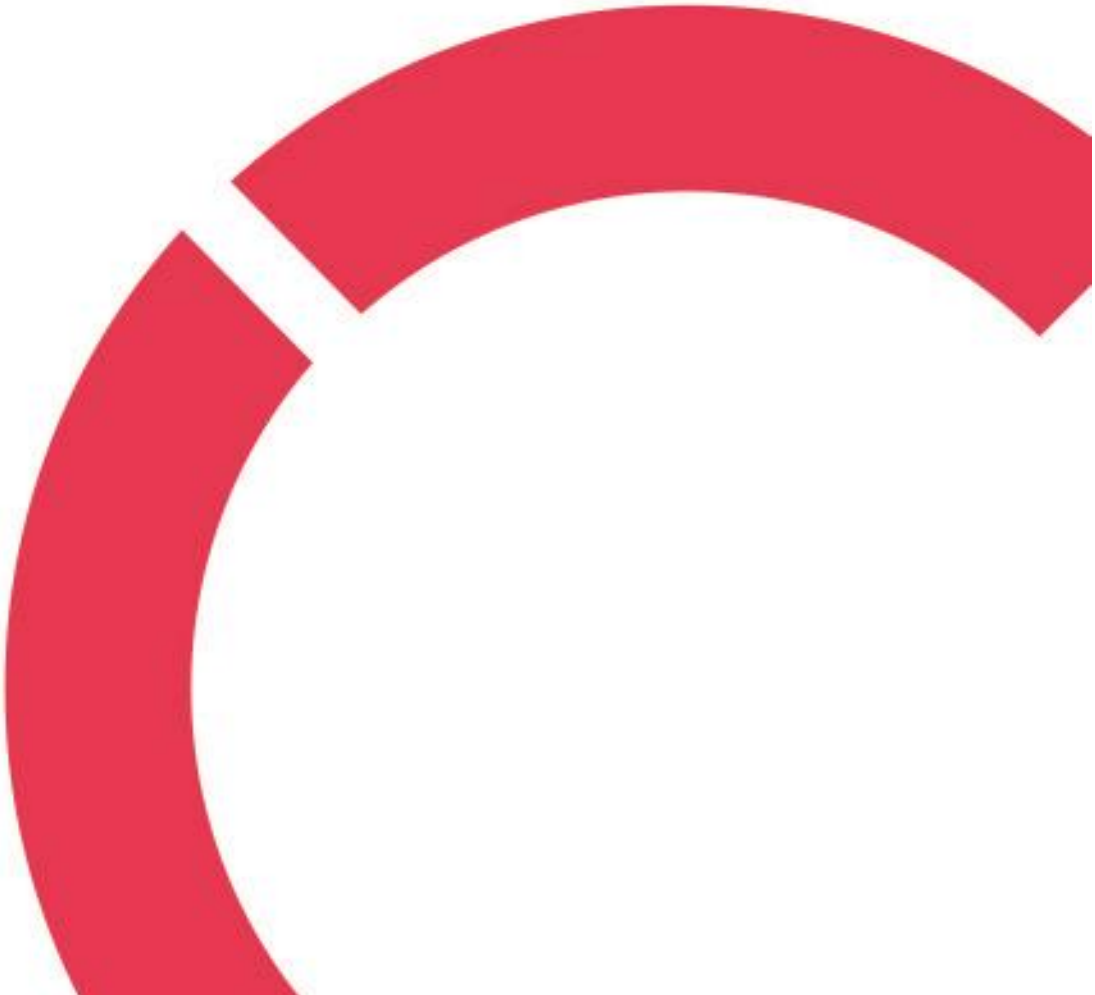
Jori Pulkkinen

Yhtä web-sivua käyttävän web-sovelluksen kehittäminen Reactilla

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutus

Kesäkuu 2023



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Kesäkuu 2023	**Tekijä/tekijät** Jori Pulkkinen
Koulutus Tieto- ja viestintätekniikka		☒ AMK ☐ YAMK
Työn nimi Yhtä web-sivua käyttävän web-sovelluksen kehittäminen Reactilla		
Työn ohjaaja Jari Isohanni		**Sivumäärä** 31
Työelämäohjaaja -		

Opinnäytetyössä tavoitteena on yhtä web-sivua käyttävän web-sovelluksen kehittäminen Reactilla. Työssä selvitettiin yhtä web-sivua käyttävän web-sovelluksen hyödyt ja haitat. Web-sivusto lataa kaiken sisällön yhdellä kerralla, jotta loppukäyttäjä voi selata sivustoa nopealla vasteajalla. Haittoja voivat olla elementtien päivittyminen paikallisesti, jos tietokoneen teho on rajallinen. Dynaaminen ja yhtä web-sivua käyttävä web-sovellus eroavat niin, että dynaaminen sivusto lähettää pyynnön palvelimelle ja päivittää koko sivun. SPA (Single-Page Application) on nopeampi kuin monen sivun (MPA Multiple-Page Application) web-sovelluksen käyttäminen.

Asiasanat
MVC, MPA, React, SPA, web-sovellus

ABSTRACT

Centria University of Applied Sciences	**Date** June 2023	**Author** Jori Pulkkinen
Degree programme Information technology		
Name of thesis Developing Single-Page Application (SPA) with React		
Centria supervisor Jari Isohanni		**Pages** 31
Instructor representing commissioning institution or company -		

Subject of the thesis is developing single-page application (SPA) with React, and point out, which are the advantages by SPA. SPA loads the entire content of the website, so end user´s experiences fast response time when browsing website. If the CPU`s (central processing unit) capability is limited, updating elements can lead to unresponsiveness or even end up crashing the application. The difference between dynamic website, and SPA is, that dynamic web updates the content by sending a request to the server and refreshing the whole webpage.

Key words
MVC, MPA, React, SPA, web application

KÄSITTEIDEN MÄÄRITTELY

DOM	Document Object Model. Käytetään kuvaamaan HTML-tiedoston mallia puurakenteena
HTML	Hypertext Markup Language. Kieli, jota käytetään verkkosivujen kehittämisessä
JSON	Tiedon käsittelyyn tarkoitettu tiedostomuoto.
JSX	Javascript XML. JavaScript-kielen lisäosa, joka mahdollistaa HTML-kielen hyödyntämisen JavaScript-tiedostossa
MPA	Multi-Page Application
React	Sovellusten käyttöliittymien kehittämiseen tarkoitettu JavaScript-kirjasto
SPA	Single-Page Application
TypeScript	TypeScript on Microsoftin kehittämä ohjelmointikieli

TIIVISTELMÄ
ABSTRACT
SISÄLLYS

1 JOHDANTO

Sovelluskehityksen nopean edistymisen myötä internetin käyttäjät haluavat tehokkaampaa käyttömukavuutta. Opinnäytetyön tavoitteena on selvittää, miten yhtä web-sivua käyttävä web-sovellus SPA (Single-Page Application) toimii. Työssä esitetään prosessi, kuinka SPA rakennetaan Reactilla.

React on avoimen lähdekoodin Javascript-kirjasto. React soveltuu parhaiten verkkosivun käyttöliittymän kehittämiseen, mutta se sopii myös sovelluksen logiikan rakentamiseen. Yhtä web-sivua käyttävän web-sovelluksen ideana on ladata tiedostot paikallisesti selaimeen web-sivun selailun käyttömukavuuden parantamiseksi. Kaikkiin tarkoituksiin yhtä web-sivua käyttävää web-sovellusta ei ole optimaalista käyttää. Esimerkiksi hakukoneen luomisessa dynaaminen verkkosivu on parempi valinta, koska prosessin käsittely tapahtuu palvelimella ja se mahdollistaa suuremman määrän datan käsittelyä. Yhtä web-sivua käyttävä web-sovellus on soveltuva, mikäli pyyntö palvelimelle ei palauta suurta datan määrää. SPAn hyödyllisyys voidaan todeta sen käytöstä tunnetuilla sivustoilla, kuten Facebook, Paypal ja Gmail. Reactin käytön hyödyt korostuvat laajemman kokonaisuuden rakentamisessa. Kehittäjille suunnatun kyselyn perusteella Reactin suosio on vain kasvanut käyttöliittymien kehittämisessä. React on julkaistu jo vuonna 2013, ja sen suosion kasvu ollut suurta, vaikka muitakin vaihtoehtoja on tullut tarjolle. React on käyttäjäystävällinen kehittäjälle, ja aloittelijatkin voivat ottaa sen helposti käyttöön. (Krotoff, 2023.)

2 TAVOITTEET

Opinnäytetyön tavoitteena oli selvittää, minkälaisia eroja yhtä web-sivua käyttävän web-sovelluksen (SPA, Single-Page Application) ja monisivuisen web-sovelluksen (MPA, Multi-Page Application) välillä loppukäyttäjä havaitsee. Sen lisäksi selvitettiin, miten käyttö muuttuu hallinnan tapahtuessa elementeissä paikallisesti ja verrata sitä palvelimen toimesta tapahtuviin muutoksiin. Työssä tutkittiin SPAn arkkitehtuuria ja toimintaperiaatetta sekä toteutettiin yksinkertainen SPA-sovellus.

