Kevin Neuman

Full Stack -mobiilisovelluksen luonti React Native -tekniikalla

CASE: Kupo

Opinnäytetyö
Kevät 2017
SeAMK Tekniikka
Tietotekniikan tutkinto-ohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Tietoverkkotekniikka

Tekijä: Kevin Neuman

Työn nimi: Full Stack -mobiilisovelluksen luonti React Native -tekniikalla

Ohjaaja: Petteri Mäkelä

Vuosi: 2017 Sivumäärä: 42 Liitteiden lukumäärä: 0

Opinnäytetyö toteutettiin Valakia Interactive Osakeyhtiölle. Valakia Interactive on Seinäjoella sijaitseva startup-yritys, jonka palveluihin kuuluvat mainonta, lisätty todellisuus, 3D-sovellukset, graafinen suunnittelu ja vuorovaikutteinen media.

Tässä opinnäytetyössä tutustuttiin moderneihin web- ja mobiilikehitystekniikoihin. Työn tavoitteena oli toteuttaa mobiilisovellus, joka toimii vuorovaikutuksessa kosketusnäytöllisen infotaulun kanssa.

Opinnäytetyö koostuu teoriaosuudesta sekä käytännön osuudesta. Teoriaosuudessa käsiteltiin front end -tekniikoita, joilla luotiin käyttöliittymät mobiilisovellukseen ja verkkosivulle. Lisäksi syvennyttiin back end -tekniikoihin, joilla palvelinpuoli toteutettiin. Työn käytännön osuudessa esitellään Kupo-mobiilisovelluksen ulkoasua ja toiminnallisuuksia. Siinä kuvataan myös palvelinpuolen toimintaa yhdessä mobiilisovelluksen ja verkkosivun kanssa.

Lopputuloksena syntyi Kupo-mobiilisovellus. Mobiilisovelluksella voi aloittaa pelin kosketusnäytölliseltä infotaululta. Peli alkaa, kun mobiilisovellus on lukenut QR-koodin infotaululta. Mobiilisovellus käyttää Node.js-palvelimen REST-ohjelmointirajapintaa tietojen hakemiseen ja muuttamiseen tietokannasta. Lisäksi luotiin React-verkkosivu, joka toimii samalla palvelimella. Verkkosivua käytetään käyttäjätilin luomisen viimeistelyyn ja salasanan vaihtamiseen.

Avainsanat: React Native, React, Node.js, MongoDB, mobiilisovellus

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Networking Technology

Author: Kevin Neuman

Title of thesis: Full Stack mobile application made with React Native

Supervisor: Petteri Mäkelä

Year: 2017 Number of pages: 42 Number of appendices: 0

The commissioner of this thesis was Valakia Interactive Limited. Valakia Interactive is a startup company located in Seinäjoki. The company's services include advertising, augmented reality, 3D-applications, graphic design and interactive media.

This thesis explored modern web and mobile development technologies. The aim was to implement a mobile application that interacts with an information display.

The thesis consists of a theoretical and a practical part. The theoretical part deals with the front end technologies, which were used to create user interfaces for the mobile application and the website. The theoretical part also includes a back end section that consists of server-side technologies. The practical part presents the layout and functionality of the Kupo -mobile application. It also describes the server-side operations together with the mobile application and the website.

The result of this thesis was the Kupo -mobile application. The mobile application uses a QR code reader to start a game on an information display. The mobile application uses Node.js-server's REST Application Programming Interface to retrieve information from the database and to modify it. The React-website is hosted on the same server. The website is used for finishing user account creation and to change a user's password.

Keywords: React Native, React, Node.js, MongoDB, mobile application

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

Asynkroninen	Asynkronisuudella ohjelmoinnissa tarkoitetaan sitä, että useita toimintoja koodissa suoritetaan samaan aikaan.
Back end	Palvelinpuoli, joka tarjoaa erilaisia palveluita sovelluksille. Tähän kuuluu myös tietokantojen käsittely.
Debuggeri	Jäljittää ohjelmointivirheitä koodista.
Front end	Käyttäjälle näkyvä osuus sovelluksesta eli käyttöliittymä.
Full Stack	Ohjelmistopino, joka käsittää käyttöliittymän ja palvelinpuolen.
HTML5 web -sovellus	Mobiililaitteille optimoitu web-sovellus, joka toimii kaikilla laitteilla.
HTTP-protokolla	Hypertext Transfer Protocol eli hypertekstin siirtoprotokolla. Selaimet ja WWW-palvelimet käyttävät tätä tiedonsiirtoon.
Hybridi sovellus	Natiivi mobiilisovellus, jonka käyttöliittymä on tehty HTML5-, CSS3- ja JavaScript-tekniikoilla. Käyttöliittymä avautuu sovelluksessa koko näytön kokoisena.
JSON	JavaScript Object Notation on JavaScript-ohjelmointikielestä riippumaton avoimen standardin tiedostomuoto. Se koostuu avain-arvo-pareista.
Mobiiliverkkosovellus	Mobiililaitteelle optimoitu verkkosivu, jossa tyylit muuttuvat näytön koon mukaan.
Natiivi mobiilisovellus	Tietylle käyttöjärjestelmälle (Android, iOS) erikseen ohjelmoitu sovellus.

NoSQL NoSQL käsite kuvastaa perinteisestä relaatiomallista poikkeavaa tietokantaa. NoSQL-tietokanta ei seuraa kiinteästi määrättyä taulukkomallia, siksi se skaalautuu hyvin.

Spinneri Ikoni, joka kuvastaa lataamista sovelluksessa.

TCP-yhteys Transmission Control Protocol on tietoliikenneprotokolla. Sen avulla luodaan yhteyksiä tietokoneiden välille, jotka ovat yhteydessä internetiin.

Yhden sivun web-sovellus

Kaikki tarvittava sivuston koodi ladataan kerralla, tämä luo samankaltaisen käyttäjäkokemuksen kuin työpöydän sovelluksissa.

1 JOHDANTO

1.1 Työn tausta

Mobiilisovelluksien suosio on ollut nousussa jo monta vuotta. Ala on valtava ja sen kasvulle ei näy loppua. Mobiilikehittäjien määrä on lisääntynyt, siksi myös mobiilisovelluksia on nykyään ennennäkemättömän paljon. Lähes jokaisella on jokin mobiililaite käytössään, siksi mobiilisovellus on hyvä keino tavoittaa suuri käyttäjäkunta.

Vuorovaikutteinen media on tehokas työkalu markkinoinnin tueksi. Valakia Interactive Osakeyhtiö toteuttaa vuorovaikutteista sisältöä erilaisille digitaalisille laitteille, kuten kosketusnäytöllisiin infotauluihin ja mobiilisovelluksiin. Niiden avulla markkinointi on osallistavaa ja informatiivista myös markkinoijalle.

1.2 Työn tavoite

Opinnäytetyön aiheena on kehittää Full Stack -ympäristö, joka koostuu mobiilisovelluksesta, verkkosivusta, palvelinpuolesta, REST-ohjelmointirajapinnasta ja tietokannasta. Opinnäytetyön tarkoituksena on käydä läpi moderneja tekniikoita, joita käytetään edellä mainitun Full Stack -ympäristön toteuttamiseen.

Opinnäytetyön tavoitteena on luoda Valakia Interactive Osakeyhtiölle mobiilisovellus, joka toimii vuorovaikutuksessa kosketusnäytöllisen infotaulun kanssa. Mobiilisovellukseen on tarkoitus kehittää QR-koodinlukija ja kirjautumisjärjestelmä. Mobiilisovelluksen tavoitteena on, että QR-koodinlukijalla aloitetaan peli kosketusnäytölliseltä infotaululta. Kirjautumisjärjestelmän avulla pelin on tarkoitus alkaa käyttäjän tiedoilla ja ohjata mahdolliset palkinnot käyttäjälle.

1.3 Työn rakenne

Luvussa 2 perehdytään front end -tekniikoihin, joita käytettiin käyttöliittymien luomiseen. Tekniikoiden käyttöä on havainnollistettu kuvioilla.

Luvussa 3 tutustutaan back end -tekniikoihin, joita käytettiin palvelinpuolen toteuttamiseen. Palvelinpuolen toimintaa on tuotu esille kuvioiden avulla.

Luvussa 4 esitellään opinnäytetyön käytännön osuus, jossa perehdytään Kupo-mobiilisovelluksen ulkoasuun ja toiminnallisuuteen.

Luku 5 on yhteenveto, jossa pohditaan opinnäytetyön tavoitteita, tuloksia ja ongelmia. Lisäksi siinä pohditaan käytettyjen tekniikoiden hyötyjä.

1.4 Valakia Interactive Osakeyhtiö

Valakia Interactive Osakeyhtiö on Seinäjoella sijaitseva startup-yritys. Yritys on perustettu vuonna 2015 ja se tarjoaa uusia tapoja mainontaan. Yrityksen tarjoamat työkalut ovat hauskoja ja tehokkaita työkaluja markkinoinnin tueksi. (Valakia Interactive [Viitattu 12.4.2017].)

Yrityksen palveluihin kuuluu räätälöidyt 3D-sovellukset, yksilölliset ratkaisut sähköisiin medioihin, vuorovaikutteinen mediamarkkinointi ja graafinen suunnittelu mainoksiin, painotuotteisiin, kotisivuihin, kuvituksiin, pakkausmateriaaleihin ja yrityksen kokonaisilmeeseen. (Valakia Interactive [Viitattu 12.4.2017].)

Lisäksi yrityksen palveluihin kuuluu lisätty todellisuus, jonka avulla on mahdollista tuoda perinteisen mainoksen tueksi sähköisen median luomat mahdollisuudet. Tekniikan avulla voidaan herättää kuvat eloon ja lisätä mainoksen informaatioarvoa. (Valakia Interactive [Viitattu 12.4.2017].)

2 FRONT END -TEKNIIKAT

2.1 React

React on JavaScript-kirjasto, joka on tarkoitettu käyttöliittymien tekoon. Sen avulla voidaan muodostaa interaktiivisia käyttöliittymiä. (React [Viitattu 7.2.2017].) React on käytössä esimerkiksi Netflix-, PayPal- ja Imgur-sivustoilla (Libscore [Viitattu 8.3.2017]). React julkaistiin vuonna 2013 (Gackenheimer 2015, 1).

Facebookin insinöörit loivat React-tekniikan ratkaisemaan haasteita monimutkaisten käyttöliittymien kehittämisessä, joissa aineistot muuttuvat ajan myötä. Käyttöliittymän on oltava ylläpidettävissä ja skaalautuva, jotta se toimisi Facebookin mittakaavassa. React on luotu Facebookin mainosorganisaatiossa. (Gackenheimer 2015, 1.)

React edisti web-kehitystä ja muutti tapaa, jolla web-sovelluksia tehdään. Se on muutos yleisesti hyväksytystä työnkulusta ja parhaista käytännöistä. (Gackenheimer 2015, 2.) Tekniikan mukana tulee runsaasti erilaisia ominaisuuksia, jotka tekevät yhden sivun web-sovelluksen tai käyttöliittymän laatimisesta lähestyttävän eri taitotasoisille kehittäjille (Gackenheimer 2015, 1).

Ainoa tapa kirjoittaa monimutkainen sovellus on pitää globaali monimutkaisuus poissa. Se rakennetaan yksinkertaisista osista, jotka ovat yhdistettyinä selvästi määriteltyihin rajapintoihin. Näin useimmat ongelmat ovat paikallisia ja on mahdollista päivittää osa rikkomatta koko sovellusta. (Raymond 2003, 14.) React käyttää tätä lähestymistapaa ongelmien ratkaisuun (Gackenheimer 2015, 2).

