Mobiilisovelluksen toteutus WordPress verkkosivuun

Guichly Hessen

Haaga-Helia ammattikorkeakoulu

Amk-opinnäytetyö

2021

Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tekijä(t)
Guichly Hessen

Tutkinto
Tietojenkäsittelyn koulutusohjelma.

Raportin/Opinnäytetyön nimi
Mobiilisovelluksen toteutus WordPress verkkosivuun.

Sivu- ja liitesivumäärä
21+3

Opinnäytetyön tarkoituksena on selvittää miten WordPress rajapinnat ja headless CMS ratkaisut toimivat. Mobiilisovelluksen toteuttamista React Native ohjelmisto kielen avulla. Työssä käydään erilaisia tekniikoita ja työkaluja, jota on oleellista osata ennen sovelluksien tekemistä.

Työssä pohditaan erilaisia näkökulmia toteuttaa mobiilisovellus valmiin WordPress verkkokauppa sekä vertaillaan eri tekniikoita staattisten ja dynaamisen sivujen välillä sen hyödyistä ja haitoista, ja sen vaikutuksista kehitykseen tulevaisuudessa.

Opinnäytetyössä toteutetaan myös demo mobiilisovelluksen kiani.fi WordPress verkkokaupasta, jonka toteutetaan React Native natiivi komponentteja käyttäen. Esitän työn edistyessä eri esimerkkejä käyttäen mobiilisovelluksen sisällöstä sen aloittamisesta valmiiksi asti.

Lopussa pohditaan mitä tuli opittua ja mobiilisovelluksen jatkokehittämisen mahdollisuuksia ja työn onnistumisesta sekä haasteista.

Lukijan on hyvä osata vähintään yksi ohjelmointikieli ymmärtääkseen koodi esimerkit ja sen käyttötarkoituksista. Luettavuuden helpottamiseksi on koottu pää termejä liite osiossa.

Asiasanat
React Native, Heddles CMS, WordPress, Ohjelmistokehitys, Mobiilisovellus

Sisällys

1 Johdanto

Tänä päivinä digitalisaatio on kasvanut huimasti. Mobiilisovelluksen käyttö on yksi niistä yleisin tapa yrityksen toiminnalle saada näkyvyyttä ja asiakkaita mukaan liiketoimintaan, yksinkertainen ja helppokäyttöinen mobiilisovellus voi tuottaa hyvät ansiot. Idean sain kaverilta Siavash Kianilta, hän mainitsi tekevänsä verkkokaupan opinnäytetyönsä WordPress verkkokaupoista.

Tutkin tässä opinnäytetyössä ensiksi, miten mobiilisovelluksien voi saada toteutettua yksinkertaisesti ja helposti. Opinnäytetyö perustuu tilastollisiin ja kirjallisuuden lähteistä, sekä omasta koulussa oppimisen taidoista. Lopuksi teen WordPressiin integroidun käytännöllisen toimivan esimerkin valitsemaani verkkokauppaan.

Toimialaisen tutkielmaan tavoitteena on myös antaa lukijoille WordPressiin liittyvistä avoimeen lähdekoodin perustuvan REST rajapinnan käytännöllisyyksistä ja sen tarkoituksista. Käytän myös front end koodaamisessa React Nativea ja sen komponentteja. back end toteutetaan headless CMS tekniikkaa käyttäen, jossa WordPress verkkokauppa toimi backend:inä.

Lopputuloksessa on tarkoitus saada lukijatkin ymmärtämään edellä mainituista tekniikoista ja ymmärtää React Nativen eri komponentit ja toiminnot; opinnäytetyö käyttäen koodata omat rankaisunsa tai tutkielman demoa muokaten tuoda omat WordPress sivustonsa näkyviin mobiili laiteisiin.

1.1 Tavoitteet ja menetelmät

Opinnäytetyö keskittyy WordPressin valmiisiin integroituihin kirjastohiin ja valmiisiin ratkaisuihin erilaisia kirjastoja käyttäen. Tavoitteena on tutkia miten WordPress REST rajapinna ominaisuudet GET pyynnöillä ja renderöidä JSON datasta React Nativen eri komponentteja käyttäen interaktiivisen esimerkki sovelluksen. Sovellus tulee käyttämään WordPressiä backend:inä se toteutetaan headless CMS tekniikkaa käyttäen React Nativen komponentteja käyttäen, josta verkkokaupasta haetaan hakusanalla tuotteita ja tulostetaan ne ruudulle.

Tämän opinnäytetyön loppu tarkoitus on tukea ja auttaa ymmärtämään lukijoita miten yksinkertaisesti oman WordPress sivunsa voi saada näkyviin puhelimissaan sekä Androidissa, että IPhonessa ja halutessaan tehdä oma rankaisun mallia seuraten tai käytä opinnäytetyössä käytettyjä tekniikoita tukemaan uusia muiden ratkaisuja.

1.2 Rajoitukset

Sekä React että WordPress ovat valtavia kehyksiä, jotka tarjoavat monia erilaisia toimintoja, joita voidaan käyttää useissa projekteissa. Tämä opinnäytetyö kattaa vain REST rajapinnan toiminnallisuudet ja tulostaa sen React Native:ven komponentteja käyttäen tulostaa JSON formaatista tyyliteltynä käyttäjälle front end:in, että sovellus toimii vain rajallisesti hakutoiminnot ja navigaatio ominaisuudet.

Esimerkiksi tämä opinnäytetyö ei tule käsittelemään WordPress teemoihin kehittämistä tai laajennuksesta, koska tämä opinnäytetyö on vain rajattu REST rajapinnasta tuotujen tietojen näyttämiseen front end:issä ja sen hieman tyylittämiseen.

1.3 Rakenne

Tämä opinnäytetyö jaetaan kolmeen osaan. Kerron ensin avainteknologioista ja selitä niiden ominaisuuksista yksityiskohtaisemmin. Kukin niistä miten toimivat ja miten niitä voidaan käyttää yhdessä ratkaisumallissa. Lisäksi kerron eri ohjelmiston osista ja koodikielistä sekä avaan käsitteitä, jotta koko projekti ja sen tarkoitus tulisi ilmi lukijalle paremmin.

Työttä voi olla vaikea ymmärtää sen sisällöstä ja termeistä, jos ei ole ajamapa jonkinlaista ohjelmoinnin kokemusta. Tämä opinnäytetyö käsittelee eri rankaisuja ohjelmoijan näkökulman kannalta, kuitenkin olen pyrkinyt avaamaan termejä ja selittämään eri tekniikoista.

Kun tekniikat ja muut käsitteet ovat tuttuja, toisessa osassa selitetään mobiilisovelluksen luonnista ja sen tyylityksestä. Sekä tuon ilmi miten edellisessä kappaleessa esitettyjä tekniikoita käytäntöön panon ohjelmistossa sivuston sisällä.

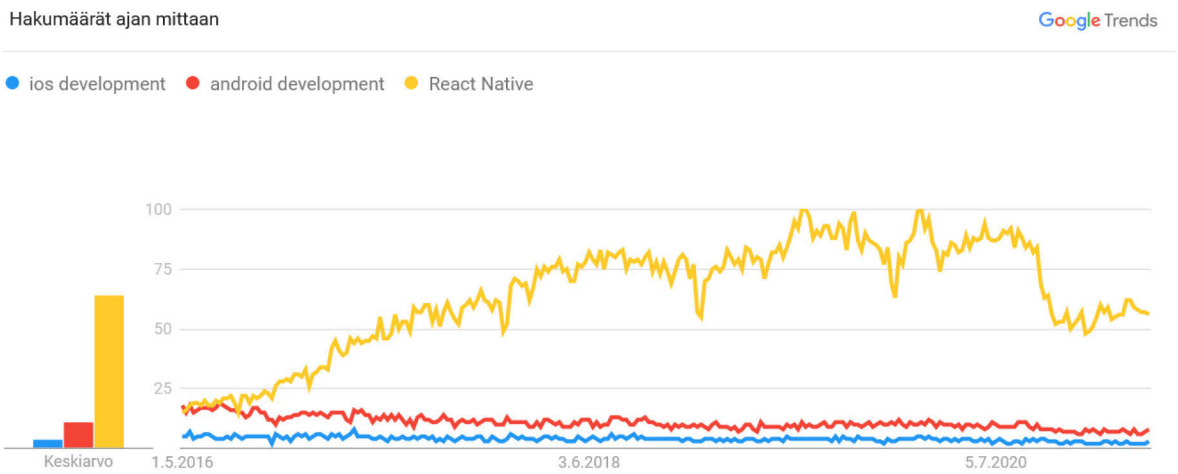
Lopuksi pohdinnassa kerron havainnoistani tutkiessani eri tekniikoita luoda mobiilisovelluksen. Enemmän kertomaan mitä eri tietoa kukin komponentit sisältävät ja miten sitä voidaan soveltaa eri käytännön tarkoituksissa esimerkkiä käyttäen, sekä kerron keskittyneesti arvion sivustojen kehittämisen mainittujen tekniikoiden eduista ja haitoista.

