Tampereen ammattikorkeakoulu

Qlik Sense -raportointiportaali räätälöitynä toteutuksena

Maija Visala

OPINNÄYTETYÖ
Maaliskuu 2021

Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

VISALA MAIJA:
Qlik Sense -raportointiportaali räätälöitynä toteutuksena

Opinnäytetyö 30 sivua
Maaliskuu 2021

Tässä opinnäytetyöprojektissa rakennettiin asiakkaan toiveiden mukainen raportointiportaali web-pohjaisena sovelluksena. Tavoitteena oli luoda räätälöity ratkaisu, joka tarjoaisi valmiita raportointityökaluja joustavamman sivurakenteen ja joka näyttäisi kullekin käyttäjälle relevanttia tietoa kohdistetusti. Portaalin data haluttiin esittää graafeina, ja graafien luomiseen päätettiin hyödyntää visualisointityökalu *Qlik Senseä*. Graafit lisättiin portaaliin web-ohjelmoinnin keinoin upotuksina.

Opinnäytetyön tavoite oli selvittää, miten asiakkaalle voidaan tarjota mahdollisimman yksilöllisiä ratkaisuja raportoinnin tarpeisiin, erityisesti visualisointien osalta. Työn tarkoitus oli puolestaan esitellä malli, jolla tuotetaan asiakkaalle toimiva, räätälöity raportointiportaali. Opinnäytetyön toimeksiantaja oli Visma Consulting Oy, joka on ohjelmistoratkaisuja sekä IT-alan konsultointia tarjoava yritys.

Opinnäytetyöprojektin tuloksena tuotettiin raportointiportaali, johon upotettiin Qlik Sense -graafeja erityisesti Qlik Sense -upotuksiin tarkoitetuilla ratkaisuilla: ohjelmointikomponentti *Qdt-komponenteilla* sekä *Capability API* -rajapintojen avulla. Capability API -rajapinnat ovat Qlik Sensen omia Javascript-rajapintoja.
Muihin upotustapoihin verrattuna Qdt-komponentit sekä Capability API osoittautuivat hyviksi ratkaisuiksi tämän projektin tarpeisiin. Näillä upotustavoilla oli mahdollista toteuttaa sovelluksen joustava sivurakenne sekä käyttäjäkohtaisesti muokkautuvat graafit.

Projektin yhtenä tavoitteena oli selvittää, onko räätälöidyn portaalin puitteissa mahdollista soveltaa itsepalvelu-BI-ajattelumallia, jossa loppukäyttäjä voi itse muodostaa uusia graafeja ilman IT- tai BI-ammattilaisen apua. Projektin aikana ilmeni, että Qdt-komponenteissa voisi olla potentiaalia tällaisiin käyttötapauksiin. Itsepalvelu-BI-mallia ei lopulta otettu käyttöön tässä opinnäytetyöprojektissa, mutta tämä olisi mielenkiintoinen aihe jatkokehitystä ajatellen.

Asiasanat: Business intelligence, web-ohjelmointi, upotukset

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Production

VISALA, MAIJA
Qlik Sense Reporting Portal as a Custom-made Solution

Bachelor's thesis 30 pages
March 2021

This thesis describes a project, in which a custom-made reporting portal was made as a web-based application. The portal's visual content was produced with a visualization tool Qlik Sense and it was added to the application by embedding graphs through means of web programming. The goal was to create a custom-made solution, which would offer a more flexible side structure for the application and targeted content for each user, compared to a ready-made reporting portal.

The objective of this thesis was to find out how to provide a unique reporting solution to the customer, especially regarding visualization. Another goal was to present a model for creating a functioning custom-made reporting portal. The project was commissioned by the software company Visma Consulting Oy.

The result of the project was a functioning reporting portal, where Qlik Sense graphs were embedded into the solution by using Qlik Sense's own embedding solutions: Qdt Components and Capability API. Compared to other embedding options, these proved to be the most suitable.