React rakennettiin käsittelemään datan näyttämistä käyttöliittymässä. Se on luotu palvelemaan laajamittaisia käyttöliittymiä, joissa suuria määriä dataa muuttuu ajan myötä. Facebook ja Instagram ovat hyviä esimerkkejä tällaisista käyttöliittymistä. (Gackenheimer 2015, 2.)

React päivittää ja esittää muutokset komponenteissa, kun tieto sovelluksessa vaihtuu. Muuttuvat näkymät tekevät koodista ennustettavan ja helposti testattavan.

React koostuu komponenteista, joista muodostuu yhdessä monimutkainen käyttöliittymä. (React [viitattu 7.2.2017].)

2.1.1 React-esimerkkisovellus

Uuden projektin luominen on helppoa, kun Node.js on asennettu tietokoneelle. Asennusohjelma on ladattavissa Node.js-verkkosivulta. Paras tapa aloittaa uuden React-sovelluksen rakentaminen on käyttää Create React App -menetelmää (React Installation [Viitattu 8.3.2017]).

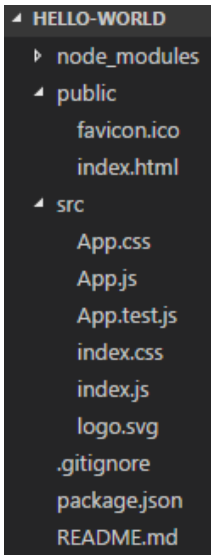
Kuvion 1 npm install -g create-react-app -komento asentaa create-react-app-menetelmän tietokoneelle. Create-react-app hello-world -komento luo hello-world-projektin. Cd hello-world -komennon avulla siirrytään hello-world-kansioon. Npm start -komento käynnistää sovelluksen. Komennot kirjoitetaan komentokehotteeseen. (React Installation [Viitattu 8.3.2017].)

```
Code

npm install -g create-react-app
create-react-app hello-world
cd hello-world
npm start
```

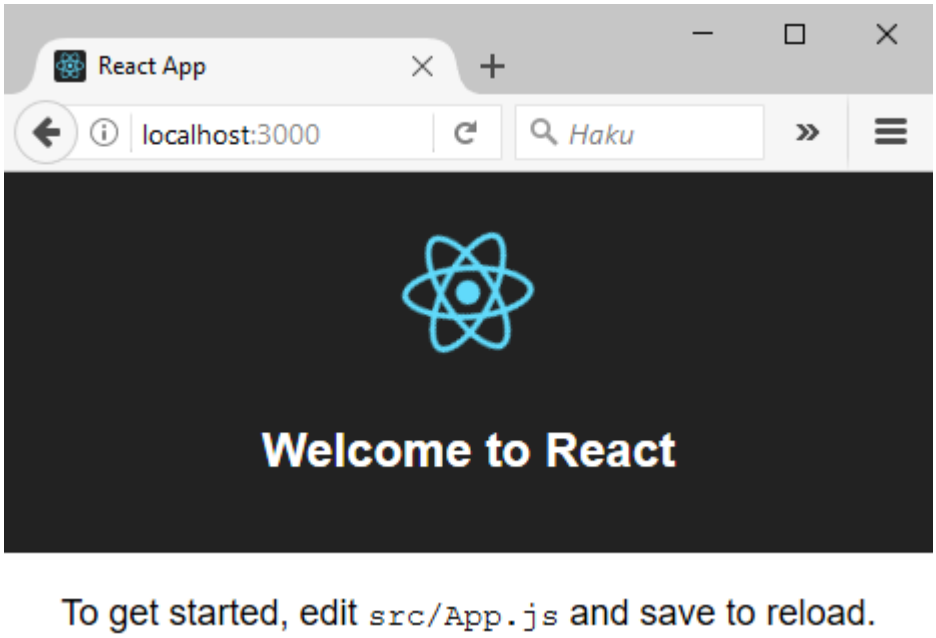
Kuvio 1. React-sovelluksen luomiseen tarvittavat komennot.
(React Installation [Viitattu 8.3.2017]).

Kuviossa 2 esitellään Create React App -menetelmän luomat tiedostot. Node_modules-kansio sisältää kaikki moduulit, jotka on listattu package.json-tiedostossa. Public-kansiossa on selaimessa näkyvät tiedostot. Src-kansiossa on sivun lähdekoodi.



Kuvio 2. React-sovelluksen kansiorakenne.

Kuviossa 3 nähdään miltä React-sovellus näyttää selaimessa. Se aukeaa osoitteessa http://localhost:3000. Sivua pääsee muokkaamaan src-kansiossa sijaitsevasta App.js-tiedostosta (kuvio 2).



Kuvio 3. React-sovellus selaimessa.

Tämä esimerkki ei käsittele palvelinpuolen logiikkaa eikä tietokantoja. Se luo pelkästään käyttöliittymän, joten sitä voi käyttää minkä tahansa palvelimen kanssa. (React Installation [Viitattu 8.3.2017].)

2.2 React Native

React Native perustuu React JavaScript -kirjastoon eli niiden koodi on samankaltaista. Samankaltaisuuden ansiosta React-tekniikalla voi luoda web-, mobiili- ja työpöytäsovelluksia. (Gackenheimer 2015, 2.)

React Native on tekniikka, jolla voidaan luoda natiivi Android- tai iOS-mobiilisovellus. Sillä ei rakenneta mobiiliverkkosovellusta, HTML5-sovellusta tai hybridi-sovellusta. Sen avulla syntyy oikea mobiilisovellus, jota on mahdotonta erottaa Objective-C- tai Java-ohjelmointikielillä tehdyistä sovelluksista. React Native käyttää perusperiaatteeltaan samoja käyttöliittymän rakennusosia kuin tavalliset Android- ja iOS-sovellukset. Ainoa ero näihin sovelluksiin on, että nämä osat yhdistetään käyttämällä JavaScript-ohjelmointikieltä ja React-kirjastoa. (React Native [Viitattu 9.2.2017].)

Kuviossa 4 nähdään, kuinka eri ohjelmointikielellä kirjoitettu komponentti voidaan yhdistää React Native -sovellukseen. Esimerkiksi Objective-C-, Java- tai Swift-ohjelmointikielillä kirjoitettu komponentti voidaan lisätä suoraan koodiin. (React Native [Viitattu 9.2.2017].)

index.android.js

```
1  import React, { Component } from 'react';
2  import {
3      View
4  } from 'react-native';
5  import { NatiiviKomponentti } from './natiivi-koodi';
6
7  class my_app extends Component {
8      render() {
9          return (
10             <View>
11                 <NatiiviKomponentti />
12             </View>
13         );
14     }
15 }
16
17 AppRegistry.registerComponent('my_app', () => my_app);
```

Kuvio 4. Eri ohjelmointikielellä kirjoitetun komponentin lisääminen React Native -sovellukseen.

2.2.1 React Native -esimerkkisovellus Android-käyttöjärjestelmälle

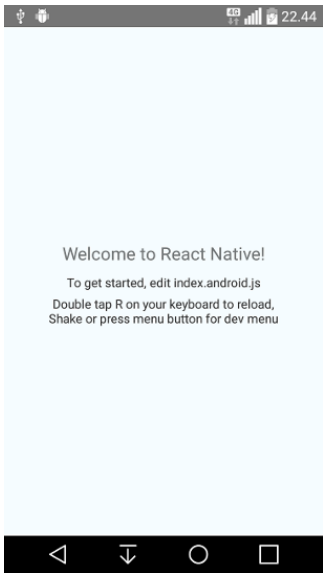
React Native -projektin luominen vaatii, että tietokoneella on asennettuna Node.js, React Native -komentorivikäyttöliittymä ja Android Studio. Android Studio tarjoaa ohjelmistokehitystyökalut ja emulaattorin, joita tarvitaan sovelluksien testaamiseen. Testaaminen onnistuu myös fyysisellä mobiililaitteella USB-kaapelin kautta. Kun testauslaite on käynnistetty, voi asennuksen suorittaa komentokehotteen kautta. (React Native Installation [Viitattu 23.3.2017].)

Kuvion 5 react-native init AwesomeProject -komento luo AwesomeProject-projektin. Cd AwesomeProject -komennon avulla siirrytään AwesomeProject-kansioon. React-native run-android -komento käynnistää sovelluksen testauslaitteelle. (React Native Installation [Viitattu 23.3.2017].)

```
react-native init AwesomeProject
cd AwesomeProject
react-native run-android
```

Kuvio 5. React Native -projektin luomiseen tarvittavat komennot.
(React Native Installation [Viitattu 23.3.2017]).

Kuviossa 6 esitellään React Native -sovellus testilaitteessa, tässä tapauksessa älypuhelimessa, joka on yhdistetty tietokoneeseen USB-kaapelilla. Sovellus toivottaa kehittäjän tervetulleeksi ja neuvoo eteenpäin.



Kuvio 6. React Native -sovellus testilaitteessa.

2.3 QR-koodinlukija

GitHub-verkkosivulta löytyy react-native-camera-moduuli, jonka avulla on mahdollista ottaa kuvia sekä lukea erilaisia viivakoodeja.

Kameran tunnistamia viivakoodityyppejä ovat

- aztec
- code128
- code39
- code39mod43
- code93
- ean13
- ean8
- pdf417
- qr
- upce. (React Native Camera [Viitattu 23.3.2017].)

QR-koodeja voi nähdä mainoksissa, mainostauluissa, yritysten ikkunoissa ja tuotteissa. Ne ovat todella suosittuja markkinoinnin yhteydessä. Esimerkiksi älypuhelimen kameralla luettu QR-koodi voi avata siinä olevan verkkosivun. Lyhenne QR tulee sanoista Quick Response eli nopeasti vastaava. QR-koodi on neliön muotoinen viivakoodi, joka on lähtöisin Japanista (kuvio 7). (QR Codes Explained [Viitattu 6.5.2013].)

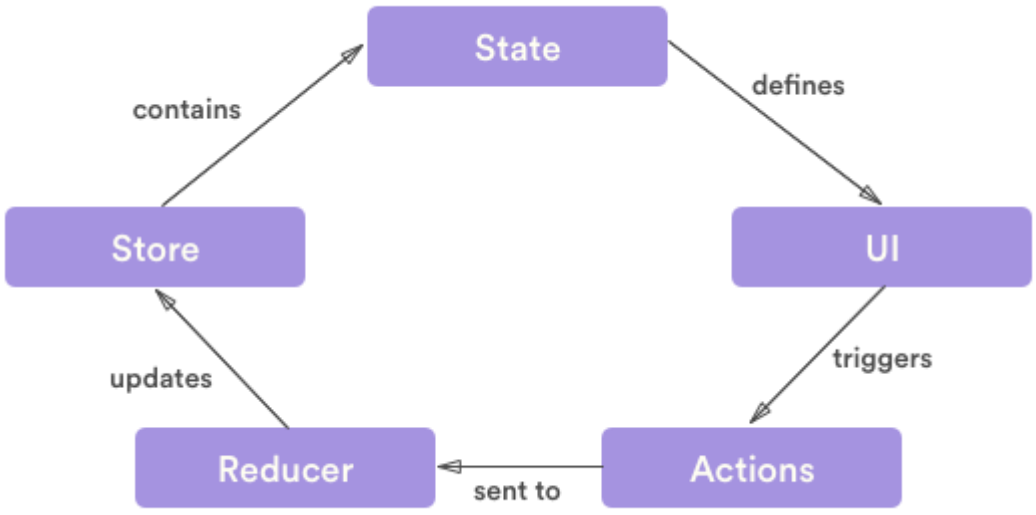
Kuvio 7. QR-koodi.

