jamk.fi

Firstbeat Sports Admin Client -työkalun suunnittelu ja toteutus

Teemu Tuomela

Opinnäytetyö
Toukokuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), Ohjelmistotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Tuomela, Teemu	Opinnäytetyö, AMK	Toukokuu 2018
	Sivumäärä 38	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Firstbeat Sports Admin Client -työkalun suunnittelu ja toteutus

Tutkinto-ohjelma
Ohjelmistotekniikan tutkinto-ohjelma

Työn ohjaaja(t)
Rantala Ari

Toimeksiantaja(t)
Firstbeat Technologies Oy

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi jyväskyläläinen Firstbeat Technologies Oy. Firstbeatin sydämen sykevälivaihteluun perustuvaa teknologiaa käytetään hyvinvointiin, kuluttajatuotteisiin sekä huippu-urheiluun. Firstbeat Sports on huippu-urheilun tarpeisiin kehitetty alusta, joka sisältää sovelluksia harjoitusten mittaamiseen sekä niiden tulosten tarkasteluun. Firstbeat Sports Admin Client on verkkoselaimessa toimiva web-sovellus, jota käyttäen luodaan uusia asiakastilejä sekä käyttäjiä Sports-alustan käyttöä varten. Firstbeat Sports Admin Clientia käyttävät myyjät, tuotetuen henkilöt, ohjelmistokehittäjät ja ohjelmistotestaajat. Käytössä olevan sovelluksen käytettävyydessä ja ylläpidettävyydessä oli havaittu puutteita. Opinnäytetyön tavoitteena oli kehittää sovelluksesta uusi versio käyttäen nykyaikaisia menetelmiä web-sovelluksen kehittämiseen.

Työ aloitettiin määrittelemällä toteutettavan sovelluksen vaatimukset olemassa olevan sovelluksen pohjalta. Opinnäytetyön aikana suunniteltiin uuden sovelluksen käyttöliittymä sekä arkkitehtuuri. Uuden sovelluksen käyttöliittymä toteutettiin käyttäen avoimen lähdekoodin React-kirjastoa. Sovelluksen tilanhallintaan ja datan välittämiseen sovelluksen eri osien välillä käytettiin Redux-kirjastoa.

Työn tuloksena syntyi lähes kaikki vaatimukset täyttävä sovellus. Tuotetulla sovelluksella on muun muassa mahdollista selata, luoda ja muokata asiakastilejä sekä käyttäjätunnuksia. Työssä tuotiin esille tietoa Reactista ja sovelluskehyksistä GWT sekä Angular, mikä voi auttaa toimeksiantajaa valitsemaan teknologioita yrityksen ohjelmistokehityksen jatkoa ajatellen.

Avainsanat (asiasanat)
React, Redux, Angular, GWT, JavaScript, Web-sovelluskehitys

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Tuomela, Teemu	Bachelor's thesis	May 2018
		Language of publication: Finnish
	Number of pages: 38	Permission for web publication: x

Title of publication

Design and implementation of Firstbeat Sports Admin Client

Degree programme

Software Engineering

Supervisor(s)

Rantala Ari

Assigned by

Firstbeat Technologies Oy

Abstract

Firstbeat Technologies Oy is a Finnish company competing in the field of wellness, consumer products and sports. Firstbeat's technologies are based on extensive research on heart rate variability. Firstbeat Sports is a platform designed to meet the requirements of the very best teams in many different sports. The platform includes software to monitor and analyse the training sessions of a whole team. Firstbeat Sports Admin Client is a web application used to create new customer accounts and credentials for the platform. It is used by sales, support and software development. The current Firstbeat Sports Admin Client has had issues with usability and maintainability. The main goal of the thesis was to develop a new version of the application using modern web development practises and technologies.

The new application had to match the requirements of the existing application. The user interface as well as the architecture of the software were designed during the thesis project. The user interface of the application was developed using React library. Redux-library was used to manage the state of the application and communication between different parts of the application.

The application developed during the thesis was able to fill almost all set requirements. The functionality includes browsing, creating and editing customer accounts and credentials. The thesis brings information about React and the software frameworks GWT and Angular that may be of use to help Firstbeat decide on future technologies for its software development team.

Keywords/tags (subjects)

React, Redux, Angular, GWT, Web application development

Miscellaneous

Sisältö

Kuviot

Termit ja lyhenteet

Base64

Koodausmenetelmä, jonka avulla esimerkiksi binääridata voidaan ilmaista ASCII-merkkijonona.

CSS

Cascading Style Sheets. Kieli, jolla määritellään tarkemmin verkkosivun esittämistapa kuten fontit ja värit.

DOM

Domain Object Model. DOM on tapa kuvata esimerkiksi HTML-dokumentin rakenne puurakenteena. Tämä mahdollistaa dokumentin lukemisen ja muokkaamisen ohjelmallisesti.

ECMAScript

Ecma Internationalin määrittelemä standardi, jossa määriteltyjä ominaisuuksia JavaScript-ohjelmointikieli toteuttaa.

Esikääntäjä

Esikääntäjä suorittaa lähdekoodille toimenpiteitä ennen kuin se päätyy lopulliselle kääntäjälle. Esimerkiksi TypeScript-esikääntäjä kääntää TypeScript-kielellä kirjoitetun tiedoston tavalliseksi selaimen ymmärtämäksi JavaScript-kieleksi.

HTML

Hypertext Markup Language. Merkintäkieli, jolla määritellään verkkosivun rakenne elementteinä. Verkkoselain esittää sivun käyttäjälle HTML-dokumentin rakenteen mukaisesti.

HTTP

Hypertext Transfer Protocol. Tiedonsiirtoprotokolla, jota verkkoselaimet käyttävät tiedonsiirtoon.

JavaScript

Ohjelmointikieli, jota verkkoselain suorittaa toteuttaakseen dynaamisuutta verkkosivulla.

Koodin minimointi

Koodin minimoinnilla tarkoitetaan lähdekooditiedostojen tiivistämistä. Usein tämä tarkoittaa muuttujien uudelleen nimeämistä mahdollisimman lyhyiksi sekä ylimääräisten välilyöntien ja rivinvaihtojen poistamista lähdekoodista.

Otsake

Otsakkeilla voidaan välittää ylimääräistä tietoa HTTP-kutsun mukana. Otsake koostuu otsakkeen nimestä sekä sen arvosta kaksoispisteellä erotettuna.

Palvelin

Ohjelmisto sekä sitä suorittava tietokone, joka tarjoilee verkkoyhteyden yli palveluita kuten verkkosivuja niitä pyytäville asiakkaille.

Sovelluskehys

Sovelluskehys tai ohjelmistokehys on kokoelma valmiita toteutuksia ja apuvälineitä, joita käyttäen rakennetaan tiettyyn tarkoitukseen kohdennettu sovellus.

1 Työn lähtökohdat

1.1 Tausta

Verkkosivut ovat kehittyneet viime vuosina hurjalla vauhdilla. Enää verkkosivut eivät ole vain staattisen tiedon esittämistä käyttäjälle, vaan ne ovat yhä kasvavissa määrin vuorovaikutuksessa käyttäjien kanssa. Monet verkkosivut muistuttavat niin ominaisuuksien määrältään kuin ulkoasultaankin perinteisiä tietokoneelle asennettavia ohjelmia. Tällaisia toiminnallisuutta sisältäviä verkkosivuja kutsutaan web-sovelluksiksi.

Web-sovelluksilta odotetaan paljon ja niiden on kyettävä vastaamaan käyttäjien odotuksiin ominaisuuksiltaan ja suorituskyvyltään. Käyttäjät ovat tottuneet Netflixin ja Spotifyn kaltaisiin web-sovelluksiin, jotka tarjoajat käyttäjälle sisältöä nopeasti ja jouhevasti. Työpöytäsovellusta vastaavan käyttökokemuksen tarjoaminen verkon yli suoraan käyttäjän verkkoselaimeen ilman asennuksia asettaa web-sovellusten kehittämiselle monia haasteita.

1.2 Toimeksiantaja

Työn toimeksiantajana toimi vuonna 2002 perustettu jyväskyläläinen yritys Firstbeat Technologies Oy (myöhemmin Firstbeat). Firstbeat on kansainvälinen hyvinvoinnin, huippu-urheilun ja kuluttajatuotteiden parissa toimiva yritys. Firstbeatin ydinosaamista on sydämen sykereaktioiden ja sykevälivaihtelun analyysiin perustuva teknologia, jolla mahdollistetaan aiempaa tarkempi mittaus ilman laboratoriolaitteistoa. (Tarinamme n.d.)

Firstbeatin hyvinvointianalyysin avulla voidaan löytää hyvinvointia tukevat tai kuluttavat elämäntavat sekä kuormittumisen tai heikon palautumisen syyt. Mittaus suoritetaan kolmen vuorokauden aikana, jonka aikana mittalaite kerää ympärivuorokautisesti dataa sydämen sykevälivaihtelusta. Samalla asiakas pitää päiväkirjaa työ- ja unijaksoista sekä muista tärkeistä tapahtumista. Mittauksen päätyttyä asiakas saa hyvinvointiasiantuntijalta palautetta sekä suosituksia hyvinvoinnin parantamiseksi. (Hyvinvoinnin ammattilaiset n.d.; Hyvinvointianalyysi n.d.)

Firstbeat Sports sen sijaan on huippu-urheilun tarpeisiin suunniteltu alusta, joka kokoaa joukkueen syketiedot ja niiden tulkinnan. Sports-tuoteperhe sisältää reaaliaikamonitorointisovelluksen, mobiilisovelluksen sekä selaimella käytettävän sovelluksen tulosten tarkasteluun. Firstbeat Sportsin avulla voidaan seurata joukkueen syketietoja ja harjoituksen tavoitteiden täyttymistä reaaliajassa. Sports tuo mahdollisuuden myös palautumisen tarkkailuun sekä maksimaalisen hapenottokyvyn arviointiin. (Joukkueurheilu n.d.)

2 Työn kuvaus

2.1 Firstbeat Sports Admin Client

Firstbeat Sports Admin Client (myöhemmin Admin Client) on Firstbeatin Sports-tuotteen asiakastilien hallintaan käytettävä web-sovellus. Admin Clientilla voidaan luoda asiakkaalle asiakastili sekä asiakastiliin liitettyjä käyttäjätunnuksia, joilla asiakas pääsee kirjautumaan ja aloittamaan Sports-tuotteiden käytön.

Asiakastilit sisältävät tietoa asiakkaasta kuten nimen, yhteystiedot, tilin voimassaoloajan, sallitut ominaisuudet sekä muita asetuksia.

Käyttäjätunnuksia voidaan luoda kahta eri tyyppiä, joista molemmilla on samat perustiedot kuten käyttäjätunnus, salasana ja sähköposti.

Valmentaja-tyyppinen käyttäjätunnus on tarkoitettu valmentajalle. Valmentajana käyttäjä voi luoda ja muokata joukkueita, valmentajia sekä urheilijoita. Valmentaja voi käyttää Sports Monitor -sovellusta harjoituksien tallentamiseen sekä tarkastella näitä harjoituksia Sports Cloud -web-sovelluksella.

Urheilija-tyyppinen käyttäjätunnus on tarkoitettu joukkueen urheilijalle, jolle tallennetaan harjoituksia. Myös urheilija voi kirjautua Sports Cloud-sovellukseen, mutta urheilija voi tarkastella ainoastaan omia mittauksiaan, ei joukkuetovereiden. Urheilija voi myös kirjautua tunnuksillaan Sports-mobiilisovellukseen, jolla hän voi tallentaa omia henkilökohtaisia harjoituksiaan.

