OULUN AMMATTIKORKEAKOULU

Sanna Eerola

VERKKOKAUPPA ANGULARILLA

VERKKOKAUPPA ANGULARILLA

Sanna Eerola
Opinnäytetyö
Kevät 2020
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Informaatioteknologia, Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Sanna Eerola
Opinnäytetyön nimi: Verkkokauppa Angularilla
Työn ohjaaja: Pekka Ojala/Teppo Räisänen
Työn valmistumislukukausi ja -vuosi: Kevät 2020 Sivumäärä: 27

Opinnäytetyön tarkoituksena on kehittää opinnäytetyöntekijän asiakkaalle verkkokauppasovellus Angular-sovelluskehystä käyttäen. Pohjana sovellukseen toimii asiakkaan aikaisemmat verkkosivut sekä monet suunnitellut uudet ominaisuudet. Sovelluksen tietokantana toimii Firebase.

Raportissa käydään läpi projektissa käytettävät web-teknologiat ja niiden taustat, verkkokaupan rakentamisen mahdollisuudet ja haasteet sekä verkkokauppasovelluksen kehittämisen vaiheet suunnittelusta lopputulokseen. Työn tavoitteena on kehittää JavaScriptin ja sen ohjelmistokehyksen Angularin osaamista ja syvempää ymmärrystä sekä tukea verkkosivuprojektin etenemistä.

Verkkokauppasovelluksen toteutus onnistui suunnitellusti ja raportin kirjoittaminen samanaikaisesti tuki kehittämistyötä. Myös asiakas oli sovellukseen tyytyväinen ja se ollaan ottamassa käyttöön pian.

Asiasanat: Angular, Firebase, JavaScript, verkkokauppa

ABSTRACT

Oulu University of Applied Sciences
Degree Programme of Information Technology

Author: Sanna Eerola
Title of thesis: Verkkokauppa Angularilla
Supervisor: Pekka Ojala/Teppo Räisänen
Term and year when the thesis was submitted: Spring 2020 Number of pages: 27

The purpose of the thesis is to develop an e-commerce application for customer using the Angular application framework. The application is based on the customer's previous website and many planned new features. The database of application is Firebase.

The report reviews the web technologies what are used in the project, the opportunities and challenges of building an e-commerce, and the steps of developing an e-commerce application from draft to final result. The goal of the work is to improve deeper understanding of JavaScript and Angular, and to support the progress of the website project.

The implementation of the e-commerce application was successful as planned and the writing of the report supported the development work. The customer was also happy with the application and it will be launched soon.

Keywords: Angular, Firebase, JavaScript, webstore, ecommerce

SISÄLLYS

LYHENTEET JA TERMIT

Kryptaus	Tiedon salaaminen, eli sen muuntaminen muotoon, josta vain tiedon vastaanottaja saa palautettua alkuperäisen tiedon.
Node.js	Avoimen lähdekoodin JavaScript-ajoympäristö palvelinpuolelle.
Palvelin	Tietoliikenteen yhteydessä tietokoneessa suoritettava palvelinohjelmisto sekä tällaista ohjelmistoa suorittava tietokone.
Pilvipalvelu	Palvelimien välinen verkosto.
Sovelluskehys	Käytössä olevan sovelluskielen päälle muodostettu valmiiksi rakennettu ohjelman osa helpottamaan kehittäjän työtä.
URL	Verkkosivuston tai tiedoston sijainti internetissä.

1 JOHDANTO

Verkko-ostamisen suosio on kasvanut jo monia vuosia räjähdysmäisesti ja tämän vuoksi myös erilaisia verkkokauppasovelluksia on olemassa valtavasti. Niiden toteuttamiseen on luotu monenlaisia apuvälineitä, joiden tarkoitus on helpottaa ja nopeuttaa sovelluksen kehittämistä. Tällaisia apuvälineitä ovat erilaiset valmiit julkaisujärjestelmät sekä sovelluskehykset.

Suuri osa verkkokaupoista toteutetaan käyttäen julkaisujärjestelmiä, kuten esimerkiksi WordPressiä tai Magentoa. Tällöin verkkokaupan rakentaminen on nopeaa ja yksinkertaista, eikä ohjelmointitaitoja tarvita välttämättä ollenkaan. Kuitenkin jos aikaa on mahdollista käyttää enemmän ja tavoitteena on luoda monimutkaisempi verkkosovellus, on se järkevää ohjelmoida itse. Valmiit sovelluskehykset, eli frameworkit, ovat loistava apukeino ohjelmoijalle. Angular on avoimen lähdekoodin TypeScript-pohjainen ohjelmistokehys, joka on yksi maailman suosituimmista JavaScript-pohjaisten sovelluksien sovelluskehyksiä.

Raportissa esitelty verkkokauppaprojekti on toteutettu Angular-sovelluskehyksellä sekä Firebasen tietokannalla. Raportissa käydään läpi ensin projektissa käytettävät web-teknologiat ja niiden taustat, jonka jälkeen kerrotaan yleisesti verkkokaupan rakentamisesta Angular-sovelluskehyksellä. Viimeinen kappale käsittelee itse verkkokauppaprojektin toteuttamista. Siinä käydään läpi työn vaiheita suunnittelusta toteutukseen, vastaan tulleita haasteita sekä projektin lopputulosta.

2 WEB-TEKNOLOGIAT JA TIETOKANTA

Verkkosovellusprojekti toteutettiin JavaScript-pohjaista suosittua sovelluskehystä Angularia käyttäen. Sovelluksen tietovarastona toimi Googlen omistama Firebasen pilvipalvelun tietokanta.

2.1 JavaScript

JavaScript on dynaaminen oliopohjainen komentosarjakieli, joka tukee useita erilaisia ohjelmointityylejä. Ohjelmointikieli tuli ensimmäisen kerran käyttöön jo vuonna 1996. Viimeisin kielen määrittely on JavaScript 1.8.5, joka pohjautuu EcmaScript-standardiin. (Wikipedia 2020, viitattu 26.4.2020)

Tällä hetkellä maailman kolmanneksi suosituinta web-ohjelmointikieltä käytettiin alun perin vain asiakaspuolella (client-side), mutta nykyään sitä voidaan käyttää myös palvelinpuolen ohjelmointikielenä (Goel 2020, viitattu 26.4.2020).

2.1.1 TypeScript

Vaikka JavaScript otettiin alun perin käyttöön vain asiakaspuolen kielenä, on se Node.js:n kehityksen myötä tullut esiin myös nousevana palvelinpuolen teknologiana. Koodin määrän kasvaessa se kuitenkin on muuttunut sekavammaksi, mikä vaikeuttaa koodin ylläpitämistä ja uudelleenkäyttöä. Lisäksi se, että se ei ole omaksunut olioasennusta, vahvaa tyyppitarkistusta ja käännösajan virheentarkistusten ominaisuuksia, estää JavaScriptiä menestymään täysivaltaisena palvelinpuolen tekniikkana. Tämän vuoksi kehitettiin TypeScript.

TypeScript on voimakkaasti tyypitetty, olio-suuntautunut ohjelmointikieli. Kun TypeScriptiä käytetään JavaScriptin sijasta, vähentää se tyypityksen ansiosta koodissa olevia bugeja.

