Opinnäytetyö AMK

Tieto- ja viestintätekniikka

2022

Melinda Backström

VERKKOKAUPPA-ALUSTA VUE.JS- JA NUXT.JS-OHJELMISTOKEHYKSILLÄ

– Oscar eCommerce

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö AMK | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2022 | 21 sivua

Melinda Backström

Verkkokauppa-alusta Vue.js- ja Nuxt.js-ohjelmistokehyksillä

- Oscar eCommerce

Tämän opinnäytetyön tavoitteena oli selvittää Vue.js- ja Nuxt.js-ohjelmistokehyksien sopivuus verkkokauppa-alustan rakentamiseen. Lisäksi tavoitteena oli tutkia, jos ohjelmistokehyksillä rakennetun sovelluksen kehitystä olisi kannattavaa jatkaa mainituilla kehyksillä.

Työssä tutkittiin ohjelmistokehyksien roolia sovelluskehityksessä ja syitä työssä käytettyjen ohjelmistokehyksien valinnalle. Työn rinnalla toteutettiin projekti, jossa mallinnettiin uusi verkkokauppa-alusta käyttämällä valittuja ohjelmistokehyksiä.

Työn tuloksena saatiin kerättyä kattavaa tietoa valittujen ohjelmistokehyksien käytettävyydestä verkkokauppa-alustan sovelluskehitykseen. Työn tulokseksi saatiin, että mainituilla ohjelmistokehyksillä on kannattavaa jatkaa verkkokauppa-alustan rakentamista.

Asiasanat:

ohjelmistokehys, sovelluskehitys, verkkokauppa, WWW, JavaScript

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2022 | 21 pages

Melinda Backström

Ecommerce Platform with Vue.js and Nuxt.js frameworks

- Oscar eCommerce

The aim of this thesis was to research the suitability of Vue.js and Nuxt.js frameworks for building an ecommerce platform.

The thesis covers the roles of frameworks in software development and details the reasons for choosing the forementioned frameworks. A part of the theory used was collected from a project done concurrently with the thesis, in which a smaller model of the ecommerce platform was built using the chosen frameworks.

The result of the thesis was comprehensive information about the chosen frameworks and their usability in building an ecommerce platform. The results prove, that the use of Vue.js and Nuxt.js in building an ecommerce platform is recommended.

Keywords:

Framework, Software Development, Ecommerce, Web, JavaScript

Sisältö

Kuvat

Lyhenteet

CSR	Client-Side Rendering, asiakasohjelmassa tapahtuvan hahmottamisen menetelmä.
CSS	Cascading Style Sheet, tyylikieli, jota käytetään verkkosivujen ulkoasun määrittelyyn.
HTML	Hyper Text Markup Language, merkintäkieli, jonka avulla teksti saadaan verkossa luettavaan muotoon.
HTTP	Hypertext Transfer Protocol, protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
JavaScript	Ohjelmointikieli, joka lisää verkkosivuille dynaamista toiminnallisuutta.
JSON	JavaScript Object Notation, esitysmuotoilu tiedonvälitykseen.
Node.js	Avoimen lähdekoodin alustariippumaton ajoympäristö JavaScript-koodin suorittamiseen palvelimella.
REST	Representational State Transfer, arkkitehtuurityyli, jota usein käytetään rajapintojen toteuttamiseen.
SEO	Search Engine Optimization, hakukoneoptimointi.
SPA	Single-Page Application, selaimessa käytettävä ohjelmisto, jossa kaikki toiminnallisuudet ja tiedot ladataan kerralla selaimeen.
SSR	Server-Side Rendering, palvelimella tapahtuvan hahmottamisen menetelmä.
TypeScript	Ohjelmointikieli, joka perustuu JavaScriptiin lisäten uusia ominaisuuksia.