Sports-tuotteen myyjät käyttävät Admin Client -sovellusta uusien asiakastilien luomiseen ja asetusten, kuten voimassaoloajan ja urheilijoiden määrän muokkaamiseen

sekä sallittujen ominaisuuksien ja palveluiden aktivoimiseen asiakkaan sopimuksen mukaisesti.

Toinen suuri käyttäjäkunta Admin Client -sovellukselle on tuotetuki. Tuotetuella on usein tarvetta tarkastella asiakastilin ja käyttäjätunnusten asetuksia selvittääkseen ongelmatilanteita. Tuotetuki myös auttaa asiakasta luomaan ja muokkaamaan käyttäjätunnuksia, mikäli asiakas ei tässä itse onnistu.

Admin Client on myös jatkuvassa käytössä Sports-tuotteiden kehitystyössä sekä ohjelmistotestauksessa.

2.2 Tehtävänanto

Opinnäytetyön tehtävänantona oli toteuttaa Admin Clientista uusi päivitetty versio. Nykyinen sovellus on jäänyt ajasta jälkeen suorituskyvyltään, käytettävyydeltään ja ulkoasultaan. Nämä ovat Admin Clientille erittäin tärkeitä osa-alueita sen suuren käytön vuoksi.

Vanha Admin Client on toteutettu Google Web Toolkitiä eli GWT:ta käyttäen. Uutta sovellusta haluttiin lähteä toteuttamaan jollakin uudemmalla teknologialla, jotta samalla saataisiin tietoa ja kokemusta erilaisten käyttöliittymäkirjastojen tuomista eduista.

2.3 Työn rajaus

Opinnäytetyö rajattiin pääasiassa käyttöliittymäpuolen toteutukseen. Jo olemassa olevia palvelinrajapintoja ei tarvinnut lähteä muokkaamaan. Uutena ominaisuutena Admin Clientiin toivottiin statistiikan näyttämistä käyttäjälle. Tätä varten palvelimelle tarvitaan uusi kutsuttava rajapinta tietokantakyselyineen, josta Admin Client voi hakea esitettävän tiedon. Tietokannan rakenteeseen ei tässä työssä tarvinnut puuttua.

Sovelluksen toteuttamiseen valittiin React JavaScript-kirjasto jo projektin alkuvaiheessa. React oli opinnäytetyön tekijälle jo ennestään tuttu oman harrastuneisuuden kautta. Myös Firstbeatillä Reactia on varovasti otettu käyttöön muihin tuotteisiin. Nämä huomioiden React oli hyvä valinta uuden Admin Clientin toteuttamiseen.

2.4 Tavoitteet

Opinnäytetyön tavoitteena oli tuottaa valmis sovellus, joka kyetään ottamaan käyttöön kesän 2018 aikana. Tavoitteena oli tehdä vaatimusten mukainen sovellus, joka helpottaa sitä käyttävien ihmisten työntekoa verrattuna vanhaan sovellukseen. Sovelluksesta haluttiin tehdä myös helposti ylläpidettävä ja jatkokehitettävä, sillä opinnäytetyön tekijä työskentelee sovelluksen parissa päivittäin myös jatkossa.

Tavoitteena oli tehdä sovelluksesta hyviä ja kestäviä käytäntöjä noudatteleva toteutus, joka voisi toimia mallina ja tutustuttavana alustana Reactiin muille Firstbeatin ohjelmistokehittäjille. Sovellus voi tuoda toimeksiantajalle arvokasta tietoa Reactin käytöstä ja auttaa valitsemaan teknologioita tuleviin projekteihin.

3 Vaatimusmäärittely

3.1 Yleistä

Vaatimusmäärittely on ohjelmistotuotannossa vaihe, jonka aikana määritellään, mitä sovelluksen täytyy kyetä tekemään, mitä palveluita se tarjoaa ja mitä rajoituksia sen toiminnalle on. Vaatimusmäärittelyllä kuvataan asiakkaan tarpeet tuotettavalle sovellukselle ja sen avulla kyetään toteuttamaan asiakkaan tarpeisiin ja toiveisiin soveltuva tuote. (Sommerville 2009, 83.)

Vaatimuksia voidaan määritellä eri tasoilla. Käyttäjävaatimusten tulisi määritellä funktionaaliset ja ei-funktionaaliset vaatimukset niin, että ne ovat ymmärrettävissä myös ilman syvempää teknistä tietämystä. Niiden olisi hyvä määritellä vain sovelluksen ulkoisia vaikutuksia, ja ne tulisi kirjoittaa luonnollisella kielellä ilman hankalaa teknistä termistöä. Järjestelmävaatimukset taasen jatkavat käyttäjävaatimuksia teknisellä tasolla. Järjestelmävaatimukset on tarkoitettu ohjelmistoinsinöörille järjestelmän suunnittelun lähtökohdaksi ja ne määrittelevät, millä tavoin toiminnallisuudet tulee toteuttaa sovelluksen sisällä. (Sommerville 2009, 94.)

3.2 Toiminnalliset vaatimukset

Toiminnalliset vaatimukset kuvaavat, mitä palveluita sovelluksen tulee kyetä tarjoamaan tai miten sovellus reagoi tiettyihin syötteisiin ja tilanteisiin. Toiminnalliset vaatimukset voivat myös joskus kuvata asioita, joita sovelluksen ei tulisi tehdä. Käyttäjävaatimuksina ilmaistuna ne tulisi kirjoittaa tarpeeksi yleisellä tasolla, jotta ne ovat ymmärrettävissä sovelluksen käyttäjän näkökulmasta. (Sommerville 2009, 84-86.)

Uuden Admin Clientin vaatimusmäärittely tehtiin vanhan sovelluksen pohjalta. Uuden sovelluksen tuli kyetä toteuttamaan kaikki samat toiminnot kuin vanhankin version, jotta se voitaisiin korvata. Vaatimuksina olivat luonti-, poisto- ja muokkaustoiminnot niin asiakastileille kuin käyttäjätunnuksillekin. Käyttäjätunnuksia täytyi olla mahdollista siirtää toiselle asiakastilille ja tunnuksille täytyi olla mahdollista lähettää salasananpalautuslinkki käyttöliittymän kautta. Urheilijoiden mittaukset täytyi olla mahdollista tuoda järjestelmästä ulos käyttöliittymän kautta.

3.3 Ei-toiminnalliset vaatimukset

Ei-toiminnalliset vaatimukset antavat vaatimuksia ja rajoitteita sovelluksen palveluiden ja funktioiden toiminnalle. Ne sisältävät usein rajoituksia vasteajan, luotettavuuden ja turvallisuuden suhteen. Ei-toiminnalliset vaatimukset vaikuttavat usein koko sovelluksen toimintaa yksittäisten ominaisuuksien sijaan. (Sommerville 2009, 87.)

Uudelle Admin Clientille määriteltiin yksi ei-toiminnallinen vaatimus; verkko-operaatioiden suorittaminen ei saa lukita käyttöliittymää. Käyttäjän täytyy voida jatkaa sovelluksen käyttöä latauksen aikana niiltä osin, kuin se on mahdollista.

4 Web-sovelluskehitys

4.1 Yleistä

Tavallisen staattisen verkkosivun luomiseen käytetään yleensä HTML-kuvauskieltä sekä CSS-tyylikieltä. HTML eli Hypertext Markup Language määrittelee sivun rakenteen, ja CSS-kielellä annetaan tarkempia ohjeita sisällön esittämisestä kuten värit ja fontit. Näitä verkkosivuja yhteen linkittämällä muodostetaan verkkosivusto.

Web-sovellus ei ole vain verkkosivusto. Web-sovellus pohjaa asiakas-palvelin -malliin, jossa web-selain toimii asiakaspäätteenä ja tarjoaa interaktiivisen palvelun ottamalla yhteyttä palvelimiin internetin välityksellä. Verkkosivusto yksinkertaisesti tarjoaa staattista sisältöä. Web-sovellus sen sijaan esittää dynaamisesti räätälöityä sisältöä käyttäjän syötteiden perusteella. (Shklar & Rosen 2003, 5.)

Web-sovellukset voivat olla todella suuria kokonaisuuksia, ja niiden kehittäminen voi vaatia syvää ymmärrystä ohjelmistojen suunnittelusta ja toteuttamisesta. Hyvä esimerkki erittäin suuresta web-sovelluksesta on Facebook. Facebookissa käyttäjä voi luoda julkaisuja ja selata muiden julkaisuja tai etsiä niitä hakutoiminnolla. Facebook näyttää jokaiselle käyttäjälle hieman erilaiselta, sillä se räätälöityy käyttäjän omien syötteiden mukaan.

Dynaamisten web-sovellusten kehittämisen mahdollistaa JavaScript-ohjelmointikieli. JavaScriptillä voidaan kirjoittaa verkkoselaimessa suoritettavaa ohjelmakoodia, joka kykenee noutamaan uutta dataa verkon yli lataamatta koko verkkosivua uudelleen ja tekemällä muutoksia suoraan verkkosivun rakenteeseen. Tällä tavoin voidaan rakentaa nopeita ja responsiivisia web-sovelluksia.

4.2 Webpack

Webpack on työkalu modernien JavaScript-sovellusten kehitykseen, jolla useita moduuleja voidaan niputtaa yhteen tiedostoon. Tämä mahdollistaa JavaScript-koodin jakamisen useisiin lähdekooditiedostoihin sekä moduuleihin ja samanaikaisesti myös tehokkaan tavan tarjoilla sovellus selaimelle yhdessä kompaktissa tiedostossa. Webpack kykenee käsittelemään monia eri tiedostotyyppejä, ja sillä on mahdollista niputtaa myös esimerkiksi tyylitiedostot yhdeksi tiedostoksi. Webpack tukee lisäosia, jotka muodostavatkin suuren osan Webpackin toiminnallisuudesta. Paljon käytettyjä lisäosia ovat koodin minimoimiseen ja optimointiin käytetyt lisäosat sekä tyylitiedostojen ajaminen CSS-esiprosessorin läpi. (Concepts n.d.)

4.3 Riippuvuuksienhallinta

Modernit JavaScript-sovellukset hyödyntävät paljon kolmannen osapuolen moduuleja. Näiden avulla säästetään kehitysaikaa käyttämällä valmiita toteutuksia sen sijaan, että kaikkea tarvitsisi tehdä itse alusta asti. Maailman suurin ohjelmistorekisteri npm sisältää yli 600 000 pakettia, joita ladataan viikossa yhteensä yli 3 miljardia kertaa. (What is npm? n.d.)

Projektissa näitä riippuvuuksia voidaan hallita siihen tarkoitetuilla ohjelmilla. Näitä ovat esimerkiksi npm ja Yarn. Paketinhallintaohjelma ylläpitää projektin sisällä package.json-tiedostoa, joka sisältää viittaukset riippuvuuksiin ja niiden versionumeroihin. Näiden ohjelmien ansiosta riippuvuuksia ei tarvitse ladata versionhallintaan, vaan jokainen kehittäjä saa ladattua ne omalle työkoneelleen helposti paketinhallintaohjelman avulla.

5 Sovelluskehykset ja -alustat

5.1 Yleistä

Ohjelmistojen kehittäminen alusta alkaen on työlästä, ja usein eri ohjelmistot sisältävät paljon yhtäläisiä ominaisuuksia. Niinpä ohjelmistojen kehittämisen helpottamiseen käytetään paljon sovelluskehyksiä.

