jamk.fi

Reaction Commerce – reaktiivinen verkkokauppa-alusta kehittäjän näkökulmasta

Jussi Vesa

Opinnäytetyö
Toukokuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), mediatekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Vesa, Jussi	Opinnäytetyö, AMK	Toukokuu 2018
	Sivumäärä 107	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Reaction Commerce – reaktiivinen verkkokauppa-alusta kehittäjän näkökulmasta

Tutkinto-ohjelma

Mediatekniikan koulutusohjelma

Työn ohjaaja(t)

Kari Niemi

Toimeksiantaja(t)

Jussi Vesa

Tiivistelmä

Yhä useammalla yrityksellä on ymmärrys, kuinka tärkeää hyvä design ja käytettävyys ovat verkkosivustoilla sekä miten ne vaikuttavat etenkin uusiin asiakkaisiin. Verkkokauppa-alustalta vaaditaan paljon, kun katalogissa voi olla tuhansia tuotteita ja palvelun pitäisi ilahduttaa kuluttajaa yksinkertaisuudellaan, nopeudellaan sekä luotettavalla toimivuudellaan.

Opinnäytetyön tehtävänä oli saada selville, kuinka hyvin nykyaikainen verkkokauppa-alusta, Reaction Commerce, suoriutuu yleisimmistä vaatimuksista sekä saada ymmärrys, miten se soveltuu käytettäväksi tuotannossa. Työn konkreettisena tavoitteena oli tuottaa kirjallinen kuvaus verkkokaupan toteutuksesta, sen teknologioista ja pääeroavaisuuksista yleisimpiin verkkokauppa-alustoihin sekä niin sanottu proof of concept -verkkokauppa, joka toteutettiin Reaction Commercella.

Opinnäytetyö osoitti, että Reaction Commerce on tarpeeksi eheä ja luotettava. Alusta on sovelias tuotantokäyttöön, sillä julkaisu tapahtuu alan hyväksi todettujen tapojen mukaisesti, jolloin versiointi ja kaupan julkaisu on toteutettu arkkitehdillisesti turvallisesti konttiympäristön avulla.

Opinnäytetyön myötä voidaan todeta, että Reaction Commerce on reaktiivinen ja reaaliaikainen, nykyaikaisilla teknologioilla ja kattavalla dokumentaatiolla julkaistu verkkokauppa-alusta. Reaction Commerce sisältää kuluttajan ja kaupan hallinnoinnin henkilöiden näkökulmasta kaikki ne ominaisuudet, kuin odotetaan. Kehittäjälle Reaction Commerce on joustava ja teknologisesti moderni sekä laaja. Reaction Commercessa on kuitenkin suorituskyvyn ongelmia kehitystyössä koodimuutosten päivittymisessä ja sivun esilatauksessa on hitautta kuluttajan vieraillessa verkkosivulla. Tuotantojulkaisun muodostaminen on hidasta ja vaatii verrattain tehokkaan kehitysalustan. Ongelmat ovat kuitenkin alustan hyötyihin verrattuna pieniä, ja voidaan todeta, että ne eivät merkittävästi heikennä Reaction Commercen asemaa miljardimyynnin kilpaillussa alalla.

Avainsanat (asiasanat)

Reaction Commerce, verkkokauppa-alusta, reaktiivinen, reaaliaikainen, palvelumuotoilu, design, käyttäjätarina, JavaScript, Node.js, Meteor, React, MongoDB

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Vesa, Jussi	Bachelor's thesis	May 2018
		Language of publication: Finnish
	Number of pages 107	Permission for web publication: x

Title of publication

Reaction Commerce – a Reactive E-Commerce Platform from the Perspective of Developer

Degree programme

Bachelor's Degree Programme of Media Engineering

Supervisor(s)

Kari Niemi

Assigned by

Jussi Vesa

Abstract

Most of today's corporates understand the importance of good design and user experience in a website and their meaning in relation to new customers. There are many requirements for an e-commerce platform, and the task is not easy when a product catalogue might contain thousands of products and the website should please the customers by its responsiveness and reliable usability.

The assignment of the thesis was to examine how well a modern e-commerce platform, Reaction Commerce performs the most common requirements. Additionally, the goal was to gain an understanding how the platform applies to usage in production. The concrete objective of the study was to produce a description of the development of such e-commerce website, its technologies and the main differences compared to most common e-commerce platforms. Finally, a proof of concept e-commerce website was developed with Reaction Commerce.

The research study indicated that Reaction Commerce is sufficiently whole and trustworthy. The platform is applicable to production use because the publishing is made using industry's best practices, where version control and web store publishing are implemented by architecturally secure, containerized environments.

With completion of this thesis it can be stated that Reaction Commerce is a reactive and real-time e-commerce platform developed with today's technologies. It is also well documented as a whole. Reaction Commerce includes all the qualities that can be expected from the customer's and developer's point of view. For developers, Reaction Commerce is a flexible and technologically modern and comprehensive platform. However, Reaction Commerce has its flaws with the performance concerning development where the time-to-effect of the code changes is poor and the time spent on preloading the first meaningful paint for customers is slow. The production build process is slow and requires a powerful development environment. These problems are minor compared to the benefits of Reaction Commerce, and it can be stated that they do not significantly decrease Reaction Commerce's position in the competitive billion-sale industry.

Keywords/tags (keywords)

Reaction Commerce, e-commerce platform, reactive, real-time, service design, design, user story, JavaScript, Node.js, Meteor, React, MongoDB

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Komennot

Sanasto

Angular

Googlen kehittämä TypeScript -pohjainen avoimen lähdekoodin webapplikaatioiden kehitykseen käytettävä ohjelmointikirjasto, joka on julkaistu vuonna 2016. Angular nimi viittaa uudempaan versioon, eikä sitä tule sekoittaa AngularJS:än.

API

Application Programming Interface. Sovelluskehityksen tai sovellusten välisen yhteyden määrittelevä ohjelmointirajapinta.

CDN

Content Delivery Network. Hajautettu sisällön jakamiseen käytetty verkko, joka muodostuu välityspalvelimista. Mahdollistaa optimaalisen kokemuksen kaikille verkkosivun käyttäjillä riippumatta geologisesta sijainnista.

CLI

Command Line Interface. Perinteinen tekstipohjainen komentokehoitteen kautta käytettävä tapa kommunikoida tietokoneen kanssa. CLI on korvattu laajalti nykytietotekniikassa graafisella käyttöliittymällä (ks. GUI).

CMS

Content Management System. Sisällönhallintajärjestelmä tarkoittaa palvelua tai tuotetta, jonka avulla voidaan hallinnoida jonkin järjestelmän sisältöä kuten tekstejä ja mediatiedostoja. Hallinnoinnin laajuus riippuu aina järjestelmästä, ja se voi tukea esimerkiksi lisäyksiä, poistoja, arkistointeja, palautuksia, varmuuskopioita ja versionhallintaa. Web-ympäristöissä sisällönhallintajärjestelmä mahdollistaa sisällön muokkaamisen ilman, että henkilön tarvitsee osata ohjelmoida.

Dependency

Paketinhallintajärjestelmään merkitty riippuvuus johonkin moduuliin, joka on vaadittu, jotta sitä voidaan käyttää.

ECMA International

European Computer Manufacturers Association. Vuonna 1961 perustettu standardiorganisaatio informaatioteknologian ja kommunikointiteknologian järjestelmille.

ECMAScript

Kaupparekisteröity standardoitu ECMA Internationalin omistama skriptikielen spesifikaatio, jonka avulla on standardoitu mm. JavaScript ja ActionScript -skriptikielet.

Funktionaalinen ohjelmointikieli

Deklaratiivinen ohjelmointiajatusmalli, joka käsittelee laskennan arvioinnit matemaattisilla funktioilla ja jossa vältetään datan tilan muutoksia tai muuttuvaa dataa. Ohjelmointi tehdään lausekkeiden tai julistuksen avulla. Funktionaalisessa ohjelmoinnissa ulostuleva arvo on aina riippuvainen funktioon annettavista parametreistä.

Git

Alun perin Linus Torvaldsin luoma vuonna 2005 julkaistu avoimen lähdekoodin versionhallintajärjestelmä (ks. VCS). Hajautettu versionhallintajärjestelmä, jota kuvataan nopeaksi ja skaalautuvaksi, ja esimerkiksi GitHub käyttää sitä.

GitHub

Vuonna 2007 julkaistu suljetun lähdekoodin verkkosivu ohjelmistoprojektien versiohallinnointia varten. Se käyttää Gittiä (ks. Git) versionhallintajärjestelmänä.

Google

Vuonna 1998 perustettu Yhdysvaltalainen teknologia-alan yritys. Googlen menestyneimpiä tuotteita ovat Google -hakukone sekä Android -mobiilikäyttöjärjestelmä.

GUI

Graphical User Interface. Graafinen tapa kommunikoida tietokoneen kanssa. Tekstipohjaiset komennot (ks. CLI) ovat korvattu visuaalisilla elementeillä kuten ikkunoilla, ikoneilla ja napeilla.

Homebrew

Max Howellin vuonna 2009 julkaisema avoimen lähdekoodin paketinhallintajärjestelmä macOS käyttöjärjestelmälle (ks. paketinhallintajärjestelmä).

IDE

Integrated Development Enviorement. Ohjelmisto, joka tarjoaa työkalut kehittää ohjelmistoja. IDE koostuu usein lähdekoodin kirjoittamiseen tarkoitetusta editorista, koonnin automatisoinnista ja vianetsijästä.

Java

Vuonna 1996 julkaistu yleiskäyttöinen oliopohjainen ja luokkia toteuttava ohjelmointikieli. Javaa ei tule sekoittaa JavaScript -skriptausohjelmointikieleen.

JIT

Just In Time. Tapa suorittaa ohjelmakoodia koodin kääntämisen aikana, toisin sanoen ajon aikana, toisin kuin ennen suoritusta. Käytännössä ohjelman lähdekoodi käännetään konekielelle, mikä suoritetaan sen jälkeen suoraan.

JSON

JavaScript Object Notation. Määritelty tapa julkistaa standardin muotoisia objekteja, jota esimerkiksi API voi palauttaa vastausviestissä.

LESS

Leaner Style Sheets. Dynaaminen kielilaajennus CSS-tyylimäärityksille.

Linus Torvalds

Suomenruotsalainen vuonna 1969 syntynyt tietotekniikan ja ohjelmoinnin asiantuntija. Hän on perustanut mm. Linuxin kernelin ja Gitin kehitystyön (ks. Linux, Git).

Linux

Tietokoneen käyttöjärjestelmä, josta on julkaistu useita eri jakeluversioita kuten Ubuntu tai Red Hat Enterprise Linux.

Meteor

Meteor Development Groupin vuonna 2012 julkaisema avoimen lähdekoodin JavaScript webympäristön ohjelmointkirjasto, joka on kirjoitettu Node.js:llä. Meteor tukee samanaikaisesti selain, Android ja iOS alustoja. Web-ympäristössä käytettäessä Meteor voi käyttää käyttöliittymässä Reactia, Angularia tai Meteor Development Groupin omaa käyttöliittymäkirjastoa nimeltä Blaze.

MongoDB

MongoDB Inc. vuonna 2009 julkaisema ilmainen ja avoimen lähdekoodin dokumenttisuuntautunut tietokanta, joka luokitellaan NoSQL-tietokannaksi. MongoDB käyttää JSON-tyylisiä dokumentteja, jotka kuvataan skeemoilla.

Mozilla

Vuonna 1998 julkaistu yhteisö ja teknologiayritys, joka tunnetaan parhaiten Mozillan internet-selaimesta.

Node.js

Vuonna 2009 julkaistu avoimen lähdekoodin alustariippumaton JavaScript suoritusympäristö (*runtime environment*), joka suorittaa JavaScript ohjelmakoodia palvelimella. Node.js on kirjoitettu C, C++ ja JavaScript ohjelmointi- ja skriptauskielillä.

NPM

Node Package Manager (Ks. Paketinhallintajärjestelmä). Tukee Windows, macOS ja Linux käyttöjärjestelmiä.

Ohjelmointirajapinta

Ks. API.

Oliopohjainen ohjelmointikieli

Ohjelmointiajatusmalli, joka perustuu objekteihin, jotka voivat sisältää dataa kussakin kentässä, toisin sanoen attribuuteissa, jotka suorittavat koodia proseduurien eli metodien avulla. Suosituin oliopohjainen ohjelmointi on luokkapohjainen, jossa objektit ovat instansseja luokista, joka määrää niiden tyypin. Esimerkkiohjelmointikieli oliopohjaiselle ohjelmoinnille on Java tai Python.

PaaS

Platform As A Service. Pilvipalvelukokonaisuus, joka mahdollistaa kehityksen, suorituksen ja hallinnoinnin ilman monimutkaista konfiguraatiota ja ylläpitoa liittyen palvelun infrastruktuuriin.

Paketinhallintajärjestelmä

Skemaattinen tapa listata ohjelmistomoduulien ja kirjastojen riippuvuuksia konfiguraatiotiedostoon, jonka avulla paketinhallintajärjestelmä asentaa ja ylläpitää niitä projektikohtaisesti.

Prototyyppipohjainen ohjelmointikieli

Prototyyppipohjainen ohjelmointi on tyyli oliopohjaista ohjelmointia, jossa perinnöllisyys suoritetaan uudelleenkäyttämällä objekteja delegaatin avulla, joka tarjoaa ominaisuudet prototyyppeinä. Prototyyppipohjaista ohjelmointia voidaan myös kuvata "luokattomana" tai "instanssipohjaisena". Esimerkkiohjelmointikieli prototyyppipohjaiselle ohjelmoinnille on JavaScript.

React

Facebookin vuonna 2013 julkaisema avoimen lähdekoodin JavaScript ohjelmointikirjasto käyttöliittymiä varten.

Reaction Commerce

Reaction Commercen avoimen lähdekoodin vuonna 2013 julkaistu JavaScript pohjainen reaaliaikainen verkkokauppa-alusta.

Reaction Platform

Myytävä PaaS -palvelukokonaisuus Reaction Commercen -verkkokauppa-alustaan ja sen infrastruktuuriin.

Reaktiivinen

Ärsykkeeseen tehtävä reaktiona näkyvä vaste, vastavaikutus. Reaktiivisuudella voidaan viitata ohjelmoinnissa datan muuttumisen heijastumiseen käyttöliittymässä reaaliaikaisesti, ilman esimerkiksi verkkosivun päivittämistä.

Riippuvuus

Ks. dependency.

Sisällönhallintajärjestelmä

Ks. CMS.

SQL

Structured Query Language. Vuonna 1974 julkaistu alun perin IBM julkaisema hakukieli, jota käytetään relaatiotietokannoissa tai relaatiodatavirtojen yhteydessä.

Tietokanta

Ohjelmisto, joka säilyttää tietoa. Yleisesti tietokanta on ylläpidossa palvelimella, josta sen tietoa haetaan, lisätään, muokataan tai poistetaan.

TypeScript

Microsoftin vuonna 2012 julkaisema ohjelmointikieli, joka kääntyy luettavaan ja standardipohjaiseen JavaScript koodiin. TypeScript lisää valinnaiset tyypitykset, luokat ja moduulit JavaScript skriptausohjelmointikieleen.

Tyyppijärjestelmä

Ohjelmointikielien tyyppijärjestelmä (*type system*) on joukko sääntöjä, jonka mukaan asetetaan tyyppiominaisuus eri konstruktoreissa, kuten muuttujiin, funktioihin tai moduuleihin. Nämä tyypitykset muodollistavat ja pakottavat implisiittiset luokitukset datamalleille ja komponenteille. Tyypityksiä voi olla esimerkiksi kerronta "funktio palauttaa merkkijonon" tai "taulukko Autot sisältää objekteja ja objektilla on kenttä vuosimalli, jonka tyyppi on kokonaisnumero".

VCS

Version Control System. Versionhallintajärjestelmä tallentaa ja ylläpitää muutoksia, jotta esim. ohjelmistokehityksen aikana voidaan palata aiempaan versioon tiedostosta tai joukosta tiedostoja. Lisäksi versionhallintajärjestelmä mahdollistaa tiedostojen vertailun.

Versionhallintajärjestelmä

Ks. VCS.

WooCommerce

Automatticin vuonna 2011 julkaisema avoimen lähdekoodin verkkokauppa-alusta, joka voidaan asentaa lisäosana WordPressiin.

WordPress

WordPress Foundationin vuonna 2003 julkaisema avoimen lähdekoodin sisällönhallintajärjestelmä, joka on kirjoitettu PHP:llä, joka käyttää MySQL-tietokantaa. WordPress on suosittu yleisesti verkkosivujen ja blogien, mutta myös verkkokauppojen perustana, ja yli 60 miljoonaa verkkosivua käyttää sitä.

XCode

Apple Inc. Julkaisema suljetun lähdekoodin ilmainen IDE (ks. IDE) macOS käyttöjärjestelmälle, jonka avulla voidaan kehittää macOS, iOS, tvOS ja watchOS -applikaatioita.

1 Työn lähtökohdat

1.1 Taustaa

Palveluita tai tuotteita hakevat asiakkaat ovat vapaita liikkumaan yrityksen sivulta toiselle sivulle ilman lisäkustannuksia, mikä johtaa siihen, että verkkosivujen kilpailu kiihtyy ja useammalla yrityksellä on ymmärrys, kuinka tärkeää hyvä design ja käytettävyys ovat sivustolla. Verkkosivut ovat yritysten väylä nykyisten sekä uusien potentiaalisten asiakkaiden välillä (Jenamani, Mohapatra & Ghose 2006, 249). Yhä useampi yritys myy ja markkinoi tuotteitaan tai palvelujaan internetissä, ja iso osa toimijoista käyttää työkaluna nykyisin totuttuun tapaan verkkokauppaa, joka on kuitenkin uusi kanava jo internetinkin lyhyessä historiassa. Verkkokauppa on parhaimmillaan yrityksen brändin ja imagon jatke. Se kantaa monen yrityksen päätoimen eli taloudellisen voiton saavuttamisen, lisää tuotteiden houkuttelevuutta ja ennen kaikkea saatavuutta. Vuoden 2015 marraskuussa koko Euroopan Unionin kansalaisista 79,3 %:lla eli noin 402 tuhannella ihmisellä oli pääsy internettiin, joten myös verkkokauppojen käyttäjäkunnan voidaan ajatella olevan yhä suurempi kuin koskaan aiemmin (Hudák, Kianičková & Madleňák 2017, 343).

Kuluttajan käyttäytymistä verkkosivuilla on monesti tutkittu. Sivun vaste tulee olla äärimmäisen nopea, sillä sekunnin kymmenyksetkin merkitsevät ja pitkällä vasteajalla on suora yhteys menetettyyn taloudelliseen tuottoon (Li, Zhou & Nie 2016, 926). Tuotteiden tulee löytyä helposti, tai muuten asiakas vaihtaa kauppaa. Käyttöliittymän on oltava helposti muistettava ja opittava sekä käyttökokemuksen on oltava positiivinen, tai muuten asiakas voi ärtyä eikä enää palaa sivustolle. (Nielsen & Norman 2000, 65.) Kun ajatellaan edellä mainittua käyttäjää, huomataan, että verkkokauppa-alustalta vaaditaan paljon. Tehtävä ei ole helppo, kun katalogi voi olla sadoista aina tuhansiin tuotteisiin ja palvelun pitäisi ilahduttaa kuluttajaa yksinkertaisuudellaan, nopeudellaan ja luotettavalla toimivuudellaan. Vaikkakin hyvä design

auttaa saavuttamaan isoja määriä vierailijoita verkkosivulle, personoitu kokemus auttaa pitämään asiakkaita merkittävämmän ajan sivustolla, jolloin hyvä design ja yksilöllinen käyttäjäkokemus ovat eduksi toisilleen. (Jenamani ym. 2006, 249–250.) Kun huomioidaan nämä seikat, tehtävä vaikeutuu entisestään. Opinnäytetyössä selvitettiin uudenlaista reaktiivista verkkokauppa-alustaa ja sen soveltavuutta nykypäivän kuluttajan tarpeisiin sekä teknologian asettamiin haasteisiin.

Opinnäytetyön aihe on liiketoiminnan näkökulmasta vääjäämättömän nykyaikainen. Globaali verkkokauppamyynti ylitti 1,5 biljoonaa eli 1,5 tuhatta miljardia USA:n dollaria myynnissä jo neljä vuotta sitten. Tästä huolimatta yritysten verkkosivut eivät aina vastaa sitä laatua ja innovatiivisuutta, mitä voisi odottaa kilpaillessa suurista taloudellisista tuotoista, kun mahdollisuus on tavoittaa asiakkaita ja kuluttajia vuoden jokaisen päivän jokaisella tunnilla. (Cassidy & Hamilton 2016, 1054.) Näistä lähtökohdista opinnäytetyöntekijä motivoitui pyrkimään kriittisesti ja laadukkaasti tutkimaan aihetta niin teoreettisesta kuin käytännönläheisestä näkökulmasta. Opinnäytetyön toimeksiantajana oli opinnäytetyöntekijä, sillä työn aihe muodostui suuresta kiinnostuksesta uuteen teknologiaan ja toisaalta tekijän näkemyksestä hyödyntää työn tuloksia tulevaisuudessa.

1.2 Tehtävä ja tavoitteet

Opinnäytetyön tehtävänä oli saada selville, kuinka hyvin nykyaikainen verkkokauppa-alusta suoriutuu yleisimmistä vaatimuksista sekä saada ymmärrys, miten se soveltuu käytettäväksi tuotannossa. Opinnäytetyötekijä halusi saada yleiskuvan siitä, kuinka Reaction Commerce eroaa perinteisistä verkkokauppa-alustoista, kuten mitkä ovat sen edut ja missä asioissa alusta epäonnistuu. Työn tehtävä on jaettu kahteen näkökulmaan: kehittäjän sekä kuluttajan. Työssä haluttiin tarkastella, millaista kehitystyö Reaction Commercella tehdessä on kehittäjän näkökulmasta, kuten millaisilla teknologoilla ja työkaluilla kehitystyö tehdään, kuinka verkkokauppa-alustaa voidaan testata sekä miten stabiili sen toiminta on. Kehittäjän näkökulman lisäksi tarkasteltiin,

kuinka hyvin alusta suoriutuu eri osa-alueilla kuluttajan näkökulmasta, kuten nopeudessa ja käytettävyydessä.

Opinnäytetyön tavoitteena oli selvittää soveltavan tutkimuksen avulla kriittisesti nykyaikaista modernia avoimen lähdekoodin verkkokauppa-alustaa nimeltä Reaction Commerce sen sijaan, että työssä tutkittaisiin perinteisempiä ja vanhempia teknologioita kuten WordPress-pohjaisia verkkokauppa-alustoja, esimerkiksi WooCommercea. Työn konkreettisena tavoitteena oli tuottaa kirjallinen kuvaus reaktiivisen verkkokaupan toteutuksesta, sen teknologioista ja pääeroavaisuuksista yleisimpiin verkkokauppa-alustoihin sekä niin sanottu *proof of concept* -verkkokauppa, joka toteutettiin Reaction Commercella.

1.3 Soveltava tutkimustyö tutkimusmenetelmänä

Opinnäytetyön tutkimusmenetelmä oli soveltava tutkimustyö. Soveltava tutkimus on uuden tieteellisen tiedon etsintää, jossa sen tulokset ovat suoranaisesti yhteydessä käytännön asioihin (Toikko & Rantanen 2009, 18). Soveltavalla tutkimustyöllä pyritään löytämään ratkaisu käsillä olevaan ongelmaan tai haasteeseen, joka eroaa perinteisestä tutkimuksesta, jossa keskitytään yleistämiseen ja teorioiden muodostamiseen. Soveltavan tutkimustyön tuloksesta ei välttämättä synny uutta tieteellistä tietoa. Soveltavan tutkimuksen heikkous on sen tuloksien muoto, jossa tuloksia ei voi yleistää eli tulokset ja tuotettu tieto ovat aina suhteessa tutkimuskysymykseen tai haasteeseen. (Dudovskiy n.d.)

Soveltavassa tutkimustyössä, kuten muissakin tutkimustöissä voidaan hyödyntää kehitystoimintaa. Kehittämistoiminnan prosessi koostuu tehtäväkokonaisuuksista, joita ovat perustelu, organisointi, toteutus, levittäminen ja arviointi. Perusteluilla tarkoitetaan, mitä ja miksi kehitetään sekä toiminnan organisoinnilla, kuka tekee ja mitäkin sekä millä resursseilla. Toteutus tarkoittaa työn konkreettista tekemistä, johon liittyy keskeisesti arviointi. Levittämisellä tarkoitetaan tuotoksen ja palveluiden levittämisen ja tietoisuuden lisäämistä. (Toikko & Rantanen 2009, 56–57.)

Kehittämistoiminnassa on aina tavoite ja kohde, jossa tavoitellaan muutosta parempaan tai tehokkaampaan tapaan kuin aiemmin. Tavoitteellisuus on keskeinen aihe kehittämisessä. Kehittämistoiminta voi kohdistua yksittäiseen toimihenkilöön tai suureen organisaatioon, yhteen tuotteeseen tai kokonaiseen tuoteperheeseen, palveluun tai asiantuntijuuteen. (Mts. 14–17.) Opinnäytetyön tavoite kohdistuu työn tuottajaan ja Reaction Commerce tuotteeseen, ja siinä tavoitteena on saavuttaa parempi verkkokauppakokemus, niin kehittäjän kuin kuluttajan näkökulmasta.

Kehittämistoiminnassa usein kuvataan sisäinen ja ulkoinen tekijä. Ulkoisia tekijöitä voivat olla esimerkiksi markkina tai kilpailukyky, kun taas sisäinen tekijä voi olla esimerkiksi yrityksen palvelun parantamisen tarve tai työprosessin kehitys. (Mts. 18.) Työn toimeksiantajan ollessa työn tekijä itse, on tämän soveltavan tutkimuksen kehittämistoiminnan tekijätyyppi sisäinen, jossa pyritään parantamaan verkkokauppojen kehityksen prosessia sekä tulevaisuudessa mahdollisesti tarjottavaa palvelua koskien Reaction Commerce -tuotteen konsultointia ja tuotantoa.

Ulkoisten ja sisäisten tekijöiden lisäksi kehittämistoiminnassa määritetään lähtökohdat sekä oleellisessa osassa oleva kysymys, miksi kehitystä tarvitaan. Lähtökohtana voi olla negatiivinen tai positiivinen asetelma, jossa negatiivinen kuvaa nykyistä vallitsevaa ongelmaa ja positiivinen toisaalta tulevaisuuden ihannekuvaa. Usein dynaaminen kehittäminen sisältää molempia lähtökohtia. Toteutukselle on eduksi, jos työn tavoite on mahdollisimman konkreettinen ja konkreettisia tavoitteita voi olla useita, verrattuna perinteiseen tutkimukseen, jossa tavoitteita on useimmin yksi. (Mts. 57–59.) Opinnäytetyön lähtökohtana on aikaisempi ohjelmointitausta sekä kokemus WordPressin kanssa toimivasta WooCommerce -verkkokauppa-alustasta. Kehittämistoiminnan kysymys hakee vastausta siihen, että miksi Reaction Commercea tarvitaan, kun on olemassa paljon muitakin verkkokauppa-alustoja. Työn lähtökohtana on dynaaminen asetelma, sillä negatiivisessa osassa on nykyisten verkkokauppa-alustojen kankeus ja vanhanaikaisuus sekä toisaalta positiivisessa osassa ihannekuva, jossa Reaction Commerce saavuttaa suuren suosion ja siitä tietäville osaajille on kysyntää tulevaisuudessa.

Kehittämistoiminnassa on käytössä erilaisia malleja, kuten lineaarimalli ja spiraalimalli. Lineaarisessa mallissa kehittämistoiminnan tehtäväkokonaisuudet hahmotetaan lineaarisen mallin avulla, jossa prosessi kuvataan vaiheittain, kuten tavoitteen määrittely, suunnittelu, toteutus ja päättäminen. Spiraalimallissa eteneminen hahmotetaan jatkuvana syklinä, jossa spiraali sisältää useita peräkkäisiä osatoteutuksia, toisin sanoen kehiä. (Mts. 64–66.) Opinnäytetyön kehittämistoiminnan malli oli spiraalimalli, jossa työssä oli tehtäväkokonaisuus eli tavoite, suunnitelma ja toteutus. Työn tulosten odotetaan olevan kelpoisia jatkokehitykseen, ja työn nähdään jatkuvan varsinaisen opinnäytetyön päätyttyä sekä uusia syklejä syntyy tulevaisuudessa aiheen parissa lisää.

Kehittämistoimintaa arvioidaan ja sen tavoitteena on antaa prosessille suunta. Yksinkertaisimmillaan arviointia tehdessä analysoidaan, kuinka hyvin kehittämistyö on saavuttanut sen tarkoituksen. Lisäksi arvioinnissa tuotetaan tietoa kehitetyn asian hyvyydestä ja pyritään osoittamaan sen toimivuus. (Mts. 61.)

1.4 Rajaukset

Opinnäytetyön laajuus rajoitettiin tarkoituksenmukaisesti sille tasolle, joka mahdollistaa tarpeeksi tarkan tutkimuksen saavuttamisen kapeammalla laajuudella sen sijaan, että työssä pyrittäisiin tutkimaan kaikkia aiheen teknologioita ja käsitteitä. Tekijä halusi rajata työn keskittymään tiukasti Reaction Commerceen ja sitä selvitettiin ainoastaan yleisimpiin verkkokauppa-alustoihin verraten, kuitenkin siten, että pääfokus on Reactionissa.

