jamk

ReactJS piensovelluksissa

Lauri Virtanen

Opinnäytetyö, AMK
Huhtikuu 2022
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), tieto- ja viestintätekniikan tutkinto-ohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Virtanen, Lauri

ReactJS piensovelluksissa

Jyväskylä: Jyväskylän ammattikorkeakoulu. Huhtikuu 2022, 42 sivua

Tekniikan ala. Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Verkkojulkaisulupa myönnetty: kyllä

Tiivistelmä

Koska verkkosovellusten toteuttaminen, ylläpitäminen, testattavuus ja turvallisuus ovat korkealla tasolla, haluttiin tutkia pystyisikö ReactJS teknologian avulla pystyä korvaamaan natiiveja Android-sovelluksia. Tietoevryn toimeksiannosta toteutettiin tutkimustyötä ReactJS:n käytöstä Android-laitteissa. Työn tarkoituksena oli tutkia voiko ReactJS teknologiaa käyttää natiivien Android -sovellusten tilalla, ympäristöissä missä kaikki laitteet ovat Android-laitteita sekä löytämään mahdolliset haasteet mitä verkkosovellukset tuovat näihin ympäristöihin.

Opinnäytetyö toteutettiin kehittämistyönä, joka jakaantui useaan eri sovellukseen sekä useaan eri teknologian tutkimiseen ReactJS kehyksen sisällä. Opinnäytetyössä verrattiin useita yleisiä ratkaisuja tiedetyistä haasteista ja näistä saatiin tuloksia, joita pystytään käyttämään hyödyksi tulevaisuudessa.

Opinnäytetyössä löydettiin useita eri vaihtoehtoja toteuttaa kriittisiä ominaisuuksia, kuten autentikaatiota, tiedonsiirtoa verkkosovellusten ja Android-käyttöliittymän välillä sekä sovellusten alustusta erilaisten tarpeiden mukaan käyttäen valmiita toolchaineja mitkä käyttävät ReactJS teknologiaa.

Tutkimustulokset olivat lupaavia, työssä ei tullut vastaan ennalta tuntemattomia ongelmakohtia vaan pystyttiin ennustamaan mitkä asiat ovat ratkaistava. Tulosten pohjalta voidaan luottavaisin mielin lähteä kehittämään pienoissovelluksia Android-laitteiden ekosysteemeihin. Pystyttiin todeta, että ReactJS:n käytössä ei tule esiintymään kriittisiä ongelmia tai kompastuskiviä mitkä olisivat toiminnaltaan epävarmempia kuin natiivit Android-sovellukset. ReactJS:n avulla pystytään tekemään valmiin kehyksen, kriittisistä kirjastoista, joiden aloittaminen nopeuttaisi sovellusten aloittamista tyhjästä. Näiden ansiosta pystytään jatkossa kehittämään nopeasti sovelluksista ensimmäiset versiot, jotka voidaan laittaa tuotantolaitteisiin testiin nopeammin kuin natiiveilla Android-sovelluksilla.

Avainsanat (asiasanat)

React, ReactJS, JavaScript, Mobiilikehitys, verkkosovellus

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Lauri Virtanen

ReactJS in small applications

Jyväskylä: JAMK University of Applied Sciences, April 2022, 42 pages

Engineering and technology. Information and Communication Technology. Bachelor´s thesis.

Permission for web publication: Yes

Language of publication: Finnish

Abstract

Because web application development, maintenance, testing, and security are on elevated level, the goal was to explore if it is possible to replace small native Android-applications with ReactJS technology. Tietoevry gave the assignment to explore ReactJS technologies within production ecosystem which uses Android-devices. The aim was to get results which supports the goal of creating multiple small applications with ReactJS instead of native Android-applications.

Thesis was done using development work. Development work was divided into multiple small applications and playgrounds where it would be possible to assess known requirements for production proof applications. The applications could not have any issues with the known requirements which Android native applications can deal with.

The results were promising, none of the needed requirements were unmet and for some problems there were multiple viable solutions to choose from. Authentication, communication of the data between Android-device and the web-applications and setup using existing ReactJS toolchains gave plenty of options if requirements change or if they are needed for certain type of applications in the future.

The thesis shows that it is not only possible to choose ReactJS as valid technology inside Android ecosystem, but for some applications it will be inherently better because it removes some of the chunkiness from the development environment and it makes publishing fixes and updates faster. Especially if it is possible to create starting point from the technologies and libraries explored within this research, it is possible to make sizable chunk of the groundwork readily available to start other applications from scratch. The big reason for that is that no unknown problems arose from using ReactJS.

Keywords/tags (subjects)

React, ReactJS, JavaScript, Mobile development, Web development

Miscellaneous (Confidential information)

Sisältö

Kuviot

Sanasto

API

Application Programming Interface mahdollistaa kahden sovelluksen keskinäisen keskustelun.

Cache

Cache on suomeksi välimuisti, sillä tarkoitetaan yleensä esimerkiksi selaimessa selaimen välimuistiin tallennettuja tietoja. Kun tiedetään että kutsut ovat raskaita tai hitaita ja on hyödyksi käyttää cachetettua tietoa esimerkiksi samalla kun haetaan uudempaa tietoa.

CI/CD

Continuous integration/continuous delivery. Järjestelmä, joka mahdollistaa jatkuvan kehityksen ja julkaisemisen

JavaScript

Ohjelmointikieli, jolla toteutetaan internet-sivujen interaktiot

Git

Versionhallintajärjestelmä

HTML

HTML on standardi merkintäkieli

Integraatiotestaus

Testausta, jolla pyritään todentamaan, että yksittäiset komponentit toimivat myös laajemmassa kokonaisuudessa.

JSON

JavaScript Object Notation, tiedostoformaatti, jota käytetään datan tallennukseen ja lähettämiseen.

JSX

Tarkoittaa JavaScript XML, joka mahdollistaa HTML:n kirjoittamisen Reactissa.

Koodin jakaminen

Koodin jakaminen (Code splitting) erilaisiin paketteihin tai komponentteihin, jotka voidaan ladata tarvittaessa tai rinnakkain. Vaikka koodin määrä pysyisi samana, alkuperäistä latausta voidaan pienentää.

Merkintäkieli

Mekintäkieltä käytetään kuvaamaan tekstin rakennetta tai esitystapaa ja sillä pystytään erottamaan logiikka ja teksti toisistaan, englanniksi Markup language.

Mock

Mockilla tai mockkaamisella tarkoitetaan esimerkiksi verkkokutsun jäljitelmää, joka ei tee kutsua, vaan tekee rakenteeltaan identtisen jäljennöksen, jota voidaan käyttää hyödyksi esimerkiksi testaamiseen tai initialisointiin.

PoC

Proof of Conceptilla tarkoitetaan ratkaisun todennusta nopeasti tehdyllä versiolla sovellusta

PWA

Progressive Web App eli Progressiivinen verkkosovellus on verkkosivu, joka pyrkii toteuttamaan natiiveja ominaisuuksia, joilla pystytään toteuttamaan offline-toiminnallisuudet ja välimuistiin varastointi.

REST

Representational State Transfer. Arkkitehtuuri tyyli, jota käytetään kutsumaan API-rajapintoja.

Snapshot-testi

Snapshot-testillä tarkoitetaan toteutuksesta otettua tilannekuvaa, jota käytetään varmistamaan, että sovelluksen käyttöliittymä ei muutu odottamattomasti.

SPA

Single-page application, eli ns. yhdensivun sovellus, joka päivittää sivuston sisältöä olemassa olevan sivun päälle dynaamisesti.

SSO

Single sign-on, kertakirjautuminen

SSR

SSR tarkoitetaan Server-Side Rendered applications, jolla tarkoitetaan sivuja mitkä eivät erikseen lähetä kutsuja tiedon hakemiseen, vaan näytetään tietoa, joka on jo serverillä, joten voidaan luopua sekunteja kestävistä haku pyynnöistä.

Tokenisaatio

Tokenisaatiolla tarkoitetaan asioiden pilkkomista mahdollisimman pienelle ja abstraktille tasolle, jolloin ne ovat helpommin käytettävissä useisiin eri tarpeisiin.

Toolchain

Toolchain on ohjelmiston kehittämiseen tarkoitettujen työkalujen rypäs, jotka ovat integroitu toimimaan keskenään. Toolchainit ovat tekijänsä määrittelemät ja eivät aina sisällä samoihin käyttötarkoituksiin olevia työkaluja. Yleisimmät toolchainit pitävät sisällään kääntäjän, näiden työkalujen linkittäjän, koko ajan, testaajan sekä suoritusajon kirjastot.

UI

Käyttöliittymä

UX

Käyttökokemus

XML

XML tulee sanoista eXtensible Markup Language, joka ei itsessään tee mitään muuta kuin säilyttää ja siirtää tietoa.

XSS

Cross-site scripting mahdollistaa internet sovelluksen tietoturva aukon hyödyntämisen, sen avulla voidaan syöttää haitallista koodia verkkosivulle tai sivustolle tunkeutumiseen.

1 Johdanto

Nykyisessä maailmassa alalla kuin alalla, erilaiset työkalut löytyvät useimmiten taskussa olevasta älylaitteesta. Älylaitteiden sisällä olevan ohjelmiston pitäisi olla helposti ylläpidettävää, tietoturvallista, nopeaa ja helposti päivitettävissä.

Piensovellukset, jotka ovat suunniteltu helpottamaan yksittäisiä työtehtäviä työelämässä mahdollistaisivat nopean prototyyppien teon sekä nopean tavoittavuuden. Näiden sovellusten avulla pystyttäisiin auttamaan työntekijöiden päivittäisiä työtehtäviä ja parantamaan työntekijöiden työviihtyvyyttä.

1.1 ReactJS vai Natiivit sovellukset

Piensovelluksien toteutus web-teknologioilla mahdollistaa nopean käyttöönoton, koska sovelluksia ei tarvitse erikseen ladata, ne mahdollistavat helpon ja nopean päivittämisen. Ongelmakohdiksi muodostuu verkko sovellusten tarve nettiyhteydelle. Riippuvuus verkosta voi olla ongelmallista monissa sijainneissa joihin kuuluvuus katkeilee, mutta tämä ongelma pitäisi pystyä ratkaisemaan paremmilla tukiasemien sijoitteluilla ja muilla sisäverkon ratkaisuilla. Toisena ongelmakohtana on suhteellinen hitaus natiiveihin sovelluksiin nähden. Kun sovellusten koko pysyy pienenä ja prosessoinnin tarve jätetään natiiveille sovelluksille niin ongelma jää minimaaliseksi tai täysin olemattomaksi.

1.2 Tavoitteet

Tavoitteena on tutkia tarvittavat teknologiset ratkaisut tukemaan pienoissovelluksien nopeaa kehittämistä järjestelmiin, joissa on laajoja taustajärjestelmiä. Taustajärjestelmiä hyödynnettäisiin tekemällä kevyitä ja nopeasti päivitettäviä käyttöliittymiä joiden avulla työntekijöiden tehokkuus ja työmukavuus kohoaisi.