1 Johdanto

Työn toimeksiantajana toimii suomalainen ohjelmistoyritys Oscar Software Oy. Työssä tutustuttiin yrityksen tarjoamaan verkkokauppaohjelmistoon Oscar eCommerceen, jonka avulla voidaan rakentaa omiin tarpeisiin sopivia verkkokauppoja, koota varastotietoja sekä myynti- ja talousraportteja yhtä sovellusta käyttäen.

Työn rinnalla toteutettiin kahden viikon mittainen projekti, jossa rakennettiin uusi versio Oscar eCommercesta käyttämällä Nuxt- ja Vue-ohjelmistokehyksiä. Työllä keskityttiin havainnoimaan ohjelmistokehyksien roolia sovelluskehityksessä sekä selvittämään, onko projektissa käytettyjen ohjelmistokehyksien käyttö verkkokauppa-alustan rakentamiseen kannattavaa eri näkökulmista tarkasteltaessa.

Luvussa 2 tutustutaan yleisesti ohjelmistokehyksiin sekä tarkemmin niihin, joita projektissa käytettiin. Luvussa keskitytään kertomaan, mikä on ohjelmistokehyksien tausta, mihin niitä käytetään ja miten niitten käyttö voi helpottaa sovelluskehitystä kehittäjän näkökulmasta. Luvussa 3 tutustutaan toimeksiantajayrityksen tarjoamaan verkkokauppa-alustaan ja käydään läpi alkuperäisen alustan toiminnot ja teknologiat. Luvussa 4 käydään läpi projektin aikana rakennettua verkkokauppa-alustaa sekä tarkemmin projektin vaatimukset ja toteutus. Luvussa 5 vertaillaan uuden version toimivuutta vanhaan ja punnitaan uuden version mahdolliset edut ja haitat. Luvussa pohditaan, miten sovelluskehitystä kannattaa jatkaa ja mitä pitää erityisesti ottaa huomioon. Luvussa 6 on yhteenveto, jossa tuodaan esille opinnäytetyön ajankohtaisuus, vaatimukset sekä tulokset.

2 Ohjelmistokehykset

Ohjelmistokehys on runko, jonka on tarkoitus nopeuttaa sovelluskehitystä. Kehys tarjoaa kehittäjälle valmiita osia ja ohjelmistokirjastoja, jonka päälle rakentamalla voidaan helposti koota tarpeidensa mukaisia sovelluskokonaisuuksia. [1.]

Ohjelmistokehyksiä käytetään helpottamaan kehittäjän työtä, ja luomaan valmis pohja, jonka päälle rakentaa. Ohjelmistokehyksen avulla kehittäjä voi kirjoittaa selkeätä ja mukautuvaa koodia, välttää koodin uudelleenkirjoittamista sekä olla varma että ohjelmistokehys ja sen osat ovat testattuja ja ajan tasalla.

Ohjelmistokehyksiä löytyy moneen eri käyttöön ja monella eri ohjelmointikielellä. Tässä työssä keskitytään WWW-ohjelmistokehyksiin, tarkemmin JavaScript-ohjelmistokehyksiin Vue ja Nuxt.

2.1 Vue.js

Vue on kevyt, laajasti käytössä oleva avoimen lähdekoodin JavaScript-ohjelmistokehys. Vue tukee sekä JavaScriptin että TypeScriptin käyttöä. Vue on yksi käytetymmistä ohjelmistokehyksistä ohjelmoijien keskuudessa sekä erittäin helppo ymmärtää että oppia [2] [3].

Vuen kehittäjä Evan You suunnitteli ohjelmistokehyksen vuonna 2014. Youn tarkoituksena oli luoda monipuolinen ohjelmistokehys, jonka pohjalta on helppoa rakentaa erityyppisiä verkkosovelluksia kuten web-komponentteja (engl. Web Components), SPA-sovelluksia (engl. Single-Page Application) sekä lisäkirjastojen avulla myös työpöytä- ja mobiilisovelluksia.

