavaan henkilöstön palkkaamisesta, kun taas no-code-alustoja pysyi hyödyntämään kuka vaan, koodauskokemuksesta riippumatta. (Microsoft, n. d.)

Vaikka low-code tarjoaa joustavuutta, se ei anna yhtä täydellistä hallintaa sovelluksen rakenteeseen ja tietoturvaan kuin alusta asti koodattu ratkaisu. Kehittäjät ovat osittain riippuvaisia alustan tarjoamista ominaisuuksista ja rajoitteista, mikä voi olla haaste yrityksille, joilla on tiukat tietoturvavaatimukset. (Microsoft, n. d.)

Mendix on low-code-alustojen johtavassa asemassa. Mallipohjaisella toteutuksellaan se tarjoaa ratkaisut sovelluskehitykseen, -testaukseen sekä sovelluksen ylläpitoon. Se on tehty etenkin liiketoiminnan tarpeisiin kohteenaan keskikokoiset ja suuret yritykset. (Kandukuri, 2022)

Outsystems sen sijaan on tarkoitettu enemmänkin IT-ammattilaisille, jotka haluavat hienosäätää sovelluksiaan. Se tukee tekoäly- ja IoT-integraatioita ja tarjoaa edistyneitä räätälöintimahdollisuuksia. OutSystems on tarkoitettu suurille yrityksille ja sen käyttö vaatii teknistä osaamista. (Kandukuri, 2022)

Microsoft Power Apps on sen sijaan helppokäyttöinen alusta, jonka avulla pystytään luomaan yksinkertaisia liiketoimintasovelluksia, etenkin yrityksille, jotka ovat muidenkin Microsoftin sovellusten kanssa tekemisissä. Hinta on alhainen, mikäli ei tarvitse monimutkaista sovellusta ja rajalliset toiminnot riittävät. Mikäli tähtäimessä on monimutkaisempi ratkaisu, pitää käyttää myös muita Microsoftin tuotteita. (Kandukuri, 2022)

2.3 React-kirjasto

React on JavaScript-koodin ja -komponenttien kirjasto ja sen avulla luodaan käyttöliittymän ulkoasu. React kehitettiin aluksi Facebookin toimesta vuonna 2011, ja myöhemmin vuonna 2013 se julkaistiin avoimena lähdekoodina. (Hámori, 2024) React on vuonna 2024 tehdyn tutkimuksen mukaan maailman toiseksi käytetyin kehys heti Node.js:n jälkeen. Sitä käyttävät monet isot brändit kuten Netflix, Apple ja PayPal, mikä viittaa siihen, että kyseessä on luotettava ja tehokas ratkaisu isojenkin sovellusten kehittämiseen. Reactin suosion mukana sen ympärille on kehittynyt iso yhteisö ja sen avulla on kehitetty jo yli 50 miljoonaa verkkosivua. (Modan, 2025)

2.3.1 Reactin toiminta

Reactin käyttö alkaa komponenttien määrittelystä. Komponentit voivat olla joko funktioita tai luokkia. Komponentit ovat Reactissa uudelleenkäytettäviä osia, jotka tekevät käyttöliittymän rakenteen ja toiminnallisuuden mahdolliseksi. Komponentit renderöidään HTML-sivulle ReacGOM.render() -funktiolla.

Selaimen oikean DOMin päivittäminen saattaa olla hidasta, joten React käyttää virtuaalista DOMia, joka on kevennetty kopio sivun rakenteesta. Muutosten tapahtuessa sivustolla React tekee uuden version virtuaalisesta DOMista ja vertaa sitä vanhaan versioon. Kun muuttunut osa on Reactille selvillä, se päivittää selaimen vain siltä osin, mikä on muutettu. Tämä tapa tekee sovelluksista nopeampia, sillä koko sivua ei tarvitse päivittää.

React-komponenteilla on tila eli komponentteihin liittyvää dataa, joka voi muuttua käytön aikana. Tila voi olla esimerkiksi käyttäjän syöttämä tieto. Tila on siis komponentin sisäistä muistia, jota se käyttää päätöksentekoon ja näkymän päivittämiseen. Toiminnallisissa komponenteissa tilaa hallitaan yleensä useState-hookilla, kun taas luokkakomponenteissa käytetään setState-metodia. (UXPin, 2024)

2.3.2 Reactin käytön syyt

Reactin käytön suosioon vaikuttaa sen oppimisen vaivattomuus, joten sen kanssa pääsee nopeasti alkuun, etenkin jos on ennestään kokemusta JavaScriptista.

Reactin avulla pystyy tekemään laadukkaita ja näyttäviä käyttöliittymiä. Käyttöliittymät vaikuttavat käyttökokemukseen, jotka voivat määrittää jatkaako sovelluksen käyttäjä selaamista vai lopettaako sen heti alkuun, joten niiden sujuva ja nopeasti reagoiva toiminta on iso asia etenkin yrityksille.

JSX-syntaksin avulla, joka toimii lisäosana Reactin kanssa, on mahdollisuus alusta asti luoda omia komponentteja, joita voi uudelleen käyttää myöhemmin

koodissa. Tämä on kehittäjälle tarpeellinen ominaisuus, sillä kehitysprosessi nopeutuu, kun samantapaista koodia ei tarvitse kirjoittaa montaa kertaa.

SEO-ystävällisyys on myös tärkeä etu. Nopeammat latausajat ja suorituskyky vaikuttavat hakukonenäkyvyyteen positiivisesti, jolloin Reactilla tehdyt sivut sijoittuvat hakutuloksissa korkeammalle. Tämä on tärkeää yrityksille, joiden tavoitteena on parantaa omaa näkyvyyttään verkossa. (Modan, 2025)

2.3.3 React-kirjaston edut ja haasteet

Reactilla on omat etunsa ja haasteensa, niin kuin muillakin teknologioilla. React on melko samanlainen kuin muutkin JavaScript-kirjastot, joten sen kanssa pääsee alkuun nopeasti, mikäli on ollut tekemisissä muiden vastaavien kirjastojen kanssa aiemmin.