Työn tavoitteena on saada tarvittava tieto, kuinka pystytään tekemään nopeasti prototyypattavia sovelluksia. Sovelluksissa on huomioitava testattavuus, käyttöliittymän ja käyttökokemuksen yhtenäistäminen, ylläpidettävyys, julkaistavuus, päivitettävyys ja luotettavuus. Tavoitteena on havaita

ongelmia, jotka johtuvat ReactJS teknologian valinnasta. Tämän lisäksi tutkitaan voisiko ongelmakohdat olla helpompia muilla teknologioilla sekä tarkistellaan, onko ReactJS:ssä mahdollisia etuja, jotka nopeuttavat kehitystyötä.

Työn aikana mahdolliset toteutuneet sovellukset, eivät ole tutkimuksen kohteita vaan sovellusten kehitystyössä saadut johtopäätökset ja tulokset ovat työn kannalta merkityksellisiä asioita.

1.3 Tutkimusmenetelmä

Soveltava tutkimus tarkoittaa tiedon hankkimista ongelman ratkomisella ja ratkaisujen löytämistä, joka tapahtuu toteuttamalla sovellustyötä. Tutkimuksessa käytetään aikaisempaa tietoa, jota hyödyntäen ratkaistaan työtä edeltävät ongelmat.

Opinnäytetyöhön käytetään tutkimusmenetelmänä soveltavaa tutkimusta. Tutkimuksen aikana käytetään olemassa olevia sovelluksia testauskehyksinä, sekä opinnäytetyötä varten tehtyä sovellusta. Uusilla sovelluksilla pystytään testaamaan haluttuja asioita, joista voidaan tehdä luotettavia johtopäätöksiä. Työhön käytetään pääasiassa internetistä löytyvää aiheeseen liittyvää aineistoa.

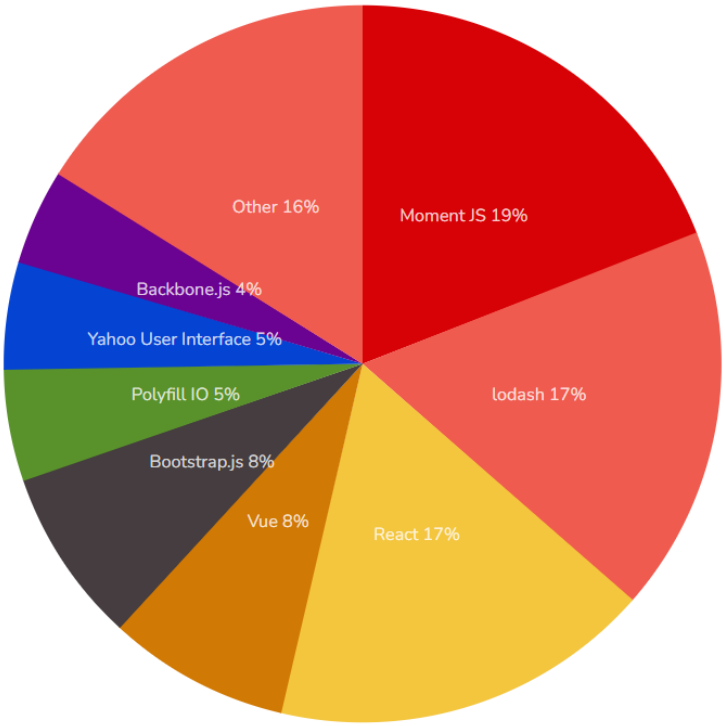
2 ReactJS kirjasto

2.1 Mikä on ReactJS

ReactJS on yksi maailman suosituimmista JavaScript kirjastoja (ks. Kuvio 1), sen avulla pystytään tekemään interaktiivisia käyttöliittymiä, jotka ovat kevyitä ja nopeita. ReactJS on komponentti pohjainen, jossa jokainen erillinen komponentti pystyy ylläpitämään omaa tilaa. ReactJS:n ei tarvitse tietää mitä taustajärjestelmiä on käytössä ja ei ole riippuvainen mistään tietystä taustajärjestelmästä. (React 2022.)

JavaScript Library Usage Distribution in the Top 1 Million Sites

Statistics for websites using JavaScript Library technologies



Top In JavaScript Library Usage Distribution in the Top 1 Million Sites

Technology	Websites	%
Moment JS	164,115	16.41
lodash	149,659	14.97
React	147,963	14.8
Vue	70,352	7.04
Bootstrap.js	67,803	6.78
Polyfill IO	43,266	4.33
Yahoo User Interface	41,529	4.15
Backbone.js	37,064	3.71
Prototype	22,090	2.21
script.aculo.us	17,707	1.77
MooTools	16,867	1.69
Socket.IO	16,124	1.61
Preact JS	15,407	1.54
TypeScript	11,577	1.16
KnockoutJS	10,657	1.07
jsTimezoneDetect	9,529	0.95
Enquire JS	6,125	0.61
mOxie	3,542	0.35
Marionette.js	2,629	0.26

Kuvio 1. JavaScript kirjastojen käyttäjämäärät (Trends buildwith, 2022.)

ReactJS:ä on mahdollista käyttää niin isossa osassa verkkosivun toteutusta, kuin halutaan. Se on helppo ottaa testiin, jos on olemassa oleva verkkosivu ja haluaa tutustua sen toiminnallisuuksiin. (Getting started with React 2022.)

Kun luodaan uusi verkkosivun toteutus tyhjästä ja haluamme, että monet kriittiset asiat on otettu huomioon ReactJS tarjoaa toolchaineja. Toolchaineissa on valmiit ratkaisut testattavuuteen, ylläpidettävyyteen, skaalautuvuuteen ja tuotantoympäristön optimointiin. (Create a new React App 2022.)

2.2 Reactin toolchainit

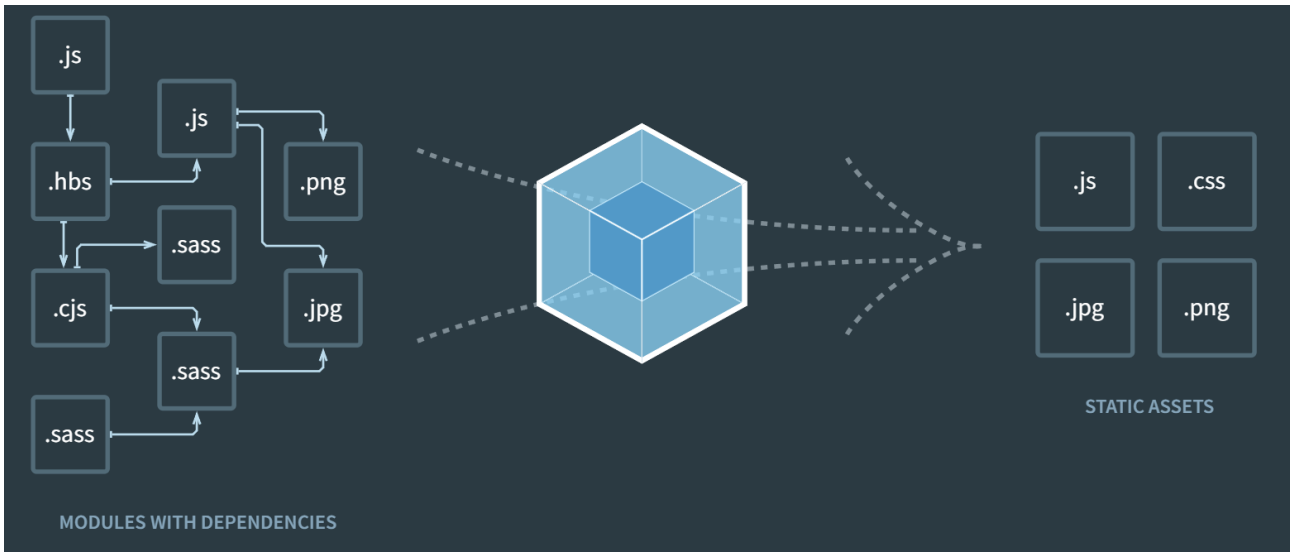
Koska kokonaisen verkkosivun tekeminen ReactJS:llä vaatii paljon tekijöitä, kuten koodin paketoinnin, kääntäjän ja koodin jakamisen. (Create a new React app 2022.) ReactJS:llä tehtyyn laajempaan verkkosovellukseen valitaan tai tehdään oma toolchain, jossa on nämä tarvittavat ominaisuudet ovat sisään rakennettuja.

Kun tiedetään, mitä mahdollisia käyttötarkoituksia haluamillamme sovelluksilla on, voidaan rajata yleisimmistä toolchaineistä meidän käyttöömme soveltuva ja aloittaa sovelluskehitystyö nopeasti. Näin ei tarvitse tehdä pienoissovellusta varten suurta pohjatyötä teknisten mahdollistajien tutkimiseksi. (Create a new React app 2022.)

2.2.1 Create React App

Create React App on hyvä ympäristö tutustua ReactJS:ään ja se on parhaita tapoja aloittaa uusi SPA ReactJS:llä. Se asettaa kehitysympäristön käyttämään uusimpia JavaScript-ominaisuuksia ja mahdollistaa nykyaikaisen kehitysympäristön. Tämän lisäksi Create React App optimoi sovelluksen tuotantoversion, sovelluksen kokoamisvaiheessa. Create React App:llä luodut sovellukset renderöidään pelkästään käyttäjän puolella. (Create a new React app, 2022.)

Create React App ei vastaa eikä ota kantaa taustajärjestelmiin eikä tietokantoihin, joten Create React Appin kanssa voidaan käyttää haluttuja taustajärjestelmiä. Create React App:n käyttää kääntäjänä Babelia sekä paketoijana webpackia, jonka toimintaperiaate voidaan nähdä kuviosta 2, nämä ovat konfiguroitu valmiiksi taustalle. Niiden käyttöönottoa ei tarvitse erikseen tehdä, joka auttaa siinä, että Create React App on mahdollista ottaa saman tien käyttöön. (Create a new React app, 2022.)



Kuvio 2. Webpack (Webpack. N.d.).

2.2.2 Next.js

Next.js on suosittu ja kevyt sovelluskehys, jota käytetään staattisiin ja serveripuolella renderöityihin sovelluksiin. Se tarjoaa tyyli- ja reititys toteutukset suoraan ja olettaa, että serveriympäristö on tehty Node.js:llä. (Create a new React app 2022.)

Next.js:n avulla pystytään käyttämään käyttöliittymän ohjelmoinnissa Reactia, tämän päällä Next.js:llä pystytään ratkaisemaan yleisiä sovellusten vaatimuksia, kuten tiedon haun, reitit ykset ja erilaiset integraatiot muihin ympäristöihin. Näiden toiminnallisuuksien lisäksi Next.js tarjoaa parannellun kehitys- ja käyttäjäkokemuksen. (What is Next.js 2022.)