Vuen suosioon on vaikuttanut ohjelmistokehyksen keveys ja nopeus. Vue käyttää Virtual DOM:ia, mikä tarkoittaa, että muutoksien tapahtuessa Vue ei päivitä koko sivua, vaan pelkästään muutetut osat, joka säästää aikaa ja resursseja. Vuen kevyt runko syntyy siitä, että alustavasti ohjelmistokehys on hyvin yksinkertainen ja vaatii ohjelmoijaa asentamaan tarvittavia lisäkirjastoja.

Tämä on kuitenkin Vuen etu, sillä se ei tuo mukanaan kirjastoja, joita ohjelmoija ei välttämättä käytä, jotka vievät turhaa tilaa sovelluksesta. [4.]

Single-File Components

Vue tarjoaa kehittäjälle SFC-tiedostomuodon (engl. Single-File Component), jolla voidaan yhdistää sivun HTML, JavaScript sekä CSS yhteen tiedostoon. Kehittäjä voi kuitenkin valita erottavansa nämä eri tiedostoihin. Kuvassa 1 näkyy esimerkki laskinkomponentista, joka on rakennettu SFC-tiedostomuotoa käyttäen.

```
<template>
    <button @click="count++">Count is: {{ count }}</button>
</template>

<script>
export default {
    data() {
        return {
            count: 0
        }
    }
}
</script>

<style scoped>
button {
    font-weight: bold;
}
</style>
```

Kuva 1. Esimerkki Vuen SFC-tiedostomuodon koodirakenteesta.

SFC-tiedostomuotoa käyttäen kehittäjä voi luoda tiedoston, joka sisältää kokonaisen komponentin. Komponentteja on helppo käyttää uudelleen sovelluksessa, vähentäen koodin uudelleenkirjoittamista ja tehostaen kehitystä.

2.2 Nuxt.js

Nuxt on Vuen päälle rakennettu avoimen lähdekoodin JavaScript-ohjelmistokehys, joka on luotu tekemään Vuella tehdystä ohjelmistokehityksestä nopeampaa ja tehokkaampaa. Nuxt mahdollistaa uusien toimintojen lisäämisen Vue-ohjelmistokehyksen päälle saumattomasti. [5.]

Nuxt tarjoaa Vuelle mahdollisuuden luoda SSR-sovelluksia (engl. Server-Side Rendering), joka tarkoittaa, että taustalla pyörivä palvelin tarjoaa HTML-tiedostot suoraan Vuen komponenteille. Kun HTML-tiedosto tuodaan kokonaisena palvelimelta, saadaan sovellukseen heti sivun metadata ja sivun SEO (engl. Search Engine Optimization), eli hakukoneoptimointi, joka parantaa sivun näkyvyyttä hakukoneissa huomattavasti. Sovellus hyötyy sivujen hahmottamisesta palvelimella myös sillä, että sivujen lataamisaika nopeutuu. [6.]

SSR ja CSR (engl. Client-Side Rendering) ovat kaksi tunnetuimmista ja käytetyimmistä hahmottamistavoista sovelluskehityksessä. Hahmottamistapaa valittaessa kannattaa ottaa huomioon sekä rakennettavan sovelluksen tekniset tarpeet että sovelluksen kohderyhmän tekniset esteet.

Server-Side Rendering

Kuva 2. Prosessi kun hahmottaminen tapahtuu palvelinpuolella.

Kun hahmottaminen tapahtuu palvelinpuolella, verkkosivusto on näkyvissä käyttäjälle ennen JavaScript-koodin latautumista palvelimelta. Verkkosivusto ei ole tässä kohtaa vielä käytettävissä, mutta sivuston kuvat, tekstit ja elementit, kuten napit, taulukot ja valikot, ovat heti näkyvissä käyttäjälle. Verkkosivuston käytettävyys syntyy vasta kun selain on suorittanut JavaScript-koodin, minkä

jälkeen Vue voi lisätä tapahtumakuuntelijat (engl. event listeners), eli yhdistää toiminnallisuudet tarvittaviin elementteihin. Kuvassa 2 nähdään pelkistetty hahmottamisen kulku SSR-menetelmää käyttäen.