Key words: Business intelligence, web programming, embedding

SISÄLLYS

1 JOHDANTO

Opinnäytetyöni asettuu liiketoimintatiedon hallinnan (BI) sekä web-ohjelmoinnin aihealueille. Työssäni kerron projektista, jossa rakennettiin asiakkaan toiveiden mukainen raportointiportaali web-pohjaisena sovelluksena. Portaalin keskeisimpänä sisältönä olivat visualisointityökalu Qlik Sensellä rakennetut graafit, jotka lisättiin sovellukseen upotuksina. Opinnäytetyöni toimeksiantaja on Visma Consulting Oy, joka on ohjelmistoratkaisuja sekä IT-alan konsultointia tarjoava yritys. Sekä räätälöidyt ohjelmistoratkaisut että tiedolla johtaminen kuuluvat yrityksen palveluihin.

Olen jakanut tämän opinnäytetyöraportin kolmeen suurempaan aihekokonaisuuteen. Ensimmäinen aihealue on teoreettisin, ja se käsittelee business intelligenceä eli liiketoimintatiedon hallintaa. Opinnäytetyöprojektilla on erittäin vahva pohja business intelligencessä, sillä koko projektin tavoite tähtää parempaan liiketoimintatiedon hallintaan. Tässä opinnäytetyössä business intelligence -teorian tavoitteena on selventää projektin taustaa – sitä, miksi kuvatun kaltaiselle raportointiportaalille on tarvetta, ja miten se voi hyödyttää loppukäyttäjän liiketoimintaa.

Visualisoinnit kuuluvat vahvasti business intelligencen aihealueelle, sillä liiketoimintadatan esittäminen visuaalisessa muodossa edistää datan hallintaa ja sitä kautta onnistunutta päätöksentekoa. Olen sisällyttänyt visualisointeihin liittyvän teorian sekä Qlik Sense -visualisointityökaluun liittyvän teorian business intelligence -kokonaisuuden alle.

Opinnäytetyöraportin toinen asiakokonaisuus liittyy varsinaiseen projektityöhön, räätälöityyn raportointiportaaliin. Kerron sovelluksen rakenteesta, teknologioista sekä haasteista. Kuvailen lisäksi, miten Qlik Sense -sovelluksen integroiminen web-sovellukseen toteutettiin.

Kolmas asiakokonaisuus liittyy ensisijaisesti web-ohjelmointiin. Käsittelen yleisellä tasolla muutamia web-kehityksen ilmiöitä ja käsitteitä, mutta keskeisimmän osion tässä kokonaisuudessa muodostavat Qlik Sense -graafien upottamiseen

tarkoitetut ohjelmointikomponentit, Qdt-komponentit. Kerron sekä Qdt-komponenttien peruskäytöstä että niiden soveltamismahdollisuuksista. Kuvailen lisäksi opinnäytetyöprojektissa esiin tulleita haasteita sekä niiden ratkaisuja.

2 OPINNÄYTTEEN LÄHTÖKOHDAT

Tässä luvussa kuvailen niitä lähtökohtia, jotka johtivat opinnäytetyössä kuvattavan projektin alkamiseen. Käsittelen myös tämän opinnäytetyön lähtökohtia ja esittelen työn tavoitteen ja tarkoituksen.

2.1 Projektin kuvaus

Koko prosessin lähtöpisteenä voisi pitää toimeksiantajayritykseni liiketoiminnasta kertynyttä dataa, jota haluttiin analysoida ja jalostaa erilaisiksi hyödyllisiksi kaavioiksi. Liiketoiminnan kannalta hyödyllinen data oli jakautunut eri lähteisiin, ja toiveena oli kehittää portaali, josta olisi helposti löydettävissä tarpeellinen data erilaisten liiketoiminnan päätösten tueksi. Tavoitteena oli, että kukin käyttäjä voisi löytää juuri itseään hyödyttävän datan mahdollisimman helposti.

Mikä siis on se prosessi, jonka avulla mahdollisesti sekavakin massa raakadataa muokataan helposti ymmärrettäviksi kaavioiksi ja kuviksi? Seuraavassa esittelen tiivistetysti, mitä osa-alueita raportointiportaali sisältää ja mitä osioita tässä opinnäytetyössä käsitellään.