2.1 Miksi React valittiin?

React valittiin opinnäytetyön yhtä web-sivua käyttävän web-sovelluksen kehittämiseen, koska se on Facebookin kehittämä moderni ja suosittu JavaScript-kehys. React soveltuu monenkokoisiin ja -tasoisiin projekteihin harrastelijasta ammattilaisiin sekä useamman tiimin hankkeisiin. React on sovellettavissa helposti myös isoihin ja monimutkaisiin kokonaisuuksiin, sillä kokonaisuus on helppo pilkkoa Reactilla pienempiin osiin. React on tarkoitettu käyttöliittymän kehittämiseen, mutta React Nativella voidaan toteuttaa sovelluksia mobiilipuolelle. Merkittävin hyöty Reactista saadaan monipuolisesti käytettävien komponenttien avulla, sillä komponentit ovat uudelleen käytettäviä. Komponenttikirjastoja voidaan tarvittaessa hyödyntää, kun halutaan tiettyä toimivuutta tai estetiikkaa, esimerkiksi react router, redux ja fluent UI ovat suosittuja komponenttikirjastoja. Fluent UI on Microsoft kehitystiimin luoma kirjasto, jolla saavutetaan samankaltainen ulkoasu kuin Microsoft Office- tuotteissa. React router on sivuston selailua helpottava kirjasto, ja sitä voidaan käyttää muuan muassa SPAn reitityksessä. Reduxin avulla pystytään kontrolloimaan sovelluksen eri tiloja ja ilmoittaa tapahtuneista virheistä. Reduxin toiminnasta kerrotaan tarkemmin luvussa 3.4. (Technostacks, 2023.)

2.2 Työn tavoitteet

Työn tavoitteena oli selvittää, kuinka SPA rakennetaan Reactilla ja miten se käytännössä toimii. Lisäksi selvitettiin SPAn hyötyjä ja eroja muihin sivustorakenteisiin verrattuna. React hyödyntää JSX-tiedostoja (JavaScript XML). Sen etuna on se, että JavaScript-tiedostoissa voidaan käyttää HTML-elementtejä ja

komponentteja voidaan muokata kätevämmin halutun näköisiksi. Tehokkaalla ja oikeaoppisella sovelluksen suunnittelulla mahdollistetaan useampien käyttäjien samanaikaiset pyynnöt. Lähetetyt pyynnöt tulevat palvelimelle, joka hakee pyydetyn tiedon yleensä tietokannasta. Palvelin lähettää vastauksen asiakkaalle, joka käsittelee vastauksen. Pyyntöjen käsittely on nopeaa, sillä resursseja on käytettävissä enemmän eikä sivua tarvitse ladata joka kerta uudelleen. JavaScript-kirjastoista tarvittavia ominaisuuksia tuodaan sovelluskoodiin import-komennolla.

Lopputyön tavoitteena oli laatia sivusto, joka esittää SPAn hyödyt käytännöllisellä tasolla. React websovellus hyödyntää komponentteja, joista sivusto koostu. Tavoitteena oli koota sivusto kutsumalla komponentteja ja saada aikaan halutun näköinen lopputulos. Eri komponentteja pystytään kutsumaan uudelleen eikä uutta sivua tarvitse koodata alusta saakka, vaan se on mahdollista rakentaa valmiista paloista. SPAn luomisessa hyödynnetään loppukäyttäjän reititystä, sillä se ei lataa dataa palvelimelta, vaan käyttää paikallisesti ladattua tietoa. Reitityksen avulla linkkiä painettaessa DOM (Document Object Model) muodostaa komponenteista uuden näkymän.

3 REACTIN OLEELLISET KÄSITTEET

Tässä luvussa käsitellään komponentteja, reititystä, DOMia ja MVC-arkkitehtuuria, jotka ovat oleellisia Reactin ja opinnäytetyön kokonaisuuden hahmottamiseksi.

3.1 Reitityksen toimintaperiaate

Sovelluksen suunnittelussa perehdyttiin yhtä web-sivua käyttävän web-sovelluksen toimintaperiaatteeseen ja sen toteutukseen. Kun sovelluspohja on luotu, luodaan reititys, joka tarvitsee react-router-dom-javascript kirjastoa toimiakseen, ja se ladataan komennolla (npm install react-router-dom). Reitityksen toiminta perustuu siihen, miten se reagoi käyttäjän tekemiin muutoksiin. Kun käyttäjä painaa linkkiä, DOM renderöi reitityksen avulla halutun sivun käyttäjän näkyville.
(React Router, n.d.)

```
2   import { BrowserRouter } from 'react-router-dom'
```

KUVA 1. App.js-tiedostoon tuodaan BrowserRouter react-router-dom-kirjastosta.

Kuvassa 1 on esitelty kirjastojen tuominen koodiin. Tuomalla koodiin kirjastoja voidaan hyödyntää valmiiksi olemassa olevia työkaluja eikä kaikkea ei tarvitse kirjoittaa alusta alkaen.

```
8       <BrowserRouter>
9       <Etusivu/>
10      </BrowserRouter>
```

KUVA 2. App.js-tiedostossa kutsutaan etusivu-komponenttia BrowserRouter-elementin sisällä.

Kuvassa 2 on elementti, jonka sisällä on kutsuttu etusivu-komponenttia. Elementtiä kutsuttaessa se voidaan sulkea ennen sulkevaa hakasulkua, jos sen sisälle ei tarvitse määritellä tai laatia mitään kutsua.

```
import {Switch, Route, Redirect,withRouter } from 'react-router-dom'
```

KUVA 3. Etusivu-komponenttiin tuotavat komponentit react-router-dom-kirjastosta.

Kuvassa 3 esitellään React-router-dom-kirjastosta tuodaan haluttavat ominaisuudet, jotka ovat tärkeitä sivuston toimivuuden kannalta.

```
11      <Switch>
12    <Route path='/etusivu' component={Koti}/>
13    <Route path='/tietoja' component={Tietoja}/>
14    <Route path='/kuvia' component={Kuvia}/>
15    <Redirect to='/etusivu'/>
16      </Switch>
```

KUVA 4. Switch-elementin sisällä määritetyt reitit etusivu-komponentissa.

Kuvassa 4 on määritellyt reitit. Route-elementillä määritetään reitit, jotta sivusto löytää oikeat komponentit. Redirect ohjaa käyttäjän takaisin etusivu-komponenttiin, jos syötettyä polkua ei löydy. Withrouter-komponenttia käytetään uudelleenohjauksen määrittelyssä.

```
21    export default withRouter(Etusivu);
```

KUVA 5. WithRouter-elementin määrittely.

Kuvassa 5 on WithRouter-elementin sisälle määritelty komponentti, jonne käyttäjä uudelleenohjataan.

3.2 Dynaaminen reititys

Dynaaminen reititys tarkoittaa reittejä, joissa on muuttujia. Tällaisia reittejä voidaan tarvita, kun luodaan sivuja esimerkiksi tuotteiden esittelyyn. On helpompaa luoda yksi dynaaminen sivu, johon haetaan tiedot tietokannasta, kuin ladata useampi sivu erikseen. Linkkiä painettaessa sivu luodaan halutuilla tiedoilla. Dynaaminen sivu luodaan sivupohjaa hyödyntäen ja tiedot sivuun haetaan usein tietokannasta. Next.js on tehnyt helpoksi dynaamisten reittien luomisen. Dynaaminen reitin tiedostonimi kääritään hakasulkeisiin esimerkiksi [Id].js.

```
import Link from "next/link";
```

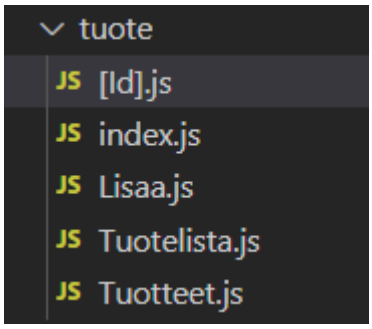
KUVA 6. Nextlink-komponenttikirjaston tuominen.

Kuvassa 6 on tuotu NextLink-kirjasto. Nextlink on Next.js:ssä käytettävä komponentti, jolla määritetään ennalta reitit helpottamaan reitityksen työstämistä.

```
<Link href={`${linkki + tuotteet.id}`} >Lisätietoja</Link></button>
```

KUVA 7. Linkin määrittely.

Kuvassa 7 on Linkki-elementti. Sitä käytetään komponentissa ja sille annetaan haluttu arvo.