2.1.2 JavaScript-sovellyskehykset

JavaScript-sovelluskehykset ovat JavaScript-kielellä rakennettuja alustoja, joiden avulla ohjelmistokehittäjät voivat muokata valmiita funktioita omia käyttötarkoituksiaan varten. Ne tekevät JavaScriptin kanssa työskentelystä helpompaa ja sujuvampaa. Erilaisia JavaScript-pohjaisia sovelluskehityksiä on olemassa paljon. Tällä hetkellä suosituimpia niistä ovat muun muassa Angular, Facebookin kehittämä React sekä Vue.js. (Goel 2020, viitattu 26.4.2020)

2.2 Angular

Angular on yksi suosituimmista avoimen lähdekoodin JavaScript-sovellyskehyksistä, jota web-kehittäjät voivat käyttää verkko-, työpöytä- sekä mobiilisovellusten rakentamiseen. Sen luoja ja ylläpitäjä on Google. Ensimmäinen Angularin versio, AngularJS, julkaistiin vuonna 2010. Seuraava versio Angular 2 poikkesi rakenteellisesti hyvin paljon edellisestä. Tämän jälkeen uusia versioita on ilmestynyt useita ja koska Google on nyt sitoutunut päivittämään Angularia kahdesti vuodessa, uusia versioita ilmestyy entistäkin tiheämmin. (Goel 2020, viitattu 26.4.2020)

Angularilla kehitettyjen sovellusten nopeutta lisää niissä käytössä oleva kaksisuuntainen tiedonsiirto, joka toteutetaan ngModel-direktiivillä. Kaksisuuntaisen tiedonsiirron ensisijainen etu on melkein automaattiset haut ja päivitykset tietokantaan. Kun tietovarasto päivittyy, myös käyttöliittymä päivitetään välittömästi. (Goel 2020, viitattu 26.4.2020)

Angular -sovelluskehyksellä rakennettu verkkosovellus on tyypillistä toteuttaa yhden sivun sovelluksena (Single Page Applications, SPA). Tällainen sivusto lataa vain yhden HTML-sivun, mikä päivitetään dynaamisesti käyttäjän ollessa vuorovaikutuksessa verkkosovelluksen kanssa. Yhden sivun sovellukset voivat siis kommunikoida taustapalvelimien kanssa päivittämättä kokonaista verkkosivua tietojen lataamiseksi sovellukseen. Näin ne tarjoavat käyttäjille paremman käyttökokemuksen, koska välilehteä vaihtamalla ei tarvitse aina odottaa sivun uudelleen latautumista. (Goel 2020, viitattu 26.4.2020)

2.3 Firebase

Firebase on Googlen omistama sovellusalusta, jolla voidaan rakentaa erilaisia mobiili- ja websovelluksia. Firebasen palveluun kuuluu useita erilaisia työkaluja ja ominaisuuksia. Näitä ovat muun muassa No SQL -tietovarasto, palvelin sovelluksille sekä kirjautumislogiikka Googlen, Facebookin ja GitHubin tunnuksilla. (Esplin 2016, viitattu 26.4.2020)

Firebase helpottaa ja nopeuttaa ohjelmoijan työtä. Ohjelmoijan ei tarvitse hallita servereitä tai koodata itse ohjelmointirajapintaa, kuten muita vastaavia palveluita käyttäessä. Firebase toimii siis serverinä, ohjelmointirajapintana sekä tietovarastona. (Esplin 2016, viitattu 26.4.2020)

Aikaisemmin yleisesti käytetyt LAMP-mallin (Linux, Apache, MySQL, PHP) palvelinympäristötekniikat toimivat HTTP-yhteyden kautta. Firebase sen sijaan toimii Websocket-yhteydellä, joka mahdollistaa paljon nopeamman kommunikoinnin tietokannan ja sovelluksen välillä. Kaikki data päivittyy Websocketin läpi välittömästi kaikille sovelluksen käyttäjille. Tämä siis mahdollistaa Firebasen tietokantaa käyttävän verkkosovelluksen todella nopean toiminnan. Firebasen tietokanta on erittäin hyvin yhteensopiva Googlen kehittämän Angular-sovelluskehyksen kanssa. (Esplin 2016, viitattu 26.4.2020)

3 VERKKOKAUPPA

Internetin käyttö ihmisten päivittäisessä arjessa on kasvanut jo useamman kymmenen vuoden ajan rajusti. Nettiyhteys on saatavilla helposti älypuhelimilla, vaikka joka hetki. Suomen Tilastokeskuksen 2019 vuonna tehdyn tutkimuksen mukaan jo 79 prosenttia suomalaisista käyttää internettiä useasti päivässä. Koska yhä useampi palvelu on saavutettavissa nopeasti ja helposti verkossa, on myös verkkokauppaostaminen yleistynyt. Tilastokeskuksen tutkimuksen mukaan jopa puolet 16–89-vuotiaista suomalaisista oli ostanut jotain netistä viimeisen 3 kuukauden aikana. (Tilastokeskus 2019, viitattu 21.4.2020)

KUVA 1. Verkkokauppaostaminen on yleistynyt.

Lähes jokaisella kivijalkamyymälällä on olemassa myös verkkokauppa ja tämän lisäksi pelkästään verkossa toimivia kauppoja on suuri määrä. Koska verkkokauppoja on paljon, on myös niiden kehittämiseen rakennettu useita erilaisia työkaluja ja apuvälineitä. On olemassa monia helppokäyttöisiä valmiita julkaisujärjestelmiä verkkokaupan perustamista varten. Osa niistä ovat jopa ilmaisia, toiset hyvinkin kalliita. Suosituimpia verkkokauppa-alustoja maailmanlaajuisesti tällä hetkellä ovat muun muassa Shopify, BigCommerce, Magento sekä WordPress-julkaisujärjestelmän lisäosa WooCommerse (Johnson 2019, viitattu 17.9.2019). Valmiilla julkaisujärjestelmällä verkkokaupan voi kätevästi ja nopeasti perustaa myös ilman minkäänlaisia ohjelmointitaitoja, mutta usein niissä on mahdollisuus monipuolisempiin muokkauksiin koodin editoimisen kautta.

Julkaisujärjestelmä ei kuitenkaan voi avata yhtä monia mahdollisuuksia sovelluskehitykseen, kuin kehittäjän oma ohjelmointi. Verkkokaupan ohjelmointi ilman valmista alustaa vie huomattavasti enemmän aikaa, mutta siinä on myös monia hyötyjä. Verkkokauppasovellus ja sen käyttöön tarkoitettu hallintapaneeli voidaan rakentaa juuri sen tulevalle käyttäjälle sopivaksi ja helppokäyttöiseksi. Käyttäjälle tarpeelliset ominaisuudet voidaan tuoda hyvin selkeästi esille ja kaikki ylimääräinen jättää pois. Kun verkkokauppasovelluksen on yhden kerran onnistuneesti ohjelmoinut, sitä on kätevä käyttää pohjana myös tuleville projekteille. Kun tekniikka taustalla on rakennettu toimivaksi, voi kehittäjä ulkoasua muuttamalla ja pienillä päivityksillä saada siitä nopeasti ja helposti aivan uudenlaisen verkkokauppasovelluksen.

3.1 Verkkokauppa Angularilla

Verkkosovelluksen ohjelmointia helpottaa suuresti erilaiset sovelluskehykset, joista Angular on yksi suosituimmista (Goel 2020, viitattu 25.4.2020). Johtuen siitä, että Angular on kehitetty pääasiassa yhden sivun sovelluksia varten, se ei välttämättä ole ensimmäinen vaihtoehto etenkään suurempien verkkokauppojen alustaksi. Tähän yksi syy oli, että sivujen URL-reititys täytyy rakentaa manuaalisesti. Kun navigaatioista vaihtaa välilehteä, ei sivu päivity itsestään kuten monen sivun sovelluksissa. Verkkokauppasivustoilla URL voi olla pitkäkin polku sisältäen tuotekategoriat, tuotteet, ostoskorin, tilaussivun jne. Kuitenkin kun kyseessä on pienempi verkkokauppa, missä tuotteita ja tuotekategorioita on pieni määrä, oli tämä sovelluskehys hyvinkin sopiva.

Angular-sovelluskehys tarjoaa useita verkkokaupalle hyödyllisiä visuaalisia ominaisuuksia. Angular on kehitetty mobiililähtöisesti, joten sivuista on helppo tehdä responsiiviset, eli kaiken kokoisille näytöille sopivat. Yhden sivun sovelluksena verkkokauppa toimii nopeasti ja sujuvasti.