Next.js:ssä on toteutettu intuitiivinen sivukohtainen reititys, joka tukee dynaamisia reittejä. Siinä on myös useita suorituskykyä parantavia toimintatapoja, isoimpana näistä on se, että oletuksena

sivut renderöidään etukäteen, joka auttaa myös hakukone optimointia. Tämän lisäksi Nextissä on automaattinen koodin jakaminen. (What is Next.js 2022.)

2.2.3 Gatsby

Gatbsy on React-pohjainen avoimen lähdekoodin kehys (Gatsby documentation, 2022). Se antaa kehittäjän käyttää Reactin komponentteja mutta kääntää sen HTML- ja CSS-tiedostoiksi, jotka tekevät sillä tehdyistä sivustoista nopeita ja kevyitä ladata. Koska sillä luodut verkkosivut ovat staattisia tiedostoja, voidaan sillä luodut sivut hostata CDN:llä koska se ei tarvitse sovellus servereitä tai tietokantoja. (Why Gatsby?, 2022.)

Gatsbyn tekijöiden (Gatsby Vision, 2022) mielestä nykyinen verkkosivujen kehittämisen ekosysteemi nähdään kymmenen vuoden sisällä vanhentuneena ja verrattavissa symboliseen konekieleen ja konekieleen. Heidän tarkoituksenansa on ollut rakentaa seuraava korkeamman tason ohjelmointikieli. Abstraktioiden tarkoituksena on luoda yksinkertaisempi kehitysympäristö, minkä avulla voidaan luoda entistä hienostuneempia ratkaisuja. (Mts.)

2.2.4 Joustavammat ja itsetoteutetut toolchainit

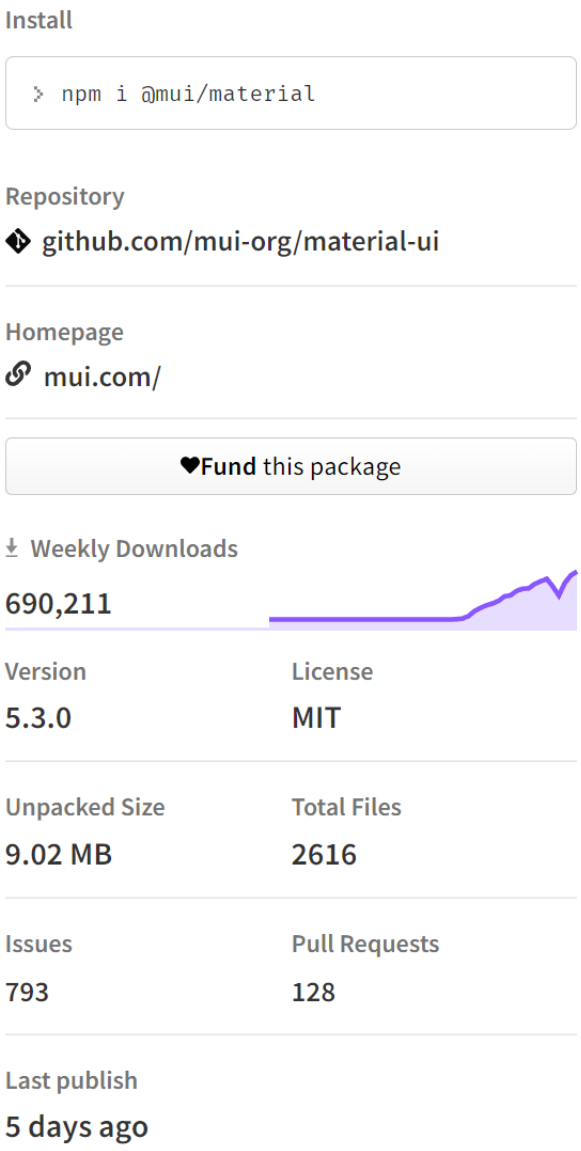
Reactia käyttäviä toolchaineja on useita muita, jotka on tarkoitettu kokeneille käyttäjille ja tiettyihin käyttötarkoituksiin. Esimerkkinä voidaan ottaa Razzle, se on kehitysympäristöltään verrattavissa Create React Appiin. Razzlea käyttäessä jää omaksi valinnaksi sovelluskehyksen muut arkkitehtuuriset päätökset kuten reititys ja datan hakeminen. Razzle abstraktoi kaikki kompleksisuutta vaativat konfiguraatiot, joita tarvitaan SPA- ja SSR-sovelluksiin ja tekee siitä vain yhden riippuvuuden. (Razzle, 2022.)

Pienempien toolchainien lisäksi on myös mahdollista luoda oma toolchain. Yleisesti ottaen tämä sisältää paketin hallinnan kuten Yarnin tai npm:n, ne antavat mahdollisuuden pystyä asentamaan ja päivittämään laajasta ekosysteemistä kolmannen osapuolen kirjastoja. Paketoijan kuten webpack tai Parcel, joka antaa kehittäjän kirjoittaa modulaarista koodia, joka kootaan yhteen pieniksi paketeiksi optimoidakseen lataus aikoja. Kääntäjän kuten Babel, joka antaa kehittäjän kirjoittaa modernia JavaScriptiä, joka toimii myös vanhempien selaimien kanssa. (Create a New React app, 2022.)

2.3 Ulkoiset paketit

ReactJS mahdollistaa helposti toisten tekemien pakettien käyttämisen npm:n avulla. Npm on maailman isoin ohjelmisto rekisteri, jota käytetään ympäri maailman ja sitä voidaan käyttää niin sisäiseen kehittämiseen kuin julkisten pakettien julkaisemiseen. (About npm, N.d.)

Riippuen projektin luonteesta ja vaatimuksista, pystytään ottamaan npm:n kautta tarvittavia toiminnallisuuksia. Käytettävien pakettien hyötynä on se, ettei yleispäteviin toiminnallisuuksiin tarvitse tehdä itse. Käytettyjen pakettien tärkeimmät vaatimukset ovat kehittäjien luotettavuus ja kuinka monta käyttäjää paketeilla on, joka kerrotaan npm:n sivuilla latauksia viikossa muodossa (ks. Kuvio 3). Yleisimmät paketit kertovat tiedossa olevista ongelmista, päivitystavoista ja Reactin version päivityksessä tulleista yhteensopivuus ongelmia.

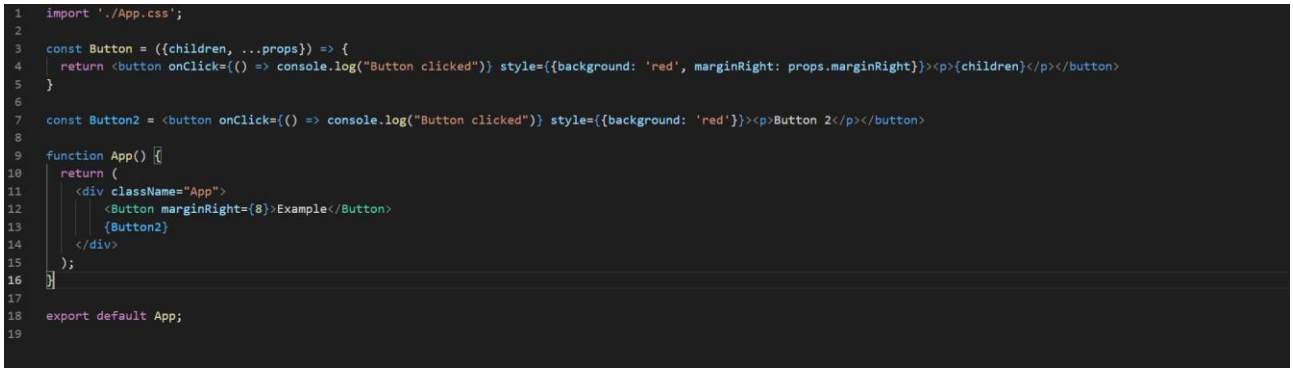


Kuvio 3. Npm paketin päätiedot (Npm example. N.d.).

2.4 JSX

JSX pohjautuu JavaScript-kieleen ja sitä suositellaan käytettäväksi ReactJS:n kanssa, kun määritellään miltä käyttöliittymä näyttää.

Kuviosta 4 voidaan nähdä, miten JSX:n avulla saadaan luotua käyttäjäystävällisiä komponentteja, jotka ovat myös uusiokäytettäviä. Esimerkin Button-nimiselle komponentille pystytään antamaan lapsi sekä ominaisuuksia, joiden mukaan tehdään haluttuja asioita.



```
import './App.css';

const Button = ({children, ...props}) => {
  return <button onClick={() => console.log("Button clicked")} style={{background: 'red', marginRight: props.marginRight}}><p>{children}</p></button>
}

const Button2 = <button onClick={() => console.log("Button clicked")} style={{background: 'red'}}><p>Button 2</p></button>

function App() {
  return (
    <div className="App">
        <Button marginRight={8}>Example</Button>
        {Button2}
    </div>
  );
}

export default App;
```

Kuvio 4. JSX esimerkki

JSX tuo myös turvallisuutta käyttäjän syöttämään tietoon, estämällä käyttäjän mahdollisen injektion koodiin. Tämä onnistuu, koska kaikki käännetään merkkijonoksi ennen kuin ne renderöidään. Tämän avulla pystytään estämään XSS-hyökkäyksiä, jotka ovat asiakaspuolella injektoituja skriptejä, joilla pystytään ohittamaan pääsyn valvonnan luomia rajoitteita. (JSX, 2022.)

Babel-kääntäjä kääntää JSX:llä luodun koodin React.createElement() -kutsuksi, Kuvion 5 elementit ovat identtisiä.

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

```
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

Kuvio 5. JSX ja React.createElement()

2.5 Create React Appista PWA

PWA-sovellukset tarjoavat verkkosovelluksille monia paikkauksia puutteisiin, joita niissä on verrattuna puhelimien natiivi sovelluksiin. PWA-sovellukset tarjoavat seuraavat edut normaaliin verkkosivuun verrattuna, hakukone optimointi, jolla saadaan parannettua näkyvyyttä metadatan avulla, asennettavuus laitteille, linkitettävyys, offline-toiminnallisuudet, sovellusalustan natiivit toiminnallisuudet kuten ilmoitukset, responsiivisuus, turvallisuus ja selain tuen. (Mozilla PWA introduction, 2022.)

2.5.1 Alustus

Create React App tarjoaa keinon tehdä sovelluksesta progressiivisen verkkosovelluksen. Alkuperäisessä konfiguraatiossa offline-tila ja välimuistin varastointi on kytketty pois päältä. Oletuksena Create React Appin service worker ei tule ulkoisten kutsujen väliin. Eikä se tallenna välimuistiin cross-origin-liikennettä kuten API-kutsuja, kuvia tai sivustolle lisättyjä muiden sivustojen upotettuja sisältöjä. (CRA PWA, 2022.)

PWA-sovellus sisältää oletus konfiguraation manifest.json-nimisessä tiedostossa, joka tarjoaa sovellukselle ikonin, nimen ja brändivärityksen mitä käyttää, kun sovellus on näytöllä. Manifest-tiedosto on kehittäjien muunneltavissa sekä siihen on hyvät ohjeet, mitä mikäkin parametri kuviossa 6 tarkoittaa ja mitä muita mahdollisia parametreja sovellukselle voidaan tämän avulla antaa. (CRA PWA, 2022.)

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

Kuvio 6. manifest.json

Alkaen Create React Appin versiosta 4 on ollut mahdollista lisätä service-worker.js-niminen tiedosto sovelluksen kansioon, joka mahdollistaa tuen Workboxin InjectManifest-liitännäisen. InjectManifest kokoaa service workerin ja injektoi siihen listan osoitteita. (CRA PWA, 2022.)

Kun luodaan uusi projekti, jolle halutaan PWA-toiminnallisuus, voidaan luoda uusi Create React App käyttäen templaatteja konsolikomennolla (ks. Kuvio 7).