Kun Reactiin tulee versiopäivitys, uusi versio säilyttää yleensä yhteensopivuuden aiempien versioiden kanssa. Tämä tarkoittaa, että olemassa olevan koodin päivittäminen vaatii vain pieniä muutoksia, mikä sujuvoittaa huomattavasti ylläpitoa. (Mateusz, n.d.)

Uudelleenkäytettävät komponentit ovat yksi Reactin suurimmista eduista. Niiden avulla samaa koodia voi hyödyntää eri osissa sovellusta, mikä auttaa isojen kokonaisuuksien hallintaa ja parantaa kehityksen tehokkuutta. (Mateusz, n.d.)

React on myös laajasti tunnettu, ja sillä on aktiivinen kehittäjäyhteisö. Tämä yhteisö tarjoaa runsaasti hyödyllisiä resursseja, kuten dokumentaatiota, opetusmateriaaleja ja valmiita komponentteja, joita voi hyödyntää omassa kehitystyössä. Ongelmatilanteissa voi kääntyä yhteisön puoleen, ja sieltä saa usein apua sekä tukea jatkuvaan oppimiseen. (Mateusz, n.d.)

Hyötyjen rinnalla Reactissa on myös joitain puolia, jotka voivat tuoda lisätyötä tai hankaluuksia etenkin kokemattomalle kehittäjälle.

React ei tarjoa valmista arkkitehtuuria tai tarkkoja ohjeita tilanhallintaan tai muihin API-kutsujen toteutukseen. Kehittäjän täytyy siis itse valita sopivat työkalut ja ratkaisut, mikä voi olla erityisesti aloittelevalle koodaajalle haastavaa. (Mateusz, n.d.)

Koska React keskittyy vain käyttöliittymän rakentamiseen, tarvitaan erillisiä kirjastoja muiden sovelluksen osa-alueiden hoitamiseen. Tämä tuo lisää haastetta kehitysprosessiin ja vaatii enemmän aikaa ja resursseja. (Mateusz, n.d.)

React kehittyy nopeasti, ja sen kanssa käytettävät npm-kirjastot päivittyvät usein. Tämä voi johtaa yhteensopivuusongelmiin ja vaatii kehittäjältä jatkuvaa seurantaa sekä päivityksiä, jotta koodi pysyy päivitysten mukana. (Mateusz, n.d.)

Vaikka Reactin virtuaalinen DOM parantaa suorituskykyä monissa tilanteissa, se ei silti poista kaikkia mahdollisia suorituskykyhaasteita. Suuremmissa ja monimutkaisemmissa sovelluksissa voi silti joutua tekemään lisäoptimointeja, jotta käyttöliittymä pysyy sujuvana. (Mateusz, n.d.)

2.4 Verkkosivuteknologioiden tulevaisuus

Teknologian ja verkkokehityksen yhtenäistyminen näkyy jo nyt, mutta tulee yleistymään tulevaisuudessa vielä enemmän.

Tekoäly ja koneoppiminen tulee näkymään vielä suuremmin tulevaisuudessa niiden tehdessä nettisivuista käyttäjäystävällisempiä ja vaivattomampia käyttää. Sivustot voivat mukautua yksilöllisiin käyttäjätarpeisiin analysoimalla dataa ja käyttäjien toimintaa sivuilla. Tekoäly voi personoida sisältöä eri käyttäjäryhmille, kun taas koneoppimisalgoritmit ennustavat käyttäjien seuraavia toimintoja. (Hridoy, 2024)

Sovellukset tulevat olemaan enemmän PWA (Progressive Web Apps)

-sovelluksia. Ne rakennetaan verkkoteknologioiden avulla, mutta ne tarjoavat käyttökokemuksen, joka vastaa sovelluksen käyttöä, ominaisuuksinaan etenkin käyttömahdollisuus ilman nettiyhteyttä, push-ilmoitukset kuten sovelluksissa sekä ne voivat ladata ja päivittää itseään taustalla, jolloin käyttökokemus ei koe turhia viiveitä. (mdn , 2025)

Tulevaisuudessa ääniohjaus ja äänikomennoilla hakeminen tulee yleistymään, jonka takia verkkosivujen optimointi on tehtävä niin, että puhekieli tulee ymmärretyksi ja haut onnistuvat ongelmattomasti. (Miramar, n.d.)

Lisätty todellisuus ja virtuaalitodellisuus tulevat siirtymään pelimaailmasta laajempaan käyttöön, kun niitä tuodaan nettisivuille ja sovelluksiin. Virtuaalitodellisuuden tuotokset mahdollistavat esimerkiksi netissä myytävien tuotteiden kokeilemisen ennen ostopäätöstä. (JeleTele Design, 2024)

Perinteinen verkkokehitys ei tule poistumaan kokonaan tekoälyn yleistymisen myötä, vaan se tulee kehittymään uusien teknologioiden ohessa. Vaikutusta voidaan verrata esimerkiksi sisällönhallintajärjestelmien kehittymiseen, jotka vähensivät tarvetta manuaaliselle koodaukselle, mutta eivät täysin poistaneet sitä. Perusymmärrys eri verkkoteknologioista tulee olemaan tarpeellista, mikäli kehittäjät käyttäisivätkin tekoälyä enenevissä määrin, ja ymmärryksen syvällisyyden tarve tulee lisääntymään mitä monimutkaisempi projekti on.

Tekoäly tarjoaa dataan perustuvia ratkaisuja ja se tekee kehitysprosessista tehokkaampaa, mutta ihmisen luovuutta, strategista ajattelua sekä huomiokykyä pieniin käyttäjäkokemuksen yksityiskohtiin se ei korvaa. Ihmisen tekemä valvonta ja koodausosaaminen on tarpeen räätälöityjen ja monimutkaisten sovellusten kehittämisessä, vaikka tekoäly tekisikin suurimman osan työstä. (Clark, 2024)

3 Verkkosivujen toteutus eri teknologioilla

3.1 Verkkosivun toteutus no-code-alustalla

Tässä luvussa esitellään verkkosivun toteutus no-code-alustalla. Tarkoituksena on havainnollistaa kehitysprosessin kulkua ja huomata vahvuudet ja rajoitteet. Toteutus toimii vertailukohtana myöhemmin Reactilla tehdylle verkkosivuversiolle.