Sovelluskehys on yleiskäyttöinen rakenne, joka jatkamalla voidaan rakentaa haluttuun tarkoitukseen käytettävä ohjelmisto. Sovelluskehys tarjoaa tukea yleisten ominaisuuksien toteuttamiseen, joita tarvitaan monissa sovelluksissa. Käyttöliittymien rakentamiseen luotu sovelluskehys yleensä tarjoaa tukea erilaisten käyttäjän aiheuttamien tapahtumien käsittelyyn sekä valmiita käyttöliittymäkomponentteja käyttöliittymän rakentamiseen. (Sommerville 2009, 431.)

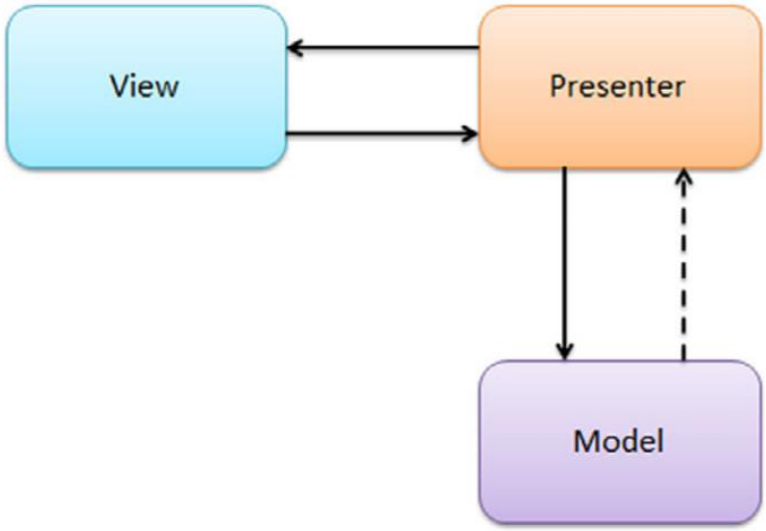
5.2 GWT

Google Web Toolkit eli GWT on työkalupakki web-sovellusten kehittämiseen, ja se on ollut yleisesti käytettävissä vuodesta 2006. GWT:n avulla voidaan web-sovelluksia kir-

joittaa Java-ohjelmointikielellä, jonka GWT:n kääntäjä kääntää selaimessa suoritettavaksi JavaScript-koodiksi. GWT sisältää erittäin paljon valmiita käyttöliittymäkomponentteja, RPC-tuen, lokalisaation, koodin minimoinnin sekä paljon muuta. (Tacy, Hanson, Essington & Tökke 2013, 4.)

GWT on Firstbeatilla yleisessä käytössä, ja Admin Clientin vanha versio on myös toteutettu GWT:lla. GWT:n valinta käyttöliittymän toteuttamiseen on toisaalta hyvä ajatus, sillä näin sekä palvelimen että käyttöliittymän ohjelmointiin voidaan käyttää Javaa. Java on edelleen TIOBE-indeksin mukaan suosituin ohjelmointikieli (TIOBE index n.d.), joten Javaa osaavia ohjelmistokehittäjiä on aina helposti löydettävissä.

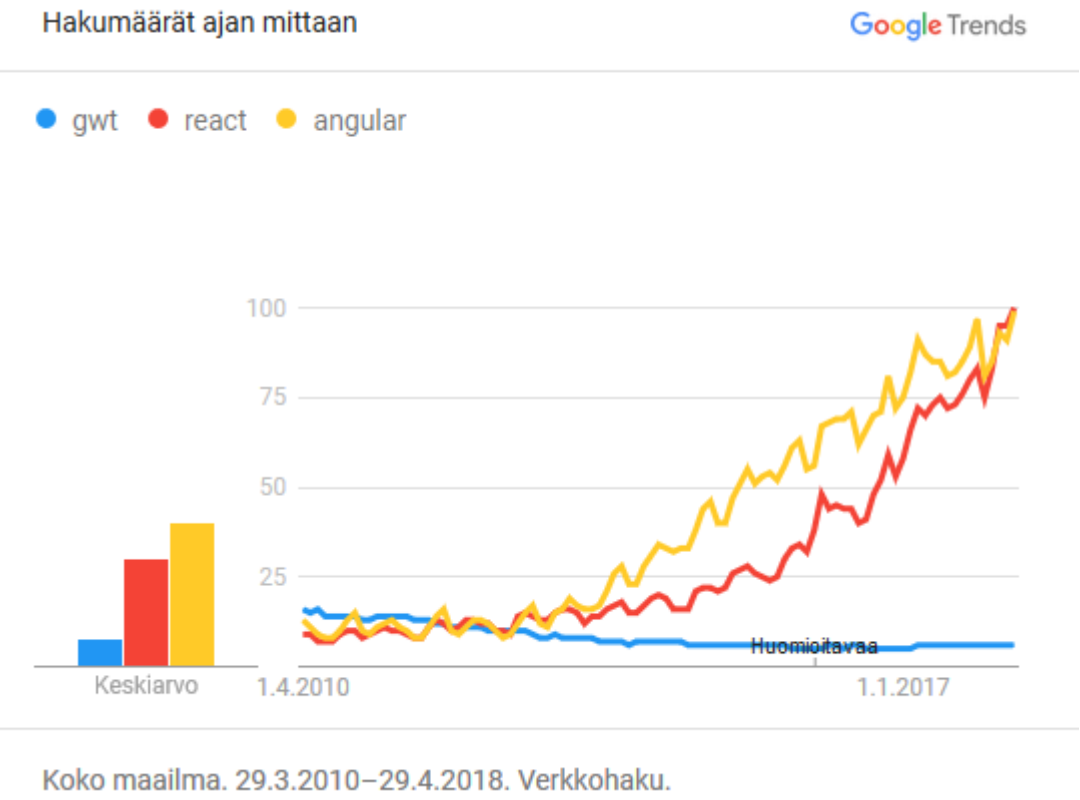
GWT:n kanssa käytetään usein MVP-arkkitehtuurimallia. MVP muodostuu sanoista Model-view-presenter. Model eli malli käärii jonkin sovelluksen sisäisen entiteetin kuten esimerkiksi asiakkaan. Asiakkaan tapauksessa malli voisi sisältää nimen sekä osoitteen. View eli näkymä sisältää käyttöliittymäkomponentit kuten listat, tekstikentät ja napit. Presenter eli esittäjä sisältää ohjelmalogiikan käyttäjän syötteiden käsittelyyn, näkymien välillä siirtymiseen sekä kutsujen tekemisestä palvelimille. Kuvio 1 havainnollistaa näiden osien välistä kommunikaatiota.



Kuvio 1. Esimerkki MVP-mallin toiminnasta (Tacy ym. 2013, 485)

Firstbeatin sisällä GWT ja MVP-arkkitehtuuri tuntuvat aiheuttavan turhan suurien lähdekooditiedostojen syntymistä. Tämä tekee lähdekoodista vaikeasti selattavan ja hidastaa sovelluksen ylläpitoa ja jatkokehitystä.

Kuviosta 2 on nähtävissä myös GWT:n suosion lasku hakumäärissä verrattuna Reactiin ja Angulariin. React ja Angular ovat hakumäärissä suosionsa huipulla. Korkea suosio takaa tuen saatavuuden ja kolmannen osapuolen kirjastojen määrän.

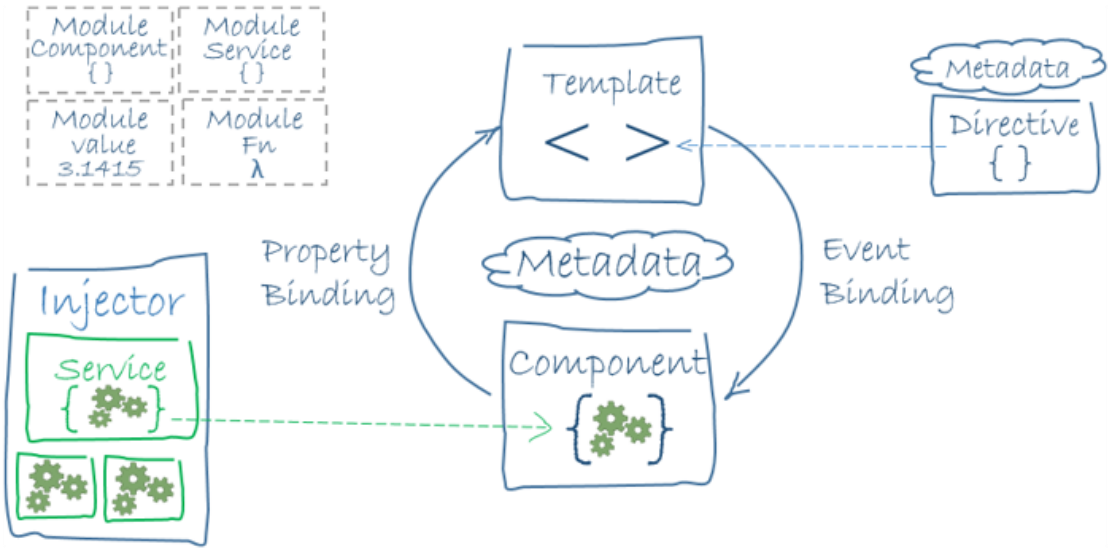


Kuvio 2. GWT:n, Reactin ja Angularin hakumäärät vuosina 2010 - 2018 (Google Trends 2018)

5.3 Angular

Angular on yksi tämän hetken suosituimpia alustoja web-sovellusten rakentamiseen. Angular on moderni alusta web-sovellusten kehittämiseen, ja sen mukana tulee kattava valikoima työkaluja suurten ja kestävien sovellusten kehittämiseen monille eri alustoille. Angularilla on mahdollista kehittää web-, työpöytä- sekä mobiilisovelluksia. Angularin kehityksessä on pyritty rakentamaan jotain suurempaa kuin tavanomainen sovelluskehys, ja siitä kuulee joskus puhuttavan ekosysteeminä. (Wilken 2018.)

Angular-sovelluksen rakentamiseen käytetään moduuleita, komponentteja, malleja, direktiivejä sekä palveluita. Kuviossa 3 on kuvattu näiden osien välisiä yhteyksiä.



Kuvio 3. Angularin osien väliset suhteet (Architecture overview n.d.)

Angular-sovellus muodostuu useista moduuleista. Moduulit kokoavat yhteenkuuluvaa koodia toimiviksi kokonaisuuksiksi. Tyypillisesti sovelluksella on aina juurimoduuli sekä useita ominaisuuksia määritteleviä moduuleja. Komponentit määrittelevät näkymiä, joita Angular näyttää ja muokkaa ohjelmalogiikan mukaisesti. Komponentit käyttävät palveluita, jotka suorittavat toiminnallisuutta kuten datan hakemista verkon ylitse tai käyttäjän syötteiden tarkistusta. (Architecture overview n.d.)

Angularin kanssa ohjelmointiin käytetään TypeScriptiä. TypeScript on JavaScriptiin pohjautuva ohjelmointikieli, joka tuo JavaScriptiin lisää ominaisuuksia kuten muuttujien tyypityksen. TypeScript käännetään tavalliseksi JavaScriptiksi, jotta sitä voidaan suorittaa selaimessa (TypeScript n.d.). Siirtyminen Angulariin tarkoittaisi siis myös TypeScriptin opettelua ohjelmistokehitystiimille.