```
npx create-react-app my-app --template cra-template-pwa
```

Kuvio 7. Konsoli komento CRA templaatille

Templaatilla luotu sovellus luo halutut tiedostot ja kansiorakenteet. Tässäkin tapauksessa offline-tila ja cacheen-tallennus ominaisuudet ovat alustavasti kytkettynä pois päältä, mutta yhden rivin muutoksella ne saadaan otettua käyttöön. (CRA PWA, 2022.)

2.5.2 PWA:n hyödyt ja haitat

Service workerin toiminnallisuudet ovat oletuksena pois päältä, koska se tuo turhauttavia kehitys-kokemuksia. Sovelluksen tallentaessa kaikki välimuistiin, ilman välimuistin tyhjennystä kehitysympäristö ei vastaa kirjoitettua koodia. Näistä johtuen on aiheellista, että oletuksena nämä asetukset eivät ole päällä, koska enemmän nämä aiheuttavat ongelmia vasta-alkajille tai jos asetuksesta ei olla tietoisia. Service worker on mahdollistettu vain tuotantoympäristössä, edellä mainittujen ongelmien takia. Kun on syitä testata offline-toiminnallisuutta paikallisesti, Suositeltu tapa on ajaa komento ”npm run build” jonka jälkeen Create React App antaa ohjeita, kuinka jatkaa. (CRA PWA, 2022.)

Koska käyttäjät eivät ole välttämättä tottuneet offline-ensin sovelluksiin, kannattaa käyttäjille ilmoittaa sovelluksen toimivan offlinessa. Päivityksestäkin on hyvä ilmoittaa, että uusi versio tulee toimintaan, kun olemassa olevat välilehdet suljetaan. Ilmoittamiseen ei ole Create React Appin puolelta tehty toiminnallisuutta, joten se jää kehittäjien tehtäväksi toteuttaa haluamallaan tavalla. (CRA PWA, 2022.)

Service workerit tarvitsevat HTTPS:n muualla paitsi localhost:lla, joten kehitystyö onnistuu ilman HTTPS:n konfigurointia localhostiin. Kun tuotannossa ei ole HTTPS-tukea niin sovellus kuitenkin toimii, vaikka mitkään service workerin tuomat hyödyt eivät ole toiminnassa.

3 ReactJS:n testaus

3.1 Yleistä ReactJS:n testauksesta

ReactJS:n komponenttien testaaminen on samankaltaista kuin muun JavaScript-koodin testaaminen. ReactJS:n testaus voidaan jakaa kahteen ryhmään, komponenttipuiden renderöinti yksittäisten toiminnallisuuksien ja käyttöliittymien testaamiseen yksinkertaistetussa testaus ympäristössä sekä kokonaisen sovelluksen testaaminen E2E-testeillä. (ReactJS testing, 2022.)

E2E-testeillä pystytään estämään ja havaitsemaan tärkeiden toiminnallisuuksien regressiota pitkin sovelluskehitystä, näillä testeillä ei testata yksittäisiä ReactJS:n komponentteja vaan sitä, että kokonaisuus toimii kuten pitää. (ReactJS testing, 2022.)

ReactJS:n testaamiseen on useita erilaisia kirjastoja ja ajajia, jokaisessa on omat hyvät ja huonot puolet, joita täytyy verrata oman sovelluskehyksen tarpeiden mukaan. (ReactJS testing, 2022.)

3.2 Testien ajajat

ReactJS:ssä on mahdollista käyttää monia eri testin ajajia, kuten Jest, mocha tai ava. Ne mahdollistavat testien kirjoittamisen puhtaalla JavaScriptillä ja ne ajetaan osana kehitys prosessia sekä jatkuvan kehityksen yhteydessä ts. versionhallinnassa. Kun testivarmuus heikkenee siitä, että testit ajetaan virtuaalisessa ympäristössä eikä esimerkiksi oikeassa selaimessa, pitää testien ajajat valita sen mukaan. (ReactJS testing, 2022.)

Yleiset testien ajajat ovat hyviä yksikkötestaukseen tai isompien komponenttien testaamiseen ReactJS:ssä, mutta jos halutaan ajaa E2E-testejä pitää testaus ympäristö olla sitä varten erikseen rakennettu. E2E-testit vaativat pitempiä työnkulkuja eritoten, kun testataan kriittisiä toiminnallisuuksia, kuten kirjautumista tai maksamista. Näitä varten voidaan käyttää erilaisia testauskehyksiä, kuten Cypress, Playwright tai kirjastoja kuten Puppeteer, jolla voidaan navigoida useiden reittien läpi testaten myös taustalla tapahtuvia asioita ja sivuvaikutuksia, joita tapahtuu selaimessa. (ReactJS testing documentation, 2022.)

3.3 Jest

Jest tulee valmiiksi asennettuna Create React Appissa ja se on Node.js-pohjainen testausajaja, joka tarkoittaa sitä, että testit ajavat aina Node-ympäristössä, eikä oikeassa selaimessa. Tämä nopeuttaa iterointinopeutta ja parantaa testien epävarmuutta. (CRA running tests, 2022.)

Jestin avulla pystytään tekemään erilaisia mockeja esimerkiksi kutsuista tai sivu historiasta. Kuviosta 8 voidaan huomata, millä tavalla on tehty Jestin avulla mockkeja kolmannen osapuolen paketin toiminnoista. Tämän avulla pystytään luomaan haluttuja parametreja verkko osoitteeseen, joita luomamme sovellus tarvitsee. Näiden ei tarvitse olla paikkaansa pitäviä parametrejä, joista sovellus saisi käytettyä oikeita hakutuloksia, kunhan parametrien muoto on oikea. (Jest mocking, 2022.)

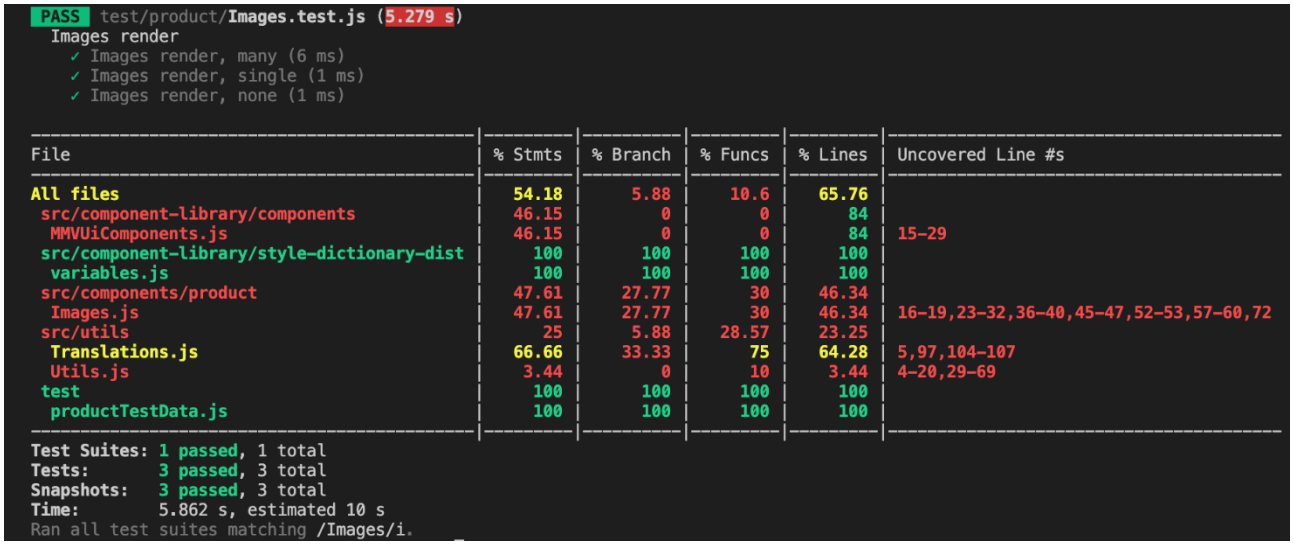
Kuviosta 8 huomataan, että Jestillä voidaan luoda jäljitelmä selaimen historiasta ja luoda siihen tarvittavat Jestin omat tyhjät funktiot. Tämän ansiosta sovelluksen tarvitessa selaimen historiaa tai sijaintia voidaan kutsua näitä jo luotuja tyhjiä funktioita, jotta pystytään testaamaan sovelluksen toiminnallisuutta. (Jest mocking, 2022.)