Sivusto toteutetaan Webflow-nimisellä alustalla, joka on varsinaisesti low-code-alusta, mutta sen ilmaisversiossa ei ole mahdollisuutta lisätä omaa koodia. Tässä yhteydessä Webflow toimii siten täysin no-code-periaatteella, ja soveltuu näin hyvin tarkasteltavaksi esimerkiksi.

Rakennettava sivusto on kuvitteellisen yrityksen kotisivu. Sivuston sisältö suunniteltiin kattamaan keskeiset elementit, joita tyypillisiltä verkkosivuilta odotetaan:

- Navigaatiovalikko
- Yrityksen esittelyosio
- Palveluiden esittely
- Referenssityöt
- Yhteydenottolomake
- Footer eli sivuston alaosa.

Sivusto rakennettiin alusta asti tyhjälle pohjalle, mutta sekä yläosan valikko että alaosan footer pohjautuivat valmiisiin mallipohjiin. Näiden osien muokkaaminen oli nopeaa, vain tekstit korvattiin omilla teksteillä sekä turhat linkit ja elementit poistettiin.

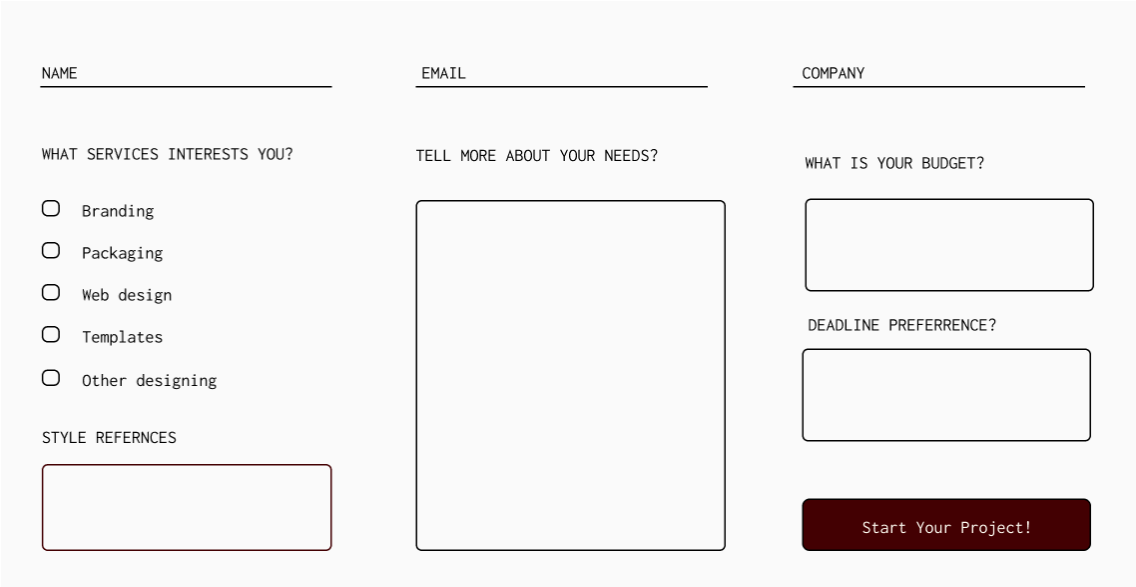
Jokainen osio rakennettiin containerin sisään, johon lisättiin eri elementtejä – kuten tekstikenttiä, kuvia ja painikkeita. Tekstisisällön lisääminen oli suoraviivaista: käyttäjä kirjoittaa tekstin suoraan kenttään ja muotoilee sen Webflown työkaluilla. Yksi merkittävä apuväline oli tyylivalitsin (style selector),

jonka avulla tietylle elementille annettu tyyli voitiin kopioida muihin kohtiin. Tämä vähensi toistoa ja varmisti ulkoasun yhtenäisyyden.

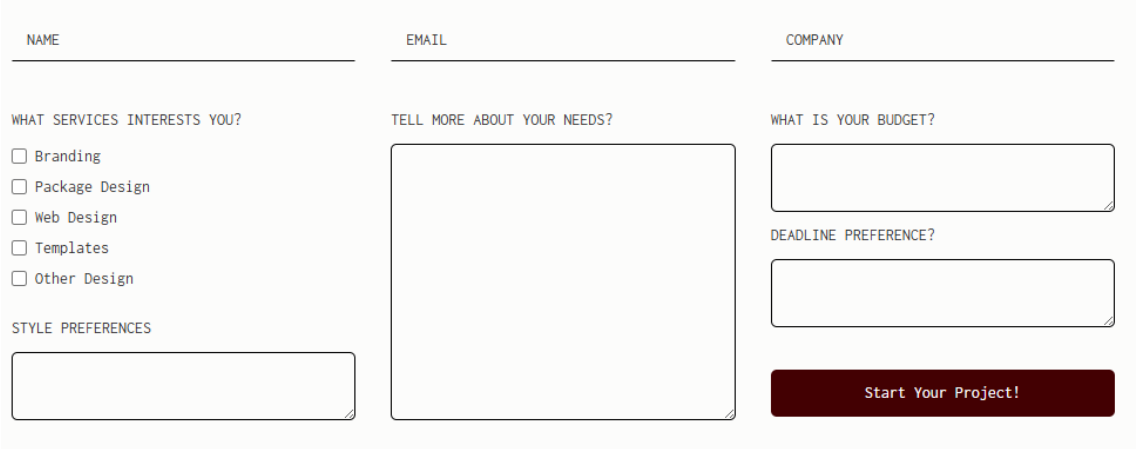
Keskeinen osio sivulla oli välilehtivalikko, jossa käyttäjä pystyi valitsemaan viidestä eri kohdasta, jolloin sisältö vaihtui valinnan mukaan. Sen käyttö oli helppoa, mikäli meni alkuperäisen asettelun mukaan, jossa valikko on vaakasuorassa ja tekstit ilmestyivät valikon alle. Halusin kuitenkin, että valikon tekstit ovat allekkain. Ensisijainen ajatus oli, että teksti ilmestyisi painamisen jälkeen kunkin valikoitavan sanan alle, mutta se olisi tarvinnut muutaman koodirivin toteutuakseen, joten asettelin tekstin tulemaan valikkojonon oikealle puolelle. Asettelu itsessään vei jonkin verran aikaa, että valikko itsessään ei liikkuisi, vaikka esille tulevan tekstin koko vaihtelisi ja suhteet valikon ja tekstin suhteen pysyisivät samana. Mobiili- ja tablettiversiossa tekstit oli kuitenkin siirrettävä meneväksi valikon alapuolelle, jotta ne mahtuisivat näytölle.