3.2 Tärkeitä huomioita verkkokauppaa kehittäessä

Verkkokauppaa rakentaessa on hyvä huomioida monia asioita jo suunnitteluvaiheessa. Verkkokaupan käytön täytyy olla helposti ymmärrettävissä erilaisille käyttäjille. Myös tietoturvallisuuteen on todella tärkeää kiinnittää paljon huomiota.

3.2.1 Helppokäyttöisyys ja saavutettavuus

Koska teknologia ja yhteydet ovat kehittyneet todella nopeiksi, verkkosovelluksen käyttäjän kärsivällisyys ei enää riitä pitkien latauksien odotteluun. Verkkosivujen on siksi tärkeää olla nopeasti ymmärrettävät ja helppokäyttöiset niitä ensimmäistäkin kertaa käyttäville henkilöille.

Verkkosivuston ulkoasun täytyy olla erittäin selkeä, eli erilaiset elementit on hyvä muotoilla ja sijoittaa niin, että niiden ulkoasu kertoo niiden merkityksestä ja suhteesta toisiinsa. Otsikkojen, tekstien, navigaation, linkkien, ja painikkeiden tulee olla ulkonäön perusteella helposti tunnistettavissa. Myös elementtien loogiseen sijoitteluun on hyvä kiinnittää huomiota. Verkkosivun Informaation määrä ei myöskään saa olla liian suuri kerralla näytettäessä, sillä se vaikeuttaa silmäilemistä ja kohteiden erottamista toisistaan. (Papunet-verkkopalvelu Kehitysvammaliitto ry 2019, viitattu 17.9.2019)

Mobiililaitteiden käyttämisen jatkuvan kasvamisen aikaan on erittäin tärkeä huomioita, että verkkosovellus on helppokäyttöinen pienemmälläkin näytöllä. Toimeksiantajan aikaisempien verkkosivujen analytiikka kertoo, että viimeisimmän vuoden aikana jo lähes kaksi kolmasosaa (65,01%) sivuston käyttäjistä ovat käyttäneet sivua mobiililaitteella. Todennäköisesti tämä luku tulee tulevaisuudessa yhä kasvamaan.

Verkkokaupan tavoitteena on lähes poikkeuksetta saada asiakas tekemään ostos. Sen vuoksi verkkokauppasovellusta kehittäessä täytyy reitistä ostoksen tekoon tehdä mahdollisimman yksinkertainen. Sivuston rakenteen eli sivujen välillä liikkumisen täytyy olla loogista ja ymmärrettävää (Papunet-verkkopalvelu Kehitysvammaliitto ry 2019, viitattu 17.9.2019). Verkkokauppasovelluksesta on tämän vuoksi hyvä jättää kaikki ylimääräinen pois sekä muistaa huomioida sovelluksen nopea käytettävyys myös mobiililaitteella.

3.2.2 Responsiivisuus

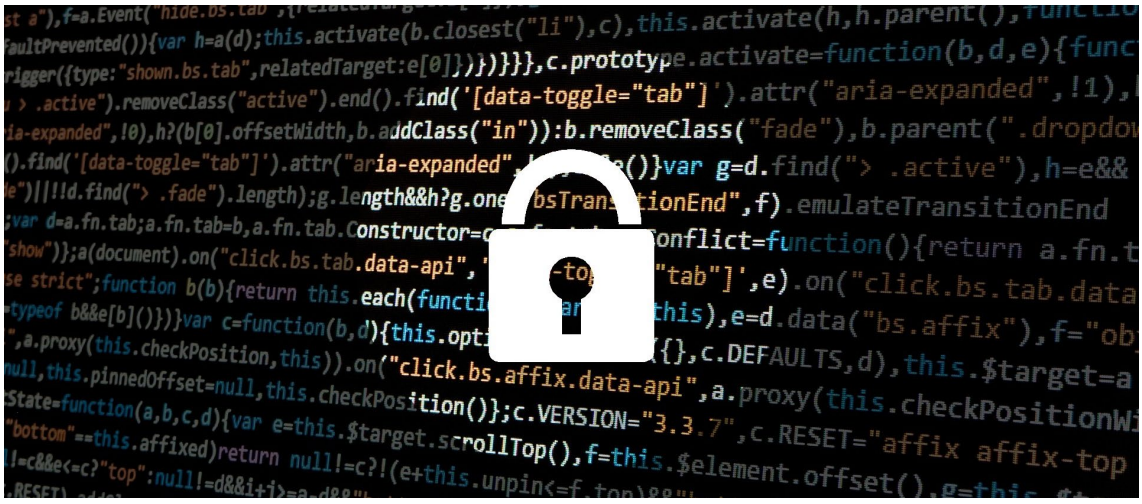
Jo suurin osa netin käytöstä tapahtuu perinteisen tietokoneen sijaan mobiililaitteella, kuten älypuhelimella tai tabletilla. Viime vuonna tehdyn tutkimuksen mukaan jopa 80 prosenttia suomalaisista käyttävät nettiä matkapuhelimellaan (Tilastokeskus 2019, viitattu 25.4.2020).

Tämän vuoksi verkkosivut on tärkeä rakentaa niin, että ne mukautuvat kaikille eri kokoisille näytöille. Tällaista sivujen kehittämistapaa kutsutaan responsiiviseksi tekniikaksi. Koska myös verkkokauppaostamisesta iso osa tapahtuu mobiililaitteella, on ostaminen tärkeää rakentaa yhtä helpoksi niin älypuhelimella kuin perinteisellä isolla näytöllä. (Hyvinvointihack 2019, viitattu 25.4.2020)

3.2.3 Tietoturva

Verkkosovelluksia kehittäessä on aina tärkeää ottaa huomioon niiden tietoturvallisuus. Verkkokauppasovelluksessa käsitellään tilaus- ja henkilötietoja, joten sen tulee olla ehdottomasti turvallinen käyttää. 54 prosenttia maailmanlaajuisista yrityksistä ovat joutuneet viimeisen vuoden aikana kyberhyökkäyksen kohteeksi, joka todistaa sen, ettei riittävään tietoturvaan kiinnitetä aina tarpeeksi huomiota (Lofgren 2020, viitattu 25.4.2020).

Ei ole vain yhtä tapaa, jolla verkkosivustoille hyökätään. Yleisimpiä uhkia ovat spämmi, haittaohjelmat, verkkosivuston palvelimen paikantaminen verkkotunnuksen tietojen avulla sekä DDoS-hyökkäykset, eli verkkosivuston palvelimien ylikuormittaminen huijaus-IP-osoitteilla. Haittaohjelmat (virukset) ovat kaikista suurin uhka verkkosivustoille. Joka päivä luodaan jopa 230 000 uutta haittaohjelmanäytettä. (Lofgren 2020, viitattu 25.4.2020)



KUVA 2. Verkkosovelluksia kehittäessä on aina tärkeää ottaa huomioon niiden tietoturvallisuus.