Työn tietoperustan laajuus pidettiin suhteessa Reaction Commercen teknologioihin ja työn lähtökohdissa asetettuihin lähtökohtiin ja tavoitteisiin. Lisäksi tietoperustassa käsitellään yleisellä tasolla verkkokauppaa sekä asiakaskokemuksen käsitteitä kuten käyttöliittymäsuunnittelu, muotoilu ja käyttäjäkokemus. Opinnäytetyötekijä pyrki

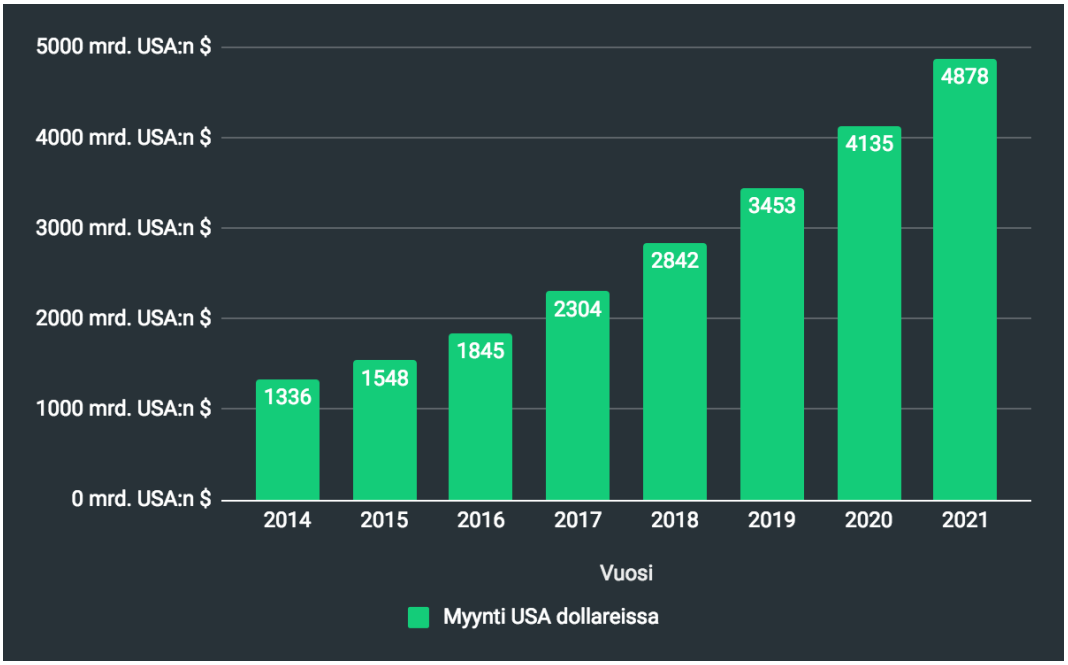
näillä rajoituksilla takaamaan tarpeeksi laadukkaan ja kriittisen soveltavan tutkimuksen saavuttamisen keskittymällä pääpainotteisesti Reaction Commercen ominaisuuksiin.

2 Verkkokauppa, teknologiat ja muotoilu

2.1 Verkkokauppa

2.1.1 Verkkokaupan myynti

Vähittäismyynti bisnekseltä kuluttajalle, eli B2C, on suuressa nousussa maailmanlaajuisesti. Vuosittainen maailmanlaajuinen verkkokauppamyynti B2C arvioidaan kasvavan (ks. kuvio 1). Vuonna 2014 verkkokauppojen myynti oli noin 1,34 miljardia USA:n dollaria ja sen arvioidaan kasvavan noin kaksinkertaiseksi seuraavan kolmen vuoden aikana. (Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars) 2018.)



Kuvio 1. Vuosittainen verkkokauppojen myynnin arvio B2C vuonna 2014–2021 (Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars) 2018.)

Verkkokauppa-alusta Shopify on ilmoittanut myyneensä heidän alustan avulla yhteensä yli 55 miljardilla USA:n dollarilla vuodesta 2006 nykyhetkeen (About Shopify n.d.) Myös Magento on ilmoittanut, että Magento Commerce -alustan vuonna 2018 myynti sen välityksellä oli 124 miljardin USA:n dollaria (Magento Commerce Investment in Europe Accelerates Growth 2018.) Demandwaren vuoden 2015 myynti oli 237 miljoonaa USA:n dollaria (Demandware Revenue, Profits - DWRE Annual Income Statement 2016.) Volusion on tehnyt 17 miljardin USA:n dollarin edestä myyntiä vuodesta 1999 alkaen (Volusion Secures $55 Million in Financing to Accelerate Growth Plans 2015.) Valitettavasti kaikkien luvuissa 2.1.2–2.1.2 esiintyvien verkkokauppa-alustojen myynnin tietoja ei löytynyt. Verkkokauppa-alustat ja niiden takana olevat yritykset liikuttavat kuitenkin paljon rahaa. Luvuissa 2.1.2–2.1.3 tutustutaan niihin tarkemmin.

2.1.2 Yleisimmät verkkokauppa-alustat

Verkkokauppa-alustojen suosio

Verkkokaupan toteutukseen on yli 120 palvelua, tuotetta ja alustaa joko kokonaisen verkkokauppa-alustan tai pelkän ostoskorin toteutukseen (Zorzini 2015). Suosituimmissa verkkokaupoissa on käytössä kokonaisvaltaisia verkkokauppa-alustoja useita kymmeniä. Lisäksi kaupoissa on käytössä myös laajasti erillisiä verkkokaupan ostoskorin toteutukseen käytettäviä tuotteita. (Ecommerce Usage Statistics 2018.)

Taulukot 1–4 sisältävät koko verkkokauppateknologioiden alaosiot, kuten maksulliset verkkokaupparatkaisut, avoimen lähdekoodin verkkokauppa-alustat, plugin ja moduuli -ratkaisut sekä valmiit ylläpidolliset ratkaisut kuten pilvipalvelut. Tarkastellessa 10 000 suosituimman verkkokauppasivun (ks. taulukko 1) teknologioita huomataan, että kaikki muut teknologiat vievät suurimman osan käyttöasteesta verrattuna muihin verkkokauppasivuihin, etenkin koko internetiä tarkastellessa (ks. taulukko 4). Muiden teknologioiden suosio on laskenut 57 %:sta huomattavasti 45 %:iin, eli laskua on ollut 12 %. Merkittävää on myös, että WooCommercen suosio on kasvanut 10 %:sta kolminkertaiseksi eli 30 %:iin, kun tarkastellaan suosituimpia 1 000 000 tai koko internetin verkkokauppoja (ks. taulukko 1, taulukko 3 & taulukko 4). (Mt.)

Taulukko 1. Suosituimman 10 000 verkkokauppasivun verkkokauppateknologiat maailmanlaajuisesti 26. maaliskuuta 2018 (Ecommerce Usage Statistics 2018).

%	Verkkokauppa teknologia
57	Muut yhteensä
13	Shopify (10 %), Shopify Product Reviews (2 %), MailChimp for Shopify (1 %)
11	Magento (7 %), Magento Enterprise (4 %)
10	WooCommerce (7 %), WooCommerce Checkout (3 %)
6	Demandware
1	BigCommerce
1	Volusion

Shopify sen sijaan on pitänyt suosionsa vakaana 12–13 %:n luokassa aina 1 000 000 suosituimpaan verkkokauppaan asti, joskin sen käyttö on laskenut koko internetin verkkokaupoissa 8 %:iin (ks. taulukko 2, taulukko 3 & taulukko 4). Shopifyn suosio on siis tasaisen vahvaa. Sen sijaan Magenton käyttö laskee 11–13 %:n käytöstä vain 4 %:iin, joten sen käyttö laskee yli kolminkertaisesti. (Mt.)

Taulukko 2. Suosituimman 100 000 verkkokauppasivun verkkokauppateknologiat maailmanlaajuisesti 26. maaliskuuta 2018 (Ecommerce Usage Statistics 2018).

%	Verkkokauppa teknologia
49	Muut yhteensä
21	WooCommerce (12 %), WooCommerce Checkout (9 %)
13	Magento (10 %), Magento Enterprise (3 %)
12	Shopify (7 %), Shopify Product Reviews (3 %), MailChimp for Shopify (2 %)
2	BigCommerce
2	DemandWare
1	Volusion

BigCommerce kasvattaa suosiotaan, kun tarkastellaan sen käyttöä kaikkein suosituimmista verkkokaupoista, jossa sen käyttö on ollut 1 %, verrattuna suosituimpaan 1 000 000 verkkokauppaan, jolloin se oli vielä 3 %. (Mt.)

Taulukko 3. Suosituimman 1 000 000 verkkokauppasivun verkkokauppateknologiat maailmanlaajuisesti 26. maaliskuuta 2018 (Ecommerce Usage Statistics 2018).

%	Verkkokauppa teknologia
45	Muut yhteensä
30	WooCommerce (19 %), WooCommerce Checkout (11 %)
9	Magento (8 %), Magento Enterprise (1 %)
12	Shopify (7 %), Shopify Product Reviews (2 %), MailChimp for Shopify (3 %)
3	BigCommerce
1	Volusion
0	DemandWare

Demandwaren kehitys on jatkuvan negatiivinen, jossa sen suosio 10 000 suosituimman verkkokaupan käytössä oli 6 %, mutta laskee nopeasti 2 %:iin ja aina 0 %:iin asti eli käyttö laskee huonoimmillaan kuusinkertaisesti. Tilanne näyttää huonoimmalta Volusionin kohdalla, kun sen käyttö on tasaisen heikkoa kauttaaltaan eli 1–0 %:n luokkaa. (Mt.)

Taulukko 4. Koko internetin verkkokauppasivun verkkokauppateknologiat maailmanlaajuisesti 26. maaliskuuta 2018 (Ecommerce Usage Statistics 2018).

%	Verkkokauppa teknologia
45	Muut yhteensä
30	WooCommerce (42 %), WooCommerce Checkout (0 %)
4	Magento (4 %), Magento Enterprise (0 %)
8	Shopify (7 %), Shopify Product Reviews (1 %), MailChimp for Shopify (0 %)
1	BigCommerce
0	Volusion
0	DemandWare

Lisäksi taulukoista huomataan, että teknologioita on paljon, ja suosituimmiksi yksittäisiksi ratkaisuiksi nousevat WooCommerce, Magento ja Shopify. Isoimman osan ratkaisuista vievät kuitenkin kaikki muut pienemmät palvelut ja itse tehdyt ohjelmistot yhteensä laskettuna. (Mt.)

Muut verkkokauppa-alustat ja teknologiat

Suosituimpien yksittäisen verkkokauppateknologioiden vierellä palvelee useita muita tuotteita ja ratkaisuja. Verkkokaupat koostuvat monesti useammasta verkkokauppateknologian moduulista, jossa esimerkiksi ostoskori, maksutapahtumat, tilausjärjestelmä ja tuotteiden hallinta voivat kaikki olla omia kokonaisuuksiaan, jotka muodostavat kaupan. Ne ovat usein maksullisia perustuen kuukausihintaan ja/tai yksittäiseen maksuun per myynti.

Näitä maksullisia teknologioita ovat esimerkiksi globaalien vaatteiden alan verkkokaupoissa seuraavia:

- **Stripe**, joka tarjoaa maksamiseen tarvittavat teknologiat

- **MailChimp**, joka tarjoaa sähköpostituksen ja asiakaspalvelun
- **MouseFlow**, **Conversio**, **Beeketing** tai **Bazaarvoice**, jotka tarjoavat analytiikan palveluita
- **Tealium**, joka tarjoaa analytiikan ja markkinoinnin integraation
- **Amazon CloudFront**, joka tarjoaa CDN palveluita verkkokaupan palvelemiseksi
- sekä **3D Cart**, joka tarjoaa valmiit ylläpidolliset toteutukset verkkokaupalle. (Global Online Clothing Stores n.d.)

Muihin kokonaisvaltaisiin verkkokauppa-alustoihin kuuluu esimerkiksi:

- **LemonStand**, joka on maksullinen ja suljetun lähdekoodin tuote
- **Squarespace Commerce**, joka on maksullinen pilvipalvelu
- **PinnacleCart**, joka on maksullinen pilvipalvelu
- **Big Cartel**, joka on ilmainen, jossa lisäpalveluista maksetaan, sekä se on suljetun lähdekoodin tuote
- **Ecwid**, joka on ilmainen ja, jonka lisäpalveluista maksetaan, sekä on suljetun lähdekoodin tuote
- **Merchium**, joka on maksullinen pilvipalvelu
- **SellBeing**, joka on maksullinen pilvipalvelu sekä
- **Sylius**, joka on avoimen lähdekoodin ilmainen tuote, ja yritys tarjoaa lisäksi maksullisia palveluita ja konsultointia. (Zorzini 2015.)

WooCommerce

WooCommercen idea sai alkunsa vuonna 2007, kun kolme internet-tuttua Mark, Magnus ja Adii alkoivat suunnitella sitä. Ennen WooCommercen syntyä he myivät maksullisia WordPress-teemoja, ja vuonna 2011 heidän 100:nnen maksullisen teeman jälkeen he julkistivat WooCommercen. (About WooCommerce n.d.)

WooCommerce on avoimen lähdekoodin verkkokauppa-alusta, joka vaatii toimiakseen WordPress CMS verkkosivun. WooCommerce on siis asennettava lisäosa WordPressiin, ja WooCommerceen itseensä on saatavilla yli virallista 400 laajennusta. Laajennusten lisäksi siihen on saatavilla teemoja, joiden avulla uuden verkkokaupan kehitys nopeutuu. Sen avoimen lähdekoodin ansiosta kehittäjä voi muokata mitä vain sen osaa niin halutessaan. Se tarjoaa kaikki verkkokaupan vaatimat ominaisuudet, kuten digitalisten, tilauspohjaisten sekä fyysisten tuotteiden myymisen, tuotearvostelut, tilaushistorian, ostoskorin, verotus asetukset, alennuskupongit, postituksen ja toimituksen asiat sekä tilauksien hoidon. (Create the online store you want n.d.)

WooCommere on kirjoitettu PHP -palvelinohjelmointikielellä, kuten WooCommercen kanssa käytettävä WordPress on myös kirjoitettu. WooCommercen lähdekoodista noin 82 % on PHP koodia, noin 14,5 % JavaScriptiä ja loput noin 3 % CSS -tyylimäärityksiä (An open source eCommerce plugin for WordPress 2018).

Kehittäjän näkökulmasta WooCommerce on yksinkertainen ottaa käyttöön, sillä sen voi lisätä suoraan WordPressin kautta löytyvästä lisäosa-kaupasta, jonka jälkeen sen lähdekoodi latautuu projektin tiedostoihin. Sen jälkeen kehittäjällä on vapaat kädet muokata sitä parhaaksi näkemällään tavalla. Kehittäjän tulee kuitenkin huomata, ettei kriittisiä osia siitä tulisi muokata. Kriittisen osien muutokset tulevat rikkoutumaan, jos WooCommercea päivitetään, ja jos päivitys asennetaan käyttöön.

Magento

Vuonna 1985 perustettu yritys Varien muutti nimensä vuonna 2010 Magento Inc., joka tarjoaa ja kehittää verkkokaupan palveluita sekä tuotteita. Palvelut kattavat fyysiset palvelut kuten myymisen integraatiopalvelut, toimitukset, bisnesmallit, digitaliset tuotteet kuten avoimen lähdekoodin verkkokauppa-alustan nimeltä Magento sekä pilvipalvelukokonaisuuden palvelun Magento Commerce. (Internet Software and Services n.d.)

Magenton ollessa avoimen lähdekoodin verkkokauppa-alusta, on sen käyttäminen ilmaista CE eli community edition versiolla. Kuitenkin suuri osa lisäosista ovat maksullisia, jonka lisäksi verkkokaupan ylläpito maksaa, kuten muissa ei-pilvipalvelullisissa kokonaisuuksissa. Lisäksi Magento tarjoaa EE eli enterprise edition version, joka on puolestaan maksullinen. (Desyatnyuk 2018.)

Magento tarjoaa hyödyllisiä ominaisuuksia kuten yhden sivun maksu-tilaus kokemuksen, joka helpottaa maksutapahtumaa. Tuotteille voi asettaa omia attribuutteja tuotekohtaisesti, joka mahdollistaa erilaisten tuotteiden myymisen. Lisäksi kehittäjän näkökulmasta hyvä puoli Magentossa on mahdollisuus jakaa useita verkkokauppoja samalla yhdellä järjestelmällä. Ominaisuuksiin kuuluu myös raportoinnin työkalut Google Analytics -integraatiolla, tuotteiden tuotekuvien zoomaamisen, toivelistat ja sähköpostilistat, joukkopäivitykset, hakukoneoptimoinnit, hinnoittelun työkalut sekä posti- ja maksutapahtumien väylien integraatiomahdollisuudet. (Magento Community edition features n.d.)

Magenton uusimman major -version, eli versio 2 on kirjoitettu noin 80 % PHP -palvelinohjelmointikielellä ja noin 6 % JavaScriptillä. Loput 10 % on HTML -merkkaustekstiä ja noin 4 % CSS tyylimäärityksiä. (Magento 2 2018.)

WooCommercen tapaan kehittäjällä on suhteellisen helppo tehtävä aloittaa Magentolla kehitystyö lataamalla sen heidän verkkosivuilta tai esimerkiksi GitHubista. Tämän jälkeen kehittäjällä on mahdollisuudet muokata verkkokauppa-alustaa, miten haluaa.

Shopify

Shopify on vuonna 2006 julkaistu pilvipalvelu- ja monikanavainen verkkokauppa-alusta. Monikanavaisuus tarkoittaa sitä, että Shopify tukee ja tarjoaa palveluita verkkokauppoihin, mobiiliapplikaatioihin, sosiaaliseen mediaan kuten Instagram, toreille, kivijalkaliikkeisiin ja pop-up kauppoihin. (About Shopify n.d.)

Myös Shopify tarjoaa vastaavia ominaisuuksia kuin WooCommerce ja Magento. Shopify mahdollistaa uuden verkkokaupan perustamisen valmiin teeman avulla, joita

on yli 100. Kaupan ulkoasua voi muokata täysin, joko palvelun käyttöliittymän tai koodin avulla. Alustaan voidaan lisätä uusia myynnin kanavia, tuotteita voi hallita, tilauksen tekeminen toimii Magenton tapaan yhden sivun kautta sekä tilauksia voidaan seurata ja hallinnoida. (Everything you need to start an online store and sell online n.d.)

Shopify tukee myös korttimaksuja sekä sen kanssa voidaan käyttää kolmannen osapuolen maksuväyliä. Myös lisäosat tai Shopifyn nimeämät *appsit* ovat tuettuja, ja niitä on yli 2000 tarjolla. Esimerkiksi MailChimp on tuettu, joka on mainittuna aiemmin tässä luvussa. (Mt.)

Shopifyn tuotteiden ollessa suljetun lähdekoodin pilvipalveluita, tarkoittaa se, ettei kehittäjillä ole suoraan mahdollisuutta tai toisaalta tarvetta ohjelmoida kaupan logiikkaa. Kaupan teemoja voi kehittäjä muokata ohjelmoinnin avulla sekä käyttöliittymän kautta Shopifyn palvelussa. (Mt.)

Shopify tarjoaa ohjelmointirajapintoja, joiden avulla voidaan ohjelmoida täysin omanlaisia kauppoja ja ostokokemuksia käyttäen Shopifyn palveluja. Shopify tarjoaa kaksi tärkeintä rajapintaa näihin toteutuksiin, eli Storefront API sekä Admin API. Storefront API:n avulla voidaan kehittää omanlaisia ostokokemuksia. Storefront API:n avulla voidaan hakea tuotteita, luoda maksu-, ostoskori-, ja tilauskokemuksia, luoda ja poistaa asiakkuuksia eli käyttäjätilejä sekä mahdollistaa kustomoidut tuotevalinnat. (Storefront API n.d.)

Admin API tarjoaa integraatiomahdollisuudet Shopifyn pääkäyttäjän hallintapaneeliin. Sen avulla voidaan tehdä kokonaisia applikaatioita, jolla hallitaan kaikkea Shopifyn kauppaan ja sen hallinnointiin liittyviä asioita, kuten:

- tuotteet ja katalogi, sekä lahjakortit
- myynnin kanavat
- postitus ja laskutus
- kaupat, kuten sijainti ja kaupan tapahtumat
- analytiikka, kampanjat ja alennukset sekä

- inventaarion hallinnointi. (Admin API reference n.d.)

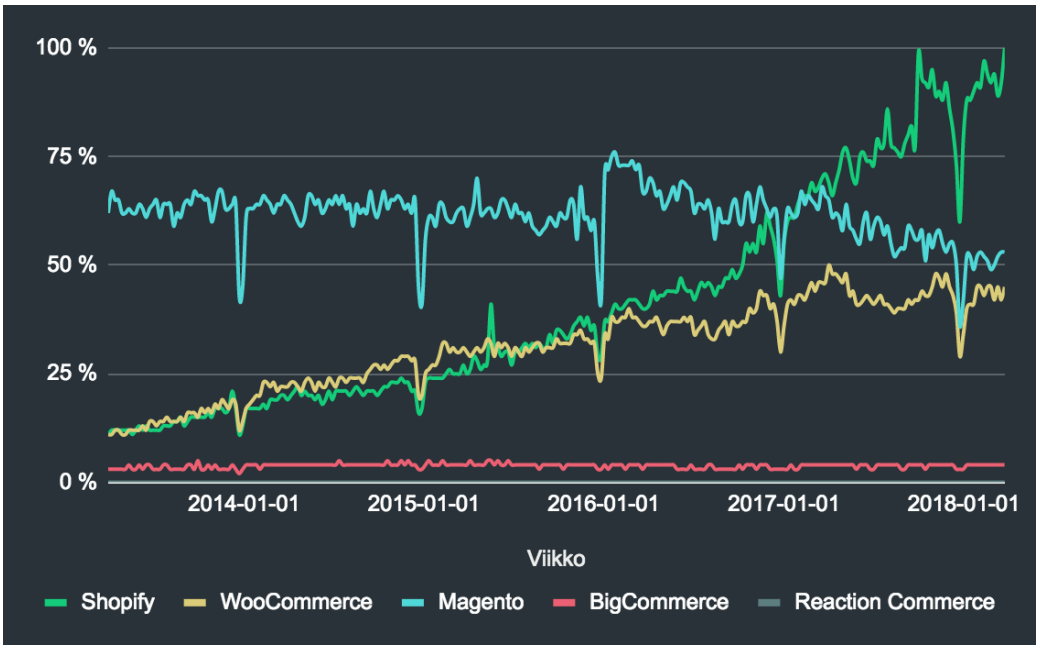
Shopifylla on näiden rajapintojen sekä yli 200:n heidän tuottaman muun avoimen lähdekoodin projektin koodit julkisina repositoreina esillä Shopifyn GitHub-käyttäjän alla. Kehittäjä voi aloittaa näiden rajapintojen ja aloitusprojektien avulla hieman WooCommercea ja Magentoa haastavammin kehittämään palvelun päälle omaa kauppakokemusta. (Shopify repositories n.d.) Shopifyn ollessa pilvipalvelu, kuuluu sen käyttämiseen kuukausittaisia maksuja, eikä sen avulla voi kehittää verkkokauppaa niin joustavasti ja edullisesti kuin kahdella aiemmin mainitulla pystyy. Shopify kuitenkin tarjoaa niin laajasti palveluita ja mahdollisuuksia myydä, että sen käyttö suuntautuu enemmän Shopifyn tarjoamiin valmiisiin pilvipalveluratkaisuihin ja applikaatioiden sisäisiin ostokokemuksiin, ennemmin kuin täysin kustomoituihin kokonaisvaltaisiin verkkokaupparatkaisuihin, joita WooComerce ja etenkin Magento edustaa.

Trendit

Verkkokauppa-alustojen suosio vaihtelee ajan kuluessa, kun uusia ja nuorekkaampia tai tuoreempia ideoita ja ratkaisuja tuodaan markkinoille. Kuten missä tahansa liiketoiminnassa, imagon luonti omaan tuotteeseen on tärkeää ja tuotteen haluttaisiin olevan suosittu sen käyttäjien keskuudessa.

Tähän ajatukseen pohjautuen on seuraavissa kuvioissa visualisoituna Googlen Trends -palvelun eli Googlen -hakukoneella tehtyjen hakujen perusteella laskelmoituja tilastoja eri optioilla. Ensimmäisenä (ks. kuvio 2) on viimeisen viiden vuoden hakujen suosio Shopifyn, WooCommercen, Magenton, BigCommercen sekä Reacition Commercen välillä globaalisti Googlen hakukoneen haulla tehtynä, jossa rajaus on normaaleihin hakutuloksiin kaikissa luokissa. Normaalit hakutulokset eivät siis ota huomioon esimerkiksi kuvahakuja, uutishakuja eikä YouTube -videopalvelun hakuja. Nämä ensimmäisen kuvion verkkokauppa-alustat olivat suosituimpia yksittäisiä alustoja, kun jätettiin pois muut pienemmät alustat yhteensä laskettuna, kuten kappaleessa *Verkkokauppa-alustojen* suosio on kerrottuna. (Google Trends: Shopify, WooCommerce,

Magento, BigCommerce, Reaction Commerce, koko maailma, viimeiset 5 vuotta 2018.)

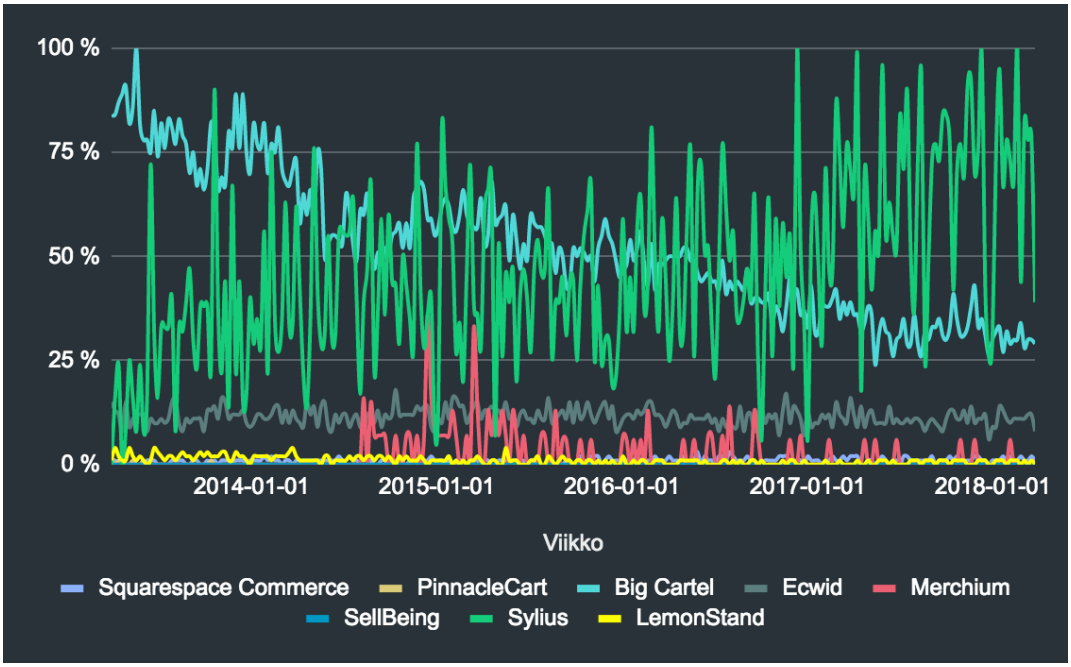


Kuvio 2. Suosituimpien verkkokauppa-alustojen hakujen suosio viimeiseltä 5 vuodelta maailmanlaajuisesti (Google Trends: Shopify, WooCommerce, Magento, BigCommerce, Reaction Commerce, koko maailma, viimeiset 5 vuotta 2018.).

Shopifyn suosio hauissa on noussut merkittävästi viimeisen kahden vuoden aikana, jonka aikana Magenton hakusuosio on laskenut. Sen sijaan WooCommercen suosio on noussut jatkuvasti, kuitenkin hitaasti. BigCommerce on tasaisen vähän haettu termi, ja Reaction Commerce jää hauissa 0–1 % väliin suhteutettuna muihin termeihin ja sen suosio on suhteessa muihin hakutermeihin olematonta. (Mt.)

Kuviossa 3 on esitetty samoin viimeisen viiden vuoden hakujen suosio LemonStandin, Squarespace Commercen, PinnacleCartin, Big Cartelin, Ecwidin, Merchiumin, Sell-Being sekä Syliuse välillä globaalisti Googlen hakukoneen haulla tehtynä, jossa rajaus on normaaleihin hakutuloksiin kaikissa luokissa. Näiden termien data on yhdistetty ja koostettu yhteen kuvioon kahdesta lähteestä, sillä Google Trends ei tue yli 5 termin

hakua kerralla. Tämä kuvio kuvaa edellisen kuvion (ks. kuvio 2) vastakohtana osan kaikista muista verkkokauppa-alustoista, jotka olivat kuvattuna kappaleessa Verkkokauppa-alustojen suosio. (Google Trends: LemonStand, Squarespace Commerce, PinnacleCart, Big Cartel, Ecwid, koko maailma, viimeiset 5 vuotta 2018; Google Trends: Merchium, SellBeing, Sylius, koko maailma, viimeiset 5 vuotta 2018.)



Kuvio 3. Muiden kokonaisvaltaisten verkkokauppa-alustojen hakujen suosio viimeiseltä 5 vuodelta maailmanlaajuisesti (Google Trends: LemonStand, Squarespace Commerce, PinnacleCart, Big Cartel, Ecwid, koko maailma, viimeiset 5 vuotta 2018; Google Trends: Merchium, SellBeing, Sylius, koko maailma, viimeiset 5 vuotta 2018).

