Webkehitys Reactilla

Minka Laasonen

2021 Laurea

Laurea-ammattikorkeakoulu

Webkehitys Reactilla

Minka Laasonen

Tietojenkäsittely

Opinnäytetyö

Marraskuu, 2021

Laurea-ammattikorkeakoulu **Tiivistelmä**

Tietojenkäsittely

Tradenomi (AMK)

Minka Laasonen

Webkehitys Reactilla

Vuosi 2021 Sivumäärä 27

Opinnäytetyön lähtökohtana oli tarve ammatillisen osaamisen kehittämiselle verkkokehittämisen alalla. Työssä lähdettiin tutustumaan React-nimiseen JavaScript kirjastoon. Se valikoitui kohteeksi, sillä se on JavaScript kirjastojen sekä sovelluskehyksien keskuudesta yksi suosituimmista. Tavoitteeksi muodostui tutustua verkkokehittämisen nykytilaan sekä Reactiin ja toteuttaa kehitysprojekti tätä uutta teknologiaa käyttäen.

Opinnäytetyössä käsitellään projektin toteutuksessa käytettyjä teknologioita sekä verkkokehittämisen ajankohtaisia trendejä. Trendien osalta keskitytään erityisesti niihin, joiden kehittämiseen Reactia voidaan hyödyntää.

Lopputuloksena syntyi yksinkertainen React-kirjastolla toteutettu verkkosovellus, joka täytti kaikki sille asetetut vaatimukset ja johon oltiin tyytyväisiä sen suhteellisesta yksinkertaisuudestaan huolimatta. Projektin kehityksen sekä opinnäytetyön kirjoituksen yhteydessä saatiin tavoiteltu peruskäsitys React-kirjaston toiminnasta sekä käytöstä kehitystyössä sekä kattava ymmärrys verkkokehittämisen trendeistä.

Asiasanat: JavaScript, React, verkkokehitys

Laurea University of Applied Sciences

Business Information Technology

Bachelor of Business Administration

Minka Laasonen

Abstract

Web Development with React

Year 2021 Pages 27

The premise for this thesis was the need to develop professional competence in the field of web development. The thesis delves into a JavaScript library called React. It was selected as the focus as it is one of the most popular among JavaScript libraries and frameworks. Exploring the current state of web development and React as well as developing a project using this new technology formed as the goals of this thesis.

This thesis addresses the technologies used in the development of the project and the current trends of web development. In the case of the trends it focuses especially on the ones which can utilize the use of React in their development.

The result of the thesis was a simple web application developed using React that fulfilled all the requirements assigned to it and in which we were satisfied with despite its relative simplicity. In the process of developing the project and writing the thesis the basic understanding of React and its development work and a comprehensive understanding of the web development trends were gained.

Keywords: JavaScript, React, web development

Sisällys

1 Johdanto

Internet on muuttunut valtavasti siitä, mitä se oli vielä kaksi vuosikymmentä sitten. Älypuhelimien saapuminen markkinoille on pakottanut verkon mukautumaan monenlaisille erilaisille laitteille pelkän tietokoneen näytön sijaan. Sen lisäksi netistä on tullut interaktiivisempi – käyttäjät haluavat netiltä muutakin, kuin vain mahdollisuuden lukea tietoa staattisilta sivuilta.

Laitteiden monimuodostumiseen sekä vaatimusten kovenemisen haasteisiin vastaamisen helpottamiseksi on kehitetty monia erilaisia työkaluja, resursseja ja avoimenlähdekoodin projekteja. Niitä ovat kehittäneet myös suuret teknologiajätit voidakseen vastata käyttäjiensä tarpeisiin. Yksi monien muiden apuvälineiden joukossa on Facebookin kehittämä JavaScript kirjasto nimeltään React.

Opinnäytetyössä perehdytään ensin verkkokehityksen nykytilaan ja käsitellään sen jälkeen tarkemmin Reactia, sitä, mikä se on ja mihin sitä käytetään, sekä muita teknologioita, joita sen ohella usein käytetään. Työn loppuosa käsittelee aiemmin läpikäytyjen tietojen avulla kehitetyn projektin toteutusta ja ihan viimeiseksi sen onnistumista, jatkokehitystarpeita sekä työtä kokonaisuudessaan.

2 Lähtökohdat

Opintojen aikana kehittynyt kiinnostus verkkosivujen kehittämistä kohtaan on johtanut kyseistä kiinnostusta vastaavien työpaikkailmoitusten tarkasteluun – etenkin harjoittelupaikkaa etsiessä. Näissä frontend- ja webkehityksen työtehtävien ilmoituksissa on usein mainittu osaamisvaatimuksena React, Angular tai Vue. Yksikään niistä ei ollut minulle tuttu opintojen kautta, vaikka toki nimellisesti tulivatkin nopeasti tutuiksi.

Angular ja Vue ovat JavaScript sovelluskehyksiä ja React on JavaScript kirjasto. Kaikkia kolmea käytetään verkkosivujen ja web sovellusten kehittämiseen. Niitä käytetään laajasti modernissa webkehittämisessä ja ainakin yhden hallitseminen on siis tärkeää alan osaajalle. Tämä tarve osaamiselle sekä opinnäytetyön aiheen puutos avasivat loistavan tilaisuuden yhdistää kaksi yhdeksi, ammatillista kehittymistä edistäväksi projektiksi.

2.1 Kehittämiskohteen kuvaus ja tavoitteet

Angular, Vue ja React kolmikosta päätin keskittyä tässä työssä Reactiin. Päätös pohjautui lähinnä sen suosioon – työpaikkailmoituksia selatessa se oli mainittu kahta muuta useammin

osaamisvaatimuksena. Lisäksi, kun kyseessä on ennemmin JavaScript kirjasto, kuin kokonainen sovelluskehys, on sen käyttämisen opettelukin hieman helpompaa.

Työssä tavoitteena on tutustua Reactiin, sen toimintaan, web kehittämisen nykytilaan sekä siihen, miten React sopii tämän hetken web kehityksen trendeihin. Lisäksi prosessin aikana toteutetaan oma projekti Reactia hyödyntäen.

2.2 Rajaukset

Kolmena suosituimpana JavaScript sovelluskehyksenä listataan usein jo aiemmin mainitut React, Angular ja Vue. Tässä työssä keskityn kuitenkin käsittelemään vain yhtä, eli Reactia. Muihin kahteen ei tämän työn puitteissa tutustuta sen tarkemmin.

Osaamistason lähtiessä nollasta itse Reactin teknologioiden kannalta, opinnäytetyö prosessin aikana tullaan opettelemaan siitä perusteet. Tähän opetteluvaiheeseen ei kuitenkaan perehdytä opinnäytetyössä. Opinnäytetyö keskittyy vain itse teknologiaan sekä sillä lopulta toteutetun projektin kuvaamiseen.

Kun kyseessä on ensikosketus Reactiin sekä sillä projektin toteuttamiseen, ei myöskään ole mielekästä pyrkiä täydellisyyteen. Työn puitteissa pyrin opettelemaan perusteet ja toteuttamaan toimivan web sovelluksen. Sovelluksen ei myöskään ole tarkoitus olla erityisen monimutkainen ja sen haastavammat osat voidaan jättää tämän työn ulkopuolelle. Tärkeintä on saada aikaan toimiva sovellus.

2.3 Keskeiset käsitteet

Frontend	Verkkosivun tai -sovelluksen käyttäjälle näkyvä osa eli käyttöliittymä.
HTML	Hyper Text Markup Language eli HTML on verkkosivujen kehittämiseen käytettävä kieli, jonka avulla verkkosivuille tehdään erilaisia elementtejä, kuten otsikoita, kuvia ja linkkejä.
JavaScript	Verkkosivujen kehittämisessä käytettävä kieli, jonka avulla sivuille voidaan lisätä erilaisia toiminnallisuuksia, kuten esimerkiksi napin painalluksesta sivulla tapahtuva muutos.
React	Facebookin kehittämä JavaScript kirjasto käyttöliittymien kehittämiseen.
Webkehitys	Web- eli verkkokehityksellä tarkoitetaan verkkosivujen suunnittelua, rakentamista sekä ylläpitoa.

3 Moderni web kehitys

2000-luvun alussa verkkosivut koostuivat lähinnä tekstisisällöstä. Niillä oli paljon erilaisia yksityiskohtia sekä räikeitäkin värejä. Sittemmin verkkosivut ovat muuttuneet melko erilaisiksi – ulkoasuista on tullut yksinkertaisempia sekä hillitympiä samalla, kun sisältö on monipuolistunut ja toiminnallisuudet lisääntyneet. (Murphy 2015.)