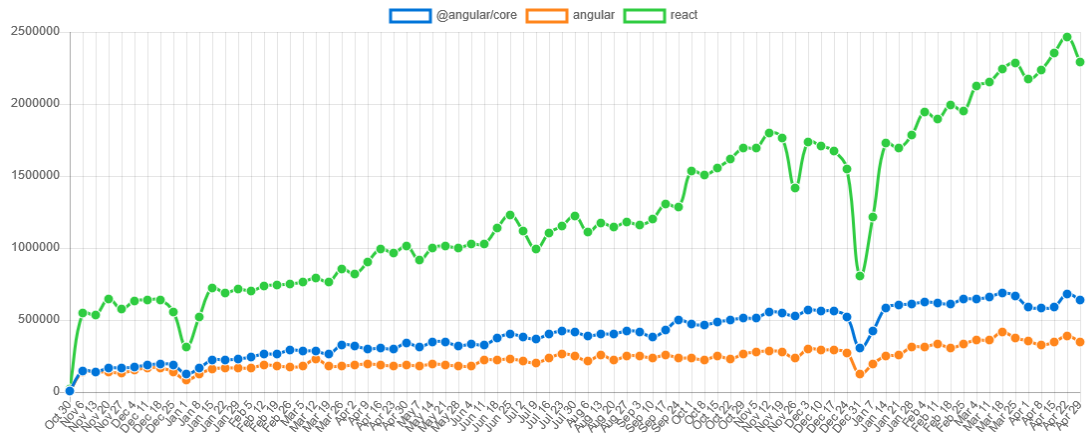
5.4 React

5.4.1 Yleistä

React on avoimen lähdekoodin JavaScript-kirjasto käyttöliittymien rakentamiseen eri alustoille. Reactia ei ole rakennettu tekemään yleisiä asioita mitä muilla sovelluskehyksillä ja alustoilla tehdään, eikä React tuo mukanaan lähellekään niin suurta määrää työkaluja ja kehitysmalleja kuin esimerkiksi Angular. React keskittyy vain sovelluksen näkymiin, eikä se ota kantaa esimerkiksi verkko-operaatioiden suorittamiseen,

lomakkeiden validointiin, navigaatioon tai lokalisaatioon. Näiden asioiden toteuttaminen on jätetty kehittäjien oman harkinnan varaan. (Thomas 2017, 6.)

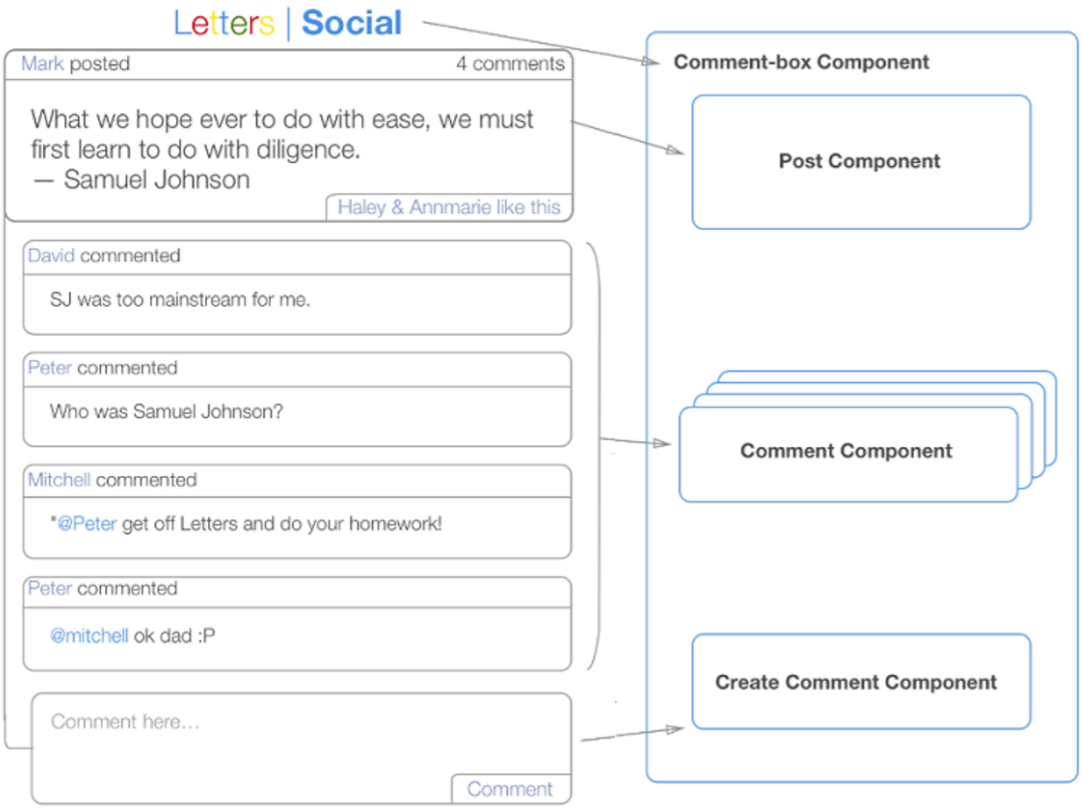
React valittiin käyttöliittymäkirjastoksi uuden Admin Clientin toteuttamiseen. React on npm-pakettien latausmäärissä suositumpi kuin Angular (ks. kuvio 4), mikä lupaa hyvää tulevaisuudelle Reactia hallitsevien kehittäjien määrässä ja tuen saatavuudessa. React on myös erittäin kevyt ottaa käyttöön, sillä se ei ole kokonainen ohjelmistokehys kuten Angular ja GWT. Keveys kuulostaa lupaavalta varsinkin, jos Firstbeatin muita GWT-sovelluksia halutaan vähitellen muuttaa React-sovelluksiksi esimerkiksi näkymä kerrallaan.



Kuvio 4. Reactin ja Angularin npm pakettien latausmäärät (NPM Trends 2018)

5.4.2 Komponentit

Reactin ajatusmalli perustuu komponentteihin, joiden avulla muodostetaan kokonainen sovellus. Komponentteja voi rakentaa moniin eri tarkoituksiin, mutta yleisesti komponentilla kuvataan jotain sovelluksen käyttöliittymän näkyvää osaa, jonka voi käsittää omaksi kokonaisuudekseen. Komponentti voi esimerkiksi olla navigaatiovalikko, lomake, päivämäärävalitsin, tekstikenttä tai nappi. Jakamalla käyttöliittymä eri komponentteihin voidaan helpommin kirjoittaa uudelleenkäytettävää ja vähemmän itseään toistavaa ohjelmakoodia. Kuviossa 5 nähdään, kuinka sovelluksen käyttöliittymä voidaan jakaa komponentteihin ja kuinka osaa komponenteista voidaan käyttää käyttöliittymässä useaan kertaan.



Kuvio 5. Esimerkkejä komponenteista (Thomas 2017, 15)

5.4.3 JSX

Käyttöliittymien kirjoittamiseen Reactilla käytetään tavallisesti JSX-syntaksia. JSX on XML-tyyppinen jatke JavaScriptiin. JSX on tarkoitettu käytettäväksi esiprosessorin kanssa, joka kääntää JSX:n tavalliseksi JavaScript-koodiksi. (Draft: JSX Specification n.d.). Näin käyttöliittymä voidaan kirjoittaa JavaScript-koodin mukaan käyttäen HTML:stä tuttuja elementtejä. Kuviossa 6 on havainnollistettu JSX:n käyttöä JavaScript-koodissa.

```
// Using JSX to express UI components.
var dropdown =
  <Dropdown>
    A dropdown list
    <Menu>
      <MenuItem>Do Something</MenuItem>
      <MenuItem>Do Something Fun!</MenuItem>
      <MenuItem>Do Something Else</MenuItem>
    </Menu>
  </Dropdown>;

render(dropdown);
```

Kuvio 6. Esimerkki JSX-syntaksista (Draft: JSX Specification n.d.)

5.4.4 Virtuaalinen DOM

DOM eli Domain Object Model on ohjelmointirajapinta HTML- ja XML-dokumenteille ja sen avulla dokumentti voidaan esittää tavalla, joka mahdollistaa ohjelmien pääsyn käsiksi sen rakenteeseen, tyyliin ja sisältöön. Verkkosivu on dokumentti, jota JavaScript voi DOM-rajapinnan avulla muokata eri tavoin. (Introduction to the DOM n.d.)

DOM elementtien päivittäminen ja muutosten esittäminen selaimessa käyttäjälle on kuitenkin jokseenkin hidasta, ja tämä vaikutus korostuu monimutkaisissa sovelluksissa. Suorituskyvyn parantamiseksi React ylläpitää verkkosivusta virtuaalista DOM-puuta. Muutokset DOM:iin rakenteeseen lasketaan ensin tähän virtuaaliseen puuhun, jonka jälkeen React päivittää oikeaan DOM:iin vain ne elementit, jotka ovat oikeasti muuttuneet. (Thomas 2017, 11-12.)

6 Toteutus

6.1 React-projektin luonti

Reactin käyttö tulee aloittaa Node.js:n sekä jonkin paketinhallintasovelluksen asentamisella. Tässä työssä pakettien hallintaan käytettiin Yarn-sovellusta.

Helpoin tapa aloittaa uusi React-sovellus on käyttää create-react-app -sovellusta (myöhemmin CRA). CRA:n voi asentaa Yarnia käyttäen komennolla:

```
yarn global add create-react-app
```

Projektin luonti onnistuu komennolla:

```
create-react-app projektin-nimi
```

CRA lataa tarvittavat tiedostot npm-rekisteristä sekä luo pienen esimerkkisovelluksen. Esimerkkisovelluksen voi käynnistää kehitystilassa komennolla:

```
yarn start
```

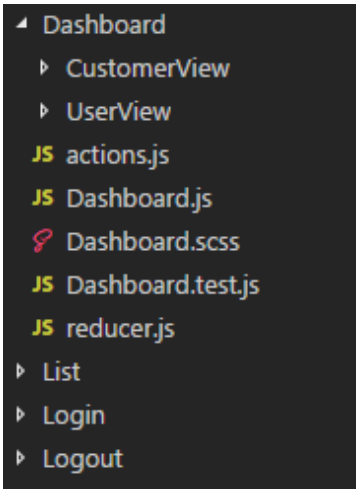
Tämän jälkeen esimerkkisovellus on nähtävissä verkkoselaimessa paikallisessa verkossa osoitteessa http://localhost:3000/. Kehitystilassa lähdekoodiin tehdyt muutokset päivittyvät tallennuksen jälkeen automaattisesti selaimessa näkyvään sovellukseen.

6.2 Projektin rakenne

Ylläpidettävyydestä pidettiin huolta heti projektia aloittaessa kiinnittämällä huomiota projektin lähdekoodin jäsentelyyn. Usein ensimmäinen ajatus on ryhmitellä lähdekoodi tyypin mukaan. Tyypin mukaan ryhmiteltynä komponentit voisi laittaa yhteen kansioon, tyylitiedostot toiseen ja yksikkötestit kolmanteen. Suuremmassa sovelluksessa lähdekoodi voi olla kuitenkin selkeämpi ryhmitellä ominaisuuksien tai komponenttien mukaan.

Tässä työssä lähdekoodit jaoteltiin ominaisuuksien mukaan. Näin kaikki yksittäiseen komponenttiin tai ominaisuuteen vaikuttava lähdekoodi on helposti löydettävissä.

Jokainen komponentti sijoitettiin omaan hakemistoonsa (ks. kuvio 7), joka sisältää itse komponentin lähdekoodin, komponentin tyylitiedoston, komponentin yksikkötestien lähdekoodit sekä mahdolliset tilanhallintaa liittyvät lähdekoodit.