Visualisoinnista nähdään, että Syliusin suosio Googlen hauissa on heilahtelevaa, mutta se on kuitenkin suosituin, ja Big Cartellin suosio on laskenut tasaisesti viimeisen 5 vuoden aikana. Ecwidin ja erityisesti Merchiumin suosio on ollut matalaa suhteutettuna Syliusiin ja Big Cartelliin, mutta vahvempia kuin muiden jäljelle jäävien. Squarespace Commercen, PinnacleCartin, SellBeing ja LemonStandin suosio hauissa on hyvin alhaista ja ne jäävät 0–2 % osuuteen. (Mt.)

2.1.3 Kehittämistyön verkkokauppa-alusta

Yleistä

Reaction Commercen ensimmäinen julkaisu oli vuonna 2014, jolloin yritys teki ensimmäisen alpha -versiojulkaisun GitHubiin. Reaction Commerce on sittemmin ollut aktiivisessa kehityksessä, ja ensijulkaisusta nykyhetkeen katsottuna on projektiin tehty yli 8200 kommittia GitHub versionhallintaan. (Reaction GitHub releases 2018).

Reaction Commerce verkkokauppa-alusta on samannimisen Reaction Commerce yrityksen ensimmäinen julkaistu tuote. Yritys on perustettu vuonna 2013 USA:ssa Santa Monicassa, Kalifornian osavaltiossa. (About us n.d.) Työntekijöitä yrityksellä on 20 mukaan lukien toimitusjohtaja Sara Hicks (Meet the team n.d).

Reaction Commere -tuote on jaettu kahteen osaan, joista ensimmäinen on tässä opinnäytetyössä käytettävä ja tutkittava avoimen lähdekoodin verkkokauppa-alusta. Toinen tuote on *Platform as a Service* eli maksullinen palvelukokonaisuus nimeltä Reaction Platform, jota ei tässä opinnäytetyössä tutkittu. Reaction Platform sisältää verkkokauppa-alustan ylläpidolliset asiat, julkaisuun ja infrastruktuurin skaalaukseen kuuluvat ratkaisut yhdessä saman avoimen lähdekoodin verkkokauppa-alustan kanssa integroituna kokonaisuutena (Retail at any scale n.d).

Verkkokauppa-alustan valinnan perusteet

Opinnäytetyön verkkokauppa-alusta valikoitui muista luvussa 2.1.2 esitellyistä verkkokauppa-alustoista ja palveluista siitä syystä, että tuote on nuori, mutta sen suosio on kasvussa, ja opinnäytetyötekijä uskoo sen markkinaosuuden kasvavan lähivuosina nousujohteisesti. Reaction Commercen valinnan muina perusteina työn pääteknologiaksi ja aiheeksi olivat sen reaktiivisuus ja reaaliaikaisuus, avoin lähdekoodi, joustavuus esimerkiksi omien teemojen, maksuväylien tai rajapintojen osalta sekä moderni teknologiapino, jonka myötä se on myös nykyaikainen työympäristö.

Ominaisuudet

Reaction Commercen ominaisuuksissa on paljon vastaavia asioita, kuin mitä merkittävimmät yksittäiset verkkokauppa-alustat WooCommere, Magento ja Shopify tarjoavat. Reaction Commerce on responsiivinen ja täysin mobiililaitteilla toimiva, jonka lisäksi projekti voidaan kääntää natiiviksi iOS tai Android -applikaatioksi tulevaisuudessa. (We’re unlike any other. Reaction Commerce is open source, easy to use, and 100% customizable n.d.)

Reaction Commerce tarjoaa monipuolisesti tuote- ja tuotteiden variaatioiden hallinnoinnin. Tilaustenhallinnassa on rakennettuna jälkitoimitukset ja varaukset sekä tuotesivuilla voi olla useita kuvia ja itse määritettyjä attribuutteja. Joustavan *tagi* -järjestelmän avulla voidaan linkittää dynaamisesti tuotteita ja kategorioita, joiden hyödyntäminen navigaatiossa ja hauissa on tehokasta. (Mt.)

Kaupan hallitsija voi käyttöliittymästä myös hallita asiakasprofiileja, kaupassa on Magenton ja Shopifyn tapaan tuettuna yhden sivun ostokokemus ja sähköpostiviestityksen pohjia voi muokata helposti. Kauppa mahdollistaa ilmoitusjärjestelmän niin asiakkaalle kuin kaupan hallitsijalle ja kauppaan voidaan ottaa käyttöön kolmansien osapuolien käyttäjätilien autentikointi kirjautumiseen. Myös vieraana tehdyt tilaukset ovat tuettuna niin halutessaan. (Mt.)

Ostotapahtumaan voi käyttää valmiita tuettuja kolmansien osapuolien maksuväyliä. Maksujen hoitamiseen ovat tuettuna Stripe, Stripe Connect, PayPal PayFlow, PayPal Express, Braintree sekä Authorize.net -maksupalvelut. Tuotteiden verotus voidaan hoitaa omalla kiinteän hinnan verotuksella, tai kolmannen osapuolen palvelulla, kuten Avalara tai TaxCloud. Myös alennukset ja kupongit ovat tuettuna, ja tilauksen postitukseen liittyvät asiat voidaan hoitaa omien postihinnoitteluiden avulla, tai hyödyntäen kolmansien osapuolien palvelua, kuten Shippo. (Mt.)

Reaction Commercessa on siis mahdollisuus käyttää valmiita maksuväyliä, mutta se tarjoaa myös mahdollisuuden omiin ratkaisuihin. Jos verkkokauppaa ylläpitävällä toimijalla on olemassa oma rajapinta maksutapahtumille, tai jos halutaan käyttää jotain muuta kolmannen osapuolen maksuväylää, kuten suomessa suosittua Paytrailia, on

sen toteuttaminen mahdollista. Kaikkiin muihinkin tapahtumiin verkkokaupassa voidaan lisätä logiikkaa, jos eri tapahtumista halutaan viedä tietoja johonkin rajapintaan. Reaction Commerce on siis joustava, eikä se rajoita viestittelyä muihin järjestelmiin. Näissä toteutuksissa tulee seurata Reaction Commercen dokumentaatiota tietoturvallisuuden ja toisaalta laadukkaan lopputuloksen kannalta. (Creating a Custom Payment Provider 2018.)

Kaupan muokkaaminen on kehittäjän näkökulmasta kohtuullisen helppoa ja se asettuu Magenton ja WooCommercen väliin ennen Shopifyä. Kehittäjä voi tuottaa Reaction Commerceen oman teeman, jolle tarjotaan valmis logiikka, ja verkkokauppa-alusta käyttää automaattisesti kehittäjän muokkaamia teemoituksia, kun ne tehdään projektissa niille varattuun kansioon. Reaction Commercella on aktiivinen *open source community,* jossa verkkokauppa-alustaa kehitetään jatkuvasti, asioista keskustellaan avoimesti ja apua yleisimpiin ongelmiin löytyy, joskin vaikeammin kuin yleisimmillä ja kauemmin markkinoilla olleille verkkokauppa-alustoille. (Mt.)

Reaction Commercessa on paljon samoja ominaisuuksia kuin aiemmin opinnäytetyössä selostetuissa verkkokauppa-alustoissa on. Reaction Commercessa on kuitenkin merkittäviä eroavaisuuksia, jotka ovat sen eduksi, kuten:

- se on reaaliaikainen ja tapahtumapohjainen, tarkoittaen, että esimerkiksi tuotetiedot ja varastotiedot ovat ajan tasalla, sivusto ei tee perinteisiä *refreshejä* ja kaupan hallitsijan kannalta muutoksien vaikutuksen näkee välittömästi
- se on ohjelmoitu JavaScriptillä, joka on usealle webohjelmoijalle tutuin ohjelmointikieli, joka helpottaa kehittäjän työtä
- se on joustava, tarjoten paljon valmista logiikkaa verkkokaupan perustamiseksi nopeasti
- se mahdollistaa täydellisen muokkauksen oman teeman avulla
- se tarjoaa tuen omien maksuväylien tai muiden integraatioiden lisäämiseen, jolloin esimerkiksi tilaukset voidaan viedä johonkin omaan rajapintaan helposti
- se on hyvin dokumentoitu, ja dokumentaatio kattaa Reaction Commercen *core* API:t, tyylikirjat, komponenttikuvaukset, muut Reaction Commercen

avoimen lähdekoodin GitHub *repositoryt* sekä yritys tarjoaa blogin ja YouTube opasvideoita tuotetta varten

- se tarjoaa modernin kehitysympäristön ja *toolingin,* jossa on valmiina tuettuna *hot-reload* eli koodimuutosten automaattinen päivittyminen*,* LESS -tyylilaajennukset, työkalut testidatan tekemiseksi sekä moderni ES6 spesifioitu JavaScript
- kaikki uudet kaupat pitävät sisällään valmiiksi tuen palvelinpuolen *renderöintiin* eli verkkosivun piirtoon, hakukoneoptimoinnin sekä lokalisaation usean kielen tuella
- Reaction Commerce sivuston ylläpidon joustavuus, jossa kehittäjä voi ylläpitää kauppaa missä haluaa, tai yritys voi halutessaan investoida Reaction Platform pilvipalveluun. (Mt.)

Koodikanta & kehittäjän näkökulma

Reaction Commercen pääohjelmointikieli on moderni JavaScript. Reaction Commercesta on kirjoitettu noin 82 % JavaScriptillä, noin 10 % on CSS tyylimäärityksiä ja loput noin 8 % HTML merkkaustekstiä. (Reaction GitHub 2018.) Reaction Commerce on kokonaisvaltainen teknologia, joka ei vaadi muita järjestelmiä, kuten esimerkiksi WooCommercen tapauksessa se vaatii WordPress CMS:än. Se tarjoaa kokonaisen verkkokauppa-alustan, jossa on kaikki kaupan hallinnointiin ja liiketoimintaan tarvittavat ominaisuudet.

Reaction Commerce muodostuu useasta avoimen lähdekoodin ohjelmointikirjastosta, joita se käyttää projektin kehystämiseksi toimivaksi verkkokauppa-alustaksi. Lisäksi sillä on muita riippuvuuksia, joita käytetään kehitysvaiheessa, jotka mahdollistavat projektin kehitystyön paikallisesti omalla tietokoneella. Jotta kehitystyön voi aloittaa, pitää Applen macOS -käyttöjärjestelmän tietokoneella asentaa Node, XCode, HomeBrew, Git sekä Meteor, joiden jälkeen voi asentaa Reaction Commercen oman työkalun, Reaction CLI:n. CLI:llä voi esimerkiksi luoda uuden projektin (ks. komento 1), lisätä siihen komponentteja ja pakata projektin sekä paikallisesti kehitystyöhön että tuotantokäyttöön. (Installation for OSX 2017). Kehitystyön aloitus poikkeaa hie-

man käyttöjärjestelmäkohtaisesti kehitystyön työkalujen asennuksen suhteen ja tarvittavat ohjelmistot on kuvattu Reactionin dokumentaatiossa. Lisäksi Reaction CLI pitää huolen, että verkkokauppa-alustan kaikki muut riippuvuudet asennetaan, ja että Reaction Commerce on ajan tasalla ja uusin versio on asennettuna. Opinnäytetyöhön toteutettiin oma versio Reaction CLI:stä, joka nimettiin projektin mukaisesti *create-reaction-app*. Työn toteutusvaiheessa eli luvussa 3 on käytetty omaa versiota CLI:stä.

```
1 # Uuden Reaction Commerce projetin luonti
2 # käyttäen omaa CLI versiota
3 create-reaction-app init uusi-projekti
```

Komento 1. Uuden Reaction Commerce projektin aloituskomento.

Reactionin riippuvuudet voi tarkistaa NPM:n käyttämästä konfiguraatiotiedostosta nimeltä *package*, joka on JSON-muotoinen tekstitiedosto. NPM mahdollistaa ohjelmistokirjastojen eli toisin sanoen pakettien asentamisen listaamalla halutut paketit konfiguraatiotiedostoon, ja ajamalla komentokehotteelta asennuskomennon. Komento tarkistaa konfiguraatiotiedostoon listatut paketit kahdesta kategoriasta, ns. kehitystyössä käytettävistä ja tuotantokäytössä käytettävistä paketeista.

Paketit asentuvat paikallisesti tietokoneelle projektin juuritason kansioon, josta esimerkiksi Reaction CLI löytää ne ja pystyy käyttämään niitä. Reaction Commercen riippuvuuksina tärkeimpinä ovat React, Meteor, GraphQL ja MongoDB, jotka on kuvattu omissa kappaleissaan opinnäytetyön tietoperustassa.

Kuten aiemmin opinnäytetyössä on mainittuna, kehitystyön aloittaminen on suhteellisen helppoa. Aloittamaan pääsee lataamalla GitHubista Reaction Commercen *repositoryn* ja asentamalla Reaction CLI -työkalun, jolla uuden projektin voi luoda. Tämän

jälkeen kehittäjällä on vapaat kädet muokata verkkokauppa-alustaa esimerkiksi luomalla uuden teeman siihen, ja lisäämällä omaa logiikkaa esimerkiksi maksutapahtumaan tai tilauksienhallintaan, samoin kuin Magentolla tai WooCommercella.

Reaktiivisuus ja reaaliaikaisuus

Reaction Commerce on kehitetty Meteorin arkkitehtuurin ympärille, josta toteutetaan tarkemmin teknologiaa, jota Meteor kutsuu DDP:ksi, eli *Distributed Data Protocol*. DDP eroaa perinteisistä rajapinnoista, joissa on yleisempi kutsu-vastaus malli, joka tarkoittaa sitä, että esimerkiksi asiakkaan siirtyessä sivustolla etusivulta tuotelistaukseen, haetaan data vasta siirroksen alkaessa, jolloin latausta joudutaan odottamaan. DDP:ssä ajatusmalli on erilainen, sillä se käyttää julkaisu-tilaus mallia viestien eli datan lähettämiseen ja vastaanottamiseen. Tällä tapaa data striimataan ja synkronoidaan palvelimen tietokannan ja asiakkaan sivun välillä. (Hicks 2017.) Käytännössä tämä tarkoittaa sitä, että kaikkien asiakkaiden asiakaspään selaimet kuuntelevat palvelimen julkaisemaa virtausta eli *striimiä*, ja selaimet vastaanottavat sen samanlaisena, kuin kaikki muutkin asiakkaat. Hyötynä tästä seuraa se, että jos verkkokaupan hallinnassa tehdään muutos, julkistetaan se reaaliaikaisesti *kaikille* asiakkaille, ilman erillisiä sivulatauksia tai päivityksiä, kun taas perinteisessä kutsu-vastaus mallissa vaadittaisiin asiakkailta erillistä pyyntöä ”saada” uusi muuttunut data, esimerkiksi tuotehinnan muutos tai korjaus tuotetietoon.

Tietoturva

Reaction Commercen kehitystiimi käyttää automatisoituja työkaluja, joilla tarkastellaan ohjelmakoodia ja sen turvallisuutta. Sen tarkoituksena on löytää mahdollisia haavoittavuuksia, jotka ohjelma raportoi automaattisesti. Tämän lisäksi kehitystiimi luottaa yhteisön tukeen, jossa mahdolliset löydetyt haavoittavuudet ilmoitetaan käyttäen Gitin *git-format-patch* komentoa (ks. komento 2), jolla haavoittavuuden korjauksen koodimuutokset eli kommitit saa yhdeksi ”paikaksi”, joka soveltuu myös

sähköpostilähetystä varten. Haavoittavuuden paikan luonti onnistuu komennon kaksi mukaisesti esimerkiksi seuraavalla komennolla. (Vulnerabilities 2018).

```
1 git format-patch master --stdout > new-hotfix.patch
```

Komento 2. Esimerkki Unix -ympäristössä ajettavasta komennosta, jolla voi luoda paikkatiedoston (mt.).

Haavoittavuuksia ilmoitetaan Reaction Commercen sähköpostiryhmään, jonne liitetään paikkatiedosto sekä yksityiskohtia haavoittavuudesta, kuten kuinka sen voi toistaa ja linkki lähdekoodiin, jossa haavoittavuus on. Haavoittavuudet yhdistetään aina jokaisen julkaisun yhteydessä.

Reaction Commercen QA eli *quality assurance* sekä turvallisuusinsinööri Josh Ewing (He 2016) kertoo haastattelussaan, että Reaction Commercen turvallisuus nojautuu siinä käytettyjen muiden teknologioiden eheyteen, joka tuo omat haasteensa. Ewing uskoo, että Meteorin turvallisuus tulee kasvamaan muutaman vuoden aikana. Hän näkee uhkana niin sanotut *man in the middle* hyökkäykset, jossa tietoa koitetaan urkkia astumalla kahden järjestelmän väliin. Myös järjestelmien väärin konfiguroinnit, kolmansien osapuolien NPM moduulit, Docker -konttiympäristö sekä Googlen V8 JavaScript ajoympäristö ovat kaikki osa turvallisuusuhkaa. Hänen mukaansa kaikki edellä mainitut vaativat erillisen turvallisuus analyysin sekä tutkimustyön. (Mt.)

Reaction Commerceen on mahdollista myös pakottaa kaikki yhteydet HTTPS -protokollaan ja kehitystyössä paikallisessa ympäristössä käytetään ei-salattua HTTP -protokollaa sen sijaan. Lisäksi on saatavilla erilaisia sisältöpolitiikkoja, joilla voidaan estää sivuston upotus muuhun verkkosivuun ja asettaa muita asetuksia sisällön sallimisen

ja lataamisen suhteen, esimerkiksi ulkoisista lähteistä. Myös Meteorin sisäiset metodikutsut voidaan validoida ja asettaa sääntöjä tietokannan muokkaamiselle. (Security 2018).

Yhteisö ja yhteisön tuki

Reaction Commercen yhteisö on avoin ja vuorovaikutteinen sen jäsenten sekä Reaction Commercen kehitystiimin välillä. Yhteisölle on avoin foorumi Reaction Commercen ylläpitämänä sekä Gitter -niminen chat-palvelu, johon yhteisön jäsenet voivat avata uusia keskusteluhuoneita, ja kommunikoida, kysyä apua tai esittää kehitysehdotuksia. Reaction Commercen kehitystiimi järjestää joka toinen viikko julkisen YouTube -livelähetyksen, joka on suunnattu kaikille kehittäjille, jossa keskustellaan ajankohtaisia asioita sekä selvitetään vastauksia yhteisön esittämiin kysymyksiin. Lisäksi näiden keskustelujen tallenteen löytyvät erilliseltä YouTube kanavalta. (Community 2018.) Tiimi vetää myös noin kerran kuukaudessa Reaction Action nimisen livetapahtuman heidän päätoimistoltaan, jossa esimerkiksi esitetään ohjeita, kuinka toteuttaa jokin ominaisuus, kerrotaan Reactionin taustalla olevasta teknologiasta tai vastataan Q&A:hin. Reaction Action tallentuu myös myöhemmin katsottavaksi sekä niistä kirjoitetaan blogikirjoituksia Reaction Commercen blogiin. (He 2016.)

Päivitykset

Päivityksistä kehittäjä saa ilmoituksen Reaction CLI:n kautta – aina, kun projektin kehityksen aloittaa, kertoo CLI, jos Reaction Commerceen on saatavilla päivitys. Päivityksen voi asentaa CLI:n avulla (ks. komento 3).

```
reaction update
# tai
reaction up
```

Komento 3. Reaction Commercen ja sen NPM-moduulien päivityskomento.

Reaction Commercen julkaisut tehdään GitHub-versionhallintaan, johon ne listautuvat omalle sivullensa. Huhtikuussa 2018 on julkaistu 87 versiota ja nykyisen uusimman version numero on 1.10.0. Pienempiä versiojulkaisuja voi tulla päivittäinkin ja suurempia päivityksiä julkaistaan esimerkiksi kuukausittain. (Reaction releases 2018.)

2.2 Yleiset teknologiat

2.2.1 JavaScript & ECMAScript

JavaScript on maailman käytetyin ohjelmointikieli GitHub -ohjelmistoversionhallinta verkkosivulla ja 97 verkkosivua 100:sta käyttää JavaScriptiä (Silva, Valente, Bergel, Anquetil & Etien 2017, 3–4). JavaScript julkaistiin vuonna 1995 ja sen käyttötarkoitus tuolloin oli olla yksinkertainen skriptauskieli. Kaksi vuotta myöhemmin ECMA International julkaisi esiversion ECMAScript spesifikaatiosta ja spesifikaation julkaisuiteraatio oli vuosittainen, kun kolmas versio spesifikaatiosta julkaistiin jo 1999. (Kwangwon & Sukyoung 2017, 1–2.)

JavaScriptin julkaisun jälkeen on se ollut verkkosivujen eli HTML- sekä CSS-tiedostojen kanssa käytettävä pääohjelmointikieli. Sitä on käytetty sivuston toiminnallisuuden lisäämiseen, ja ohjelman ajo on tapahtunut aina kohdeselaimessa, toisin sanoen käyttäjän laitteella. Tämä tarkoittaa sitä, että verkkosivun palvelin on vain tarjonnut tiedostot selaimelle, mutta skriptin ajo on tapahtunut käyttäjän tietokoneella tai nykyisin esimerkiksi älypuhelimella. Node.js:n myötä on JavaScript koodin

suorittaminen myös mahdollista palvelimella ja esimerkiksi raskaammat toiminnot saadaan laskettua palvelimella, jolloin päätelaitteelle ei tule niin suurta kuormitusta. Node.js mahdollistaa myös ohjelmien ns. ennenaikaisen piirron, *pre-render*, jolloin sivusto latautuu entistä nopeammin. Lisäksi Node.js:n avulla voidaan ohjelmointi kirjoittaa yhdellä ohjelmointikielellä eli JavaScriptillä sekä palvelimella- että päätelaitteen selaimella skriptiä ajaessa.

JavaScript on prototyyppipohjainen- sekä oliopohjainen tulkattava skriptausohjelmointikieli. Siinä on 7 tyyppiä: *Undefined*, *Null*, *Boolean*, *Number*, *String, Symbol* ja *Object*. Tyypit ovat dynaamisesti määriteltäviä. Objektien ominaisuudet voivat muuttua eikä kaikkia ominaisuuksia ole pakko määritellä objektia luodessa. JavaScript ohjelmat voivat myös luoda dynaamisesti ohjelmakoodia alustajan eli konstruktorin avulla, kuten ajastinfunktio eli *setTimeout* -funktio tekee. Dynaamisen tyypityksen vastakohta on staattisesti tyypitetty ohjelmointikieli, kuten Java, jossa tyypit pitää määritellä tarkasti, eikä oliolle voi lisätä ominaisuuksia ohjelman ajon aikana. Lisäksi JavaScript on funktionaalinen ohjelmointikieli ja se käyttää ns. anonyymejä funktioita laajalti. (Mts. 3–4.)

Nykypäivän JavaScript webteknologioiden ohjelmointikirjastot tukeutuvat ECMA Internationalin asettamiin spesifikaatioihin, tarkemmin spesifikaation ECMA-262 8. julkaisun mukaisesti määrittelemään ECMAScript 2017 spesifikaatioon (ECMA International 2017, 56). Yleisimmin ECMAScriptin versiosta käytetään lyhyempää nimeä version järjestyksen mukaisesti, kuten ES8, edellä mainitulle spesifikaatiolle.

Käytännössä standardoitu moderni ohjeistus kirjoittaa JavaScriptiä mahdollistaa kehittyneet skriptit ohjelmoida JavaScriptiä, jonka lisäksi ECMAScriptiin tehdyt määritelmät saavat tuen myös moderneihin selaimiin, joista ovat vastuussa selaimien kehittäjät, kuten Google tai Mozilla (Valentine & Reid 2013, 1–3). Uuden ECMAScriptin ominaisuuksien tuki vaihtelee selainten välillä, eikä aina kaikkia määritelmiä ole mahdollista käyttää riippuen kohdeselaimen tai ympäristön ominaisuuksista, jolla lopputuotteen toivotaan toimivan.

ECMAScript 2017 adoptio on kuitenkin hidasta ja vain pieni prosentti JavaScript projekteista käyttää edellä mainittua ES8 versiota. Useammat projektit pohjautuvat vielä

huomattavasti vanhempaan ES5 versioon. ES5 spesifikaatio on julkaistu vuonna 2009 ja spesifikaation käyttöaste kehittyy hitaasti (Kwangwon & Sukyoung 2017, 2).

JavaScriptin ominaisuudet muodostavat paljoltikin kritisoidun ohjelmointikielen, jossa tyypit voivat muuttua ohjelman ajon aikana tai objektille voi lisätä tai poistaa ominaisuuksia ohjelman ajon aikana. Kielen luonteen ollessa heikosti rajoitettu aiheuttaa se ohjelmoijalle ns. vaikeasti tunnistettavia ohjelmointivirheitä, kuten ajonaikaiset-tyyppivirheet eli tyyppi-epäkorrelaatioita (esim. merkkijono "1" voi käyttäytyä kuten numero ”1”) tai tapauksia, jossa ohjelmoija muokkaa objektin kopiota alkuperäisen objektin sijaan tai lisää objektiin ominaisuuden muokkauksen sijaan. ECMAScript standardin yksi tehtävä on pyrkiä minimoimaan edellä mainittuja ohjelmointivirheitä määrittelemällä tiukemmat rajat kirjoittaa JavaScriptiä.

Lisäksi on kehitetty erilaisia tyyppijärjestelmiä, eli *type systems*, joiden avulla voidaan vähentää virheitä sekä ohjelman kirjoitus, että tulkkausvaiheessa. Tyyppijärjestelmiä ovat esimerkiksi virtaukseen perustuva, toisin sanoen *flow based* tyyppijärjestelmä nimeltä Flow tai asteittainen tyyppijärjestelmä, eli *gradual typing*, kuten ylijoukko JavaScript toimintoja, nimeltä TypeScript. (Mts. 17–18.) Nämä järjestelmät ovat ohjelmoijan käytettävissä ohjelmointikirjastojen muodossa, jotka voi asentaa ja ottaa JavaScript pohjaiseen projektiin käyttöön esimerkiksi NPM:n avulla. Lisäksi ohjelmointiin käytettävä tekstinkirjoituseditori tarjoaa tuen tyyppijärjestelmille ja editori voi huomauttaa esimerkiksi väärin tyypitetystä muuttujasta jo ennen ohjelman ajoa. JavaScript ohjelmointikieli ei tarjoa itsessään mitään staattisia tyyppitarkistuksia, mutta sen ohjelmointiin on saatu tyyppiturvallisuutta tutkijoiden ehdotusten sekä tyyppijärjestelmien avulla, jotka ovat parantaneet myös JIT-koonnin suorituskykyä. (Mts. 16.)

2.2.2 Node.js

Node.js on vuonna 2009 julkaistu tapahtumapohjainen palvelinpuolen ohjelmointikieli JavaScript ympäristöihin. Se mahdollistaa palvelinohjelmien ohjelmoinnin samaan tapaan kuin perinteiset palvelinohjelmointikielet, kuten PHP. Tarkemmin

Node.js keskittyy verkkoteknologisiin seikkoihin sekä malliin, jossa sillä luodut ohjelmat interaktioivat verkkoyhteyden kanssa. (Ornbo 2012, 6.)

Google avasi V8 -nimisen JavaScript-moottorin lähdekoodin avoimeksi, jota oli aiemmin käytetty Googlen Chrome selaimessa. Node.js suunnitteluvaiheessa otettiin kyseinen moottori käyttöön, sillä se soveltui sen määritelmiin hyvin. V8 etuina olivat sen nopeus, keskittyminen verkkoteknisiin haasteisiin, kuten HTTP, DNS ja TCP -verkkoteknologioihin sekä sen käyttämä webympäristöistä tuttu ohjelmointikieli JavaScript. (Mts. 6–7.)

Node.js soveltuu käytettäväksi laajalti kevyistä ohjelmista aina suuriin webapplikaatioihin. Sen suunnittelun myötä kohdeapplikaatio voi olla esimerkiksi reaaliaikainen järjestelmä, verkkojärjestelmä tai tuhansien samanaikaisten käyttäjien mahdollistava järjestelmä. (Mts. 7.)

Node.js noudattaa myös ECMAScript spesifikaatiota, josta se toteuttaa mm. moduulipohjaisen tavan ohjelmoida. Node.js käyttää moduuleja, joista koostuu sillä tehtyjen ohjelmien koodin rakenne ja näistä uudelleenkäytettävistä lohkoista koostuu paketteja. Paketteja voi lisätä Node.js applikaatioon samaan tapaan, kuin moderneissa ECMAScriptin spesifikaation JavaScript applikaatioissa käyttämällä *import* lauseketta. Node.js:n kehutuin piirre on nimenomaan normi suunnitella pieniä moduuleja tai toisin sanoen paketteja, jotka tekevät yhden asian hyvin. Node.js toteutti tämän suunnitteluvaiheen tavoitteen muun muassa NPM -pakettihallintajärjestelmän avulla, joka on oleellinen osa Node.js teknologiakokonaisuutta. NPM on levinnyt käyttöön myös perinteisiin JavaScript projekteihin, eikä rajoitu pelkästään Node.js applikaatioihin. (Casciaro & Mammino 2016, 6–11.)

2.3 Kehittämistyön käyttämät teknologiat

2.3.1 Kehittämistyön pääteknologiat

React

React on Facebookin vuonna 2013 julkaisema avoimen lähdekoodin JavaScript ohjelmointikirjasto käyttöliittymiä varten. Reactilla kirjoitetaan määritellyllä tavalla komponentteja, jotka esittävät niille annetun datan, eli staten ja propsien mukaisesti esityksen komponentista, keskittyen puhtaasti vain käyttöliittymään. Poiketen joistakin muista JavaScript ohjelmointikirjastoista, React ei toteuta MVC mallia itsessään. Lisäksi React voi hyödyntää JSX syntaksia, jossa React komponentit voidaan kirjoittaa siten, että komponentin kirjoitusasussa on yhdistettynä perinteistä JavaScript koodia sekä HTML-elementtejä (ks. kuvio 4). JSX syntaksi on suosituin tapa kirjoittaa Reactia ja JSX:llä kirjoitetuissa komponenteissa koodin luettavuus paranee. (React n.d.)

```
1  const Button = props => {
2    const { kind, ...other } = props;
3    const className = kind === 'favorite' ? 'btn-fav' : 'btn-default';
4    return <button className={className} {...other} />;
5  };
6
7  const App = () => {
8    return (
9      <div>
10       <Button kind={'favorite'}>
11         Add to favorites
12       </Button>
13     </div>
14   );
15 };
```

Kuvio 4. Reactin JSX syntaksilla kirjoitettu yksinkertainen komponentti.