2.4 Redux

JavaScript-sovellusten muuttuessa koko ajan monimutkaisemmiksi koodin täytyy käsitellä tiedon eri tiloja enemmän kuin koskaan aikaisemmin. Käyttöliittymä luo haasteita tilojen seuraamisessa, kun joudutaan hallitsemaan aktiivisia reittejä, valittuja välilehtiä, spinnereitä ja sivunumerointia. (Redux Motivation [Viitattu 8.3.2017].)

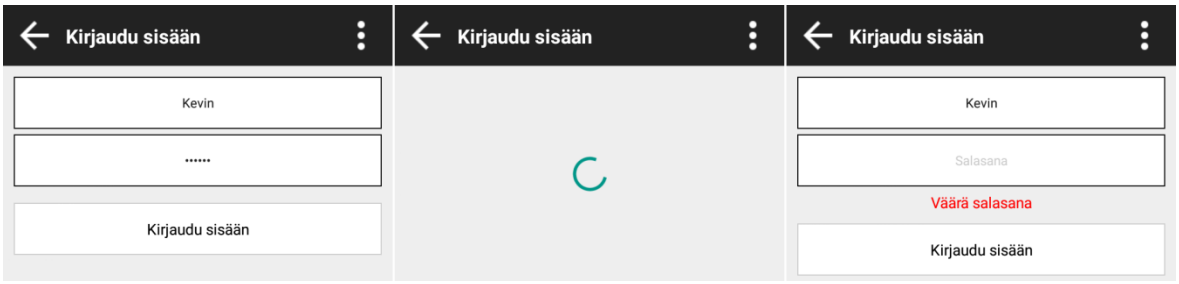
Redux auttaa kirjoittamaan johdonmukaisesti käyttäytyviä sovelluksia, joita on helppo testata, ja jotka toimivat eri ympäristöissä (front end ja back end). Reduxin tavoitteena on luoda tilanhallintakirjasto minimaalisella ohjelmointirajapinnalla, mutta täysin ennustettavalla käyttäytymisellä. (Redux Read Me [Viitattu 8.2.2017].)

Ydinajatuksena on, että sovelluksen kaikki tilat on tallennettu yhteen objektipuuhun. Tätä objektipuuta kutsutaan nimellä store. Ainoa tapa muuttaa tiloja store-objektipuussa on actionin avulla. (Redux Read Me [Viitattu 8.2.2017].) Action lähettää dataa sovelluksesta store-objektipuuhun (Redux Actions [Viitattu 24.3.2017]). Reducerit määrittelevät, miten actionit muuttavat tiloja store-objektipuussa (Redux Read Me [Viitattu 8.2.2017]). Kuviossa 8 esitellään Redux-työvaiheet käyttöliittymässä.



Kuvio 8. Redux-työvaiheet.
(Journey into React Part 6 [Viitattu 26.8.2016]).

Kuvion 9 tapahtumia voidaan seurata kuvion 8 avulla. Kirjaudu sisään -painike laukaisee actionin, joka lähetetään reduceriin. Store päivittyy ja uudet tilat (state) määrittelevät käyttöliittymän (UI).

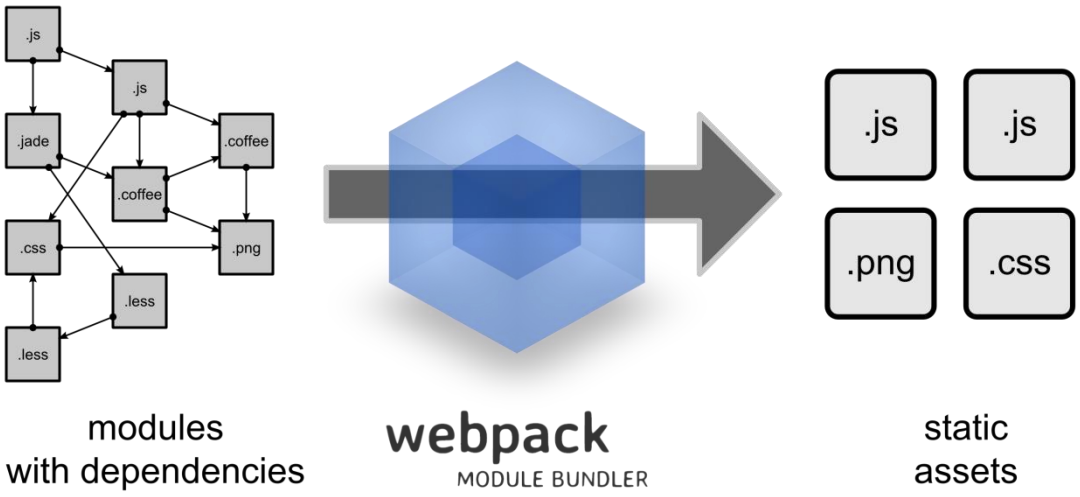


Kuvio 9. Näkymät muuttuvat tilojen mukaan.

2.5 Webpack

Webpackin päätarkoitus on yhdistää JavaScript-tiedostot selainkäyttöä varten. Päätarkoituksen lisäksi sillä on mahdollista pakata monia muitakin tiedostomuotoja. (GitHub webpack [Viitattu 7.2.2017].)

Webpack sopii hyvin suuriin yhden sivun web-sovelluksiin. Se pakkaa kaikki moduulit ja luo niistä vastaavia staattisia tiedostoja (kuvio 10). Sivuston latausaika nopeutuu, kun tiedostot on pakattu yhteen. (What is webpack [Viitattu 9.2.2017].)



Kuvio 10. Moduulien pakkaus webpackilla.
(What is webpack [Viitattu 9.2.2017]).

Webpack tukee myös useita laajennuksia. UglifyJsPlugin on laajennus, joka supistaa pakatut JavaScript-tiedostot mahdollisimman pieneksi. Lisäksi se sekoittaa koodin niin, että sitä on mahdotonta lukea. (List of webpack plugins [Viitattu 9.2.2017].)

3 BACK END -TEKNIIKAT

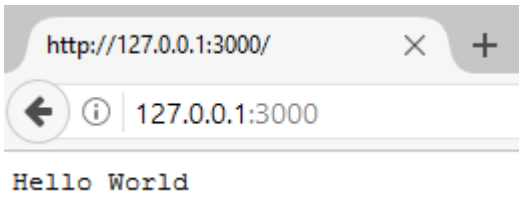
3.1 Node.js

Node.js on järjestelmäriippumaton JavaScript-ajoympäristö. Sen avulla voidaan rakentaa skaalautuvia verkkosovelluksia. Kuvion 11 mukainen esimerkkisovellus antaa vastauksen Hello World, kun joku ottaa siihen yhteyden esimerkiksi selaimella (kuvio 12). Palvelin on lepotilassa, kun sillä ei ole liikennettä. (About Node.js [Viitattu 9.2.2017].)

server.js

```
1   const http = require('http');
2
3   const hostname = '127.0.0.1';
4   const port = 3000;
5
6   const server = http.createServer((req, res) => {
7       res.statusCode = 200;
8       res.setHeader('Content-Type', 'text/plain');
9       res.end('Hello World');
10  });
11
12  server.listen(port, hostname, () => {
13      console.log(`Palvelin käynnissä osoitteessa http://${hostname}:${port}/`);
14  });
```

Kuvio 11. Node.js-esimerkkisovellus.

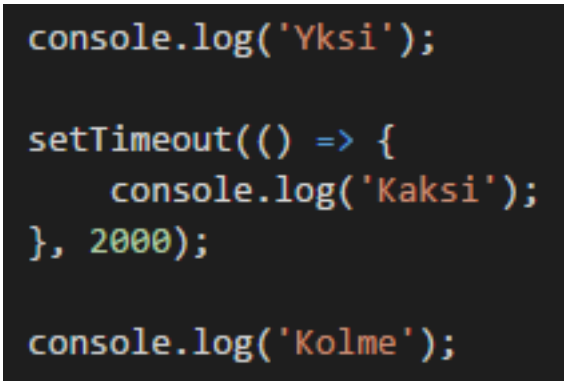


Kuvio 12. Node.js-esimerkkisovelluksen vastaus selaimessa.

Perinteiset web-palvelin-tekniikat edellyttävät uutta prosessorin säiettä jokaiselle yhteydelle. Se luo ongelman, jonka takia järjestelmän resurssit eivät lopulta riittäisi. Node.js käyttää yhtä säiettä, mutta se ei estä prosessorin tuloa tai lähtöä. Siksi Node.js pystyy käsittelemään kymmeniä tuhansia yhteyksiä samanaikaisesti. Juuri tämän ansiosta Node.js on suosittu web-sovelluksissa, joissa on paljon liikennettä. (Krol 2014, 8.)

Node.js koostuu pienestä joukosta moduuleja, jotka tekevät tietyn asian todella hyvin. Moduulien työkalujen avulla voi työskennellä tiedostojärjestelmän, TCP-yhteyksien, HTTP-protokollan, turvallisuuden ja suoratoistojen kanssa. (Krol 2014, 9.)

Yksi Node.js-ajoympäristön tehokkaimmista ominaisuuksista on, että se on asynkroninen. Asynkronisuuden ansiosta muu koodi ei esty, jos joudutaan odottamaan jotakin tapahtumaa (kuvio 13). (Krol 2014, 9.)



Kuvio 13. Asynkroninen esimerkki, jossa ulostulo olisi Yksi, Kolme ja kahden sekunnin jälkeen Kaksi.

3.2 Node Package Manager

Node Package Manager eli npm on Node.js-ajoympäristön ohjelmisto arkisto, joka on maailman suurin avoimen lähdekoodin kokoelma (Node.js [Viitattu 9.2.2017]). Sen avulla löytää tuhansia moduuleja, joita voi asentaa ja käyttää sovelluksessa. Käytettävissä olevat moduulit ovat tarkasteltavissa npmjs.org-verkkosivulla. (Krol 2014, 10.)

Moduulien nimet tallentuvat package-tiedostoon, joka on JSON-tiedostomodossa. Sen avulla npm osaa asentaa sovellukseen kuuluvat moduulit node_modules-kansioon. Package-tiedostossa on myös moduulien versionumerot (kuvio 14).

package.json

```
{
    "name": "junat_liikenteessa",
    "version": "0.0.1",
    "private": true,
    "scripts": {
        "start": "node node_modules/react-native/local-cli/cli.js start",
        "test": "jest"
    },
    "dependencies": {
        "moment": "^2.17.1",
        "react": "15.4.2",
        "react-native": "0.42.0",
        "react-native-maps": "^0.12.4",
        "react-native-tab-view": "0.0.56",
        "react-redux": "^5.0.2",
        "redux": "^3.6.0",
        "redux-thunk": "^2.2.0"
    },
    "devDependencies": {
        "babel-jest": "19.0.0",
        "babel-preset-react-native": "1.9.1",
        "eslint-config-react-native": "^1.6.0",
        "jest": "19.0.2",
        "react-test-renderer": "15.4.2"
    },
    "jest": {
        "preset": "react-native"
    }
}
```

Kuvio 14. React Native -projektin package-tiedosto.

3.3 Express.js

Express.js on nopea, minimalistinen web-kehys Node.js-ajoympäristölle. Se tarjoaa monia ominaisuuksia web- ja mobiilisovelluksille. HTTP-menetelmien ja middlewaren avulla ohjelmointirajapinnan tekeminen on nopeaa. Hyvä suorituskyky säilyy, koska Express.js tarjoaa vain olennaiset ominaisuudet peittelemättä Node.js-ajoympäristön ominaisuuksia. (Express [Viitattu 9.2.2017].)