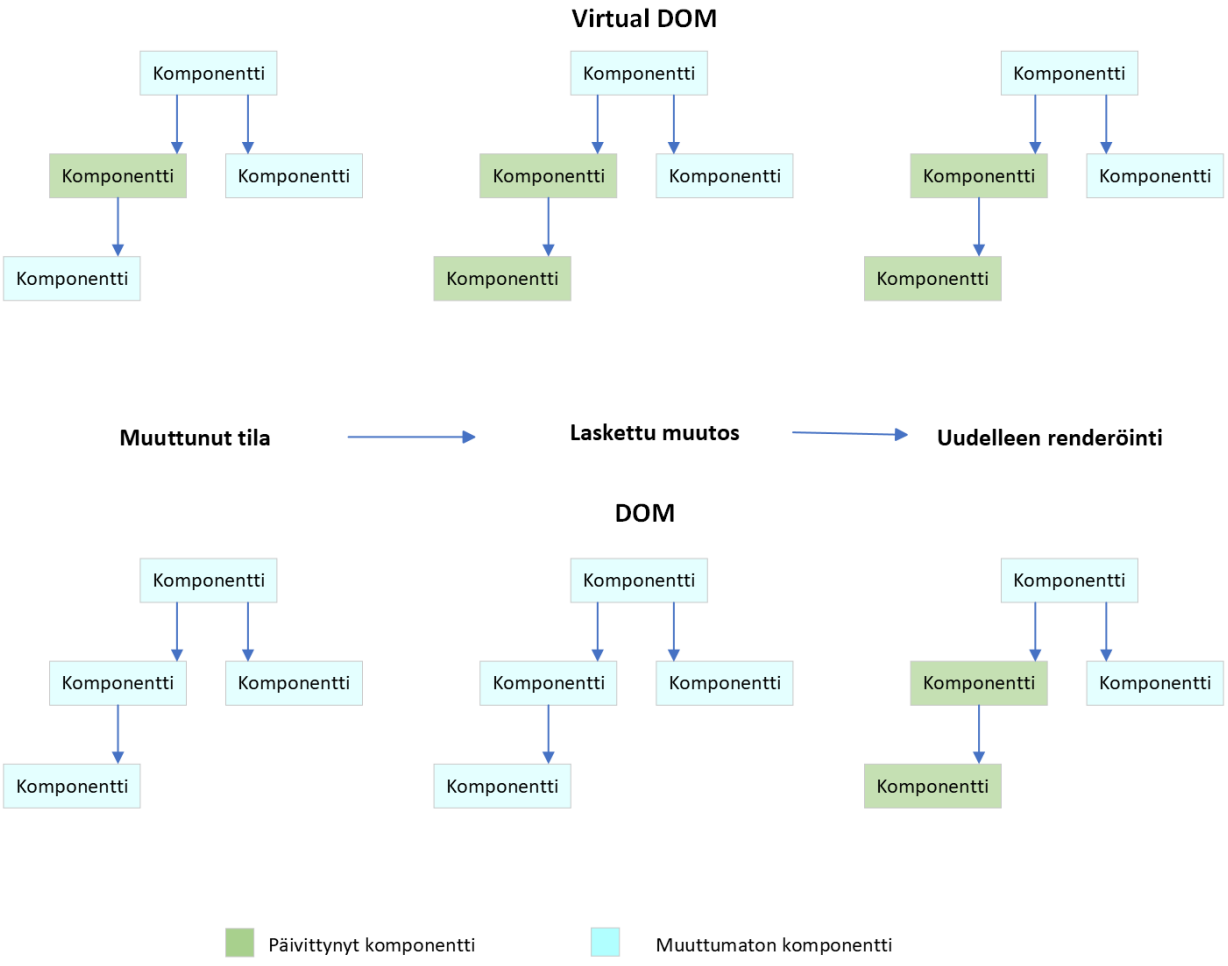


KUVA 8. Dynaamisen reitin määrittely.

Kuvassa 8 on Dynaaminen reitti luotu tuote-nimisen polun alle. Reitit on luotuja helpottamaan käyttäjän selailukokemusta. Jos sivustolla on paljon sisältöä, useamman reitin luominen selkeyttää sivujen luomista ja on myös mieleinen kokemus loppukäyttäjän kannalta.

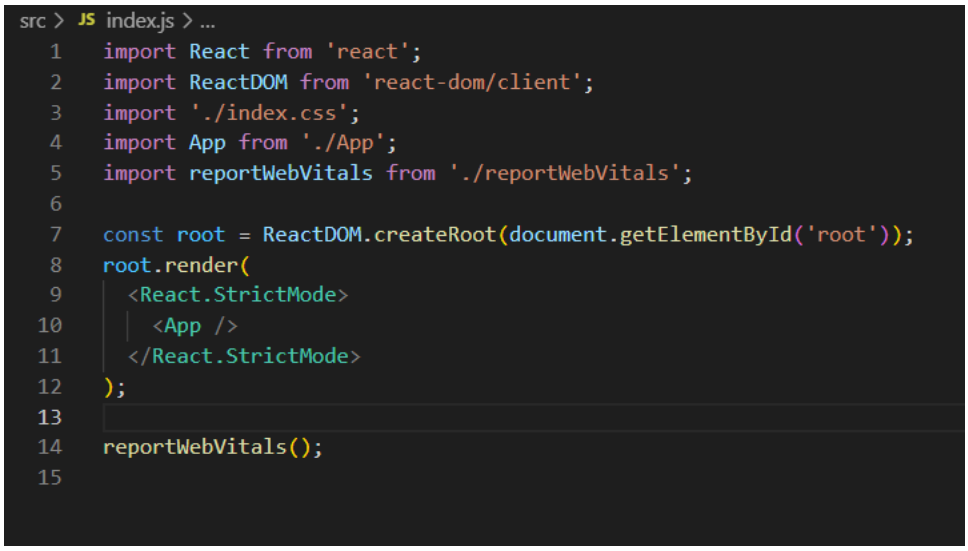
3.3 DOM (Virtual Document Object Model)

DOM (Document Object Model) on malli, jonka päälle sivusto rakennetaan. Moderneissa sivuissa DOM ei pysty päivittymään nopeasti, koska se lataa koko näkymän joka kerta uudelleen päivittymisen yhteydessä. DOMin avuksi on Reactiin kehitetty React DOM tai toiselta nimeltään Virtual DOM. Virtual DOM teettää identtisen mallin DOMista ja käyttää sitä vertailukohtana. (Hyeny, 2021.)



KUVA 9. Virtual DOM ja DOM toimintaperiaate (Hyeny, 2021) muokattu.

Virtual DOM toimii sillä periaatteella, että se seuraa tapahtuvia muutoksia. Jos Virtual DOM huomaa muutoksen, se päivitetään mahdollisimman vähäisillä muutoksilla. Tässä Virtual DOM käyttää erotus-algoritmia, siinä luetaan molemmat tiedostot ja algoritmi laskee, miten haluttuun tulokseen päästään mahdollisimman pienillä muutoksilla kuvassa 9 esitetyllä tavalla. (Hyeny, 2021.)



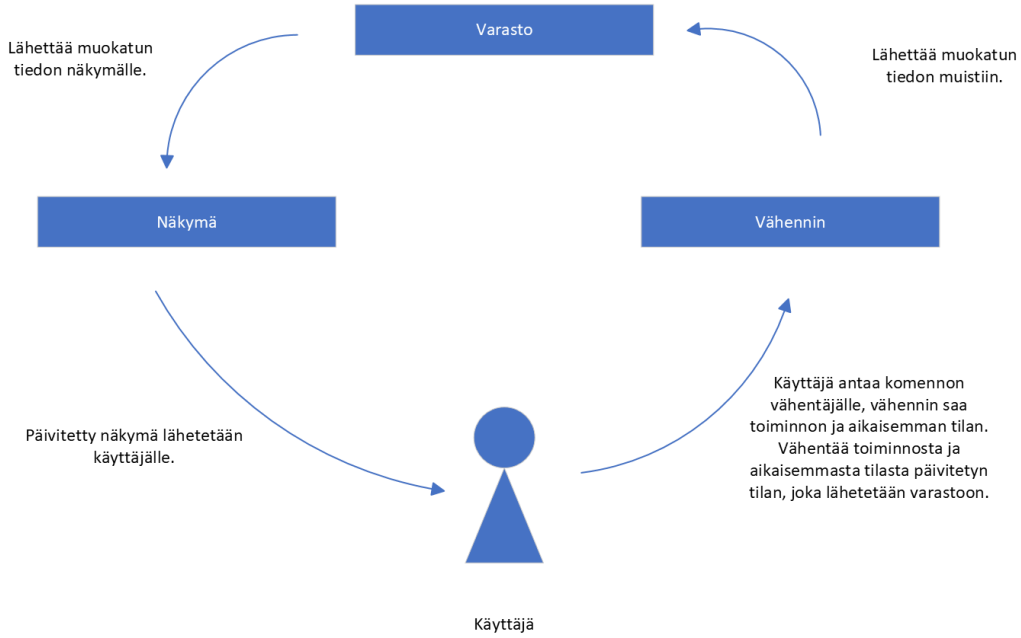
KUVA 10. Index.js-tiedosto

Virtual DOM uudelleen renderöi elementit. Virtual DOM päivittää vain elementit, joissa on tapahtunut muutos. Tämä nopeuttaa sivuston toimintaa. Komponentti kääritään ReactDOM render metodin sisään. (Immukul, 2023.) Kun tapahtumat ovat päivitetty, Virtual DOMissa tiedot annetaan DOMille, jotta selaimen näkymä päivittyy (Hyeny, 2021). Kuvassa 10 on näytetty, kuinka Virtual DOMia käytetään koodissa.