SSR-menetelmää suositellaan sivustoille, jotka sisältävät paljon tekstiä, kuvia tai muita HTML-elementtejä. Menetelmä on myös sopiva, jos kohderyhmään kuuluu henkilöitä, joiden selaimet eivät ole ajan tasalla.

Client-Side Rendering

Kuva 3. Prosessi kun hahmottaminen tapahtuu asiakasohjelmassa.

Hahmottamisen tapahtuessa asiakasohjelmassa (engl. client) verkkosivusto ei tule näkyviin käyttäjälle ennen kuin Vue on saanut tarvittavat tiedot palvelimelta, jonka jälkeen sovellus on heti käytettävissä. Kuvassa 3 nähdään pelkistetty hahmottamisen kulku CSR-menetelmää käyttäen.

CSR-menetelmää ehdotetaan käyttämään, jos kohderyhmä osoittautuu käyttävän uusimpia selainversioita. Sillä vaikka sovelluksen ensihahmotus kestää SSR-sovellukseen verrattuna kauemmin, sivujen välillä liikkuminen on nopeampaa, sillä uusia sivustoja ei haeta palvelimelta uudestaan. On kuitenkin huomioitava, että vanhentunut selain ei välttämättä pysty tuomaan ison sovelluksen kaikkia sivustoja kerralla, tai joutuu lataamaan sivustoa kauan, mikä alentaa sivustonkävijän käyttäjäkokemusta. [7.]

Hybridimenetelmä

Hahmottamistavan valinta ei ole aina suoraviivaista. Kun kehityksen alla on sovellus, joka hyötyisi sekä SSR- ja CSR-menetelmästä, voidaan päättää

käyttää molempia menetelmiä sovelluksessa. Tätä kutsutaan hybridimenetelmäksi (engl. hybrid approach).

Hyvä esimerkki sovelluksesta mihin hybridimenetelmää kannattaa käyttää on blogisivusto. Blogin tekstit ja kuvat halutaan nopeasti näkyviin sivuston vierailijoille, ja SSR-menetelmällä tämä onnistuu helpoiten. Blogin omistaja haluaa kuitenkin suojatun tavan kirjautua sisällöntuotannon puolelle sekä uusien blogikirjoituksien luonti lähettää kutsuja palvelimelle, jotka palvelin pitää hyväksyttää kirjoituksen julkaisemiseksi. Tähän osaan sovelluksesta sopii parhaiten CSR-menetelmä.

Tietoturva

Hahmottamistapaa valittaessa kannattaa ottaa huomioon eri hahmottamistapojen tietoturvariskit. CSR- ja SSR-menetelmien eroavaisuuksien myötä sovelluksien kohtaamat hyökkäysuhat poikkeavat toisistaan.

CSR-menetelmä usein vaatii tiedon hakemista suoraan palvelimelta. Palvelimelta tullut vastaus saattaa sisältää yksityistä tai turvattua tietoa. Vaikka tietoa ei hahmoteta verkkosivulle, voi siitä aiheutua säännösten noudattamiseen liittyviä ongelmia, sillä selain vastaanottaa tiedon. SSR-menetelmällä tieto haetaan tietokannasta, eikä sitä lähetetä suoraan selaimelle mikä suojaa tietoturvallisuutta. [8.]

3 Oscar eCommerce -ohjelmisto

Oscar eCommerce on suomalainen verkkokauppaohjelmisto, joka on suunniteltu yhdistämään myynti, varastonhallinta sekä taloudenhallinto yhteen tuotteeseen [9]. Tämä tarkoittaa, että verkkokaupan lisäksi asiakas saa käyttöönsä kaupan varastotiedot, voi tilata uusia tuotteita valmistajilta, vastaanottaa maksuja ja kerätä raportteja sekä tilastoja suoraan ohjelmasta. Verkkokauppaohjelmisto tukee myös integrointia ulkopuoliseen toiminnanohjausjärjestelmään.