Ennen älypuhelimia web kehitys oli jaettu kahteen: verkkoselaimille kehittäminen ja kaikki muu. ”Kaikki muu” piti sisällään silloiset puhelimet sekä joitain muita käsin pideltäviä laitteita. Lähestymistapaan piti kuitenkin tulla muutos, kun älypuhelimet ja sittemmin tabletit ilmestyivät markkinoille. Vaatimukseen vastaukseksi alkoi kehittyä responsiivisia eli mukautuvia verkkosivuja, jotka nimensä mukaan mukautuvat siihen laitteeseen, jolla niitä tarkastellaan. (Esposito 2016.)

Älypuhelimien tulon lisäksi verkko on mullistunut monella muullakin tavalla. Internetissä ei ole enää pelkästään verkkosivuja, vaan nykyään yksi webkehittämisen isoista trendeistä ovat myös web sovellukset. Web sovellukset voivat olla hyvin laajoja sekä monimutkaisia kokonaisuuksia, jotka voivat olla jopa työpöytä- ja mobiilisovelluksiakin mutkikkaampia. (Nehra 2020.)

Responsiivisuus pitää edelleen pintansa web kehittämisen trendien aallonharjalla, sillä mobiililaitteita käytetään aina vain enemmän; vuoden 2020 lopussa maailmassa oli 5.22 miljardia älypuhelimen käyttäjää, mikä on noin 66 % maailman väestöstä (Jay, 2021). Tämän päivän trendeihin kuuluvat myös edellä mainitut web sovellukset, niistä etenkin yhden sivun applikaatiot eli SPA:t (engl. single page application) ja progressiiviset verkkosovellukset (engl. PWA, progressive web apps). Trendeissä nähdään lisäksi muun muassa ääniohjautuvuutta, chat-botteja, Push-ilmoituksia sekä AMP, eli Accelerated Mobile Pages, joka on Googlen julkaisema uusi sovelluskehys mobiilisivujen kehittämiseen. (Kuprenko 2021.)

Seuraavissa kappaleissa käsitellään web kehittämisen trendeistä responsiivisia verkkosivuja sekä verkkosovelluksia, sillä niiden kehittämisessä voidaan käyttää myöhemmin työssä käsiteltävää React-kirjastoa.

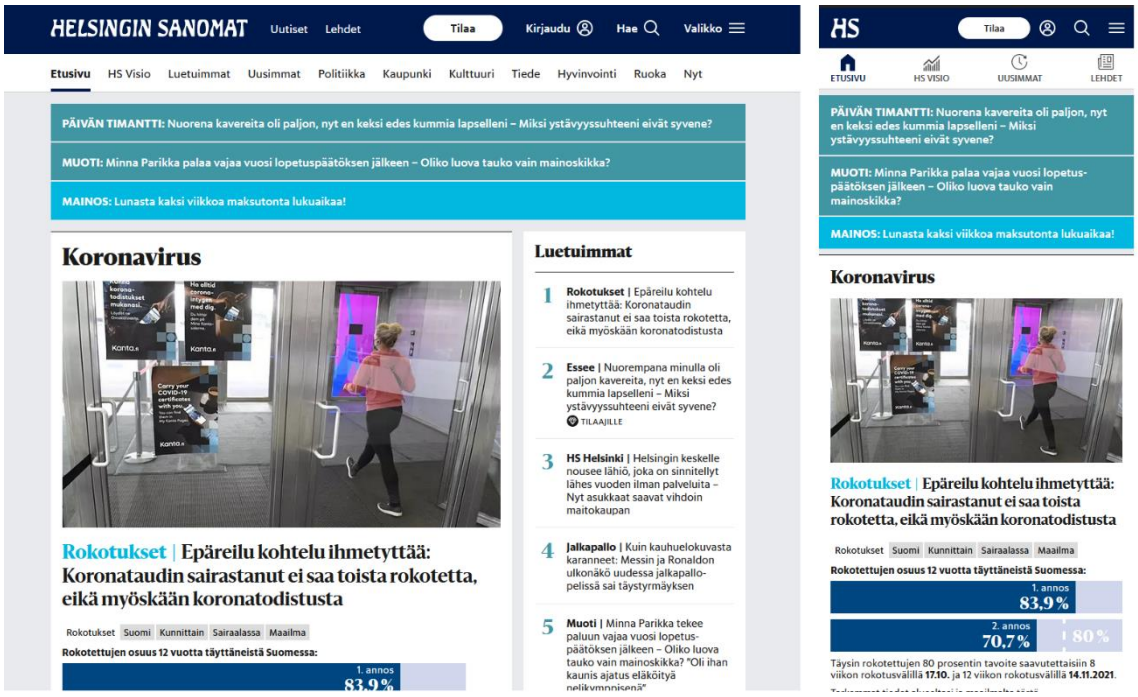
3.1 Responsiiviset verkkosivut

Jo vuonna 2005 kansainvälinen verkkostandardien kehittämiseen omistautunut World Wide Web Consortium (W3C) tiedosti mobiililaitteiden tuomat haasteet verkkokehitykselle. Ratkaisuna W3C julkaisi visionsa niin sanotusta Yhdestä Netistä (One Web), jossa sisältö olisi helposti saatavilla niin tietokoneilla kuin mobiililaitteillakin. (Gardner 2011.)

Vision toteuttamiseksi syntyi myöhemmin responsiivinen verkkosuunnittelu (responsive web design), jossa verkkosivun sisältö ja arkkitehtuuri mukautuu käytetyn laitteen näytön

leveyteen. Responsiiviset verkkosivut hyödyntävät joustavaa asettelua, jossa eri elementtien leveydet on usein määritelty prosenttimäärissä kiinteän pikselimäärän sijaan. Näin elementtien koko muuttuu näytön koon mukana. (Gardner 2011.)

Responsiivinen sivun muotoilu siis pohjautuu käytössä olevan laitteen leveyteen, jonka mukaiseksi verkkosivun asettelu mukautuu, kuten esimerkiksi Helsingin Sanomien sivut kuviossa 1. Eri elementit voivat vaihtaa paikkaa tai jotkin niistä voivat hävitä sivulta kokonaan riippuen laitteen leveydestä. Näin estetään tilanteita, jossa käyttäjän tarvitsisi zoomailla sivulla nähdäkseen tai saavuttaakseen sen eri osat.



Kuvio 1: Responsiivinen suunnittelu Helsingin Sanomien sivuilla

Monille, etenkin yrityksien, sivuille hakukoneoptimointi on tärkeää. Myös tässä mobiililaitteilla toimivat sivut ovat lähes välttämättömät, sillä Googlen algoritmi arvottaa hakutuloksia mobiilitoimivuuden mukaan. Jo vuodesta 2016 asti Google on tehnyt testejä hakutuloksien järjestämisestä verkkosivujen mobiiliversion pohjalta (Google Search Central 2021a) ja vuodesta 2019 alkaen uusien sivujen arvotus hakutuloksissa on tapahtunut sivujen mobiiliversioiden mukaan (Google Search Central 2021b). Näin ollen mobiililaitteilla toimivat sivut nousevat hakutuloksien kärkeen ja niillä toimimattomat sivut tippuvat hännille.

Hakukonetulosten paraneminen lisää luonnollisesti kävijöitten määrää sivulla, sillä moni valitsee hakutuloksista yhden ensimmäisistä vaihtoehdoista. Lisäksi sivun toimiessa mobiililaitteilla käyttäjien käyttäjäkokemus paranee ja he viettävät mieluummin enemmän aikaa tutkien sivuja, mikä taas johtaa yrityksillä asiakkaiden lisääntymiseen. (Raja 2018.)

3.2 Websovellukset

Jokainen Internetin käyttäjä on todennäköisesti käyttänyt jossakin vaiheessa jotakin web- eli verkkosovellusta. Muun muassa sosiaalisen median alustojen verkkoversiot ovat verkkosovelluksia, samoin kuin sähköpostikin. Selaimella käytettyinä esimerkiksi Twitter, Facebook ja Gmail ovat verkkosovelluksia.

Nimensä mukaankin verkkosovellukset ovat verkossa toimivia sovelluksia, jotka rakennetaan yleisillä verkon kehitysvälineillä: HTML, CSS ja JavaScript. Verkkosovelluksia ei tarvitse asentaa omalle laitteelleen, sillä niitä käytetään verkossa, mikä onkin niiden merkittävin ero tavallisiin sovelluksiin verrattuna. Verkkoon rakennettuina ne pystyvät toimimaan minkä tahansa laitteen selaimella, jos sillä on tuki niiden kehittämiseen käytettyihin teknologioihin. (Fling 2009.)