3.4 MVC-arkkitehtuuri

Reactissa voidaan hyödyntää MVC (Model-View-Controller) -arkkitehtuurimallia, joka on kehitetty jo 1970-luvun lopussa ja jota on sovellettu vuosien mittaan lukuisiin eri teknologioihin (Walter, 2008). MVC-mallin hyöty saadaan parhaiten näkyviin suuremmissa projekteissa, joissa syntyy paljon koodia, ja kokonaisuuden hallinta helpottuu, kun tiedetään toteutettavien moduuleiden looginen kehys. Myös ohjelmien elinkaaren kannalta oleellinen ylläpidettävyys paranee mallin myötä.

Mallissa Model edustaa tietorakennetta ja siihen liittyvää logiikkaa, View huolehtii käyttäjälle esitettävästä näkymästä, ja Controller hallitsee muutoksia.
MVC-malli helpottaa useamman henkilön työskentelyä yhden projektin parissa, koska tehtäviä voidaan jakaa mallin perusteella ja monimutkainen liiketoimintalogiikka voidaan näin pilkkoa helpommin hallittaviin osiin.
React -kehityksessä React itsessään muodostaa View -osuuden rakenteesta, koko mallin luomiseen tarvitaan Flux tai Redux datan kulkua varten. (Deutsch, 2017.)



KUVA 11. MVC:n toimintaperiaate Reactissa, (Mattiazzi, 2021) muokattu.

Reactissa käytettävä MVC-arkkitehtuuri on hieman erilainen kuin alkuperäinen malli. Suurin ero on Reactin yksisuuntaisuus, kun alkuperäisessä mallissa sallitaan kaksisuuntainen liikenne osien välillä.
Reactin MVC-mallissa Control on Reducer eli vähennin ja kaikki vaaditut muutokset toteutetaan tässä vaiheessa. Model on Storage eli varasto, jonne lähetetään päivitetty tila. Näkymä on samanniminen kuin alkuperäisessä MVC-arkkitehtuurissa ja toimii samalla periaatteella kuin alkuperäinen. Käyttäjän näkymä päivitetään tehtyjen muutosten perusteella kuvan 11 mukaisesti. (Mattiazzi, 2021.)

3.5 JSX (JavaScript)

JSX eli JavaScript xml on tiedostomuoto, jonka avulla pystytään sisällyttämään html-elementtejä Javascript-koodiin. JavaScript ei suoraan ymmärrä html-tageja, joten jsx käy läpi tiedoston etsien kaikki html-elementit. Tämän jälkeen se luo React-elementin, jonka sisällä html-elementti luodaan. JSX ehkäisee tietoturvauhkia. Ennen renderöintiä arvot on tarkistettu ulkopuolelta syötetyn koodin varalta. (React, 2023.)
XSS eli Cross Site Scripting-hyökkäys onnistuessaan saattaa vuotaa esimerkiksi käyttäjän henkilökohtaisia tietoja (KirstenS, n.d).

3.6 TypeScript

TypeScript on Microsoftin kehittämä ohjelmointikieli, jota käytetään Reactissa. TypeScript on hyvin pitkälti sama kieli kuin JavaScript ja kääntyy samana kielenä. TypeScript on ikään kuin paranneltu versio JavaScriptistä, ja siitä on muun muassa helpompi löytää virheet koodista. JavaScript ei palauta virhettä, vaan palauttaa virheellisen arvon, kun TypeScript palauttaa virhekoodin. TypeScriptillä on parempi ylläpito, tuki on aktiivisempaa ja yhteensopivuusongelmia on JavaScriptiä vähemmän. (Finnegan, 2019.)

3.7 Komponentit

Reactissa yleisesti käytettyjä komponenttikirjastoja ovat muun muassa Material UI, React 360 ja React Motion. Material UI on kehittäjien arvostama komponenttikirjasto graafisen suunnitteluun ja elementtien muotoilemiseen. React 360 luo sivuille kolmiulotteisen maailman, jota voidaan hyödyntää virtuaalitodellisuuden toteuttamisessa. React Motion auttaa komponenttien animoimisessa ja antaa sivuille modernia tuntua. (Technostacks, 2023.)

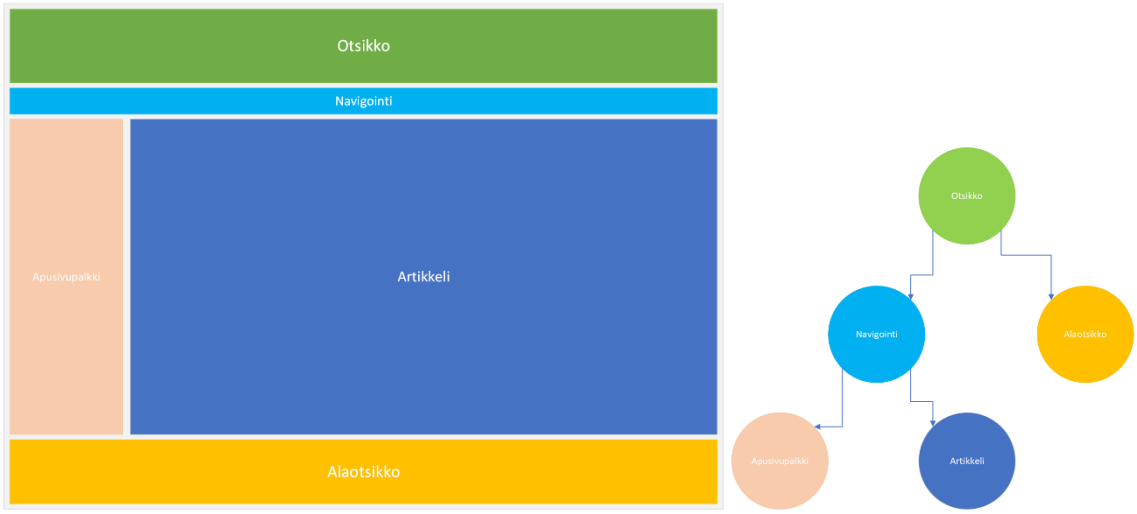
3.8 Yleisesti käytettyjä komponenttikirjastoja

Taulukossa 1 esitellään yleisimpiä komponenttikirjastoja. Komponenttikirjastojen käyttötarkoitukset eriävät toisistaan ja niitä on kehittäjille monipuolisesti saatavilla. Tavoitteena on esitellä lukijalle kokonaisuus hyödyllisistä työkaluista, joita on saatavilla web-sovelluksen kehittämisessä.

TAULUKKO 1. Yleisimpiä komponenttikirjastoja (Technostacks, 2023)

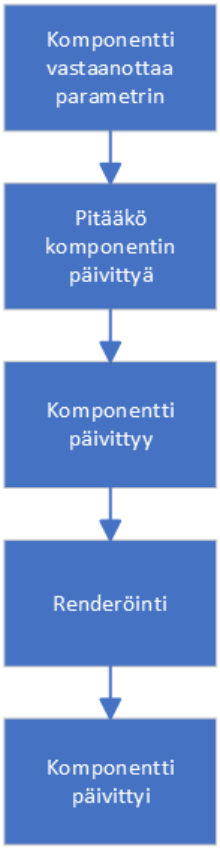
Kirjasto	**Käyttötarkoitus**	**URL**
Material UI	Tyylityökalu	https://mui.com/core/
React router	Reititystyökalu	https://reactrouter.com/en/main
Redux	Sovellusten tilanhallinta-työkalu	https://redux.js.org/
Ant Design	Tyylityökalu	https://ant.design/
React 360	VR-työkalu	https://www.npmjs.com/package/react-360
Semantic UI React	Tyylityökalu	https://react.semantic-ui.com/

Esimerkki komponenttien asettelusta sovelluksessa ja hierarkia näkymälle.



KUVA 12. SPAn komponenttien esimerkkiasettelu ja hierarkia, (Mahra, 2020) muokattu.