3.2.4 Hallintapaneeli ja tietokanta

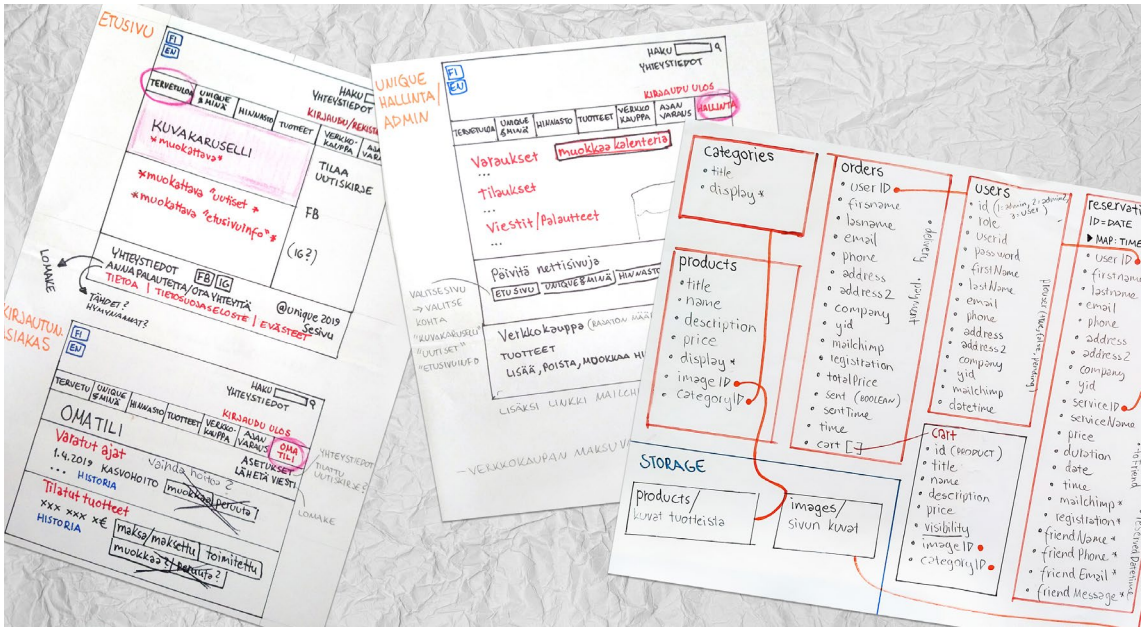
Tärkeä osa toimivaa verkkokauppaa on hyvä hallintapaneeli. Hallintapaneelin kautta verkkokaupan ylläpitäjä voi muokata, poistaa ja lisätä tuotteita sekä tuotekategorioita, nähdä asiakkaiden tilaukset ja mahdollisesti myös muokata niitä tarvittaessa.

Tuotteet ja tilaukset tallennetaan Firebasen tietokantaan. Verkkokauppasovellusta kehittäessä kannattaa aina suunnitella tietokanta huolellisesti. Sen voi mallintaa piirroksena, jotta kokonaisuuden pystyy hahmottamaan hyvin jo ennen sovelluksen rakentamisen aloittamista.

4 PROJEKTIN TOTEUTUS

4.1 Suunnittelu

Verkkokauppasovelluksen rakentaminen aloitettiin paperille piirretyillä suunnitelmilla. Paperilla olevia kuvia on helppo tarvittaessa muokata ja kommentoida, toisin kuin hahmotelmia tai koodattuja malleja tietokoneella. Seuraavaksi suunnitelmat käytiin läpi aloituspalaverilla yhdessä asiakkaan kanssa.



KUVA 3. Verkkokauppasovelluksen ensimmäiset suunnitelmat paperille piirrettyinä.

Asiakkaalla aikaisemmin käytössä ollut verkkosivu oli rakennettu WordPress-jukaisualustalla ja sivujen navigaatiosta oli linkitykset erilliseen verkkokauppaan (myCashFlow) sekä ajanvarauspalveluun (varaa.com). Nyt tavoitteena oli rakentaa verkkosivusovellus, missä kaikki tarvittavat ominaisuudet olisivat sisäänrakennettuina. Verkkokaupan ja ajanvarausjärjestelmän lisäksi sivuille tulisi selkeä ja helppokäyttöinen hallintapaneeli, jossa sivuston ylläpitäjä pääsisi helposti muokkaamaan verkkokaupan tuotteita, ajanvarauksen kalenteria ja verkkosivujen sisältöä. Vanhoissa verkkosivuissa oli edellä mainittujen lisäksi myös paljon pieniä vikoja ja puutteita, mitkä uusien sivujen toteutuksessa huolehdittaisiin myös kuntoon.

Kokonaisuus siis pyrittiin hahmottamaan ja suunnittelemaan jo alussa mahdollisimman hyvin, jotta vältyttäisiin yllättäviltä ongelmilta. Sovelluksen tekijä ei ollut koskaan aikaisemmin toteuttanut näin laajaa sovellusta Angular-sovelluskehyksellä, joten senkin vuoksi erityisen huolellinen suunnittelu oli tarpeen. Lisäksi asiakas toi palaverissa esille muutamia uusia toiveita ja ideoita, mitkä sovellusta kehittäessä olisi tarvittavaa muistaa.

4.2 Asennukset

Oli tärkeää, että sovelluksen toteuttavalle tietokoneelle asennettiin sen hetkiset uusimmat versiot NodeJS:stä sekä Angularista. Näin varmistettiin, että kaikki uusimmat ominaisuudet ovat varmasti käytössä tarvittaessa. Angularista on viime vuosina tullut useita uusia versioita. Edellinen versio ”Angular 6” julkaistiin 4.3.2018 ja 18.10.2018 ”Angular 7”, joka oli uusin versio projektia aloittaessa. (Wikipedia 2019, viitattu 20.11.2019)

Asennusten jälkeen luotiin uusi projekti koodieditorilla (Visual Studio Code) sekä GitHub varasto (repository) projektille. GitHub on koodin ylläpitoalusta versionhallintaa ja yhteistä työskentelyä varten. Se mahdollistaa työskentelyn projektissa mistä tahansa ja sen avulla on aina mahdollista palata aikaisempaan versioon esimerkiksi virheen sattuessa (GitHub 2016, viitattu 31.8.2019). Vaikka projektilla oli vain yksi tekijä, oli tärkeää turvata koodi pilveen. Näin sovellus ei ollut täysin riippuvainen kehityksestä käytettävästä tietokoneesta, virheet pystyttiin helposti korjaamaan ja projektin etenemistä oli kätevä seurata.

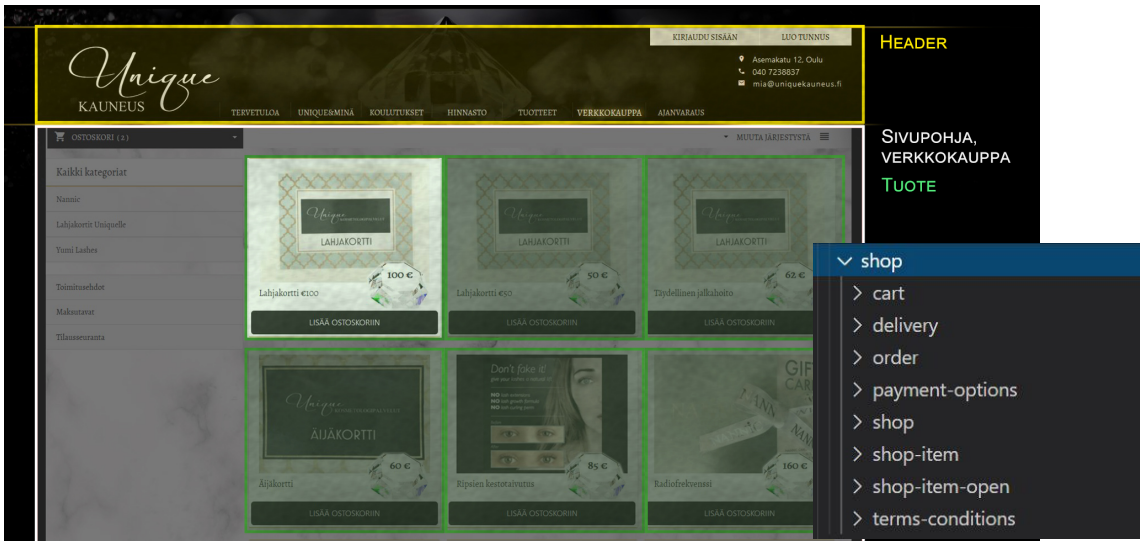
Sovellukseen asennettiin jo alkuvaiheessa muutama lisäosa: Bootstrap, Angular Material sekä jQuery helpottamaan tyylikkään ulkoasun tekemistä sekä tarjoamaan muita hyödyllisiä ominaisuuksia. Bootstrap on yksi maailman suosituimmista sovelluskehyksistä, jonka avulla pystytään kehittämään responsiivisia projekteja HTML-, CSS- ja JavaScript-sovelluksissa valmiita pohjia hyödyntäen (Staff 2019, viitattu 21.4.2020). Angular Material (material design components for angular) on Angular-sovelluksille rakennettu lisäosa, joka tarjoaa laajan valikoiman valmiita komponentteja, jotka helpottavat sovelluksen jäsenneltyä muotoilua (Raju 2020, viitattu 21.4.2020). Verkkokauppasovelluksen tekijä uskoi tarvitsevansa sovelluksen kehityksessä myös jQuerya, joka on nopea, pieni ja monipuolinen JavaScript-kirjasto (The jQuery Foundation 2020, viitattu 21.4.2020).