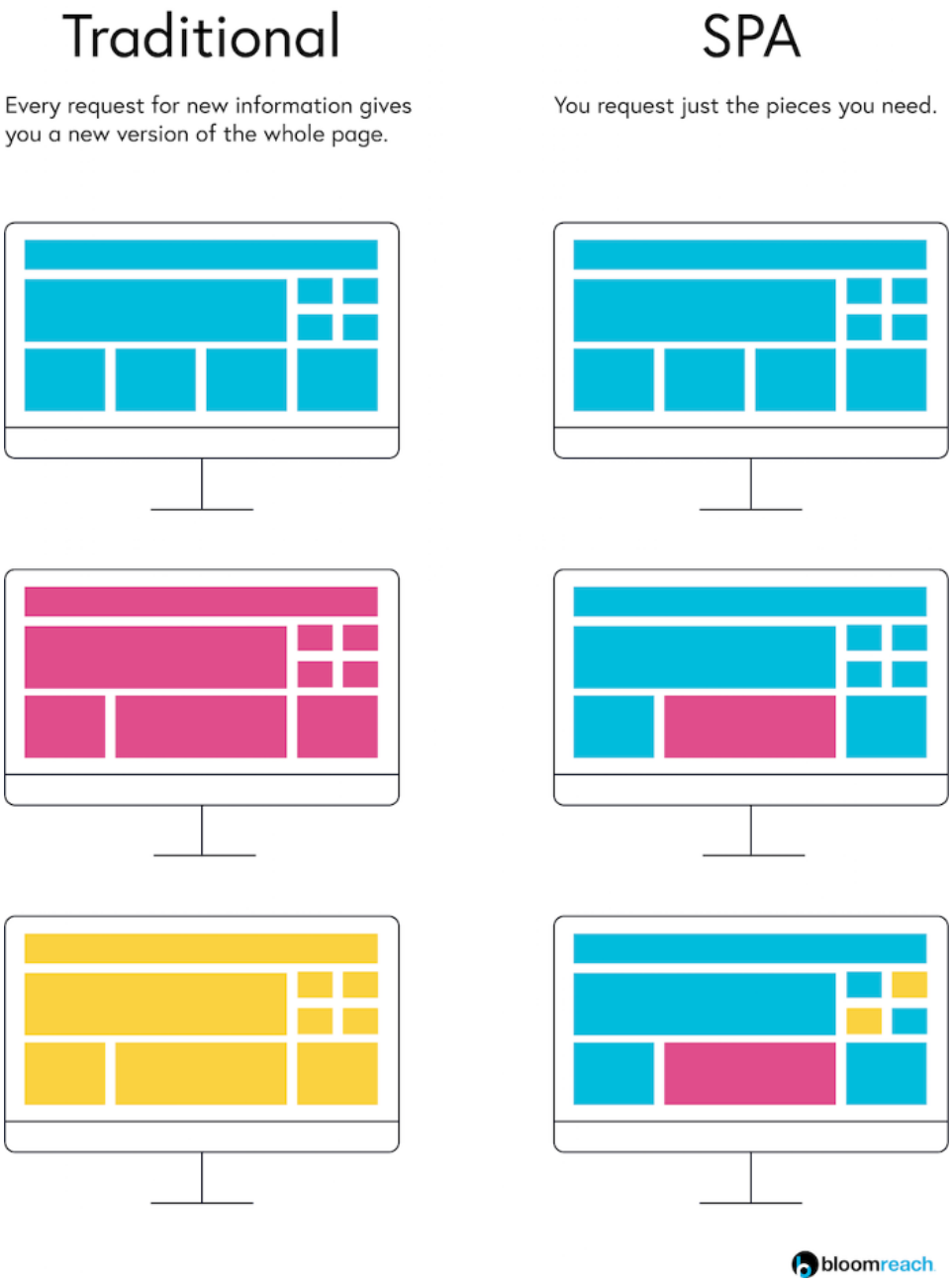
Samoista rakennusvälineistä huolimatta verkkosivuilla ja -sovelluksillakin on eroja. Etenkin interaktiivisuus on yksi näiden kahden merkittävimpiä eroja. Verkkosovellukset ovat monipuolisia työkaluja, joilla käyttäjät voivat suorittaa monenlaisia tehtäviä. Ne pyrkivät tarjoamaan käyttäjilleen mahdollisimman paljon natiivisovellusta muistuttavan käyttäjäkokemuksen. (Fling 2009.) Verkkosivutkaan eivät kuitenkaan ole täysin ilman interaktiivisuutta, sillä monilta perinteisimmiltäkin verkkosivuilta löytyy esimerkiksi mahdollisuus kommentoimiseen. Tästä syystä verkkosivuja ja -sovelluksia ei voi erotella pelkästään interaktiivisuuden kannalta, eikä niitä ole aina helppoa täysin erottaa toisistaan.

Työpöytäsovelluksiin verrattuna verkkosovelluksien etuihin kuuluvat muun muassa se, ettei niitä tarvitse ladata omalle laitteelle, eikä päivittää itse – verkkosovellusten päivitys tapahtuu keskitetysti ja verkossa oleva versio on aina ajan tasalla. Ne eivät myöskään ole riippuvaisia käytetyn laitteen käyttöjärjestelmästä, vaan toimivat minkä tahansa laitteen selaimessa. Verkkosovelluksien kehittäminen on näistä syistä nopeampaa ja halvempaa, kuin natiivisovellusten. Perinteiset verkkosovellukset kuitenkin häviävät tavallisille sovelluksille nopeuden sekä Internet yhteyden tarpeen. Lisäksi esimerkiksi laitteen kameraa on helpompi ja turvallisempi käyttää natiivisovelluksissa. (Tandel & Jamadar 2018.)

Kaksi suosittua tapaa rakentaa verkkosovellus ovat SPA ja PWA. SPA on yksisivuinen verkkosovellus (engl. Single Page Application) ja PWA on progressiivinen verkkosovellus (engl. Progressive Web App). Nämä kaksi ovat jokseenkin samankaltaisia, mutta niistä löytyy myös eroja. (Vu 2019.)

Nettisivut ovat täynnä toistuvaa sisältöä, joista osa pysyy täysin muuttumattomana riippumatta siitä, missä osassa sivustoa käyttäjä on ja mitä hän siellä tekee. Tällaisia elementtejä ovat esimerkiksi logot, navigaatiopalkit sekä ylä- ja alatunnisteet. SPA:t käyttävät tätä toistoa hyväkseen. Tavallisista verkkosivuista poiketen käyttäjän esimerkiksi navigoidessa eri sivulle

SPA ei lataa koko sivua uudestaan, vaan päivittää vain sen osan sivusta, jossa muutoksia tapahtuu (kuvio 2). Pienten osien lataaminen isojen kokonaisuuksien sijaan nopeuttaa huomattavasti sivun toimintaa. (Lawson 2021.)



Kuvio 2: Tavallisen verkkosivun ja SPA:n toiminnan ero (Lawson 2021)

SPA:n heikkous voi tulla juuri sen toimintatavasta, sillä sen pitää ladata kaikki tarvittava sisältö palvelimelta, kun sovellus avataan. Isoissa sovelluksissa, joissa ladattavaa sisältöä on paljon, käyttäjä voi joutua odottamaan sen avautumista hetken. Hyvä esimerkki tästä on Gmail, jonka käytössä kaava on nähtävillä; käyttäjän täytyy odottaa hetki, kun sähköposti avautuu, mutta sen jälkeen sitä on helppo ja nopea käyttää. Tätä ongelmaa ratkaisemaan on

olemassa erilaisia työkaluja, joiden avulla latausaika voidaan optimoida niin, ettei käyttäjän tarvitse tuijottaa tyhjää sivua odottaessaan sivun latautumista. Muita tunnettuja esimerkkejä yksisivuisista sovelluksista Gmailin lisäksi ovat muun muassa Facebook, LinkedIn ja Netflix (Vu 2019).

Progressiiviset verkkosovellukset puolestaan kurovat natiivi- ja verkkosovellusten välistä kuilua umpeen mobiililaitteilla. Ne toimivat monella tapaa samankaltaisesti, kuin laitteelle asennettavat natiivisovelluksetkin – ne tuntuvat ja näyttävät samanlaisilta ja PWA:n kuvakkeen voi lisätä aloitusnäytölle samaan tapaan, kuin asennetun sovelluksenkin. (Tandel & Jamadar 2018.)

Muihin verkkosovelluksiin verrattuna progressiivisilla verkkosovelluksilla on yksi merkittävä etu; kyky toimia myös ilman internet yhteyttä. Offline toiminnan mahdollistaa JavaScript ohjelma (service worker), joka myös nopeuttaa sivun latautumista. Progressiivisella verkkosovelluksella on lisäksi myös manifesti eli kuvailutiedosto, jonka avulla se toimii tavallisen sovelluksen tavoin, sekä salattu https-yhteys. PWA pystyy myös hyödyntämään mobiililaitteen omia resursseja, kuten kameraa ja GPS paikannusta. (Lindström 2019.)