Kuvion 15 mukainen esimerkkisovellus käynnistää palvelimen ja kuuntelee yhteyksiä porttiin 3000. Sovellus vastaa Hello World -tekstin pyynnöille, jotka ovat tulleet oletusreittiin (/). Jokaiseen muuhun reittiin se lähettää 404 Not Found -vastauksen, koska niitä ei ole olemassa. (Express ”Hello World” example [Viitattu 9.2.2017].)

server.js

```
1   var express = require('express');
2   var app = express();
3
4   app.get('/', function (req, res) {
5       res.send('Hello World');
6   });
7
8   app.listen(3000, function () {
9       console.log('Esimerkkisovellus käynnissä portissa 3000');
10  });
```

Kuvio 15. Node.js-esimerkkisovellus, joka käyttää Express.js-web-kehystä.

Reititys määrittää, kuinka sovellus vastaa pyyntöön tietyssä päätepisteessä. Päätepiste on reitti yhdistettynä johonkin HTTP-pyyntömenetelmään (taulukko 1). (Express basic routing [Viitattu 9.2.2017].)

Taulukko 1. HTTP-pyyntömenetelmät.
(Understanding REST [Viitattu 9.2.2017]).

GET	**Tiedon hakemiseen**
POST	**Uuden kokonaisuuden lisäämiseen**
PUT	**Kokonaisuuden päivittämiseen**
DELETE	**Pyytää, että resurssi on poistettava**

Kuviossa 16 esitellään yksinkertaisia reittejä erilaisilla HTTP-pyyntömenetelmillä. Samaan reittiin tullut pyyntö tekee eri asian riippuen HTTP-pyyntömenetelmästä. Jos esimerkiksi selaimella vierailee reitissä (/), saa vastauksen Hello World, koska selain käyttää GET-pyyntömenetelmää.

```
app.get('/', function (req, res) {
    res.send('Hello World');
});

app.post('/', function (req, res) {
    res.send('Got a POST request');
});

app.put('/user', function (req, res) {
    res.send('Got a PUT request at /user');
});

app.delete('/user', function (req, res) {
    res.send('Got a DELETE request at /user');
});
```

Kuvio 16. Esimerkki yksinkertaisten reittien määrittelemisestä. (Express basic routing [Viitattu 9.2.2017]).

3.4 MongoDB

MongoDB on järjestelmäriippumaton NoSQL-tietokanta. Se käyttää joustavaa datamallia, joka muistuttaa JSON-tiedostomuotoa. Dokumentit voivat sisältää yhden tai useampia kenttiä, ja ne voivat vaihdella dokumentti kohtaisesti. Tämä joustavuus auttaa kehittäjätiimejä muuttamaan datamalleja nopeasti, kun sovelluksen vaatimukset vaihtuvat. Kehittäjät pääsevät käsiksi dokumentteihin kaikilla suosituilla ohjelmointikielillä. (MongoDB [Viitattu 1.3.2017].)

MongoDB-tiimin tavoitteena ei ollut tehdä tietokantaa, joka sopii kaikille. Tärkeää oli, että se olisi todella nopea, massiivisesti skaalautuva ja helppo käyttää. MongoDB sopii täydellisesti analytiikan sekä monimutkaisten tietorakenteiden ongelmien ratkaisemiseen. (Membrey, Hows & Plugge 2014, 2.)

MongoDB-tietokannasta tulisi olla useampi varmuuskopio. Jos yksittäinen tietokanta vioittuu, voidaan se helposti palauttaa toiselta palvelimelta. Koska MongoDB pyrkii olemaan nopein mahdollinen tietokanta, sen palauttaminen kaatumisen jälkeen on vaikeampaa. Kehittäjät uskovat, että useimmat vakavat kaatumiset olisivat joka tapauksessa johtaneet koko palvelimen kaatumiseen. MongoDB-

tietokannan voi asentaa Linux-, Mac OS-, Windows- ja Solaris-käyttöjärjestelmille. (Membrey, Hows & Plugge 2014, 2.)

MongoDB sisältää ominaisuudet, joita tarvitaan organisaatioiden nykyään kehittämiin sovelluksiin. WiredTiger-säilömoduuli tuottaa hyvän yleisen suorituskyvyn ja ennustettavan pienen viiveen tietojenkäsittelyssä. (MongoDB [Viitattu 1.3.2017].)

3.5 Mongoose

Mongoose on MongoDB-tietokannan kokoelmien muotoilemiseen tarkoitettu työkalu. Mongoose perustuu kaavioihin, jotka määrittävät kokoelmien dokumenttien muodon. Jotta kaavioita voidaan käyttää, täytyy ne ensin muuttaa malleiksi. (Mongoose [Viitattu 8.3.2017].)

Käyttäjänimi ja salasana (kuviossa 17) ovat pakollisia kenttiä. Lisäksi käyttäjänimen täytyy olla uniikki, eli toista samannimistä ei saa löytyä tietokannasta.

user.js

```
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  var userSchema = new Schema({
5      username: { type: String, required: true, unique: true },
6      password: { type: String, required: true }
7  });
8
9  var User = mongoose.model('User', userSchema);
10
11 module.exports = User;
```

Kuvio 17. Mongoose user -kaavion muunnos malliksi.

Muunnoksen jälkeen mallia voidaan käyttää koodissa uuden käyttäjän tallentamiseen (kuvio 18).

```
var User = require('../models/user');

var user = new User();
user.username = 'admin';
user.password = 'password';

user.save(function (err) {
    if (err) throw err;

    console.log('User saved successfully!');
});
```

Kuvio 18. Uuden käyttäjän tallentaminen MongoDB-tietokantaan.

3.6 JSON Web Tokens

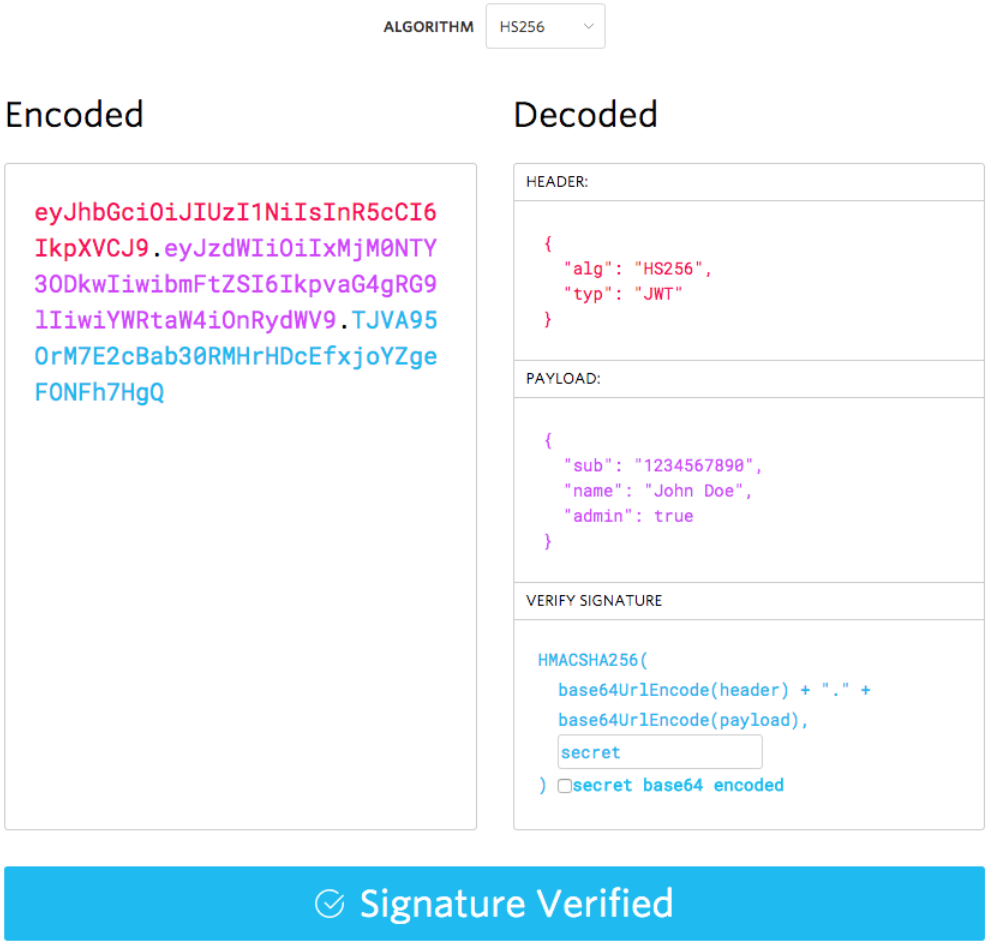
JSON Web Tokens (JWT) on avoin standardi, joka mahdollistaa turvallisen tavan lähettää tietoa kahden osapuolen välillä JSON-tiedostomuodossa. Tiedot voidaan todentaa ja ne ovat luotettavia, koska ne ovat digitaalisesti allekirjoitettu salausavaimella. (JSON Web Tokens [Viitattu 8.3.2017].)

JWT voidaan lähettää POST-pyyntömenetelmän parametrina, URL-osoitteessa (kuvio 19) tai HTTP-ylätunnisteen mukana. JWT:n pieni koko tekee lähetyksestä nopeaa. JWT sisältää kaiken tarvittavan tiedon käyttäjästä, minkä ansiosta tietokantaa ei tarvitse tiedustella kuin kerran. (JSON Web Tokens [Viitattu 8.3.2017].)

Kuvio 19. Esimerkki JWT:n käytöstä URL-osoitteessa.

Yleisimmin JWT on käytössä kirjautumisen yhteydessä. Kun käyttäjä on kirjautunut sisään, jokainen pyyntö, minkä käyttäjä tekee, sisältää JWT:n. Sen avulla käyttäjä pääsee käsiksi hänelle tarkoitettuihin palveluihin ja tietoihin. Myös tietojen lähetys onnistuu turvallisesti. Digitaalinen allekirjoitus varmistaa, että sisältöä ei ole käsitelty luvattomasti. (JSON Web Tokens [Viitattu 8.3.2017].)

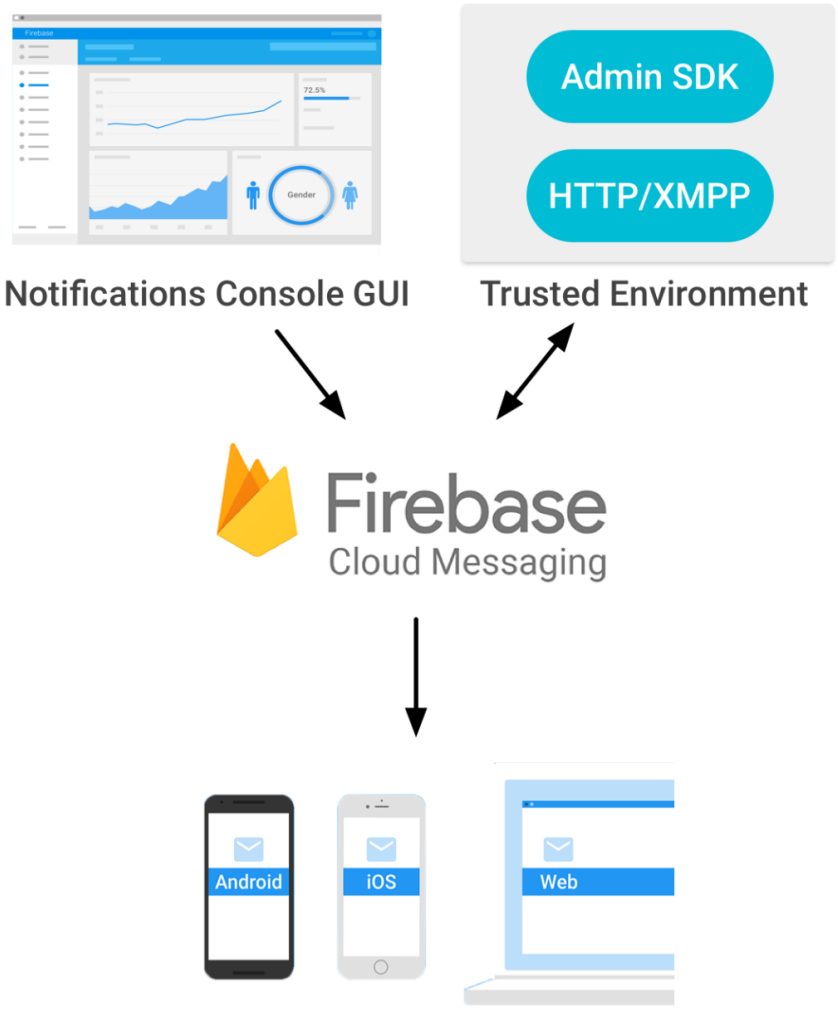
JWT koostuu kolmesta erillisestä osasta, jotka ovat Header, Payload ja Signature (kuvio 20). Payload sisältää lähetettävät tiedot esimerkiksi käyttäjän tiedot. (JSON Web Tokens [Viitattu 8.3.2017].)