4.3 Sivupohja

Angularille tyypilliseen tapaan myös opinnäytetyössä kehitettävä verkkosivusovellus rakennettiin yhden sivun sovelluksena (Hackr.io 2019, viitattu 31.8.2019). Verkkosivuston kaikille sivuille oli käytössä sama sivupohja, johon tulostettiin sisältö Firebasen tietokannasta. Tämä siis mahdollisti sen, että sivustoa ei tarvinnut aina ladata uudelleen välilehteä vaihtaessa vaan ainoastaan sisältö vaihtui. Esimerkiksi sivuston käyttäjän Internet-yhteyden katketessa voi hän silti selata verkkosivustoa vaihdellen välilehtiä navigoinnista. Sivujen sisällön tulostaminen tietokannasta käsin mahdollisti myös sen, että ylläpitäjä pystyisi päivittämään sisältöä kätevästi hallintapaneelin kautta.

Sivun yläosa (header) ja alaosa (footer) rakennettiin omista komponenteistaan. Tässä vaiheessa muokattiin myös sovelluksen ulkoasua asiakkaan kanssa tehdyn suunnitelman mukaisesti hyödyntäen Bootstrapin elementtejä. Sivuston responsiivinen navigaatio onnistui myös nopeasti Bootstrapin valmiilla pohjalla. Ulkoasun viimeistely tapahtuisi vasta myöhemmin, mutta oli tärkeää, että sivuilla oli jo tässä vaiheessa lopullista ulkonäköä.

Kun sivuston pohja oli valmiina, oli edessä verkkokaupan rakenteen toteuttaminen. Verkkokaupan ulkoasu saatiin tehtyä Bootstrapin verkkokauppa-pohjalla, jota sitten muokattiin sopivaksi. Verkkokauppa koostui kahdeksasta erillisestä komponentista: Verkkokaupan sivupohja, tuoteruutu, yksittäinen tuote avattuna, tilaussivu, toimitusehdot, maksutavat sekä tilausseuranta. Komponentit olivat näkyvissä aina tarvittaessa.



KUVA 4. Verkkokauppasovelluksen koostuminen päällekkäisistä komponenteista.

4.4 Tuotteiden tulostus tietokannasta

Verkkokaupan rakentamisvaiheessa esimerkkituotteet tulostettiin taulukosta (array). Kun ulkoasu ja verkkokaupan muu koodi oli valmiina, oli aika tuoda data oikeasta tietokannasta.

Firebaseen pystyi kirjautumaan jo olemassa olevalla Googlen tilillä, eikä uutta rekisteröitymistä siis tarvinnut erikseen tehdä. Firebaseen luotiin uusi projekti verkkosivuja varten. Tämän jälkeen tietokantaan lisättiin uudet kokoelmat (collection) tuotteille sekä tuotekategorioille sekä muutama valmis tuote testaamista varten.

Ohjeita Firebasen liittämisestä Angular-sovellukseen löytyi netistä runsaasti, mikä johtuu todennäköisesti siitä, että nämä molemmat ovat Googlen omistamia alustoja. Firebasen liittäminen verkkosovellukseen sujui ongelmitta. Seuraavaksi sovelluksen koodiin luotiin tuotteille oma service-elementti. Sinne tehtiin funktio, jossa tuotteet haetaan Firebasen tietokannasta (KUVA 5). Sitä hyödyntämällä tuotteet saatiin listattua muuttujaan tuotekomponentissa (KUVA 6). Kun tuotteet olivat haettuina products -muuttujaan, pystyttiin tarvittavat tiedot tuotteista tulostamaan tuotekomponentin html-tiedostossa (KUVA 7).

```
getProducts() {
  return new Promise<any[]>((resolve) => {
    this.firestore.collection('products').snapshotChanges()
    .subscribe(snapshots => {
      resolve(snapshots)
      this.products = snapshots;
    })
  })
}
```

KUVA 5. Funktio, jolla verkkokaupan tuotteet haetaan tietokannasta.

```
getProData() {
  this.productService.getProducts()
  .then(result => {
    this.products = result;
    this.getCartData();
  });
}
```

KUVA 6. Funktio tuotteiden komponentissa, millä haettiin servicestä tuotteet products -muuttujaan.

```
<div [@productToCart]=currentState  class="card h-100 pointer">

  <div class="price-bg aleg" (click)="openProduct(product?.payload.doc.data().title)">
    <b>{{product?.payload.doc.data().price}} €</b>
  </div>
    <div class="img-div pointer">
      <img (click)="openProduct(product?.payload.doc.data().title)" class="card-img-top"
           src={{imageURL}} alt="product?.payload.doc.data().title">
    </div>

    <h4 class="card-title pointer left aleg" (click)="openProduct(product?.payload.doc.data().title)">
      {{product?.payload.doc.data().name}}
      <small>{{product?.payload.doc.data().name2}}
      <span *ngIf="!product.payload.doc.data().name2">.</span></small>
    </h4>

    <button disabled *ngIf=isAdded type="button" class="btn btn-light buy-btn added-btn"
    (click)="addToCart(product?.payload.doc.data().name)">
      Lisätty!
    </button>
    <button *ngIf=!isAdded type="button" class="btn btn-light buy-btn"
    (click)="addToCart(product?.payload.doc.data().name)">
      LISÄÄ OSTOSKORIIN
    </button>

</div>
```

KUVA 7. Tuotteiden tietojen tulostus products -muuttujasta html tiedostossa.

4.5 Ostoskori ja evästeet

Evästeet (cookies) ovat pieniä käyttäjälle näkymättömiä tiedostoja, joita käytetään lähes jokaisella verkkosivustolla useaan eri tarkoitukseen. Evästeisiin voidaan tallentaa tietoa sivuston käyttäjän käyttäytymisestä, jolloin sivut voidaan räätälöidä sopivalla tavalla tai toteuttaa käyttäjälle parhaiten soveltuvaa markkinointia. Jotkut evästeet lähettävät tietoa myös kolmannelle osapuolelle, eli jollekin toiselle sivustolle kuin missä käyttäjä paraikaa on. Osa evästeistä on kuitenkin ehdottoman välttämättömiä sivuston olennaisten toimintojen kannalta ja ilman niitä sivustoa ei voi käyttää ollenkaan. Välttämättömiä evästeitä voidaan kutsua myös toiminnallisiksi tai automaattisiksi evästeiksi. (Solla 2020, viitattu 21.4.2020)

Verkkokauppasovellus ei toimi ilman evästeitä, koska käyttäjän valitsemat tuotteet tallentuvat evästeisiin, kun ne lisätään ostoskoriin. Angular-sovelluksille löytyi erittäin hvvä valmis kirjasto evästeiden hyödyntämiseen (ngx-cookie-service), jota oli helppo käyttää verkkokauppasovelluksessa. Kirjaston asennuksen jälkeen sovellukseen lisättiin funktiot, joilla valitut tuotteet tallennettiin evästeisiin ja haettiin sitten näkyviin ostoskorissa. Kun käyttäjä valitsee tuotteen, näkyy se välittömästi ostoskorissa ja on sivustolle tallennettuna evästeenä niin kauan kuin on tarpeen. Tuotteen voi myös

poistaa ostoskorista tai muuttaa sen kappalemäärää. Kun käyttäjä on tehnyt tilauksen, evästeet poistuvat automaattisesti ja ostoskori tyhjenee.