Sivustolle tehtiin slider, jossa oli kolme erilaista kuva-teksti-yhdistelmää ja sitä pystyi liikuttamaan nuoli-ikoneista, mutta myös siihen lisättiin sivua elävöittämään automaattinen ajastin, jolloin kuva vaihtuisi 5 sekunnin jälkeen, mikäli slideria ei ollut selattu jo käsivalinalla. Jokaisessa kohdassa oli kuva vasemmalla ja oikealla päällekkäin otsikko sekä tekstiä. Asettelu sekä elementin mukauttaminen omiin tarpeisiin onnistui sujuvasti.

Viimeisenä osiona lomakkeen luominen oli kaikista haasteellisinta. Tavoitteena oli tehdä 3 x 3-taulukko, niin, että se sisältää monivalintakohdan, tekstivastauskohtia sekä erikokoisia vastauslaatikoita sekä lähetysnäppäimen (Kuva 3.). Haasteita tuotti erikokoisten tekstiruutujen lisääminen, niin, että kuitenkin jokainen sarake pysyisi pituudeltaan samana sekä lisäksi viimeisen sarakkeen sisältämät neljä eri elementtiä. Haaste ratkaistiin lisäämällä Div-lohko keskimmäiseen soluun ja sen sisälle kaksi tekstiruutua kysymyksineen. Lopputuloksesta (Kuva 4.) tuli kuitenkin hyvin samanlainen kuin nettisivun mallinna olleessa mockupissa oli suunniteltu.



Kuva 3. Haluttu lomakkeen lopputulos



Kuva 4. Webflowlla tehty lomake

Ulkoasussa oli paljon muokkausvaraa. Fontteja pystyi lisäämään Google Fonttien listalta sekä pystyi myös tarvittaessa lataamaan omia, kustomoituja fontteja. Fonttien lisäksi typografiaa pystyy esimerkiksi muuttamaan koon, rivivälin ja värin muokkaamisella. Samoin elementtien koko, väri, reunukset ja varjostukset oli muokattavissa, lisäksi niiden asettelua pystyi muokkaamaan flexboxin ja gridin avulla.

Webflow tarjoaa valmiita design-komponentteja, kuten galleriaratkaisuja, hinnaston esillepanoa, footerin ja Hero-kuvan asetteluja. Kirjastoja oli myös ladattavissa, niin ilmaiseksi kuin maksullisestikin, jolloin tarvittaessa ulkoasusta olisi saanut vielä erilaisemman ja ainutlaatuisemman.

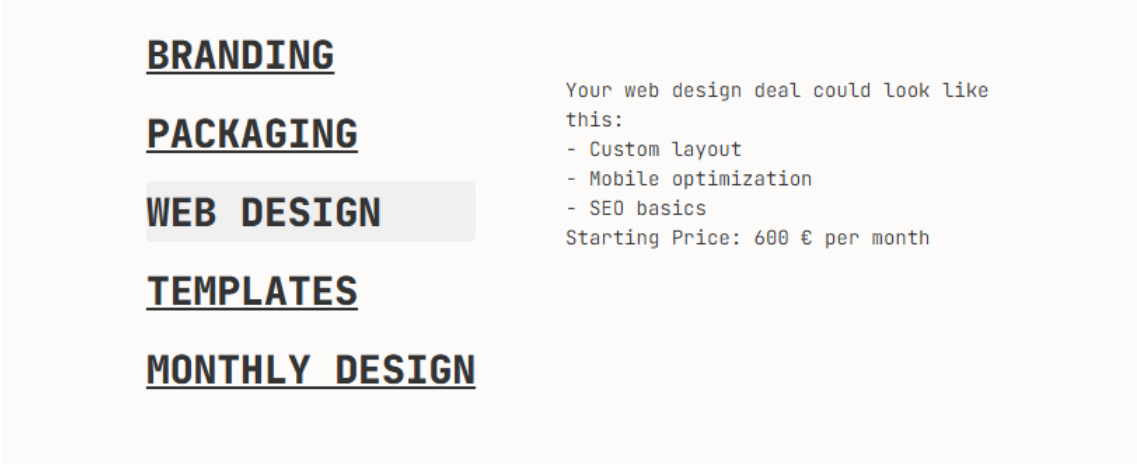
Nettisivun pystyi muokkaamaan myös mobiili- ja tablettiversioon sopivaksi, esimerkiksi muuttamalla fonttikokoja, marginaaleja ja elementtien järjestystä.

3.2 Verkkosivun toteutus Reactilla

React-projekti aloitettiin luomalla kansio ja tiedostot käyttäen Vite-rakennustyökalua. Projektissa App.jsx-tiedosto toimi pääkomponenttina ja sen sisällä alikomponentit renderöitiin. Jokainen vaativampi komponentti tehtiin erilliseen tiedostoon, NavBar.jsx, ServicesSection.jsx, Hero.jsx, ContactForm.jsx, Slider.jsx ja Footer.jsx, lisäksi kaikki tyylit olivat sijoitettu tiedostoittain styles.css-tiedostoon.