Oscar eCommerce tarjoaa eri laitteisiin mukautuvan (engl. responsive), hakukoneoptimoidun ja käytettävyyttä huomioivan verkkokauppa-alustan, jonka päälle on helppoa lisätä haluamiaan tietoja ja tuotteita. Verkkokaupan osiot ovat tehty helposti muokattaviksi, joten värien, valikkojen ja sivujen muotoilu onnistuu vaivatta.

Oscar eCommercen tarkoitus on tarjota verkkokauppa-alusta monille eri asiakkaille, joiden tuotteet saattavat erota toisistaan. Alusta on rakennettu soveltuvan myös yritysmarkkinointiin. Verkkokaupan muokattavuus on siksi erittäin tärkeä ominaisuus. Muokattavuus on toteutettu asetuksilla, joita säätämällä asiakas voi itse valita omaan verkkokauppaansa tarvittavat ominaisuudet. Ulkoasumuutoksien kuten värien, kuvien kokojen ja logojen lisäämisen lisäksi asetuksilla voidaan säätää verkkokaupan osioiden näkyvyyttä.

Oscar eCommercella rakennettu verkkokauppa pystyy tarjoamaan asiakkailleen mahdollisuuden rekisteröityä verkkokauppaan. Asiakas voi kirjautumalla saada tietoja aikaisemmista tilauksistaan sekä lisätä tietoja, joiden avulla ostaminen onnistuu nopeammin tulevaisuudessa. Ostoksien maksamisen turvallisuuden takaamiseksi Oscar eCommerce tukee suomalaisen Paytrail maksupalvelun käyttöä, jolla on Safe Pay-sertifikaatti [10] [11].

4 Next-Gen-projekti

Yksi työn tavoitteista oli selvittää Vue- ja Nuxt-ohjelmistokehyksien toimivuutta verkkokauppa-alustan rakentamisessa. Tässä kappaleessa käydään läpi työn yhteydessä rakennetun projektin suunnitelma, toteutus sekä lopputulos.

Projektissa rakennettu versio nimettiin Next-Gen:iksi. Next-Gen on ulkonäöltään ja toiminnoiltaan suora kopio alkuperäisestä verkkokauppaohjelmistosta, Oscar eCommercesta, mutta pienemmässä mittakaavassa. Tämä tehtiin helpottaakseen versioiden vertailua keskenään.

4.1 Suunnitelma

Next-Gen:in kehittämisen tarkoituksena oli selvittää rajatulla laajuudella eCommerce-alustan selainpuolen (engl. frontend) ja palvelinpuolen (engl. backend) eriyttämisen mahdollisuudet.

Selainpuolen toteutus rajattiin kahteen sivustonäkymään, ostoskoriin sekä asetuksiin, joiden avulla voidaan määritellä valinnaisia tietoja kuten varastosaldon näkymistä tuotesivuilla. Sivustonäkymiksi valittiin tuotenäkymä ja tuoteryhmänäkymä. Rajapinnan tiedot päätettiin siirtyvän JSON-muodossa. Tämä mahdollistaa palvelinpuolen sekä selainpuolen alustariippumattomuuden.

Next-Gen:in kehityksen vaatimukset ovat listattu käyttäjätarinana (engl. user story), jonka avulla verkkokaupan toiminnallisuudet ovat helposti ymmärrettävissä. Käyttäjätarinan perusteella voidaan arvioida sekä selain- että palvelinpuolen tarpeita.

Käyttäjätarina

Käyttäjätarinassa asiakas viittaa sivustolla olevaa käyttäjää. Käyttäjätarinassa kerrotaan mitä asiakas pystyy tekemään sivustolla. Käyttäjätarinan pohjalta saadaan selvitettyä mitä ominaisuuksia sivusto tarvitsee.