4.6 Tilaukset ja maksaminen

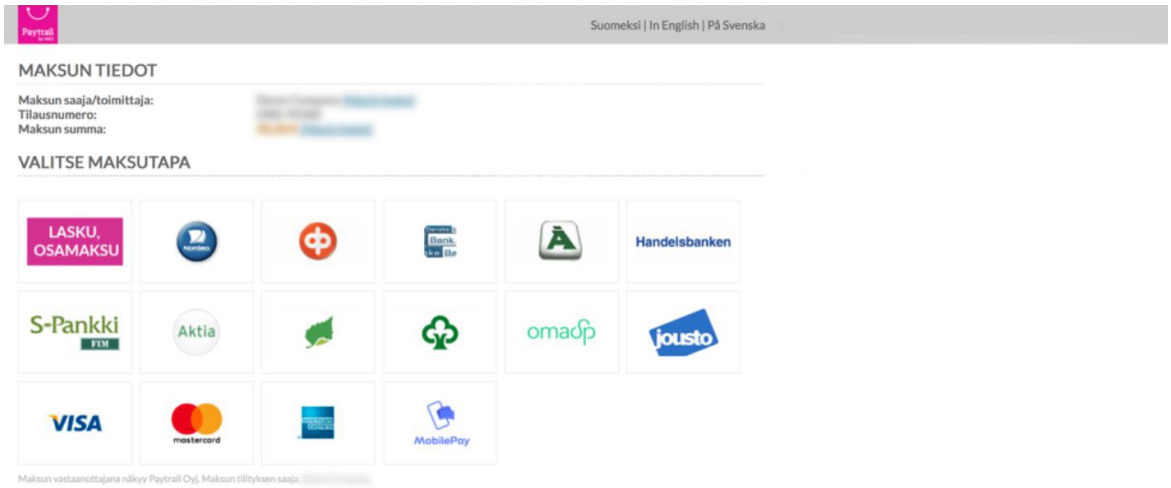
Kun verkkosovelluksen käyttäjä on lisännyt haluamansa tuotteet ostoskoriin, voi siitä klikata joko ”Muokkaa ostoskoria” tai ”Osta tuotteet”. Ostoskorin muokkausta klikatessa aukeaa ostoskori koko sivulle. Valitut tuotteet ja niiden tiedot ovat nähtävissä taulukossa ja niiden kappalemäärää voi myös muokata. Taulukon alimmalla rivillä näkyy tilauksen loppusumma ja ”Siirry kassalle” -nappia klikkaamalla pääsee tilaussivulle.

Tilaussivulla on ensimmäisenä lomake tilaajan yhteystiedoille. Tähän oli tarkoitus tehdä myöhemmin automaattinen tietojen täyttö sisään kirjautuneille käyttäjille. Myös tilauksen tekemisen yhteydessä tulisi voida kirjautua sisään, jolloin lomakkeen tiedot täyttyvät heti automaattisesti. Lomakkeen täytön yhteydessä kentät tarkistetaan (validoidaan) eli katsotaan, että tiedot on täytetty oikein. Mikäli jokin arvoista puuttuu tai ei ole kelvollinen, siitä tulee näkyviin virheilmoitus.

KUVA 8. Lomakkeen virheilmoitukset.

Yhteystietolomakkeen jälkeen käyttäjä voi valita haluamansa toimitustavan ja maksutavan tilaksellensa. Jos toimitustavasta tulee lisäkuluja, ne lisätään tilauksen loppusummaan. Lopuksi on nähtävillä yhteenveto tilauksesta ja asiakkaan täytyy vielä hyväksyä tilaus- ja toimitusehdot sekä todistaa ettei ole robotti Googlen tarjoaman reCAPTCHA-palvelun avulla. Tämä on ilmainen palvelu, mikä suojaa verkkosovellusta roskapostilta ja väärinkäytöltä (Liu 2018, viitattu 21.4.2020). ReCAPTCHA oli helppo asentaa sovellukseen Googlen tarjoaman ohjeistuksen avulla.

Tilauksen maksamista varten löytyi monia erilaisia palveluita. Perehtymisen ja vertailujen jälkeen projektin verkkokauppasovelluksen maksaminen päätettiin toteuttaa Paytrail-palvelulla. Kun tilauksen lomake on täytettynä ja pyydetyt valinnat tehty, siirrytään sivulta automaattisesti Paytrailin tarjoamalle maksusivulle, mikäli maksutavaksi on valittu ”Verkkopankki, luottokortti tai erämaksu” (KUVA 9).

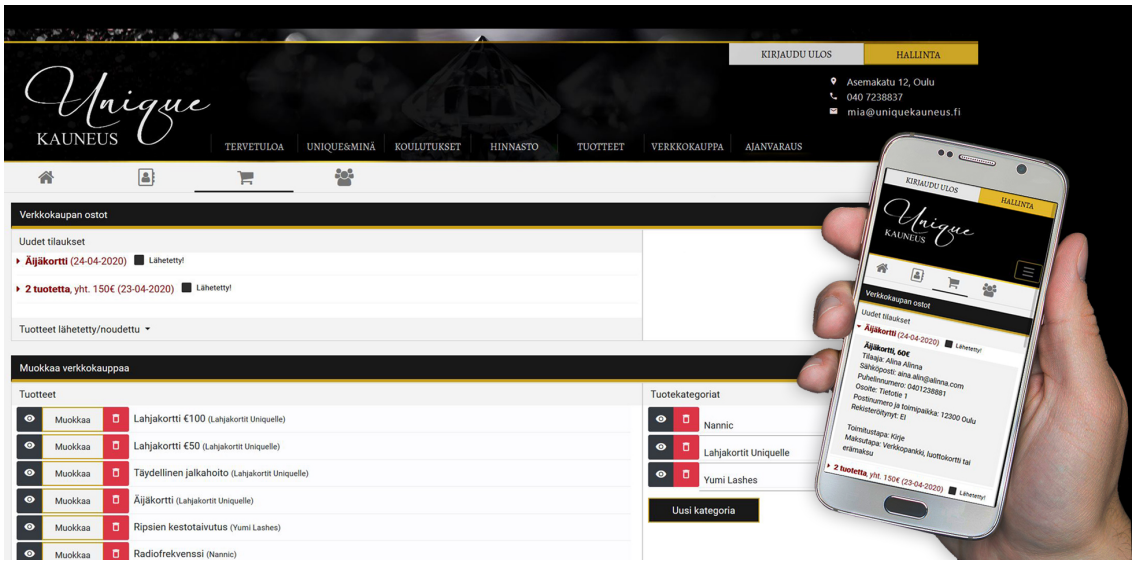


KUVA 9. Paytrail maksusivu.

Paytrail tarjoaa myös mahdollisuuden ohittaa maksusivu toteuttamalla maksutavan valinta suoraan verkkopalvelun puolella. Tulevaisuudessa on mahdollisesti tarkoitus integroida tämä maksusivu suoraan verkkokauppasovellukseen.