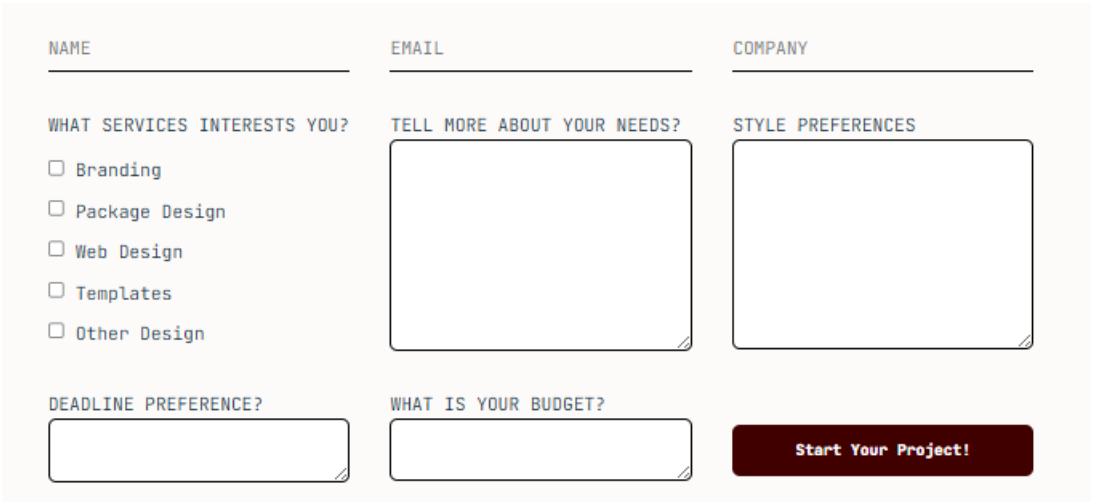
Ulkoasu toteutettiin käyttämällä CSS:ää ja tekstityylit pystyi toteuttamaan samanlaisina kuin Webflowta käytettäessäkin, sillä fontit pystyivät tuomaan linkillä Google Fonteista projektin index.html-tiedostoon, jonka kautta niitä pystyi käyttämään komponenttien tyyleissä.

Erillisistä komponenteista nopeinten toteutettavissa oli palveluvalikko (Kuva 5), yläpalkin navigaatiovalikko sekä footer. Tyylit olivat hyvin toteutettavissa, etenkin, kun pystyi käyttämään tekstityylejä, jotka oli luotu jo aikaisemmin muiden osioiden käyttöön. Eniten aikaa näissä vei tekstien asettelu haluttuun kohtaan ja ylävalikon asettelu responsiiviseksi näytön koon vaihtuessa.



Kuva 5. Palveluvalikko

Haasteita tuotti eniten yhteydenottolomakkeen asettelu, sillä kolmanteen sarakkeeseen ei saanut neljää kohtaa, niin, että lomakkeen koko ulkoasu olisi muuten näyttänyt sopuisalta keskenään. Kuitenkin tyylit sai samanlaiseksi kuin Webflowllakin. Lopulta päädyin sijoittamaan vastauslaatikot hieman eri kohtiin ja erikokoisiksi (kuva 6), jotta lomakkeesta tulisi ulkoisesti selkeän näköinen, vaikkei se täysin vastaakaan aiemmin Webflowlla tehtyä.



Kuva 6. Yhteydenottolomake Reactilla

Lisäksi työtä React-pohjaisessa kehityksessä lisäsi ajoittaiset kirjoitusvirheet ja niiden paikantaminen.

3.3 Kehitysprosessin vertailu

Yksisivuisen nettisivun rakentamiseen, kolmella erikoiselementillä (slider, yhteydenottolomake, palveluvalikko) meni Webflowta käyttäen noin yksi kokonainen työpäivä. Kehitystä helpotti valmiit elementit sekä sivun yläosaan että alaosaan. Samainen sivu Reactilla toteutettuna vaati noin puolitoista työpäivää, sillä elementtien luominen vaati enemmän ajattelua sekä yritystä ja erehdystä. Kirjoittamisen lisäksi kirjoitusvirheet lisäsivät ajankäyttöä React-pohjaisessa projektissa.

Oppimiskynnys Webflown käyttöön on matala. Netissä on tarjolla videoita ja ohjeistuksia työkalun käytöstä sekä Webflow Forum, jossa pystyy kysymään kysymyksiä ja Webflown käyttäjät pystyvät auttamaan ongelmanratkaisussa.

Reactin käyttö sen sijaan tarvitsee vähän enemmän opettelua ja muistikapasiteettia, jotta koodin luominen onnistuu luontevasti. YouTube on täynnä erilaisia videoita, joissa Reactilla rakennetaan jokin kokonaisuus tai pienempi elementti. Nämä videot opettavat paljon, mutta huonona puolena on se, että React päivittyy nopeasti ja jo pari vuotta vanhassa videossa saatetaan käyttää jo vanhentuneita syntakseja. Toisaalta ajankohtaisia tapoja voi etsiä netistä ja esimerkiksi tekoälyn avulla tiedon saa käsiinsä sekunneissa.

Nettisivujen ylläpito onnistuu vaivattomasti Webflowlla ja kokemattomampikin käyttäjä pystyy opettelemaan ylläpitoon ja nettisivun sisällön päivityksiin tarvittavat taidot nopeasti. Reactilla tehtyjen nettisivujen ylläpitoa vaikeuttaa se, että päivittäjän tulisi olla tietoinen tiedostorakenteesta sekä koodin toiminnasta. Reactin versiota päivitetään myös usein ja ongelmaksi saattaa koitua, että elementit lakkaavat toimimasta uudempien päivitysten myötä, joten tarkkuutta vaaditaan enemmän ylläpitäessä koodattua nettisivua.