Kuvio 20. JWT-esimerkki, jossa vasemmalla on JWT ja oikealla se on purettu. (JSON Web Tokens [Viitattu 8.3.2017]).

3.7 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) on Googlen järjestelmäriippumaton viestintäratkaisu. FCM lähettää ilmoituksia Android-, iOS- ja web-sovelluksiin. (Firebase Cloud Messaging [Viitattu 24.3.2017].) GitHub-verkkosivulta löytyy fcm-push-moduuli, jolla lähetetään ilmoituksia Node.js-palvelimella. Ilmoitukset kulkevat Firebase-alustan kautta sovelluksen käyttäjille (kuvio 21).



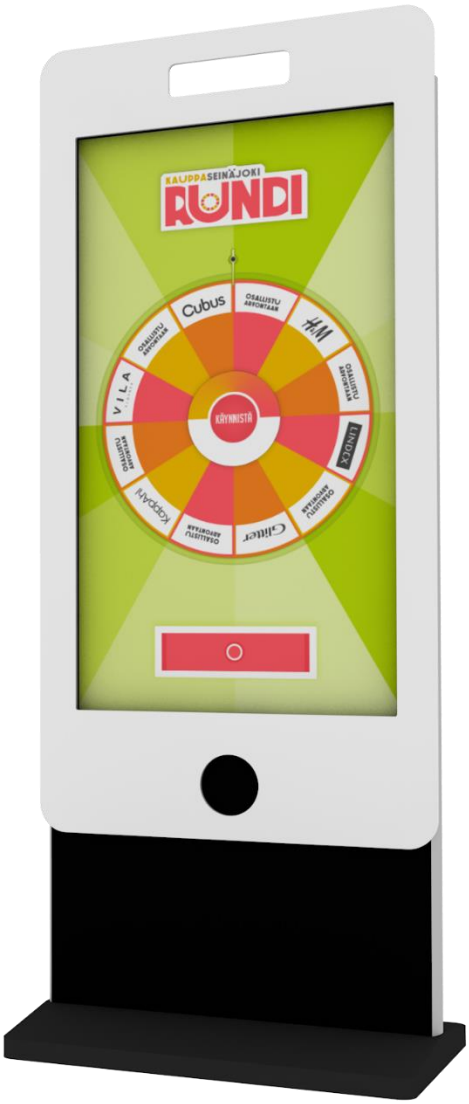
Kuvio 21. Firebase Cloud Messaging -arkkitehtuuri.
(Firebase Cloud Messaging [Viitattu 24.3.2017]).

Firebase-alustan avulla on mahdollista lähettää viestejä myös Firebase-verkkosivun kautta. Viestin voi lähettää yhdelle käyttäjälle tai kaikille sovelluksen käyttäjille samanaikaisesti. Firebase tarjoaa myös reaaliaikaisen tietokannan pilvi-palveluna.

4 CASE: KUPO

Opinnäytetyössä luotiin Kupo-mobiilisovellus React Native -tekniikalla Valakia Interactive Osakeyhtiölle. Mobiilisovelluksen käyttötarkoitus on aloittaa yrityksen tekemä peli kosketusnäytölliseltä infotaululta, joita löytyy esimerkiksi kauppakeskuksista (kuvio 22). Pelin aloitus tapahtuu mobiilisovelluksen QR-koodinlukijalla.

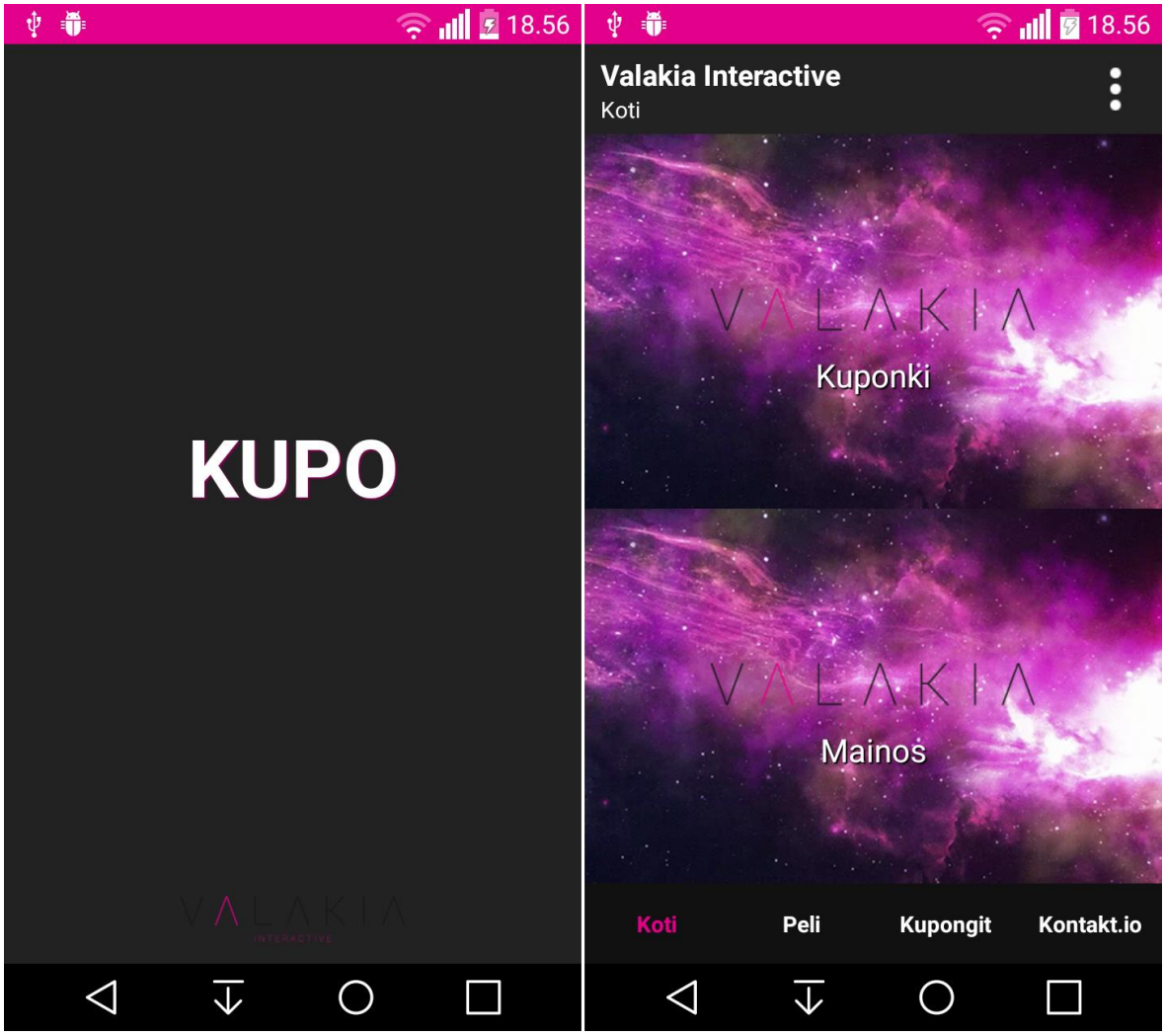
Mobiilisovellus käyttää Node.js-palvelimella olevaa REST-ohjelmointirajapinta, jonka kautta on mahdollista hakea ja päivittää tietoa MongoDB-tietokannasta. Palvelimella on myös React-verkkosivu, jossa viimeistellään käyttäjätilin luominen. Lisäksi React-verkkosivulla voi vaihtaa käyttäjätilin salasanan.



Kuvio 22. Kosketusnäytöllinen infotaulu.
(Valakia Interactive Kauppaseinäjoki [Viitattu 24.3.2017]).

4.1 Mobiilisovelluksen aloitusnäkymät

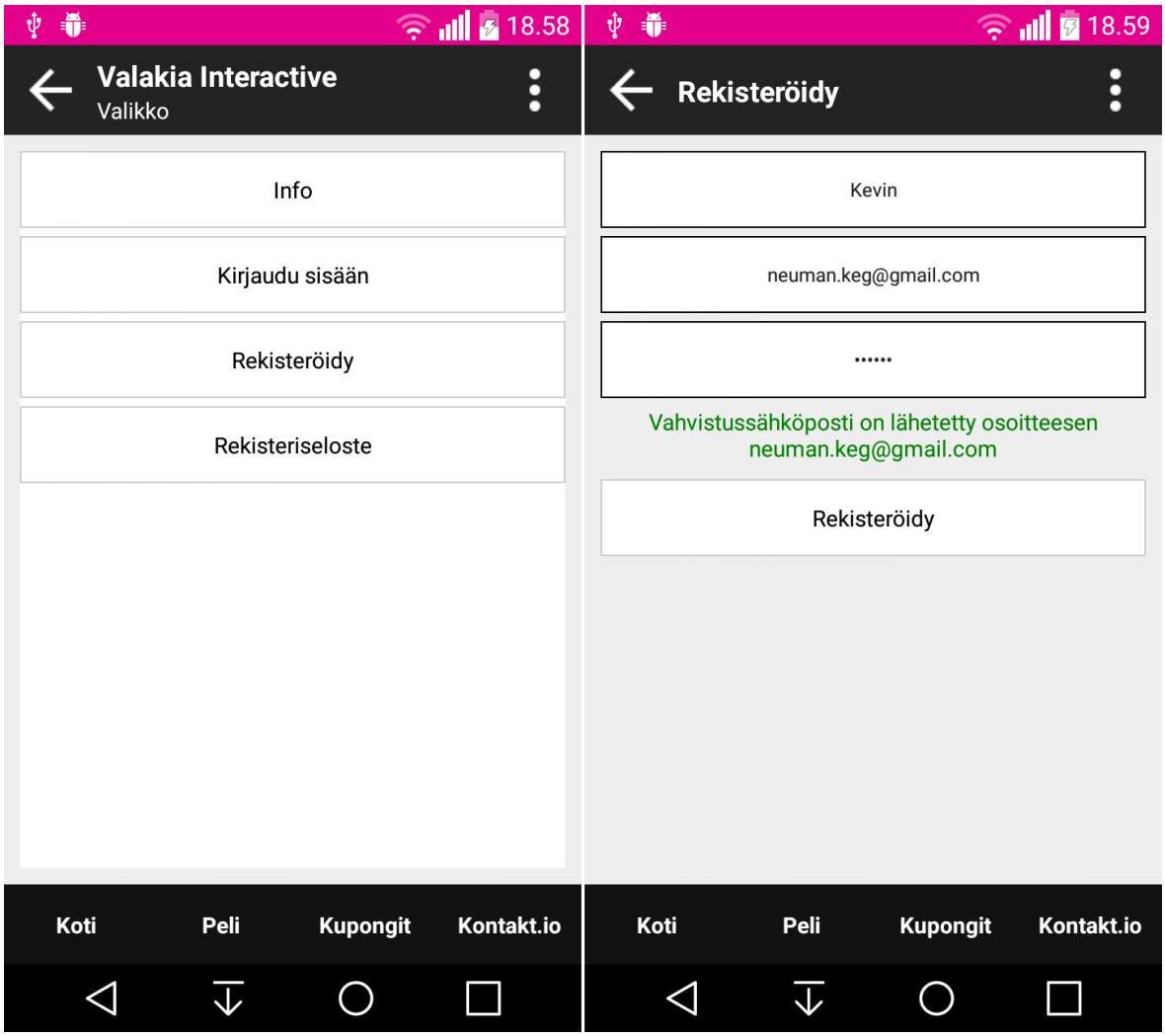
Mobiilisovellus aukeaa Kupo-näkymään, joka muuttuu sekunnin jälkeen Koti-näkymäksi (kuvio 23). Alareunassa sijaitsevan palkin avulla voi navigoida sovelluksessa. Oikeasta ylänurkasta pääsee Valikko-näkymään, jossa voi rekisteröityä ja kirjautua sisään.