Osa komponenteista ovat tilattomia ja toiset tilallisissa komponentteja, tilallisissa käytetään setState()-metodia. Kuvassa 12 esitellään SPAn komponenttien esimerkkiasettelu. Hyötynä tilan käytössä on se, että sitä voidaan käsitellä komponentin sisällä. Tilattomissa voidaan käyttää Props-metodia. Props voidaan välittää komponentille, mutta arvoja voidaan vain lukea komponentissa.

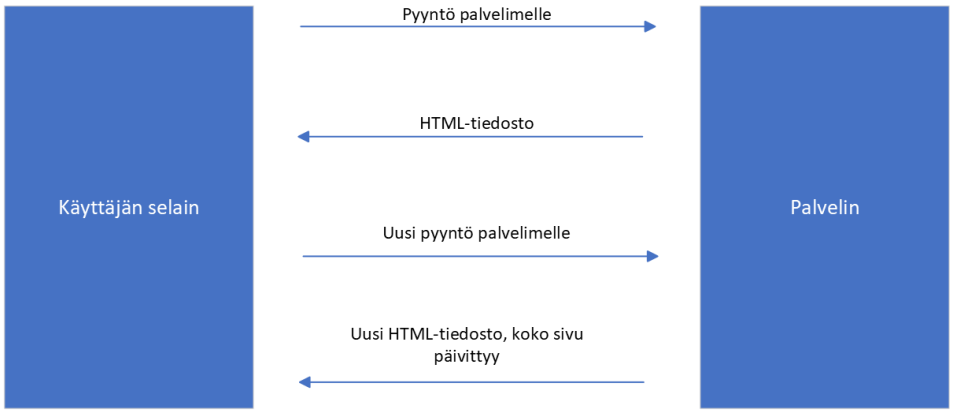


KUVA 13. SPAn komponentin tila eri vaiheissa, (Fedosejev 2015, 76) muokattu.

Fedosejev (2015, 76) kirjoittaa kirjassaan komponentin päivitysmetodeista tapahtumajärjestyksessä ja ensimmäisessä vaiheessa komponentti vastaanottaa Props-metodin avulla arvon. Tätä arvoa voidaan käyttää setState()-lauseen avulla. Seuraavassa vaiheessa määritellään, muuttaako asetettu arvo komponentissa mitään. Kolmannessa vaiheessa annetaan arvojen argumentit DOMille. Jos arvoja muuttaa, DOM renderöi HTML-elementit uudelleen. Viimeisessä vaiheessa tiedotetaan aiemmin saadut arvot. Näitä voidaan hyödyntää ja todentaa, että komponentti päivittyi. (Fedosejev 2015, 76.)
SPAn komponenttien eri vaiheiden tilat etenevät kuvan 13 mukaisesti.

4 MPA (MULTI-PAGE APPLICATION)

MPA (Multi-Page Application) on perinteisempi tapa luoda sivusto. Muutoksen tapahtuessa sivustolla pyyntö lähetetään palvelimelle ja näkymä päivitetään. Jos toteutettavan sivuston logiikka on monimutkainen tai tarvitaan suuria määriä tiedonhakua, MPA on todennäköisesti parempi valinta.
MPA-toteutuksessa käytetään tyypillisesti vakiintuneita teknisiä alustoja, joissa esimerkiksi tietoturva on sisäänrakennettuna. Esimerkiksi palvelimilla voidaan käyttää tietokantaratkaisuja, kuten Microsoft SQL Server, joihin muun muassa valtuuskäsittely on perusominaisuutena. (ADK Group, projekt202`s NE Region, 2022.)

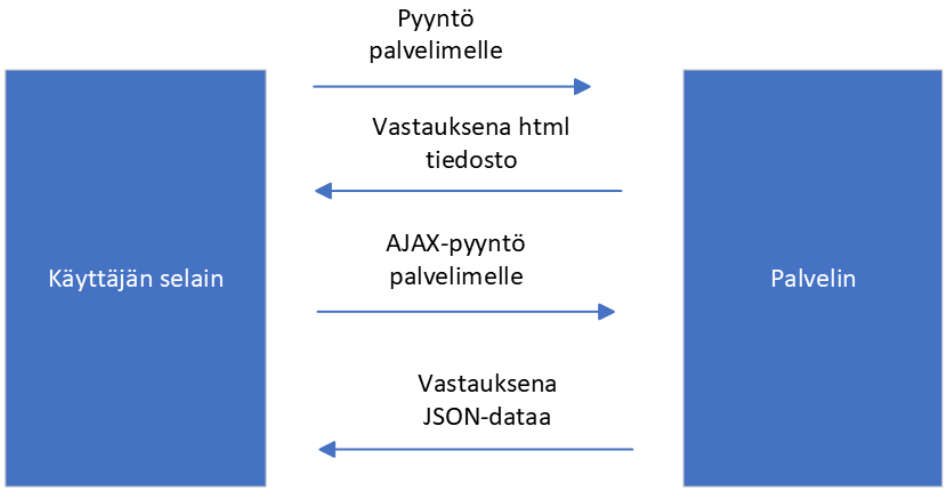


KUVA 14. MPA-toimintamalli, (Agilie 2022) muokattu.

MPA-toteutuksen käyttäjäkokemus ei ole välttämättä yhtä sulava kuin SPAssa johtuen useampien sivujen latauksista. Myös MPA-sivuston soveltuvuudessa mobiililaitteille saattaa olla enemmän haasteita kuin SPAhan verrattuna. Tyypillisesti MPA-sivuston kehittäminen on työläämpää ja vaatii enemmän ratkaisujen suunnittelua kuin SPA-toteutuksessa. (ADK Group, projekt202`s NE Region, 2022.) MPAn toimintamalli kuvassa 14.

5 SPA (SINGLE-PAGE APPLICATION)

Opinnäytetyössä selvitettiin, mitä hyötyjä SPAn käytöstä on ja miten SPA eroaa muista sivustorakenteista. Staattisella verkkosivulla html-dokumentissa ei ole muuttuvia elementtejä, ja tämän tyyppinen verkkosivu sopii henkilökohtaiseksi aloitusverkkosivuksi. Dynaamisessa verkkosivustossa on päivittyviä elementtejä. Erona SPAn ja dynaamisen sivuston välillä on se, että sivuston päivittyessä dynaamisessa sivustossa palvelin tekee raskaan työn ja lataa koko sivun uudelleen. SPAssa tämä tapahtuu selaimessa ja selain päivittää vain tarvittavan osan sivua. Tämä näyttäytyy käyttäjälle sivuston sujuvana toimintana. Oleellinen SPAn ominaisuus on komponenttien käyttäminen, ne ovat uudelleenkäytettäviä web-sivun rakennuspalikoita. Komponentteja pystytään kutsumaan uudestaan useisiin sivuihin, ja näin saadaan helposti sivut näyttämään samalla pohjalla tehtynä. Useampaa web-sivua käyttävän web-sovelluksen (MPA) verrattuna SPAn käyttö on nopeampaa. Kun SPAn sivu on ladattu, se ei kuormita paljon verkkoa. Loppukäyttäjälle on hyvin ratkaisevaa sivuston käyttökokemus, siksi SPA nopealla vasteajalla voi olla parempi ratkaisu asiakkaan ostopäätöksen kannalta, esimerkiksi verkkokaupan tapauksessa. (Joshi, 2023.)



KUVA 15. SPAn toimintaperiaate, (Sharma, 2021) muokattu.

SPA-sovelluksessa voidaan käyttää kaikille sivuille yhteistä alkulatauksessa muodostettua puskuroitua tietovarastoa, joka helpottaa ohjelmointityössä ja tiedot ovat käytettävissä jopa offline käyttötapauksessa, kun verkkoa ei ole saatavilla (ADK Group, projekt202`s NE Region, 2022). SPAn toimintaperiaate kuvassa 15.

SPA toimii JavaScript-kirjastolla, joten tätä tukeva ominaisuus pitää olla päällä sivun toimimiseksi. Ilman kehittäjän muistinhallintaa Javascript saattaa aiheuttaa muistivuotoja. Muistivuotoja tyypillisesti tapahtuu ohjelmaan koodiin kirjoitetun virheen johdosta, muistia on osoitettu virheellisesti käytettäväksi tai muisti on unohdettu vapauttaa. (GeeksforGeeks, 2022.)
Paikallisesti selaimella suoritettavassa Javacriptissä voi ilmetä tietoturvaongelmia. Cross-site scripting (XSS) on tietoturvauhka, jota voidaan hyödyntää SPAssa. XSS hyödyntää puutteita tietoturvassa, minkä avulla pystytään lisäämään omaa koodia.
SPA ei ole hakukoneena paras vaihtoehto, sillä MPA on parempi käsittelemään isoja määriä dataa. (Kaur, 2021.)