Webflowlla nettisivujen kehittäminen sekä pienten sivujen julkaisu on ilmaista, kun käytetään webflow.io-domainia. Muuten Webflow tarjoaa kolmea erilaista vaihtoehtoa sivuston suuruuden mukaan. Halvin nettisivuvaihtoehto on maksaa 18 $ per kuukausi, mikäli laskutus tapahtuu kuukausittain ja 14 $ kuukaudessa,

mikäli laskutus tapahtuu vuosittain. Vähän suuremmille sivuille julkaisu ja ylläpitopalvelu maksaa 29 $ / 23 $ per kuukausi ja suurille yli 300-sivuisille nettisivuille julkaisu ja ylläpito maksaa 49 $ / 39 $ kuukaudessa. (Webflow, ei pvm)

Reactin käyttö itsessään on ilmaista, mutta jotta sivun voi julkaista, tarvitaan erillinen hosting-palvelu sekä verkkotunnus. Esimerkiksi GoDaddy-palvelulla pienimpien sivujen hosting, sisältäen verkkotunnuksen, maksaa 9,99 € kuussa, kalleimmillaan isoille sivuille 99,99 € kuukaudessa. (GoDaddy, ei pvm)

Seuraavassa taulukossa (Taulukko 1.) koottu Reactin sekä no-code-/low-code-alustojen ominaisuuksia tehdyn kehitysprosessin pohjalta.

Taulukko 1. Vertailutaulukko Webflow ja React-toteutuksien välillä.

Ominaisuus	Low-code/no-code	React
Kehitysaika	Noin 1 työpäivä. Nopea rakentaa valmiiden komponenttien ja käyttöliittymän avulla.	Noin 1,5 työpäivää. Enemmän suunnittelua, kirjoittamista ja virheiden korjausta.
Oppimiskynnys	Matala, visuaalinen käyttöliittymä, paljon ohjeita ja neuvoja saatavilla.	Keskivaativa, vaatii aiempaa koodausosaamista tai vahvaa perehtymistä projektin alussa.
Haasteet	Monimutkaisempien elementtien mukautus voi vaatia koodia, jota ilmaisversiossa ei voi käyttää.	Syntaksit voivat vanhentua nopeasti, edellyttää versiopäivitysten seurantaa.
Kustannukset	Ilmainen kehitykseen ja pieniin julkaisuihin, muuten 14–49 $/kk riippuen sivuston koosta.	React itsessään ilmainen, hosting + domain noin 10–100 €/kk.
Ylläpito	Helppoa – automaattinen päivitys, ei vaadi teknistä osaamista.	Vaativampaa – koodin ja riippuvuuksien ylläpito käyttäjän vastuulla.
Julkaisu / hosting	Sisältyy Webflown maksullisiin suunnitelmiin.	Vaatii erillisen hosting-palvelun (esim. Netlify, GoDaddy)

4 Verkkosivuteknologioiden vertailu

4.1 Low-code- ja no-code-alustan ja React-kirjaston edut ja haasteet

4.1.1 Low-code- ja no-code-alustat

Low-code- ja no-code-alustat tarjoavat tehokkaan ja käyttäjäystävällisen tavan verkkosivujen toteutukseen, ilman että taustalla tarvitsee olla syvällistä teknillistä osaamista. Yksi suurimmista eduista on kehityksen vauhti, jonka takaavat valmiit komponentit, vedä-ja-pudota-toiminnallisuus sekä selkeä käyttöliittymä. Näiden avulla yksisivuinen verkkosivusto voi valmistua jopa yhden työpäivän aikana kokemattomankin käyttäjän käsissä. Etenkin pienten ja keskisuurten verkkosivuprojektien kohdalla nopeus tuo merkittäviä säästöjä ajassa sekä kustannuksissa.

Matala oppimiskynnys tekee koodittomista ratkaisuista houkuttelevia käyttäjille, jotka eivät ymmärrä ohjelmointia. Verkkosivun voi toteuttaa ja päivittää ilman koodin kirjoittamista, ja alustat tarjoavat runsaasti ohjeita, esimerkkivideoita sekä yhteisön tukemaan oppimista ja ongelmanratkaisua. Visuaalinen kehitysympäristö mahdollistaa ulkoasun säätämisen suoraan käyttöliittymässä ja sivuston muuttumista pystyy seuraamaan reaaliajassa muutoksia tehdessä.

Haittapuolena low-code- ja no-code-ratkaisuissa on rajoitettu mukautettavuus. Erikoistoiminnallisuudet tai räätälöidyt ratkaisut voivat vaatia maksullisia lisäosia tai koodinpätkiä, mikä on hankaluus etenkin koodittomissa alustoissa. Lisäksi alustasidonnaisuus on vahva ja sivuston siirtäminen suoraan toiseen järjestelmään voi olla vaikeaa tai mahdotonta ilman uudelleenrakentamista.

Kustannustasolla maksuttomat versiot riittävät vain pienimuotoiseen käyttöön, ja omien verkkotunnusten sekä laajemman toiminnallisuuden hyödyntäminen vaatii maksullisen tilauksen. Pitkällä aikavälillä nämä kustannukset voivat kasvaa huomattavasti, erityisesti suuremmissa projekteissa.

4.1.2 React-kirjasto

React tarjoaa kehittäjälle täyden hallinnan niin ulkoasun kuin toiminnallisuuksienkin suhteen. Sovellukset ei sisällä valmiita elementtejä, vaan kaikki suunnitellaan ja rakennetaan itse, mikä mahdollistaa täysin räätälöidyt ratkaisut. Yksi Reactin keskeisistä eduista on komponenttipohjainen rakenne, joka mahdollistaa koodin uudelleenkäytön ja nopeuttaa kehitystä erityisesti silloin, kun sivuilla on toistuvia samankaltaisuuksia.

Reactin virtuaalinen DOM-rakenne parantaa suorituskykyä reagoimalla käyttöliittymän muutoksiin nopeasti ja tehtävät päivitykset näkyvät saumattomasti käyttäjälle. Reactin skaalautuvuuden takia se sopii niin pieniin kuin laajoihinkin projekteihin, joissa sivustoa kehitetään ja laajennetaan ajan myötä. Kehittäjäyhteisö on suuri ja tarjolla on lukuisia valmiita kirjastoja, joilla voi lisätä sivuston toiminnallisuuksia tai helpottaa kehitysvaiheita.