React tuli suosioon, kun Facebook julkaisi Instagram -nimisen sosiaalisenmedian verkkosivun heidän mobiiliapplikaatioiden rinnalle, joka oli kehitetty Reactilla. Facebookin insinöörit halusivat testata ohjelmistokirjaston joustavuutta oikean maailman kohteeseen ennen sen julkistamista avoimen lähdekoodin kirjastoksi. (Hunt, O'Shannessy, Smith & Coatta 2016, 56–57.)

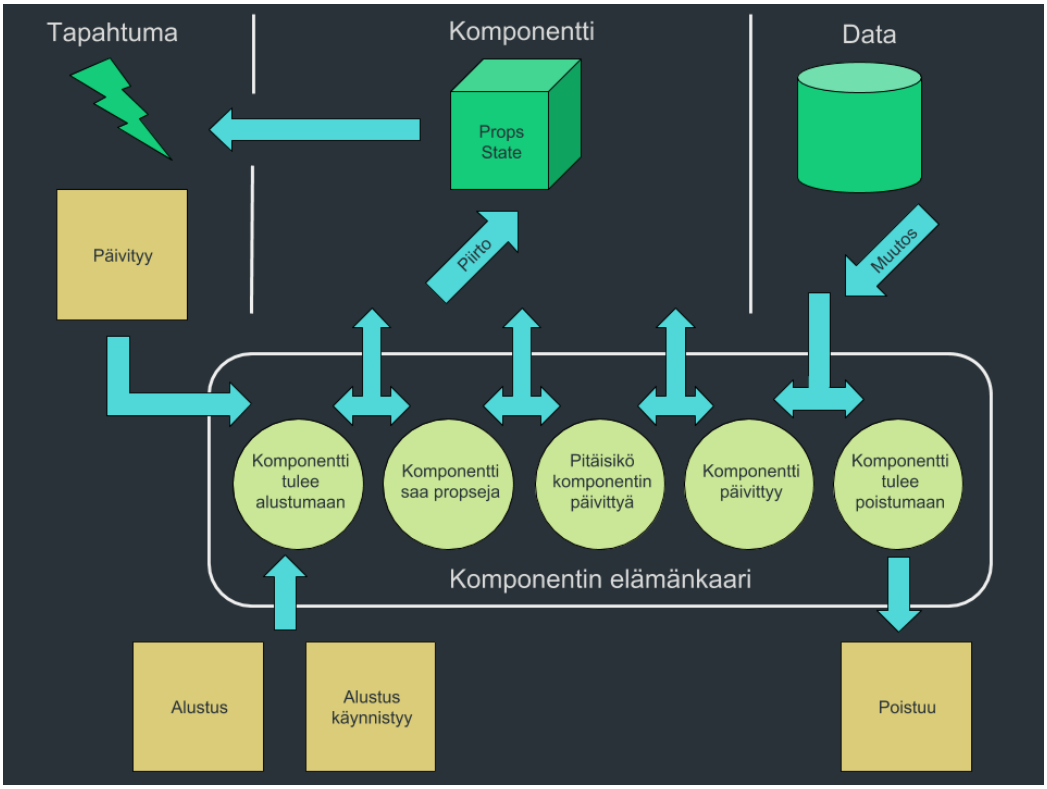
Reactin idea oli alun perin sisäisesti Facebookin sisällä ratkaista ongelma, jossa mainoksien esitykseen käytetty projekti nimeltä *Bolt* pyrittiin korvaamaan vaihtoehtoisella ratkaisulla. Boltin ideologiassa oli paljon samaa kehitettyyn Reactiin verrattuna, sillä molemmat pyrkivät päivittämään käyttöliittymää tapahtumapohjaisesti ja reaaliaikaisesti, jossa datan muutokset heijastuisivat käyttöliittymäkomponentteihin. Haasteena Boltin kanssa oli epävarmuus tilasta, onko komponentti päivittynyt jo vai tuleeko se edes päivittymään, riippuen lähdedatasta sekä se, että Bolt ei voinut päivittää vain osaa sillä kirjoitetusta komponentista. Reactia testattiin sen alkuvaiheilla, jossa sillä kirjoitettiin Facebookin verkkosivuston tykkäys ja kommentointi käyttöliittymä uudelleen, ennen kuin kirjastoa jatkokehitettiin pidemmälle. (Mts. 57.)

Boltin kehityksen ja käytön aikana löydetyistä haasteista React sai alustan kehittyä ja ratkaista kyseiset ongelmat. Reactin esittämä käyttöliittymä nojautuu täysin dataan, ja datan muuttuessa päivitetään vain kyseistä dataa käyttävä komponentti, joka parantaa suorituskykyä sekä takaa reaaliaikaisen esityksen. Korkeammalla tasolla React käsittelee käyttäjän ohjelmoimaa koodia niin sanotussa "mustassa laatikossa" samanaikaisesti, kun se ottaa vastaan mitä vain dataa sille on määritetty otettavaksi vastaan. Tämä malli mahdollistaa millaisen vain komponenttiesityksen toteuttamisen, millaisella vain datalla, jonka lisäksi React ei puutu datamalliin, kuten datan tyyppiin, lähteeseen tai tietueisiin. (Mts. 57–58.)

Edellä kerratun datan muutosten ketjun lopussa React muuntaa komponenttiesityksen ohjelmoinnin valideiksi HTML -spesifikaation elementeiksi, kuten *div* tai *img,* joissa attribuutit ovat määriteltynä datan mukaisesti, esimerkiksi *div* -elementin sisältö tai *img* -elementtiä kuvaava *alt*- ja kuvan lähteen *src*-attribuutit. Myöhemmin datan muuttuessa voi esiintyä vaihe, jolloin React suorittaa vertauksen tulevan datan ja nykyisen datan välillä ja piirtää tältä pohjalta virtuaalisen DOM-puun päivityksen. Jos muutoksia ei ole, päivitystä ei tehdä, mutta jos päivityksiä on datassa, muutetaan

vain ne komponentit, jotka ovat datasta kiinnostuneita. Piirto-metodin palauttama data ei ole merkkijono, eikä DOM-elementti, vaan sen sijaan se palauttaa kevyen kuvauksen, millainen DOM-elementin pitäisi olla (Hunt 2013). Tämän perusteella React tekee komponenttiesityksen ja lopuksi nämä elementit lisätään "oikeaan" selaimen piirtämään DOM-puuhun. Viimeisessä vaiheessa React-komponentin elinkaarta komponentti poistuu virtuaalisesta DOM-puusta, jos esimerkiksi käyttäjä vaihtaa näkymää käyttöliittymässä, joka heijastuu jälleen selaimen oikeaan DOM-puuhun.

Tämä ketju (ks. kuvio 5) toistuu aina datan muuttuessa, siis jopa satoja kertoja sekunnissa joissakin reaaliaikasovelluksissa, kuten pörssiapplikaatioissa. Tämä oli aiheuttanut Reactin kehityksen alkuvaiheilla keskustelua suorituskyvystä. Suorituskykyyn liittyvät huolet kuitenkin osoitettiin vääriksi, sillä Reactia testattiin usean testin läpi verraten muihin JavaScript -ohjelmointikirjastoihin, eikä suorituskyvyssä ollut ongelmia. (Mts. 57.)



Kuvio 5. React-komponentin elämänkaari.

React on kuitenkin jatkuvassa kehityksessä sen menestyksen sekä Facebookin sisäisen ohjelmointityön tarpeiden takia, joka mahdollistaa jatkuvat päivitykset kirjastoon. Facebookilla on yli 50 000 kpl React komponentteja käytössä usean projektin yhteydessä ja yritys on sitoutunut Reactin teknologiaan sekä jatkuvaan kehitystyöhön sen parissa. Reactin kehitystiimi julkaisee tärkeitä muutoksia *asynkroniseen piirtämiseen* liittyen. (Vaughn 2018.)

Ylempänä esitetyn malliin on tulossa React version 16.3 julkaisun myötä muutoksia, jossa osa React-komponentin elinkaaren osista tullaan *deprekoimaan* eli ne merkitään poistuvaksi seuraavassa versiossa eli 16.4 julkaisun myötä. Kyseiset elinkaaren vaiheet ovat:

- *componentWillMount* eli komponentti tulee alustumaan
- *componentWillRecieveProps* eli komponentti saa propseja sekä
- *componentWillUpdate* komponentti päivittyy. (Mt.)

Nämä elinkaaren vaiheet saavat ohjelmoijan kannalta uuden metodinimityksen, jossa metodien nimet aloitetaan *unsafe_* -merkinnällä. Unsafe ei merkitse tietoturvallista vaaraa, vaan se viittaa elinkaaren vaiheen epävarmuuteen komponentin elinkaaressa. Ylemmän listan elinkaaren vaiheet ovat siis epävakaita, ja näitä metodeja React kutsuu sisäisesti useaan kertaan. Tämä tarkoittaa sitä, että esimerkiksi *asynkroniset* HTTP-kutsut API:in voivat tehdä sarjan kutsuja turhaan tai data ei ole palautunut rajapinnasta, ennen kuin React on jo piirtänyt ensimmäisen piirron komponentista. Vastaavia ohjelmoijan kannalta triviaaleja tilanteita vastaan React-kehitystiimi on tuomassa 16.3 version myötä kaksi uutta elinkaaren vaihetta, jotka ovat *getDerivatedStateFromProps* ja *getSnapshotBeforeUpdate.* (Mt.)

getDerivatedStateFromProps korvaa yhdessä jo olemassa olevan *componentDidUpdate* -vaiheen kanssa kaikki tapaukset, joissa tarvitsisi käyttää poistuvaa *componentWillRecievePropsia.* Tämä elinkaari tapahtuu, kun komponentti on alustettu ja, kun se saa uusia propseja vastaan. Se voi palauttaa *objektin,* joka päivittää *staten*, tai *null* joka ilmoittaa, että uudet *propsit* eivät vaadi *staten* päivitystä. (Mt.)

getSnapshotBeforeUpdate korvaa samoin edellisen vaiheen tavoin *componentDidUpdate* -vaiheen kanssa kaikki tapaukset, joissa tarvitsisi käyttää poistuvaa *componentWillUpdatea.* Tätä elinkaarta kutsutaan juuri ennen *mutaatioita* eli muutoksia, kuten DOM-päivitystä. Tämän vaiheen palautettava arvo annetaan kolmanneksi parametriksi metodille *componentDidUpdate.* (Mt.)

Nämä muutokset eivät aiheuta välittömiä muutoksia olemassa oleviin React-projekteihin, mutta ne auttavat uusia tai aktiivisessa kehityksessä olevia projekteja tulemaan stabiilimmiksi ja ohjelmoijan kannalta ennakoitavimmiksi. Osa erilaisten React komponenttien tai pakettien ylläpitäjistä joutuvat kuitenkin lähiaikoina tekemään muutoksia, jotka takaavat niiden toiminnan myös tulevaisuudessa. (Mt.)

Meteor

Meteor on julkaistu virallisesti vuonna 2014 ja sen kehittäjille tarkoitettu esiversio julkaistiin vuonna 2012. Se saavutti heti suuren suosion ja sitä ladattiin useita tuhansia kertoja nopeasti julkaisun ilmestyttyä Hacker News -verkkosivustolla, sillä tarve *full-stack* ohjelmointikirjastolle oli alalla suuri. (Finley 2014.)

Meteor on ns. *full-stack* eli asiakaspään selaimen sekä palvelimen kattava avoimen lähdekoodin ohjelmointikirjasto. Meteorin kehitys tehdään JavaScriptillä, ja se tukee perinteisen internet selaimen lisäksi applikaatiopalveluita ja mobiilikäyttöjärjestelmiä. Meteor tarjoaa useita yleisimpiä avainominaisuuksia, kuten käyttäjätilit sekä työkaluja, kuten applikaation *build tool* ja joukon muita Node.js ja JavaScript paketteja, jotka helpottavat kehitystyötä. (Introduction n.d.)

Meteorin ideologiaan kuuluu periaate, jossa Meteor applikaatio ei jaa palvelimelta HTML -tiedostoja, vaan sen sijaan se tarjoaa datan, jonka päätelaitteen selain piirtää. Lisäksi Meteorin avulla käyttöliittymä saadaan päivittymään reaaliaikaisesti datan kanssa. (Mt.)

Meteor tarjoaa ilmaisen ohjelmistokirjaston lisäksi Reaction Commercen tapaan maksullista PaaS palvelukokonaisuutta. Palvelukokonaisuuden nimi on Galaxy, joka

pohjautuu Dockerin konttiteknologiaan ja Amazonin pilvipalveluinfrastruktuuriin. (Meteor Hosting n.d.)

GraphQL

GraphQL on Facebookin vuonna 2012 aloittama projekti, joka oli aluksi vain heidän sisäinen projekti, joka myöhemmin julkaistiin avoimen lähdekoodin projektiksi vuonna 2015. GraphQL tukee nykyään useaa ohjelmointikieltä kuten Go, Ruby, Scala, Java, .Net, Python sekä JavaScriptiä. (Byron 2015.) GraphQL on hakupohjainen kieli ohjelmointirajapintoja varten sekä palvelinpuolen *runtime,* joka pystyy suorittamaan hakuja datamallien tyyppien määritysten mukaisesti. GraphQL on universaali hakupohjainen kieli, joka toimii useimpien tietokantojen ja ohjelmointikielien kanssa – se ei ole rajoittunut mihinkään yksittäiseen tietokantaan tai pilvipalvelun tietojärjestelmään. Sen toiminta nojautuu ainoastaan olemassa olevaan dataan sekä koodiin. (Intoduction to GraphQL n.d.)

GraphQL palvelu vaatii kaksi määritystä ohjelmoinnissa. Halutuille hauille pitää tehdä malli, jossa on tyypit sekä kentät, jotka kuvaavat datan (ks. kuvio 6). Malleihin kuvataan tallennettavan datan muoto, samaan tapaan kuin esimerkiksi MySQL -tietokannoissa. (Mt.)

```
type Blog {
  id: ID
  title: String
  published: Date
  text: String
  author: Author
}
type Author {
  id: ID
  firstName: String
  lastName: String
  blogs: [Blog]
}
```

Kuvio 6. GraphQL-kielen kaksi mallia, esimerkkinä miten blogi ja sen kirjoittaja voitaisiin esittää.

Tämän lisäksi tulee olla määriteltynä *query* jokaiselle haulle (ks. kuvio 7). Queryyn määritellään ehtoja, miten tietoa haetaan ja mitä tietoa haetaan. Haut voivat olla myös monipuolisempia, jolloin niissä on mahdollista käyttää muuttujia, filttereitä ja ehtoja. (Mt.)

```
type Query {
  blog(id: ID!): Blog
  author(id: ID!): Author
}
```

Kuvio 7. GraphQL-kielen *query,* joka kuvaa blogin ja kirjoittajan haun.

Määritelmien jälkeen voidaan GraphQL-palveluun lähettää pyyntöjä. Palvelu voi olla kuuntelemassa kutsuja perinteisten REST-rajapintojen tapaisesti URL:n osoittamassa palvelimessa tai web-palvelussa. Perinteisesti ohjelmointirajapinnoissa palvelu, eli palvelimella ajettava ohjelma, päättää itse, mitä sen rajapintaan lähetettyyn kutsuun

vastataan eli millaista dataa se palauttaa. GraphQL eroaa tästä täysin, ja lähetettyä pyyntöä vastaa aina saman muotoinen JSON-objekti (ks. kuvio 8). Tämä tarkoittaa sitä, että GraphQL-palveluun lähetetystä kutsusta voidaan aina tietää, millaista dataa palvelu palauttaa, mikä on hakukielen yksi vahvoista ominaisuuksista. Ensimmäinen GraphQL-palveluun tehty haku valitoidaan, jotta voidaan olla varmoja, että se viittaa ainoastaan mallinnetun datan kenttiin ja sen tyyppeihin, jonka jälkeen se suorittaa määritetyt funktiot, jotka tuottavat lähetettävän vastauksen. (Mt.)

```
# haku
{
  blog {
    title
    text
    author {
      id
      firstName
    }
  }
}

# vastaus
{
  "blog": {
    "title": "GraphQL is great",
    "text": "GraphQL is developed by Facebook...",
    "author": {
      "id": "592c4565b4cbac058f1383f8",
      "firstName": "Jussi"
    }
  }
}
```

Kuvio 8. GraphQL-palveluun lähetettävä esimerkki haku, ja sen palauttama vastaus.

GraphQL vahvuuksia on monia. Kuten ylläolevasta kuviosta näkee, haun palauttama data vastaa täysin hakua, joka helpottaa ohjelmoijan työtä ja tosiaalta antaa vapauden muotoilla palautettavaa dataa, sen mukaan mitä halutaan. Blogissa perinteisesti ohjelmoijan tarvitsee tehdä rajapintaan kaikki halutut rajapinnat ja niitä vastaavat tietokantahaut sekä metodit, joita tarvitaan. Tässä esimerkissä ohjelmoija tekisi ainakin rajapinnan, joka listaa blogit vain otsikon ja päivämäärän kanssa, ja

rajapintaa käytetään blogien listaamiseen. Tämän lisäksi luultavasti tehtäisiin haku, joka palauttaa blogin kaikki tiedot, joita käytetään itse blogin esittämiseen. (Byron 2015.)

Jos kuitenkin blogin vaatimukset muuttuvat, kuten blogiin halutaan uusi ominaisuus, vaikkapa tykkäyksien lukumäärä, tarvitsee ohjelmoijan muuttaa kaikkia rajapinnan metodeja, joissa tykkäyksien määrä halutaan käyttöön. Lisäksi pitää muokata tietokantaa. Vastaava tilanne GraphQL-hakukieltä käyttäen onnistuu helposti – ohjelmoija lisää haluttuihin hakuihin lisäksi tykkäyksien määrän kentän, lisää sen blogin malliin, jonka jälkeen hakua voi tehdä vapaasti, eikä muokkausta ole tarvinnut toteuttaa, kuin muutamaan osioon. Samaan tapaan pitää tietokantaan lisätä tuki. (Mt.)

GraphQL hakua vastaavan palautettavan datan lisäksi sen vahvuus on hierarkisuus. GraphQL mahdollistaa *sisäkkäiset* haut, jossa yhteen hakuun voi sisällyttää usean eri tyyppisen datan tietoja. Vastaava toiminto perinteisissä rajapinnoissa voi olla vaikea toteuttaa ja se voi aiheuttaa ongelmia esimerkiksi *asynkronisen* datan kanssa. Lisäksi GraphQL tukee vahvoja tyypityksiä, jossa tyypityksiä pitää aina vastata joukko kenttiä, ja jos haussa on virhe, saa ohjelmoija näistä hyvän virheilmoituksen ennen haun ajoa, kuten SQL-tietokantahauissa on totuttu näkemään. (Mt.)

GraphQL vahvuuksiin luetaan malli, jossa GraphQL toimii protokollana datan ja mahdollisten dataa käyttävien funktioiden ja muun ohjelmalogiikan välillä, eikä se puutu varsinaisesti datan säilytykseen, joka lisää joustavuutta ja toisaalta mahdollisuutta adoptoida GraphQL olemassa oleviin, hyvin laajoihinkin projekteihin tai tuotteisiin. GraphQL on myös itsehallinnoiva, mikä tarkoittaa sitä, että sen palauttaman datan muotoa voidaan tarkistaa tekemällä hieman normaalista hausta poikkeava haku, joka palauttaa varsinaisen datan sijaan dataa kuvaavan mallin. Käytännössä tämä helpottaa ohjelmoijan työtä, jossa jokaista datamallia ei välttämättä ole tarpeellista kuvata dokumentaatiossa tai ohjelmoija voi esimerkiksi web-palvelun kautta hakea ja tarkistaa datan tyyppejä. (Mt.)

2.3.2 Kehittämistyönteknologian avustavat teknologiat

MongoDB

MongoDB Inc. on vuonna 2007 perustettu yritys, joka julkaisi MongoDB tuotteen avoimen lähdekoodin ohjelmistoksi vuonna 2009. Tarve MongoDB:lle tuli silloisen DoubleClick nimisen yrityksen, joka nykyisin tunnetaan nimellä Google, tarpeesta tarjota satoja tuhansia mainoksia per sekunti. Se muodostui teknologiseksi haasteeksi, joka piti ratkaista, ja tätä haastetta varten kehitettiin MongoDB. (MongoDB unleashes the power of software and data for innovators everywhere n.d.)

MongoDB on dokumenttipohjainen tietokanta. Dokumenttipohjainen tietokanta tarkoittaa, että data on tallennettuna JSON-tiedostoa muistuttavassa muodossa (ks. kuvio 9). Dokumenttipohjainen tietokanta mahdollistaa paremman skaalaavuuden siitä syystä, että siinä voidaan tallentaa taulukkoja ja sisäisiä dokumentteja, joka ei ole mahdollista relaatiotietokannan rivi–sarake mallissa. Tämä mahdollistaa monipuolisen hierarkian yhden dokumentin sisällä, joka vastaa modernin oliopohjaisen ohjelmoinnin mallia, kuinka datan ajatellaan muodostuvan. Lisäksi eroavaisuus perinteiseen relaatiotietokantaan on kiinteästi määritettyjen arvojen suuruuksien pois jättäminen. Tämä tarkoittaa sitä, että esimerkiksi tietokannan kenttään, jossa säilytetään tekstiä, ei tarvitse määrittää maksimipituutta tai kokoa, kuten SQL:ssä merkittäisiin esimerkiksi *int* tai *long*. (Chodowor 2013, 1–5.)

```
{
    "_id" : "Q3yYufJsLeypXF5Nv",
    "createdAt" : ISODate("2018-03-17T11:09:09.460Z"),
    "username" : "admin",
    "emails" : [
        {
            "address" : "admin@localhost",
            "verified" : true,
            "provides" : "default"
        }
    ],
    "name" : "Admin",
    "profile" : {
        "invited" : false
    },
    "groups" : [
        "yfvaJAFhoXkDeExtK"
    ],
    "userId" : "Q3yYufJsLeypXF5Nv",
    "updatedAt" : ISODate("2018-04-01T10:59:06.649Z"),
    "shopId" : "J8Bhq3uTtdgwZx3rz",
    "state" : "new",
    "acceptsMarketing" : false
}
```

Kuvio 9. Esimerkkidokumentti MongoDB tietokannasta.

Perinteisissä relaatiotietokannoissa, kuten MySQL, tieto on tallennettuna riveihin ja sarakkeisiin (ks. kuvio 10). Tiedon tallennusmuoto vastaa siis esimerkiksi Microsoftin toimistotyökalun Excelin tapaa esittää tietoa, jossa tieto on rivissä ja sarakkeissa, ja jossa liitoksia tai viitteitä toisiin tauluihin luodaan. Samalla periaatteella esimerkiksi MySQL -tietokannoissa voidaan hakea tietoa monesta taulusta kerralla, sillä taulujen tieto hajautetaan usein useaan pienempään tauluun.

```
# esimerkkirivejä SQL datasta
|  id  |  name  |    message    |
--------------------------------
|   1  |  Jussi | Hei kaikki!   |
|   2  |  Janko | Asia kunnossa |
|   3  |  Simo  | Viittitsä?    |
```

Kuvio 10. Tekstimuotoon kirjattu esimerkki MySQL tietokantakutsun palauttamasta datasta.

MongoDB:n hyviä ominaisuuksia on useita, jotka eroavat perinteisistä relaatiotietokannoista. Sen ollessa niin sanottu *general-purpose database* eli yleiskäyttöön soveltuva tietokanta, se tukee normaalien tietokantatoimintojen lisäksi myös muita. Yleisesti tietokannoista löytyy tuki perustoimintoihin eli tiedon luontiin, lukemiseen, päivittämiseen ja poistamiseen. Näiden lisäksi MongoDB tukee indeksointia myös pitkille teksteille ja ryhmiteltyä koontia useasta pienestä datasta, jonka tietokanta osaa optimoida. MongoDB tukee myös datan elinajan määritteleviä tietoja, jotka ovat hyödyllisiä esimerkiksi lokitiedoissa, jotka osaavat tuhoutua tietyn ajan kuluttua; ja tiedostotallennusta, jossa MongoDB tukee protokollia, tallentaa ja säilyttää suuria tiedostoja sekä metatietoja. (Mts. 2–3.)

MongoDB ei ole kuitenkaan täydellinen tietokanta, ja se ei tue yleisiä relaatiotietokannan ominaisuuksia, kuten yhdistyksiä tai monimutkaisia monirivisiä siirtoja. Näiden poisjättäminen oli arkkitehdillinen päätös MongoDB:n skaalaavuuden parantamiseksi. Muita merkittäviä uhrauksia ovat tietokannan käyttämän tilan kustannuksella ostettu nopeus. Tietokanta ennenaikaisesti *jyvittää* dataa, joka vapauttaa tietokannan palvelimen suorittimen suorituskykyä. Lisäksi se käyttää niin paljon palvelimen välimuistia, *RAM,* kuin mahdollista, jossa se säilyttää ennakoituja indeksejä. Näistä optimoinneista johtuen tietokanta on nopea, mutta silti tietokanta antaa mahdollisuuksien mukaan tehtäviä suoritettavaksi asiakaspään ohjelmointiin, joka vapauttaa suorituskykyä palvelimella. Ne laskennat suoritetaan joko *drivers* eli ajurien tai käyttäjän applikaation koodin avulla. (Mts. 3–5.)

2.4 Tuotteistus

2.4.1 Palvelumuotoilu

Palvelumuotoilun avulla kuvataan asiakaslähtöisesti ongelmanratkaisun keinoin ratkaisu asiakkaan kokemaan ongelmaan, jossa huomioidaan käyttäjien haasteet sekä ympäröivä ympäristö. Onnistunut palvelumuotoilu suunnitellaan moniammatillisella ryhmällä, jonka yhteistyön tuotoksena syntyy toteutuskelpoinen suunnitelma asiakkaan toivomalle palvelulle tai koko palveluliiketoiminnalle. Palvelumuotoilu on innovointia, jossa yhdistetään palvelun kuluttajan eli asiakkaan sekä palveluntuottajan eli liiketoiminnan tarpeet. Palvelumuotoilu antaa siis konkreettisesti työkaluja sekä mallin yrityksen liiketoiminnan uudistamiseksi tai parantamiseksi, niin liiketoiminnallisen kannattavuuden kuin toisaalta asiakaskokemuksen osalta. (Mansén n.d; Miksi hyödyntäisin palvelumuotoilua? n.d.)

Palvelumuotoilu tuo innovatiivisia ideoita palvelumuotoiluprosessin aikana ja ne ovat pääroolissa palvelun muotoilussa. Se vaatii asiakkaan ja asiakkaan kontekstin ymmärtämisen, kaukonäköisyyden tulevaisuuden palvelukokonaisuuksiin ja niiden prototyyppien kokeilua. Edellä kuvattu kokonaisvaltainen lähtökohta voi yhdistää usean alueen ammattiosaamisen, kuten palveluiden markkinoinnin, palvelun operatiiviset asiat sekä interaktiosuunnittelun. Ne kaikki yhdistyvät palvelusuunnitelmallisessa designissa, sen käytänteissä ja tuotetuissa työkaluissa. (Patrício, Gustafsson & Fisk 2018, 1.)

Huomataan, että palvelumuotoilu voi vaikuttaa vaikealta ja monimutkaiselta prosessilta, ja siihen liittyy joitakin haasteita. Palvelumuotoilu ja innovointi ovat vielä huonosti tunnettuja konsepteja sekä ne kumpikin muodostuvat useammasta lähestymistavasta ja metodista. Niillä on erilaiset tietoteorialliset juuret, joita ei ole vielä yhdistetty, mikä vaikeuttaa niiden hyödyntämistä näissä erilaisissa ongelmanratkaisun tilanteissa. Teknologia mahdollistaa ja avaa uusia tilanteita, mutta sen hyödyntämisestä palvelun innovoinnissa palvelumuotoilun kautta tiedetään vain vähän. Myöskään palvelumuotoilun ja innovaation vaikutuksista toisiinsa ei tiedetä tutkimuksien vähäisyyden takia. (Mts. 3–4.)

Palvelumuotoilu kuvataan asiakaslähtöiseksi ongelmanratkaisun keinoksi, mutta se on siirtymässä laajempaan ihmislähtöiseen näkökulmaan. Yhdistämällä useita perspektiivejä ja metodeja voidaan laajentaa palvelumuotoilua ja innovointia. Tämän laatuiset laajemman mallin alueet, jotka vaativat enemmän huomiota ovat:

- palvelumuotoilun ja palveluinnovoinnin yhteyden parantaminen
- osallistumisen syventäminen usean tekijän välillä, kuten yrityksen sisäisten toimihenkilöiden, ammattikuntien, asiakkaan toimihenkilöiden ja tiimien osalta sekä
- teknologian hyödyntäminen kehittääkseen näitä aloja. (Mts. 6–12.)

2.4.2 Käyttöliittymä ja asiakaskokemus

Käyttöliittymällä tarkoitetaan tapaa interaktioida laitteiden, ohjelmistojen tai palveluiden kanssa. Käyttöliittymä on perinteisesti graafinen, kuten tietokonetta tai puhelinta käyttäessä, mutta ei rajoitu siihen. On myös kehitetty äänellä käytettäviä käyttöliittymiä, kuten henkilökohtaiset digitaaliset avustajat; Applen Siri, Amazonin Alexa tai Googlen Assistant. (User Interface (UI) Design n.d.)