Raportointiportaalin rakenne

Rakentamamme raportointiportaali on melko monimutkainen kokonaisuus. Kuten web-sovelluksissa yleensäkin, myös tämä sovellus jakautui kahteen osa-alueeseen: siihen mitä loppukäyttäjälle näytetään (frontend) ja siihen, mitä ”julkisivun” takana tapahtuu (backend).

Tämän kaltaisessa raportointiportaalissa on tärkeää, että logiikka pidetään erillään siitä, mitä käyttäjä näkee. Toisaalta käyttäjän pitää myös voida varmistua siitä, että graafeissa nähtävä data on aitoa ja täsmällistä. Datalähteiden on siis oltava ”läpinäkyviä”. Lopputuotteen käyttäjän täytyy voida jäljittää alkuperäinen data ja varmistaa datan oikeellisuus. Toisaalta on myös tärkeää, että kaiken taustalla toimiva logiikka pysyy erillään siitä, mitä käyttäjä näkee ja tekee. Data ei saa

”vaarantua”, sillä muuten alkuperäinen tavoite datan oikeellisuudesta ei enää päde. (Sherif 2016, Introduction to Practical Business Intelligence.)

Rakentamassamme raportointiportaalissa data haetaan kolmesta eri lähteestä. Nämä lähteet ovat asiakkaan omia liiketoimintatietoa käsitteleviä ohjelmistoja, jotka sisältävät esimerkiksi henkilöstöön ja myyntiin liittyvää dataa. Datan käsittelyn logiikka perustuu siihen, että näistä kolmesta lähteestä haettu data tallennetaan halutussa muodossa ja halutussa laajuudessa omaan tietokantaansa. Lopuksi data kopioidaan tietokannasta visualisointityökaluun, joka luo datasta havainnollisia graafeja.

Datan saattaminen alkuperäisistä lähteistä tietokantaan on kokonaan oma prosessinsa, joka noudattelee ETL-menetelmän (extract, transform, load) sekä tietovarastomallin periaatteita. Nämä prosessit olivat erittäin keskeinen osa projektia ja sovelluksen toimintaa. Tässä työssä ei kuitenkaan käsitellä näitä prosesseja, vaan keskitytään siihen, mitä lopputuotteen käyttäjä näkee. Työn keskiössä ovat siis nimenomaan visualisoinnit sekä web-ohjelmointi.

Teknologioita

Web-ohjelmoinnin osalta projektin kenties tärkein teknologia oli Javascript-kielen kirjasto React, jolla ohjelmoitiin loppukäyttäjälle näkyvää sisältöä eli frontendia. React linkittyi myös visualisointeihin, sillä graafit integroitiin web-sovellukseen juuri Reactin avulla. Visualisoinnit itsessään rakennettiin visualisointityökalu Qlik Sensellä, joka on yleisesti käytetty BI-työkalu raporttien luomiseen. Kerron Qlik Sensestä tarkemmin luvussa 3.3.2.

Muita projektissa käytettyjä teknologioita olivat muun muassa palvelinpuolen ohjelmointiin tarkoitettu ohjelmointikehys Spring Boot sekä ETL-työkalu Talend Open Studio. Projektissa oli käytössä MariaDB-tietokanta.

2.2 Opinnäytteen tavoite ja tarkoitus

Tässä opinnäytetyöraportissa tarkastelen, täyttääkö rakentamamme portaali asiakkaan esittämät alkuperäiset toiveet. Tavoitteeni on myös peilata portaalia valitsemaani teoriaa vasten. Haluan selvittää, noudatteleeko portaali teorialähteiden antamaa kuvaa hyvistä tiedolla johtamisen, visualisointien ja web-ohjelmoinnin käytänteistä. Opinnäytetyön tavoite on selvittää, millä keinoin asiakkaalle voidaan tarjota mahdollisimman yksilöllisiä ratkaisuja raportoinnin tarpeisiin, erityisesti visualisointien osalta. Tämän tavoitteen saavuttamiseksi kuvailen rakentamamme raportointiportaalin kehitystyössä esiin tulleita ratkaisuja. Toisin sanoen opinnäytetyöni tarkoitus on esitellä malli, jolla tuotetaan asiakkaalle toimiva, räätälöity raportointiportaali.