Joidenkin, etenkin pienien kehitystiimien ja yksittäisten kehittäjien mielestä yksi progressiivisten verkkosovellusten eduista natiivisovelluksiin verrattuna liittyy sovelluskauppoihin; PWA:t eivät nimittäin ole saatavilla perinteisissä sovelluskaupoissa, vaan netissä. Sovelluskaupat ottavat oman osuutensa sovellusten myyntituotoista, esimerkiksi Applen App Store, ottaa 15 prosentin osuuden alle miljoonan vuodessa tuottavien sovellusten hinnasta ja 30 % sen yli menevistä. Kaikki eivät välttämättä halua menettää tätä osuutta omista tuotoistaan. Lisäksi sovellusten saaminen sekä niiden päivittäminen sovelluskauppoihin voi olla vaivanloista, sillä niiden täytyy mennä sovelluskaupan hyväksynnän läpi. Verkkosovelluksissa kehittäjät saavat itse tuotot sovelluksen myynnistä ja saavat tärkeät päivitykset heti käyttäjien saataville. (Newman 2021.)

Verkkokaupasta puuttumisessa on kuitenkin myös huonot puolensa. Progressiivisten verkkosovellusten ollessa vielä suhteellisen uusi asia, lähtee harva mobiilikäyttäjä etsimään uutta sovellustaan netistä. Mobiililaitteille sovelluksia etsitään ensisijaisesti sovelluskaupoista ja tässä verkkosovellukset häviävät potentiaalisia käyttäjiä natiivisovelluksille. Lisäksi Applen käyttöjärjestelmillä kaikki PWA:n ominaisuudet eivät toimi, se ei esimerkiksi voi lähettää ilmoituksia iOS laitteilla. Joitakin näistä puuttuvista toiminnoista pidetään välttämättöminä natiivisovelluksen tapaisen kokemuksen tarjoamiselle. Tärkeiden toiminnallisuuksien puutteen takia verkkosovellukset siis häviävät iOS laitteilla natiivisovelluksille käytettävyydessä. Androidilla tuki verkkosovelluksille on huomattavasti parempi ja iOS laitteillakin tilanne saattaa vielä tulevien järjestelmäversioiden myötä parantua. (Newman 2021.)

Molemmat, sekä PWA että SPA ovat sovellustyyppisiä käyttäjäkokemukseltaan, mutta näistä kahdesta PWA muistuttaa enemmän tavallista sovellusta ja voi siten tarjota paremman käyttäjäkokemuksen. Tällä hetkellä kovassa nousussa olevat progressiiviset verkkosovellukset ovat kuitenkin kalliimpia kehittää, kuin yksisivuiset sovellukset. (Vu 2019.)

4 Käytetyt teknologiat

Opinnäytetyön yhteydessä toteutetun projektin tekemisessä tärkeimmäksi muodostuivat kolme teknologiaa – React ulkoasun toteutukseen, Node.js taustan toiminnallisuuksiin sekä paketinhallintajärjestelmä projektin yleiseen hallintaan.

4.1 React

Vuoden 2010 tienoilla Facebookin kehittäjät törmäsivät ongelmaan – alustan Facebook Ads-sovelluksen ominaisuuksien kasvaessa tarvittiin lisää kehittäjiä pitämään sen toiminta virheettömänä. Mutta kasvava tiimien koko ja lisääntyvät ominaisuudet tekivät koodin hallitsemisesta vaikeaa. Ongelmaan tarvittiin parempia ratkaisuja. (Occino 2015.)

Jordan Walke loi ongelmaan ratkaisuksi prototyypin, jonka pohjalta myöhemmin kehitettiin React. Vuonna 2013 Facebook julkaisi Reactin avoimen lähdekoodin version. Sen jälkeen React on kehittynyt paljon ja onnistunut kohoamaan yhdeksi suosituimmista webkehityksen työkaluista, vaikuttaen omalta osaltaan paljon siihen, miten verkkoa kehitetään. (Occhino 2015.)

Vaikka React usein listataankin yhdessä Angularin ja Vuen kanssa yhdeksi kolmesta suosituimmasta JavaScript sovelluskehyksistä (eng. framework), ei se oikeastaan ole sitä. React on JavaScript kirjasto, joka on kehitetty ensisijaisesti käyttöliittymien kehitykseen. Toisin kuin kaksi kilpailijaansa, sitä ei ole tarkoitettu palvelinpuolen kehittämiseen ja jotta sitä voitaisiin siihen hyödyntää, tarvitaan kolmannen osapuolen JavaScript kirjastoja. (Starikov 2019.)

JavaScript kirjastot ovat kokoelmia valmiiksi kirjoitettua koodia, jota kehittäjät voivat käyttää tavallisimmissa tehtävissä, joita JavaScriptillä suoritetaan. Tällaisia ovat esimerkiksi animaatiot tai hakukentän ennakointi. Usein toistuvien asioiden uudelleen koodaaminen on työlästä ja vie turhaa aikaa, sillä saman ominaisuuden ovat koodanneet jo monet muutkin ihmiset aiemmin. Tässä JavaScript kirjastot auttavat tarjoamalla saman koodin valmiina hyödynnettäväksi. Näin helpotetaan sekä nopeutetaan ohjelmoijan työtä. (Morris 2021.)

React on JavaScript-kirjasto, joka on tarkoitettu käyttöliittymän (frontend) rakentamiseen. Tämä tarkoittaa sitä, että Reactin avulla voidaan rakentaa kaikki se, minkä käyttäjä näkee ja

minkä kanssa hän on vuorovaikutuksessa käyttäessään verkkosivua tai -sovellusta. (Morris 2021.)

Sen lisäksi, että React tarjoaa valmista koodia käyttäjilleen, sillä on myös kaksi muuta tärkeää ominaisuutta: JSX ja VirtualDOM. (Morris 2021.)

JSX tulee sanoista JavaScript Extension Syntax. JSX on siis JavaScriptin laajennus, joka tavallaan yhdistelee HTML sekä JavaScript koodia, antaen kehittäjien kirjoittaa HTML koodia suoraan JavaScript tiedostoon, mikä ei muutoin olisi mahdollista. Tästä esimerkki kuviossa 3, jossa elementti on kirjoitettu JSX:ää hyödyntäen ja kuvio 4, jossa JSX:ää ei ole käytetty. Lisäksi se tekee muuttujien sekä JavaScript koodin upottamisen suoraan HTML elementtien sisälle helpoksi, kuten kuviossa 5. (React 2021a)

```
const myelement = <h1>Otsikko JSX:ällä</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```

Kuvio 3: HTML elementti JSX:ällä kirjoitettuna

```
const myelement = React.createElement('h1', {}, 'Otsikko ilman JSX:ää');

ReactDOM.render(myelement, document.getElementById('root'));
```

Kuvio 4: HTML elementti kirjoitettuna ilman JSX:ää

```
const Esimerkki = () => {
    const nimi = "Matti";