2 Avainteknologiat

Tässä kappaleessa käydään läpi tutkien syvemmin eri ohjelmisto kielten toimivuudesta ja teknikoiden käyttötarkoitusta ja teoriaa niistä lyhyesti.

2.1 React Native

React Native on saanut suosionsa mobiilisovelluksen kehittämisen kehityskehyksensä, joka perustuu React arkkitehtuuriin ja sen perustana on JavaScript, joka kehitettiin 2013 kansanvälisessä hackathon projektissa, jonka piti Facebook ja julkaistiin iOS ja Android käyttöön 2015 julkisuuteen GitHubissa (TechAhead 2020.). Ennen 2015 tehtiin ohjelmat webiin JavaScriptiä ja HTML5 käyttäen ja androidille Cordova:lla, mutta Ract Native oli ensimmäinen, joka yhdisti iOS ja Android. Ensinäkin Google trendin mukaan React Naitev:en hakukoneiden hakuteoksien kärkeen nousi ensimmäistä kertaa ohi Android ohjelmistokehityksen ohi syyskuussa 2016 ja edelleen on kärjessä (Shoutem 2016.).



Kuva 1. Google trendin kokoamillaan verkkohaku viimeisten 5 vuoden akana

Reat Native käyttävät muun muassa Instagram, Facebook, Discord ja Skype.

2.1.1 JSX

React Native komponentit kirjoitetaan JavaScript, joka tunnetaan yleisemmin nimellä JSX, joka on luotu JavaScript ja XML koodin yhdistämällä yhdeksi kokonaisuudeksi. JSX on tapa yhdistää sekä logiikka että merkinnät yhdessä tiedostossa. React Native ei vaadi JSX: n käyttöä, mutta monien mielestä siitä on hyötyä työskennellessään käyttöliittymäsuunnittelun elementtien kanssa JavaScriptin koodin sisällä, koska se käyttää samanlaista syntaksia kuten HTML syntaksi. JSX näyttää TypeScript ohjelmointikieleltä, mutta sen käytössä on kaikki JavaScriptin ominaisuudet, mikä tekee JSX:stä erittäin tehokkaan (Facebook Inc 2021a.).

Kuvassa 2 esitän yksinkertaisen esimerkin JSX:stä. Ensin JavaScript muuttuja ”name” on luotu, ”return” funktion jälkeen sitä voidaan käyttää JSX:n sisällä pistämällä se kiharaisiin aaltosulkeisiin (Facebook Inc 2021a.).

```
const Cat = () => {
  const name = "Guichly";
  return <Text>Hello, I am {name}!</Text>;
};
```

Kuva 2. Esimerkki lausekkeiden upottamisesta JSX:ään (Facebook Inc 2021a)

Itse asiassa kelvollista JavaScript lauseketta voidaan käyttää missä tahansa JSX:n sisällä yksinkertaisesti käärimällä se kiharaisiin aaltosulkuihin. Kuvassa 3 näkyy JSX: n käyttö JavaScript lausekkeen sisällä. ”getFullName” funktion sisälle kaarisulkeet sisällä on muuttujat ”return” palauttaa lausekkeen kolme kertaa tässä esimerkissä on ja lisää lopuliseen tulosteeseen ”Rum”, Tum”,”Tugger” eroteltuna (Facebook Inc 2021a.).

```
const getFullName = (firstName, secondName, thirdName) => {
  return firstName + " " + secondName + " " + thirdName;
};

const Cat = () => {
  return <Text>Hello, I am {getFullName("Rum", "Tum", "Tugger")}!</Text>;
};
```

Kuva 3. Esimerkki JSX-lausekkeesta uudelleen käytettävyydestä (Facebook Inc 2021a)

React:in ja Ract Nativen suurin eroavaisuus on siinä, että kaksi komponenttia pystyy kommunikoimaan saman luokan sisällä. ”Parent” komponentti on ”Cafe”, joka tulostaa ruudulle jokaisen ”Cat” funktion sisällön. Sisältö pysyy muuttumattomana joka kerta kun funktiota kutsutaan. ”Text” komponentti pysyy muuttumattomana (Facebook Inc 2021a.). Sen käytöstä kuvassa 4.

```
const Cat = () => {
  return (
    <View>
      <Text>I am also a cat!</Text>
    </View>
  );
};

const Cafe = () => {
  return (
    <View>
      <Text>Welcome!</Text>
      <Cat />
      <Cat />
      <Cat />
    </View>
  );
};
```

Kuva.4 Esimerkki ”Parent” ja ”Child” komponentista (Facebook Inc 2021a)

Tässä kuvassa 5 on tyyli komponentin ja kommentti funktion käytöstä yksinkertainen esimerkki piirto ”Parent komponentit sisällöstä. Ensimmäiseksi ”StyleSheet” komponentit tuodaan React Natiivin kirjastosta ja luodaan tyyli komponentit. Otetaan tyyli käyttöön ”Text” komponentin sisällä lisäämällä ” style={styles.textStyle}” ”style” on muuttuja ja pistetään JSX:n koodi kiharaisiin aaltosulkeisiin sisälle. Kommentit toimivat ”render” komponenttien ulkopuolella ”//” käyttäen, mutta sisäpuolella kommentoidessa on käytettävä ”multi line” funktiota pistämällä kommentit aaltosulkeisiin sisälle ja sen jälkeen ”*/tekstiä/*”. Tämä on koska, JSX funktiot ”render” komponentin sisällä, Ilman aaltosulkeita se näkyy tavallisena tekstinä. Muutoin tyylityksen syntaksi on CSS mukainen (Facebook Inc 2021b.).

```
//Parent Komponenti
const Cat = () => {
  return (
    <View>
      Tämä on vain teksitä
      {/*Tyyli komponenttia käytetään teksti komponentissa*/}
      <Text style={styles.textStyle}>This is styled text</Text>
    </View>
  );
};

//Luodaan tyyli komponentti
const styles = StyleSheet.create({
  textStyle: {
    fontSize: 16
  }
});
```

Kuva 5. Esimerkki Tyylien käyttö teksti komponentin sisällä

2.1.2 Props

Tietojen siirtämiseksi komponenttien välillä React käyttää objektia, jota kutsutaan nimellä ”props” joka on lyhenne ”ominaisuuksista”. Komponentit ovat teoriassa kuin JavaScript-toiminnot, ne hyväksyvät sattumanvaraisia data. Komponentin avulla yhdessä tulostavat näytölle näkymän datan sisällöstä. Tämä myös mahdollistaa helposti komponenttien uudelleen käytön eri komponenttien attribuuttien sisällä. (Facebook Inc 2021a.). Kuvassa 6 on esimerkki props:ien siirtämisestä komponentille. React Native kuten React:ista että propsit ovat ”read only” data mitä ei voida muokata suoraan funktion sisällä, sen määrittää back endi, mutta tarvitsee ”key” id elementin React Native tietää jokainen elementti mitä tulostetaan ruudulle, on uniikki itsessään.

```
const Greeting = (props) => {
  return (
    <View style={styles.center}>
      <Text>Hello {props.name}!</Text>
    </View>
  );
}

const LotsOfGreetings = () => {
  return (
    <View>
      <Greeting name='Rexxar' />
      <Greeting name='Jaina' />
    </View>
  );
}
```

Kuva 6. Esimerkki Props:en käytöstä (Facebook Inc 2021b)

2.1.3 State

State hallitsee komponenttien tilaa, joka kerta funktion tiedon päivittyessään koko komponentin sisältö udelleen renderöidään. Propsi ja Staten on hyvin samanlainen toiminto ja voi tämän takia vaikea omaksua ensiksi, mutta eroaa että ”state” ainoastaan vain katselee kyseisen komponenttien funktioiden tiloja ja ”props” sisältää dataa mitä ”parent” ja ”child” komponentit käyttävät kommunikoidessaan (Facebook Inc 2021a.).