```
let mockLocation = { search: "?outlet=123" };
let mockHistory = { goBack: jest.fn(), push: jest.fn(), replace: jest.fn() };

jest.mock('react-router-dom', () => ({
    ...jest.requireActual("react-router-dom"),
    useLocation: () => mockLocation,
    useHistory: () => mockHistory,
    useNavigate: jest.fn()
}));
```

Kuvio 8. Jest mock funktiot

Jestin testeihin voidaan lisätä parametrinä –coverage, jonka avulla saadaan testien tuloksista eriteltynä listana mitä on testattu ja miten laajasti on testattu (ks. Kuvio 9), tämän avulla nähdään nopeasti mitä tiedostoja on testattu ja mikä on testien kattavuus.



Kuvio 9. Jestin testien tulokset

3.4 Setup

Jest tarkistaa kaikki tiedostot, jotka on nimetty tietyllä tavalla ja pitää niitä testi tiedostoina. Suositeltuna tapana on laittaa testit samaan kansioon kuin tiedostot, joita niillä testataan, jotta tiedostojen tuonti helpottuu testitiedostoissa (CRA running tests, 2022.).

Testit on hyvä laittaa toimimaan versionhallintaan, jotta versionhallintaan ei saada sovellusta laitettua siinä kunnossa, ettei se läpäise testejä. Tämä helpottaa piensovellusten testaushelppoutta, jos sovellusta luodaan testit edellä, jotka testaavat sitä mitä lopputuloksen pitää vastata. Esimerkkinä voidaan ottaa sivu missä on esimerkki teksti niin voidaan kirjoittaa testi, joka odottaa, että sivulta löytyy tämä testi ja antaa epäonnistuneen testituloksen niin kauan kunnes sivulta ei löydy tekstiä. Tätä logiikkaa voidaan laajentaa toiminnallisuuksiin ja kokonaisiin komponentteihin.

4 Kohdeympäristö

4.1 Kohdeympäristön vaikutus kehitykseen

Verkkosivuna toteutetut sovellukset mahdollistavat sen, että sitä pystytään käyttämään jokaisella laitteella missä on internet yhteys ja internet selain. Tästä huolimatta, kun tiedetään mitkä ovat tärkeimmät tuotantoympäristön laitteet, kannattaa varmistaa, että sovellukset on optimoitu ulkoasullisesti sekä toiminnallisesti kyseisillä laitteilla.

Kohdeympäristön ollessa tiedossa, tiedetään mihin toiminnallisuuksiin loppukäyttäjä on tottunut, ja mitkä ovat käyttäjälle tuttuja käyttökokemus polkuja. Halutut toiminnallisuudet saattavat vaikuttaa joidenkin piensovellusten teknologiavalintoihin.

4.2 Android-laitteet

Piensovelluksia käyttävät laitteet tulevat olemaan mobiililaitteita, joissa on Android versio 10 sekä kehitystiimin hallussa oleva yritysselain ja työpöytä toteutus. Tämän takia sovelluskehityksessä ei tarvitse huomioida vanhempia laitteita, eikä sitä miten tietyt ominaisuudet toimivat muissa selaimissa.

Koska Android-ympäristön yleiset käyttökokemukset ovat tunnettuja, voidaan tehdä piensovelluksen suunnittelussa teknologisia rajauksia yksinkertaisten testien perusteella. Kun on tiedossa, että piensovellusten ei tarvitse keskustella käyttöympäristön kanssa niin voidaan tehdä päätös, että kyseinen sovellus voidaan tehdä ReactJS:n avulla. Kun tiedetään, että sovelluksen pitää pystyä keskustelemaan muiden sovellusten kanssa joko ottamalla tai lähettämällä komentoja toisiin sovelluksiin, tulee vastaan useita teknologisia haasteita.

4.3 Datan käsittely Android-sovellusten ja ReactJS sovelluksen välillä

Android-kehityksen ohjenuorien mukaisesti, tiedon jakaminen sovellusten välillä olisi hyvä tehdä Android-jakonäkymän kautta. Koska sovelluksia käyttävät tuotantolaitteet ovat Android-laitteita voidaan tehdä isojakin toiminnallisuuksia, jotka eivät suoraan toimi muissa ympäristöissä. (Android guide for sending, 2022.)

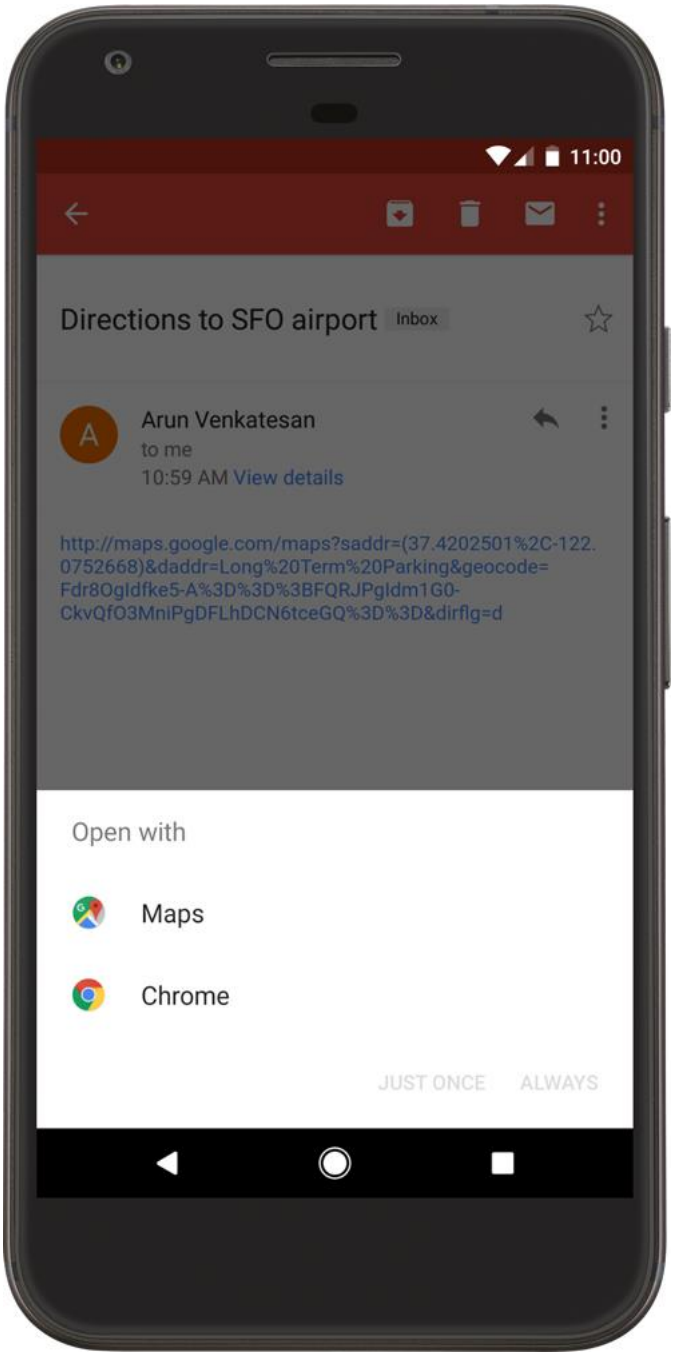
Käyttöliittymästä on hyvä piilottaa tai ilmoittaa käyttäjälle toimimattomista toiminnallisuuksista. Mikäli on tiedossa, että loppukäyttäjät eivät käytä sovellusta muissa ympäristöissä niin toiminnallisuudet voidaan jättää näkyviin kehitystyön helpottamisen varjolla.

4.3.1 Android App Links

Android tarjoaa kolme erilaista tapaa tehdä sisäisiä sovellusten linkityksiä. Näiden avulla voidaan määritellä oletus sovelluksia toimintojen avaamiseen, verkkosivun avaamista sovellusten sijasta,

jos sovellusta ei ole asennettu ja muilla tavoin helpottaa käyttäjän pääsemistä halutun sisällön luokse. (Android guide for app links, 2022.)

Deep linkeiksi kutsutaan niitä URI:a, jotka vievät käyttäjän tiettyyn kohtaan sovelluksessa eikä vain sovelluksen ensimmäiseen näkymään. Näitä luodaan intent filtereillä, jotka mahdollistavat oikean sovelluksen aukaisemisen joko jakonäkymän (ks. Kuvio 10.) avulla tai suoraan. (Android guide for app links, 2022.).



Kuvio 10. Jakonäkymä (Android guide for app links, 2022.).

Verkkolinkit, ovat kuin Deep Linkit, jotka käyttävät HTTP ja HTTPS protokollia. Androidin versiosta 12 alkaen nämä verkkolinkit avautuvat aina selaimessa. Aikaisemmissa versioissa jos sivusta oli asennettu PWA-sovellus laitteelle, avautuu näkyviin jakonäkymä mistä käyttäjä voi valita avaako sovelluksen selaimessa vai suoraan asennetussa sovelluksessa. (Android guide for app links, 2022.)

Android App Linkit ovat verkkolinkkejä, jotka käyttävät myös HTTP ja HTTPS protokollia ja niillä on autoVerify-ominaisuus. AutoVerify viestii järjestelmälle, että sen pitäisi verifioida kuuluuko sovellus URL-tunnukselle intent filttereissä ja avataanko se oletuksena määriteltyyn linkkiin. (Android app links verification, 2022.) Tämän avulla voidaan suoraviivaistaa käyttäjäkokemusta, kun ei tarvitse näyttää kuviossa 10 nähtyä jakonäkymää vaan voidaan halutuissa tapauksissa navigoida suoraan sovellukseen (mts.).

Android App Links on turvallinen ja tarkkaan määritelty, se tarjoaa sulavan käyttäjäkokemuksen koska sovellus tarjoaa saman HTTP URLin sovellukselle ja verkkosivulle. Mikäli käyttäjällä ei ole sovellusta asennettuna voidaan avata verkkosivu ilman virheilmoitusten näyttämistä käyttäjälle. (Android guide for app links, 2022.)

4.3.2 Tietokanta

Tietokannan avulla, voidaan käyttää samoja API-rajapintoja Android-sovelluksessa ja verkkosivulla, jolla voidaan tallentaa ja hakea tietoa samasta tietokannasta. Tietokannan käytön etuna on se, että jos tiedon pitää olla pysyvämmin tallessa, niin tietokannasta hakemalla saadaan tietoa haettua myös myöhemmin.

Tietokantojen asentaminen ja ylläpitäminen vaatii huomattavasti enemmän työtä ja huolellisuutta verrattuna muihin tapoihin, mutta hyvänä puolena on se, että tietoa voidaan näyttää alustariippumattomasti ja voidaan itse määritellä teknologiat mitä halutaan käyttää ja osa näistä tarpeista voi olla jo olemassa sovelluksen ekosysteemissä, kun tälle on tarvetta.

4.3.3 Web socketit

Web sockettien avulla luodaan reaaliaikainen tietovirta. Web socketit ovat edistynyttä teknologiaa verkkosivujen tekemisessä, sillä voidaan avata kaksisuuntainen interaktiivinen keskustelukanava käyttäjän selaimen ja taustajärjestelmien kanssa. (Mdn web socket dokumentaatio, 2022.)

Web sockettien avulla voidaan lähettää ja vastaan ottaa tietoa, jota olisi muuten vaikea siirtää reaaliajassa erilaisten ympäristöjen välillä. Web socketit vaatii taustalle serverin, joka kuuntelee tiet-

tyä porttia TCP-serverillä, joka seuraa tiettyä protokollaa. Serveri voidaan kirjoittaa millä vain serveripuolen ohjelmointikielellä, joka tukee Berkeleyn-socketteja, kuten C, C++ ja PHP. (Mdn web socket server documentation, 2022.)

Web socketin serverillä voidaan tehdä useita toimintoja kuten; tarkistaa yhteydettä ja vastausnopeutta, viestin pilkkominen, tietosisällön dekoodaus ja tiedon lukeminen ja tiedon paljastaminen.

4.3.4 Custom

Kun on mahdollista tehdä muutoksia ympäristön selaimeen ja sitä kautta lähettämään Android-laitteen työpöydälle viestin minkä avulla pystytään lähettämään Android-intentin, niin ei tarvitse miettiä monimutkaisia taustajärjestelmiä yksinkertaisten viestien käsittelyyn.

Yksinkertaisimmillaan pystytään luomaan toiminnallisuuden verkkosovellukseen (ks. Kuvio 11.), joka kutsuu selaimen toiminnallisuutta (ks. Kuvio 12.) ja lähettää tälle haluttua tietoa.