- Asiakas voi selata tuotteita
- Asiakas voi selata tuoteryhmiä
- Asiakas voi avata tuotenäkymän
- Asiakas voi lisätä tuotteen ostoskoriin tuoteryhmänäkymästä
- Asiakas voi lisätä tuotteen ostoskoriin tuotenäkymästä
- Asiakas voi poistaa tuotteen ostoskorista
- Asiakas voi muokata tuotteen tilausmäärän ostokorissa
- Asiakas voi tyhjentää ostoskorin

4.2 Toteutus

Projektin toteuttamiseen varattiin kaksi viikkoa ja seitsemän hengen kehittäjäryhmä. Kehittäjäryhmästä neljä työskenteli selainpuolen kanssa ja loput kolme palvelinpuolen parissa. Tämän työn kirjoittaja osallistui projektiin kehittäjänä.

Selainpuolen kehitykseen kuuluu sovelluksen näkymät, ulkoasu sekä reititykset. Ohjelmistokehyksiksi valittiin Vue kehittäjäryhmän osaamisen perusteella sekä Nuxt tuomaan verkkokauppa-alustalle SSR-hahmottamistavan mahdollisuus.

Palvelinpuolen toteutukseen sisältyy tietokanta sekä palvelin. Selainpuoli lähettää pyynnön palvelimelle ohjelmointirajapintaa käyttäen. Palvelin hakee ja palauttaa tiedot tietokannasta selaimen pyynnön perusteella.

4.2.1 Ympäristön alustus

Projektin alustamiseen vaadittiin Node.js sekä Nuxt-ohjelmistokehyksen asentaminen. Koska Nuxt-ohjelmistokehys on rakennettu Vuen päälle, ei Vuen erillistä asentamista tarvittu.

Projektin luonti onnistui vaivatta kattavien dokumentaatioiden avulla. Alustaminen on enimmäkseen automatisoitua, eikä vaatinut kehittäjiltä paljon aikaa.

4.2.2 Rajapinta ja tiedonsiirto

Tiedonsiirtoon käytettiin hybridimenetelmää, eli sekä SSR- ja CSR-hahmottamistapoja. Käyttäjän saapuessa sivulle tai näkymään, haetaan sivuston tiedot SSR-menetelmää käyttäen suoraan palvelimelta. Tämän jälkeen kaikki tapahtuu CSR-menetelmällä, eli tieto haetaan palvelimelta ja sivulle päivittyy vain pyydetty tieto.

Ohjelmointirajapinta toteutettiin REST APIa käyttämällä. API (engl. Application Programming Interface) lähettää pyynnön palvelimen pyynnönvälittimelle (engl. request handler), joka ohjaa kutsun oikeaan päätepisteeseen (engl. endpoint). API palauttaa JSON-muotoisen tiedon, jonka selainpuoli hahmottaa käyttäjälle. REST-arkkitehtuurityyliä käyttämällä voidaan parantaa ohjelmiston suorituskykyä, skaalaavuutta ja luotettavuutta [12].

Pyyntöjen lähetykseen käytettiin HTTP-protokollaa. HTTP-protokollan metodeilla voidaan hakea, lisätä, muokata ja poistaa tietoja tietokannasta [13]. Projektissa käytetty tietokanta sisältää esimerkkejä tuotteista, joita verkkokaupassa saattaisi löytyä.

4.2.3 Käyttöliittymä

Käyttöliittymä rakennettiin SFC-tiedostomuotoa käyttäen, joka yhdistää sivun muotoilun, logiikan ja mallinteen (engl. template). Näin saatiin rakennettua erilaisia komponentteja, joita voitiin käyttää uudelleen sivuston eri näkymissä. Kaikilla sivuilla käytettäviksi komponenteiksi syntyi navigointipalkki, sivupalkki sekä ylätunniste.