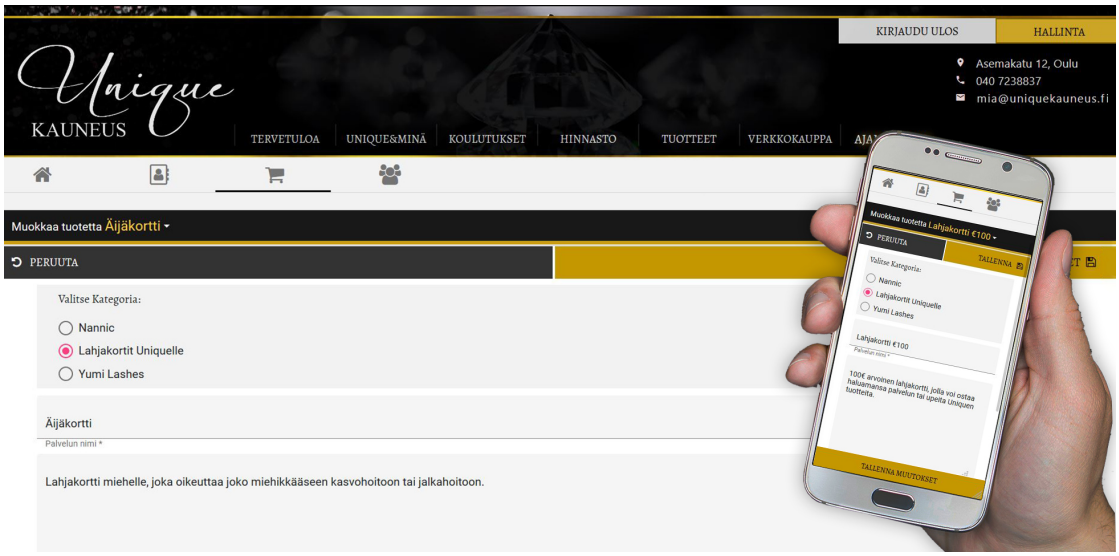
4.7 Verkkokaupan hallinta

Verkkokauppasovelluksessa sivuston ylläpitäjän täytyy voida kätevästi lisätä, muokata ja poistaa tuotteita ja tuotekategorioita sekä hallinnoida asiakkaiden tekemiä tilauksia. Tätä tarkoitusta varten sovellukselle luotiin hallintapaneeli-sivu, mihin oli myöhemmin tarkoitus rakentaa sisäänkirjautumisjärjestelmä. Hallintapaneeliin tehtiin neljä välilehteä, joista yksi oli tarkoitettu verkkokaupan hallinnoimiseen. Hallintapaneeli pyrittiin pitämään mahdollisimman yksinkertaisena ja oli tärkeää, että sitä voidaan helposti käyttää myös mobiililaitteella.



KUVA 10. Hallintapaneelin verkkokauppavälilehti.

Tuotteet saatiin haettua jo aiemmin rakennettua tuote-servicen funktiota hyödyntäen, mutta nyt tarvittiin myös uudet funktiot tuotteiden lisäämisen, muokkaamiseen ja poistamiseen. Tuotteet listattiin allekkain hallintapaneelin verkkokauppa-välilehdellä. Listauksessa näkyy tuotteen nimi, hinta, tuotekategoria sekä kolme nappia tuotteen näkyvyyttä, muokkaamista ja poistamista varten. Tuotteiden näkyvyyden muokkausmahdollisuus tehtiin asiakkaan toiveesta siksi, että ylläpitäjä voi luoda jotain tuotteita jo etukäteen sivustolle odottamaan julkaisua. Muokkaus-nappia klikatessa, kyseessä oleva tuote avautuu uudessa komponentissa, jossa sitä voi muokata lomakkeella. Samanlainen, mutta tyhjä lomake avautuu, kun verkkokauppavälilehden tuotetaulukon alta klikataan ”Uusi tuote”. Tuotelistauksen viereen listattiin tuotekategoriat samalla tyylillä. Niiden nimiä kuitenkin pystyi muokkaamaan jo suoraan taulukosta hallintapaneelin verkkokauppavälilehden etusivulla.



KUVA 11 Verkkokaupan tuotteiden muokkaus lomakkeella.

Nyt tuotteiden ja verkkokaupan tilausten hallinta oli mahdollista hallintapaneelin kautta, mutta se oli näkyvissä kaikille sivuston käyttäjille. Oli aika rakentaa sisäänkirjautumisjärjestelmä sivuston ylläpitäjälle. Tätä varten luotiin uusi komponentti sisäänkirjautumisen sivuksi sekä kaksi uutta servicea (user- ja authentication-service). Tarkoituksena oli rakentaa jo tässä vaiheessa pohjaa myöhemmin tehtäville asiakastileille ja niiden luomiseen. Tietokantaan lisättiin uusi kokoelma käyttäjille ja niiden tiedot haettiin tietokannasta user-servicessä olevalla functiolla (KUVA 12). Sisäänkirjautumisessa tarkastettiin, löytyyhän syötetty käyttäjätunnus ja (kryptattu) salasana tietokannasta. Seuraavaksi määriteltiin sivuston reititykseen ”hallinta”-polun oikeudet ainoastaan ylläpitäjäkäyttäjälle (KUVA 13). Näin ollen hallintapaneelin sivua ei enää voinut muokata tai nähdä ilman sisäänkirjautumista.

```
// Haetaan käyttäjät tietokannasta:
getUsers() {
    new Promise(() => {
        this.firestore.collection('users').snapshotChanges()
        .subscribe(snapshots => {
          this.num = snapshots.length;
            for (let i= 0; i <= (this.num-1); i++) {
                new Promise((resolve) => {
                    this.firestore.collection('users').snapshotChanges()
                    .subscribe(snapshots => {
                    resolve(snapshots[i].payload.doc.data());
                    })
                }).then(data => this.result[i] = (data));
            }
        })
    });
    return this.result;
}
```

KUVA 12. Käyttäjien tietojen hakeminen Firebasen tietokannasta

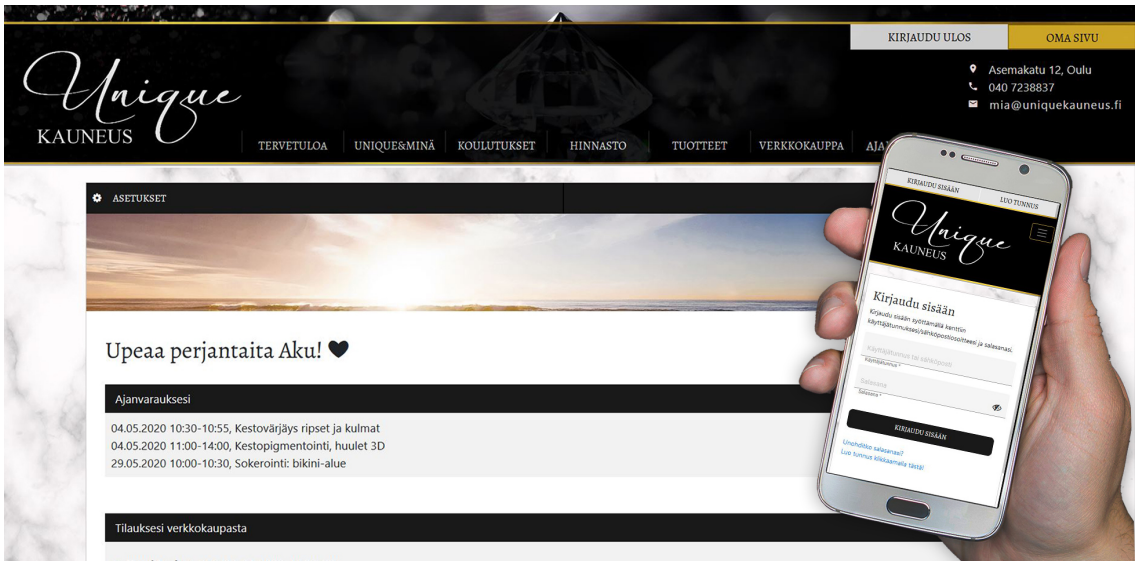
```
{
    path: 'hallinta',
    component: PageComponent,
    canActivate: [AuthGuard],
    data: { page: 'admin', roles: [Role.Admin]}
},
```

KUVA 13. Sivuston hallintapaneelin määritteleminen ainoastaan ylläpitäjäkäyttäjälle.

4.8 Asiakaskäyttäjätilien luonti ja kirjautuminen

Sivuston käyttäjille oli suunniteltu mahdollisuus luoda käyttäjätili, jolla voitaisiin nopeuttaa tilausten tekemistä hyödyntäen tilauslomakkeen automaattista täyttöä, helpottaa viestittelyä ylläpitäjän kanssa sekä mahdollistaa tehtyjen tilausten seuranta. Kirjautumisjärjestelmä oli rakennettuna jo valmiiksi ylläpitäjän käyttäjätilin tekemisen yhteydessä. Samaan tapaan kuin ylläpitäjälle, tietyt oikeudet verkkosovellukseen määriteltiin myös asiakaskäyttäjille. Asiakastilin käyttäjille ei siis tietenkään anneta oikeutta nähdä tai käyttää hallintapaneelia, mutta sen sijaan heille on näkyvissä oma sivu.