Kuvassa 7 on esimerkkinä näyttö laskurista, joka laskee kuinka monta kertaa käyttäjä painaa nappia. Ensiksi luodaan ”counter” ja ”setCounter” mutuja ”useState” on aloitus tila sovelluksen käynnistäessä, joka on merkittynä 0, kun käyttäjä paina nappia ”onPress” funktio kutsuu ja muokkaa arvoa ”setCounter” muuttujassa, edelleen ”counter” arvo ei muutu koska props:it ovat ”read only” data sen jälkeen näkymä uudelleen renderöidään ja ”counter” päivittyy.

```
const CounterScreen = () => {
  const [counter, setCounter] = useState(0);

  return (
    <View>
      <Button
        title="Increase"
        onPress={() => {
          setCounter(counter + 1);
        }}
      />
      <Text>Current Count: {counter}</Text>
    </View>
  );
};
```

Kuva 7. Esimerkki State käyttö tarkoituksesta (Facebook Inc 2021b)

2.2 Staattiset sivut

Internetin tulon jälkeen staattisia sivustoja on luotu perinteiseen tapaan. Ensimmäinen koskaan luotu verkkosivusto oli staattinen verkkosivusto, koska tuolloin ei ollut muita vaihtoehtoja. Vaikka sivustojen kehittämiseen käytetty tekniikka ovat jatkaneet edistymistään siitä lähtien, staattiset sivustot tarjoavat edelleen ominaisuuksia, jotka tekevät niistä houkuttelevia vaihtoehtoja, jopa nykyään (Camden & Rinaldi 2017.).

Tässä on muutama staattisen sivuston monista eduista, kuten Raymond Camden ja Brian Rinaldi mainitsee kirjassaan ”Working with static sites” (Camden & Rinaldi 2017.).

Nopeus: Staattinen sivusto on nopea, koska sama HTML on kaikille käyttäjille, eikä mikään dynaaminen sivu pitää renderöidä ennen kuin se tulostaa näkymän ruudulle, joka voi olla hidasta. Nopeus on erittäin tärkeää, koska on osoitettu, että jos latausaika ylittää kolme sekuntia, käyttäjät todennäköisesti poistuvat sivustolta.

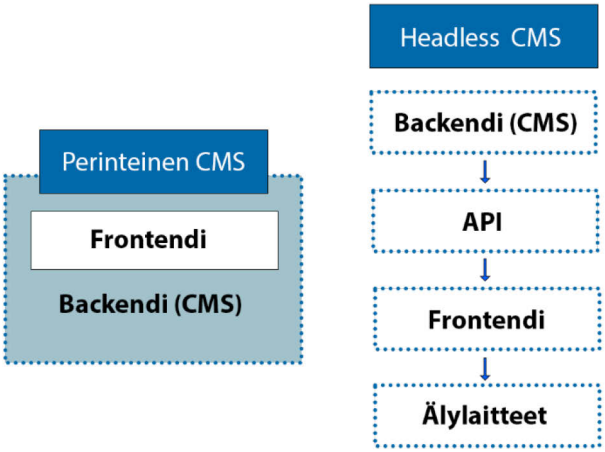
Suojaus: Staattiset sivustot ovat turvallisempia kuin dynaamiset sivustot. Yleensä, kun sivusto tuhotaan, se johtuu siitä, että CMS on vanhentunut, mutta koska staattinen sivusto on vain staattinen HTML ja CSS, CMS: n haavoittuvuuden riski on kokonaan poissa. Tietenkään mikään sivusto ei ole 100% turvassa erilaisilta tietoturva hyökkäyksiltä, mutta staattisen sivun ansiosta sivustoa on vaikeampaa vahingoittaa.

Monipuolisuus: Koska staattiset verkkosivustot eivät ole riippuvaisia CMS:sstä, verkkosivustojen ulkonäön mahdollisuudet ovat rajattomat. Se voidaan räätälöidä täysin sovelluksen tarpeiden mukaan.

Isännöiminen: Staattinen sivusto myös eliminoi sovelluksen isännöinnin vaivat. Koska vain staattisia tiedostoja se asennetaan kerran, sen takia on olemassa loputtomia tapoja, miten ja missä laiteissa ohjelmistot toimivat.

2.3 Headless CMS

Storyblok artikkelissa Dominik Angerer selittää päättömän CMS:n. "headless CMS” on back end sisällön hallintajärjestelmä ”CMS”, joka on rakennettu alusta alkaen sisältövarastoksi, joka tekee sisällön saataville RESTful sovellusliittymän kautta näytettäväksi millä tahansa laitteella (Angerer 2021.). Missä perinteisessä ratkaisussa CMS: on kytkettynä front end:iin suoraan back end:iin kanssa, headless CMS:ssä eivät ole. Tämä mahdollistaa datan helpon jakamisen monille eri kanaville, jota kuvastaa eri älylaiteet katso kuva 8. Sana ”headless” päätön tulee sovelluksen front end:in irrottamasta back end:istä.



Kuva 8. Periteisen ja headless CMS toiminnan kuvastus

Tyypillisessä staattisessa sivustossa sisältö tallennettaisiin tiedostojärjestelmään. Headless CMS antaa mahdollisuuden tallentaa samat tiedostot CMS:ään ja sitten siirtää data ulos API:n kautta front end:iin sieltä älylaitteiden ruuduille. Sen sian, että data on koneella paikallisissa tiedostossa, sillä hyödyt CMS editorista ovat valtavat (Camden & Rinaldi 2017.).

2.4 WordPress

WordPress julkaistiin alun perin vuonna 2003 bloggausalustana. Vuosien aikana se kasvoi olla yksi suosituimmista sisällönhallintajärjestelmistä, ja tänään WordPressia pidetään pikemminkin CMS:nä kuin myös blogityökaluna (Ratnayake 2017.).

WordPress perustuu PHP:hen ja MySQL:ään. Se hallitsee 32% verkosta kaikilla henkilökohtaiset blogit, salkut yrityssivustoille. WordPress ei ole vain CMS, vaan myös auttaa sovelluksien rakentamisessa, tarjoaa sivut eri kiellä, URL-reitityksen ja vastaa HTTP-pyyntöihin. WordPressin keskeinen piirre on sen yksinkertaisuus, se on rakennettu nopeaan ja sujuvaan nettisuvin lumoamista varten (WordPress Org 2021.).

2.4.1 WordPress REST API

WordPress käyttää omaa REST sovellusliittymää, jonka avulla kehittäjät voivat olla vuorovaikutuksessa sivustojen kanssa lähettämällä ”get” pyyntöjä ja vastaanottaa JSON-objekteja. WordPress REST rajapinta on suunniteltu sovelluksille yksinkertaisen pääsyn WordPressin sisältöön kivuttomasti ja helposti. Ohjelmointi kielet, jotka tukevat JSON datan käyttöä ja lisäksi voivat tehdä HTTP-pyyntöjä ovat toimivia rankaisuja (WordPress Org, 2021a.).

3 Mobiilisovelluksen Toteutus

Päätin tehdä sovelluksen kokonaan React Native:lla täten ei tarvita koodata erikseen Androidille ja iOS-alusoille omaa versiota. Expo tekee asiasta helpomman, se CRNA-paketin hallinnan järjestelmä, joka toimi NPM:n kautta. Sen avulla muutokset päivittyvät Android emulaattorissa saman tien ruudulle ja sen avulla voidaan luoda konfiguraatiot ja aloituksen yhdellä komennolla, muutoin joudutaan tekemään iOS tai Androidin rankaisut kukin omilla alustoillaan Android Studiossa tai Xcode:ssa (Facebook Inc 2021c.).

Expossa on huononna puolena se, että siinä voidaan vain käyttää React Nativen komponentteja mitkä ovat määritettynä oletuksena kielessä. Mutta tämä soveltuu täydellisesti tähän projektiin, koska kyseessä on headless CMS rankaisu, jossa ei tarvita ulkoisia komponentteja taakka rankaisuja. Expo tarjoaa halutessaan poistumismahdollisuuden myöhempiin rankaisuihin siirtymällä ohjelma taustaisin ympäristöihin. Itse käytän Kiani Solutions:in omistamaa WordPress verkkokauppa ”kiani.fi” esimerkki sovelluksen luomisessa.

3.1 Koodaus ympäristön määrittäminen

Kehitysprosessi alkaa asentamalla Node.js paketin hallinnan järjestelmän ”npm”, joka asentaa eri kirjastoja komentokehoteen kautta syöttämällä ”npm install paketin-nimi”. Kun Expo.io ja Yarn on asennettuna, voit valinnaisesti asentaa Android Studion kautta virtuaalisen älypuhelimen, josta muutokset ja virhekoodit näkyvät saman tien, kun muutoksia tehdään koodiin. WordPress:in eri laajennukset eivät ole välttämättömiä, koska Expo.io startupperin sisältää komponentit sovelluksen toimimiselle, ja se helpottaa kehitystä huomattavasti. Kehityksen aloittamiseksi sinulla on oltava myös paikallisesti käynnissä oleva WordPress-sivusto tai julkinen WordPress-sivusto, jos kyseessä on julkinen sivusto pitää olla REST API otettuna käyttöön.

3.1.1 Expo

Expo on yleisten React-sovellusten kehys ja kehitys alusta. Se sisältää erilaisia työkaluja ja palveluta, jotka on rakennettu React Native ja natiivialustoille, joiden avulla voit kehittää, rakentaa, ottaa käyttöön ja nopeasti toistaa iOS, Android ja verkkosovelluksia samasta JavaScript tai TypeScript koodipohjasta (Expo.io 2021.).