6 SPA ESIMERKKITOTEUTUS

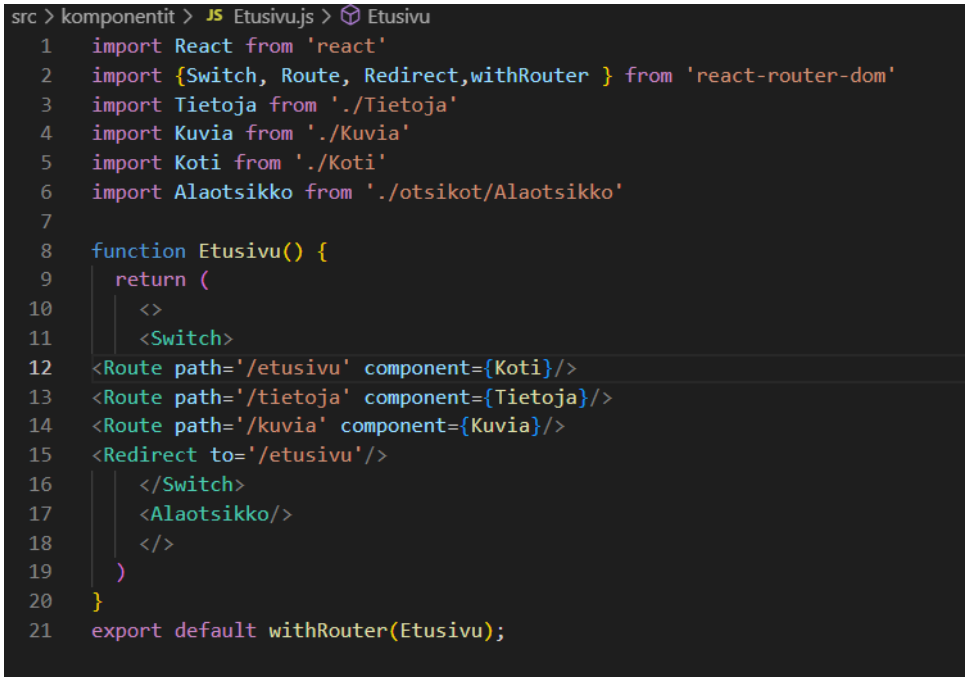
SPAn luominen aloitetaan asentamalla tekstieditori, jolla on mahdollista kirjoittaa ohjelmointikoodi web-sovellukseen ja kehitysympäristössä on mahdollista pyörittää sovelluspalvelua. Pohja SPAlle aloitetaan komennolla: (npx create-react-app .). Create-react-app ei ole nykystandardeilla paras tapa aloittaa projektia, koska sitä ei ole ylläpidetty, eikä se siten ole ajan tasalla. Vaihtoehtona voidaan esimerkiksi käyttää Viteä. Vite on ohjelmisto, jolla voidaan rakentaa Javascript web-sovelluksia helpommin. Reactin kehittäjän sivuilla on ehdotettu käytettäväksi esimerkiksi nextjs react- kehystä.
Luotu pohja voidaan käynnistää komennolla npm start ja todeta sen toimivuus. Tärkein tiedostopohjassa on app.js, ja tästä tiedostosta aloitetaan sivuston rakentaminen. Selkeyden kannalta komponentit ovat sijoitettuina omaan kansioon ja tarvittaessa esimerkiksi otsikoille voi luoda oman kansion. Hakasulkeita käytetään, kun palautettavia elementtejä on enemmän kuin yksi. Vaihtoehtoisesti voidaan käyttää <React.Fragment> -ominaisuutta, joka ajaa saman asian.

src > JS App.js > App

```
1   import React from 'react'
2   import { BrowserRouter } from 'react-router-dom'
3   import Etusivu from './komponentit/Etusivu'
4
5   export default function App() {
6     return (
7       <>
8       <BrowserRouter>
9       <Etusivu/>
10      </BrowserRouter>
11      </>
12    )
13  }
```

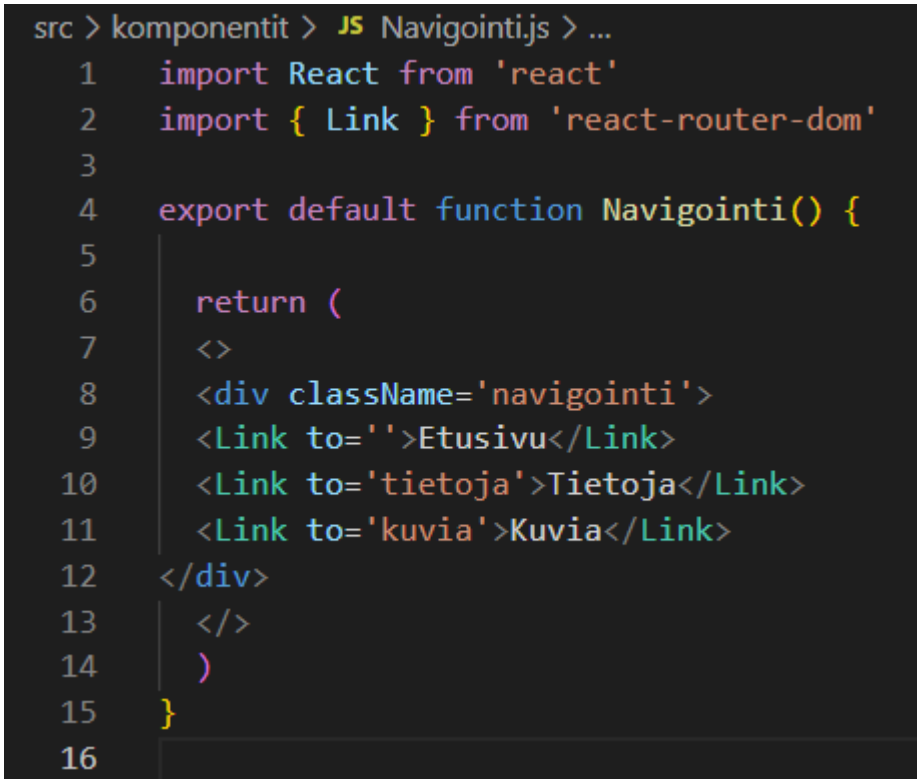
KUVA 16. App.js -tiedosto, Etusivu-komponentti on kääritty BrowserRouter-toteutukseen.

BrowserRouter on osa reitityksen toimintaa, sillä saadaan toteutettua näkymän päivitys sivujen linkkiä painaessa. Kuvassa 16 näytetty reitityksen toimintaa käytännössä.
Reitit määritellään Etusivu.js-tiedostossa, Switch käy läpi kaikki mahdolliset reitit ja valitsee reitin, jonka osoite vastaa osoitekentän osoitetta. Jos osoitekenttään on syötetty väärä osoite, Redirect-ominaisuus ohjaa käyttäjän takaisin etusivulle.



KUVA 17. Etusivu.js tiedosto, Switch-komponentin sisälle on määritelty käytettävät reitit ja Redirect-toiminto, joka palauttaa takaisin etusivulle.

Kuvassa 17 reitityksen toiminta etusivu-komponentissa.
SPAn kokonaisuudesta puuttuvat vielä linkit, joiden avulla käyttäjä päivittää halutun näkymän.

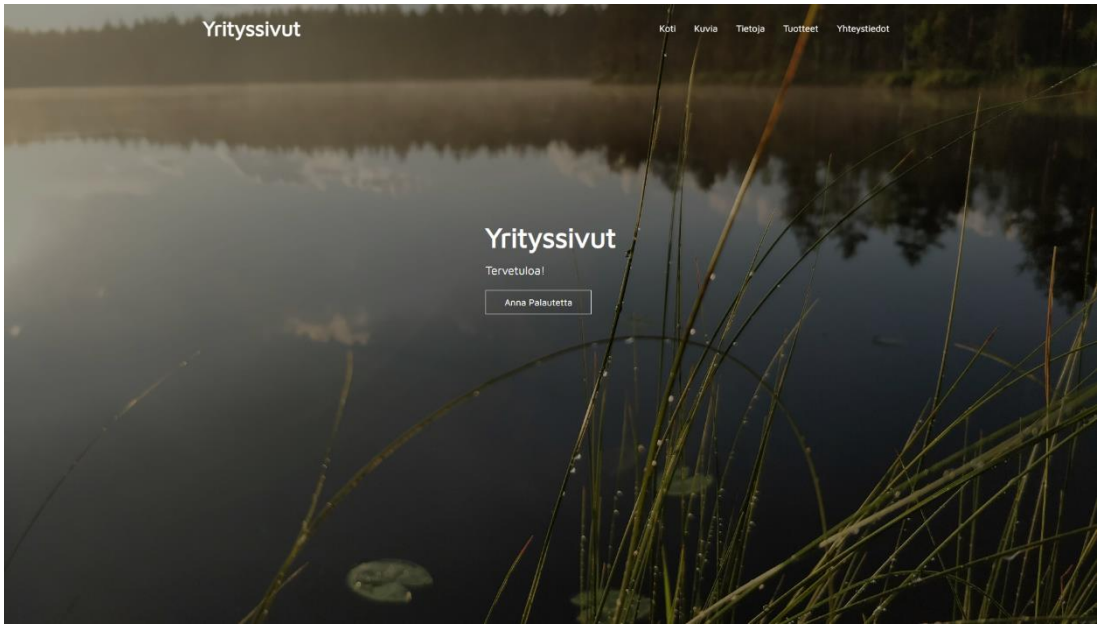


KUVA 18. Navigointi.js tiedosto, Esimerkki linkkien käytöstä.