Kuvio 7. Lähdekoodin jaottelu projektissa

6.3 Reititys

Toteutettu sovellus on niin sanottu Single Page Application. Sovellus sisältää vain yhden HTML-sivun, eikä navigointi sovelluksen sisällä aiheuta selaimessa sivun uudelleenlatausta. Uutta sisältöä ladataan palvelimelta tarpeen mukaan ja navigoinnin seurauksena HTML-dokumenttiin piirretään tarvittavat komponentit.

Reititys tai navigaatio sovelluksen sisällä toteutettiin hyödyntämällä react-router -moduulia. Moduuli sisältää toiminnallisuutta, jonka avulla sovellus voi päätellä selaimen osoiterivin URL-osoitteesta kulloinkin käyttäjälle esitettävän näkymän (ks. kuvio 8).

Näkymiä voi määritellä käyttämällä moduulin mukana tulevaa Route-komponenttia. Komponentti ottaa ominaisuuksina URL-osoitteen sekä kyseisessä osoitteessa esitettävän komponentin.

Route-komponenttia jatkettiin käärimällä se ProtectedRoute-komponenttiin. ProtectedRoute-komponentti päästää käyttäjän sille annettuun URL-osoitteeseen vain, jos käyttäjä on kirjautuneena. Muutoin käyttäjä ohjataan automaattisesti kirjautumisnäkymään.

```
<Switch>
  <ProtectedRoute exact path='/' component={Dashboard} />
  <ProtectedRoute path='/customer/:id?' component={CustomerView} />
  <ProtectedRoute path='/user/:id?' component={UserView} />
  <ProtectedRoute exact path='/administration' component={AdministrationView} />
  <ProtectedRoute exact path='/statistics' component={Statistics} />
  <Route exact path='/login' component={Login} />
  <Route exact path='/logout' component={Logout} />
  <Route render={() => <h1>404</h1>} />
</Switch>
```

Kuvio 8. Reititysten määrittely sovelluksessa

6.4 Autentikointi

Turvallisuussyistä käyttäjän kirjautumista ei muisteta vierailujen välillä. Kun käyttäjä navigoi sovellukseen, hänet ohjataan kirjautumisnäkymään. Kirjautuminen on toteutettu käyttäen HTTP Basic-autentikointia, jossa käyttäjätunnukset lähetetään palvelimelle Authorization-otsakkeessa base64-enkoodattuna. Turvallisuuden vahvistamiseksi kirjautuminen on toteutettu myös kaksivaiheisena.

Käyttäjä kirjautuu sovellukseen ensin käyttäjätunnuksella sekä salasanalla. Mikäli tämä on käyttäjän ensimmäinen kirjautuminen kyseisellä selaimella, hän saa tunnukselleen määritettyyn sähköpostiin kirjautumiskoodin, joka käyttäjän on syötettävä tunnustensa lisäksi. Onnistuneen kirjautumisen jälkeen käyttäjä ohjataan päänäkymään.

6.5 Tilanhallinta

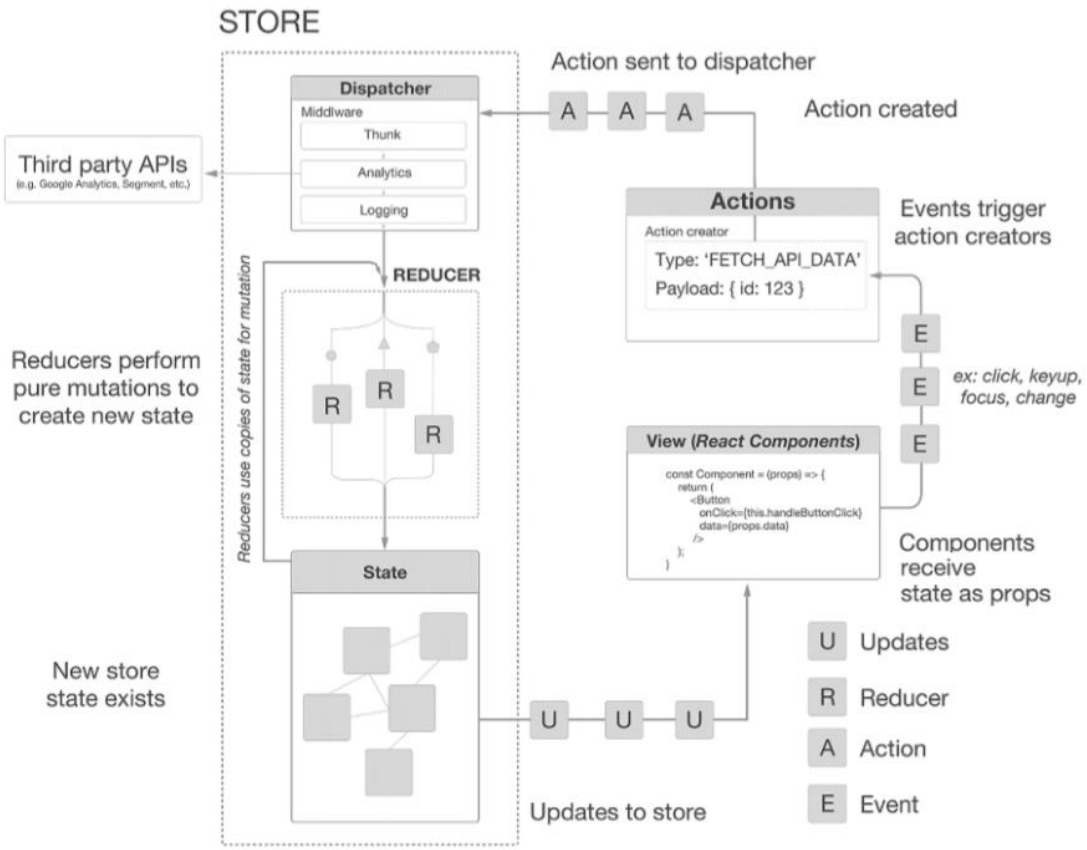
Monimutkaisissa sovelluksissa yksi suurista haasteista on tilanhallinta. Sovelluksen tila sisältää usein palvelimelta haettua dataa sekä käyttäjän syöttämää dataa, jota ei ole vielä tallennettu palvelimelle. Lisäksi käyttöliittymän eri komponentit voivat käsitellä ja esittää samaa dataa, jolloin siihen tehtyjen muutosten heijastaminen eri puolille sovellusta muodostuu ongelmaksi. (Motivation N.d.)

Sovelluksen tila vaikuttaa erityisesti käyttöliittymään. Nykyaikaiset JavaScript-sovellukset sisältävät usein monimutkaisia käyttöliittymäkomponentteja. Näitä käyttöliittymäkomponentteja halutaan joissain tilanteissa esittää eri tavalla, tai niitä ei välttä-

mättä haluta esittää ollenkaan. Esimerkiksi lomakkeen tallennuksen painikkeen toiminta voidaan haluta estää, jos käyttäjä ei ole vielä syöttänyt kaikkea välttämätöntä tietoa lomakkeeseen.

Erilaisia tilanhallintamenetelmiä kannattaa yleensä harkita vain monimutkaisiin sovelluksiin, tai vasta kun komponenttien sisäisen tilan käyttäminen alkaa tuntua hankalalta. Tilanhallintakirjastot pakottavat kehittäjät käyttämään tiettyä muottia, joka usein tekee monien yksinkertaisten asioiden kirjoittamisesta suuritöisempää.

Tässä työssä sovelluksen tilanhallintaan käytettiin apuna Redux-kirjastoa. Reduxin yleiskuvaus on esitetty kuviossa 9. Reduxia käytettäessä sovelluksen koko tila tallennetaan yhden olion sisälle. Kun koko sovelluksen tila on helposti nähtävillä yhdestä paikasta, on virheiden löytäminen kehityksen aikana helpompaa. Myös sovelluksen tilan tallentaminen palvelimelle ja sen palauttaminen muodostuu erittäin helpoksi. Myös tilaan tehtyjen muutoksien peruuttaminen on helpompi toteuttaa. (Three Principles n.d.)



Kuvio 9. Reduxin yleiskuvaus (Thomas 2017, 181)

Sovelluksen tilaa päivitetään laukaisemalla toimintoja. Toimintoja voi laukaista esimerkiksi painikkeet sovelluksen käyttöliittymässä. Toiminto on yksinkertainen JavaScript-olio, joka sisältää tiedon muutoksista. Toimintoja palautetaan usein *action creator*-funktioiden paluuarvona (ks. kuvio 10). Jokaisen sovelluksen tilaan vaikuttavan muutoksen kuvaaminen toimintojen avulla helpottaa suunnattomasti kokonaisuuden hallintaa. Kehityksen aikaiset ongelmatilanteet on helpompi selvittää, kun jokainen tilaan tehty muutos voidaan jäljittää sen laukaisseeseen pisteeseen. (Core concepts n.d.)

```
export function selectCustomer(customerId) {
  return {
    type: SELECT_CUSTOMER,
    customerId
  }
}
```

Kuvio 10. Action creator-funktio, joka palauttaa toiminnon

Tila ja toiminnot solmitaan yhteen *reducer*-funktioilla. Nämä funktiot ottavat argumentteina sovelluksen tämänhetkisen tilan sekä toiminnon ja palauttavat sovelluksen uuden tilan. Tilan päivitys voidaan jakaa useaan *reducer*-funktioon, jolloin kukin näistä käsittelee vain palaa koko sovelluksen tilasta. (Core concepts n.d.)

Kuviossa 11 esitelty *reducer*-funktio käsittelee asiakastilien joukkueita. Funktion argumenttina saama state-olio on osa koko sovelluksen tilaa. Tässä tapauksessa state-olio on kokoelma avain-arvo-yhdistelmiä, johon tallennetaan kunkin asiakastilin joukkueet. Avain on asiakastilin yksilöivä tunniste, ja arvo on olio, joka sisältää listan asiakastilin joukkueista sekä verkko-operaation tilan. Funktio päättelee toiminnon tyypin ja sen sisältämän datan mukaan, minkälaisen olion se palauttaa.

```
export function groupsByCustomerReducer(state = {}, action) {
  switch (action.type) {
    case FETCH_GROUPS:
      return {
        ...state,
        [action.customerId]: {
          ...state[action.customerId],
          isFetching: true
        }
      }

    case FETCH_GROUPS_SUCCESS:
      return  {
        ...state,
        [action.customerId]: {
          isFetching: false,
          items: action.groups
        }
      }

    case FETCH_GROUPS_FAIL:
      return {
        ...state,
        [action.customerId]: {
          isFetching: false,
          items: []
        }
      }

    default:
      return state
  }
}
```

Kuvio 11. Reducer-funktio

Palvelimen REST-resurssien kutsumiseen käytettiin JavaScriptin fetch-rajapintaa. Fetch mahdollistaa verkko-operaatioiden suorittamisen samaan tapaan kuin XMLHttpRequest, mutta fetch käyttää JavaScriptin Promiseja, jotka tekevät kutsujen kirjoittamisesti helpompaa ja selkeämpää.