Projekti ympäristö luodaan seuraavasti avaamalla komentokehote ja sinne syöttäen seuraavat komennot seuraavissa järjestyessä, ” npm install --global expo-cli” tämä asentaa Expo.io:n komentoliittymän globaalisti ja luodaan seuraavaksi projekti komennolla ”expo init projektin-nimi” ja valiten ”tyhjä malline”, missä React Nativen komponentit ovat valmiina asennettuna. Seuraavaksi avataan komentokehote projektin luodussa kansiossa ja

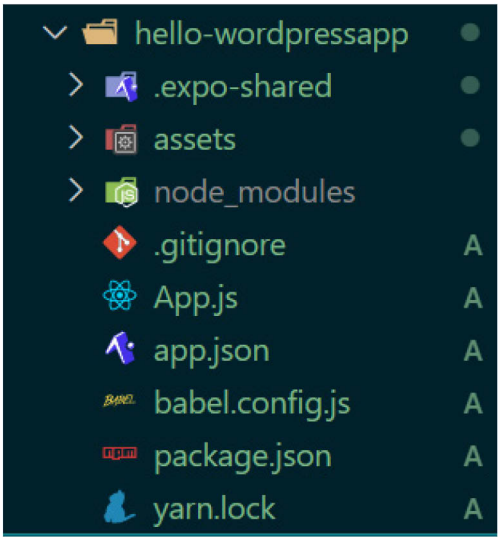
syötetään ”expo start”, joka käynnistää projektin automaattisesti lokaalisti koneella (Expo.io 2021.).

3.1.2 Yarn

Yarn on paketinhallinta järjestelmä. Sen avulla voit käyttää ja jakaa esim. JavaScript koodia muiden kehittäjien kanssa ympäri maailmaa. Se on avuksi eri ongelmien ja helpottaa kehittämän sovelluksia nopeasti ja vaivattomasti. Koodi jaetaan eri pakettien kautta, joita kutsutaan myös nimellä moduuli. Paketti sisältää kaiken jaettavan koodin ja eri paketit konfiguroidaan ”package.json” tiedostossa erikseen (Yarn 2020.).

Yarn on saatavilla NPM:n kautta ja asennetaan seuraavalla komennolla syöttäen komentokohteeseen kehitysympäristöstä riippumatta ”npm install --global yarn” (Yarn 2020.). Tässä esimerkki sovelluksessa käytetään Yarn paketin hallita järjestelmää React Native:n komponenttien asentamisessa, koska se on kevyempi ja nopeampi projektin hallinnan kannalta.

3.2 Ohjelmiston rakenne



Kuva 9. Aloitus projektin rakenne ja sisällöstä

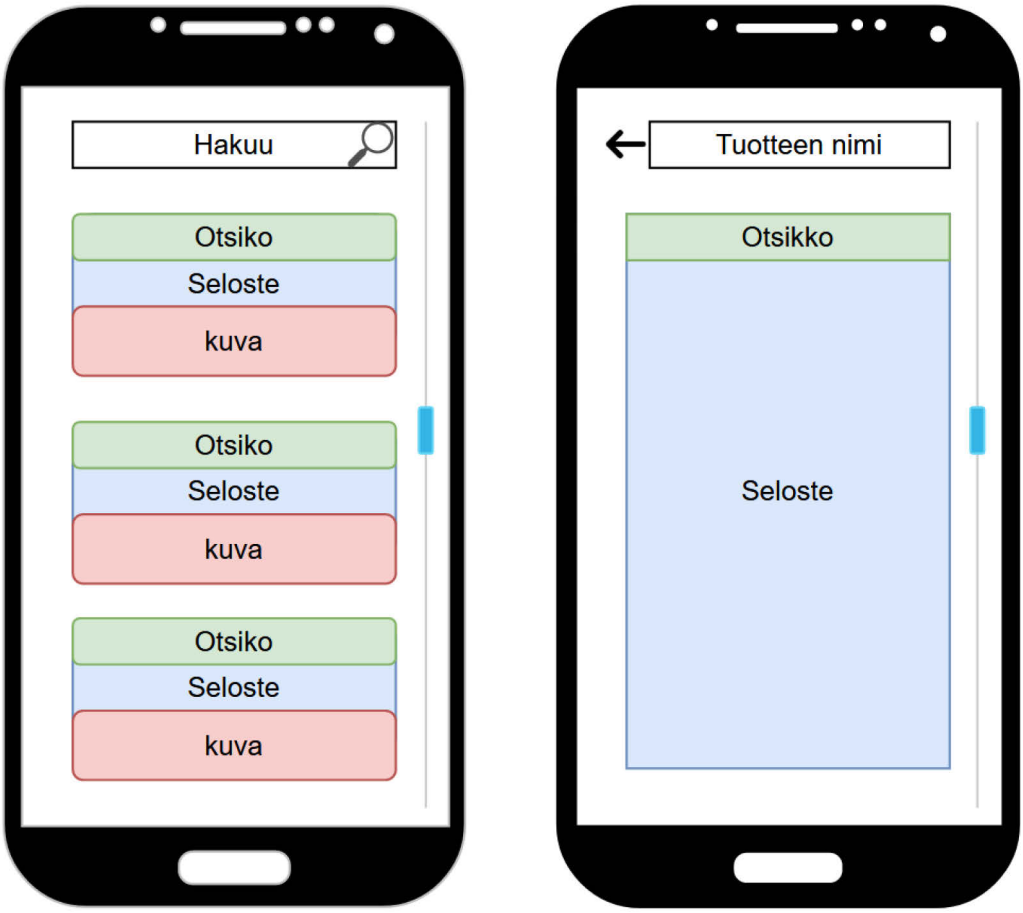
Portaalisivuston kehittämisen kannalta tärkeimmät tiedostot ovat App.js ja ”app.json”, joissa määritetään eri konfiguraatiot projektille ja kirjoitetaan koodia App.js tiedostoon ja se näyttää tulosteen käyttöliittymässä. Muita tiedostoja muokkaa automaattisesti Expo.io ja Yarn paktin hallinta järjestelmät.

3.3 Kyselyt

Projektin ydin on tehdä kyselyjä WordPressistä ja tuoda JSON formaatista dataa sovellukseen ja sieltä renderöidä data ruudulle. Tulen käyttämään WordPressin REST rajapinnasta seuraavia ominaisuuksia:

- “wp-json/wp/v2/product”

3.4 Käyttöliittymä



Kuva 10. Luonnos applikaation näkymistä

Kyseessä on WordPress verkkokauppa, tavoitteena on saada näkyviin tuotteiden otsikot ja selosteet sekä kuvan. Hakutoiminto pitäisi toimia hakutermejä käyttäen tuotteille ja erillinen näkymä valitulle tuotteille; toimiva navigaatio ja nappi takaisin etusivulle. Rullaus funktion on toimittava myös. Tämä ei vastaa sovelluksen lopullisesta ulkonäöstä sitä voidaan muutella tarpeen tullen toteutuksen aikana.

4 Koodin sisältö ja selosteet

Kiani.fi on verkkokauppa, joka on perustettu 2018 myymään joka päivittäisiä tuotteita, tässä mobiilisovelluksen toteutuksessa keskityn enimmäkseen tuotteiden hakemiseen ja sen näyttämisen ruudulla listamaisessa muodossa yksitellen React Nativen uudelleen käytettävillä komponentteineen kanssa. Navigaatio onnistuu eri näkymiltä helposti valmiiden komponentti kirjastojen ansiosta, lisäksi hakkuu komponentti funktio tuo eri käyttäjän syötetyn termien kautta kysely funktioiden avulla WordPress API:an kautta dataa. Kokonaisuudessaan tulee olemaan toimiva ratkaisu verkkokaupan.

React ohjelmaa koodin kirjottaessa, voit käyttää tavallisia HTML-elementtejä (<ol>, <ul>, <table>, jne.). React Native -ohjelmalla nämä elementit korvataan alustakohtaisilla React-komponenteilla (Eisenman 2017a.).

Kaikki käyttöliittymän elementit ovat React Native komponentteja, jokainen käsiteltävä tiedoston on tuotava erikseen ne seuraavalla tavalla, kuten kuvassa 11 on esitetty.

```
import React from "react";
import { View, Text, StyleSheet, Image, TouchableOpacity } from "react-native";
import { useNavigation } from "@react-navigation/native";
```

Kuva 11. Lista eri React Nativen komponenteista ja niiden tuonneista

React Native ohjelman eri komponentit löysin pääasiassa koodiesimerkkien kautta, joita on sittemmin modifioitu vastaamaan toteutettavaa sovellusta. Jatkokehityksessä se antaa mahdollisuuden pohtia tarkemmin, mitkä komponentit ovat parhaita tämän tyyppisille sovelluksille, mutta demo versiolle pyrin valitsemaan komponentit, jotka löytyvät React Native natiivi kirjastosta valmiiksi integroituna. Mobiilisovellus käyttää myös omia komponenttejaan, joiden avulla koodia voidaan jakaa pienempiin osiin, että eri komponentti kehyksien ympäristöstä tulee puhtaampi kokonaisuus ja koodia on helpompi lukea, kun jatkokehitykseen on tarvetta.