Omalla sivulla asiakaskäyttäjät näkevät sovelluksessa tekemänsä ostokset kaikkine tietoineen, sekä merkinnän, milloin tilaus on lähetetty tai noudettu liikkeestä. Omalla sivulla pystyy myös muokkaamaan tilin asetuksia ja lähettämään viestin kätevästi suoraan ylläpitäjälle.



KUVA 14. Asiakkaan käyttäjätilin oma sivu.

Seuraavaksi rakennettiin asiakaskäyttäjän rekisteröintisivu. Käyttäjän täytettyä tarvittavat tiedot lomakkeeseen tarkistetaan, löytyykö saman nimistä käyttäjätunnusta tai sähköpostia jo tietokannassa olevista käyttäjistä. Mikäli näin on, siitä ilmoitetaan lomakkeessa virheilmoituksena (KUVA 15). Kun kaikki tiedot lomakkeeseen on onnistuneesti täytetty, ne tallentuvat välittömästi Firebasen tietokantaan ja käyttäjä saa myös sähköpostivahvistuksen ilmoittamaansa sähköpostiosoitteiseen. Uusi käyttäjätili on heti käytettävissä.

KUVA 15. Tunnuksen luonnin lomakkeen virheilmoitus.

4.9 Haasteet ja lopputulos

Vaikka suunnitelmat projektia varten pyrittiin tekemään huolellisesti ja mahdollisimman yksityiskohtaisesti, moni asia muokkaantui verkkosovelluksen toteutuksen edetessä. Tietyt asiat olivatkin ajateltua monimutkaisempia ja veivät runsaasti aikaa, mutta mitään isompaa ongelmaa ei onneksi ilmaantunut. Angularilla rakennettaviin sovelluksiin löytyy paljon erilaisia ohjeita ja tutoriaaleja netistä, mutta koska Angularista on tullut menneinä vuosina useita eri versioita, osa ohjeista oli suunnattu vanhemmille versioille. Ohjeita sovelluksen eri osa-alueisiin joutui siis paljon soveltamaan ja yhdistelemään.

Verkkosovelluksen lopputulos onnistui hyvin, vaikka toteutuksen aikataulu venyi alkuperäisestää suunnitelmasta. Kaikki sovitut ominaisuudet saatiin onnistuneesti toteutettua.

5 POHDINTA

Yllätyksenä tuli, miten paljon enemmän aikaa verkkosovellusprojekti vei verrattuna alkuperäiseen arvioon. Varsinaisia isoja ongelmia ei missään vaiheessa onneksi ollut, mutta tietyt asiat vaativat vain kaikkine yksityiskohtineen reilusti aikaa. Tuntui järkevältä kirjoittaa opinnäytetyön projektiosiota samaan aikaan kuin itse verkkosivuprojekti eteni.

Projektin edetessä suunnittelun tärkeys korostui entisestään. Vaikka ennen projektin toteuttamisen aloitusta tehtiin huolelliset suunnitelmat, niitä jouduttiin muokkaamaan ja kehittämään useaan kertaan. Myös koodin kommentointi sekä muuttujien ja funktioiden selkeä nimeäminen osoittautui erittäin tärkeäksi, kun sovelluksen tiedostomäärät ja koodirivit kasvoivat.

Haastavimmiksi osioiksi verkkokauppasovelluksen kehittämisessä osoittautuivat ostoskorin saaminen toimimaan sujuvasti evästeitä hyödyntäen sekä useamman samanlaisen tuotteen lisääminen tilaukseen. Sisäänkirjautumis- ja rekisteröintijärjestelmän rakentaminen sen sijaan onnistui huomattavasti suunniteltua nopeammin, sillä niihin löytyi todella hyviä ohjeita netistä.

Opinnäytetyön teoriaosuuteen paneutuminen lisäsi ymmärrystä projektissa käytetyistä web-teknologioista. Opinnäytetyön tekijä sai paljon uutta tietoa Firebasesta ja sen monipuolisista käyttömahdollisuuksista, vaikkakin itse verkkosovelluksessa hyödynnettiin Firebaselta ainoastaan tietokantaa ja tietovarastoa tuotekuville. Angular oli myös suhteellisen uusi tuttavuus opinnäytetyön tekijälle, koska hän oli käyttänyt sitä aikaisemmin ainoastaan harjoitustöissä. Angularin teoriaan paneutuminen tuki verkkosovelluksen kehittämistä kyseisellä kielellä ja antoi entistä laajemman kokonaiskuvan sovelluskehyksestä.

Työ oli kokonaisuudessaan monellakin tapaa opettava. Näitä oppeja opinnäytetyön tekijä toivottavasti pääsee tulevaisuudessa paljon hyödyntämään ja kasvattamaan tietämystä entisestään.

LÄHTEET

Esplin, C. 2016. What is Firebase? Viitattu 26.4.2020, https://howtofirebase.com/what-is-firebase-fcb8614ba442

GitHub. 2016. Hello World. Viitattu 31.8.2019, https://guides.github.com/activities/hello-world/

Goel, A. 2020. 10 Best JavaScript Frameworks to Use in 2020. Viitattu 26.4.2020, https://hackr.io/blog/best-javascript-frameworks

Goel, A. 2020. Why should you learn Angular?? Viitattu 31.8.2019, https://hackr.io/blog/why-should-you-learn-angular

Hyvinvointihack 2019. Responsiivinen nettisivujen suunnittelu. Viitattu 25.4.2020, http://hyvinvointihack.com/2019/05/26/responsiivinen-nettisivujen-suunnittelu/

Johnson, T. 2019. The Top 10 Absolute Best Ecommerce Platforms in 2019. Viitattu 17.9.2019, https://cpcstrategy.com/blog/2019/02/ecommerce-platforms/

Liu, W. 2018. Introducing reCAPTCHA v3: the new way to stop bots. Viitattu 21.4.2020, https://webmasters.googleblog.com/2018/10/introducing-recaptcha-v3-new-way-to.html

Lofgren, L. 2020. Website Security Guide. Viitattu 25.4.2020, https://www.quicksprout.com/website-security/

Papunet-verkkopalvelu Kehitysvammaliitto ry. 2019. Helppokäyttöiset verkkosivut. Viitattu 17.9.2019, https://papunet.net/saavutettavuus/helppokayttoiset-verkkosivut

Raju, S. 2020. What is Angular Material and How to Implement it? Viitattu 21.4.2020, https://www.edureka.co/blog/what-is-angular-material/

Solla, K. 2020. Digitreenit: Mitä nettisivujen evästeet oikein tekevät? Onko ne pakko hyväksyä? Viitattu 21.4.2020, https://yle.fi/aihe/artikkeli/2020/02/22/digitreenit-mita-nettisivujen-evasteet-oikein-tekevat-onko-ne-pakko-hyvaksya#siivoan

Staff, A. 2019. What is Bootstrap? An In-depth Guide of the Framework. Viitattu 21.4.2020, https://wpamelia.com/what-is-bootstrap/

The jQuery Foundation. 2020. What is jQuery? Viitattu 21.4.2020, https://jquery.com/

Tilastokeskus. 2019. Puolet suomalaisista ostanut verkkokaupasta viimeisen kolmen kuukauden aikana. Viitattu 21.4.2020, http://www.stat.fi/til/sutivi/2019/sutivi_2019_2019-11-07_tie_001_fi.html

Wikipedia. 2019. Angular (web framework). Viitattu 20.11.2019, https://en.wikipedia.org/wiki/Angular_(web_framework)

Wikipedia. 2020. JavaScript. Viitattu 26.4.2020, https://fi.wikipedia.org/wiki/JavaScript