Tässä opinnäytetyöprojektissa pyrin löytämään uusia hyviä käytäntöjä ja selvittämään, millaisia uusia ratkaisuja toimeksiantajayritys voi lisätä palvelutarjontaan. Opinnäytetyö voi olla hyödyllinen sekä selvitys- että perehdytystarkoituksessa. Työssä selvitetään hyviä käytäntöjä raportointiportaalin rakentamiseen ja dokumentoidaan niitä erityisesti toimeksiantajaorganisaation myöhempää käyttöä varten.

3 RAPORTOINTIPORTAALI: BUSINESS INTELLIGENCE

Kuten sanottua, raportointiportaali nojaa vahvasti liiketoimintatiedon hallintaan eli business intelligenceen. Tässä luvussa käsittelen business intelligencen käsitettä ja raportointiportaalin BI-taustaa. Visualisoinnit ovat yksi BI:n osa-alueista, joten olen sisällyttänyt sekä visualisointeihin että Qlik Senseen liittyvän teorian tähän lukuun.

3.1 Business intelligence

Termille *business intelligence* löytyy varmasti monia erilaisia määritelmiä. Ahmed Sherif (2016) antaa kirjassa *Practical Business Intelligence* seuraavanlaisen määritelmän: business intelligence on bisnesympäristössä tapahtuvaa toimintaa, jossa tuotetaan päätöksiä analyyttisen datankäsittelyn ja datan esittämisen keinoin. (Sherif 2016, Introduction to Practical Business Intelligence.)

Business intelligence liittyy läheisesti analytiikan käsitteeseen. Analytiikka käsittää monenlaista datan hyödyntämistä: tilastojen tarkastelua ja määrällistä analysointia, ennustavien mallien rakentamista sekä päätöksiä edistävien aineistojen luontia. Analytiikka on osa business intelligenceä. Business intelligencestä käytetään toisinaan suomenkielistä nimitystä liiketoimintatiedon hallinta. Kyse on siis niin sanotusta älykkäästä liikkeenjohdosta, jossa teknologiat ja prosessit valjastetaan datan käsittelyyn. Tätä kautta saadaan parempi ymmärrys liiketoiminnasta. (Davenport & Harris 2007, 26.)

Business intelligence käsittää analytiikan lisäksi varsinaisen tiedon keräämisen ja raportoinnin. Tiedonkeruun ja raportoinnin tasolla voidaan kysyä esimerkiksi kysymykset ”*mitä tapahtui?*” tai ”*mikä on ongelma?*”. Kun siirrytään tiedonkeruusta analytiikan puolelle, kysymyksiksi muodostuvat esimerkiksi ”*miksi näin tapahtui?*” ja ”*mitä tapahtuu seuraavaksi?*”. Mitä paremmin kerättyä tietoa hyödynnetään, sitä suurempi on yrityksen saama kilpailuetu. (Davenport & Harris 2007, 26 – 27.)

3.2 Datan merkitys

Business intelligence ei ole tämän opinnäytetyön tärkein osa-alue. On kuitenkin hyvä ymmärtää, mikä tarkoitus raportointiportaalin rakentamisen taustalla on. Portaalia ei luonnollisesti rakennettu vain ”huvin vuoksi” tai siksi, että joku voisi katsella, millaisia graafeja yrityksen datasta voi saada aikaiseksi. Perimmäinen tarkoitus on tuottaa liiketoiminnan kasvua ja kilpailuetua.