4.1 View

React Nativein yleisin komponentti on useiden alustojen tukema <View> komponentti, jota voidaan käyttää joustavasti käyttöliittymässä. Tätä komponenttia käytetään näkymän eri osien kartoittamiseen aivan kuten HTML-elementin elementin käyttämistä. Kuten iOS pohjaisessa koodissa se on <UIView> ja Android:ssa se on taas <View> (Eisenman 2017b.).

View komponentti on elementti, joka tukee Flexbox-asettelua, erilaisia tyylejä, kosketusohjaimia ja erilaisia helppokäyttötoimintoja. Se on natiivi komponentti, joka on alustasta riippumaton ja sen avulla tulostetaan näytölle näkymä View komponenttini sisällä olevasta sisällöstä. Sille voidaan sisentää tarvittaessa monia lapsi komponentteja sisäkkäin myös sen itsensä (Facebook Inc 2021d.).

Kuvassa 12 näkyy WordPress verkkokauppa demo sovelluksesta esimerkki käyttö aktiivisuusindikaattori komponentin piirtäjänä:

```
return (
    <View style={loadingStyle}>
        <ActivityIndicator size="large" color="#00ff00" />
    </View>
);
```

Kuva 12. View komponentin käyttö WordPress verkkokauppa demo mobiili sovelluksessa

4.2 FlatList

”FlatList” komponentin avulla voidaan helposti näyttää verkkokauppatuotteet listamaisessa näkymissä, jossa on kukin oma ”key” tunniste mistä React Native pystyy renderömään eri tuotteiden sisällöt eri ”props” datan sisällön kautta. Datan pitäisi olla listamainen ja muuttumaton attribuuteista riippumatta (Eisenman 2017a.).

WordPress verkkokauppa demo mobiilisovelluksessa FlatList komponentille tuodaan eri tuotteiden data, joka tulee WordPress API:ssta JSON formaattina ja ”renderItem” kehys palauttaa eri propseja käyttäen data, joka etsii ja tulostaa listan jokaisesta tuotteista käyttäen seuraavia attribuutteja otsikko, tuotteen seloste ja kuvan listamaisessa muodossa ruudulle.

4.3 useNavigaation

”useNavigaatio” toimi uudelleen käytettävien sisäkkäin olevine ”child” komponenttien kanssa vaivattomasti ja nopeasti, lähinnä kyseistä komponenttia käytetään eri näkymien välillä navigoimiseen ja seiltä takaisin eri ruutujen välillä. Tämä on yksinkertainen tapa tehdä navigointi käyttämällä React Nativen navigaatio komponenttia näin dokumentaatiossa kerrotaan (Facebook Inc 2021e.).

4.4 Touchable Opacity

Jokaisessa mobiiliapplikaatiossa käytetään nykyään pääsääntöisesti koskettamalla, jotakin nappia tai suurentamalla kuvina tai selaamalla vaikka artikkeleja. Monesti

kosketettavia elementtejä kuvataan napeilla, mutta aina niitä ei ole mielekästä käyttää, joten tätä varten React Nativessa on erilaisia kosketuskomponentteja (Facebook Inc, 2018g).

Worpress verkkokauppa demo mobiilisovelluksessa käytin ”Touchable Opacity” komponentitta tekee jokaisesta tuotenäkymästä kosketus kelpoisen, johon voidaan lisätä funktio, joka tässä tapauksessa tuo yksittäisen tuotteen ruudulle ja näyttää vain yhden ”child” komponentin osion ruudulla, koska kyseessä on React Nativen komponentti se toimii sekä iOS että Android laitteissa sen takia päädyin käyttämään sitä rankaisuissani.

4.5 Text

Lähes kaikissa sovelluksissa erilaisten tekstien piirtäminen on perustoiminto. HTML muodossa tekstin määritelmä on erilainen, mutta yleisin tapa on käyttää <p> elementtiä. Lisäksi voit käyttää ja -elementtejä määritelläksesi attribuutteja tekstissä. React Native mobiilisovelluskehitys ympäristöissä <Text> komponentti esittää eri pelkkänä raakana tekstinä, jota ei voida suoraan muokata HTML kaltaisilla muokkaus menetelmillä. Tarpeen tullen niitä voidaan muotoilla kukin erikseen tyyli komponenttien avulla ja tuomalle ne teksti komponenttien käyttöön (Eisenman 2017a.).

Mobiili demo sovelluksessa <Text> komponenttia käytin tekstien kokojen määrittämiseen otsikoissa ja tuotteen selosteen tekstin editoimisessa. Kerron enemmän komponenttien tyylittämisestä enemmän tyyli osiossa.

4.5.1 Regex

Regex on lyhenne säännöllistä lausekkeesta, jonka avulla voit luoda tekstiä, jotka auttavat vastaamalla haettavan termin etsimiseen ja paikantamiseen tekstissä. Perl on hyvä esimerkki ohjelmointikielestä, joka käyttää säännöllisiä lausekkeita. Sitä voidaan käyttää teksti editorissa tai komentokehotteessa (Computer hope 2020.).

```
{data.title.rendered.replace(/<\/?[^>]+(>|$)/g, "")}
```

Kuva 13. Regex ilmaisu otsikon siistimiseen

Kuvassa oleva säännöllinen lauseke siivoa kaikki html tagit pois JSON datasta ja palauttaa komponentille pelkästään tekstiä siistittynä. Käytin säännön luomiseen ”https://regexr.com/” sivua. Sääntö etsii jokaisetta rivistä ja poistaa kaiken sisällön, joka alkaa merkeillä ”<” ja lopettaa haun ”>” jälkeen ja React Native ei renderöi löydettyjä teksti dataa.

4.6 Tyylit

Kiani.fi netti selain sivustolla on käytetty monia eriä tyylejä ja animaation elementtejä, tässä rankaisussa ei tulla toteuttamaan samankaltaisia ratkaisuja, vain keskitytään yksinkertaiseen sovelluksen ulkoasuun toteuttamiseen.

Tyylittelyn hoitaa React Native ja sen natiivi StyleSheet komponentti, Felexbox jäsentää datan napakasti yhteen eri laatikoiden rajauksia käyttäen. Värimaailma on valittu visualisoimaan eri osiot kustakin näkymissä ”child” komponenttien tyylittelystä hoitaa ”parent” komponentti, ettei turhaa koodia johduttaisi kirjoittamaan moneen kertaan uudestaan. Näin varmistetaan yhdenmukainen tyylitys kussakin näkymissä.

4.6.1 StyleSheet

React Native elementit voidaan muotoilla useilla eri tavoilla. Tyylit voidaan sijoittaa yhteen elementtiin sisälle, tai sitä voidaan räätälöidä jokaiselle komponentille kontekstiin mukaisesti erikseen samassa JavaScript tiedostossa tai sen voi sijoittaa täysin itsenäiseen tyylitiedostoon. Toteutettavassa sovelluksessa päädyin määrittelemään tyylin komponentin kontekstissa, koska ainakin alkuvaiheessa minusta se oli selvempi. Jos lisäät erilaisia käyttöliittymäelementtejä, jotka toistetaan myöhemmin, sinun on ehkä määritettävä ne erikseen tyylitys template:ssa.

React Native ei käytä mitään oma erityistä kieltä sisällön tyylin määrittelemiseen, kuten CSS tyylien määrittelyyn, mutta syntaksin on hyvin samanlainen kuin CSS:ssä, mutta suurin ero on tyyli attribuuttien esitystavassa. React Native sovelluksessa tyylinimet kirjoitetaan yhdessä kuten on CamelCase tavassa, CSS:ässä käytettään tavuviivaa sijaan erottamaan sanat toisistaan, kuten kuvassa 14 on esimerkki mobiilisovelluksen kehyksen tyylityksen määrittävästä koodista.

React Native syntaksi

```
thumbnailContainer: {
  width: "90%",
  backgroundColor: "#c3cafd",
  marginVertical: 10,
  borderRadius: 10,
},
```

CSS syntaksi

```
thumbnailContainer {
    width: 90%;
    background-color: #c3cafd;
    margin-bottom : 10;
    border-radius : 10;
}
```

Kuva 14. Ract Native ja CSS tyylien syntaksista

Kuten oikeassa puolimmaisessa CSS kuvassa esimerkissä huomataan kaksoispisteen kadonneen tyylin määrittävissä kehyksessä ja attribuuttien kaksoispisteen jälkeen tulevissa arvoissa ja pilkut ovat korvattu puolipisteillä. Muiden CSS tyylinen välillä erittelyyn ei käytetä erityistä merkkiä kuten huomataan, pilkun kadonneen kehyksen sulkevasta kaarisulkeiden edestä.