    return(
        <h1>Hei {nimi}!</h1>
    );
};
```

Kuvio 5: JavaScript koodin upottaminen suoraan HTML elementin sisälle JSX:än avulla

Jokaisen verkkosivun keskiössä on HTML. Selain lukee sivun HTML tiedostot ja näyttää ne käyttäjälle. Samalla, kun selain lukee tiedostoja, se tekee niistä dokumenttioliomallin eli DOM:in (engl. Document object model), joka kuvaa HTML dokumentin rakenteen puuna.

Verkkosivuille voidaan tehdä erilaisia toiminnallisuuksia esimerkiksi JavaScriptin avulla muokkaamalla dokumenttioliomallia. (Morris 2021.)

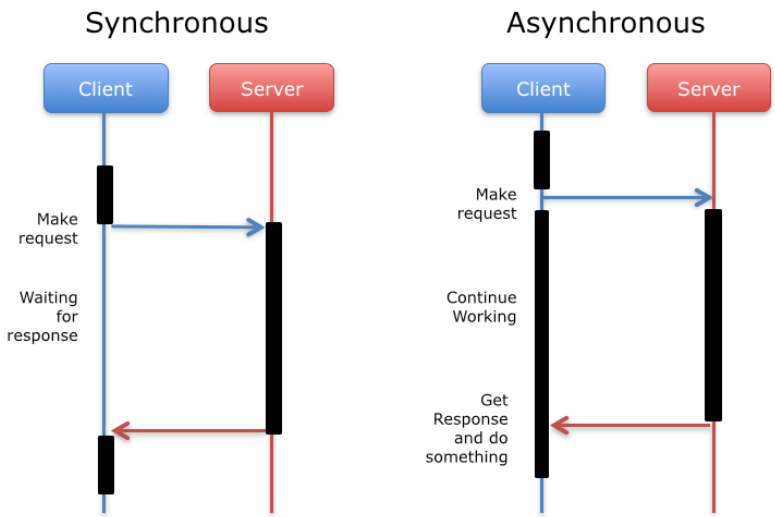
Jos kehittäjällä on käytössään JSX dokumenttioliomallin päivittämiseksi ja muokkaamiseksi, luo React asian nimeltään Virtual DOM. Virtual DOM on kopio sivun dokumenttioliomallista, jota React käyttää määritelläkseen, mitkä osat oikeasta DOM:ista täytyy muuttaa. Näin ollen, kun sivulle tehdään muutoksia Reactin tapauksessa JSX:än avulla, vertaillaan Virtual DOM:ia oikeaan DOM:iin ja päivitetään jälkimmäisestä vain ne osat, joissa muutoksia tapahtuu. Tämä nopeuttaa sivun toimintaa huomattavasti verrattuna tavalliseen tapaan, jossa koko DOM päivitettäisiin, vaikka muutoksia tapahtuisikin vain pienessä osassa sivua. (Morris 2021.)

React on hyvin monipuolinen ja se toimii kaikilla selaimilla sekä käyttöjärjestelmillä. Sillä on paljon potentiaalia sopia minkä tahansa käyttöliittymän kehittämiseen mille tahansa laitteelle. (Boduch A. & Dereks R. 2020.) Tällä hetkellä se on käytössä isoilla alustoilla, esimerkiksi Instagram ja Facebook käyttävät sitä verkkoversioissaan, joista se onkin alun perin saanut alkunsa (Occhino 2015).

4.2 Node.js

Node.js on vuonna 2009 ilmestynyt asynkroninen ja tapahtumakeskeinen JavaScript ajoympäristö. Se mahdollistaa JavaScriptin käytön palvelinpuolella. Sitä käytetään JavaScript sovellusten kehittämiseen, kuten web applikaatioihin, palveluprosesseihin ja työpöytäsovelluksiin. (Young, Meck & Cantelon 2017.)

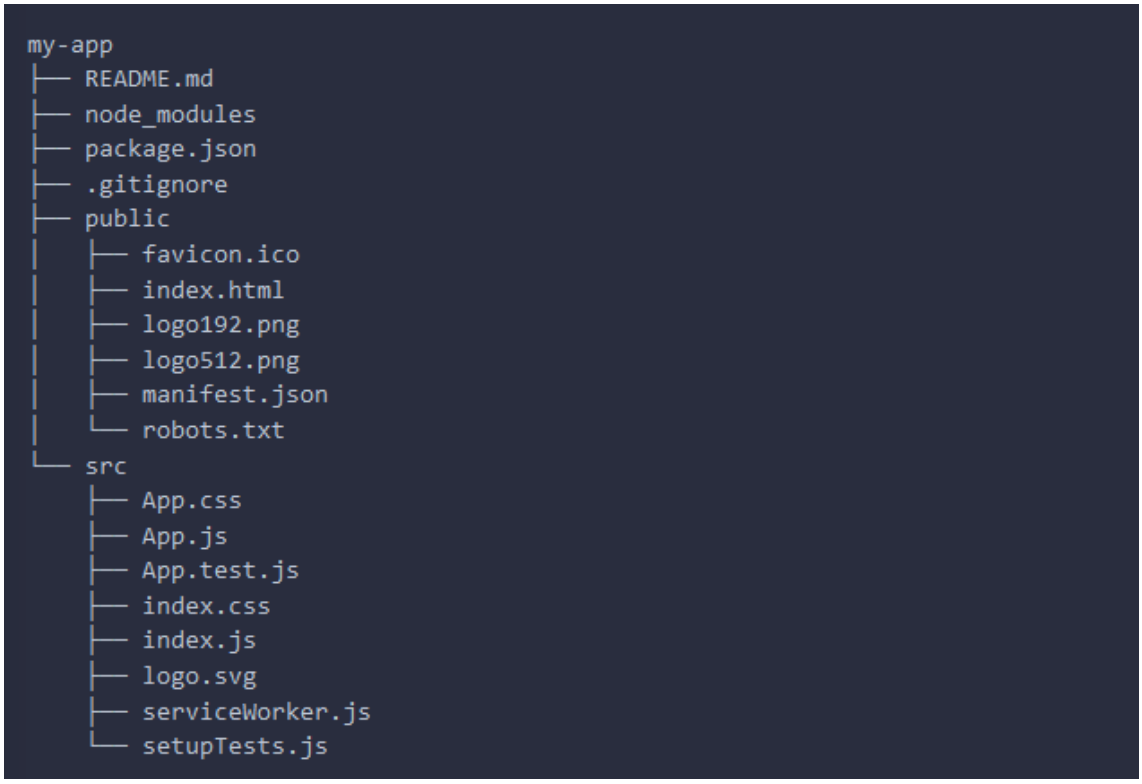
Käyttäjät eivät halua odottaa tyhjänpanttina sillä välin, kun ohjelma lataa. Tästä syystä Node.js hyödyntää tukkiutumatonta I/O mallia, asynkronisia ohjelmointirajapintoja sekä tapahtumia välttääkseen palvelun tukkiutumisen. Toisin, kuin synkroninen pyyntö, asynkroninen ei jää odottamaan toimettomana vastausta palvelimelta, vaan siirtyy käsittelemään muita pyyntöjä ja palaa palvelimelta saatuun vastaukseen, kun se on saatavilla. (Young ym. 2017) Kuvio 6 esittää asynkronisen ja synkronisen haun toimintamekanismit visuaalisesti.



Kuvio 6: Synkroninen vs. asynkroninen pyyntö (Giridhar 2017)

Ominaisuuksiensa puolesta Node.js sopii hyvin paljon dataa käyttäviin sovelluksiin, kuten striimaukseen. Node.js onkin käytössä esimerkiksi Netflixillä ja Uberilla. (Kopecky 2020.)

Yksi helppo ja nopea tapa aloittaa uusi React projekti on käyttää hyväkseen komentorivillä käytettävää create-react-app-käskyä. Se luo kehitysympäristön valmiiksi, eikä kehittäjän itse tarvitse huolehtia sen luomisesta. (React 2021b.) Create React App luo alustavan projektirakenteen (kuvio 7) ja asentaa tarvittavat riippuvuudet. Node.js tarvitaan tämän käskyn käyttämiseen. (McCurdy 2020.)



Kuvio 7: Create React App käskyn luoma projektirakenne (McCurdy 2020)

4.3 Yarn

JavaScriptille on olemassa kaksi suosittua paketinhallintajärjestelmää: NPM ja Yarn. Pidempiikäinen NPM hallitsee listoja vielä latauksien puolesta, mutta Yarn tulee nopeasti perässä ja on selvästi johdossa GitHub aktiivisuudessa. (Npm trends 2021.)

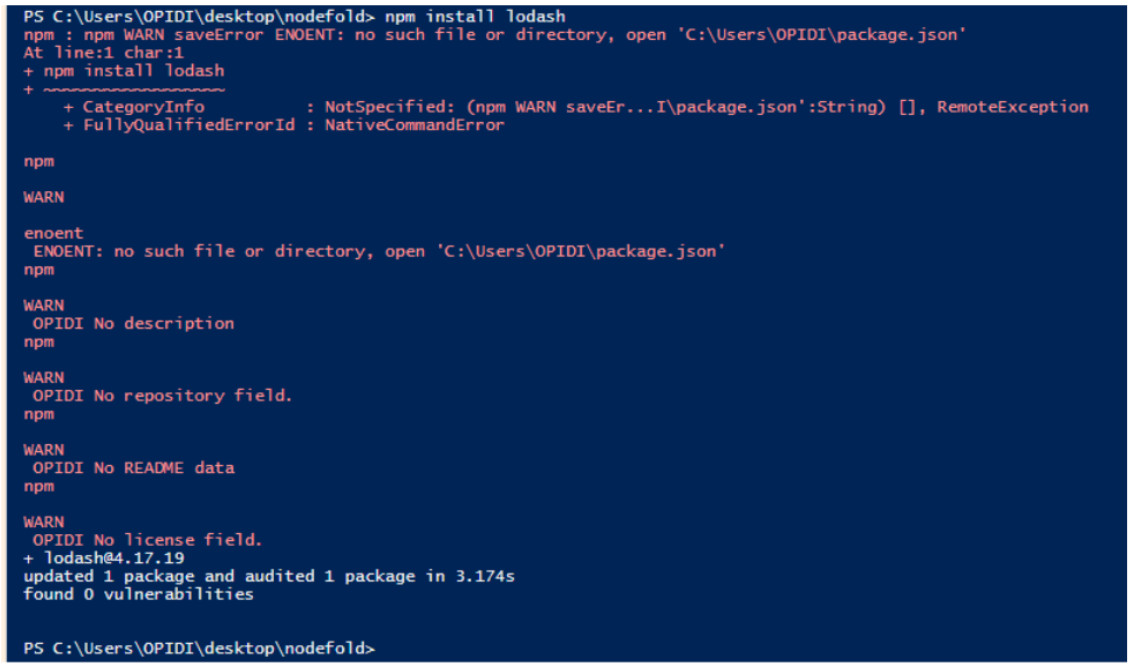
Yarn eli Yet Another Resource Negotiator on Facebookin kehittämä paketinhallintajärjestelmä. Se kehitettiin vuonna 2016 korjaamaan Facebookin kehittäjien NPM:ässä kohtaamia yhtenäisyys-, suoritus- ja turvallisuushuolia. (Nakazawa, McKenzie, Kyle, 2016.) Tätä nykyään silloiset ongelmat on jo NPM:ästä korjattu. (Bar-Gil 2020.)