```
import { IconButton } from '@material-ui/core';
import { Share } from '@material-ui/icons';

export default function ShareString(props) {
    const shareString = () => {
        if (typeof CustomBrowser !== 'undefined')
            CustomBrowser.shareString(props.string); // eslint-disable-line
    }

    return (
        <IconButton id="shareBtn" edge="start" color="inherit" onClick={shareString} aria-label="share">
            <Share color="primary"/>
        </IconButton>
    );

}
```

Kuvio 11. Verkkosivun toiminto datan lähetykseen.

```
@JavascriptInterface
fun shareString(data: String){

    try {
        // send broadcast intent including string, custom launcher will catch it
        val intent = Intent()
        intent.action = BARCODE_SHARING_ACTION
        intent.putExtra(BARCODE_SHARING_DATA , data)

        context.sendBroadcast(intent)

        // move browser to back
        val activity = context as Activity
        activity.moveTaskToBack( nonRoot: false)
    } catch (e: Exception ) {}

}
```

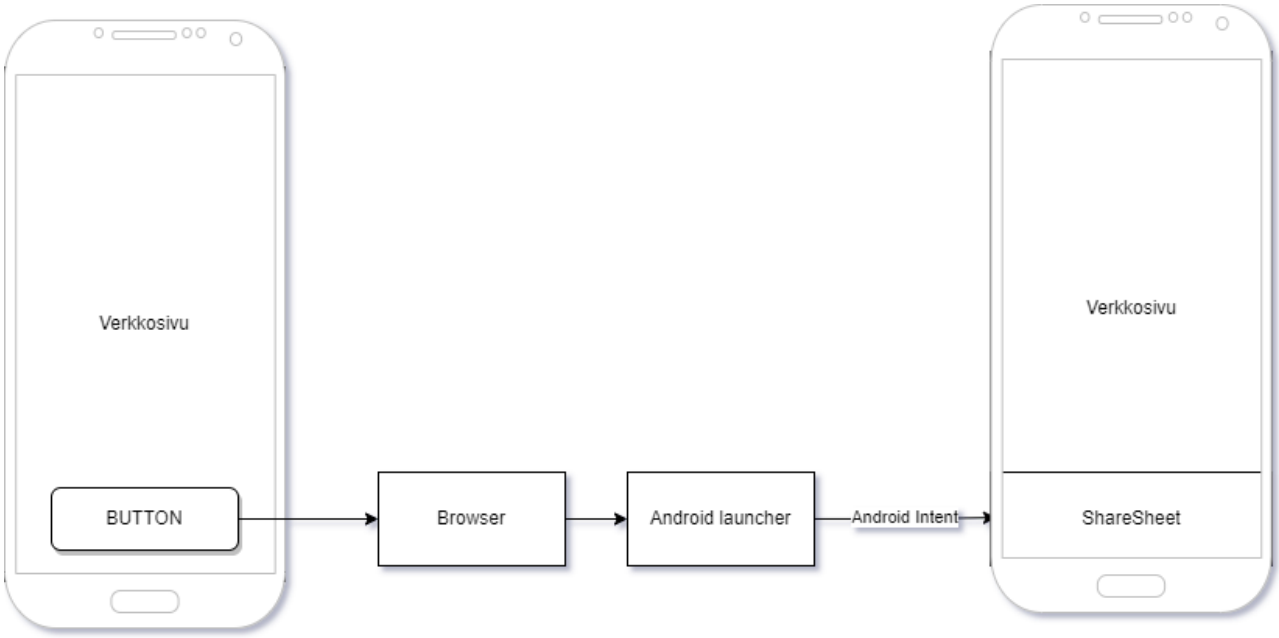
Kuvio 12. Selaimen toiminto tiedon jakamiseen

Selaimen toiminnallisuus datan siirtämiseen selaimesta Androidille on myös hyvin yksinkertainen, sitä varten tarvitsee tehdä uusi Android-intent, jolle annetaan action- ja datakenttä. Android-työpöydän päässä voidaan määritellä vastaanottaja mikä tekee toiminnallisuuden saadessaan actionin nimen, joka on määritelty BARCODE_SHARING_ACTION:n nimiselle ominaisuudelle kuviossa 13.

```
internal const val BARCODE_SHARING_ACTION = "com.organization.project.launcher.share_string"
internal const val BARCODE_SHARING_DATA = "com.organization.project.launcher.string_value"
```

Kuvio 13. Intentin action ja data

Android työpöydän saadessa intentin, jonka nimi osuu määrittelemään nimeen, voidaan tehdä työpöydällä intentillä ja sen tiedoilla mitä halutaan. Tämän jälkeen voidaan avata takaisin se verkkosivu, josta on tultu työpöydälle tai työpöydän avaamaan uuteen sovellukseen. Tämä nopeuttaa ja helpottaa yksinkertaisien kutsujen lähettämistä, mutta se vaatii pääsyn muokkaamaan työpöytää, selainta ja verkkosivua. Ilman pääsyä yhteenkin näistä pitäisi tukeutua muihin ratkaisuihin.



Kuvio 14. Jakonäkymän flow

Kuviosta 14 voidaan havainnoida, miten yksinkertaisesti tietoa pystytään liikuttamaan sovellusten ja teknologioiden välillä, mikäli on pääsy sovellusten lähdekoodiin. Viestien lähettäminen verkkosovelluksesta toiseen verkkosovellukseen tai natiiviin sovellukseen muuttuu yksinkertaiseksi, eikä ratkaisu tarvitse verkkoyhteyttä tiedon siirtämiseen.

5 Piensovellusten UI/UX

5.1 Miksi yhtenäinen UI/UX-toteutus

Useiden piensovellusten mahdollistaminen, vaatii useasti tuotantoympäristöissä sen, että käyttöliittymät ja käyttökokemukset menevät sovellusten rajojen yli. Toisin sanoen piensovellusten välillä liikkuminen pitää tuntua saumattomalta eikä sitä pysty käyttöliittymästä kertomaan, että olemme eri sovelluksessa. Tämän pitäisi pystyä onnistumaan, vaikka kehittäjänä olisi eri kehittäjä tai eri organisaatio.

Päästäksemme sovellusten väliseen harmoniaan, taustalle tarvitaan järjestelmiä mitkä mahdollistavat sen, että käyttöliittymille ja käyttökokemuksille löytyy yksi ainoa totuuden lähde. Pienissä kokonaisuuksissa näiden järjestelmien tekeminen on verrattain yksinkertaista, mutta jos taustalla pyörii kymmeniä erilaisia sovelluksia sekä useita eri organisaatioita, jotka tuovat tuotteensa

ekosysteemiin niin tällaisen design-kokonaisuuden käyttöönotto on oltava suoraviivaista ja nopeaa.

5.2 Piensovellusten UI/UX-toteutus

Piensovellusten tekemiseen käytetään Amazonin luomaa Style Dictionaryä. Style Dictionaryn avulla pystytään luomaan tyylit kerran riippumatta alustasta tai kielestä, joita otetaan käyttöön eri ympäristössä Design tokenien avulla. Design tokenit mahdollistavat nopeat UI-muutokset esimerkiksi erilaisten kampanjoiden ajaksi ja mahdollistavat erilaiset brändäykset sovellusten sisällä.

Design tokeneille voidaan konfiguroimalla antamaan monia useita tiedostomuotoja (ks. Kuvio 17.) ulos samasta lähteestä, tämän avulla pystytään tekemään ilman duplikaatti työtä käytännöllinen tapa käyttää tokenisaatiota useiden alustojen mahdollistamiseen. (Style Dictionary documentation, 2022.)

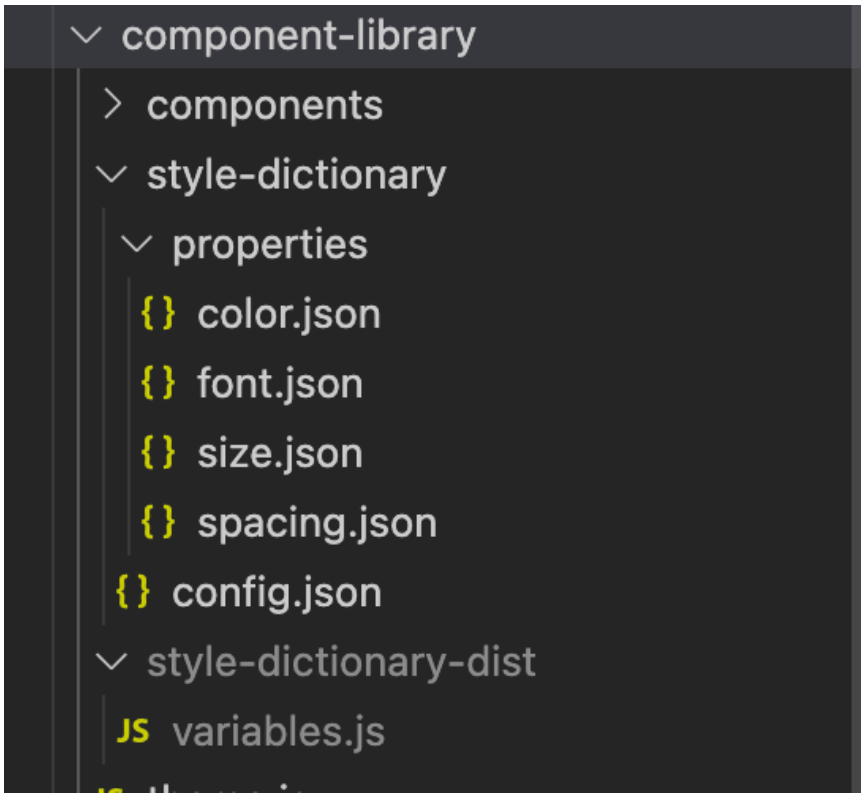
Ekosysteemin ollessa Android-sovellukset ja ReactJS:llä tehdyt verkkototeutukset sekä mahdollisesti monet muut tulevaisuuden sovellukset, tallaisen tokenisaation alustaminen mahdollisimman nopeasti ennen useamman sovelluksen tekemistä on hyödyllistä. Samaan projektiin, samalle asiakkaalle tai samaan organisaation tehdyt sovellusekosysteemit kuitenkin pääsääntöisesti noudattavat samoja väri skeemoja ja käyttöliittymä ja -kokemus filosofiaa.

Style Dictionary voidaan konfiguroida (ks. Kuvio 17.) käymään läpi kaikki JSON-tiedostot halutuista kansioista, kun määritellään source-kenttä konfiguraatio tiedostossa. Konfiguraatio tiedostossa määritellään mihin, missä muodossa ja missä formaatissa haluttu ulosanti halutaan. Kuviossa 17 on määritelty, että ulosanti tulee JavaScript-muodossa variables-nimiseen tiedostoon ja sen formaatti on javascript/es6. Tällä tavalla saadaan tiedoston tyylit muotoon, joka on yhtenäinen sovelluksen kanssa (ks. kuvio 15).