Hyvin toteutettu käyttöliittymäsuunnittelu määrittää käyttäjän kokemuksen käyttää laitetta tai ohjelmistoa. Lisäksi sillä voidaan saavuttaa käyttäjän kiitollisuutta, kun käyttöliittymää käytetään suunnittelijan haluamalla ja ajattelemalla tavalla. Ohjelmistojen ollessa abstrakteja on käyttöliittymä ainoa tapa interaktioida sen kanssa. (Mt.)

Käyttöliittymän pääfokus tulisi olla käytettävyydessä ja tehokkuudessa. Tämä tarkoittaa käytännössä sitä, että käyttäjän pitäisi pystyä suorittamaan haluamansa tehtävä helposti ilman, että hänen tarvitsee keskittyä käyttöliittymään suoranaisesti. Käytettävyys ja tehokkuus eivät aina ole ainoat käyttöliittymälle asetetut kriteerit, ja esimerkiksi peleissä voidaan suunnitella leikittelevi tai hauskoja tapoja suorittaa toimintoja. (Mt.)

Käyttöliittymäsuunnittelussa keskitytään designiin käyttäjälähtöisesti. Suunnittelussa halutaan ottaa huomioon sellaisia asioita, jotka mahdollistavat toimia laitteen tai ohjelmiston kanssa sekä varmistetaan selkeys digitalisten tai fyysisten elementtien ymmärrettävyydessä sekä helppokäyttöisyydessä. Käyttöliittymä on interaktion, visualisuuden ja informatiivisen arkkitehtuurin kokonaiskonsepti. Käyttöliittymäelementtejä ohjelmistoissa ovat *esimerkiksi* syöttöelementit: painikkeet, tekstikentät, navigaatioelementit: hakukenttä, ikonit, informatiiviset elementit: notifikaatiot, edistymistä kuvaavat mittarit sekä säiliöelementit: kuten haitarilista. (User Interface Design Basics n.d.)

Designin lisäksi käyttöliittymältä vaaditaan yleisiä periaatteellisia asioita. Seuraava lista on esitettynä aakkosjärjestyksessä, eikä se kuvaa asioiden tärkeysjärjestystä.

- **anteeksiannettavuus**: toleranssi yleisille ja välttämättömille virheille, virheiden välttäminen ennakkoon, kun mahdollista, suojaus katastrofisia virheitä vastaan ja ilmoitus virheen sattuessa
- **arvattavuus**: käyttäjän tulisi pystyä tunnistamaan elementit toisistaan, toimenpiteiden tuloksille tulee tarjota vihjeitä ja ohjeita sekä toimenpiteitä ei saa yhdistää
- **esteettömyys**: järjestelmän käytettävyys ilman muokkausta, niin monelle ihmisellä kuin mahdollista
- **hallittavuus**: käyttäjän on pystyttävä tekemään toimintoja nopeasti, pysäyttämään toimintoja, jatkamaan virheen tapahtuessa ja kontekstin on oltava käyttäjän näkökulmassa sekä tehtävien suorittaminen pitäisi olla joustavaa käyttäjän taitojen, tapojen ja mieltymyksien myötä
- **joustavuus**: järjestelmä on hienovarainen käyttäjien eroavaisuuksien ja taitotasojen suhteen, kuten käyttäjän taidot ja tiedot, kokemus, henkilökohtainen mieltymys, tavat sekä hallitsevat olosuhteet
- **kokonaisvaltaisuus**: järjestelmän pitäisi olla helposti opittavissa ja ymmärrettävissä sekä virtaus toimintojen, vasteiden, visualisten esiintymien ja informaation pitäisi olla järkevässä järjestyksessä, helposti muistettavissa ja kontekstiin liitettävissä

- **kompromissit**: lopullinen design perustuu sarjaan kompromisseja tasapainossa yleisten designin ristiriitojen kanssa sekä ihmisten vaatimukset ovat aina etusijalla teknisten toteutusten rinnalla
- **läpinäkyvyys**: käyttäjälle tarjotaan oikeus keskittyä toimintoon tai tehtävään ilman huolta käyttöliittymän mekaniikasta
- **määriteltävyys**: muokattavuus ja miellyttävyyden parantaminen muokattavuuden tunteen myötä
- **palautettavuus**: järjestelmän pitää sallia toimintojen peruuttaminen tai edelliseen tilanteeseen palaaminen vaikeuksien ilmaantuessa, varmistaa ettei käyttäjä menetä tehtyä työtä virheen tai laite-, ohjelmisto- tai kommunikaatiovirheen myötä
- **responsiivisuus**: järjestelmän pitää vastata käyttäjän pyyntöihin nopeasti sekä tarjota välitön vahvistus käyttäjän suorittamille toimille visualisesti, tekstuaalisesti ja äänellisesti
- **selkeys**: käyttöliittymän visuaalinen, kielellinen sekä elementaarinen selkeys elementtien, toimintojen, vertauskuvien suhteen
- **suoruus:** tehdä toimintoja, jossa saatavilla olevat vaihtoehdot tulisivat olla näkyvillä ja tekojen vaikutus tulisi näkyä
- **tehokkuus**: siirtymien tulisi virrata vapaasti, navigaation polun tulisi olla lyhyt ja käyttäjän seuraavan toiminnon pitäisi olla ennalta aavistettavissa aina, kun mahdollista
- **tunnistettavuus**: käyttäjän kieli tai sanasto, luonnollinen ja käyttäjän käyttäytymiseen perustuva käyttöliittymä ja tosimaailman vertauskuvat
- **uppoutuvuus**: käyttäjällä on mahdollisuus uppoutua kontekstiin ilman merkittäviä häiriötekijöitä
- **visualinen miellyttävyys**: tarkoituksenmukainen elementtien erottelu, ryhmittely, elementtien linjana asettelu, kolmiulotteinen esitys sekä värien ja grafiikan tehokas ja yksinkertainen käyttö
- **yhdenmukaisuus:** ulkoasun, toiminnan, toimintojen osalta kauttansa, kuten komponenttien yhdennäköisyys, toimintojen seuraukset eivät muutu sekä peruselementtien sijainnit eivät muutu
- **yhteensopivuus:** käyttäjän, tehtävän ja tuotteen kanssa, sekä omaksuttavuus käyttäjän perspektiivistä

- **yksinkertaisuus**: kaikkein yksinkertaisin käyttöliittymä kuin mahdollista, jossa tarpeettomat elementit ovat piilossa, kunnes niitä tarvitaan tarjoten tärkeimmät toiminnot ensin, oletusarvojen tarjoaminen, yleisimpien toimien yksinkertaisuuden tarjoaminen vaikeampien toimintojen kustannuksella sekä yhdenmukaisuuden ja yhtenäisyyden tarjoaminen. (Galitz 2007, 41–57.)

Hyvän verkkosivun design ei rajoitu pelkästään luettavuuteen tai väriteemaan. Designin ja käytettävyyden tärkeys on ymmärretty tärkeäksi osaksi kokemusta, jonka asiakas verkkosivulla tai tarkemmin verkkokaupassa saa. Sivustot useimmiten sisältävät valtavasti tietoa ja alasivuja, jolloin asiakas voi usein selata sivua vailla päämäärää. Tämä johtaa menetettyyn kiinnostukseen sivustoa tai tuotteita kohtaan, kun asiakas jättää vierailematta tärkeimmät sivut tai ei löydä oikeaa tuotetta. Sivusto tarvitsee siis hyvin muodostetun navigaation, joka ohjaa käyttäjän oikean tiedon luo oikeaan aikaan, joka kasvattaa kilpailukykyä. (Jenamani ym. 2006, 248–265.)

Tutkijat ovat ehdottaneet monimuotoisia yksilöllisen käyttäjän kokemuksellisia personallisointeja, joilla pyritään kasvattamaan kiinnostusta sivua kohtaan ja toisaalta johdattamaan käyttäjä tärkeimmille sivuille tai tuotteille. Näistä tutkimuksista on johdettu ajatus, jossa jokaiselle käyttäjälle muodostetaan osittain uniikki navigaatio. (Mts. 248–265.)

Srivastavan ja Cooleyn (2000, 17–20) tutkimuksen mukaan on pyritty mittamaan kaupallisen verkkosivun designin vaikutusta yleisellä tasolla US Fortune 500 -yritysten osalta. Tutkimuksessa löydettiin useita haasteita, kun käyttäjän käyttäytymisestä kerättyä tietoa analysoitiin ja suurin haaste oli tietoturva. Tutkimuksen mukaan personalisointi osana verkkosivun käyttäjäkokemusta on monen webapplikaation, kuten verkkokaupan tärkein yksittäinen asia. Dynaamiset suositukset verkkokaupoissa olivat hyvin huomiota kiinnittäviä kohteita.

2.4.3 Design

Design tarkoittaa Oxfordin sanakirjan mukaan englannin kielessä suunnitelmaa tai piirrosta, joka kuvaa objektin toiminnot tai käytön ennen sen tekemistä. Toisin sanoen sillä kuvaillaan myös taiteen tai toiminnan avulla suunnitelman tai piirroksen toimivuutta ennen sen tuottamista. (Definition of design in English n.d.) Design mielletään suomen kielessä suunnitelmana, muotoiluna, grafiikkana, taiteena tai piirroksena, jossa kuvaillaan jonkin esineen tai ohjelmiston ulkoasu.

Designeriltä vaaditaan laajalti taitoja, jossa pelkkä visualinen ilmaisu ei riitä. Nociletti Dziobczenskin ja Personin (2016, 1–14) mukaan designerin pitäisi osata hallinnollisia muotoilun asioita, prosessinhallinnan taitoja sekä ohjelmiston käytön taitoja. Tarkemmin esimerkiksi seuraavia:

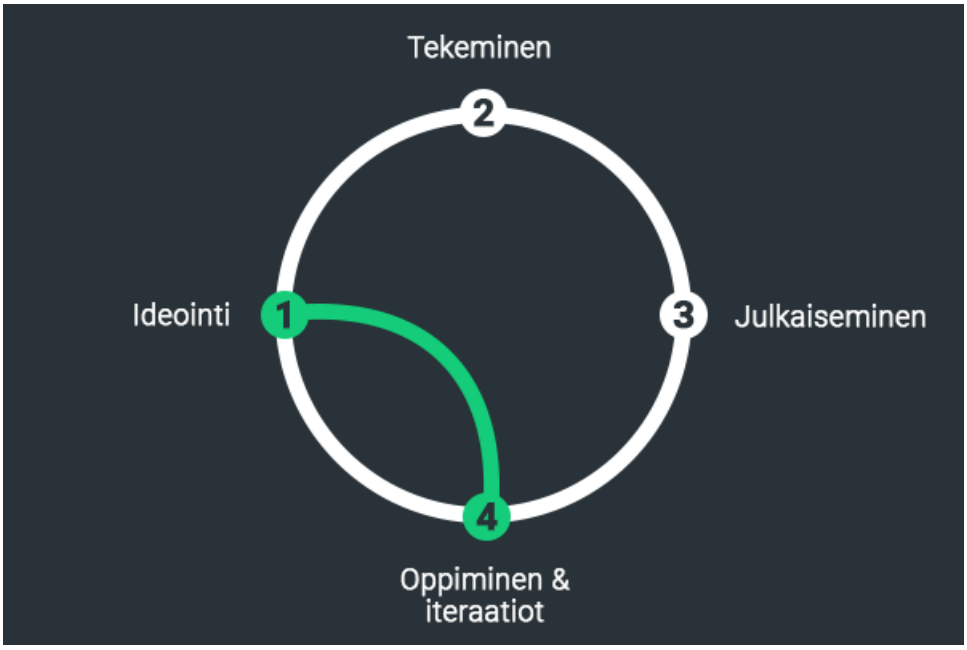
- tiedottamisen taitoja, kuten luovat tapaamiset ja ideoinnit
- luovia ja esteettisiä taitoja, kuten vahva visualinen ymmärrys ja uusien design konseptien, grafiikkojen ja asetelmien luonnin taito
- muotoilun trendien tiedonhaun ja tutkimuksen taito
- tuotantotaitoja, kuten printtituotanto
- idean ja konseptien luonnin taito
- esiintymistaito
- ongelmanratkaisutaito
- käyttäjä ja markkinoiden tiedonhaun taito
- asiakassuhteiden hallinta
- projektin suunnittelun ja johtamisen taito
- tiimityön taito
- tiimin johtamisen taito
- 3D-mallintamisen taito
- 2D, 3D, animaatio, toimisto ja sisällönhallinnan ohjelmistojen käytön osaaminen
- digitalisen kuvan manipulaation hallinta
- kuvittamisen taito

- valokuvauksen taito
- typografian hallinta sekä
- visuaalisen koordinoinnin taito, kuten kyky omaksua taideohjaus ja taito seurata päätettyjä tyylejä ja ohjannuoria. (Mt.)

Design on siis paljon muutakin, kuin pelkkä ulkoasullinen määritelmä. Design on prosessi sekä tuote, ja voidaan ajatella, että muotoilun prosessi on yhtä tärkeää, kuin itse muotoilu eli lopputulos. Muotoilulla luodaan arvoa ja merkitystä ihmisille. Lopullinen tuote on aina yhteydessä sen käyttäjään – ihmiseen. Tuote ei tuo arvoa, jos se ei tarjoa ihmisille mahdollisuutta tehdä tai kokea jotain arvokasta ja merkityksellistä. Toisaalta se on merkityksetöntä, jos se ei mahdollista asioiden tekemistä sillä tavalla, miten nähdään soveltuvaksi ja kontekstiin sopivaksi, kuten miten on totuttu ja miten tuote on myyty. (Sääksjärvi 2016.)

Muotoilulla luodaan uusia tuotteita, jossa ei koskaan ole yhtä oikeata vastausta. Designin avulla ratkaistavaan ongelmaan voi olla monia lopputuloksia eli muotoiluja, jotka kaikki ratkaisevat ongelman. Nähdään, että muotoilijan pitää omata tietoa ja metodeja sekä osata käyttää monia työkaluja tai työmenetelmiä. Design antaa luvan kokeilla uusia ideoita, visualisoida ja tehdä prototyyppejä – tulevaisuuden ongelmiin, joita ei vielä ole olemassa – ja näillä muotoiluilla voidaan tuottaa uutta. (Mt.)

Muotoilun tai tuote- ja liikeidean kehittämiseen on keksitty metodeja, joissa ideaa voidaan kokeilla nopeasti muutamassa päivässä ilman kuukausien työtä. Nopeassa aikavälissä idea validoidaan, eli selvitetään, onko se kannattava, onko arvolupaus määritelty oikein ja todetaanko se kelpoiseksi jatkokehittää valmiiksi tuotteeksi tai designiksi asti. Tällaista ketterää metodia käytetään työkaluna, jossa ohitetaan perinteisen kehityksen varsinainen rakentaminen sekä julkaisu, ja keskitytään ainoastaan ideointiin, oppimiseen sekä iterointiin (ks. kuvio 11). (Design Sprint – Testattu prototyyppi alle viikossa n.d.)



Kuvio 11. Design sprinttiä kuvaava malli (Design Sprint – Testattu prototyyppi alle viikossa n.d.).

Muotoilun, tuotteiden ja palveluiden metodia kutsutaan *Design Sprintiksi*, jota käyttää useat kansainväliset huippuyritykset, kuten Google, Slack ja Ebay. Design sprintistä on hieman erilaisia versioita, joissa kuitenkin ideologia on samanlainen, eli nopeaa ideointia ketterästi. Alun perin design sprintin on kehittänyt Yhdysvaltalainen Jake Knapp. Seuraavaksi on kuvattu yksi malli design sprintistä. Esimerkki sprintti koostuu neljästä vaiheesta:

- haasteiden määrittely ja sprintin tavoitteet ja laajuus
- päätös testattavasta ideasta
- realistisesti mitoitettu prototyyppi ja sen tekeminen sekä
- Käyttäjätestaus. (Mt.)

Vaiheet jakautuvat päiviin, eli sprintti kestää neljä päivää, ja jokaiselle päivälle on selvä päämäärä, josta ei tule poiketa. Sprintin ensimmäisessä vaiheessa tuotetaan paljon ideoita ja ratkaisuja, jotka muodostuvat määritellyn haasteen mukaisesti. Toisessa vaiheessa suodatetaan ideat, ja valitaan niistä paras. Sen jälkeen ideasta määri-

tellään prototyyppi. Prototyypin määrittelyyn käytetään *storyboardia*, eli kuvakäsikirjoitusta, jossa vaiheistetaan muotoilun, tuotteen tai palvelun käyttötapaus. Storyboardiin kuvitetaan siis kuvitellun käyttäjän ”kulku” eri vaiheiden läpi. Esimerkiksi, jos haasteena olisi se, että verkkokaupan tuotteita katsotaan paljon, mutta vain harva siirtyy tuotesivulta ostoskoriin, voitaisiin vaiheisiin kuvata asiakkaan kulku. Asiakkaan kulku voisi olla esimerkiksi etusivulta tuotteisiin ja tuotelistauksesta yksittäiseen tuotteeseen. Tuotesivulle kuvattaisiin itse prototyyppi, eli ratkaisu, kuinka asiakas parhaiten ja todennäköisemmin siirtyisi ostoskoriin asti. Kolmannessa vaiheessa tehdään realistinen prototyyppi, esimerkiksi samaan esimerkkitapaukseen liittyen voitaisiin tuottaa uudet käyttöliittymäkuvat, joissa on esitettynä ratkaisu ongelmaan, kuten selkeä ja helposti huomattava ”lisää ostoskoriin” painike sekä esimerkiksi vuokaavio, jossa näkyy, että asiakas siirretään ostoskoriin automaattisesti napin painamisen jälkeen. Lisäksi kolmannessa vaiheessa hankitaan prototyypille testaajat eli testikäyttäjät, jotka eivät kuulu design sprintin työryhmään. Viimeisessä eli neljännessä vaiheessa prototyyppi testataan oikeilla testikäyttäjillä ja kerätään palaute. Palautteen pohjalta muodostetaan selkeät vaiheet, kuinka prototyyppiä tulee hioa, ja päätetään, onko se toteuttamisen arvoinen idea. (Mt.)

Design sprinttiin osallistuu noin 5–8 henkilöä, jotka ovat mieluiten eri ammattialojen edustajia tai heillä on eri työtehtävät yrityksessä tai yhteisössä. Rooleina ryhmästä olisi hyvä löytyä päätöksentekijä, joka loppukädessä tekee lopullisen päätöksen prototyypistä ja sen jatkokehitysmahdollisuudesta sekä suunnittelija, joka visualisoi idean prototyypin tai prosessin. Muita rooleja ovat markkinoinnin vastaava henkilö, joka tuottaa prototyypille ”sanoman” tai ”viestin” sekä asiakkaat tunteva henkilö, kuten *customer success manager*, joka osaa nähdä asiakkaan markkinoinnin ja myynnin läpi. (Mt.)

Design sprintin konkreettinen tulos on testattu ja validoitu prototyyppi, kuten esitys, prosessi, mobiilisovellus tai imago. Lisäksi tuloksena on kooste sprintistä ja erityisesti käyttäjäpalautteista. On mahdollista myös priorisoida käyttäjätarinat, joiden yhteyteen määritellään kustannus- ja työaika-arviot, sekä viimeisenä voidaan tehdä sijoittajille suunnattu esite. (Mt.)

3 Reaktiivisen verkkokaupan toteutus

3.1 Määrittely

Opinnäytetyössä tehtävä verkkokauppa oli niin sanottu *proof of concept* eli malli verkkokaupasta, joka esittää sen toiminnan ja perimmäisen ajatuksen verkkokauppa-alustan taustalla, ilman että siitä tehtäisiin liian laajaa. Englannin kielessä käytetään usein termiä *MVP* eli *mininum viable product.* Tämä tarkoittaa siis minimivaatimuksilla toimivaa, mutta määritelmän mukaista tuotetta.

Ohjelmistojen määrittelyyn käytetään erilaisia metodeja ja työkaluja, kuten vaatimusmäärittelyjä tai *user storyja* eli käyttäjätarinoita. Perinteisissä tavoissa tuotteen määrittely on tehty ensin dokumentoimalla, jota on seurannut tuotteen tekeminen. Nykyisin nopeasti iteroivassa ohjelmistokehityksessä on siirrytty ketterämpiin kehityskeinoihin, jossa mallina on ohjelmiston tuottaminen ennen dokumentointia. Tutkimuksissa on havaittu, että tiukkojen määritelmien kuten yhdenmukaisuuden, vakauden, muokattavuuden ja kokonaisuuden määrittäminen on vaikeata. Vaikeuden lisäksi niistä on vaikea ymmärtää, millaisia vaatimuksia oikeat käyttäjät ja esimerkiksi osakasomistajat toivovat. On havaittu, että käyttäjälähtöinen käyttäjätarinoiden avulla kehitetty ohjelmistokehitys on hyväksi. Ihmisiä kiinnostaa luonnostaan tarinat ja tarinoita on helpompi seurata ja ymmärtää. Lisäksi puhemuodon eli sanallisesti kuvatun määritelmän, kuten ”*asiakas Anssi haluaa tehdä ostoksen yhden napin avulla, jos hän on jo aiemmin ostanut samasta verkkokaupasta, ja hän käyttää samaa tietokonetta”* avulla muodostetut metodit integroituvat hyvin osaksi ketterää kehitystä. Toimintojen julkaiseminen asiakkaille on nopeampaa ja samalla se toimii riskien hallinnan keinona. Kun ohjelmistoon on julkaistu kriittiset ominaisuudet, tulee harkita kompromisseja nopean julkaisun ja ei-toiminnallisten määritelmien, kuten vakauden tai skaalautuvuuden välillä. Kompromissien tuloksista tehdään käyttäjätarinoita myöhemmille ketterän kehityksen jaksoille eli *sprinteille*. (Niu, Brinkkemper, Franch, Partanen & Savolainen 2018, 86–90.)

Työn määrittelyyn käytettiin kuvitteellista projektia, joka kuvattiin käyttäjätarinoilla ja *casella* eli tapauksella. Käyttäjätarinoihin kuvattiin toiminnot, joita toisaalta tuoteomistaja eli *product owner*, pääkäyttäjä eli *shop owner*, hallinnoija eli *shop manager,* kaupan asiakas eli *user* sekä kaupan kehittäjä eli *developer* vaativat käytännön näkökulmasta. Lisäksi siinä määriteltiin, miten lopullisen verkkokaupan tulisi toimia heidän näkökulmastaan. Tapauksessa kuvattiin projekti yleisemmällä tasolla, kuten verkkokaupan hankkija, liiketoiminta, suurpiirteinen organisaatiokuvaus ja kehitystyön tarpeellisuus ja sen takana oleva motiivi ja tavoitteet.

Käyttäjätarinat

Käyttäjäryhmien kuvitteellisissa käyttäjätarinoissa kuvattiin haluttuja toimintoja. Toiminnot ovat määriteltynä kahteen luokkaan: *toiminnallisiin* ja *ei-toiminnallisiin* vaatimuksiin. Toiminnalliset vaatimukset kuvaavat käytännön toimintoja ja millaisia vaatimuksia niillä on. Ei-toiminnalliset vaatimukset kuvaavat sen sijaan esimerkiksi suorituskyvyn, skaalaavuuden ja vasteajan vaatimuksia.

Käyttäjäryhmiä oli viisi, jotka kuvaavat verkkokauppaa käyttävien henkilöiden tarpeet eri näkökulmista. Perinteisellä asiakkaalla on eri intressit kuin esimerkiksi kaupan haltialla. Käyttäjäryhmät olivat:

- tuoteomistaja eli *product owner,* joka sisältää verkkokaupan ja sen taustalla olevan yrityksen päävastuussa olevan henkilön, joka tekee viimekädessä päätöksiä ja vastaa esimerkiksi ominaisuuksista, ulkoasullisista seikoista tai kehityskäytänteistä, kuten ohjelmistotuotannon ketterän kehityksen mallista
- pääkäyttäjä eli *shop owner,* joka sisältää niin sanotut kaupan pääkäyttäjät, joilla on oikeus esimerkiksi lisätä kauppiaita, maksuvaihtoehtoja tai muuttaa yrityksen laillisia tietoja kuten osoitetietoja
- hallinnoija eli *shop manager,* joka sisältää kaupan tuotteiden, tilausten, asiakaspalvelun, blogin ja muun sisällön tuoton, eheyden varmistuksen ja muun yleisen hallinnoinnin
- asiakas eli *user*, joka käsittää asiakkuudet, niin kirjautuneille kuin vieraileville käyttäjille sekä

- kehittäjä eli *developer*, joka sisältää verkkokaupan kehityksestä vastaavat henkilöt kuten ohjelmoijat, ylläpitäjät, laadunvarmistajat, ohjelmiston dokumentoijat sekä tuotantotoiminnan vastaavat, kuten palvelimen ylläpitäjät.

Tuoteomistaja -ryhmän tarinat

Tarina 1: Haluan, että kun verkkokauppasivun avaa, se edustaa yrityksemme imagoa selkeydellä ja helppokäyttöisyydellä. Sivuston värien tulisi tulla olemassa olevasta brändäyksestämme, koska sivun pitää olla yhdistettävissä meihin helposti.

Tarina 2: Haluan, että verkkokaupan toiminta on minun suuntaani läpinäkyvää, tarkoittaen, että haluan tietää ketkä käyttäjät voivat tehdä mitäkin muutoksia. Lisäksi haluan saada tietää, jos ohjelmistoon tehdään muutoksia, koska olen päävastuussa koko kaupan toiminnasta ja olen linkki yrityksen henkilöiden välillä.

Pääkäyttäjä -ryhmän tarinat

Tarina 1: Haluan, että voin helposti lisätä uuden maksutavan verkkokauppaan, sekä muokata verotus- ja postitusasetuksia, koska käyttämämme postin yritykset muuttavat hintojaan usein, johon haluan reagoida.

Tarina 2: Haluan, että voin lomieni lähestyessä siirtää tai antaa uudet pääkäyttäjän tai hallinnoijan oikeudet minua tuuraavalle työntekijälle helposti, koska en halua lomallani hoitaa työasioita.

Hallinnoija -ryhmän tarinat

Tarina 1: Haluan, että pystyn tarkistamaan tehokkaasti olemassa olevien tilausten tilanteen ja, että voin löytää tilauksen helposti, sillä tilaukset suoraan asiakkaalle ovat uusi asia meille. Uskon, että minulle tulee tarve tarkistaa tilauksia, jos asiakkaalla on kysyttävää esimerkiksi väärän tuotteen saapuessa tilauksessa.

Tarina 2: Haluan, että voin nähdä tuotteidemme varastosaldot helposti. Nykyisin joudun soittamaan useaan eri kauppaamme, kun asiakas tiedustelee tuotteiden saatavuutta, joka vie työaikaani.

Tarina 3: Haluan, että tuotteiden hallinnointi on helppoa, mutta ennen kaikkea tuotteiden tietojen on oltava ajan tasalla asiakkaille, koska nykyisin tuotteiden hintojen muuttaminen on hidasta liikkeissämme. Joskus voi viedä viikkoja, ennen kuin tuote on samanhintainen niissä kaupoissa, joissa sen haluaisimme olevan.

Tarina 4: Haluan, että kumppaniyrityksemme voivat itse hallinnoida heidän tuotteitaan ja maksutietojaan, sillä teemme jo nykyisin sellaisia sopimuksia, joissa osa myymistämme tuotteista maksetaan kolmannelle osapuolelle. Emme halua olla vastuussa tuotetietojen tai hintojen oikeellisuudesta.

Asiakas -ryhmän tarinat

Tarina 1: Haluan, että verkkokauppa on nopea ja selkeä käyttää, ja että löydän tuotteen helposti. Tiedän usein, millaista tuotetta etsin jo ennen kuin näen valikoiman.

Tarina 2: Haluan, että ostoksen teko on helppoa ja, että hyödyn ostoksistani esimerkiksi tulevaisuudessa alennuskupongeilla. En mielelläni käytä aikaani tietojen täyttämiseen ja haluan, että hyödyn juuri kyseisessä verkkokaupassa asioinnista.

Tarina 3: Haluan, että verkkokauppa toimii ja on saatavilla 24/7, ja että tilaukseni onnistuu ilman virheitä. En aio yrittää tilausta myöhemmin, jos sivusto on kaatunut, enkä halua joutua ottamaan yhteyttä kauppaan tilauksen tietojen muuttuessa virheen takia.

Kehittäjä -ryhmän tarinat

Tarina 1: Haluan, että verkkokaupan ohjelmointi onnistuu nykyaikaisilla työkaluilla, jotka jo tunnen entuudestaan, sillä työskentelen tehokkaammin tutuilla teknologioilla.

Tarina 2: Haluan, että verkkokaupan julkaisu onnistuu tehokkaasti ilman turhaa manuaalista työtä. Käyttämiemme ketterien kehitysmenetelmiemme takia julkaisemme usein, mutta pieniä ominaisuuksia kerralla, enkä halua, että prosessi on vaikea tai aikaa vievä.

Tarina 3: Haluan, että ohjelma on vakaa sen testattuani, ja että palvelu on tarjolla jatkuvasti ilman ongelmia. Testattuani työni, haluan luottaa verkkokauppa-alustaan, että se toimii varmasti.