Paketinhallintajärjestelmät, kuten Yarn ja NPM, on tarkoitettu helpottamaan ohjelmien asentamista, päivittämistä sekä poistamista. Ne ovat komentorivillä käytettäviä ohjelmia, jotka nimensä mukaan hallitsevat paketteja eli erilaisia tiedostorykelmiä. Paketinhallintajärjestelmät helpottavat huomattavasti kehitystyötä hoitamalla ihmisen puolesta tiedostojen asetukset ja poistamiset sekä päivitykset muutamalla käskyllä sen sijaan, että se tarvitsisi tehdä itse tiedosto kerrallaan. (Levins 2020.)

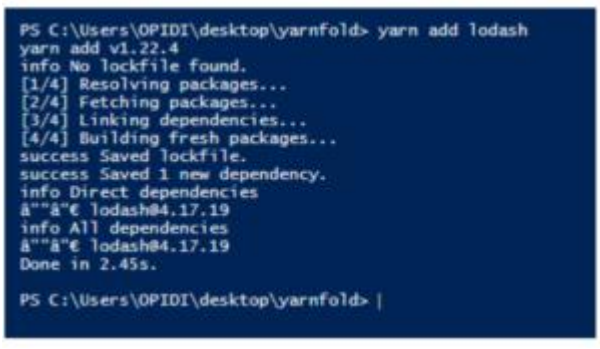
Etenkin suurien projektien kanssa paketinhallintajärjestelmät ovat tarpeellisia, sillä isojen projektien kohdalla sen riippuvuuksien hallitseminen itse on vaikea, ellei jopa mahdotonkin,

tehtävä. Paketinhallintajärjestelmät tekevät projektien toiminnan kannalta välttämättömien osien hallinnasta helppoa. (Hooda 2020.)

Toiminnaltaan NPM ja Yarn ovat samankaltaisia, sillä molemmat on kehitetty suorittamaan samaa tehtävää. Huomattavimpia eroavaisuuksia ovat: Yarnilla on joitain komentoja, joita NPM:ällä ei ole, molemmat luovat erilaisia tiedostoja ja tiedostorakenteita, ja molempien tuloste komentorivillä on erilainen. Yarnin tuloste komentorivillä esimerkiksi ohjelmaa asentaessa on helpommin ymmärrettävä, kuin NPM:än, sillä Yarnin tuloste on rakennettu puumaisesti helpottamaan ymmärrystä, kuten kuvioista 8 ja 9 näkee. (Hooda 2020.)



Kuvio 8: NPM tuloste ohjelmaa asentaessa (Bar-Gil, 2020).



Kuvio 9: Yarn tuloste ohjelmaa asentaessa (Bar-Gil, 2020).

Yarn on suosittu erityisesti React-kehittäjien keskuudessa. Vuonna 2017 NodeSourcen perustajajäsen Dan Shaw sanoi heidän asiakkaistaan noin neljänneksen alkaneen käyttää Yarnia.

Hänen mukaansa Yarnin käytöllä oli vahva korrelaatio Reactin käytön kanssa. (Krill 2017.) Ikävä kyllä uudempia tai tarkempia statistiikkoja Yarnin ja Reactin käytön korrelaatioon ei löytynyt. Shawinkin antama tieto oli mitä todennäköisimmin pelkkä arvio ja nyt neljä vuotta myöhemmin jo vanhentunut, mutta se kertoo silti hyvin Yarnin nopeasta suosion kasvusta jo alle vuosi sen julkaisun jälkeen. Ei ole kuitenkaan mikään ihme, jos Facebookin kehittämää Reactia käytetään yhdessä Facebookin kehittämän Yarnin kanssa.

5 Projektin toteutus

Opinnäytetyötä varten toteutettiin React-projekti. Ideana oli antaa vastaus klassiseen kysymykseen ”mitä tänään on ruokana?” arpomalla käyttäjälle yksi vaihtoehto listasta. Sovelluksessa oli tarkoitus olla valmiina lista vaihtoehdoista, mutta sen lisäksi käyttäjä pystyisi lisäämään omia vaihtoehtojaan ja poistamaan valmiita sekä itse lisättyjä vaihtoehtoja listasta.

Projektin toteutus alkoi sen suunnittelulla (minkä yhteydessä tehtiin myös projektin toiminnallisuuksien määrittely), edistyen sen jälkeen käytännön toteutukseen, johon kuuluivat toiminnallisuuksien käytännön toteutus sekä ulkoasun toteutus.

5.1 Suunnittelu

Suunnittelu on kaikenlaisille projekteille hyvin tärkeä vaihe. Siinä kartoitetaan projektin toiminnalliset ominaisuudet sekä ulkonäkö. Näiden asioiden miettiminen etukäteen helpottaa sen toteuttamista ja vähentää virheiden määrää. Usein kehitysprojekteissa, ennen suunnittelua, tehdään projektin määrittely, mutta tässä pienehkössä projektissa nämä kaksi yhdistettiin.

Toiminnallisuuksien määrittelyä varten voidaan tehdä vuokaavioita ja ulkoasun suunnittelua varten toiminnallisia prototyyppejä, mutta tässä projektissa lähestymistapa oli yksinkertaisempi. Näin pienen projektin toteutuksessa ei koettu mielekkääksi kuluttaa aikaa kovin monimutkaiseen suunnitteluun tai määrittelyyn. Ulkoasun suunnittelu aloitettiin tekemällä yksinkertainen, raaka luonnos tulevasta projektista paperille.