REST-resursseja kutsuvissa funktioissa käytettiin myös JavaScriptiin ES2017-spesifikaatiossa tuotua asynkronisia funktioita. Tämä ominaisuus mahdollistaa asynkronisen logiikan kirjoittamisen samaan tapaan kuin synkronisen koodin. Sen sijaan että verkko-operaation vastaus käsiteltäisiin fetch-funktion palauttamassa Promise-

oliossa, vastaus voidaan tallentaa muuttujaan käyttämällä await-avainsanaa. Tällä tavalla kirjoitettuna verkko-operaatiot voidaan kirjoittaa todella selkeään muotoon (ks. Kuvio 12).

```
export async function getCustomers() {
  const url = `${server}/sports-service/admin/customers`
  const response = await fetch(url, { headers: getAuthHeader() })
  const data = await response.json()
  const customers = data.customers.customer || []
  return customers
}
```

Kuvio 12. Koodiesimerkki asiakastilien hakemisesta palvelimelta

Näin kirjoitettuna verkko-operaatioiden virheet voidaan käsitellä try-catch -lausekkeella erittäin selkeästi (ks. kuvio 13).

```
function fetchCustomers() {
  return async function (dispatch) {
    dispatch(requestCustomers())

    try {
      const customers = await getCustomers()
      dispatch(receiveCustomers(customers))
    } catch (e) {
      dispatch({ type: RECEIVE_CUSTOMERS_FAIL, error: e })
    }
  }
}
```

Kuvio 13. Koodiesimerkki verkko-operaation virheenkäsittelystä

6.6 Yksikkötestaus

Yksikkötestaamisella varmistutaan, että lähdekoodin yksittäiset osat toimivat odotetulla tavalla. Yksikkötestaamisessa lähdekoodi jaetaan pieniin testattaviin yksiköihin, kuten funktioihin. Yksikkötestaamisen suurin hyöty tulee esiin, kun sitä suoritetaan jatkuvasti kehityksen aikana. *Test Driven Development* on ohjelmistokehityksen menetelmä, jossa ennen varsinaisen ohjelmakoodin kirjoittamista kirjoitetaan sen yksikkötestit. Näin testit toimivat samalla suunnittelun dokumentaationa. (Unit Test Basics 2016.)

CRA:lla luotu React-projekti tukee yksikkötestien suorittamista ilman lisäkonfiguraatiota. Testit kirjoitetaan omiin tiedostoihin test.js-tiedostopäätteellä. Testit voidaan suorittaa komentoriviltä komennolla:

```
yarn test
```

Komennon käynnistämä testauskehys jää tarkkailemaan muutoksia lähdekoodissa. Muutosten tallentamisen jälkeen testit ajetaan automaattisesti uudelleen. Kuviossa 14 on esitelty *customersReducer*-funktiota testaava yksikkötesti. Kyseinen funktio tallentaa palvelimelta haettujen asiakastilien tiedot sovelluksen tilaan.

```
import { RECEIVE_CUSTOMERS } from './actions'
import { customersReducer } from './reducer'

it('adds customers to application state', () => {
  const state = {
    customers: {
      items: []
    }
  }

  const action = {
    type: RECEIVE_CUSTOMERS,
    customers: [ { name: 'Test customer' }, { name: 'Another customer'} ]
  }

  state.customers = customersReducer(state.customers, action)

  expect(state.customers.items.length).toEqual(2)
})
```

Kuvio 14. Esimerkki reducer-funktion yksikkötestistä

Opinnäytetyön aikana yksikkötestien kirjoittamista tehtiin vain vähän. Tavoitteena oli antaa vähintäänkin esimerkki React-sovelluksen yksikkötestaamisesta toimeksiantajalle ja muille Firstbeatin ohjelmistokehittäjille.

6.7 Käyttöliittymätoteutus

6.7.1 Tavoitteet

Käyttöliittymän suunnittelussa tähdättiin käytettävyyteen. Värimaailma rajoitettiin pääasiassa mustavalkoiseen käyttämällä Firstbeatin tunnusomaista punaista korostusvärinä. Tekstin ja taustan välinen kontrasti pidettiin riittävän suurena luettavuuden takaamiseksi. Kaikki tyylit toteutettiin itse, valmiita tyylikirjastoja ei käytetty.

Uuden Admin Clientin käyttöliittymän haluttiin toimivan tarpeen vaatiessa myös mobiililaitteilla. Käyttöliittymää lähdettiin kehittämään Mobile First -periaatteen mukaisesti. Käyttöliittymä rakennettiin ensin toimimaan ja näyttämään hyvältä pienellä näytöllä. Tämän jälkeen CSS:n Media Queryn avulla asetettiin näyttökoon raja-arvoja, joiden ylittyessä käyttöliittymän ulkoasua muutetaan hyödyntämään suurempaa näyttöpinta-alaa.

6.7.2 Asiakastilien ja käyttäjätunnusten selailunäkymä

Tilien selailunäkymässä käyttäjä voi selata sekä asiakastilejä että niiden alla olevia käyttäjätunnuksia. Käyttäjä voi seuloa asiakastilejä niiden nimen tai niille asetetun hallitsevan administraattorin mukaan. Käyttäjätunnuksia on mahdollista seuloa nimen tai käyttäjätunnuksen roolin mukaan. Molempien listojen hakuehdot voi nollata painamalla ”tyhjennä”-painiketta.

Avattaessa näkymä ensimmäisen kerran sovellus palvelimelta kaikki asiakastilit sekä kaikki administraattorit. Asiakastilit esitetään käyttöliittymän vasemman puoleisella listalla. Administraattorit haetaan, jotta asiakastilejä voidaan seuloa niitä hallitsevan administraattorin mukaan.

Kun käyttäjä valitsee asiakastilin vasemman puoleiselta listalta, sovellus tekee pyynnön palvelimelle hakeakseen valitun asiakastilin käyttäjätunnukset sekä joukkueet. Käyttäjätunnukset esitetään oikean puoleisella listalla. Käyttäjätunnukset ja joukkueet tallennetaan sovelluksen tilaan kyseisen asiakastilin alle. Jos käyttäjä myöhemmin valitsee saman asiakastilin uudelleen, ei käyttäjätietoja eikä joukkueita tarvitse hakea palvelimelta enää uudelleen.

Valittaessa käyttäjätunnus sovellus hakee palvelimelta valitun käyttäjätunnuksen profiilitiedot. Profiilitiedot ovat Sports-järjestelmässä käyttäjätunnuksista erillinen resurssi, jotka sisältävät suurimmaksi osaksi urheilijaa koskevia tietoja kuten pituuden, painon ja maksimisykkeen. Myös profiilitiedot tallentuvat sovelluksen tilaan, eikä niitä tarvitse hakea samaa käyttäjätunnusta valittaessa enää uudelleen.

6.7.3 Asiakastilin muokkausnäkymä

Asiakastilin muokkausnäkymä avautuu, kun käyttäjä on valinnut asiakastilin ja painanut ”katso”-painiketta tai ”luo uusi”-painiketta. Näkymän keskiössä on lomake, jonka tiedot on esitäytetty valitun asiakastilin tiedoilla. Vaihtoehtoisesti jos käyttäjä on luomassa uutta asiakastiliä, lomakkeen kentät ovat tyhjät uusien tietojen syöttämistä varten.

Olemassa oleva asiakastili voidaan poistaa ”poista”-painikkeella. Tämän jälkeen käyttäjän täytyy vielä vahvistaa operaatio ”kyllä” tai ”ei”-painikkeella.

Uutena ominaisuutena käyttöliittymään tuotiin asiakastilin statistiikan esittäminen. Kaikkien asiakastilin alle tehtyjen mittausten määrä, sekä niiden jakauma harjoitusmittausten ja pikapalautumistestien välillä on esitetty ympyrädiagrammissa. Myös asiakastilin käyttäjätunnusten määrä sekä niiden jakauma valmentajien ja urheilijoiden kesken on esitetty toisessa ympyrädiagrammissa. Kuvaajat toteutettiin käyttämällä chart.js-kirjastoa.

6.7.4 Käyttäjätunnusten muokkausnäkymä

Käyttäjätunnusten muokkausnäkymä on hyvin samankaltainen kuin asiakastilien muokkausnäkymä. Näkymän keskiössä on lomake sekä käyttäjätunnuksen että profiilitietojen syöttämiseen.

Käyttöliittymästä on mahdollista poistaa käyttäjätunnukset, lähettää käyttäjälle salasanan palautuslinkki, tuoda kaikki urheilijan mittaukset fbe-tiedostossa tai siirtää käyttäjätunnus jonkin toisen asiakastilin alaisuuteen.

Jos valittuna on urheilijan käyttäjätunnus, esittää käyttöliittymä kaikki urheilijan mittaukset sekä niiden jakauman harjoitusmittausten ja pikapalautumistestien välillä ympyrädiagrammissa.

6.7.5 Administraattorien selailunäkymä

Administraattorien selailunäkymässä käyttäjä voi selata, muokata sekä luoda uusia järjestelmän administraattoreita. Näitä käyttäjiä ei ole mahdollista poistaa, jotta Sports-järjestelmän lokit pysyvät eheinä.

Valittaessa administraattori listalta avautuu valitun käyttäjän tiedot lomakkeelle. Painamalla ”muokkaa”-painiketta lomakkeen kentät aktivoituvat tietojen syöttämistä varten. Uuden administraattorin tiedot voi syöttää painamalla ”luo uusi”-painiketta.

6.7.6 CSS-moduulit

Tyylien toteuttamisen apuna käytettiin CSS-moduuleita. CSS-moduuli on tyylitiedosto, jossa kaikki luokkien nimet ovat paikallisesti määriteltyjä. Tämä mahdollistaa sen, että komponentit voivat omissa tyylitiedostoissaan huoletta käyttää mitä tahansa nimiä CSS-luokille ilman pelkoa, että tyylit vaikuttaisivat jonkin toisen komponentit ulkoasuun. CSS-moduulit vähentävät tarvetta kirjoittaa pitkiä CSS-valitsimia, jolloin tyylitiedostot pysyvät siistimpinä ja helpommin luettavina.

CSS-moduulit kirjoitetaan samalla tavalla kuin tavanomaisetkin CSS-tiedostot. Käännösvaiheessa Webpack muuttaa CSS-luokat globaalisti uniikeiksi tunnisteiksi.

6.7.7 SASS

Tyylien kirjoittamisen avuksi otettiin SASS. SASS on CSS-kielen laajennus, joka helpottaa tyylien määrittelyä tuomalla kehittäjän käyttöön muuttujat, sisäkkäiset luokat ja perinnän sekä paljon muuta.

SASS vaatii toimiakseen CSS-esikääntäjän, joka kääntää SASS-syntaksin tavalliseksi verkkoselaimen ymmärtämäksi CSS-syntaksiksi. Tarvittavien riippuvuuksien asentamisen jälkeen SASSin käyttö ei tuo kehitykseen ylimääräisiä vaiheita. Tyylitiedoston tallentamisen jälkeen Webpack ajaa SASS-esikääntäjän automaattisesti kaikille tyylitiedostoille.

7 Tulokset

7.1 Vaatimusten täyttyminen

Opinnäytetyön aikana saatiin toteutettua lähes kaikki sille asetetut vaatimukset täyttävä sovellus. Osaa vaatimuksista ja ominaisuuksista ei saatu vietyä projektin aikana aivan loppuun asti, vaan ne vaativat vielä lisätyötä opinnäytetyön jälkeen.

Admin Clientin käyttötarkoituksen tärkeimmät ominaisuudet saatiin toteutettua. Tuotetulla sovelluksella on mahdollista luoda ja muokata asiakastilejä, asiakastilien käyttäjiä sekä järjestelmän ylläpitäjiä. Asiakastilejä ja käyttäjätunnuksia on helppo etsiä käyttöliittymän rajausominaisuuksilla.

Uutena ominaisuutena Admin Clientiin lisätty statistiikan esittäminen jäi aikataulurajoitteiden vuoksi tavoitteista. Palvelinpäähän ei ehditty toteuttaa rajapintaa, josta tarvittavat statistiikat haettaisiin. Käyttöliittymäpuoli saatiin kuitenkin toteutettua käyttämällä esimerkkidataa, joka generoidaan selaimessa. Ominaisuuden loppuun vieminen vaatisi todennäköisesti vain muutaman päivän työn.