```
{
  "spacing": {
    "small": {
      "value": 4
    },
    "medium": {
      "value": 8
    },
    "large": {
      "value": 16
    }
  }
}
```

```
export const FontLineHeightLarge = "44px";
export const LayoutWidthFull = "100%";
export const ButtonHeightLarge = "48px";
export const ButtonWidthFull = "100%";
export const IconSizeBase = "16px";
export const IconSizeSmall = "16px";
export const IconSizeMedium = "32px";
export const IconSizeLarge = "96px";
export const SpacingSmall = 4;
export const SpacingMedium = 8;
export const SpacingLarge = 16;
```

Kuvio 15. Tokenien tekeminen vasemmalla ja oikealla esimerkki ulosannista



Kuvio 16. Komponenttikirjaston ja Style Dictionaryn rakenne esimerkki

```
{
  "source": ["./src/component-library/style-dictionary/properties/**/*.json"],
  "platforms": {
    "js": {
      "transformGroup": "js",
      "buildPath": "./src/component-library/style-dictionary-dist/",
      "files": [
        {
          "destination": "variables.js",
          "format": "javascript/es6"
        }
      ]
    }
  }
}
```

Kuvio 17. Style Dictionaryn configuraatio tiedosto

5.3 Tokenisaation ylläpidettävyys

Style Dictionary mahdollistaa sen, että ylläpidettävyys pysyy hyvin pienenä, kun tyyleille löytyy yksi totuuden lähde. Kun Style Dictionarystä muokataan tyylejä niin tyylit päivittyvät automaattisesti kaikissa sovelluksissa, joissa ne ovat käytössä. Tämä mahdollistaa helposti esimerkiksi sovellusten sisäiset kampanjat tai brändin muutokset.

Koska Style Dictionaryllä tehdyt tokenit ovat todennäköisesti käytössä useassa eri sovelluksessa, tarvitaan käytännöllinen tapa toteuttaa paketin hallintaa, mistä on helppo ja nopea ottaa muutoksia muissa sovelluksissa käyttöön.

Npm tarjoaa yksilöille oman ja tiimeille sekä organisaatioille oman hinnoittelun, joka mahdollistaa loputtoman määrän julkisia ja yksityisiä paketteja sekä paketti tai tiimi kohtaisen oikeuksien hoidon. Npm:n avulla pystytään asentamaan ja ottamaan käyttöön paketteja kuvio 18 mukaisella komennolla.

```
npm install sax@0.1.1
npm install @myorg/privatepackage@1.5.0
```

Kuvio 18. Npm version asentaminen (Npm commands, N.d.).

Toisena vaihtoehtona on tehdä versionhallintaan sovellus, mikä otetaan käyttöön halutuissa sovelluksissa, joilla on pääsy samaan versionhallinnan ympäristöön. Tätä varten jokaiselle sovellukselle, joka haluaa käyttää pitää määritellä pääsy oikeudet versionhallinnan asetuksista, tällöin voidaan asentaa npm:n komentorivin komennolla suoraan versionhallinnasta (ks. Kuvio 19.).