Tapaus

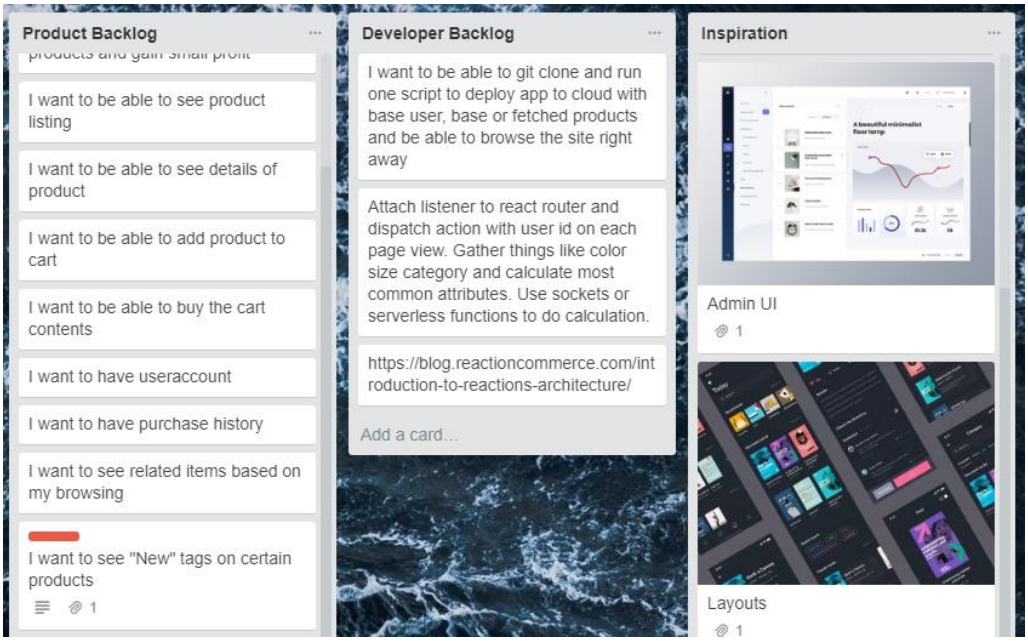
Tapaukseen esiteltiin kuvitteellinen yritys nimeltä *Vasios* ja sen kehitystoiminnan projekti, jonka tavoitteena oli tuottaa uusi verkkokaupparatkaisu liiketoiminnan vahvistamiseksi sekä uusien myynninkanavien saavuttamiseksi. Yritys on globaalilla markkinalla toimiva muoti- ja vaatetusalan kauppaketju, joka omistaa hieman yli 400 liikettä pääosin Keski- ja Pohjois-Euroopan, Pohjois- ja Itä-Aasian sekä Pohjois- ja Etelä-Amerikan alueilla. Liikkeistä noin 12 % toimii isossa kauppakeskuksessa, noin 5 % myy tuotteita esimerkiksi ulkoilmatoreille ja festivaalialueille sekä muihin tapahtumiin ja loput noin 83 % ovat perinteisiä liikkeitä. Yritys tuottaa omia tuotteitaan sekä heidän erityispiirteensä on asukokonaisuuksien tai tarvikekokonaisuuksien tarjonta. Lisäksi he myyvät paljon muiden valmistajien tuotteita, ja osa valmistajista haluaa itse hallinnoida täysin tuotteen hinnoittelua. Myös osan valmistajien kanssa on poikkeava sopimus, jossa tuotteiden myynti menee suoraan tuotteen omistamalle yritykselle tai heidän edustamalle organisaatiolle, kuten hyväntekeväisyysjärjestöille. Pientä osaa tuotteista tarjotaan kuukausihinnoittelulla, jossa tuotesarjan tuote toimitetaan asiakkaalle esimerkiksi kuukausittain.

Yrityksen organisaatio yleisellä tasolla muodostuu 7 pääosakkaasta, yli 20 itsenäisestä- ja yrityssijoittajasta, yli 3200 työntekijästä liikkeissä ja muissa toimitiloissa sekä noin 20 muotisuunnittelijasta ja 2 sosiaalista mediaa hoitavasta henkilöstä. Lisäksi on joitakin kymmeniä myynnin henkilöitä, jotka hoitavat eri tuotteiden ostoja tai sopimusasioita. Yrityksen pääosakkaat yhdessä kolmannen osapuolen asiantuntijoiden

kanssa muodostivat kehitystiimin, ja he näkivät tarpeen laajentaa jo tunnetuksi tullutta kauppaketjua verkkokauppa myyntiin, joka avaisi uusia markkinoita, kasvattaisi myyntiä ja parantaisi yrityksen imagoa ja tunnettavuutta. Kehitystyön tavoitteena oli parantaa yrityksen tuottavuutta, löydettävyyttä, parantaa myynnin joustavuutta uusien asiakkuuksien kanssa ja avata uusi asiakaskanava tulevan verkkokaupan yhteyteen toivotun blogin avulla.

3.2 Suunnittelu

Opinnäytetyön verkkokaupan suunnittelu muodostui määritelmien pohjalta. Suunnittelun alkuvaiheessa oli kehitysympäristön ja yleisesti projektinhallinnan asioiden suunnittelu. Kehitysympäristö suunniteltiin Reaction Commercen dokumentaation pohjalta, joka määritteli sen vaatimat teknologiat ja ympäristön. Kehitysympäristö suunniteltiin siis vastaamaan tarvittavaa tasoa, jona toimi paikallinen oman tietokoneen käyttöjärjestelmä ja siihen asennetut työkalut, kuten ohjelmoinnin IDE (ks. sanasto) ja NPM-paketit. Projektinhallintaan opinnäytetyössä käytettiin *boardia* eli tehtävätaulua Trello -nimisen ilmaisen palvelun kautta (ks. kuvio 12). Tauluja luotiin kaksi – toinen opinnäytetyön kirjallista- ja toinen ohjelmointityön tekemistä varten. Tauluja ylläpidettiin työn edetessä. Verkkokauppaan suunniteltiin myös tarvittavat automatisoinnin skriptit, jotka olivat yksi määritys verkkokaupalle.



Kuvio 12. Trello-palvelun ohjelmoinnin tehtävätaulu.

Verkkokaupan hallinnoinnin suunnitteluun käytettiin Reaction Commercen dokumentaatiota, josta selvisi, että kaupan hallinnointi onnistuu alustan hallintapaneelin kautta, eikä työssä tarvitse tuottaa omaa hallinnointia. Verkkokaupan ylläpidon suunnittelu muodostui julkaisussa käytetyn kolmannen osapuolen pilvipalvelun kautta hankitun alustan avulla. Julkaisuympäristö suunniteltiin myös Reaction Commercen dokumentaation pohjalta, jossa julkaisualustasuositus oli Docker-konttialusta, mutta julkaisualustaan jouduttiin tekemään myös omia määrityksiä luvun 3.6 mukaisesti ongelmien myötä. Konttialusta eli *container* mahdollistaa tuotetun verkkokaupan julkaisun kokonaisena ympäristönä. Reaction CLI -komentokehoteohjelmassa on komento, jolla projektin voi muuntaa Docker -yhteensopivaksi. (Deploying Reaction using Docker 2018.).

3.3 Toteutus

Opinnäytetyön toteutus aloitettiin helmikuun 2018 alussa opinnäytetyön aloitustapaamisen jälkeen. Opinnäytetyön tekemiseen jäi noin kolme ja puoli kuukautta ennen palautuspäivää, toukokuun 2018 puolessa välissä. Opinnäytetyön toteuttamiseen on suunniteltu noin 405 tuntia eli 15 opintopistettä.

Kehitysympäristö

Toteutusvaiheen alussa pystytettiin paikallinen kehitysympäristö, jossa voitiin ohjelmoida Reaction Commercen applikaatiota sekä testata ja kehittää automatisoinnin skriptejä. Kehitysympäristöjä oli kaksi opinnäytetyön toteutuksessa, joista toinen oli macOS käyttöjärjestelmällä ja toinen Windows 10 Pro käyttöjärjestelmällä.

Tietokantaympäristö

Paikallisen kehitysympäristön toteutuksen jälkeen perustettiin ympäristö, joka tarjoaisi tietokannan julkaistulle verkkokaupalle sekä paikalliseen kehitykseen käytettäväksi. Reaction Commerce käyttää MongoDB-tietokantaa, joka määriteltiin DigitalOceanin tarjoamalle Linux Ubuntu -käyttöjärjestelmän pilvipalvelupalvelimelle. Palvelin on erillinen luvun 3.6 esitetystä julkaisupalvelimesta, eikä niitä pidä pitää samoina. Palvelimelle määriteltiin paikallisen kehitystietokoneen julkinen SSH-avain, jota vasten kirjautuminen todennettiin palvelimeen. SSH-avain lisättiin Ubuntun käyttämään *authorized_keys* -nimiseen tiedostoon, josta järjestelmä tarkistaa ja varmentaa etäkirjautumisen. Palvelimelle määriteltiin myös palomuuri, joka sallii ainoastaan http ja HTTPS protokollan verkkokutsut sekä reitti, jossa sallittiin kehitysympäristön verkon julkinen IP-osoite, jossa sallittiin yhteys MongoDB:n käyttämään osoitteeseen sekä porttiin. Tämä mahdollisti etäyhteyden tietokantaan, jonka lisäksi yhteys määriteltiin MongoDB:n konfiguraatiotiedostoon nimeltä *mongd.conf*. Näillä toimilla voitiin yhteys tietokantaan varmentaa turvallisesti, jolloin Reaction Commerce voisi

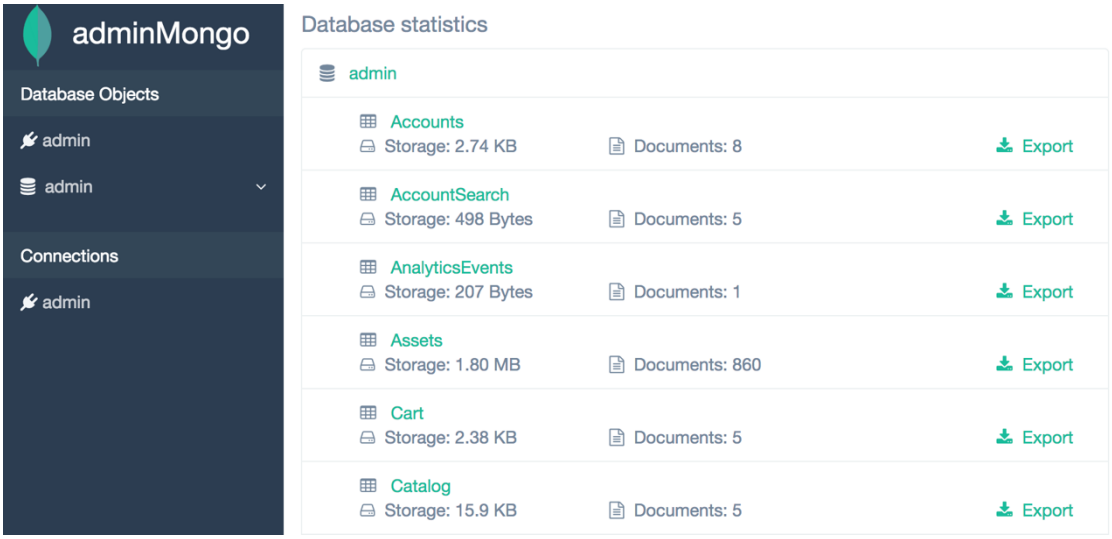
käyttää erillisellä palvelimella ylläpidettyä tietokantaa. Viimeinen vaihe Reaction Commercea varten oli lisätä tietokannan osoite paikallisen kehitysympäristön ympäristömuuttujiin. Reaction Commerce on varannut muuttujan nimeltä *MONGO_URL*, johon määritellään määrämuotoinen osoite tietokantaan (ks. komento 4).

```
# Ympäristömuuttujan asetus Unix ympäristössä ja
MONGO_URL=mongodb://käyttäjä:salsana@domain:portti/tietokanta
# Windows ympäristössä
setx MONGO_URL "mongodb://käyttäjä:salsana@domain:portti/tietokanta"
```

Komento 4. Ympäristömuuttujien asettamisen esimerkki Unix ja Windows ympäristöissä.

Reaction Commercen dokumentaatiossa on listattuna kaikki ympäristömuuttujat, joita voi määritellä verkkokauppa-alustalle, kuten oletus pääkäyttäjä tai Reaction Commercen käyttämä virheiden kirjaustaso. Jotta verkkokauppa-alusta käyttää ulkoista palvelimella ylläpidettyä tietokantaa, tulee komennon 4 ympäristömuuttuja antaa parametrina mukaan komentoon, kun alustan kehitys aloitetaan. (Configuration 2018.) Näitä parametreja käytettiin toteutuksen julkaisuvaiheessa luvussa 3.6.

Varsinaisen tietokannan ja tietokantayhteyden lisäksi toteutettiin web-käyttöliittymä tietokantaa varten, josta voisi helposti tarkistaa dokumentteja toteutusvaiheessa. Lisäksi käyttöliittymä tarjoaisi karkean monitoroinnin tietokannan käytölle. (Features 2018.) Käyttöliittymään käytettiin avoimen lähdekoodin projektia nimeltä adminMongo. Se asennettiin palvelimelle, ja siihen määriteltiin lähiverkon osoite, josta se oli saatavilla. Osoitteelle tehtiin luvun 3.6 tapaisesti määritelmät nginx-palvelimen ja letsencryptin avulla, jonka jälkeen käyttöliittymä oli saatavilla julkisesta domainista (ks. kuvio 13).



Kuvio 13. Web-käyttöliittymä MongoDB-tietokannan selailua varten.

Verkkokaupan toteutus

Verkkokaupan ohjelmointiin ja määriteltyjen ominaisuuksien toteuttamiseen käytettiin useita valmiita ominaisuuksia sekä komponentteja, joita löytyy Reaction Commercesta. Muokkaukset verkkokauppa-alustan toiminnollisuuteen toteutettiin Reaction Commercen dokumentaation mukaisesti tuottamalla oma ylätason teema. Teeman alla voidaan verkkokauppa-alustaan *rekisteröidä* uusia komponentteja (ks. kuvio 14), korvata olemassa olevia valmiita komponentteja, tai tehdä tyylimuutoksia oletusteemaan.

```
import React from "react";
import { Components, registerComponent } from "@reactioncommerce/reaction-components";

const Loading = () => (
  <div className="spinner-container spinner-container-lg">
    <Components.CircularProgress indeterminate={true} />
  </div>
);

registerComponent("Loading", Loading);

export default Loading;
```

Kuvio 14. Komponentin rekisteröinti Reaction Commerceen.

Teema toteutetaan projektiin sille erikseen varattuun kansiorakenteeseen. Näin Reaction Commerce osaa sisällyttää muutokset, jotka korvaavat oletus komponentti- ja tyylitoteutuksia. Teema tulee julkaista Reaction Commercen käytettäväksi *register.js* -tiedoston avulla, joka sijaitsee teeman kansion juuressa (ks. kuvio 15).

```
// Tiedoston ja oman teeman sijainti projektissa:
// ./imports/plugins/custom/create-reaction-app/register.js
import { Reaction } from "/server/api";

// Oman teeman rekisteröinti
Reaction.registerPackage({
  // Nimi, joka esiintyy esimerkiksi asetuksissa
  label: "Create-reaction-app theme",

  // Uniikki nimi, jota käytetään tiedon hakemiseen tietokannasta
  name: "create-reaction-app-theme",

  // Kuvake työkalupalkissa
  icon: "fa fa-bars",

  // Automaattisen käytön asetus. Asettaa tietokantaan enabled: true
  autoEnable: true,

  // Lisäasetuksia
  settings: {},

  // Reitit ja muut rekisteröintiosiot suhteessa layouttiin
  registry: []
});
```

Kuvio 15. Teeman rekisteröinnin ilmoittava tiedosto.

Reaction Commerce käyttää dynaamisia tyylikielimäärityksiä LESS:in avulla. Kaikki verkkokauppa-alustan tyylimääritykset on määritelty LESS:in mahdollistaviin muuttujiin, jotka alustan käännösvaiheessa muuntuvat selaimille sopivaksi perinteisiksi CSS-tyylimäärityksiksi. Tämä mahdollisti työn toteutuksessa tyylimuutokset tehokkaasti, jolloin pohjana käytettiin Reaction Commercen tyylimuuttujien tiedostoa nimeltä *variables*. Tiedostosta otettiin kopio projektin teeman kansioon, jossa siihen voitiin tehdä muutoksia, jotka ylikirjoittivat oletustyylit. Tyylien muuttujien tiedostoon on määritelty esimerkiksi värit, fontit, tyylien *media queriet* eli kohdat, joissa tyyliin voidaan asettaa muutoksia käytettävän päätelaitteen ominaisuuksien mukaisesti, kuten

näyttökoon avulla. Lisäksi se sisältää useita Reaction Commercen komponenttien tyylimäärityksiä sekä paljon muita määrityksiä.

Tyylimuutosten lisäksi toteutettiin verkkokauppaan muutama lisäkomponentti. Sivun yläosaan päänavigoinnin alle tehtiin pieniä viestejä ja kielen vaihtamista sekä kirjautumispainikkeita varten oma lisänavigaatio. Komponentti voisi esittää esimerkiksi ajankohtaisia asioita kuten alennuskuponkeja tai muita tarjouksia. Päänavigaation alle toteutettiin myös uusi komponentti brändäystä varten, jossa voidaan esittää kuva sekä esimerkiksi yrityksen arvoja. Aivan sivun alaosaan tehtiin myös *footer* eli komponentti, jossa yleensä on hyödyllisiä linkkejä tai lisää navigaatio linkkejä. Footeriin toteutettiin tuotekategorioiden linkit, muita linkkejä sekä pohja yhteydenottolomakkeelle. Etusivun tuotelistaukseen tehtiin muutoksia, jossa tuotteet esitellään yksittäisinä komponentteina, joihin lisättiin pieni animaatio hiiren liikkeisiin reagoimaan. Tuotesivulle tehtiin myös muutoksia muun muassa ostoskorin painikkeeseen ja fontteihin.

Opinnäytetyöhön toteutettiin myös oma versio Reaction Commercen CLI eli komentokehoteohjelmasta, jolla esimerkiksi voi luoda uuden projektin, kuten luvussa 2.1.3 kerrottiin. Työn toteutuksessa tehtiin siis CLI:stä oma versio GitHubissa, jota kutsutaan *forkiksi*. Forkattuun versioon tehtiin tarvittavat muutokset, jotta projektiin valikoitu nimi *create-reaction-app* saatiin yhteen sopimaan työkalun kanssa, joka mahdollisti projektin selkeämmän erottelun perinteisestä Reaction Commerce projektista. Uusi CLI vastaa Reaction Commercen kehitystiimin tuottamaa ohjelmaa, mutta opinnäytetyötekijä halusi siitä sellaisen version, jolla projektin NPM paketin voi erottaa ja komennot saisivat yksilölliset termit. Lisäksi tämän toteutuksen ajateltiin auttavan jatkokehityksen aikana erottamaan Reaction Commercen ydin-toteutuksen ja työn tuotoksen mahdolliset virheet. Toisaalta koodimuutosten erottelu auttaisi projektin NPM pakettia käyttäviä kehittäjiä tunnistamaan paketit toisistaan ja kohdistamaan tukipyynnöt oikein ylläpidosta vastaaville henkilöille, kuten opinnäytetyötekijälle tai Reaction Commercen kehitystiimille.

Automatisointi

Kehittäjän kannalta verkkokaupan määrittelyyn asetettiin vaatimus, jossa kuvattiin verkkokaupan julkaisu ja sen automatisointi sekä uuden verkkokaupan tuottamisen aloittaminen skriptien avulla. Skriptin tarkoitus on vähentää manuaalista ihmistyötä, jossa skripti tai toisin sanoen *botti* hoitaa tehtävän käskettäessä.

Opinnäytetyöhön toteutettiin kaksi automatisoitua vaihetta, joista toinen perustaa paikallisen kehitysympäristön macOS-käyttöjärjestelmällä valmiiksi ja toinen, joka testaa ja julkaisee verkkokaupan tuotantoympäristöön palvelimelle. Skriptit vaativat kuitenkin pieniä muutoksia käyttäjältä ja esimerkiksi palvelimen asetukset sekä palvelimen oma ympäristö voivat erota suuresti, joten varsinkin julkaisun skripti vaatii käyttäjäkohtaista muokkausta. Pääpiirtein pohjat molemmille toimille kuitenkin toteutettiin.

3.4 Testaus

Opinnäytetyön projektin testaus toteutettiin Reaction Commercen verkkokauppa-alustaan määriteltyjen testien avulla sekä suorittamalla manuaalista ”käyttäjätestausta” opinnäytetyötekijän toimesta. Reaction Commerce tukee kahden tyyppisiä testejä, joita ydinkehitystiimi suosittelee. Toinen testeistä ei vaadi kehittäjältä erillistä määrittelyä, vaan se toimii, kun alustan normaalit asennukset, joita vaaditaan sen kehittämiseenkin, on tehty. Kyseinen testi on *integration test* eli integraatiotesti, jonka kehittäjä voi ajaa komentokehotteelta (ks. komento 5). Testi on toteutettu Mochalla, joka on Node.js toteutettu JavaScript-ympäristöjen testausalusta sekä Meteorin testityökalulla, joka hallinnoi yleisellä tasolla Meteor applikaatioiden testausta. Toinen Reaction Commercen dokumentaation ja ydinkehitystiimin suosittelema testaustapa on *acceptance test* eli hyväksyntätestaus. Hyväksyntätestaus suorittaa simuloituja käyttäjätoimia applikaation asiakaspäätä eli front-endia koskien, kuten kirjautumisia tai ostotapahtumia. Hyväksyntätestaus vaatii kuitenkin kehittäjältä useita

määrityksiä ja lisäasennuksia, ennen kuin testejä voidaan suorittaa. Hyväksyntätestaus käyttää toimiakseen Java SE -kehityspaketin, Selium Standalone Serverin, ChromeDriverin, Allure -testiraportoijan sekä valinnaisen BrowserStackin, jos tämä testi halutaan automatisoida. (Testing 2018.)

```
1 # Integraatiotestin suorittaminen
2 create-reaction-app test
```

Komento 5. Reaction Commercen integraatiotestin suorittava komento.

Käyttäjätestauksessa testattiin verkkokauppa-alustan toimintaa normaalein käyttäjätoimin kuten ostosten, kirjautumisen ja kaupan hallinnan muutoksien kautta, kuten käyttäjäryhmien oikeuksien oikeellisuuden testauksilla ja kaupan vero- ja maksuvaihtoehtojen kautta. Lisäksi kaupan toimintaa testattiin usealla päätelaitteella samanaikaisesti, minkä avulla pyrittiin ratkaisemaan mahdolliset eroavaisuudet eri laitteiden käyttäjäkokemuksen välillä.

3.5 Ylläpito

Opinnäytetyön projektia ylläpidetään GitHub -ohjelmistoprojektien versionhallintaa tarjoavalla verkkosivulla. Koko työn koodi on julkaistu avoimena lähdekoodina, kuten projektin pohjalla oleva Reaction Commerce myös on. Ohjelmistokehitys versioitiin ensin kehittämällä ominaisuuksia *development*-haaraan versionhallinnassa. Haarat ovat GitHub repositoryn sisäisiä toisistaan irtonaisia ”versioita”. Haaroihin voi tuottaa

esimerkiksi uusia ominaisuuksia, jotka sitten valmistuessaan yhdistetään eli *merge-tään* repositoryn *master* eli päähaaraan. Näin kehitystyö voidaan eriyttää irralliseksi, kunnes kyseinen osa, ominaisuus tai muu muutos on valmis. Lisäksi haaroihin tuotetut ominaisuudet voidaan katselmoida muiden projektia ylläpitävien toimesta ja vasta hyväksynnän jälkeen yhdistys voitaisiin toteuttaa.

GitHub *repositoryn* lisäksi projekti on saatavilla NPM -paketinhallinan kautta, jonka verkkosivu on npmjs.com. Projekti on saatavilla julkisena pakettina nimellä *create-reaction-app*. Paketin voi asentaa NPM:n komentokehoteohjelman avulla ajamalla komennon (ks. komento 6).

```
1 npm install create-reaction-app
2 # tai
3 npm i create-reaction-app
```

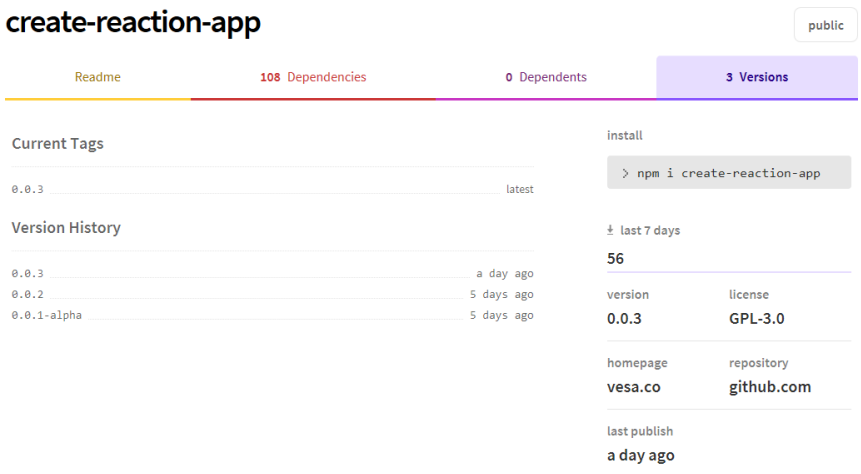
Komento 6. Opinnäytetyön NPM paketin asennuskomento.

Julkaiseminen NPM:ään on ilmaista, jos paketti on julkinen. Julkaisun voi suorittaa komentokehotteelta ajamalla komennon projektin kansiossa. Paketin versiointi sekä nimi tulevat NPM:n konfiguraatiotiedostosta *package.json*. Jokaisella NPM julkaistulla paketilla tulee olla uniikki nimi ja versionumero. Julkaiseminen tapahtuu ensin kirjautumalla tai rekisteröitymällä uudella käyttäjällä NPM -palveluun, jonka jälkeen voi ajaa julkaisukomennon tai paketin päivityksen sekä julkaisun (ks. komento 7).

```
# rekisteröinti
npm adduser

# tai kirjautuminen
npm login

# nykyisen käyttäjän tarkistaminen
npm whoami

# julkaisu
npm publish

# version päivitys ja julkaisu
npm version <versio> && npm publish
```

Komento 7. NPM kirjautuminen sekä julkaisun ja päivityksen komennot.

NPM mahdollistaa versioidun julkaisun GitHubissa olevasta projektista, jossa käyttäjät voivat asentaa paketin uusimman version helposti komennon avulla. Toisaalta, jos paketin uusimmassa versiossa on tunnettu virhe tai *bugi*¸ voivat käyttäjät asentaa esimerkiksi hieman vanhemman version, kunnes virhe uusimmassa versiossa on korjattu. Lisäksi NPM tarjoaa tilastoja paketin suosiosta ja laajentaa sen saatavuutta GitHubin käyttäjien ulkopuolelle (ks. kuvio 16).



Kuvio 16. NPMjs.com:ssa olevan opinnäytetyön NPM-paketin tietoja.

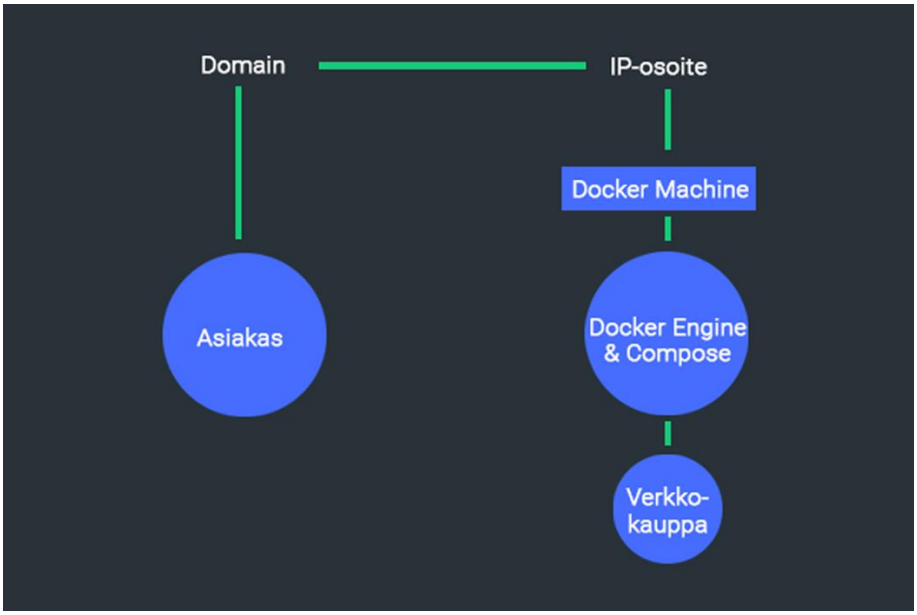
Ohjelmakoodin ja projektin ylläpito poikkeaa julkaistun version ylläpidosta, josta on kerrottu luvussa 3.6.

3.6 Julkaisu

Opinnäytetyön projekti julkaistiin DigitalOcean -PaaS-pilvipalveluntarjoajan ympäristössä. Ympäristö valikoitui opinnäytetyötekijän päätöksestä, sillä tekijällä on ylläpidossa ja julkaistuna oma verkkosivusto ja muutama oma rajapinta samalla palveluntarjoajalla, jonka lisäksi DigitalOcean tarjoaa useita hyvin dokumentoituja ratkaisuja ja palveluita edullisesti. Julkaisuun valittiin Docker-konttialusta, jossa PaaS-alustan konfiguraatioon valittiin neljän gigatavun välimuistillinen ja kahden virtuaalisen prosessorin versio, johon kuuluu 25 gigatavua SSD eli *solid state drive* tallennustilaa. Reaction Commercen palvelinvaatimuksissa oli ainoastaan dokumentoitu välimuistin vähimmäismäärä, joka oli kaksi gigatavua. (Deploying 2018.) Julkaisuympäristön palvelimeksi valikoitui siis Docker-konttialusta, johon tehtiin kuitenkin joitakin määrityksiä, sillä myöhemmin luvussa 3.6 esitettyjen ongelmien vuoksi jouduttiin työ julkaisemaan dokumentaation suosimasta tavasta poiketen.