Opinnäytetyön aikana vanhaan Admin Clientiin on lisätty uutta toiminnallisuutta, joka täytyisi myös lisätä uuteen toteutukseen. Näiden lisäämisellä tuotettu sovellus voisi toimia yrityksen ohjelmistokehityksen ja testauksen käytössä. Käyttöönotto myyjien ja tuotetuen käyttöön edellyttää puutteellisten ominaisuuksien loppuun viemistä ja kattavaa järjestelmätestausta kesän 2018 aikana.

7.2 Vanhan ja uuden toteutuksen vertailu

Vanhaan Admin Clientiin verrattuna uusi näyttää huomattavasti modernimmalta ja raikkaammalta. Uusi sovellus hyödyntää paremmin näytön leveyttä, eikä ole rajattu yhtä kapeaksi kuin vanha sovellus. Huomattavasti parempaa on sovelluksen käyttö mobiililaitteella. Vanha sovellus ei ottanut pieniä näyttöjä huomioon lainkaan. Uusi sovellus muokkautuu erinomaisesti pienille näytöille, eikä käyttäjän tarvitse selata sivua sivusuunnassa.

Uusi sovellus ei lukitse käyttöliittymää verkko-operaatioiden ajaksi. Vanha sovellus näyttää latausanimaatiota ponnahdusikkunassa koko verkko-operaation ajan. Tämä

estää sovelluksen muun käyttämisen ja on hermoja kiristävää. Uusi sovellus sen sijaan näyttää pienen latausindikaattorin sen komponentin päällä, joka odottaa palvelimen vastausta. Käyttäjä voi halutessaan navigoida sovelluksen eri näkymien välillä verkko-operaatioiden aikana.

Vanhassa sovelluksessa asiakastilejä ja käyttäjätunnuksia selattiin omissa näkymissään. Tällöin heti asiakastilin tai käyttäjätunnuksen valitessaan sovelluksen käyttäjä näki valitun asiakastilin tai käyttäjätunnuksen tarkat tiedot. Uudessa sovelluksessa yhtenä toiveena oli sijoittaa nämä listat samaan näkymään. Tämä nopeutti asiakastilin käyttäjän etsimistä ja muokkaamista, mutta toisaalta haittaa käyttäjätunnuksen löytämistä esimerkiksi sähköpostiosoitteen perusteella. Selailunäkymien hakutoimintoja tulisi monipuolistaa, jotta tämän muutoksen haittapuolet saadaan poistettua.

CSS-kielen laajennus SASS tuotiin uutena Admin Clientiin. SASS on jo käytössä Firstbeat Sports Cloud-tuotteen kehityksessä ja sen tuominen myös Admin Clientiin on hyvä lisä.

7.3 React Firstbeatin ohjelmistokehityksessä

Siirtyminen GWT:n käytöstä Reactiin on tänä päivänä erittäin kannatettava ajatus. GWT on ollut hyvä valinta laajojen web-sovellusten kehittämiseen aikana, jolloin JavaScriptin ominaisuudet eivät olleet yhtä kattavat kuin Java-ohjelmointikielen. Viime vuosien ECMAScript-standardit ovat kuitenkin tuoneet JavaScriptiin paljon kaivattuja ominaisuuksia, jotka tekevät JavaScriptistä erittäin tuottavan ohjelmointikielen.

Reactin käyttö tekee ohjelmistokehityksestä mukavampaa ja nopeampaa. Kehityksen ketteryys tulee ilmi jokaisen pienen muutoksen jälkeen. Automaattisen kääntämisen ja päivityksen johdosta lähdekooditiedoston tallentamisen jälkeen muutokset ovat nähtävissä selaimessa huomattavasti GWT:tä nopeammin. GWT ei suoraan tue sovelluksen automaattista uudelleenkääntämistä tiedostojen tallentamisen jälkeen, vaan käännösvaihe täytyy laukaista lataamalla sovellus uudelleen selaimesta.

React auttaa kirjoittamaan modulaarisempaa ja ylläpidettävämpää ohjelmakoodia kuin GWT. Komponenttiajattelumalli ajaa ohjelmistokehittäjää pilkkomaan ohjelmakoodin selkeisiin kokonaisuuksiin. GWT:n MVP-mallissa datan ja näkymien välisen

suhteen sitovat esittäjäluokat paisuvat helposti liian suuriksi ja monimutkaisiksi, joka tekee ohjelmakoodin lukemisesta hankalaa.

GWT ei ole myöskään erityisen hyvä alusta mobiilisovellusten kehittämiseen. GWT-sovellukset ovat käytettävissä mobiililaitteilla ainoastaan verkkoselaimen kautta tai niin sanottuina hybridisovelluksina. React-kehittäjien taidot ovat helposti hyödynnettävissä myös mobiilikehityksessä. React Native on ohjelmistokehys natiivien mobiilisovellusten kehittämiseen, jossa käytetään samoja periaatteita kuin Reactissa.

React on vain kirjasto käyttöliittymien kehittämiseen, toisin kuin Angular, joka on paljon toiminnallisuutta ja valmiita ratkaisuja sisältävä ohjelmistokehys. Reactin keveydestä on todennäköisesti hyötyä, mikäli olemassa olevaa GWT-sovellusta lähdettäisiin vähitellen muuttamaan React-sovellukseksi esimerkiksi näkymä kerrallaan.

Angularin mukana on käytännössä otettava käyttöön myös TypeScript, joka tuo JavaScriptiin uusia ominaisuuksia. TypeScript tuo mukanaan kehitystä helpottamaan muun muassa muuttujien tyypityksen, joka auttaa vähentämään virheiden syntymistä ohjelmakoodin kirjoitusvaiheessa. TypeScript on hyvä lisä kehitystyöhön, mutta sen käyttöön otossa kannattaa pohtia hidastaako se uuden alustan käytön aloittamista. Reactin käytön voi aloittaa JavaScriptin osaamisella, mutta TypeScript on myös kohtuullisen helppo ottaa käyttöön Reactiin myöhemmin, jos sen hyödyt nähdään tarpeellisina.

8 Pohdinta

Opinnäytetyön tuloksiin voi olla suurilta osin tyytyväinen. Vaikka tuotettu sovellus sisältääkin pieniä puutteita, täyttää se kuitenkin suurilta osin kaikki sille asetetut toiminnalliset vaatimukset. Muutaman päivän työllä toteutus voitaisiin ottaa vanha sovelluksen rinnalle ohjelmistokehityksen ja testauksen käyttöön. Jatkokehityksellä ja kattavalla testaamisella sovellus on mahdollista saada kesän aikana loppukäyttäjien käytettäväksi.

Sovelluksen ulkoasu jäi ehkä eniten odotuksista. Ulkoasun suunnitteluun olisi pitänyt käyttää enemmän aikaa ja hyödyntää enemmän sovellusta käyttävien mielipiteitä. Käytettävyydessä ja suorituskyvyssä otettiin kuitenkin askeleita oikeaan suuntaan.

Opinnäytetyö tuo jossain määrin tietoa eri käyttöliittymäteknologioista ja niiden tuomista hyödyistä. Etenkin Reactin ja GWT:n välisiä eroja on tuotu esille. Angularin osalta tiedonanto jää hieman pinnalliseksi, sillä Angularilla ei toteutettu opinnäytetyön aikana mitään vertailtavaa toteutusta.

Pinnan alta löytyy kuitenkin täysin uusi koodipohja moderneilla teknologioilla. Toteutuksessa tuotiin esille useita monimutkaisen ja laajan sovelluksen ylläpidettävyyden sekä suorituskyvyn kannalta tärkeitä käytäntöjä. Sovelluksen käyttöliittymän eri komponentit on jaettu loogisesti ja tarpeeksi selkeisiin kokonaisuuksiin. Redux-kirjaston käyttö sovelluksen tilanhallinnassa selkeyttää datan kulkua sovelluksen eri komponenttien välillä ja auttaa jäljittämään virheitä aiheuttavia tilanteita niiden alkupisteeseen. JavaScriptin uusimpia ominaisuuksia hyödyntämällä onnistuttiin monet tyypillisesti hankalalukuiset ominaisuudet kirjoittamaan erittäin tiiviisti ja ymmärrettävästi.

Toimivan sovelluksen lisäksi tavoitteena oli tehdä toimivia ja moderneja käytäntöjä esille tuova koodipohja, joka voisi toimia Reactiin tutustuvalle ohjelmistokehittäjälle hyvänä esimerkkinä. Uskon, että toteutus toimii tähän tarkoitukseen hyvin.

Lähteet

Architecture overview. N.d. Artikkeli Angularin internetsivuilla. Viitattu 23.4.2018. https://angular.io/guide/architecture

Concepts. N.d. Artikkeli Webpack:n internetsivuilla. Viitattu 23.4.2018. https://webpack.js.org/concepts/

Core concepts. N.d. Artikkeli reduxjs:n internetsivuilla. Viitattu 29.4.2018. https://redux.js.org/introduction/core-concepts

Draft: JSX Specification. N.d. JSX Spesifikaation luonnos. Viitattu 23.4.2018. https://facebook.github.io/jsx/

Google Trends. N.d. Google Trends -palvelun internetsivut. Viitattu 29.4.2018. https://trends.google.fi/trends/

Hyvinvoinnin ammattilaiset. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 23.4.2018. https://www.firstbeat.com/fi/tyo-ja-hyvinvointi/hyvinvoinnin-ammattilaiset/

Hyvinvointianalyysi. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 23.4.2018. https://www.firstbeat.com/fi/tyo-ja-hyvinvointi/hyvinvointianalyysi/

Introduction to the DOM. N.d. Artikkeli Mozilla Developer Network sivustolla. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Joukkueurheilu. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 23.4.2018. https://www.firstbeat.com/fi/huippu-urheilu/joukkueurheilu/

Middleware. N.d. Artikkeli Reduxin internetsivuilla. Viitattu 1.5.2018. https://redux.js.org/advanced/middleware

Motivation. N.d. Artikkeli Reduxin internetsivuilla. Viitattu 29.4.2018. https://redux.js.org/introduction/motivation

Shklar, L. & Rosen, R. 2003. Web Application Architecture: Principles, Protocols and Practices. John Wiley & Sons Ltd.

Sommerville, I. 2009. Software Engineering. Pearson Education.

Tacy, A., Hanson, R., Essington, J. & Tökke, A. 2013. GWT In Action. Second Edition. New York: Manning Publications.

Tarinamme. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 22.4.2018. https://www.firstbeat.com/fi/yritys/tarina/

Thomas, M. 2017. React In Action. Version 9. MEAP Edition. Manning Publications.