On toki myös tilanteita, missä joudutaan muokkaamaan tyylejä vain kyseisen yksittäisen kehyksen elementissä siinä, on hyvänä tapana käyttää sen määriteltyä "inline" menetelmää. Tyylitys kirjoitetaan suoraan kehykseen sisälle, miten kuvassa 15 näkyy kahden kaarisulkeiden sisälle, koska ensimmäinen kaarisulkeet ovat React Native käytävää kehystä toinen on tarkoitettu CSS:lle se toteutetaan seuraavalla tavalla ”style={{attribuutteja}}”.

```
<Image
  style={{ width: "98%", height: 150, borderRadius: 10 }}
  source={{
    uri: data?._embedded["wp:featuredmedia"]["0"].source_url,
  }}
```

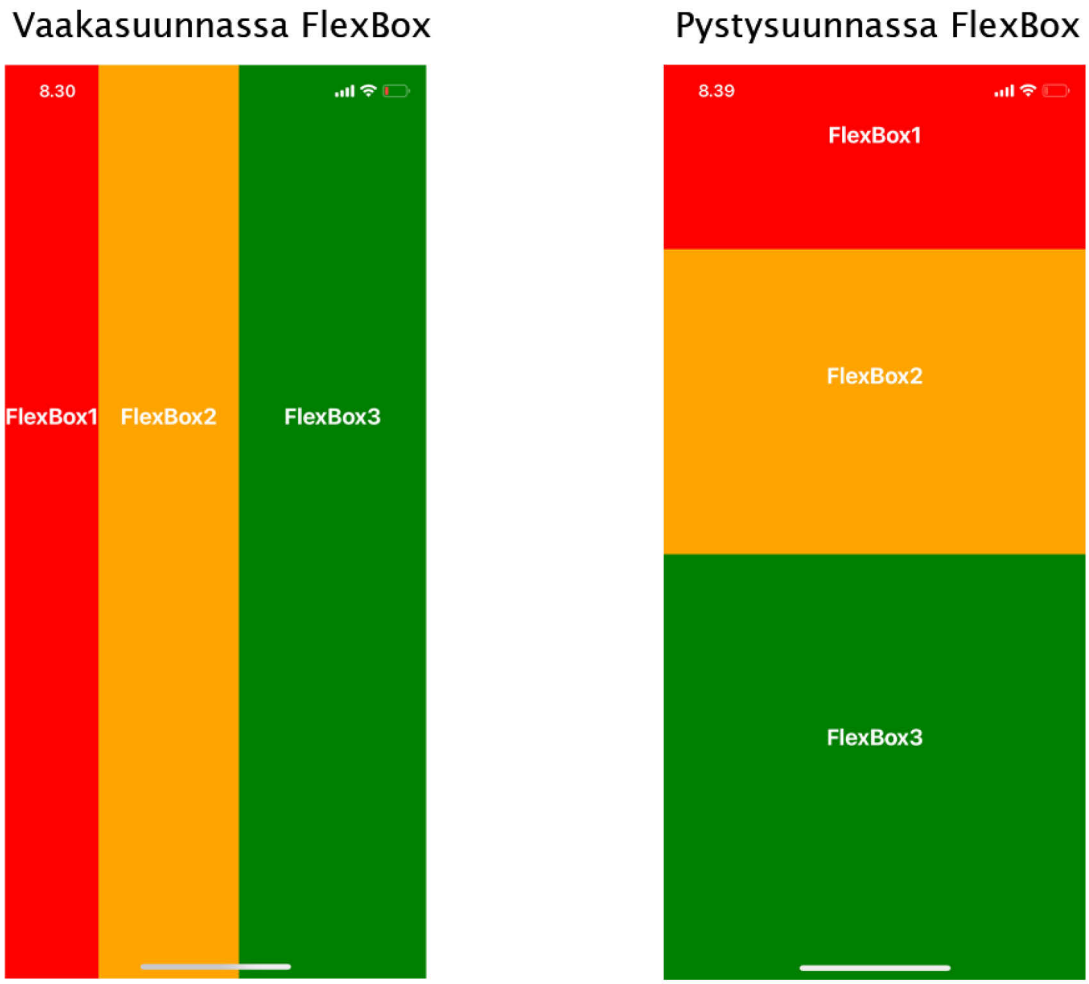
Kuva 15. Inline tyyli syntaksista

Style-attribuuttia tulisi kuitenkin käyttää varoen, koska se menettää tyylitiedoston edut; vastaavien elementtien tyylimääritelmä on tehtävä aina erikseen, tässä tapauksessa elementin tyyli voi olla erilainen ja muokattavuus vaikuttaa. Samaan aikaan, jos rivien välillä on monia tyylimäärittelyjä, tämä voi myös vaikuttaa koodin luettavuuteen negatiivisella tavalla.

4.6.2 Flex box

React Native -sovelluksessa oletusarvoisesti Flexboxia käytetään jakamaan sovelluksen sisällön näyttö, joten eri elementit voivat toimia näytön koosta riippumatta, eikä erillisiä mediakyselyjä vaadita, kuten yleensä käytetään verkkokehityksessä. Flexbox ei ole ainoa tapa tyylittää ja jakaa näyttö, mutta se on erittäin kätevä tapa katsella reagoivasti useilla eri näytöillä.

Flexbox toimii yleensä samalla tavalla kuin CSS, mutta muutamana erona on siinä, että elementit jaetaan ”flexDirection” kehyksen kanssa vaakasuunnan sijaan pystysuunnassa ja flex attribuutti hyväksyy vain yhden oletus arvon. (Facebook Inc 2020h.). Jos elementin kokoa ei määritellä erikseen leveys tai korkeusarvon perusteella, elementin koko määräytyy FlexBox:in oletus arvon perusteella. Kuvassa 16 on esimerkki Flexbox:in käytöstä näytön jakamiseen.



Kuva 16. FlexBox:en oletus näkymät

Flexbox:illa on myös useita muita tapoja käsitellä elementtien sijaintia näytöllä. Kun tarvittavat elementit on piirretty ruudulle, niiden sijainti voidaan käsitellä "justifyContent" kehyksellä, esimerkiksi kun ne eivät käytä kaikkea käytettävissä olevaa tilaa, FlexBox:in eri sisällöt voidaan kohdistaa vaakasuoraan ja "alignItems" kehyksellä ne voidaan taas tulostaa näytölle pystysuunnassa.

Demo sovelluksessa jaetaan eri tuotteiden näkymät ”row” kehyksellä. Jokainen tuote tulostuu ruudulle vaakasuunnassa. Tulevaisuudessa tullaan enemmän käyttämään navigoimisessa eri sivujen välillä ja sen sisällön näytölle tulostamisessa yksinkertaisesti Flexbox:sia käyttäen.

4.7 Mobiilisovelluksen API rajapinnan toteutus

Ratakisun toteuttamisessa päätin käyttää React Nativen ”Fetch” komponenttia, joka hakee ”get” pyynnöllä kiani.fi verkkokaupasta tuotteiden data JSON formaatissa ja sovellus pistää ne ”props”:eihin, josta data jaetaan eri komponenttien välillä. Axios on toinen keino hakea dataa WordPress REST rajapinnasta mutta, tämä ratkaisu käyttää yksinkertaista haku

menetelmää, jota kutustaan käyttäjän syöttämän tuote haun perusteella, täten ei ole tarvetta Axio:lle, joka pystyy käsittelemään monia eri hakuja samanaikaisesti eri attribuutteja käyttäen. (Kelhini 2019.).

Jos on tarvetta käyttää Axios:sta sen on asennettava Yarn paketti hallinnan järjestelmästä, sen jälkeen pyynnöt voidaan tehdä seuraavasti.

Esimerkki Axios:ksen datan tulostamisesta konsolin:

```
const link= https://demo.wp-api.org/wp-json/wp/v2/haetava/dataa

axios.get (link).then(response => {console.log(response.data);});
```

Esimerkki Fetch komponentin datan tulostamisesta konsolin:

```
const link= https://demo.wp-api.org/wp-json/wp/v2/haetava/dataa

fetch(link).then(response => response.json()).then(data => {console.log(data)});
```

Tässä mobiilisovellus ratkaisussa kiaani.fi WordPress REST rajapintaan otetaan yhteyttä seuraavaa osoitetta käyttäen” wp-json/wp/v2/product”, joka hakee kaikista tuotteista tiedot JSON muotoisena ja se palauttaa data, josta voidaan käsitellä data halutulla tavalla.