Näkymien luonti osoittautui helpoksi, sillä näkymät perustuivat nykyiseen Oscar eCommerce versioon, eli käyttöliittymäsuunnittelua ei tarvittu. Näkymien välillä liikkumiseen käytettiin Nuxtin tarjoamaa reititystä. Reititys oli käyttöliittymän rakentamisessa haastavin. Jotta sovellus pystyy tukemaan monien verkkokauppojen luontia, pitää sivujen ja tuotteiden päätepisteet tuoda tietokannasta, sen sijaan että ne olisivat kovakoodattuina sovellukseen.

4.3 Lopputulos

Projektin lopputuloksena syntyi käyttäjäystävällinen ja mukautuva verkkokauppa-alustan mallinnus. Projekti saatiin toteutettua annetun aikavälin aikana onnistuneesti.

Tuotenäkymässä on tuotteen nimi, esittelyteksti, kuva sekä hinta. Tuoteryhmänäkymässä on ryhmään kuuluvien tuotteiden nimet, kuvat ja hinnat. Molemmissa näkymissä on tuotteen kohdalla ”Lisää ostoskoriin”-nappi ja komponenttien avulla sivurakenne pysyy yhtenäisenä sivusta riippumatta. Tuotteiden lisäys ja poisto ostoskorista onnistuu. Tuotteiden yhteinen hinta näkyy käyttäjälle ostoskorissa.

5 Pohdinta

Työn tarkoituksena oli selvittää uuden Oscar eCommerce version rakentamisen kannattavuutta. Versioiden välillä löytyi selviä eroja, jotka osoittavat uuden version rakentamisen olevan suositeltavaa.

Nykyisessä Oscar eCommerce versiossa palvelin muodostaa valmiiksi koko sivuston tai näkymän. Uuden version hahmottamistapa mahdollistaa verkkosivuston hakukoneoptimoinnin ja nopeuttaa sivustojen välistä liikkumista, joka edistää käyttäjäystävällisyyttä.

Mallinnuksen tietoturvaan ei projektin aikana ehditty syventyä, mutta REST API ja hahmottamiseen käytetty hybridimenetelmä tuovat mukanaan alustavaa suojausta sovellukseen. Kehitystä jatkettaessa pitää ottaa huomioon verkkokauppa-alustan käyttäjien kirjautuminen ja siihen liittyvä autentikointi sekä auktorisointi.

Uusi versio on osoittautunut hyväksi pohjaksi, jonka päälle on suotavaa lähteä lisäämään tarvittavia ominaisuuksia. Kehittäjäryhmälle haastavimpana koettu reititys on saatu toteutettua projektin aikana ja uusien päätepisteiden luonti REST APIlla on koettu mutkattomaksi.

Versioiden vertailun perusteella voin suositella Vue- ja Nuxt-ohjelmistokehyksiä verkkokauppa-alustan rakentamiseen ja kehotan toimeksiantajayritystä jatkamaan Next-Gen-projektin kehitystä.

6 Yhteenveto

Opinnäytetyön tavoitteena oli selvittää Vue- ja Nuxt-ohjelmistokehyksien toimivuus verkkokauppa-alustan sovelluskehityksessä. Työn aikana suoritettiin projekti, jossa näillä ohjelmistokehyksillä tehtiin verkkokauppa-alustan mallinnus. Suoritettua projektia vertailtiin toimeksiantajayrityksen alkuperäiseen verkkokauppaohjelmistoon.

Toimeksiantajayritys koki, että alkuperäisen verkkokauppaohjelmiston uusiminen olisi yritykselle ajankohtaista. Tämän opinnäytetyön tavoitteena oli selvittää mahdollisen muutoksen vaikutusta verkkokauppaohjelmistoon sekä kerätä tietoa, jonka avulla muutoksen kannattavuutta voidaan selvittää.