Paperille tehty raaka luonnos piti sisällään hahmotelman projektin visuaalisesta ilmeestä ja sen eri elementeistä. Näiden lisäksi paperille listattiin toiminnallisuudet, joita kyseisessä sovelluksessa tulisi olemaan, tämä toimi osaltaan projektin määrittelynä.

Paperiluonnoksen jälkeen tehtiin vielä tarkempi luonnos ulkoasusta tietokoneella, mikä piti sisällään myös sovelluksen värit. Näiden värien päättämiseen käytettiin apuna coolors.co-sivustoa, joka antaa käyttäjän luoda värikarttoja helposti arpomalla. Palvelussa saa päättää värien lukumäärän, lukita haluamansa värit paikoilleen ja jatkaa muiden värien arpomista.

5.2 Toiminnallisuuden toteutus

Suunnitteluvaiheessa paperille tehtiin lista sovelluksen vaatimuksista, sen eri elementeistä ja siitä, mitkä elementtien toiminnallisuudet tulisivat olemaan. Lista oli suurin pirtein seuraavanlainen:

- Toimittava mobiililaitteilla
- Arvonta pyörä, jota käyttäjä voi pyöräyttää napin painalluksella
    - Pyöräytyksen yhteydessä arvotaan ruoka käyttäjälle
- Lista ruokalajeista
    - Joitain valmiita esimerkkejä
    - Käyttäjälle mahdollisuus lisätä ja poistaa asioita listasta

Ensin toteutettiin lista, jossa eri ruoat tulisivat olemaan näkyvillä. Ominaisuuksiltaan lista tulisi muistuttamaan usein yhtenä ohjelmoinnin aloitus tehtävänä toteutettavaa tehtävä listaa (to do list). Kuten tehtävälistaankin, tähän listaan pystyttäisiin lisäämään ja poistamaan elementtejä.

Loppujen lopuksi projekti sai teknisesti kaksi listaa; yhden käyttäjän syötteille ja toisen valmiille vaihtoehdoille. Tämä siitä syystä, että kahden erillisen listan tekeminen oli teknisesti helpompaa niiden hieman toisistaan poikkeavan toimintatavan takia. Ensimmäinen lista muodostuu sovelluksessa jo valmiiksi olevista vaihtoehdoista ja on heti näkyvillä käyttäjälle sovelluksen avatessa, kun taas toista listaa ei edes ole vielä olemassa siinä vaiheessa, sillä se muodostuu kokonaan käyttäjän antamista syötteistä. Molemmat listat kuitenkin näytetään samassa paikassa ja näyttävät käyttäjälle vain yhdeltä ainoalta listalta.

Molempien listojen tultua valmiiksi oli aika siirtyä seuraavaan osan toteutukseen – eli nappiin, joka tulisi arpomaan käyttäjälle ruuan. Samaisen napin tehtävä olisi myös pyöräyttää sivulle tulevaa pyörää, mutta ensin keskityttiin vain ruuan arpomiseen listalta.

Arpominen tapahtuu niin, että napin painalluksella sovellus hakee sivulta aiemmin mainitut kaksi listaa – käyttäjän syötteestä sekä valmiista vaihtoehdoista kootun listan. Koska listoja on käytännössä kaksi, ne yhdistetään yhdeksi ja tuloksena tulevasta isosta listasta arvotaan yksi tulos käyttäjälle.

Tämän jälkeen napin painalluksesta puuttui vielä sen toinen tehtävä, eli pyörän pyöräytys. Itse pyörä rakennettiin HTML:än ja CSS:än avulla. Arpomisen sijasta napin painallus vaihdettiin aktivoimaan funktio, joka kutsuisi arpomisen suorittavan funktion sekä pyöräytyksen suorittavan funktion. Tämä muita funktioita kutsuva funktio sai nimekseen startRaffle.

Napin painalluksesta arvonnan aloittava funktio etsii sivulta pyörän ja pyöräyttää sitä satunnaisen määrän kierroksia. Pyörän itse visuaalinen pyörähdys toteutettiin CSS transition-ominaisuuden avulla, sillä pelkkä JavaScript olisi vain töksäyttänyt pyörän arpomaansa kohtaan ilman visuaalista pyörimistä.

Pyörän pyöräyttäminen oli kuitenkin vain visuaalinen efekti, sillä sovellushan osasi arpoa ruuan käyttäjälle ilman pyöräytystäkin. Ongelma siis oli, että arvonnan tuloksena tullut ruoka ilmestyi näkyviin, vaikka pyörä vielä pyöri. Tätä ongelmaa ratkomaan tehtiin startRafflen sisälle setTimeout-metodi, joka kutsuu listoja käsittelevän funktion, kun pyörä on pyörähtänyt eli kuusi sekuntia napin painalluksen jälkeen. Kuvio 10 näyttää strartRaffle-funktion valmiin rakenteen.

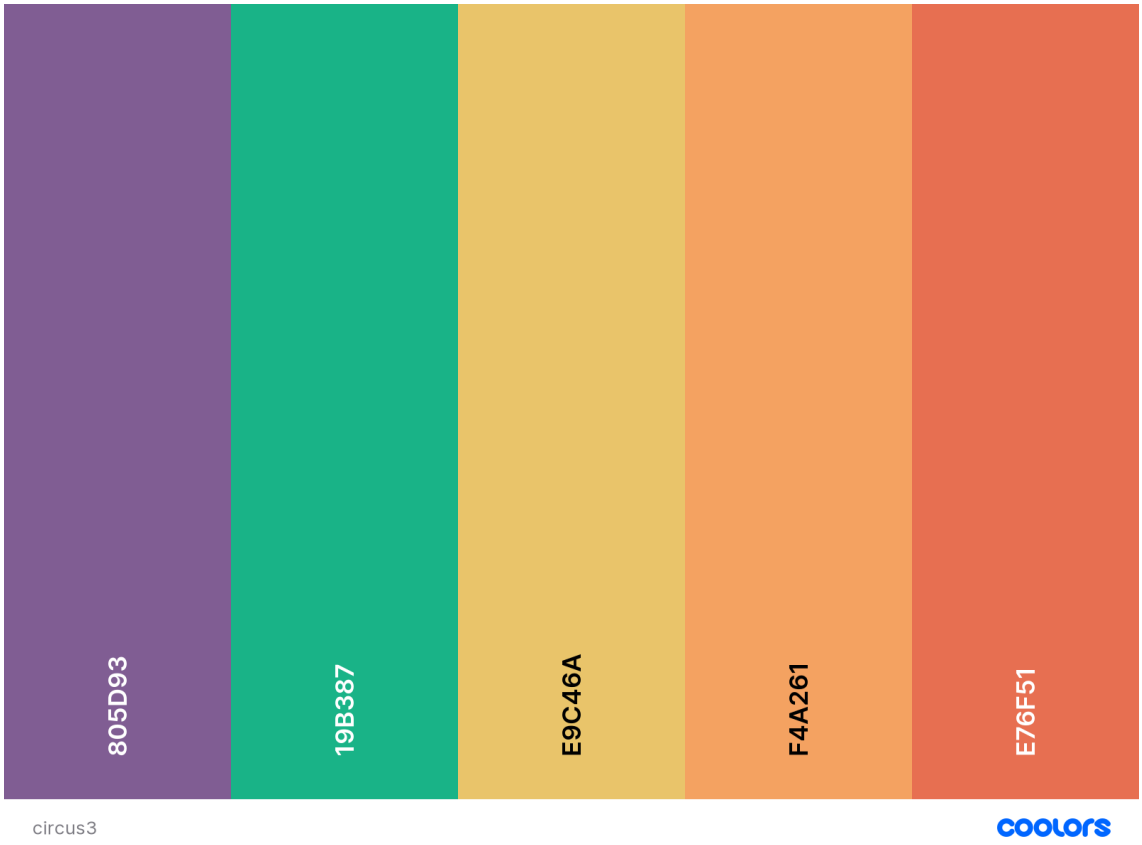
```
function startRaffle() {
    setResult("");
    rotateWheel();