Myös kehitysympäristöön tuli asentaa vaadittavat työkalut Dockeria varten, koska Dockerin konttitiedostoa pyrittiin tuottamaan paikallisella kehityskoneella lähtökohtaisesti. Työkalut oli dokumentoitu Reaction Commercen dokumentaatiossa. Docker vaatii sen omista työkaluista Docker Enginen, Docker Composen ja Docker Machinen. **Engine** tarjoaa ydintoiminnallisuudet Docker-konttitiedostojen koontia ja suorittamista varten. **Compose** määrittelee ja tarjoaa tuen monien konttien tuen samalle applikaatiolle samanaikaisesti, tarkoittaen, että esimerkiksi pääohjelman käyttämä kontti voi ladata muita konfiguraatioita toisen kontin sisältä. **Machine** automatisoi Dockerin verkkoisännän eli *hostin* pystyttämisen omassa verkossa tai pilvipalveluissa. (Docker 2018.) Docker luo siis kokonaisen käyttöjärjestelmän, jota ajetaan samaan tapaan kuin virtuaalitietokonetta ajettaisiin, joka yleensä tapahtuu yleensä palvelimella. Docker-ympäristö voi sisältää esimerkiksi jonkin Linux jakeluversion kuten

CentOS, Ubuntu tai muun käyttöjärjestelmän. Se voi sisältää jonkin applikaation sekä sen määritykset. Lisäksi Docker-ympäristö voi sisältää joitain palvelinmäärityksiä sekä muita yleisiä määrityksiä, jotka yhdessä muodostavat Dockerin konfiguraatiotiedoston mukaisesti määritetyn ympäristön, joka on kokonainen ja valmis käytettäväksi (ks. kuvio 17).



Kuvio 17. Asiakkaan pyyntö avata verkkokauppa Docker-arkkitehtuurilla kuvattuna (Docker 2018).

Reaction Commercen dokumentaatio tarjosi kaksi vaihtoehtoa julkaisua varten. Toisessa ja niin sanotussa ”helpossa” mallissa luodaan ensin uusi Docker-ympäristö DigitalOcean-pilvipalveluun, jonka jälkeen tehdään Docker-konttitiedosto, joka lopuksi julkaistaan palvelimelle. Lisäksi määritetään vielä Docker Machine kääntämään palvelimen IP-osoitteeseen tulleet kutsut Dockerin hostille. Lopuksi tuotantokäytössä tulisi määritellä palvelimen DNS eli *Domain Name Service,* joka välittää nimipalvelimen avulla domainin, eli esimerkiksi www.vesa.co, kutsut palvelimen IP-osoitteeseen, joka välittyisi siten Dockerille. (Mt.)

Toinen vaihtoehto julkaisua varten oli itse määriteltävä tapa tuottaa Docker-konttitiedosto, jossa kehittäjällä on täysi hallinta, mitä tuotetaan ja millaisilla määrityksillä. Opinnäytetyötekijä pyrki tällä vaihtoehtoisella tavalla tuottamaan Docker-konttitiedoston verkkokaupan julkaisua varten, joka tuotettaisiin jokaista julkaistavaa muutosta varten. Konttitiedoston tuottamiseen käytetty komento suoritettaisiin esimerkiksi paikallisesti kehitysympäristössä. Kehitysympäristöön tulisi tätä varten tehdä Docker-ympäristö. Opinnäytetyöntekijä toteutti ympäristön macOS käyttöjärjestelmällä testausta varten seuraavilla komennoilla. (ks. komento 8). Komento poikkeaa jonkin verran käyttöjärjestelmä- ja laitekohtaisesti. Opinnäytetyötä kehitettiin myös Windowsilla macOS:n lisäksi.

```
# Docker työkalujen asennus Brew-asennusohjelman avulla
brew install docker docker-compose docker-machine \
xhyve docker-machine-driver-xhyve
# Käyttöoikeuksien lisäys
sudo chown root:wheel $(brew --prefix)/opt\
/docker-machine-driver-xhyve/bin/docker-machine-driver-xhyve
sudo chmod u+s $(brew --prefix)/opt/docker-machine-driver-xhyve\
/bin/docker-machine-driver-xhyve
# Docker-ympäristön luominen
docker-machine create default --driver \
xhyve --xhyve-experimental-nfs-share
# Projektin kansiossa ajettava komento,
# joka osoittaa Docker-komennot oikeaan ympäristöön
eval $(docker-machine env default)
```

Komento 8. Docker-ympäristön luontiin käytetyt komennot macOS:lla.

Docker-konttitiedosto sisältää siis kaiken, mitä sen avulla luotu applikaatio tarvitsee toimiakseen, jolloin tuotteen julkaisu on helppoa ja voidaan olla varmoja, että version ympäristö toimii samaan tapaan monella eri palvelimelle, olettaen, että niiden Dockerit on luotu ja määritelty vastaavasti. Tämä mahdollistaa tehokkaan julkaisun, versioiden nopeat peruutukset, jos esimerkiksi uusi versio sisältää kriittisen vian ja versionhallinnan myös koodin lisäksi, jos konttitiedostot lisätään versionhallintaan. Docker käyttää erillistä tiedostoa, joka määrittelee konttitiedoston luonnin, jonka

nimi on *Dockerfile*. Tiedostoon voidaan määritellä aiemmin kerrotun mukaisesti esimerkiksi toisen Docker-tiedoston käyttäminen, tarvittavat komennot, kuten NPM ja Reaction Commercen asennus ja koontikomennot, ympäristömuuttujien määrittelyn komennot sekä tärkeimpänä komennot, jotka konttitiedosto suorittaa, kun se julkaistaan käyttöön (ks. komento 9).

```
# Esimerkki asennuskomennosta
RUN meteor npm install --production
# Esimerkki ympäristömuuttujista
ENV ROOT_URL "http://localhost"
ENV PORT 3000
# Portin määrittely, joka on sama,
# kuin Reaction Commerce käyttää
EXPOSE 3000
# Dockerin ajamat komennot, kun julkaistu
# Docker-konttitiedosto suoritetaan
CMD ["node", "main.js"]
```

Komento 9. Pieni osa Dockerfilen määrityksistä.

Opinnäytetyön toteutuksen loppuvaiheella määritettiin Dockerfileen tarvittavat määritykset, joilla työn verkkokauppa-alusta voitaisiin saada toimimaan. Dockerfile sisältää applikaatioon tehdyt muokkaukset ja omat määritykset, teemat ja muut toiminnallisuudet sekä tuotteet, käyttäjät ja tietokannan. Docker-konttitiedoston tuottamista pyrittiin toteuttamaan Reaction Commercen dokumentaation mukaisella komennolla (ks. komento 10). Konttitiedoston tuottaminen aiheutti opinnäytetyötekijälle ongelmia, sillä Dockerfilen määritykset oli vaikea saada oikein toimivan alustan saavuttamiseksi, vaikka ne olivat dokumentaation mukaiset sekä konttitiedoston muodostamisen suoritus on raskas kehitysympäristölle ja sen tietokoneelle. Lisäksi Reaction Commercen ollessa todella laaja web-applikaatio, vaati komennon suoritus paljon välimuistia. Prosessin suoritus kesti vähintään 15 minuuttia macOS kehitysympäristössä, kun käytössä oli 2015 vuoden 2-ytiminen 2,9 GHz Intel i5 prosessorilla, 8

Gt välimuistilla ja 512 Gt SSD-levyllä varustettu Apple MacBook Pro. Konttitiedoston muodostaminen osoittautui myös haasteelliseksi Windows ympäristössä tehokkaammasta tietokoneesta huolimatta.

```
# Konttitiedoston luonnin komento, jossa argumenttina
# annettiin Nodelle suuri määrä muistia
docker build --build-arg TOOL_NODE_FLAGS="--max-old-space-size=8192" \
# Konttitiedoston nimi
-t create-reaction-app:0.0.3 .
```

Komento 10. Dockerin konttitiedoston muodostaminen.

Suurin yksittäinen ongelma työn julkaisuvaiheessa oli Node.js:n vaatiman muistin suuri määrä, joka oli vaikea allokoida Dockerissa siten, että konttitiedoston tuottaminen onnistui. Tästä Node.js:n ja sen käyttämän V8 suoritusympäristön *heap memory* ongelmasta johtuen, ei opinnäytetyötekijä onnistunut tuottamaan toimivaa Dockerin konttitiedostoa. Tätä vaihetta edistettiin työn toteutuksessa usean viikon ajan, jossa lopputuloksena oli aina epäonnistunut komennon suoritus. Kyseistä komentoa ajettiin myös DigitalOceanin tarjoamalla pilvipalvelukoneella, johon määriteltiin runsaammin tehoa verrattuna paikalliseen kehitysympäristöön, joka ei kuitenkaan auttanut varsinaiseen ongelmaan. Dockerfileen määrityksiin kokeiltiin myös täysin alkuperäistä määritystä Reaction Commercen dokumentaation mukaisesti, josta huolimatta komento ei suoriutunut onnistuneesti. Opinnäytetyötekijä selvitti kyseistä ongelmaa useasta Reaction Commercen tarjoavasta kanavasta, jossa viaksi todettiin Meteor applikaation käyttämä suuri muistimäärä, mutta johon ei ole yksiselitteistä korjausta. Muun muassa muistin lisäallokointia testattiin, mutta tuloksetta.

Työn julkaisun viimeisessä vaiheessa Dockerin komentojen epäonnistuneiden suoritusten jälkeen päädyttiin julkaisu toteuttamaan siihen tapaan, kuin Node.js applikaatioita yleensä julkaistaan. Verkkokauppa-alustasta julkaistiin uusi versio GitHub -versionhallintaan, josta otettiin manuaalisesti *pull* eli päivitys palvelimella. Palvelin oli

sama, jolla aiemmin oli pyritty suorittamaan Dockerin komento. Node.js applikaation suoritus toteutettiin PM2 -nimisellä prosessien hallinnantyökalulla, jolla applikaation suorituksen komento saatiin ”valvontaan”. Valvonnalla tarkoitetaan sitä, että mahdollisen virheen ja applikaation prosessin sulkeuduttua, osaa PM2 automaattisesti käynnistää prosessin uudelleen. PM2 määriteltiin käyttäen komentoa (ks. komento 11). PM2 -prosessien hallintatyökalua käyttäessä erityisen tärkeää on antaa parametreina ympäristömuuttujat *MONGO_URL* ja *ROOT_URL*. Niillä määriteltiin Reaction Commercen käyttämä tietokantayhteys sekä verkkokaupan julkinen osoite. (Configuration 2018.) Tietokantayhteyteen määriteltiin opinnäytetyötekijän palvelimella sijaitsevan MongoDB:n osoite, jota käytettiin myös kehitystyössä työn toteutusvaiheessa. Näin tuotteet, käyttäjät ja kaikki muut tallennetut tiedot olivat yhtenäisiä.

```
1 MONGO_URL=mongodb://user:password@domain.com:27017/database_name \
2 ROOT_URL=domain.com \
3 pm2 start create-reaction-app -n joku-nimi
```

Komento 11. Komento, jolla ympäristömuuttuja saadaan prosessille käytettäväksi PM2 käyttäessä.

Näiden vaiheiden jälkeen oli Reaction Commercen verkkokauppa saatavilla palvelimella sisäisesti sen sisäverkossa eli osoitteessa *http://localhost:3000.* Jotta verkkokauppa olisi julkisesti saatavilla domainin takaa, tehtiin palvelimeen vielä *reverse proxy* eli päinvastainen välityspalvelin nginx-palvelimen ja DNS ojauksien avulla. Nginx määriteltiin ohjaus, jossa kutsu palvelimen domainiin ohjautuu edelleen palvelimella sisäverkon sisällä olevalle verkkosivulle, jonka sisältö tarjotaan vastauksena kutsuun. Lopuksi suojattiin yhteys käyttämällä letsencrypt -nimistä avoimen lähdekoodin projektia, jolla voi luoda SSL-varmenteita. Varmenteen luonnin jälkeen määriteltiin nginx käyttämään sitä, ja pakottamaan HTTPS-protokollan käyttämisen. Myös

DigitalOceanin hallinnassa lisättiin DNS ohjaukset halutuille ali-osoitteille, jossa ohjaus tehtiin oikealle palvelimelle. Lopputuloksena kutsu domainiin, joko *create-reaction-app.vesa.co* tai *shop.vesa.co*, ohjautui palvelimen IP-osoitteeseen, josta kutsu ohjautui edelleen palvelimella sisäisesti osoitteeseen localhost:3000. Kutsuun vastattiin verkkokaupan sisällöllä onnistuneesti suojatun yhteyden yli.

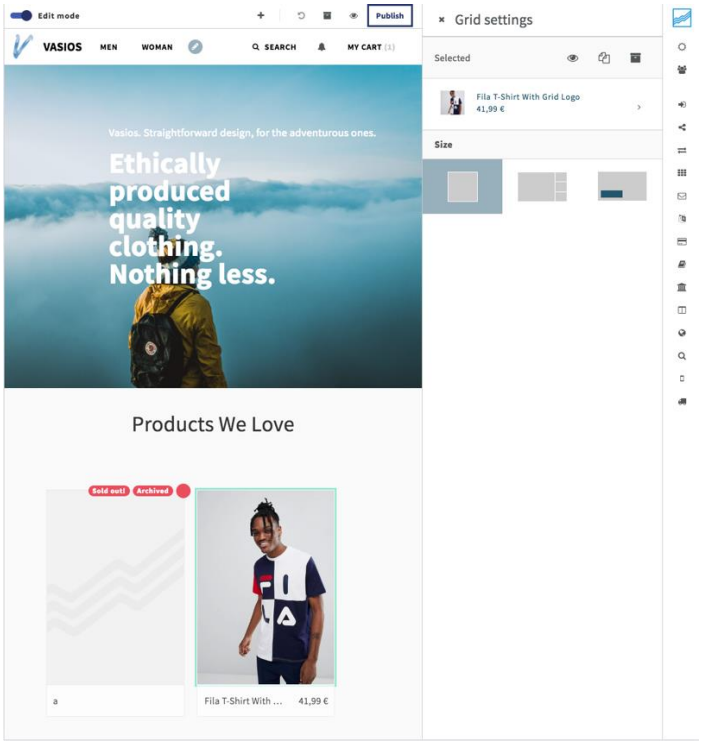
Opinnäytetyön tuotokset ja niiden julkaisut löytyvät opinnäytetyön valmistumishetkellä seuraavista osoitteista:

- Verkkokauppa ja sen verkkokauppa-alustaa esittelevä verkkosivu on työn julkaisuhetkellä nähtävissä osoitteessa **create-reaction-app.vesa.co** tai **shop.vesa.co**
- Julkinen NPM-paketti on saatavilla osoitteessa **npmjs.com/package/create-reaction-app**
- Julkinen GitHub repository on saatavilla osoitteessa **github.com/jussi-vesa/create-reaction-app**.

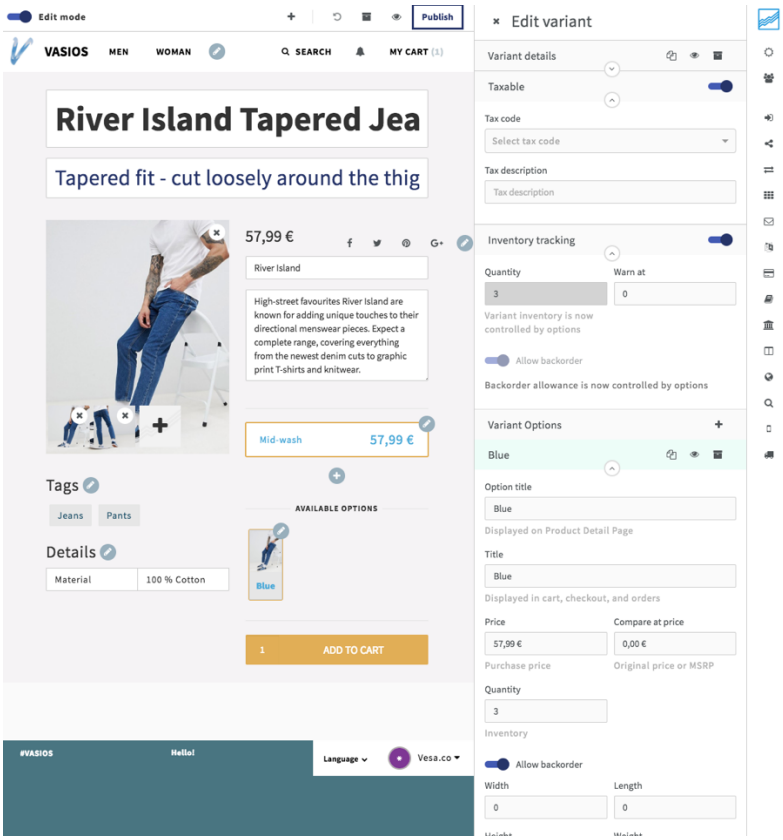
4 Selvityksen tulokset

Tuotos

Opinnäytetyön selvitystyön konkreettinen tuotos oli *proof of concept* -verkkokauppa, joka on toteutettu Reaction Commercen avulla (ks. kuvio 18). Verkkokauppa sisältää työn määrittelyssä ja suunnittelussa annetut kriteerit, jolloin kauppa on käytettävä ja siitä löytyy useimmat tarvittavat omaisuudet. Verkkokaupassa on tuotelistaukset, tuotteiden luonti (ks. kuvio 19) rekisteröityneet ja vierailevat käyttäjät, hallinnolliset käyttäjät, jotka ovat kuvattuna luvussa 3.1, maksu- ja postitustoiminnallisuudet sekä verkkokaupan kehittäjille ympäristön ja julkaisun automatisoinnin työkalut.



Kuvio 18. Tuotetun verkkokaupan etusivu.



Kuvio 19. Tuotteen muokkaussivu.

Verkkokauppa onnistuttiin toteuttamaan idean toimivaksi osoittamalla tavalla, jossa kauppa lunastaa vaatimukset. Lisäksi verkkokaupan design ja käyttäjäkokemus toteutettiin määritelmien ja etenkin käyttäjätarinoiden pohjalta. Tuotoksen ollessa *proof of concept* -esitys kaupasta, on siinä kuitenkin joitain puutteita viimeistelyssä.

Tulosten puutteet

Selvitystyön tuloksiin ei opinnäytetyötekijä selvittänyt suorituskyvyllisiä testejä, joissa olisi voitu vertailla Reaction Commercen suoriutumista esimerkiksi tuhansien tuotteiden laajuisten kauppojen osalta, verraten yleisimpiin verkkokauppa-alustoihin. Näiltä osin työssä ei selvinnyt, kuinka hyvin Reaction Commerce suoriutuu tai olisiko suorituskyvyn selvityksellä saatu esille erilaisia tuloksia sen soveliaisuudesta tuotantokäyttöön.

Opinnäytetyössä ei toteutettu laajaa ja yksinomaan Reaction Commercen ydinkomponentteihin kohdistuvaa muokkausta, josta johtuen työssä ei selvinnyt Reaction Commercen joustavuus niiltä osin, jossa alusta päivittyy uudelle pääversiolle. Työssä selvisi kuitenkin, että jos edellä kuvattuja muutoksia toteutettaisiin, ne suoritettaisiin samoin kuin muutkin muutokset, eli muutokset palvelisivat *pluginina* ja olisivat irrallinen osa Reaction Commercen pääkoodikantaan nähden. Tämän selvityksen puuttuessa ei voida kuitenkaan tehdä johtopäätöstä vaikutuksista päivitysprosessiin, joka on osa tuloksien puutteellisuutta.

Tuloksista puuttuu myös tarkempi selvitys kuluttajan yksilöllisen kokemuksen huomioimisesta. Vain osaa tätä käsitettä selvitettiin etenkin istuntojen säilymisen kannalta, mutta tarkempaa selvitystä esimerkiksi käyttäjän seurannasta ja käytöksen verkkokaupassa aiheuttavista tapahtumista ei tehty.

Lopuksi tuloksissa on puutteita koskien Docker julkaisua, sillä työn verkkokauppaa ei onnistuttu julkaisemaan Dockerin avulla. Kuitenkin osittainen tulos voidaan todeta, koskien julkaisun ongelmallisuutta Docker-ympäristöissä. Työssä ei pystytty näistä ongelmista sekä resurssien puitteissa selvittämään, kuinka julkaisu voisi toimia tuotantokäytössä.

Tulokset, hyödynnettävyys & muiden tutkimustulosten vertailu

Opinnäytetyö osoitti, että Reaction Commerce on tarpeeksi eheä ja luotettava tuotantokäyttöön. Vakaan toiminnan ja vähäisten ongelmien määrän avulla opinnäytetyötekijä kokee, että alusta on sovelias tuotantokäyttöön. Lisäksi julkaisu tapahtuu alan hyväksi todettujen tapojen mukaisesti, jossa versiointi ja kaupan julkaisu sekä julkaisujen peruutus on toteutettu arkkitehdillisesti turvallisesti konttiympäristön avulla. Julkaisu ei kuitenkaan ole ongelmaton ja suuryritykset voisivat hyötyä Reaction Commercen tarjoamasta maksullisesta Reaction Platform PaaS-alustakokonaisuudesta. Reaction Commerce on reaktiivinen ja reaaliaikainen, moderneilla teknologioilla ja kattavalla dokumentaatiolla julkaistu avoimen lähdekoodin verkkokauppa-alusta.

Se onnistuu kehittäjän näkökulmasta tarjoamalla kiitettävällä tasolla olevan dokumentaation ja sellaisen teknologiapinon, jolla itseään jatkuvasti kehittävät ohjelmoijat ovat tottuneet työskentelemään ja, joka on todettu muissakin isoissa projekteissa tai organisaatiossa onnistuneiksi valinnoiksi. Reaction Commerce onnistuu myös siten, että kehittäjällä on täysi hallinta kaupan toteutuksesta, hän voi osallistua alustan kehitykseen sen ollessa avoimen lähdekoodin projekti ja saada apua useasta Reaction Commercen ydinkehitystiimin tarjoamista kanavista, kuten *community calleista* tai foorumeilta. Reaction Commercen ollessa reaktiivinen ja reaaliaikainen, selvityksessä huomattiin, että kehittäjä pystyi seuraamaan muutosten vaikutusta usealla päätelaitteella vähäisellä vaivalla.

Kuluttajan ja hänen käytettävyyden näkökulmasta Reaction Commerce onnistuu tarjoamalla kaikki yleisimmät toiminnot, joita kuluttaja verkkokaupassa asioidessa voisi vaatia. Alusta tarjoaa ostosten tekemisen myös ilman rekisteröintiä ja kuluttaja voi palata kauppaan myöhemmin, jossa kauppa osaa palvella edellisen istunnon mukaisesti tarjoten yksilöllisen käyttäjän kokemuksen, kuten täyttämällä ostoskorin edellisillä tuotteilla tai muistamalla osoitetiedot. Kauppa toimi luotettavasti sisältäen normaalit verkkokaupan toiminnot mukaan lukien alennuskupongit, verotuksen, postituksen ja tilauksen tilan sekä ilmoitukset, joita kuluttaja verkkokauppaa käyttäessään odottaa.

Opinnäytetyössä selvitettiin, että Reaction Commerce on vähintään lievästi ongelmallinen kehittäjän näkökulmasta, sillä kehitystyön huomattiin olevan paikalliselle tietokoneelle verrattain raskasta. Kehitystyössä tehtyjen ohjelmakoodimuutosten heijastuminen reaaliaikaisesti eli niin sanottu *live reload* -ominaisuus saattoi kestää ajoittain jopa kymmeniä sekunteja, ennen kuin muutokset tulivat voimaan. Reaction Commercen asennuksen havaittiin olevan hidas, sekä kehitystyön aloitus jo olemassa olevassa ympäristössä oli keskiverto JavaScript projekteja hitaampi. Myös julkaisuversion konttitiedoston muodostaminen on hidasta ja prosessi vaatii isäntätietokoneelta paljon välimuistia sekä prosessointikykyä. Konttitiedoston muodostaminen oli tässä työssä erityisen ongelmallista. Ongelmaan liittyen selvisi, että vastaavia ongelmia on esiintynyt myös muilla kehittäjillä, mutta varsinaisia täysesteitä julkaisuun ei opinnäytetyötekijälle tullut vastaan eri kanavia lukiessaan.

Kuluttajan selatessa verkkokauppaa, opinnäytetyössä selvitettiin, että Reaction Commerce -verkkosivu latautuu kohdeselaimelle hitaasti, jolloin niin sanottu *first meaningful paint* -suorituskyky on huono eli ensitila, jolloin sivulla on merkittävää sisältöä kuluttajalle, oli hidas. Kuluttaja odottaa sivustolta nopeaa vastetta (Li ym. 2016, 926). Opinnäytetyötekijä kokee ensilatauksen hitauden olevan merkittävä Reaction Commercen heikkous.

Reaction Commerce eroaa yleisimpiin verkkokauppa-alustoihin kaikkein selvimmin siinä, että se tarjoaa välittömät muutosten heijastumiset kaikille verkkokauppaa käyttäville asiakkaille ja muille käyttäjäryhmille reaaliaikaisesti. Opinnäytetyössä löydettiin, että esimerkiksi tuotteen tuotekuvaa vaihtamalla, julkistettiin uusi kuva välittömästi kaikille verkkokauppaa selaaville käyttäjille, ilman, että kohdeselain tekisi erillistä sivun vaihdosta tai sivun uudelleen latausta. Käytännössä tällä mekanismilla saavutetaan välitön vaikutus, jos kaupassa selviää kriittinen virhe, kuten tuotehinnan tai varastosaldon poikkeuksellinen virheellisyys, joihin tehdyn korjauksen toivottaisiin astuvan heti voimaan.

Opinnäytetyön soveltavan tutkimuksen tuloksien hyödynnettävyys kohdistuu heille, joille Reaction Commerce ei ole entuudestaan tuttu. Työn tuloksista voidaan hyödyntää Reaction Commercen soveliaisuus tuotantokäyttöön, jos esimerkiksi kartoitettai-

siin uuden projektin verkkokauppa-alustaa työssä esitettyjen luvun 2.1.2 verkkokauppojen välillä. Lisäksi Reaction Commercen vaatima sovellusarkkitehtuuri on kuvattu luvuissa 2.1.3 ja 3.3–3.6, ja kuvauksien avulla voidaan määritellä kehitys- ja tuotantoalustat käytettäväksi Reaction Commercen kanssa. Lisäksi tuloksista voidaan huomioida verkkokauppa-alustan ongelmat etenkin suorituskyvyn ja julkaisun osalta. Erityisesti opinnäytetyön tuloksista voidaan hyötyä, jos tehdään vertailua eri verkkokauppa-alustojen välillä, sillä Reaction Commerce opinnäytetyön tuloksissa selvisi sen reaktiivisuus ja reaaliaikaisuus, joita muut verkkokauppa-alustat eivät tarjoa. Konkreettisesti katsottuna tuloksista voidaan hyödyntää opinnäytetyössä tuotettua projektia, jota voi käyttää uuden verkkokauppaprojektin pohjana.

5 Pohdinta & johtopäätökset

Opinnäytetyön tutkimustehtävänä oli selvittää, kuinka hyvin Reaction Commerce suoriutuu yleisimmistä verkkokauppa-alustan vaatimuksista sekä saada ymmärrys, miten se soveltuu käytettäväksi tuotannossa. Lisäksi opinnäytetyötekijä halusi saada yleiskuvan siitä, kuinka Reaction Commerce eroaa perinteisistä verkkokauppa-alustoista. Soveltavassa tutkimustyössä opinnäytetyötekijä selvitti, että Reaction Commerce täyttää kaikki verkkokauppa-alustan vaatimukset, sillä se sisältää holistisesti kaikki ne ominaisuudet, joita verkkokaupalta vaaditaan mukaan lukien vakaan toiminnan ja vähintään tyydyttävän hyvän suorituskyvyn.

Löydetyt haasteet & puutteet

Opinnäytetyötekijä huomasi työn aikana, että sähköisissä tietokannoissa ei ollut kuin harvoja artikkeleja, julkaisuja tai kirjoja työn luvussa 2.1.2 ja 2.1.3 esitellyistä verkkokauppa-alustoista. Tästä syystä työn lähteisiin on käytetty kyseisiä aiheita selvittäessä

verkkokauppa-alustojen omia dokumentaatioita ja muita sivustoja, joista on löydetty tietoa. Nämä poimitut tiedot eivät kuitenkaan perustu tutkimuksiin, joten verkkokauppa-alustojen puhtaasti teoreettinen vertailu ei ollut mahdollista ja alustoista kerrottiin sen sijaan pääpiirteittäin niiden tukemat ominaisuudet, jonka lisäksi työntekijä avasi hieman kehittäjän näkökulmasta kunkin verkkokauppa-alustan ominaisuuksia. Etenkin Reaction Commercen ollessa nuorin opinnäytetyössä selvitetyistä verkkokauppa-alustoista, oli siitä tiedon löytäminen paikoin jopa vaikeaa, eikä aiempia tutkimuksia siitä ollut olemassa.

Reaction Commercen lyhyen markkina-ajan myötä teoreettisen tiedon vähäisyyden lisäksi opinnäytetyötekijä huomasi, että kehitystyön aikana vastaan tulleisiin ongelmiin oli vaikeaa löytää vastauksia. Kuitenkin Reaction Commercen ollessa avoimen lähdekoodin ohjelmisto, voi ongelmista tehdä uuden GitHub -ongelman eli *issuen*, jonka kautta voi saada yhteisöltä tai Reaction Commercen pääkehittäjiltä apua. Ongelmailmoitusta merkitessä tulee kuitenkin miettiä tarkkaan sen tarpeellisuus ja sitä ennen tulee aina pyrkiä hakemaan tietoa muualta ensin.