Thomas Davenportin (2007) kirja *Analysoi ja voita* on muodostunut eräänlaiseksi BI-kirjallisuuden klassikoksi. Kirja esittelee syitä, miksi datan merkitys on niin suuri nykypäivän liike-elämässä, ja miksi analytiikka tarjoaa erinomaisen kilpailuedun. Davenportin mukaan lähes kaikenlainen liiketoiminta voi hyötyä analytiikasta. Perinteiset kilpailuedut, kuten esimerkiksi maantieteellinen sijainti, ovat menettäneet merkityksensä. Myöskään tuoteinnovaatiot tai uudet teknologiat eivät takaa kilpailuetua, sillä uudet keksinnöt päätyvät pikavauhtia kopioinnin kohteeksi. Se, mikä tänä päivänä erottaa menestyvän yrityksen kilpailijoista, on kyky tehdä hyviä päätöksiä ja tehostaa liiketoimintaa. (Davenport & Harris 2007, 28.)

Hyvien päätösten taustalla on yleensä huolellinen tiedon kerääminen ja sen analysointi. Davenport kutsuu analyysiä systemaattisesti hyödyntäviä yrityksiä analyyttisiksi kilpailijoiksi. Analyyttisella kilpailijalla on useita keinoja tehostaa liiketoimintaansa ja saavuttaa kilpailuetua. Davenportin (2007) mukaan analytiikka voi auttaa yritystä tunnistamaan parhaat asiakkaat ja asettamaan tuotteensa hinnan ihanteelliselle tasolle. Jos yritys myy kulutustavaraa, varastonhallinta ja toimitusketjun suunnittelu analytiikan avulla kehittävät yrityksen toimintaa. Yhtä lailla analytiikka voi tuoda etuja henkilöstöhallinnossa. Analyyttinen kilpailija pyrkii nimittäin palkkaamaan parhaat henkilöt, kehittämään heidän taitojaan ja pitämään heidät tyytyväisinä. Viimeisenä esimerkkinä Davenport mainitsee yritysfuusiot. Onnistuneen yrityskaupan taustalla on yleensä tarkkaa analysointityötä. (Davenport & Harris 2007, 28 – 29.)

On tärkeää muistaa, että analytiikan tarkoitus on johtaa myös toimintaan. Analyyttinen kilpailija tekee harkittuja päätöksiä ja ottaa niistä kaiken hyödyn. Kilpai-

luetu syntyy siitä, että yritys valitsee muista erottavan ominaisuuden (tai ominaisuuksia) omassa liiketoiminnassaan ja optimoi tämän ominaisuuden analytiikan avulla. (Davenport & Harris 2007, 28 – 29.)

Sherif korostaa, että business intelligence ei ole sidoksissa mihinkään yksittäiseen työkaluun, vaan BI:tä voidaan tuottaa monilla eri työkaluilla – jopa sellaisilla, joita ei alun perin ole suunniteltu BI-käyttöön. Ahmed Sherif huomauttaa, että BI-työkalujen tärkein tehtävä on tuoda data loppukäyttäjän tarkasteltavaksi. Varsinaisen bisneslogiikan tulisi tapahtua tietovaraston (datawarehouse) puolella. (Sherif 2016, Introduction to Practical Business Intelligence.)

3.3 Visualisoinnit

”Analyysi on arvokasta vain, jos se johtaa toimintaan”, toteaa Thomas Davenport kirjassaan Analysoi ja voita. Davenportin mukaan on tärkeää, että ihmiset voivat jakaa näkemyksiään esimerkiksi portaalien ja raportointityökalujen avulla. (Davenport 2007, 216.) Pelkkä raakadata ei siis useimmiten pysty viestimään tarkastelijalle mitään kovin olennaista. Dataa voidaan käyttää päätöksentekoon silloin, kun siitä voidaan poimia samanlaisena toistuvia kuvioita. Ja se, joka päätöksen tekee, haluaa luonnollisesti nähdä jotakin todisteita näistä toistuvista kuvioista. Visualisoinnit palvelevat tätä tarkoitusta.