    setTimeout(function () {
        getLists() }, 6000);
}
```

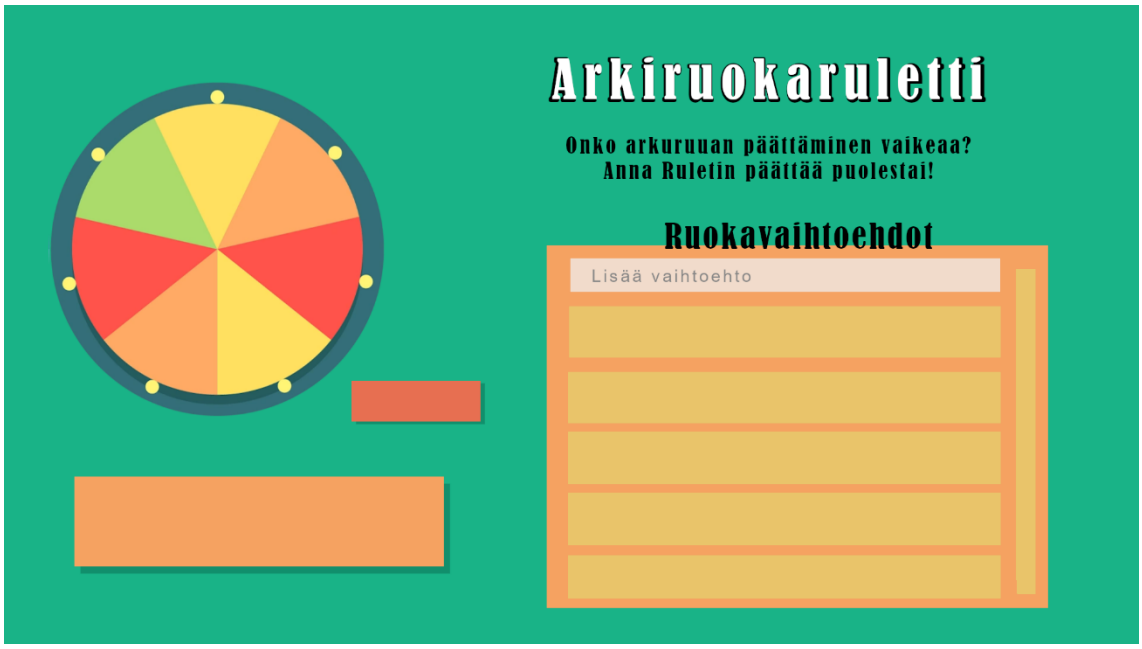
Kuvio 10: Listat hakeva funktio getLists on setTimeout-metodin sisällä, mikä viivästyttää sen suorittamista kuudella sekunnilla.

5.3 Visuaalinen ilme

Sovelluksen toiminnallisten ominaisuuksien valmistuttua oli aika puuttua sen visuaalisen ilmeeseen. Tätä varten oli jo suunnitteluvaiheessa tehty luonnokset sekä värikartta. Valmiin värikartan (kuvio 11) sekä tietokoneella tehdyn luonnoksen (kuvio 12) pohjalta ulkoasua oli helppoa lähteä rakentamaan.



Kuvio 11: Coolors.co-sivulla luotu värikartta



Kuvio 12: Kuvankäsittelyohjelmalla toteutettu suunnitelma sovelluksen ulkoasusta

Mobiili mukautuvuuden toteutukseen sekä elementtien sijoitteluun sivulla käytettiin apuna Bootstrap-nimistä sovelluskehystä. Sen avulla sivun eri elementit saa automaattisesti skaalautumaan näytön leveydelle sopiviksi hyvin helposti.

Viimeisenä loppusilauksena projektiin lisättiin vielä animaatioita animate.css nimisen kirjaston avulla.

6 Tulokset

Kehitystyön tuloksena syntyi yksinkertainen, mutta toimiva verkkosovellus, joka arpoo käyttäjälle ruuan annetusta listasta. Sovellus tekee juuri ne asiat, jotka sen oli suunnitelman mukaan tarkoituskin pystyä tekemään. Kuviossa 13 näkee vielä valmiin projektin käyttöliittymän.

Kuvio 13: Projektin valmis käyttöliittymä

7 Yhteenveto

Opinnäytetyön tavoitteena oli tutustua erityisesti Reactiin, mutta myös muihin sen ohella käytettäviin teknologioihin sekä verkkokehityksen nykytilanteeseen ja trendeihin. Työn puitteissa tutustuttiin etenkin responsiivisiin verkkosivuihin sekä kahden tyyppisiin verkkosovelluksiin.

Teoriaan tutustumisen lisäksi työssä toteutettiin projekti, jonka avulla tutustuttiin Reactin käyttöön myös käytännössä. Projekti onnistui hyvin ilman suurempia vastoinkäymisiä ja sen lopullinen versio täyttää juuri ne kriteerit, jotka sille oli annettu.

Opinnäytetyölle annetut tavoitteet täyttyivät hyvin, teoriaosuuden kirjoittamisesta ja aiheisiin tutustumisesta sai hyvän käsityksen verkkokehittämisen nykytilasta sekä Reactin ja muiden teknologioiden teorioista. Projektin toteutuksesta taas sai puolestaan hyvän käsityksen teknologioiden toiminnasta käytännössä. Oppimista toki vielä riittää, mutta tarkoitus ei ollutkaan vielä päästä erityisen syvälle etenkään itse kehittämiseen.

8 Jatkokehitysehdotukset

Toteutettu projekti on melko yksinkertainen. Yksi selkeä kehityskohde olisi lisätä siihen käyttäjille mahdollisuus kirjautua sisään, jolloin käyttäjän muodostaman listan voisi tallentaa hänen käyttäjätunnukselleen. Näin projekti vastaisi paremmin verkkosovelluksen kriteerejä ja käyttäjäkokemusta saisi parannettua, sillä listaa ei tarvitsisi muokata joka kerta erikseen itselleen mieluisaksi.

Tämä kuitenkin olisi vaatinut palvelinpuolen (backend) kehittämistä sekä mahdollista tietokantaintegraatiota, mistä kumpikaan ei kuulunut opinnäytetyön aiheen piiriin, sillä tavoitteena oli tutustua React-kirjastoon ja sitä käytetään käyttöliittymien rakentamiseen. Lisäksi se olisi pidentänyt itse opinnäytetyön valmistumista etukäteen vaikeasti määriteltävällä ajalla, mikä ei olisi ollut mielekästä. Tavoite kuitenkin oli saada työ valmiiksi syksyn loppuun mennessä ja siitä aikataulusta olisi voitu helposti lipsua, mikäli tietokantojen kanssa olisikin tullut ongelmia. Lisäksi itse opinnäytetyöstä olisi tullut laajempi.

Kirjautumistoiminnallisuuden voisi kuitenkin projektiin lisätä vielä jossakin myöhemmässä vaiheessa, sillä sekin olisi hyväksi ammatilliselle kehitykselle. Lisäksi se olisi kiinnostava pulma ratkaista, eikä välttämättä erityisen hankalakaan toteuttaa.

Kirjautumismahdollisuuden lisäämisen lisäksi projektin ulkonäköä voisi vielä muuttaa, sillä itse tekijä ei pysty varmaan koskaan olemaan siihen täysin tyytyväinen.

Lähteet

Painetut

Esposito D. 2016. Modern Web Development. Washington: Microsoft Press.

Fling B. 2009. Mobile Design and Development. Sebastopol: O'Reilly Media.

Gardner, B. 2011. Responsive Web Design: Enriching the User Experience. Sigma 11. Falls Church: Noblis, 13-19.

Young A., Meck B. & Cantelon M. 2017. Node.js in action. New York: Manning.

Sähköiset

Bar-Gil G. 2020. NPM vs. Yarn: Which Package Manager Should You Choose? WhiteSource Software. Viitattu: 24.9.2021. https://www.whitesourcesoftware.com/free-developer-tools/blog/npm-vs-yarn-which-should-you-choose/

Boduch, A. & Dereks, R. 2020. React and React Native: A complete hands-on guide to modern web and mobile development with React.js. E-kirja. Birmingham: Pact Publishing.

Giridhar C. 2017. How To Simplify Networking In Android: Introducing The Volley HTTP Library. Viitattu: 24.9.2021. https://www.smashingmagazine.com/2017/03/simplify-android-networking-volley-http-library/

Google Search Central. 2021a. SEO starter guide. Viitattu: 12.11.2021. https://developers.google.com/search/docs/beginner/seo-starter-guide?hl=en%2F

Google Search Central. 2021b. Mobile-first indexing best practices. Viitattu: 12.11.2021. https://developers.google.com/search/mobile-sites/mobile-first-indexing

Hooda P. 2020. Difference between npm and yarn. GeeksforGeeks. Viitattu: 24.9.2021. https://www.geeksforgeeks.org/difference-between-npm-and-yarn/

Jay A. 2021. Number of Smartphone and Mobile Phone Users Worldwide in 2021/2022: Demographics, Statistics, Predictions. Viitattu: 2.11.2021. https://financesonline.com/number-of-smartphone-users-worldwide/

Lawson K. 2021. What is a single page application and why do people like them so much? Bloomreach. Viitattu: 12.10.2021. https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html#blog

Levins C. 2020. What Is a Package Manager and Is Your Business Ready for One? Air Sea Containers. Viitattu: 24.9.2021. https://www.airseacontainers.com/blog/what-is-a-package-manager/

Lindström S. 2019. Progressive web application (PWA) yhdistää verkkosivun ja natiivin mobiilisovelluksen parhaat ominaisuudet. Ite wiki. Viitattu: 12.10.2021. https://www.ite-wiki.fi/blog/2019/09/progressive-web-application-pwa-yhdistaa-verkkosivun-ja-natiivin-mobiilisovelluksen-parhaat-ominaisuudet/

Kopecky C. 2020. What is Node.js? A beginner's introduction to JavaScript runtime. Educative. Viitattu: 23.9.2021. https://www.educative.io/blog/what-is-nodejs

Krill P. 2017. NPM or Yarn? Node.js devs pick their package manager. InfoWorld. Viitattu: 12.11.2021. https://www.infoworld.com/article/3172725/npm-or-yarn-nodejs-devs-pick-their-package-manager.html

Kuprenko V. 2021. 8 Web Development Trends Every CTO Should Be Ready for in 2021. Cleveroad. Viitattu: 5.10.2021. https://www.cleveroad.com/blog/web-development-trends

McCurdy N. 2020. Getting Started. Create React App. Viitattu: 15.10.2021. https://create-react-app.dev/docs/getting-started

Morris, S. 2021. Tech 101: What is React JS? Skillcrush. Viitattu: 22.9.2021. https://skillcrush.com/blog/what-is-react-js/#toc

Murphy É. 2015. 8 Years, 4 Major Trends: The Evolution of Websites From 2007 to Now. HubSpot. Viitattu: 1.10.2021. https://blog.hubspot.com/marketing/website-trends-since-2007

Nakazawa C., McKenzie S. & Kyle J. 2016. Yarn: A new package manager for JavaScript. Facebook Engineering. Viitattu: 12.11.2021. https://engineering.fb.com/2016/10/11/web/yarn-a-new-package-manager-for-javascript/

Nehra M. 2020. The Evolution of Web Development & Its Modern Trends. Decipher Zone. Viitattu: 1.10.2021. https://www.decipherzone.com/blog-detail/evolution-web-development

Newman J. 2021. Why some developers are avoiding app store headaches by going web-only. FastCompany. Viitattu: 3.11.2021. https://www.fastcompany.com/90623905/ios-web-apps

Npm trends. 2021. Npm vs Yarn. Viitattu: 12.11.2021. https://www.npmtrends.com/npm-vs-yarn

Occihno T. 2015. React.js Conf 2015 Keynote - Introducing React Native. Konferenssi tallenne. Meta developers. Viitattu 24.11.2021. https://youtu.be/KVZ-P-ZI6W4

Raja. 2018. What Is Responsive Web Design and Why Is It Important? Viitattu: 1.10.2021. https://www.dotcominfoway.com/blog/importance-of-responsive-web-design/

React. 2021a. Introducing JSX. Viitattu: 23.11.2021. https://reactjs.org/docs/introducing-jsx.html

React. 2021b. React - A JavaScript library for building user interfaces. Viitattu: 15.10.2021. https://reactjs.org/

Starikov M. 2019. Most Used JavaScript Frameworks for Quick Software Development: Which to Choose. Viitattu: 22.9.2021. https://stfalcon.com/en/blog/post/javascript-frameworks-for-software-development

Tandel S. & A. Jamadar. 2018. Impact of Progressive Web Apps on Web App Development. International Journal of Innovative Research in Science, Engineering and Technology, 7 (9), 9439-9444. Viitattu: 25.11.2021. http://www.ijirset.com/upload/2018/september/22_Design.pdf

Vu L. 2019. PWA vs SPA: Alike but Different. SimiCart. Viitattu: 12.10.2021. https://www.simicart.com/blog/pwa-vs-spa/

Kuviot