Aiheena verkkokaupat ja verkkokauppa-alustat ovat itsessään laaja käsite. Opinnäytetyötekijä halusi kuitenkin selvittää oleellisesti aiheeseen liittyviä käsitteitä, kuten luvun 2.4 alaluvut. Tämä sekä muiden teknologioiden selvitys luvussa 2.2 kasvattivat työn tietoperustaa ja näin ollen ne laajensivat työn, joka aiheutti opinnäytetyöntekijälle haasteen, jossa työn laadun ja laajuuden tasapainottaminen työn aikataulun kanssa oli haastavaa. Opinnäytetyötekijä kokee, että työn laajuus ei heikennä työn laatua merkittävästi sekä laajuus on työn tulosten kannalta positiivinen osa työtä.

Opinnäytetyön konkreettisen tuotoksen toteutusvaiheessa resurssien puitteissa ei ollut mahdollista toteuttaa verkkokauppa-alustaa käyttävän yksittäisen kuluttajan seurantaan ja käytöksen vaikutuksen heijastumisen toimintoja tai ominaisuuksia. Työn tavoitteissa oli selvittää yksilöllistä kokemusta verkkokauppa-alustalla, johon ei toteutettu erillisiä ominaisuuksia, jonka johdosta työn toteutus tältä osin on puutteellinen. Lisäksi toteutuksessa ei ollut mahdollista tuottaa julkaisua Dockerin avulla onnistuneesti, mutta opinnäytetyötekijä kokee epäonnistumisen kohdistuvan verkkokauppa-alustaan, ennemmin kuin opinnäytetyötekijän taitoihin tai yrittämisen puutteeseen.

Opinnäytetyön myötä voidaan todeta, että löydettiin osittainen uusi ongelma-alue, jossa verkkokaupan julkaisu muodostui haastavaksi. Tämä johtui Node.js:n tarvittavasta välimuistin määrästä, mutta ongelmaa ei onnistuttu korjaamaan dokumentoitujenkaan muistin lisäallokointien jälkeen. Opinnäytetyötekijä kokee, että ongelma sijaitsee Meteor -ohjelmistokehyksessä, sillä Reaction Commercen arkkitehtuuri on toteutettu sen avulla, ja asiakas- sekä palvelinpuolen logiikka nojautuu siihen. Opinnäytetyötekijä uskoo, että kyseiset ongelmat ovat suhteellisen tuore aihe, sillä JavaScript ja Node.js applikaatioiden suosio sekä ennen kaikkea laajuus ja monimutkaisuus ovat skaalautuneet ylöspäin viimevuosina runsaasti. Aiheena laajat JavaScript ja Node.js applikaatiot ja niiden vaatimat suoritusympäristöt ovat vähintään osittainen uusi ongelma-alue, joita voisi tutkia tarkemmin.

Löydetyt oivallukset & onnistumiset

Opinnäytetyötekijä koki, että työ oli sopivan haastava niin ohjelmoinnin ja yleisesti ohjelmistokehityksen, kuin selvitystyön kirjallisenkin osion osalta, jossa työntekijä oppi paljon tärkeitä ja ajankohtaisia asioita verkkokaupoista, palvelumuotoilusta, suunnittelusta, designista ja tietenkin Reaction Commercesta ja sen teknologoista. Opinnäytetyötekijä koki Reaction Commercen olevan hyvin nykyaikainen, joustava, kattava ominaisuuksiltaan ja kelvollinen tuotantokäyttöön.

Työssä opittiin, millaisilla menetelmillä uusia ohjelmistoja määritellään ja suunnitellaan asiakaslähtöisesti sekä käyttäjätarinoiden, tapauskuvauksen, palvelumuotoilun ja yleisen designin ja käyttöliittymäsuunnittelun avulla. Näitä työkaluja hyödyntäen saadaan holistinen kuva, millaista tuotetta ollaan saavuttamassa ja mitä se vaatii. Idean ytimessä on asiakas ja muut tuotetta välittömästi käyttävät ihmiset, kuten verkkokaupan tapauksessa esimerkiksi kaupan ylläpitäjä. Opinnäytetyötekijä uskoo, että hyvä suunnitelma nimenomaan käyttäjälähtöisesti on oikea tapa suunnitella tuote tai palvelu sen ketteryyden ja oikeisiin asioihin keskittymisen takia. Asiakkaan huomioiminen on erityisen tärkeää etenkin verkkokaupoissa, sillä yksinomaan asiakas voi helpoiten tuottaa yrityksen liiketoiminnalle tuottoa.

Opinnäytetyössä onnistuttiin erityisen hyvin selvittämään perusteellisesti Reaction Commercen pääeroavaisuudet muihin verkkokauppa-alustoihin sekä kuluttajan että kehittäjän näkökulmasta. Työn tietoperustaan selvitettiin laajasti teknologisia käsitteitä sekä tuotteistamisen metodeja, joista opinnäytetyötekijä on tyytyväinen ja kokee niiden lisäävän työn arvoa. Lisäksi opinnäytetyössä onnistuttiin kiitettävällä tasolla selvittämään, millaisia teknologioita Reaction Commerce käyttää toimiakseen, mitä teknologiat vaativat esimerkiksi julkaisualustalta ja kuinka näiden avulla saavutetaan Reaction Commercen päävahvuus eli reaktiivisuus ja reaaliaikaisuus sekä toisaalta näiden teknologiavalintojen asettamat haasteet ja ongelmat alustan raskauden kustannuksella.

Eettisyys & luotettavuus

Opinnäytetyön raportin sisältö ei sisällä kenenkään henkilön tai tutkimuksen tai muun kirjallisuuden plagiointia, eikä työn sisältö johdata harhaan tai jätä pois oleellisia tai merkittäviä seikkoja. Selvityksen tekemiseen ja sen tuloksiin on sisällytetty kaikki löydetyt havainnot eikä saavutettuja tuloksia ole vääristetty.

Työssä käytetyt lähteet on merkitty asiaan kuuluvalla tavalla ja muiden tutkijoiden tuotoksia ei vähätellä. Lähdeviite ilmoitettiin tekstissä siinä kohdassa, jossa lähteen sisältöä on käytetty, jonka lisäksi lähde on merkitty tekstiin viitteeksi sekä lähdeluetteloon. Lähteiden sisältöjä ei ole vääristetty. Työhön on käytetty laajasti lähteitä usealta taholta, eikä siinä ole sulkeuduttu käyttämään yksittäisten tutkijoiden tai organisaatioiden tuottamiin tutkimuksia tai muita lähdeviitteitä kuten verkkosivuja. Lähdeviitteissä on pääosin vieraskielisiä viitteitä. Opinnäytetyö on kirjoitettu neutraaliin sävyyn, eikä siinä loukata ketään ryhmää, vähemmistöä tai yksittäistä ihmistä. Työn pohdinnassa on osittain ilmaistu opinnäytetyötekijän ajatuksia työn muusta neutraalista sävystä poiketen.

Yhteenveto, kehitysideat & jatkotutkimusaiheet

Opinnäytetyön myötä voidaan todeta, että Reaction Commerce on reaktiivinen ja reaaliaikainen, moderneilla teknologioilla ja kattavalla dokumentaatiolla julkaistu verkkokauppa-alusta. Reaction Commerce sisältää kuluttajan ja kaupan hallinnoinnin henkilöiden näkökulmasta kaikki ne ominaisuudet, kuin odotetaan. Kehittäjälle Reaction Commerce on joustava ja teknologisesti moderni sekä kattava. Reaction Commercessa on suorituskyvyn ongelmia kehitystyössä koodimuutosten päivittymisessä. Sivun esilatauksessa on hitautta kuluttajan vieraillessa sivulla. Tuotantojulkaisun muodostaminen on vähintään hidasta ja vaatii verrattain tehokkaan kehitysalusta. Ongelmat ovat kuitenkin alustan hyötyihin verrattuna pieniä, ja voidaan todeta, että ne eivät merkittävästi heikennä Reaction Commercen asemaa miljardimyynnin kilpaillussa alalla (Cassidy & Hamilton 2016, 1054).

Kaiken kaikkiaan opinnäytetyön tehtävän ja tavoitteen täyttäminen onnistui aiheen laajuudesta ja suhteellisen tiukasta aikataulusta ja resursseista huolimatta. Opinnäytetyö valmistui ajallansa ja aikataulun mukaisesti. Opinnäytetyötekijä pitää saavutettuja tuloksia ja kirjallista raporttia pääosin onnistuneena ja kokee aiheen erittäin kiinnostavaksi, haastavaksi ja teknologisesti Reaction Commercen myötä nykyaikaiseksi. Työn teoreettiset viitteet ja uusien teknologioiden kautta selvitetyt asiat mahdollistivat opinnäytetyötekijän ammentamaan paljon ammatillista oppia sekä henkilökohtaista taitoa laajojen käsitteiden hallinnoinnin ja ymmärtämisen osalta. Reaction Commerce on myös laajalti tutkimaton, jonka takia opinnäytetyötekijä pitää työn aihetta tärkeänä ja näiltä osin tuloksia erityisen hyödyllisinä myös ammattialalle.

Opinnäytetyön tekemisen aikana havaittiin kehitysideoita, joita olivat yleisesti verkkokauppa-alustan jatkokehitys ja muokkaaminen persoonallisemmaksi, mutta työn aikataulun puitteissa laajempi kehitys ei ollut mahdollista. Työn tulosten jatkokehityksen kannalta olisi tärkeää muodostaa tiiviimpi ymmärrys, kuinka Reaction Commerce toimisi optimaalisimmin. Myös Reaction Commercen jatkokehityksen seuranta olisi tärkeää jatkokehitysmielessä. Verkkokauppa-alustaan on tulossa tuki edustapuolen GraphQL-haulle, jota käyttämällä voisi keventää Reaction Commercen kehitystyötä korvaamalla Meteorin DDP:n. Meteorin aiheuttamiin suorituskyvyn ongelmiin haetaan ratkaisuja suorituskyvyn parantamisen tai teknologiasta pois siirtymisen

avulla, josta ei ole vielä tehty päätöstä. Lisäksi Reaction Commercen kehitystiimi tekee opinnäytetyön julkaisun aikaan paljon muutoksia paremman suorituskyvyn puolesta, joka on usean tuotteen samanaikaisen muokkaamisen kanssa seuraavana tulevien ominaisuuksien listalla. Jatkokehityksessä tulisi seurata näitä tulevia julkaisuja sekä niiden tuomia etuja olisi edullista hyödyntää jatkossa.

Opinnäytetyön soveltavan tutkimuksen jatkotutkimusaiheita opinnäytetyötekijän näkökulmasta olisi selvittää, kuinka yksittäistä kuluttajaa voisi huomioida hänen vierailessa verkkokaupassa paremmin ja kuinka tämä yksilöllinen tarkkailu voisi heijastua konversioon ja yleisesti asiakkaiden mieltymykseen asiointia kohtaan. Verkkokauppa-alustan julkaisun optimoimisen jatkoselvitys olisi tärkeä osa työn kannalta, jos Reaction Commercella kehitettäisiin tuotantokäyttöön verkkokauppoja oikeille asiakkaille. Reaction Commercen julkaisuun on olemassa työssäkin käytetty suositeltu ja hyvin dokumentoitu malli, joka kuitenkin vaatii optimointia ja selvitystä sen ongelmallisuudesta johtuen. Viimeisenä jatkotutkimusaiheena olisi selvittää suurien JavaScript ja Node.js applikaatioiden vaatimien ympäristöjen- ja teknologioiden vaatimuksia, jotta voitaisiin ratkaista Reaction Commercen kaltaisten projektien optimaalinen toiminta.

Lähteet

About Shopify. N.d. Yleistietoa Shopifyn www-sivuilla 31.3.2018. Viitattu 31.3.2018. https://www.shopify.com/press.

About us. N.d. About -sivu Linked In www-sivuilla. Viitattu 17.2.2018. https://www.linkedin.com/company/reaction-commerce.

About WooCommerce. N.d. Proudly bootstrapped, and built on a foundation of work-life balance, we have ambitions to be the ultimate WordPress toolkit provider. Yrityksen aikajana WooCommercen www-sivuilla. Viitattu 31.3.2018. https://woo-commerce.com/about/.

Admin API reference. N.d. Admin rajapinnan dokumentaation pääsivu Shopifyn www-sivuilla. Viitattu 31.3.2018. https://help.shopify.com/api/reference.

An open source eCommerce plugin for WordPress. 2018. WooCommercen GitHub www-sivu. Viitattu 31.3.2018. https://github.com/woocommerce/woocommerce.

Byron, L. 2015. GraphQL: A data query language. Blogipäivitys GraphQL www-sivuilla. Viitattu 29.3.2018. http://graphql.org/blog/graphql-a-query-language/.

Casciaro, M. & Mammino, L. 2016. Node.js Design Patterns. 6–11. p. 2. Google Books. Viitattu 22.3.2018. https://books.google.fi/books?id=55WqDQAAQ-BAJ&dq=nodejs&hl=fi&source=gbs_navlinks_s.

Cassidy, L. & Hamilton, J. 2016. A design science research approach to website benchmarking. James Cook University, Australia. Benchmarking: An International Journal, 23, 5, 1054–1075. Viitattu 15.2.2018. https://janet.finna.fi. EBSCOhost.

Chodowor, K. 2013. MongoDB: The Definitive Guide. Powerful and Scalable Data Storage. 1–5. p. 2. Google Scholar. Viitattu 29.3.2018.https://scholar.google.fi/.

Community. 2018. Dokumentaation sivu yhteisöstä Reaction Commercen www-sivuilla. Viitattu 7.4.2018. https://docs.reactioncommerce.com/reaction-docs/mas-ter/community-channels.

Configuration. 2018. Dokumentaatio Reaction Commercen www-sivuilla. Viitattu 5.5.2018. https://docs.reactioncommerce.com/reaction-docs/master/configuration.

Create the online store you want. N.d. Create the online store you want with power-ful features that come straight out of the box with our free WooCommerce plugin. WooCommercen ominaisuussivu heidän www-sivuilla. Viitattu 30.3.2018. https://woocommerce.com/features.

Creating a Custom Payment Provider. 2018. What is a ”Payment Provider”. Doku-mentaatiosivu Reaction Commercen www-sivuilla. Viitattu 9.5.2018. https://docs.re-actioncommerce.com/reaction-docs/master/creating-a-payment-provider.

Definition of design in English. N.d. Sanan määritelmä Oxfordin sanakirjan www-sivuilla. Viitattu 2.4.2018. https://en.oxforddictionaries.com/definition/design.

Demandware Revenue, Profits - DWRE Annual Income Statement. 2016. Taloustietoja Amigo bullsin www-sivuilla. Viitattu 2.4.2018. https://amigobulls.com/stocks/DWRE/income-statement/annual.

Deploying. 2018. Dokumentaatio Reaction Commercen www-sivuilla. Viitattu 21.4.2018. https://docs.reactioncommerce.com/reaction-docs/master/deploying.

Deploying Reaction using Docker. 2018. Dokumentaatio Reaction Commercen www-sivuilla. Viitattu 21.4.2018. https://docs.reactioncommerce.com/reaction-docs/master/deploying-reaction-using-docker.

Design Sprint – Testattu prototyyppi alle viikossa. N.d. Kirjoitus Meomin www-sivuilla. Viitattu 5.4.2018. https://www.meom.fi/design-sprint/.

Desyatnyuk, V. 2018. ECommerce Platforms Comparison 2018: Shopify vs Magento vs WooCommerce vs OpenCart vs PrestaShop. Top 5 eCommerce platforms comparison | Which is the best eCommerce platform?. Artikkeli Cart 2 Cart www-sivuilla. Viitattu 31.3.2018. https://www.shopping-cart-migration.com/blog/61-must-know-tips/ecommerce-platforms-comparison-shopify-vs-magento-vs-woocommerce-vs-opencart-vs-prestashop.

Docker. 2018. Dokumentaatio Reaction Commercen www-sivuilla. Viitattu 27.4.2018. https://docs.reactioncommerce.com/reaction-docs/master/deploying-reaction-using-docker.

Dudovskiy, J. N.d. Applied Research. Artikkeli Research Methodologyn www-sivuilla. Viitattu 2.4.2018. https://research-methodology.net/research-methodology/research-types/applied-research/.

ECMA International. 2017. Standard ECMAScript 2017 Specification, 8, 56–57. Viitattu 17.2.2018. https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

Ecommerce Usage Statistics. 2018. Statistics for websites using Ecommerce technologies. Verkkokauppateknologioiden tilastoja Built Within www-sivuilla. Viitattu 30.3.2018. https://trends.builtwith.com/shop.

Everything you need to start an online store and sell online. N.d. Verkkokaupan ominaisuukia Shopifyn www-sivuilla. Viitattu 31.3.2018. https://www.shopify.com/online.

Features. 2018. AdminMongon GitHub sivu GitHubin www-sivuilla. Viitattu 11.5.2018. https://github.com/mrvautin/adminMongo.

Finley, K. 2014. An open source platform that makes building apps cheap and easy. Artikkeli Wired www-sivuilla. Viitattu 27.3.2018. https://www.wired.com/2014/10/meteor/.

Galitz, W.O. 2007. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. 41–57 p. 3. Google Books. Viitattu 14.3.2018. https://books.google.fi/books?hl=fi&lr=&id=Q3Xp_Awu49sC&oi=fnd&pg=PR5&dq=user+interface+design+process&ots=l-4ZJZ9k16&sig=Z8i6ZFZCkP5oOBSvL1ylz53Xj94&redir_esc=y#v=onepage&q=user%20interface%20design%20process&f=false.

Global Online Clothing Stores. N.d. Clothing Stores using Premium Technology. Vaatteita myyvien verkkokauppojen maksullisten teknologioiden listaus Built Within www-sivuilla. Viitattu 30.3.2018. https://builtwith.com/ecommerce/global/clothing.

Google Trends: LemonStand, Squarespace Commerce, PinnacleCart, Big Cartel, Ecwid, koko maailma, viimeiset 5 vuotta. 2018. Hakutilastotietoja Googlen Trens www-sivulla. Viitattu 31.3.2018. https://trends.google.com/trends/explore?date=today%205-y&q=LemonStand,SquareSpace%20Commerce,PinnacleCart,Big%20Cartel,Ecwid.

Google Trends: Merchium, SellBeing, Sylius, koko maailma, viimeiset 5 vuotta. 2018. Hakutilastotietoja Googlen Trends www-sivulla. Viitattu 31.3.2018. https://trends.google.com/trends/explore?date=today%205-y&q=Merchium,SellBeing,Sylius.

Google Trends: Shopify, WooCommerce, Magento, BigCommerce, Reaction Commerce, koko maailma, viimeiset 5 vuotta. 2018. Hakutilastotietoja Googlen Trends www-sivulla. Viitattu 31.3.2018. https://trends.google.com/trends/explore?date=today%205-y&q=Shopify,WooCommerce,Magento,Bigcommerce,Reaction%20Commerce.

He, S. 2016. A Reaction Action Recap. Blogikirjoitus Reaction Commercen blogin www-sivuilla. Viitattu 7.4.2018. https://blog.reactioncommerce.com/rsvp-to-our-meetup/.

He, S. 2016. The Future of Open Source Security. Blogikirjoitus Reaction Commercen blogin www-sivuilla. Viitattu 7.4.2018. https://blog.reactioncommerce.com/the-future-of-open-source-security/.

Hicks, S. 2017. Why Reaction is Real-Time. Blogikirjoitus Reaction Commercen blogin www-sivuilla. Viitattu 7.4.2018. https://blog.reactioncommerce.com/why-reaction-is-real-time/.

Hudák, M., Kianičková, E. & Madleňák, R. 2017. The importance of E-mail marketing in E-commerce. University of Zilina, Slovakia. Procedia Engineering, 192, 342–347. Viitattu 15.2.2018. https://janet.finna.fi. EBSCOhost.

Hunt, P. 2013. Why did we build React? Blogkirjoitus Reactin www-sivulla. Viitattu 24.3.2018. https://reactjs.org/blog/2013/06/05/why-react.html.

Hunt, P., O'Shannessy, P., Smith, D. & Coatta, T. 2016. React: Facebook's Functional Turn on Writing JavaScript. Communications of the ACM, 59, 12, 56–62. Viitattu 23.3.2018. https://janet.finna.fi. EBSCOhost.

Installation for OSX. 7.12.2017. Dokumentaation asennussivu Reaction Commercen www-sivuilla. Viitattu 17.2.2018. https://docs.reactioncommerce.com/reaction-docs/master/installation-osx.

Internet Software and Services. N.d. Company Overview of Magento, Inc. Yrityksen yleistiedot Bloombergin www-sivuilla. Viitattu 30.3.2018. https://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=99719241.

Introduction to GraphQL. N.d. Johdanto GraphQL dokumentaatiossa GraphQL www-sivuilla. Viitattu 29.3.2018. http://graphql.org/learn/.

Introduction. N.d. Johdanto Meteor ohjelmointikirjastoon Meteorin www-sivuilla. Viitattu 26.3.2018. https://guide.meteor.com/.

Jenamani, M., Mohapatra, P. K.J & Ghose S. 2006. Design benchmarking, user behaviour analysis and link-structure personalization in commercial web sites. Indian Institute of Technology, Kharagpur. Internet Research, 16, 3, 248–266. Viitattu 11.3.2018. https://janet.finna.fi. ABI/INFORM Collection (ProQuest).

Kwangwon, S & Sukyoung, R. 2017. Analysis of JavaScript Programs: Challenges and Research Trends. ACM Computing Surveys, 50, 4, 1–18. Viitattu 5.3.2018. https://janet.finna.fi. EBSCOhost.

Li, Y., Zhou, G. & Nie, B. 2016. Improving Web Performance in Home Broadband Access Networks. College of William and Mary, USA. Wireless Personal Communications, 92, 3, 925–940. Viitattu 15.2.2018. https://janet.finna.fi. EBSCOhost.

Magento 2. 2018. Magento version 2 GitHub www-sivu. Viitattu 31.3.2018. https://github.com/magento/magento2.

Magento Commerce Investment in Europe Accelerates Growth. 2018. Uutinen Magenton www-sivuilla. Viitattu 2.4.2018. https://magento.com/news-room/press-releases/magento-commerce-investment-europe-accelerates-growth.

Magento Community edition features. N.d. Tuotteen ominaisuuksia Magestoren www-sivuilla. Viitattu 31.3.2018. https://www.magestore.com/magento-introduction/magento-community-edition-features.

Mansén, M. N.d. Mitä on hyvä palvelumuotoilu?. Blogikirjoitus Reaktorin www-sivuilla. Viitattu 1.4.2018. https://www.reaktor.com/blog/mita-on-hyva-palvelumuotoilu/.

Meet the team. N.d. About -sivu Reaction Commercen www-sivuilla. Viitattu 17.2.2018. https://reactioncommerce.com/about.

Meteor Hosting. N.d. Galaxy nimisen palvelun tuotesivu Meteorin www-sivuilla. Viitattu 27.3.2018. https://www.meteor.com/hosting.

Miksi hyödyntäisin palvelumuotoilua?. N.d. Artikkeli Service Design Toolkitin www-sivuillla. Viitattu 1.4.2018. http://sdt.fi/miksi_palvelumuotoilu.html.

MongoDB unleashes the power of software and data for innovators everywhere. N.d. Yrityksen tarina MongoDB www-sivuilla. Viitattu 29.3.2018. https://www.mongodb.com/company.

Nicoletti Dziobczenski, P. & Person, O. 2016. What is sought from graphic designers? A first thematic analysis of job offers for graphic design positions in the United Kingdom. 2016. Proceedings of DRS2016, 1–14. Viitattu 2.4.2018. https://janet.finna.fi. Aaltodoc.

Nielsen, J. & Norman, D.A. 2000. Usability On The Web Isn't A Luxury. Informationweek, 773, 65, 65–73. Viitattu 15.2.2018. https://janet.finna.fi. EBSCOhost.

Niu, N., Brinkkemper, S., Franch, X., Partanen, J. & Savolainen, J. 2018. Requirements Engineering and Continuous Deployment. IEEE Software, 35, 2, 86–90. Viitattu 18.4.2018. https://janet.finna.fi. ProQuest. ABI/INFORM Collection.

Ornbo, G. 2012. Sams Teach Yourself Node.js in 24 Hours. 6–7. Google Books. Viitattu 22.3.2018. https://books.google.fi/books?id=KGt-FxUEj48C&printsec=frontcover&dq=nodejs&hl=fi&sa=X&ved=0ahUKEwi5x-2E9P_ZAhWqJcAKHUtPB4QQ6AEILzAB#v=onepage&q=nodejs&f=false.

Patrício, L., Gustafsson, A. & Fisk, R. 2018. Upframing Service Design and Innovation for Research Impact. Journal of Service Research, 21, 1, 1–12. Viitattu 1.4.2018. https://janet.finna.fi. EBSCOhost Business Source Elite.

React. N.d. Yleiskuvaus React ohjelmointikirjastosta Facebookin Code www-sivulla. Viitattu 24.3.2018. https://code.facebook.com/projects/176988925806765/react/.

Reaction GitHub releases. 2018. Reaction Commercen versiojulkaisut GitHubin www-sivuilla. Viitattu 1.4.2018. https://github.com/reactioncommerce/reaction/releases?after=v0.1.2.

Reaction GitHub. 2018. Reaction Commercen GitHub –sivu GitHubin www-sivuilla. Viitattu 1.4.2018. https://github.com/reactioncommerce/reaction.

Reaction releases. 2018. Julkaistut päivitykset Reaction Commercen GitHub www-sivulla. Viitattu 7.4.2018. https://github.com/reactioncommerce/reaction/tags.

Retail at any scale. N.d. Tuotesivu Reaction Commercen www-sivuilla. Viitattu 17.2.2018. https://reactioncommerce.com/hosting.

Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars). 2018. Verkkokauppojen myynnin tilasto Statistan www-sivuilla. Viitattu 2.4.2018. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/.

Security. 2018. Dokumentaatiosivu Reaction Commercen www-sivuilla. Viitattu 9.5.2018. https://docs.reactioncommerce.com/reaction-docs/master/security.

Shopify repositories. N.d. Shopifyn julkiset ohjelmistoprojektit heidän GitHub käyttäjän alla GitHubin www-sivuilla. Viitattu 31.3.2018. https://github.com/Shopify.

Silva, L.H., Valente, M.T., Bergel, A., Anquetil, N. & Etien, A. 2017. Identifying Classes in Legacy JavaScript Code. Journal of Software: Evolution & Process, 29, 8, 3–5. Viitattu 17.2.2018. https://janet.finna.fi. EBSCOhost.

Srivastava, J. & Cooley, R. 2000. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. University of Minnesota, Minneapolis. SIGKDD Explorations, 1, 2, 12–21. Viitattu 11.3.2018. http://yildiz.edu.tr/~aktas/courses/CE-0114890/g10-p3.pdf.

Storefront API. N.d. Storefront rajapinnan dokumentaation esittely Shopifyn www-sivuilla. Viitattu 31.3.2018. https://help.shopify.com/api/storefront-api.

Sääksjärvi, M. 2016. What Is Design: The Delft Design Approach. Blogikirjoitus Design Forum Finlandin www-sivuilla. Viitattu 2.4.2018. http://www.design-forum.fi/blog/what-is-design-the-delft-design-approach/.

Testing. 2018. Testauksen dokumentaatio Reaction Commercen www-sivuilla. Viitattu 4.5.2018. https://docs.reactioncommerce.com/reaction-docs/master/testing-reaction.

Toikko, T. & Rantanen, T. 2009. Tutkimuksellinen kehittämistoiminta: näkökulmia kehittämisprosessiin, osallistamiseen ja tiedontuotantoon. Tampere University Press.

User Interface (UI) Design. N.d. Johdanto Interaction Design Foundationin www-sivuilla. Viitattu 13.3.2018. https://www.interaction-design.org/literature/topics/ui-design.

User Interface Design Basics. N.d. Kuvaus Usability.govn www-sivuilla. Viitattu 14.3.2018. https://www.usability.gov/what-and-why/user-interface-design.html.

Valentine, T. & Reid, J. 2013. JavaScript Programmer's Reference, 1–3. Viitattu 9.3.2018. Google Kirjat. https://books.google.fi/books?id=edqP-9fFwx8C&lpg=PA3&ots=vNB01Ec4DR&dq=browser%20support%20es%20features%20publication&hl=fi&pg=PA3#v=onepage&q=browser%20support%20es%20features%20publication&f=false

Vaughn, B. 2018. Update on Async Rendering. Blogipäivitys Reactin www-sivuilla. Viitattu 29.3.2018. https://reactjs.org/blog/2018/03/27/update-on-async-rendering.html.

Volusion Secures $55 Million in Financing to Accelerate Growth Plans. 2015. Lehdistöjulkaisu Mozun www-sivuilla. Viitattu 2.4.2018. https://www.mozu.com/press/volusion_secures_55m_financing_expand_mozu/.

Vulnerabilities. 2018. Dokumentaatio haavoittavuuksista Reaction Commercen GitHub www-sivulla. Viitattu 7.4.2018. https://github.com/reactioncommerce/reaction-docs/blob/master/developer/testing/vulnerabilities.md.

We’re unlike any other. Reaction Commerce is open source, easy to use, and 100% customizable. N.d. Verkkokauppa-alustan ominaisuuksia heidän www-sivuilla. Viitattu 2.4.2018. https://reactioncommerce.com/features.

Zorzini, C. 2015. All The Ecommerce Platforms & Shopping Cart Softwares: 120+ List. Artikkeli Ecommerce Platformsin www-sivuilla. Viitattu 30.3.2018. https://ecommerce-platforms.com/articles/shopping-carts-software-ecommerce-platforms.