```
npm install "git+ssh://git@ssh.gitlab.xxx.fi:port/organization/project/application.git#semver:^1.x.x" --save
```

Kuvio 19. Npm install versionhallinnasta

5.4 Style Dictionaryn testattavuus

Style Dictionaryn avulla luotujen komponenttien testaus ei eroa ReactJS:llä luotujen komponenttien testauksesta. Komponentteja ei tarvitse testata Style Dictionaryä vasten, vaan komponentteja testaan toiminnallisuuksiltaan. Näiden lisäksi käyttöliittymää voidaan testata snapshot-testeillä, jotka varmistavat, että käyttöliittymät näyttävät siltä miltä halutaan.

6 Piensovellukset tuotantoympäristössä

6.1 Varhainen tuotantoon julkaiseminen

Työn tekemisen aikana ei ollut tavoitteena saada sovellusta vietyä todelliseen tuotantoympäristöön, vaan sovellukset jaettiin muutamalle laitteelle, jotka ovat tuotantoa vastaavassa. Tällä tavalla luotiin ketterän kehityksen ympäristö, mihin pystytään päivittämään nopealla syklillä sovelluksia eikä tarvitse aina tehdä tuotantokelvollisia päivityksiä. Tämä oli kriittinen tapa toimia, että pystytään kokeilemaan erilaisia kirjastoja ja tapoja oikeilla loppukäyttäjillä.

6.2 Autentikointi

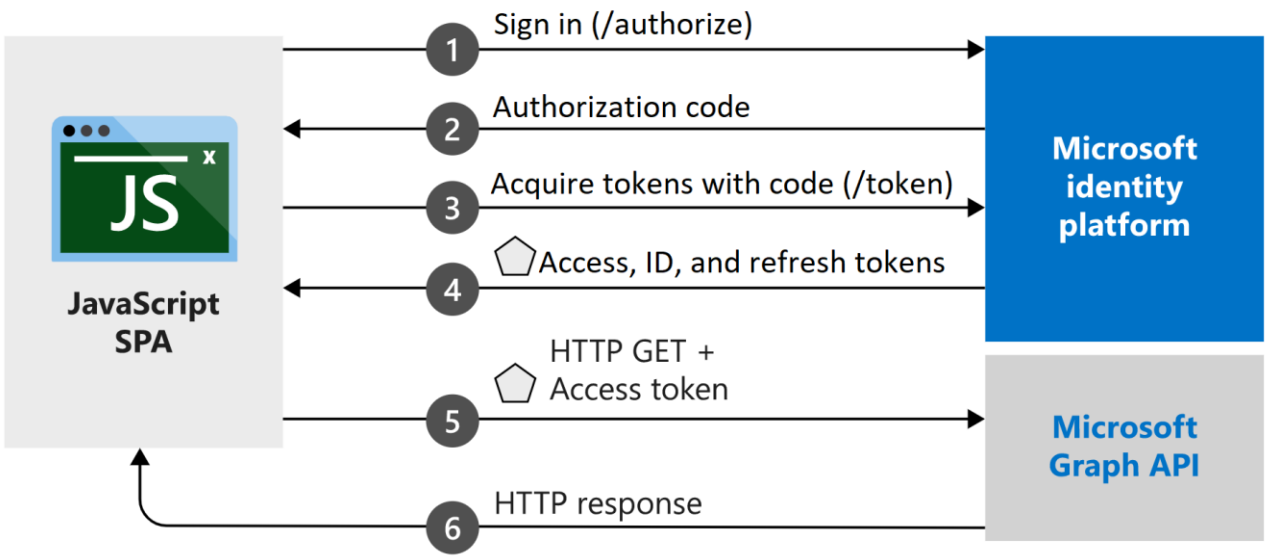
Loppukäyttäjät kirjautuvat Azure AD:n kautta tuotantolaitteisiin sisään, tämä mahdollistaa sen, ettei piensovelluksissa tarvitse erikseen kirjautua, mutta se on silti autentikoinnin takana käyttäen Azure AD:n SSO:ta.

SSO:n avulla voidaan taustalla lähettää kutsu taustajärjestelmille, mitkä varmistavat, että käyttäjä on kirjautunut ja että käyttäjällä on oikeus käyttää sovellusta. Tämän avulla pystytään ryhmittelemään käyttäjille näkyvyys sovelluksiin Android käyttöjärjestelmässä.

Autentikointi on hyvä tehdä pienissä sovelluksissa mahdollisimman huomaamattomasti taustalla, koska useiden sovellusten välillä liikkuminen olisi muutoin täysin mahdotonta. Reactin puolella pitää varmistaa, ettei sovellukseen pääse sisälle ilman voimassa olevaa autentikaatio tokenia ja tämän autentikoinnin avuksi on useita erilaisia paketteja npm:ssä.

6.3 Microsoftin autentikaatio kirjasto (MSAL)

MSAL:a voidaan käyttää autentikoinnissa apuna, kirjasto on Microsoftin ylläpitämä, joten se on looginen vaihtoehto, kun käytetään Azure AD:ta. Kuviosta 20 nähdään MSAL:n avulla tehdyn tutoriaali sovelluksen perustoiminta periaate tämä on hyvä esimerkki siihen, mitä kutsuja tarvitsee tehdä autentikaatiota varten. (MSAL documentation, 2022.)



Kuvio 20. MSAL (MSAL documentation, 2022.).

MSALista saadaan vaadittavat toiminnallisuudet autentikaatiota varten, se tarjoaa mahdollisuudet kirjautua sekä uudelleenkirjautua käyttäen ponnahdusikkunoita, uudelleen ohjauksia ja komponentin, jolla päästään näihin käsiksi. Tämän lisäksi MSALilla on mahdollista pyytää istuntotunnusta taustalla, jota tarvitaan API-kutsuja varten. (MSAL documentation, 2022.)

Hiljainen istuntotunnuksen kutsuminen parantaa sovellusten käyttökokemusta, kun tunnus uusitaan ilman, että käyttäjälle näytetään kirjautumislomaketta, jolla käyttäjä voi jatkaa kirjautuneena oloa. Hiljainen tunnuksen uusiminen tarkistaa ensimmäisenä cachetetun, voimassa olevan tunnuksen. Toisena se käyttää refresh-tokenia hankkiakseen uuden istuntotunnuksen. Joissain tapauksissa pitää varmistaa, ettei käyttäjä pääse jatkamaan ilman kirjautumissivun tarvetta, esimerkiksi pääsyoikeutta johonkin resurssiin mitä ei saa näyttää tai ladata muuta kuin varmasti autentikoituneelle käyttäjälle. Samalla voidaan kysyä käyttäjältä, että haluaako hän jatkaa kyseiseen resurssiin. (MSAL documentation, 2022.)

MSALista löytyvät funktiot ja komponentit ovat kehitystiimin muokattavissa eikä siitä tarvitse ottaa käyttöön kuin ne osiot mitä kyseisessä sovelluksessa tarvitaan. Rajaamisella saadaan pienennettyä sen kuormitusta ja yksinkertaistaa koodin luettavuutta autentikaation osalta. (MSAL documentation, 2022.)

6.4 Ylläpidettävyys

Yksittäisten sovellusten ylläpidettävyys ja kriittisten toimintojen korjaaminen on natiiveja Android-sovelluksia vasten huomattavasti kevyempää ja nopeampaa. Ei tarvitse päivittää kaikkia tuotantolaitteita, riittää että verkkosivu, johon ohjataan laitteista, on päivittynyt. Tämän avulla pystytään isompien ongelmien aikana esimerkiksi vetämään alas koko verkkosivu ja ohjaamaan muualle, tai ottamaan versionhallinnasta aikaisempi versio ja julkaisemaan se näkyviin tuotannon verkkosivulle.

Natiiveissa sovelluksissa kriittisten päivityksen ongelmaksi muodostuu se, että joka tapauksessa pitää pystyä päivittämään laitteen asennuksia. Kun kyseessä on isompi määrä laitteita, kaikkien laitteiden päivittäminen yhtäaikaisesti on mahdotonta koska sovellusasennukset vaativat aina verkkoyhteyden ja tietyn akkumäärän. Näiden takia asennusten ja päivitysten asentamiseen on olemassa erilaisia logiikoita, kuten hiljaisena aikana ja laitteen ollessa laturissa ajetaan tärkeät päivitykset ja tällä tavalla ei pystytä nopeasti pakottamaan jokaista laitetta päivittymään ilman erillistä ohjeistusta.

6.5 Piensovellusten testattavuus

Kun kaikki erilliset kriittiset toiminnallisuudet tehdään omiksi sovelluksiksi, tämän pitäisi yksinkertaistaa sovellusten toimintoja ja ulkoasuja. Tämän vuoksi sovellusten yksikkötestaus ja automaatiotestaus nousevat tärkeään osaan, koska ajan kuluessa ja muiden sovellusten päivitysten takia on hyvä pystyä testaamaan kaikkia sovelluksia sovellusten ekosysteemissä, joista saadaan luotettavia tuloksia.

Testien kirjoittaminen sovelluskohtaisesti helpottuu sillä, että sovellukset pysyvät erittäin pieninä. Pienoissovellusten integraatio- ja E2E-testaus nousevat tärkeään osaan, kun katselmoidaan koko tuotantoympäristön ekosysteemiä. Pitää pystyä testaamaan erilaisten sovellusten väliset navigoinnit ja toteamaan, että reitit ykset jokaiseen suuntaan ja tiedon mukana kulku onnistuu täydellisesti.

E2E-testejä pystytään tietenkin tekemään tehokkaasti käyttämällä testaajaa. Ihmisen käyttämisen helppoutena on se, että pystytään testaamaan pitkiä monimutkaisia käyttäjäpolkuja. Näiden aikana käydään useissa eri sovelluksissa ja tapahtuu business kriittisiä toimintoja. Toiminnot pitää

pystyä testaamaan useiden reittien keskellä. Tällaisten testien kirjoittaminen automaatiotesteiksi on työlästä ja monimutkaista ja niiden refaktorointi saattaa olla kustannustehotonta.

7 Pohdinta

Tavoitteena oli tutkia, onko ReactJS:n avulla mahdollista luoda tuotantovalmiita piensovelluksia Android kehitysalustan päälle. Työn tarkoituksena oli selvittää piensovelluksien tietoturvallisuus, päivityshelppous, prototyyppien teko, toiminta varmuus sekä se miten paljon sovelluksesta voidaan uusiokäyttää. Ennen työn aloittamista tiedettiin, että JavaScript-pohjaiset sovellukset olisivat helpommat päivittää tuotantoympäristössä, joka mahdollistaisi nopeiden prototyyppien ja PoC:n tekemisen. Työ pohjautuu internetistä löytyvään tietoon sovelluskehityksestä, ReactJS:stä ja varsinkin siitä löytyvistä erilaisista toolchaineista, sen lukuisista kirjastoista sekä työn aikana saatuihin tuloksiin tutkimistamme asioista.

Saamamme tulokset olivat lupaavia, ReactJS:n valitseminen teknologiaksi vaikutti siltä, että ei tarvinnut karsia sovelluskehityksen toiminnallisuuksista tai taustakriteereistä.

Ennen tutkimuksen aloittamista, isoimpina kysymysmerkkeinä olivat navigaatio ja tiedon välitys natiivien sovellusten ja verkkosovellusten välillä, sekä yleiset käyttäjäkokemus kysymykset. Kuten kuinka paljon loppukäyttäjä huomaa suorituskyky eroja tai sitä, että sovelluksen on käytännössä oltava aina nettiyhteydessä kiinni.

Työn aikana selvisi, että on useita eri tapoja hallita tiedon välitystä sovellusten kesken. Natiivista sovelluksesta natiiviin vai natiivista verkkosovellukseen vai verkkosovelluksesta natiiviin, tietenkään osa näistä ratkaisuista ei tule kysymykseenkään useiden eri syiden takia. Löysimme kaksi mahdollista tapaa, joista helpommaksi toteuttaa tämän kaltaisessa ekosysteemissä osoittautui niin sanotut itsetehdyt funktiot jokaiseen kohtaan sovellusten navigointia. Tähän pitää tehdä jonkun verran jatkotutkimusta, joissa selvitetään, mikä on Googlen Android-kehityksen ohjenuorat näihin tapauksiin. Tämä on tärkeä toiminnallisuus tulevan pienoissovellus ekosysteemin sisällä, joten vaikka saimme lupaavia vastauksia tämän työn aikana, on jatkotyötä vielä tehtävä ennen lopullista päätöstä.

Verkkosovelluksen tekeminen PWA-sovellukseksi osoittautui pienitöiseksi, tämä toiminnallisuuden sovellusten toiminnallisuuksien toimisen sisällä myös ilman verkkoyhteyttä. Hyötyä tästä on vaikea havainnoida koska sovellukset ovat pieniä ja todennäköisesti loppukäyttäjä ei pysty jatkamaan töitään ennen kuin saa onnistuneen vastauksen taustajärjestelmistä. Tietenkin turhien virhe viestien väheneminen ja sovellusten kaatumisten väheneminen on yksi iso hyöty, kun mahdollistetaan offline-toiminnallisuus jokaisessa piensovelluksessa.

Isona positiivisena yllätyksenä osoittautui verkkosovellusten tekeminen nopealla syklillä ja muutosten julkaisemisen keveys verrattuna Android-sovelluksiin. Tämä kyseenalaistaa tiettyjen sovellusten tekemisen, että onko mitään hyötyä monissa jatkokehityksen tapauksissa valita natiivia Android-sovellusta. Mikäli tähän päädytään, on siihen oltava hyvät perustelut.

Tällaisia pieniä sovelluksia pystyy varmasti tekemään monilla muillakin nykyaikaisilla verkkosovellus kehityksen teknologioilla ja työkaluilla, mutta väärä paikka käyttää resursseja tällaisessa pohdinnassa on liian tarkka vertailu teknologioiden välillä ennen työn aloittamista. Tämän työn aikana ReactJS ja Create React App osoittautui erittäin hyviksi ratkaisuiksi tällaisen ekosysteemin sisällä tapahtuvaan sovelluskehitykseen. Mikäli ReactJS:stä tai Create React App toolchainistä aiheutuu tulevaisuudessa ongelmia, voidaan nämä vaihtaa toiseen ReactJS:n toolchainiin tai ReactJS johonkin muuhun suosituista JavaScript-kehyksistä. Työssä tutkitut asiat ovat monilta osin yleispäteviä kaikkiin JavaScript-kehyksiin, joten vaikka ReactJS osoittautuisi tulevaisuudessa huonoksi ratkaisuksi voidaan tämän työn tuloksia hyödyntää jatkossa.

Lähteet

About npm. N.d. Npm:n dokumentaatio. Viitattu 12.3.2022. https://docs.npmjs.com/about-npm.

Android app links verification. 2022. Android developer guide for app link verifications. Viitattu 13.3.2022. https://developer.android.com/training/app-links/verify-site-associations

Android guide for app links. 2022. Android developer guide for app links. Viitattu 13.3.2022. https://developer.android.com/training/app-links

Android guide for sending. 2022. Android developer guide for sending data. Viitattu 13.3.2022. https://developer.android.com/training/sharing/send

CRA PWA. 2022. Create React App PWA dokumentaatio. Viitattu 12.3.2022. https://create-react-app.dev/docs/making-a-progressive-web-app/

CRA running tests. 2022. Create React App testing documentation. Viitattu 13.3.2022. https://create-react-app.dev/docs/running-tests/

Create a New React App. 2022. Reactin dokumentaation uuden sovelluksen luominen. Viitattu 13.2.2022. https://reactjs.org/docs/create-a-new-react-app.html

Gatbsy documentation. 2022. Gatsbyn dokumentaatio. Viitattu 19.3.2022. https://www.gatsbyjs.com/docs/

Gatsby Vision. 2022. Gatsbyn dokumentaatio. Viitattu 12.3.2022. https://www.gatsbyjs.com/docs/conceptual/gatsby-core-philosophy/

Getting started in React. 2022. Reactin dokumentaatio. Viitattu 13.2.2022. https://reactjs.org/docs/getting-started.html

Jest mocking. 2022. Jest documentation, mocking. Viitattu 13.3.2022. https://jestjs.io/docs/mock-functions

JSX. 2022. 12.3.2022. https://reactjs.org/docs/introducing-jsx.html

Mdn web socket documentation. 2022. Mozilla Developer Network dokumentaatio web socketeista. Viitattu 13.3.2022. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

Mdn web socket server documentation. 2022. Mozilla Developer Network dokumentaatio web soketing servereistä. Viitattu 13.3.2022. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers

Mozilla PWA introduction. 2022. Mozillan dokumentaatio Viitattu 12.3.2022. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction

MSAL documentation. 2022. Microsoftin MSAL dokumentaatio. Viitattu 19.3.2022. https://docs.microsoft.com/en-us/azure/active-directory/develop/tutorial-v2-react

Npm commands. N.d. Npm documentation for commands. Viitattu 17.4.2022. https://docs.npmjs.com/cli/v8/commands/npm-install

Npm example. N.d. Npm page for MUI-library. Viitattu 17.4.2022. https://www.npmjs.com/package/@mui/material

Razzle. N.d. Razzle dokumentaatio. Viitattu 12.3.2022. https://razzlejs.org/

React. 2022. ReactJS:n kotisivut. Viitattu 13.2.2022. https://reactjs.org/

ReactJS testing documentation. 2022. ReactJS testing environments documentation. Viitattu 13.3.2022. https://reactjs.org/docs/testing-environments.html

ReactJS testing. 2022. ReactJS testing documentation. Viitattu 13.3.2022. https://reactjs.org/docs/testing.html

Style Dictionary documentation. 2022. Style Dictionaryn dokumentaatio. Viitattu 13.3.2022. https://amzn.github.io/style-dictionary/#/

Trends buildwith. 2022. JavaScript library trends. Viitattu 17.4.2022. 17.4.2022 https://trends.builtwith.com/javascript/javascript-library

Webpack. N.d. Webpack documentation. Viitattu 17.4.2022. https://webpack.js.org/

What is Next.js?. 2022. Next.js:n dokumentaatio. Viitattu 26.2.2022. https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs

Why Gatsby?. 2022. Why Gatbsy. Viitattu 19.3.2022. https://www.gatsbyjs.com/why-gatsby/