Toisaalta Reactin käyttöönotto vaatii enemmän teknistä osaamista. Oppimiskynnys on korkeampi, sillä kehittäjän on hallittava JavaScript sekä Reactin keskeiset käsitteet. Kehitysprosessi voi olla hitaampi, sillä jokainen osa-alue on toteutettava manuaalisesti. Tämä tekee projektista joustavan mutta samalla aikaa vievän.

Ylläpito voi olla haastavaa, koska muutokset edellyttävät yleensä koodin tuntemusta ja teknistä asiantuntemusta. Myös Reactin ja siihen liittyvien kirjastojen jatkuva kehittyminen voi aiheuttaa yhteensopivuusongelmia: päivitykset voivat rikkoa toimintoja tai vaatia vanhan koodin päivittämistä uuteen muotoon.

4.2 Suositukset pk-yrityksille ja verkkokehittäjille

Teknologiavalinta verkkosivuprojektiin tulisi perustua projektin laajuuteen, tavoitteisiin ja käytettävissä oleviin resursseihin. Sekä low-code-/no-code-työkalut sekä React-kirjasto omaavat vahvuutensa, mutta niiden hyödyllisyys vaihtelee projektikohtaisesti.

Koodittomat alustat sopivat etenkin projekteihin, joissa

- tarvitaan vain pieni tai simppeli verkkosivu esimerkiksi yritykselle tai tapahtumalle
- halutaan tehdä työkalu yrityksen sisäiseen käyttöön
- yrityksellä ei ole ohjelmointiosaajaa tai resursseja palkata tai kouluttaa sellaista

Vähäkoodiset alustat tarjoavat selkeästi lisää mahdollisuuksia no-code-alustoihin verrattuna. Ne soveltuvat parhaiten

- Keskisuurille verkkosivuille ja sovelluksille, jotka tarvitsevat mukauttamista, mitä no-code-alustat eivät pysty tarjoamaan
- sovelluksiin, jotka vaativat integraatiota kolmannen osapuolen järjestelmiin
- tilanteisiin, joissa tarvitaan nopeaa kehitysaikataulua, mutta silti skaalautuvaa ja monipuolista alustamallia

React tai muu perinteinen ohjelmointitapa on suositeltava valinta silloin, kun

- käyttöliittymä tulee olemaan räätälöity, interaktiivinen ja monimutkainen, kuten asiakasportaalit tai verkkokaupat
- Nettisivusta ei haluta alustariippuvaista
- Jatkokehitysmahdollisuus halutaan pitää saatavilla
- Yrityksellä on käytettävissään asiantunteva kehitystiimi

5 Lopuksi

Tässä opinnäytetyössä tarkasteltiin verkkosivujen kehittämistä modernien kehitysteknologioiden avulla. Erityisesti vertailtiin low-code- ja no-code-alustojen sekä Reactin käytettävyyttä verkkosivuprojektien toteutuksessa. Työn tavoitteena oli selvittää, millaisiin projekteihin teknologiat soveltuvat parhaiten, millaisia etuja ja haasteita niihin liittyy sekä miten eri vaihtoehdot vaikuttavat kehitysprosessiin, ylläpitoon ja jatkokehitykseen.

Opinnäytetyössä selvisi, että low-code- ja low-code-alustat soveltuvat projekteihin, joissa aikataulu ja budjetti ovat rajallisia eikä syvällistä teknistä osaamista ole käytettävissä. Näillä alustoilla voidaan tuottaa visuaalisesti näyttäviä ja toiminnallisia sivuja, mutta niiden mukautettavuus ja integraatiokyky ovat rajallisia. Alustasidonnaisuus ja tekniset rajoitteet tekevät niistä huonosti soveltuvia projekteihin, jotka edellyttävät jatkuvaa laajentamista tai monimutkaista logiikkaa.

React osoittautui joustavaksi ja skaalautuvaksi teknologiaksi, joka soveltuu erityisesti pitkäikäisiin ja teknisesti haastaviin projekteihin. Sen komponenttipohjainen rakenne tekee koodista uudelleenkäytettävää sekä hallittavampaa.

Tuloksia voidaan hyödyntää yrityksissä, joissa teknologiavalinnat tehdään vajavaisella perehtymisellä vaihtoehtoihin. Työssä laadittiin suosituksia ja vertailupohja siitä, milloin yksinkertainen low-code- tai no-code-ratkaisu riittää ja milloin olisi kannattavampaa käyttää enemmän resursseja teknillisesti vaativampaan toteutukseen perinteisellä ohjelmointityylillä.

Jatkossa olisi hyödyllistä lähteä tutkimaan muita valintapäätökseen vaikuttavia ominaisuuksia, kuten käyttäjäkokemusta, hakukoneoptimoinnin mahdollisuutta sekä turvallisuuskysymyksiä. Mikäli pitkittäistutkimukseen on resursseja, nettisivujen kustannuksellinen vertailu tuottaisi varmasti arvokasta tietoa, etenkin kun tiedetään, että kustannusvaihteluihin voi vaikuttaa teknologioiden kokoaikainen kehitys. Yritysten näkökulmaa voisi syventää haastattelemalla

yrityksiä heidän kokemuksistaan eri teknologioista sekä valintaprosessiin vaikuttavista tekijöistä.