Kuvio 23. Mobiilisovelluksen aloitusnäkymät.

4.2 Käyttäjätilin rekisteröiminen

Pelin aloittaminen QR-koodinlukijalla vaatii sisäänkirjautumisen. Mobiilisovelluksessa on kirjautumisjärjestelmä, jonka avulla luodaan oma käyttäjätili. Rekisteröitymiseen tarvitaan käyttäjätunnus, sähköposti ja salasana (kuvio 24).



Kuvio 24. Mobiilisovelluksen Valikko- ja Rekisteröidy-näkymät.

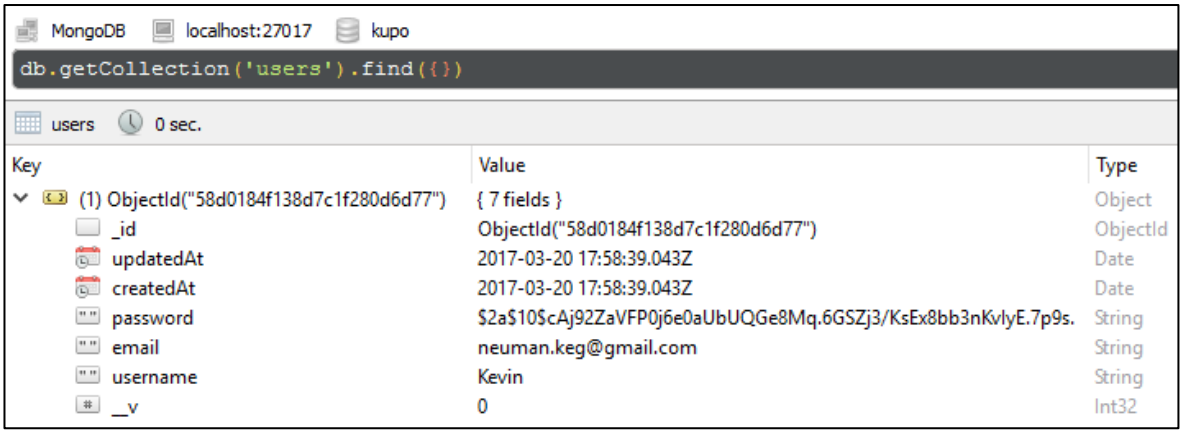
Rekisteröidy-painike (kuvio 24) lähettää annetut tiedot POST-pyyntömenetelmällä REST-ohjelmointirajapinnan api/signup-päätepisteeseen. Node.js-palvelin tarkistaa MongoDB-tietokannasta, ettei käyttäjätunnus tai sähköposti ole jo käytössä. Jos tietokannasta ei löydy samankaltaisuuksia, palvelin lähettää annettuun sähköpostiin linkin käyttäjätilin luomiseen (kuvio 25). Vahvista sähköposti -linkki on voimassa yhden tunnin.

Kuvio 25. Vahvistussähköposti.

React-verkkosivu (kuvio 26) lähettää URL-osoitteen perässä olevan JWT:n POST-pyyntömenetelmällä REST-ohjelmointirajapinnan api/verify-päätepisteeseen.

Kuvio 26. Sähköpostin vahvistussivu.

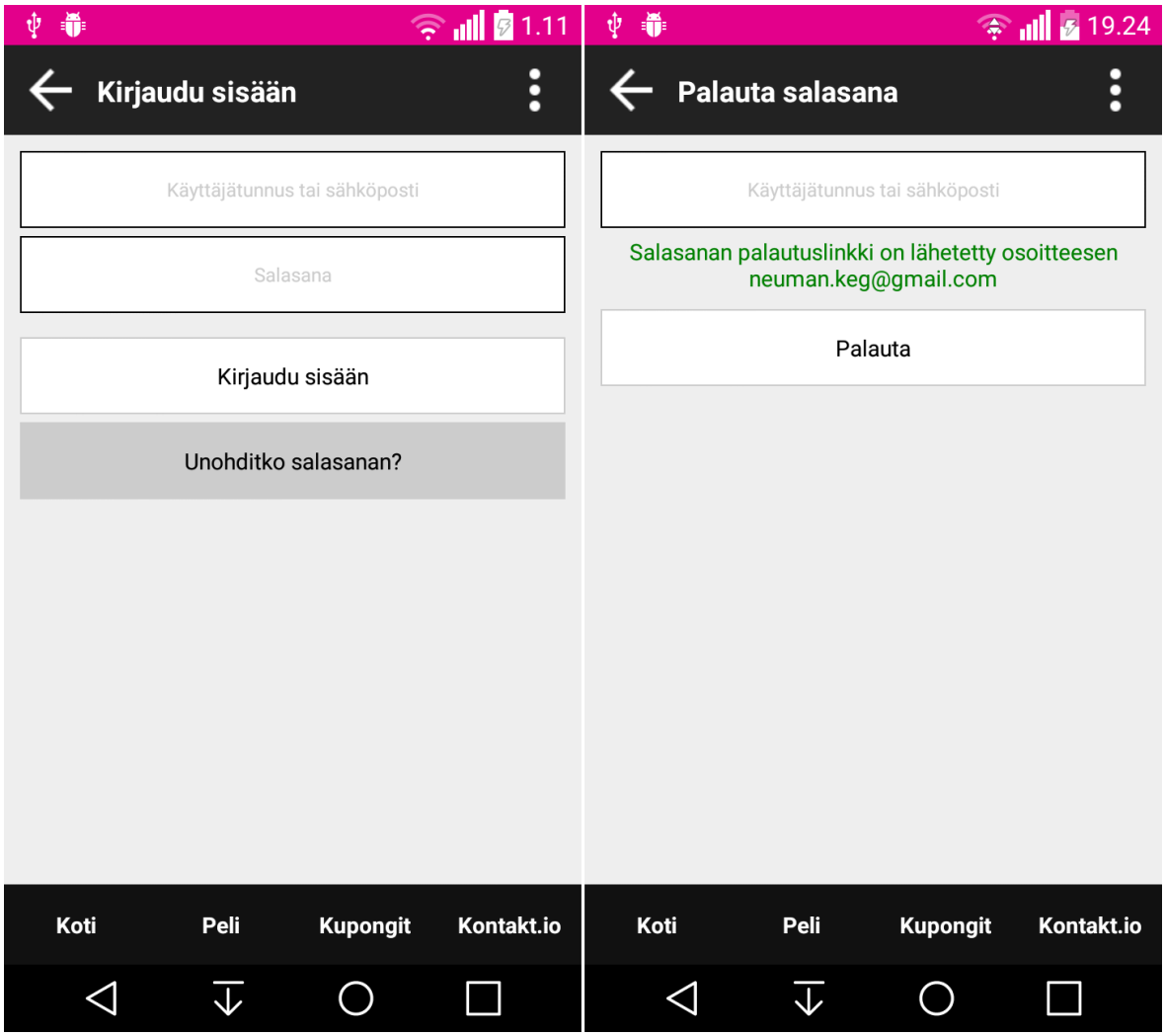
Node.js-palvelin tallentaa uuden käyttäjän MongoDB-tietokantaan. Kuviossa 27 on esitelty käyttäjän tiedot MongoDB-tietokannassa.



Kuvio 27. Käyttäjän tiedot MongoDB-tietokannassa.

4.3 Salasanan palauttaminen

Salasanan unohtuessa voidaan se asettaa uudelleen mobiilisovelluksen avulla. Palauta-painike (kuvio 28) lähettää annetun käyttäjätunnuksen tai sähköpostin POST-pyyntömenetelmällä REST-ohjelmointirajapinnan api/forgot-päätepisteeseen. Sovellukseen tulee ilmoitus, että palautuslinkki on lähetetty annettuun sähköpostiosoitteeseen.

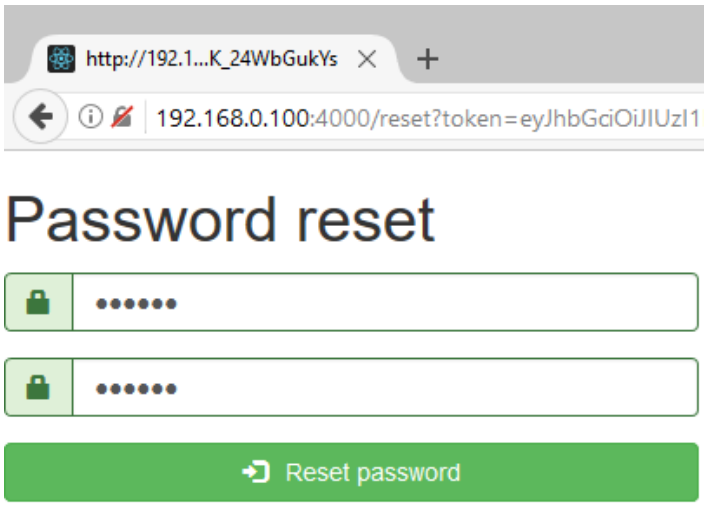


Kuvio 28. Mobiilisovelluksen Kirjaudu sisään- ja Palauta salasana -näkymät.

Jos käyttäjätunnus tai sähköpostiosoite löytyy jo MongoDB-tietokannasta, lähettää Node.js-palvelin sähköpostiin linkin salasanan vaihtoa varten (kuvio 29). Vaihda salasana -linkki on voimassa yhden tunnin.

Kuvio 29. Sähköposti salasanan vaihtamiseen.