Käytin attribuutteja kohdentaakseni haku ruudulla haettavan termien kautta haettavissa tuotteiden datan tulostamisessa käyttäjien ruudulle. Etusiivussa on lista kaikista tuotteista, milloin käyttäjä rullaa 5 tuotetta alaspäin niin sovellus tekee uuden pyynnön seuraavasta 5 eri tuotteesta ja piirtää ne näytölle.

5 Pohdintaa

Opinnäytetyössäni toteutin mobiilisovelluksen Kiani Solutions yritykselle heidän WordPress verkkokaupasta kiani.fi, jossa myydään joka päivittäisiä tuotteita maailmanlaajuisesti. Kerroin tutkielmassani, miten React Native on saanut suosionsa ja miten sitä voidaan käyttää eri menetelmien avulla. Päädyin yksinkertaiseen ratkaisuun, jossa headless CMS runko toimii parhaiten sivujen staattiseen tapaan hakea tietoa ja tuoda sen käyttäjille käyttövalmiiksi eri laitteissa.

Minulla oli React osaamista, joka auttoi demo sovellusta tehdessäni; perehdyin enemmän React Native natiivi komponentein, jotka ovat käyttövalmiita ilman lisä konfiguraation tarvetta. Kerron miten esimerkkejä käyttäen eri tilanteista mitä käytin koodin toteuttamisessa. Lisäksi kerroin eri tavoista tyylittä eri kehyksiä koodissa.

Opinnäytetyön edistyi periaatteessa odotusten mukaisesti, mutta tilanne mitä eletään koronan takia, tuotti ongelmia työn aikatauluttamiseen ja sen noudattamiseen. Aikaa ei ollut paljon työn tekemiseen, seuraavalla kerralla pitää suunnitella paremmin aikataulutus ja aihe pitäisi olla selkeämpi ja yksityiskohtaisempi, mutta sain tehtyä yksinkertaisen esimerkki ratkaisun. Aluksi joudin perehtymään React Natiivin komponentteihin ja sen toiminnallisiin arkkitehtuureihin. Sen jälkeen suunnittelin ulkonäön sovellukselle; siitä sain tuotteet haettua ja tulostettua käyttäjän ruudulle onnistuneesti ja sain demo sovelluksen valmiiksi toimivana ratkaisuna.

Kiani.fi tarjoaa monia eri vaihtoehtoja laajan WordPress REST rajapintansa ansiosta, mistä voidaan hakea tietoja rajattomasti miltä tahansa alustoista. Työssä rajapinta oli edelleen kehitysvaiheessa, joten käyttöliittymä on muutettava. Jatkokehityksen ideana voi olla uusien näkymien tuomisen eri navigointi näkymillä ja lisäksi tuoda verkkokaupan toiminnalliset funktiot sovelluksen käytön, johon sisältyisi ostoskassi ja käyttäjien kirjautuminen omaan profiilinsa. Tällä hetkellä ollaan toteuttamassa käyttäjien sisäänkirjautumista varten tarvitsemista rajapinnan käyttö avaimista, mitä asiakas saisi turvallisesti ja salattuna tietonsa netin välityksellä verkkokaupasta mobiilisovellukseen ja sieltä takaisin verkkokauppaan.

Työssäni opin paljon mobiilikehityksestä ja huomasin, miten helposti React osaamista voidaan siirtää React Native oppimiseen ja yksinkertaistesti tehdä mobiilisovelluksen nopeasti ja vaivattomasti natiivi komponenteilla ilma mitään lisä kirjastojen asentamista. Monet kolmansien osapuolten React komponenttikirjastot tarjoavat samanlaisia kirjastoja React Native ohjelmalle muuntamisen helpottamiseksi. Suurin haaste siirtymävaiheessa on oppia uusia komentoja ja ymmärtää alustariippuvuudet.

Lähteet

Camden, R. & Rinaldi, B. 2017. Working with static sites. O'Reilly Media Inc, California. Luettu: 09.05.2021.

Computer hope. Regex. Luettavissa: https://www.computerhope.com/jargon/r/regex.html. Luettu: 22.5.2021.

Dominik Angerer. Headless CMS explained in 5 minutes. Luettavissa: https://www.storyblok.com/tp/headless-cms-explained. Luettu: 05.08.2021.

Expo.io, Introduction to Expo, Luetavissa: https://docs.expo.io/. Luettu: 10.05.2021
Yarn, Getting Started, https://classic.yarnpkg.com/en/docs/getting-started. Luettu: 10.05.2021.

Eisenman, B. 2017, Learning React Native, 2nd Edition, O'Reilly Media Inc. California. Luettavissa: a, https://learning.oreilly.com/library/view/learning-react-native/9781491989135/ch04.html. b, https://learning.oreilly.com/library/view/learning-react-native/9781491989135/ch02.html. Luettu: 19.5.2021.

Facebook Inc 2021a.https://reactnative.dev/docs/intro-react. Luettu: 25.04.2021.

Facebook Inc 2021b. https://reactnative.dev/docs/tutorial. Luettu: 25.04.2021.

Facebook Inc 2021c. https://reactnative.dev/docs/getting-started. Luettu: 09.05.2021.

Facebook Inc 2021d. https://reactnative.dev/docs/view. Luettu: 20.05:2021.

Facebook Inc 2021e. https://reactnavigation.org/docs/use-navigation. Luettu: 20.05:2021.

Facebook Inc 2021f. https://reactnative.dev/docs/touchableopacity. Luettu: 20.05:2021.

Facebook Inc 2021g. https://reactnative.dev/docs/touchableopacity. Luettu: 20.05:2021.

Facebook Inc 2021h. https://reactnative.dev/docs/flexbox. Luettu: 21.05:2021.

Faraz Kelhini, 2019. Axios or fetch(): Which should you use?. Luettavissa: https://blog.logrocket.com/axios-or-fetch-api. Luettu: 21.05.2021.

Ratnayake, N. 2017, Packt Publishing, WordPress Web Application Development, 3d edition. Luettavissa: https://www.oreilly.com/library/view/wordpress-web-application/9781787126800/b176fa4a-1921-442d-a25f-3a62c29752f5.xhtml. Luettu: 08.05.2021.

Shoutem 2016. A brief history of React Native. Luettavissa: https://medium.com/react-native-development/a-brief-history-of-react-native-aae11f4ca39. Luettu: 23.04.2021.

TechAhead 2020. The history of React Native: Facebook’s Open Source App Development Framework. Luettavissa: https://www.techaheadcorp.com/blog/history-of-react-native. Luettu: 23.04.2021

WordPress Org. 2021a. https://wordpress.org/about/features. Luettu: 08.05.2021.

WordPress Org. 2021b. https://developer.wordpress.org/rest-api/reference. Luettu: 08.05.2021