TIOBE index, N.d. Viitattu 22.4.2018. https://www.tiobe.com/tiobe-index/

TypeScript. N.d. TypeScript-kielen internetsivut. Viitattu 29.4.2018. https://www.typescriptlang.org/

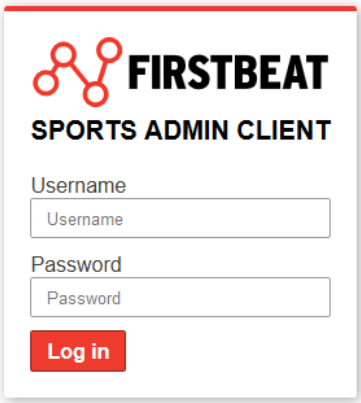
Unit Test Basics. 7.1.2016. Artikkeli Microsoft Developer Network sivuilla. https://msdn.microsoft.com/en-us/library/hh694602.aspx

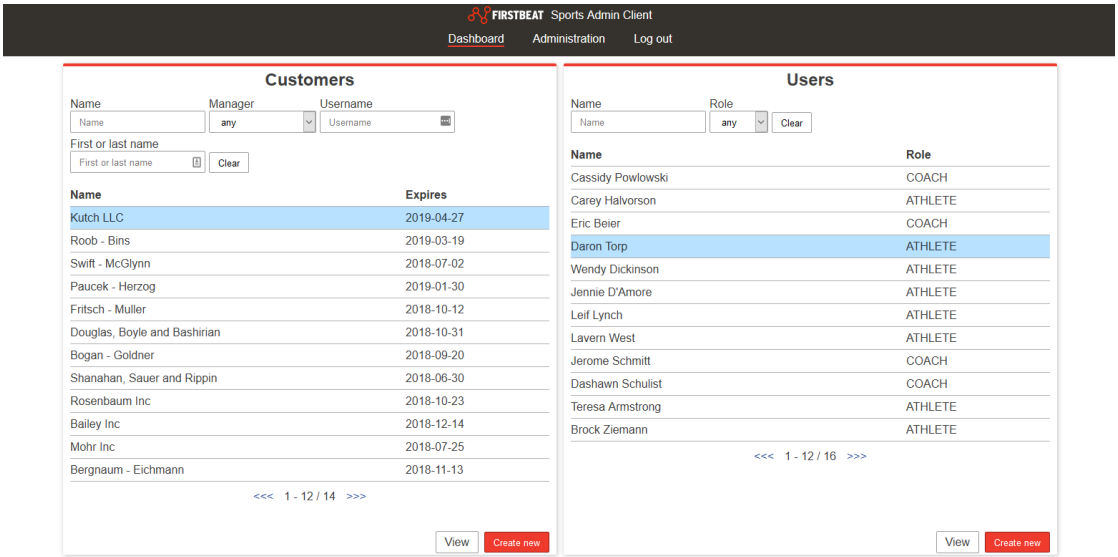
What is npm? N.d. Artikkeli npmjs:n internetsivuilla. Viitattu 23.4.2018. https://docs.npmjs.com/getting-started/what-is-npm

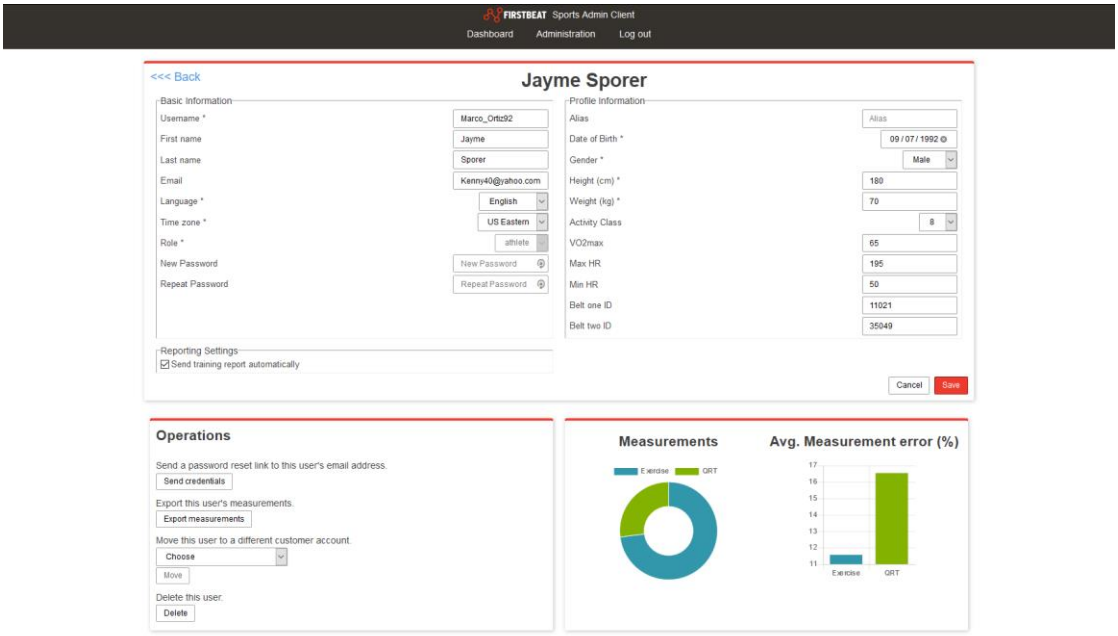
Wilken, J. 2018. Angular In Action. Manning Publications.

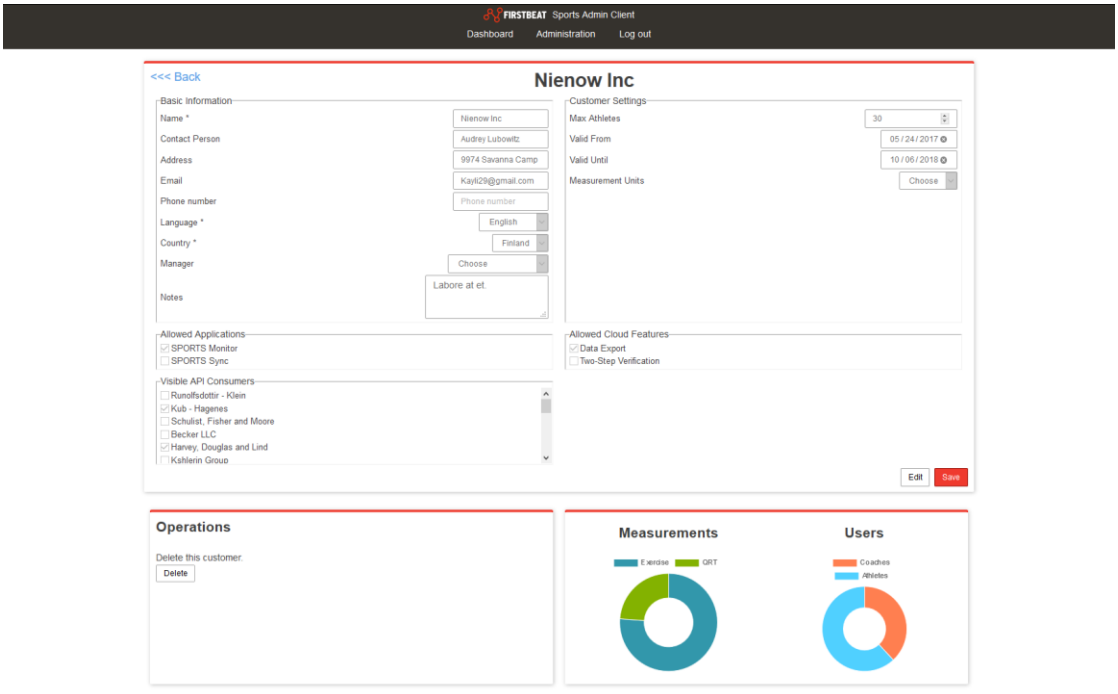
Liitteet

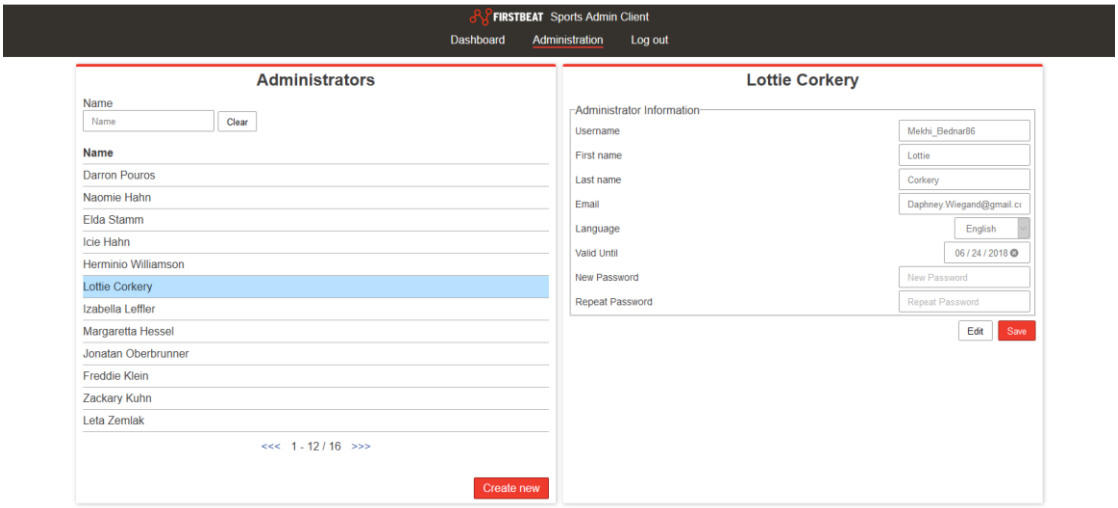
Liite 1. Kuvakaappauksia käyttöliittymästä






FIRSTBEAT Sports Admin Client
Dashboard
Administration
Log out
<<< Back
Jayme Sporer
Basic Information
Username *
Marco_Ortiz92
First name
Jayme
Last name
Sporer
Email
Kenny40@yahoo.com
Language *
English
Time zone *
US Eastern
Role *
athlete
New Password
Repeat Password
Profile Information
Alias
Date of Birth *
09 / 07 / 1992
Gender *
Male
Height (cm) *
180
Weight (kg) *
70
Activity Class
8
VO2max
65
Max HR
195
Min HR
50
Belt one ID
11021
Belt two ID
35049
Reporting Settings
Send training report automatically
Cancel
Save
Operations
Send a password reset link to this user's email address.
Send credentials
Export this user's measurements.
Export measurements
Move this user to a different customer account.
Choose
Move
Delete this user.
Delete
Measurements
Exercise
QRT
Avg. Measurement error (%)
17
16
15
14
13
12
11
Exercise
QRT


FIRSTBEAT Sports Admin Client
Dashboard
Administration
Log out
<<< Back
Nienow Inc
Basic Information
Name *
Nienow Inc
Contact Person
Audrey Lubowitz
Address
9974 Savanna Camp
Email
Kayli29@gmail.com
Phone number
Phone number
Language *
English
Country *
Finland
Manager
Choose
Notes
Labore at et
Customer Settings
Max Athletes
30
Valid From
05 / 24 / 2017
Valid Until
10 / 06 / 2018
Measurement Units
Choose
Allowed Applications
SPORTS Monitor
SPORTS Sync
Allowed Cloud Features
Data Export
Two-Step Verification
Visible API Consumers
Runolfsdottir - Klein
Kub - Hagenes
Schulist, Fisher and Moore
Becker LLC
Harvey, Douglas and Lind
Kshlerin Group
Edit
Save
Operations
Delete this customer.
Delete
Measurements
Exercise
QRT
Users
Coaches
Athletes


FIRSTBEAT Sports Admin Client
Dashboard
Administration
Log out
Administrators
Name
Name
Clear
Name
Darron Pouros
Naomie Hahn
Elda Stamm
Icie Hahn
Herminio Williamson
Lottie Corkery
Izabella Leffler
Margaretta Hessel
Jonatan Oberbrunner
Freddie Klein
Zackary Kuhn
Leta Zemlak
<<< 1 - 12 / 16 >>>
Create new
Lottie Corkery
Administrator Information
Username
Mekhi_Bednar86
First name
Lottie
Last name
Corkery
Email
Daphney.Wiegand@gmail.cc
Language
English
Valid Until
06 / 24 / 2018
New Password
New Password
Repeat Password
Repeat Password
Edit
Save