Testataan toimivuus käynnistämällä terminaalista sovellus, SPAn toimivuus voidaan varmistaa painamalla linkkejä. Avaamalla selaimen kehittäjän työkalut nähdään, että Internet-välisivussa ei tapahdu muutoksia, kun linkkiä painetaan. Kuvassa 18 esitellään linkkien toimintaa.
Sivuille lisätään sisältöä ja tyylisivulla annetaan visuaalisesti tyydyttävä vaikutelma.

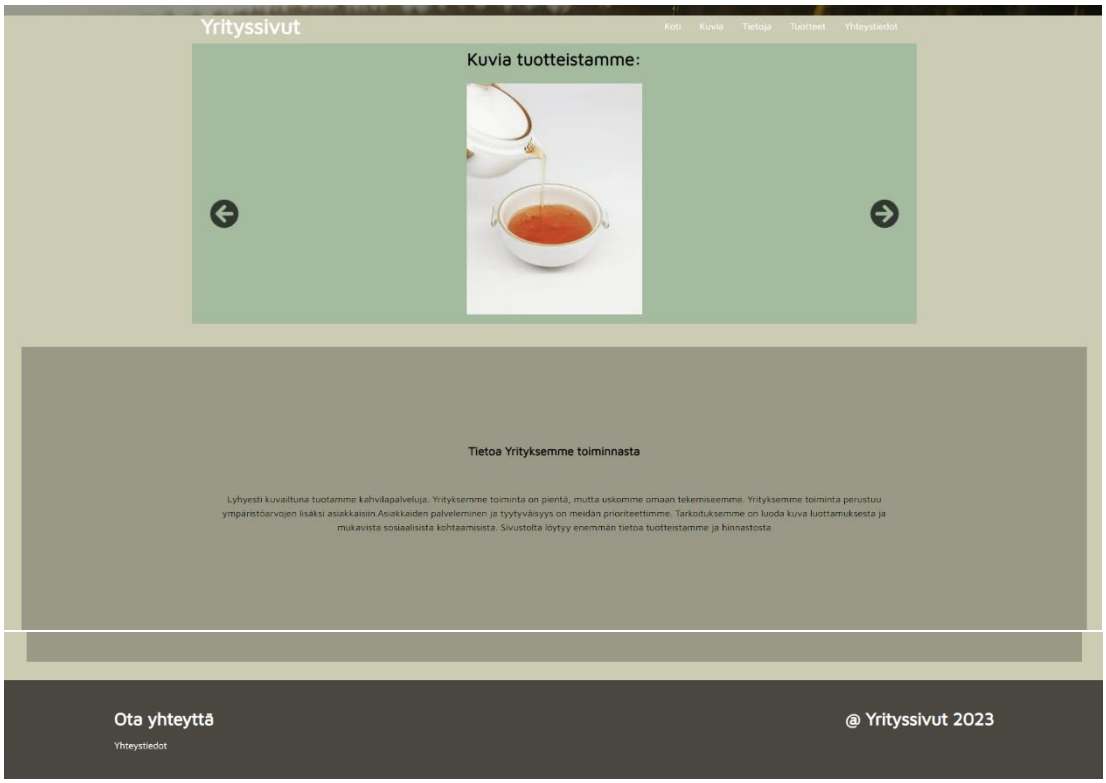
7 REFACTOROINTI

Tässä vaiheessa havaittiin, että kehittäjän sivuilla kehotettiin käyttämään muuta kuin Reactin omaa kehystä kehityksessä, ja päädyttiin vaihtamaan pohjaa.
Aloittaminen uudella kehyksellä vei oman aikansa, mutta alkuun pääsemisen jälkeen huomattiin nextjs:n olevan hyvin pitkälti samanlainen. Reititys on automatisoidumpaa. Projektin pohja luotiin komennolla (npx create-next-app).



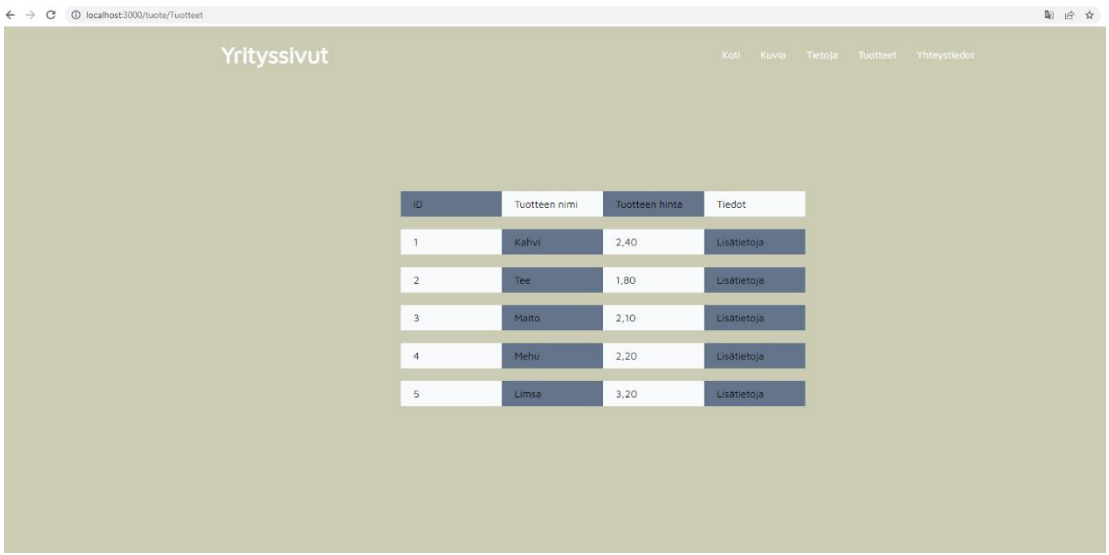
KUVA 19. Etusivun alkunäkymä selaimessa.

Sivustolle on haettu modernin websivun tyyliä ja apuna sivuston luomisessa on käytetty Tailwind cssää. Kyseessä on kirjasto, kirjaston ideana on antaa tyyli elementeille. Sen sijaan tyylisivustossa määritellään suoraan elementeille listaamalla halutut muutokset. Tästä syystä tyylisivuun ei tule paljoa sisältöä. Kuvassa 19 etusivun alkunäkymä.



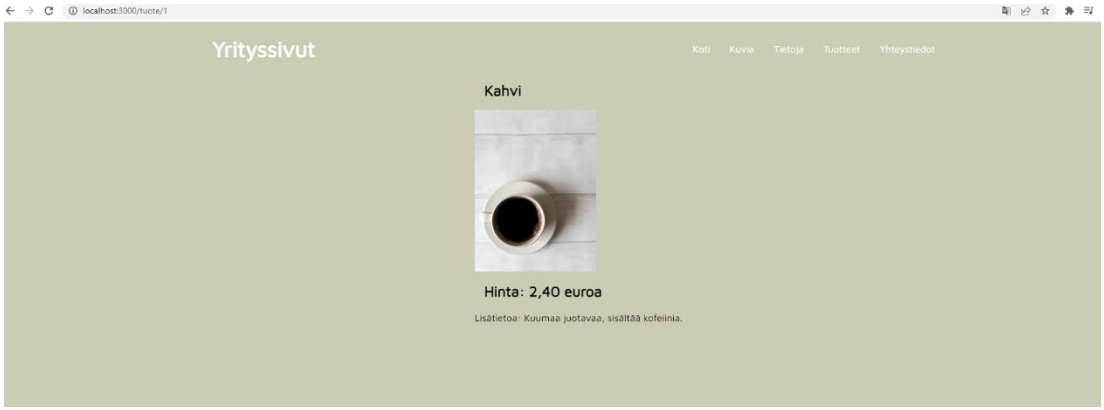
KUVA 20. Etusivun muu sisältö.

Etusivu sisältää otsikon, navigoinnin, gallerian, esittelyn ja alaotsikon. Jokainen osio on erillinen komponentti, jotka luovat sivun kokonaisuuden. Kuvassa 20 etusivun muu sisältö.