Työn tuloksena selvisi, että Vue- ja Nuxt-ohjelmistokehyksien käyttö verkkokauppa-alustan uusimiseen on suositeltavaa. Työ erittelee myös ohjelmistokehyksillä rakennetun projektin eroavaisuudet toimeksiantajayrityksen alkuperäiseen ohjelmistoon, minkä pohjalta voi toimeksiantajayritys päättää jatkavansa tai olla jatkamatta uuden verkkokauppa-alustan sovelluskehitystä.

Opinnäytetyö täytti työlle asetetut tavoitteet. Kokonaisuudessaan opinnäytetyö antaa hyvän näkemyksen yritykselle Vue- ja Nuxt-ohjelmistokehyksistä ja siitä, miten niitten käyttö verkkokauppa-alustassa voisi olla heille hyödyksi. Oman tutkimuksen ja pohdinnan jälkeen kannustan yritystä jatkamaan verkkokauppa-alustan kehitystä valituilla ohjelmistokehyksillä.

Opinnäytetyön laatijana olen oppinut paljon Vue- ja Nuxt-ohjelmistokehyksistä sekä niitten käyttötarkoituksesta sovelluskehityksessä. Olen tyytyväinen tulokseen sekä tapaan, miten tulokseen on päädytty. Haluan kiittää toimeksiantajayritystä ja erityisesti henkilöitä, jotka olivat osana opinnäytetyön aikana tehdyssä projektissa.

Lähteet

[1] Ranjan, Ritesh 2021. What is a Framework in Programming & Why You Should Use One. Saatavilla: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/#what-is-a-framework [Luettu 20.3.2022]

[2] Stack Overflow 2021. 2021 Stack Overflow Annual Developer Survey. Saatavilla: https://insights.stackoverflow.com/survey/2021#section-most-loved-dreaded-and-wanted-web-frameworks [Luettu 20.3.2022]

[3] Patel, Jeel 2022. 9 Reasons Why Vue.js Is Best for A Robust Web App Development. Saatavilla: https://www.monocubed.com/blog/why-vuejs-gaining-popularity/ [Luettu 20.3.2022]

[4] Vue N.d. Frequently Asked Questions. Saatavilla: https://vuejs.org/about/faq.html [Luettu 20.3.2022]

[5] Nuxt 2022. What is Nuxt? Saatavilla: https://v3.nuxtjs.org/guide/concepts/introduction/ [Luettu 22.4.2022]

[6] Ibsen, Marius 2021. 3 Ways of Rendering on The Web. Saatavilla: https://medium.com/compendium/3-ways-of-rendering-on-the-web-4363864c859e [Luettu 22.4.2022]

[7] Ram, Prashant 2021. Server Side Rendering (SSR) vs. Client Side Rendering (CSR) vs. Pre-Rendering using Static Site Generators (SSG) and client-side hydration. Saatavilla: https://medium.com/@prashantramnyc/server-side-rendering-ssr-vs-client-side-rendering-csr-vs-pre-rendering-using-static-site-89f2d05182ef [Luettu 22.4.2022]

[8] Adservio 2021. Server Side Rendering Advantages. Saatavilla: https://www.adservio.fr/post/server-side-rendering-advantages [Luettu 22.4.2022]

[9] Oscar eCommerce N.d. Oscar eCommerce. Saatavilla: https://www.oscarecommerce.fi/ [Luettu 2.5.2022]

[10] Paytrail N.d. Yhteensopivat alusta. Saatavilla: https://www.paytrail.com/yhteensopivat-alustat [Luettu 2.5.2022]

[11] Safe Pay N.d. Safe Pay by Qvik. Saatavilla: https://www.safe-pay.fi [Luettu 2.5.2022]

[12] Red Hat 2020. What is a REST API? Saatavilla: https://www.redhat.com/en/topics/api/what-is-a-rest-api [Luettu 19.5.2022]

[13] MDN Web Docs 2022. HTTP request methods. Saatavilla: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods [Luettu 19.5.2022]