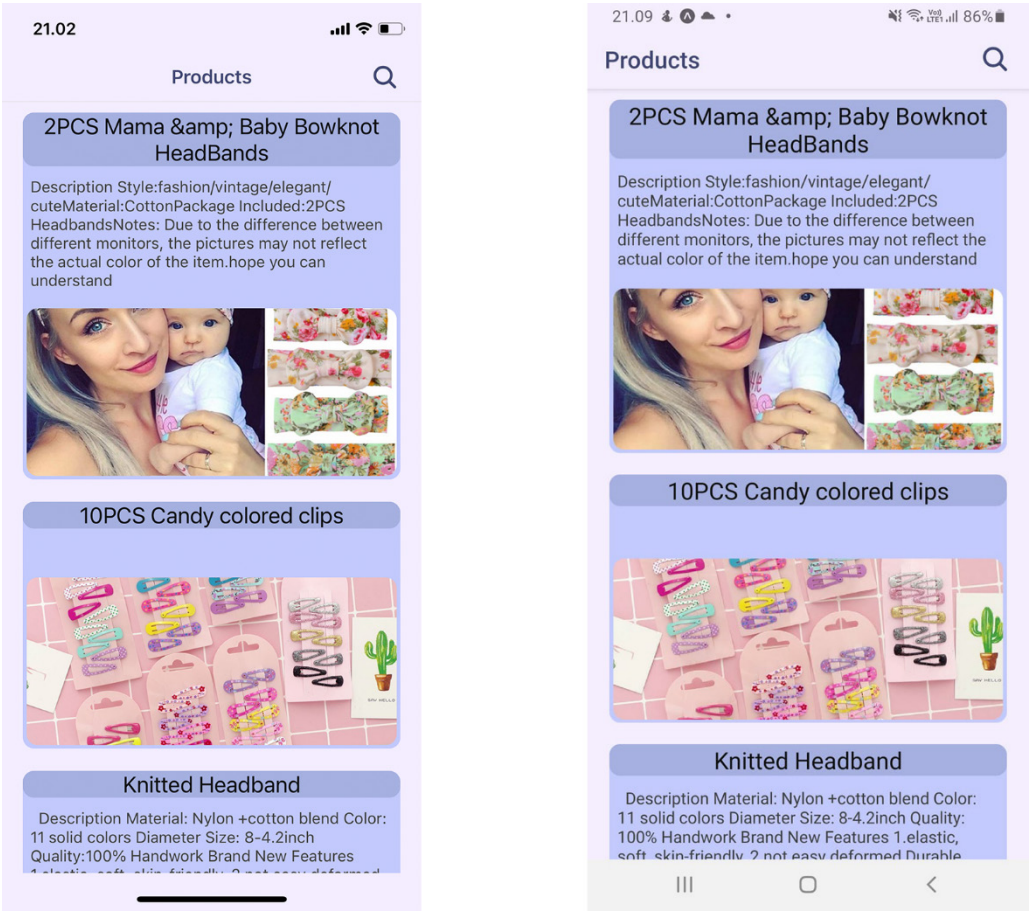
Liitteet

Liite 1. Termit ja lyhenteet

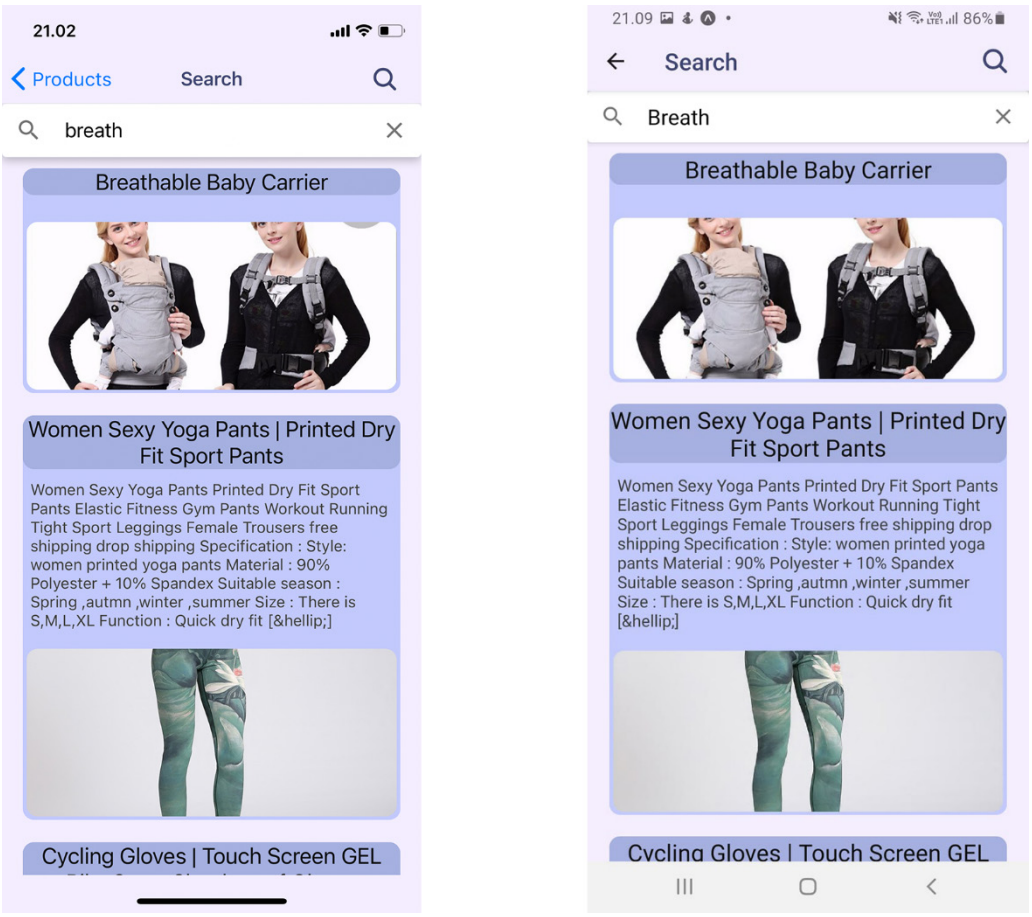
Front-end	Sovelluksen digitaalinen näkymä, jonka kautta käyttäjä navigoi älylaiteessa.
Back-end	Sovelluksen koodi varasto, johon käyttäjä ei pääse näkemään tai muokkaamaan.
API	Lyhenne sansata ”Application programming interface”, joka tarkoittaa ohjelmointirajapinta.
REST API	Lyhenne sanasta ”Representational State Transfer”, joka on arkkitehtuurimalli sovelluksien tuottamisessa.
CMS	Lyhenne sansata Content Management System, joka tarkoittaa sisällön hallintajärjestelmä.
NPM	Lyhenne sanasta Node Package Management, joka on paketin hallinnan järjestelmä.
JSON	Lyhenne sanasta JavaScript Object Notation, Se on dataa ulkoisesta lähteestä.
Headless	Sovellus, jossa front end on erotettu kokonaan back end:istä ja kommunikoi vain ulkoisen API:n kautta.
Render	Sovelluksen piirtäminen kätilöitymään toisin sanoen renderöinti.
CRNA	Lyhenne sanasta ”create react native app”, tapa luoda React Native sovellus.
CamelCase	Tapa kirjoita koodia alkukirjain pienellä muut sanat yhteen erotettuna isolla alkukirjaimella ilman välilyöntejä.

Liite 2. Lopputuloksen React nativen mobiilisovelluksen kuvakaappaus

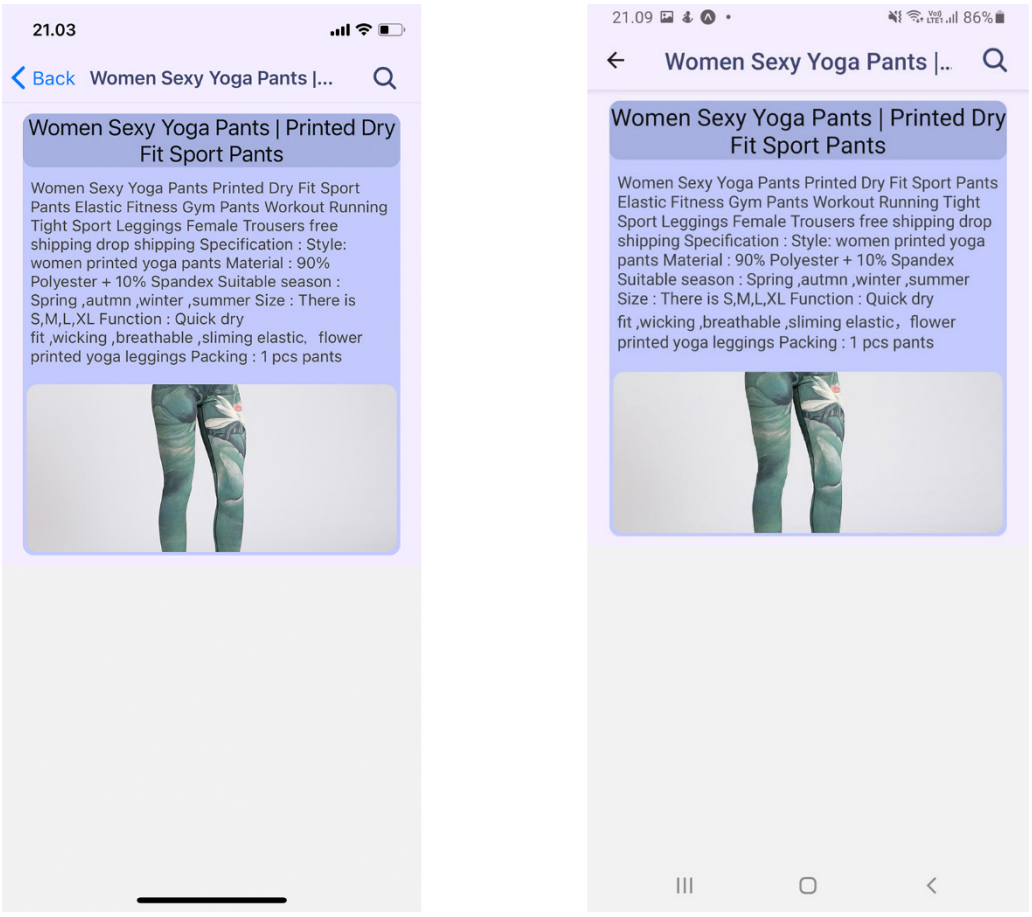
Kuvan kaappaukset ovat otettu fyysisistä laitteista. Vasemmalla puolella iOS älypuhelimesta”Iphone11 Pro” ja oikealla puolella Android älypuhelimesta ”Samsung Galaxy S9+”.



Kuva 1. Etusivu tuote näkymästä



Kuva 2. Tuotteen hakkuu näkymä, hakusananaan ”breath”



Kuva 3. Yksittäisen tuotteen näkymä