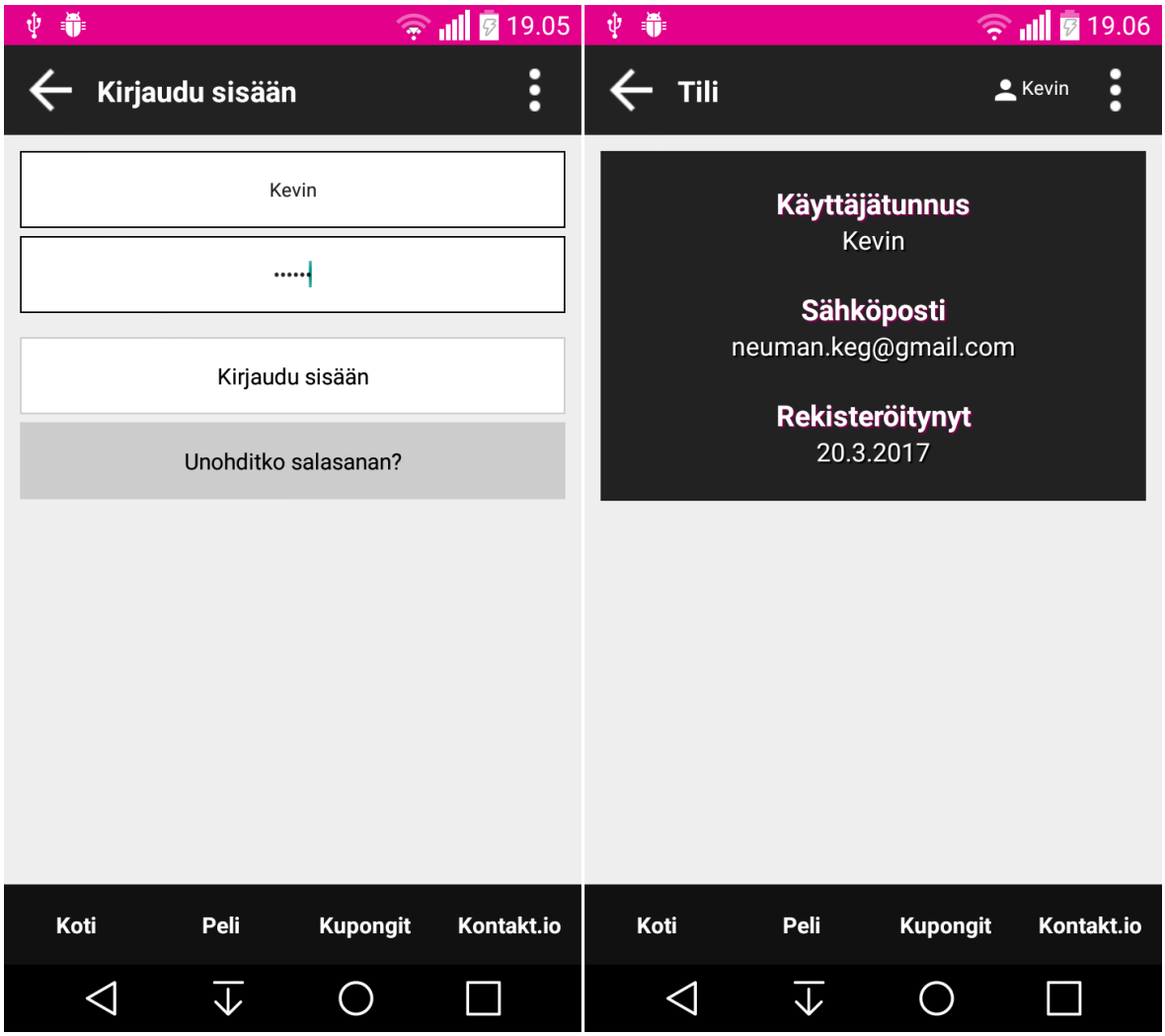
Molempien kenttien salasanojen on oltava vähintään 6 merkkiä pitkiä, sekä niiden tulee vastata toisiaan (kuvio 30). Reset password -painike lähettää uuden salasanan POST-pyyntömenetelmällä REST-ohjelmointirajapinnan api/reset-päätepisteeseen. Node.js-palvelin tallentaa uuden salasanan MongoDB-tietokantaan.



Kuvio 30. Sivu salasanan vaihtamiseen.

4.4 Mobiilisovellukseen kirjautuminen

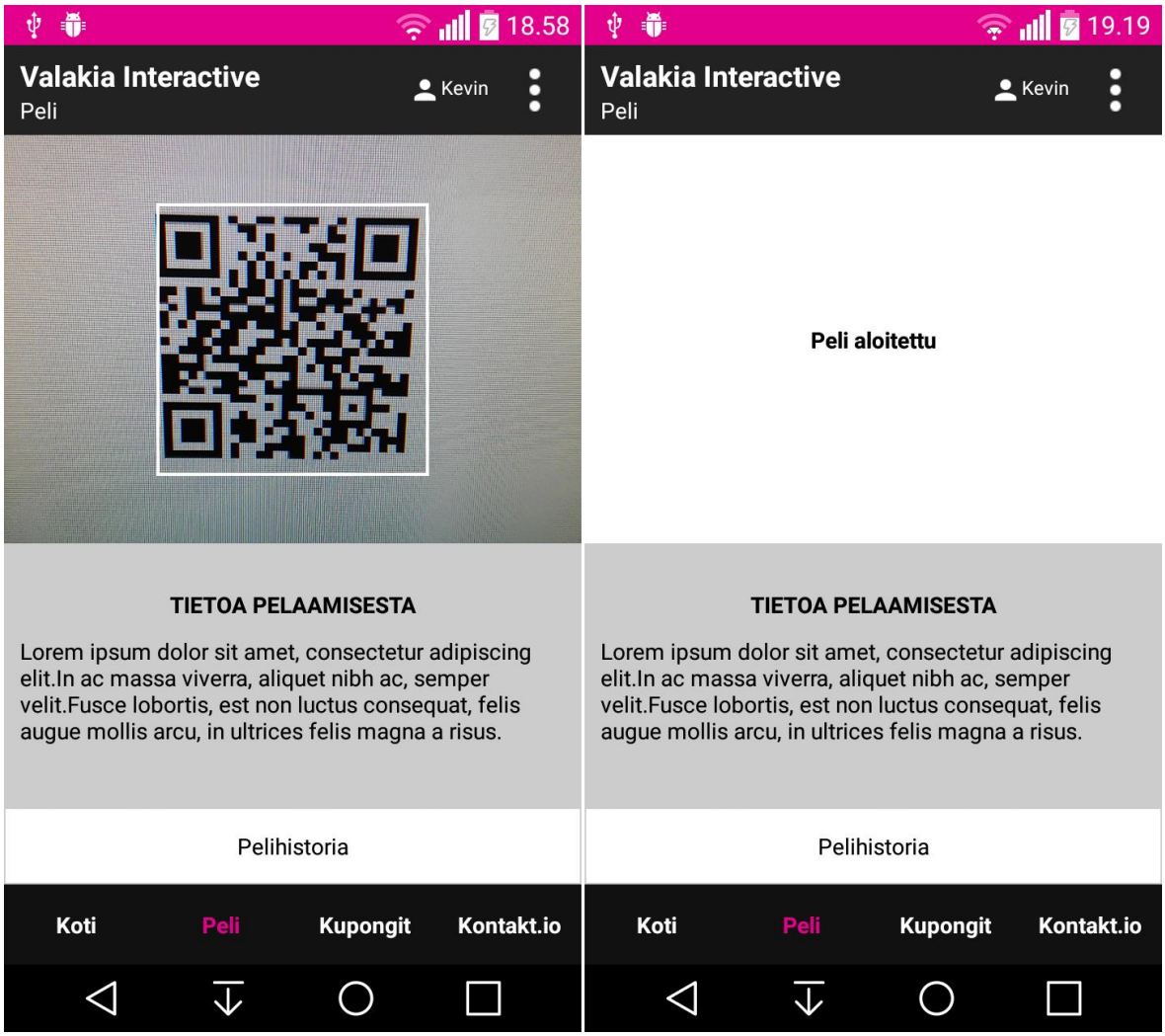
Kirjaudu sisään -painike (kuvio 31) lähettää kirjautumistiedot POST-pyyntömenetelmällä REST-ohjelmointirajapinnan api/login-päätepisteeseen. Node.js-palvelin etsii MongoDB-tietokannasta vastaavan käyttäjätunnuksen tai sähköpostin, jonka jälkeen se vertailee salasanoja. Jos salasanat täsmäävät, palvelin palauttaa käyttäjän tiedot sisältävän JWT:n mobiilisovellukseen.



Kuvio 31. Kirjaudu sisään- ja Tili-näkymät.

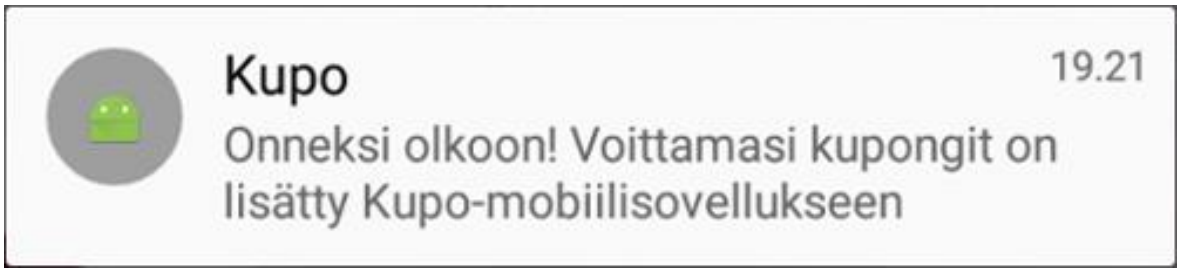
4.5 Mobiilisovelluksen QR-koodinlukija

Pelin aloittamiseen käytettävä QR-koodinlukija sijaitsee Peli-näkymässä (kuvio 32). QR-koodin lukemisen jälkeen peli alkaa käyttäjän tiedoilla.



Kuvio 32. Peli-näkymä.

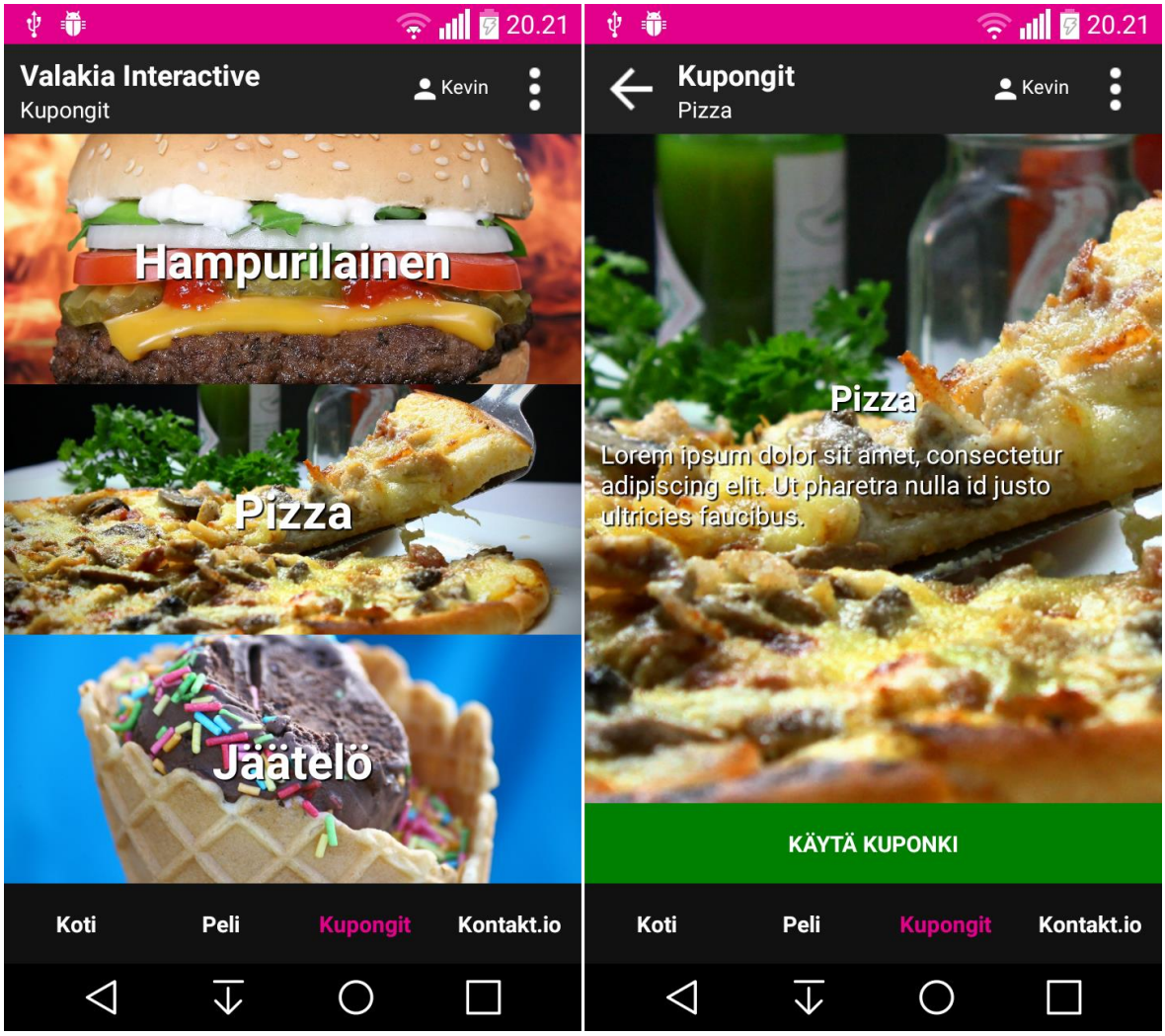
Kun peli loppuu, se lähettää tiedot tuloksista POST-pyyntömenetelmällä REST-rajapintaan. Node.js-palvelin tarkistaa, onko kyseessä voitto tai häviö. Palvelin tallentaa tuloksen käyttäjän tietoihin MongoDB-tietokantaan. Palvelin lähettää ilmoituksen voitosta puhelimeen push-viestinä (kuvio 33).