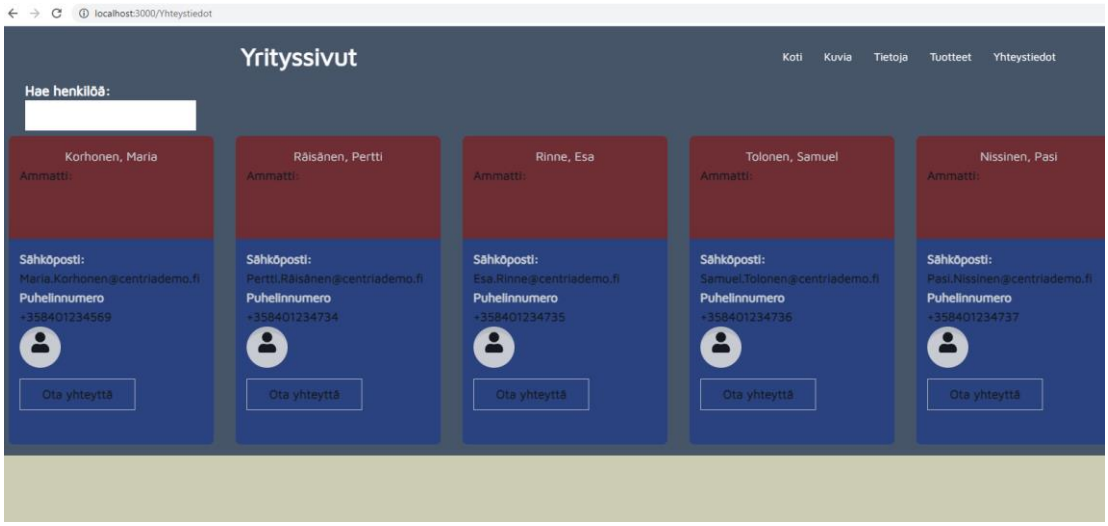
KUVA 21. Tuotesivu, tuotteet haettu JSON-tiedostosta ja listattu järjestyksessä käyttäjälle näkyviin.

Tuotesivu sisältää JSON:sta haettua tietoa. Tiedot ovat tuotteittain esillä kuvan 21 mukaisesti.



KUVA 22. Dynaamisesti luotu sivusto tuotteiden esittelyyn.

Lisätietoja-painikkeella pääsee dynaamisesti luotuun sivuun, jonka näytettävät tiedot perustuvat valittavan tuotteen id:n arvoon. Näin pystytään esittämään käyttäjälle halutut tiedot tuotteesta. Kuvassa 22 esimerkki dynaamisesti sivusta.



KUVA 23. Yhteystiedot-sivu, josta voidaan hakea henkilöitä yhteydenottoa varten.

Yhteystiedot-sivu sisältää työtä varten luotuja ei olemassa olevia käyttäjiä, joiden yhteystietoja voidaan hakea. Ota yhteyttä -painiketta painaessa syntyy lomake yhteydenottoa varten esitäytetyillä tiedoilla halutun yhteyshenkilön tiedoilla. Kuvassa 23 yhteystieto-sivu esiteltynä.

8 YHTEENVETO

Opinnäytetyön tavoitteena oli yhtä web-sivua käyttävän web-sovelluksen rakentaminen Reactin avulla. Työssä esiteltiin käytännön esimerkki, miten SPA toimii, ja kuinka sitä voidaan hyödyntää verkkosivujen luomisessa.

Opinnäytetyön painopiste oli käytännön toteutuksessa. Aluksi työn aloittaminen oli hyvin hankalaa, koska aihealue oli melko tuntematon, materiaalia aiheesta oli hyvin marginaalisesti saatavilla ja kirjallisuutta Reactista on hyvin vähän. Käytännön esimerkissä esiteltiin johdonmukaisesti kaikki työn vaiheet, joita sen luomisessa ilmeni. Muuan muassa opinnäytetyön kirjoittamisen aikana React-kehitystiimi julkaisi Reactiin paljon uusia ominaisuuksia, joten jouduttiin valitsemaan, mitä tässä työssä oli olennaista tuoda esille. Reactin vanhalla pohjalla työskentelyyn kului paljon aikaa, mutta onneksi suurin osa työstä pystyttiin siirtämään melko vaivattomasti uudelle pohjalle. Jälkeenpäin pohdittiin, mitä muita vaihtoehtoja Reactin tilalle olisi ollut käytettävissä.

Kehitys on nopeaa ja nykyään on paljon erilaisia JavaScript-kehyksiä. React on julkaissut muuan muassa palvelinkomponentteja, jotka helpottavat tiedon hakemista tietokannasta. Esimerkkinä uudemmista kehyksistä Svelte on saanut suosiota. Svelten koodin kirjoitus on yksinkertaisempaa kuin Reactin, koska se on automatisoidumpaa ja koodin kirjoittamisen tarve on vähäisempää.

Opinnäytetyötä tehdessä merkittävä havainto oli se, että teknologia kehittyy huimaa vauhtia ja opiskelun on oltava jatkuvaa vastatakseen työelämän vaatimuksiin. Myös loppukäyttäjien odotukset sovelluksen ominaisuuksista ovat kohonneet. Sovellusta kehittäessä React, sen kirjastot ja arkkitehtuuri tulivat syvällisemmin tutuksi antaen erinomaiset valmiudet käytännön työhön.

LÄHTEET

ADK Group, projekt202`s NE Region. 2022. *SPA vs MPA Applications: What Are the Differences?* Medium. Saatavissa: https://medium.com/@theadkgroup/spa-vs-mpa-applications-what-are-the-differences-7dc004e62397. Viitattu 1.6.2023.

Deutsch, D. 2017. *Understanding MVC Architecture with React* **Medium**. Saatavissa: https://medium.com/createdd-notes/understanding-mvc-architecture-with-react-6cd38e91fefd. Viitattu 1.6.2023.

Fedosejev, A. 2015. *React. Js Essentials*. Birmingham: Packt Publishing, Limited. Saatavissa: ProQuest Ebook Central. Viitattu 11.3.2023.

Finnegan, C. 2019. *The Major Benefits of Using Typescript*. Medium. Saatavissa: https://medium.com/swlh/the-major-benefits-of-using-typescript-aa8553f5e2ed. Viitattu 1.6.2023.

GeeksforGeeks. 2022. *What is Memory Leak? How can we avoid?* Saatavissa: https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/. Viitattu 1.6.2023.

Hyeny. 2021. *Virtual DOM vs Real DOM.* Medium. Saatavissa: https://hyeny.medium.com/virtual-dom-vs-real-dom-44d442eb2501. Viitattu 1.6.2023.

Immukul. 2023. *ReactJS Virtual DOM*. GeeksforGeeks. Saatavissa: https://www.geeksforgeeks.org/reactjs-virtual-dom/. Viitattu 1.6.2023.

Joshi, K. 2020. *Alternatives to Create React App.* Hackernoon. Saatavissa: https://hackernoon.com/alternatives-to-create-react-app. Viitattu 1.6.2023.

Kaur, A. 2021. *What is Single Page Application (SPA)? Pros and Cons with Examples.* Net Solutions. Saatavissa: https://www.netsolutions.com/insights/single-page-application/. Viitattu 6.5.2023.

KirstenS. n.d. OWASP. Cross Site Scripting (XSS). Saatavissa: https://owasp.org/www-community/attacks/xss/ . Viitattu 1.6.2023.

Krotoff, T. 2023. *FrontendFrameworksPopularity.md*. Github Gist. Saatavissa: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190. Viitattu 1.6.2023.

Mahra, D. 2020. *The Best Approach To Design React Component Hierarchy.* Saatavissa: https://medium.com/javascript-essentials/the-best-approach-to-design-react-component-hierarchy-978bb152dbb2. Viitattu 1.6.2023

Mattiazzi, R. 2021. *How React and Redux brought back MVC and everyone loved it.* Rangle. Saatavissa: https://rangle.io/blog/how-react-and-redux-brought-back-mvc-and-everyone-loved-it. Viitattu 1.6.2023.

React Router. n.d. *Tutorial.* Saatavissa: https://reactrouter.com/en/main/start/tutorial. Viitattu 1.6.2023.

React. 2023. *Introducing JSX.* Saatavissa: https://reactjs.org/docs/introducing-jsx.html. Viitattu 8.5.2023.

Sharma, T. 2021. *React SPA for Increased Performance & CX*. Royalcyber. Saatavissa: https://www.royalcyber.com/blogs/react-spa-for-increased-performance-and-customer-experience/. Viitattu 1.6.2023.

Technostacks. 2023. *The Top React Component Libraries that are Worth Trying*. Saatavissa https://technostacks.com/blog/react-component-libraries/. Viitattu 1.6.2023.

Vakhnenko, H. 2022. *Single-Page Apps vs Multiple-Page Web Apps.* Agilie. Saatavissa: https://agilie.com/blog/single-page-apps-vs-multiple-page-web-apps. Viitattu 1.6.2023.

Walter, S. 2008. *The evolution of MVC.* Saatavissa: http://stephenwalther.com/archive/2008/08/24/the-evolution-of-mvc . Viitattu 1.6.2023.