Lähteet

30 Days Coding. (n.d.). *30 Days Coding*. Retrieved Maaliskuu 7, 2025, from https://blogs.30dayscoding.com/blogs/mysql/introduction-to-mysql/overview-of-mysql/history-and-evolution/

Arola, N. (2022, Maaliskuu 12). *Nitro.* Retrieved Maaliskuu 9, 2025, from https://www.nitro.fi/matkapaivakirja/miksi-valita-headless-cms

Berners-Lee, T. (1991, Lokakuu 29). *Web Design Museum*. Retrieved Maaliskuu 7, 2025, from https://www.webdesignmuseum.org/web-design-history/tim-berners-lee-published-a-document-called-html-tags-1991

Bildr.com. (n.d.). Retrieved Huhtikuu 8, 2025, from https://www.bildr.com/

Bubble. (2025, Helmikuu 24). *Bubble.* Retrieved Huhtikuu 6, 2025, from https://bubble.io/blog/best-no-code-app-builder/

Cardillo, A. (2024, Lokakuu 21). *Exploding Topics.* Retrieved Maaliskuu 9, 2025, from https://explodingtopics.com/blog/how-many-websites-on-the-internet

Clark, S. (2024, Toukokuu 28). *CMSWIRE.* Retrieved Huhtikuu 1, 2025, from https://www.cmswire.com/digital-experience/can-traditional-web-development-survive-ai/

Creek, J. (2022, Toukokuu 10). *Medium.* Retrieved Maaliskuu 7, 2025, from https://medium.com/@jacok/javascript-was-invented-by-brendan-eich-in-1995-344d28727f46

Dev, Y. (2021, Elokuu 26). *Dev.* Retrieved Maaliskuu 9, 2025, from https://dev.to/yongdev/brief-history-of-bootstrap-3gi6

Dirox. (2024, Lokakuu 15). *Dirox.* Retrieved Maaliskuu 7, 2025, from https://dirox.com/post/the-many-uses-of-php-history-and-applications

Editoral Staff of Wpbeginner. (2025, Maaliskuu 4). *wpbeginner.* Retrieved Maaliskuu 7, 2025, from https://www.wpbeginner.com/news/the-history-of-wordpress/

Gillen, C. (2025, Syyskuu 27). *Zapier.* Retrieved Maaliskuu 13, 2025, from https://zapier.com/blog/low-code-vs-no-code

GoDaddy. (n.d.). Retrieved toukokuu 16, 2025, from https://www.godaddy.com/en/hosting/web-hosting

Gyo, S. (2019, Elokuu 13). *LogRocket.* Retrieved Maaliskuu 8, 2025, from https://blog.logrocket.com/the-history-and-legacy-of-jquery

Hámori, F. (2024, Toukokuu 30). *RisingStack.* Retrieved Huhtikuu 8, 2025, from https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

Hridoy, R. (2024, Elokuu 12). *DEV.* Retrieved Huhtikuu 1, 2025, from https://dev.to/rashedulhridoy/the-future-of-web-development-emerging-trends-and-technologies-every-developer-should-know-1ceb

hurja. (2024, Kesäkuu 12). *Hurja.* Retrieved Maaliskuu 9, 2025, from https://www.hurja.fi/blogi/pilvipalveluiden-palvelumallit-paas-iaas-baas-ja-saas/

Jednaszewski, M. (2024, Maaliskuu 5). *Mendix.* Retrieved Maaliskuu 12, 2025, from https://www.mendix.com/blog/understand-no-code-vs-low-code-development-tools/#what-is-no-code-development

JeleTele Design. (2024, Joulukuu 12). *JeleTele Design.* Retrieved Huhtikuu 4, 2025, from https://www.jeleteledesign.fi/blogi/verkkosivujen-tulevaisuus-uudet-teknologiat-ja-trendit

Kandukuri, G. (2022, Kesäkuu 9). *Saxon.* Retrieved Huhtikuu 7, 2025, from https://saxon.ai/blogs/power-apps-vs-mendix-vs-outsystems-comparison-of-low-code-development-platforms/

Khan, T., & Downie, A. (2024, Elokuu 16). *IBM.* Retrieved Maaliskuu 12, 2025, from https://www.ibm.com/think/topics/no-code

Mateusz. (n.d.). *Blurify.* Retrieved Huhtikuu 11, 2025, from https://blurify.com/blog/react-js-from-a-developers-point-of-view-pros-cons/

mdn . (2025, Helmikuu 21). *mdn web docs.* Retrieved Huhtikuu 7, 2025, from https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Guides/What_is_a_progressive_web_app

Microsoft. (n. d.). *Microsoft Power Platform.* Retrieved Maaliskuu 19, 2025

Miramar. (n.d.). *Miramar.* Retrieved Huhtikuu 1, 2025, from https://www.miramar-group.co.uk/insights/the-future-of-web-dev/

Modan, S. (2025, Tammikuu 23). *peerbits.* Retrieved Huhtikuu 12, 2025, from https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html

Outsystems. (n.d.). *Outsystems.* Retrieved Maaliskuu 12, 2025, from https://www.outsystems.com/low-code/vs-no-code

Raj, A. (2024, Helmikuu 4). *AlmaBetter.* Retrieved Maaliskuu 7, 2025, from https://www.almabetter.com/bytes/articles/history-of-css

SAP. (n.d.). *SAP.* Retrieved Maaliskuu 12, 2025, from https://www.sap.com/products/technology-platform/build/what-is-low-code-no-code.html

Shaduk, H. (2023, Huhtikuu 6). *Simform.* Retrieved Maaliskuu 9, 2025, from https://www.simform.com/blog/angular-vs-react/

Simplified. (n.d.). *Simplified.* Retrieved Maaliskuu 9, 2025, from https://www.simplified.fi/blogi/mita-on-low-code

UXPin. (2024, Huhtikuu 25). *UXPin.* Retrieved Huhtikuu 8, 2025, from https://www.uxpin.com/studio/blog/what-is-react/

Vadita. (n.d.). *Talent500.* Retrieved Maaliskuu 9, 2025, from https://talent500.com/blog/difference-between-css-and-css3/#include-in-indexing

W3C. (2014, Joulukuu 9). *W3C.* Retrieved Maaliskuu 8, 2025, from https://www.w3.org/TR/html5-diff/

Webflow. (n.d.). Retrieved toukokuu 16, 2025, from https://webflow.com/pricing

WP Marmite. (n.d.). *WP Marmite*. Retrieved Maaliskuu 7, 2025, from https://wpmarmite.com/en/wordpress/history/