Kuvio 33. Puhelimeen tullut push-viesti voitosta.

4.6 Mobiilisovelluksen kupongit

Pelistä on mahdollista voittaa erilaisia palkintoja alennuskuponkien muodossa. Mobiilisovelluksen Kupongit-näkymä sisältää käyttäjän kaikki kupongit. Kuponki aukeaa koko näytöllä, kun sitä painaa (kuvio 34). Kupongin voi käyttää KÄYTÄ KUPONKI -painikkeesta, jonka jälkeen kuponki näkyy listassa harmaana.



Kuvio 34. Kuponki-näkymät.

4.7 Mobiilisovelluksen tulevaisuuden näkymät

Kupo-mobiilisovellus on tällä hetkellä kehitetty vain Android-käyttöjärjestelmälle. React Native -tekniikan ansiosta sen lähdekoodia ei tarvitse muuttaa paljon, että siitä saataisiin tehtyä iOS-sovellus. Sen jälkeen sovellus voitaisiin julkaista Google Play -sovelluskaupan lisäksi myös App Store -sovelluskaupassa.

5 YHTEENVETO JA POHDINTA

Opinnäytetyössä tutustuttiin moderneihin käyttöliittymän sekä palvelinpuolen tekniikoihin. Näitä tekniikoita käytettiin React Native -mobiilisovelluksen ja React-verkkosivun käyttöliittymien luomiseen sekä Node.js-palvelinpuolen toteuttamiseen. Palvelinpuolen REST-ohjelmointirajapinta syntyi nopeasti Node.js -ajoympäristön ja Express.js-web-kehyksen avulla.

Tässä opinnäytetyössä kehitettiin Kupo-mobiilisovellus, jolla voi aloittaa pelin kosketusnäytölliseltä infotaululta mobiilisovelluksen QR-koodinlukijalla. Mobiilisovellus saa tietonsa palvelimen REST-ohjelmointirajapinnasta. Mobiilisovellus käyttää apunaan verkkosivua käyttäjätilin luomisen viimeistelyyn ja käyttäjän salasanan vaihtamiseen. Työn kehitys eteni ilman suurempia ongelmia ja asetetut tavoitteet toteutuivat aikataulun mukaisesti.

Ongelmana opinnäytetyössä ilmeni Kupo-mobiilisovelluksen kirjautumisjärjestelmään liitetty vaihtoehto, jolla pystyi rekisteröitymään ja kirjautumaan puhelinnumerolla. Mobiilisovellukseen annettiin puhelinnumero, jonka jälkeen puhelimeen saapui tekstiviestillä numerosarja. Numerosarjan avulla pystyi luomaan käyttäjätilin ja kirjautua sovellukseen. Tämä menetelmä olisi nopeuttanut käyttäjätilin luontia ja kirjautumisprosessia. Siitä luovuttiin satunnaisten ongelmien takia, joihin tässä työssä ei perehdytty.

Tämän työn perusteella voidaan sanoa, että React ja React Native ovat hyviä tekniikoita dynaamisten käyttöliittymien luomiseen. React JavaScript -kirjaston toimintaperiaatteiden ja renderöintilogiikan omaksumisen jälkeen on helppo siirtyä esimerkiksi React-web-kehityksestä React Native -mobiilisovelluskehitykseen. Tämä johtuu siitä, että React Native perustuu React JavaScript -kirjastoon eli niiden koodi on samankaltaista.

Node.js yhdessä MongoDB-tietokannan kanssa mahdollistaa palvelinpuolen toteuttamisen suuressakin mittakaavassa. Lisäksi palvelimen kehitys tapahtuu JavaScript-ohjelmointikielellä, mikä tekee siitä houkuttelevan vaihtoehdon.

Tämä opinnäytetyö lisäsi ja vahvisti osaamista työssä käytettyjen tekniikoiden osalta. Tulevaisuudessa tämän työn tekijän on helpompi lähteä opettelemaan ja kokeilemaan uusia tekniikoita. Uskon, että tästä työstä on hyötyä myös muille ohjelmistokehittäjille, jotka haluavat oppia näitä tekniikoita.

LÄHTEET

About Node.js. Ei päiväystä. About Node.js. [Verkkosivu]. Node.js Foundation. [Viitattu 9.2.2017]. Saatavana: https://nodejs.org/en/about

Express "Hello World" example. Ei päiväystä. Express "Hello World" example. [Verkkosivu]. Node.js Foundation. [Viitattu 9.2.2017]. Saatavana: http://expressjs.com/en/starter/hello-world.html

Express basic routing. Ei päiväystä. Express basic routing. [Verkkosivu]. Node.js Foundation. [Viitattu 9.2.2017]. Saatavana: http://expressjs.com/en/starter/basic-routing.html

Express. Ei päiväystä. Node.js web application framework. [Verkkosivu]. Node.js Foundation. [Viitattu 9.2.2017]. Saatavana: http://expressjs.com

Firebase Cloud Messaging. Ei päiväystä. Firebase Cloud Messaging. [Verkkosivu]. Google. [Viitattu 24.3.2017]. Saatavana: https://firebase.google.com/docs/cloud-messaging

Gackenheimer. C. 2015. Introduction to React. Apress, 1–2.

GitHub webpack. 7.2.2017. GitHub webpack README. [Verkkosivu]. GitHub. [Viitattu 9.3.2017]. Saatavana: https://github.com/webpack/webpack/blob/ea4be07d451e42c6ee02854dd8bffa3a3d3e37c8/README.md

Journey into React Part 6. 26.8.2016. Journey into React Part 6. [Verkkosivu]. eMeents Media. [Viitattu 24.3.2017]. Saatavana: https://www.davidmeents.com/blog/manage-state-connect-to-api-redux-axios

JSON Web Tokens. Ei päiväystä. JSON Web Tokens Introduction. [Verkkosivu]. Auth0. [Viitattu 8.3.2017]. Saatavana: https://jwt.io/introduction

Krol. J. 2014. Web Development with MongoDB and Node.js. Packt Publishing, 8–10.

Libscore. Ei päiväystä. Libscore. [Verkkosivu]. Libscore. [Viitattu 8.3.2017]. Saatavana: http://libscore.com/#React

List of webpack plugins. Ei päiväystä. List of webpack plugins. [Verkkosivu]. Webpack. [Viitattu 9.2.2017]. Saatavana: https://webpack.github.io/docs/list-of-plugins.html#uglifyjsplugin

Membrey. P., Hows. D. & Plugge. E. 2014. MongoDB Basics. Apress, 2.

MongoDB. Ei päiväystä. What Is MongoDB. [Verkkosivu]. MongoDB. [Viitattu 1.3.2017]. Saatavana: https://www.mongodb.com/what-is-mongodb

Mongoose. Ei päiväystä. Mongoose Schemas. [Verkkosivu]. Mongoose. [Viitattu 8.3.2017]. Saatavana: http://mongoosejs.com/docs/guide.html

Node.js. Ei päiväystä. Node.js. [Verkkosivu]. Node.js Foundation. [Viitattu 9.2.2017]. Saatavana: https://nodejs.org/en

QR Codes Explained. 6.5.2013. QR Codes Explained. [Verkkosivu]. How-To Geek. [Viitattu 4.4.2017]. Saatavana: https://www.howtogeek.com/162394/qr-codes-explained-why-you-see-those-square-barcodes-everywhere

Raymond. E. 2003. The Art of UNIX Programming. Addison-Wesley Professional, 14.

React Installation. Ei päiväystä. React Installation. [Verkkosivu]. Facebook. [Viitattu 8.3.2017]. Saatavana: https://facebook.github.io/react/docs/installation.html

React Native Camera. Ei päiväystä. React Native Camera. [Verkkosivu]. GitHub. [Viitattu 23.3.2017]. Saatavana: https://github.com/lwansbrough/react-native-camera

React Native Installation. Ei päiväystä. React Native Installation. [Verkkosivu]. Facebook. [Viitattu 23.3.2017]. Saatavana: https://facebook.github.io/react-native/docs/getting-started.html

React Native. Ei päiväystä. A framework for building native apps using React. [Verkkosivu]. Facebook. [Viitattu 9.2.2017]. Saatavana: https://facebook.github.io/react-native

React. Ei päiväystä. A JavaScript library for building user interfaces. [Verkkosivu]. Facebook. [Viitattu 7.2.2017]. Saatavana: https://facebook.github.io/react

Redux Actions. Ei päiväystä. Redux Actions. [Verkkosivu]. Redux. [Viitattu 24.3.2017]. Saatavana: http://redux.js.org/docs/basics/Actions.html

Redux Motivation. Ei päiväystä. Redux Motivation. [Verkkosivu]. Redux. [Viitattu 8.3.2017]. Saatavana: http://redux.js.org/docs/introduction/Motivation.html

Redux Read Me. Ei päiväystä. Redux Read Me. [Verkkosivu]. Redux. [Viitattu 8.2.2017]. Saatavana: http://redux.js.org

Understanding REST. Ei päiväystä. Understanding REST. [Verkkosivu]. Pivotal Software. [Viitattu 9.2.2017]. Saatavana: https://spring.io/understanding/REST

Valakia Interactive Kauppaseinäjoki. Ei päiväystä. Valakia Interactive Kauppaseinäjoki. [Verkkosivu]. Valakia Interactive. [Viitattu 24.3.2017]. Saatavana: http://www.valakia.fi/project/kauppaseinajoki

Valakia Interactive. Ei päiväystä. Valakia Interactive. [Verkkosivu]. Valakia Interactive. [Viitattu 12.4.2017]. Saatavana: http://www.valakia.fi

What is webpack. Ei päiväystä. What is webpack. [Verkkosivu]. Webpack. [Viitattu 9.2.2017]. Saatavana: http://webpack.github.io/docs/what-is-webpack.html