Visualisoinnit ovat käytännössä graafeja ja kaavioita, jotka esittävät dataa visuaalisessa muodossa. Useimmille ihmisille on helpompaa tarkastella erilaisia visuaalisia kaavioita kuin lukea puhtaasti numeerista dataa. Visualisointien paljastamat mallit ja trendit antavat arvokasta ainesta analyyseihin ja päätöksentekoon. (Sahay 2017, 3.)

Kirjassa *Fundamentals of Data Visualization* Claus Wilke (2019) pohtii visualisointien kahta näkökulmaa: tieteellistä tarkkuutta sekä miellyttävää, informatiivista ulkoasua. On ensiarvoisen tärkeää, että graafi esittää lähdedatan oikein. Dataa ei saa esittää sellaisessa muodossa, että se johdattaa katsojaansa harhaan. Toisaalta myös visuaalisuudella on oma merkityksensä. Wilken mukaan datan harkittu esitysmuoto voi korostaa graafin viestiä. Huonot värivalinnat ja

epätasapainoiset elementit voivat puolestaan häiritä katsojan tulkintaa. (Wilke 2019, Introduction.)

Hyvän visualisoinnin rakentaminen vaatii jonkin verran suunnittelua: mikä graafityyppi esittää datan parhaalla tavalla? Miten muotoilla graafi niin, ettei se johda katsojaa harhaan? Mitä värejä, kuvioita ja muita esteettisiä elementtejä graafi sisältää?

Wilken mukaan visualisointi voi olla monella tavalla huono tai jopa virheellinen. Visualisointi voi olla ulkoasultaan epämiellyttävä, vaikka sen esittämä data olisikin sinänsä virheetöntä. Toisaalta visualisointi voi olla ulkoisesti miellyttävä, mutta sen asettelussa on ongelmia ja sen välittämä viesti on joko epäselvä, harhaanjohtava tai tarpeettoman monimutkainen. Pahimmassa tapauksessa graafi on yksiselitteisesti virheellinen. Sen välittämä viesti on objektiivisesti katsoen tai matemaattisesti vääristynyt. (Wilke 2019, Introduction.)

Hyvää graafia on vaikeaa määritellä yksiselitteisesti. Graafia voisi kuitenkin pitää ainakin hyväksyttävänä, jos se ei ole yllä mainituilla tavoilla huono. Hyvän graafin ominaispiirteinä voisi pitää ainakin informatiivisuutta ja miellyttävää ulkoasua. (Wilke 2019, Introduction.)

Kun kyseessä on valmis visualisointityökalu – tässä tapauksessa Qlik Sense – suuri osa graafien visuaalisista elementeistä on jo valmiiksi hyvien visualisointikäytäntöjen mukaisia. Esimerkiksi graafin mitta-asteikko generoituu automaattisesti, jolloin sen oikeellisuus ei vääristy. Qlik Sense tarjoaa myös automaattisesti neutraalin värityksen sekä soveltuvat fontit. Käyttäjän on toki mahdollista tehdä muutoksia mainittuihin ominaisuuksiin, mutta tässä opinnäytetyössä kuvatussa projektissa muutoksia ei tehty.

Vaikka graafien visuaaliset ominaisuudet olisivatkin valmiiksi annettuja, jotkin kehittäjän tekemät päätökset voivat vaikuttaa graafin ulkoasuun ja ymmärrettävyyteen. Tällaisia päätöksiä ovat esimerkiksi se, mitkä tiedot asetetaan x-akselille ja mitkä y-akselille sekä se, mitkä otsikot ja nimikkeet graafille annetaan. Tietysti oma vaikutuksensa on myös sillä, miten graafit asetellaan verkkosivulle, ja minkä kokoisina ne esitetään – suuremmat graafit vaikuttavat tärkeämmiltä kuin pienet.

3.3.1 Raportointiportaalin graafeja

Jo portaaliprojektin alusta lähtien kehitystiimillä oli käytössään asiakkaan laatima suunnitelma siitä, mitä graafeja raportointiportaali sisältäisi, ja mitä dataa ne esittäisivät. Kehitystiimin työnkuvaan ei siis kuulunut graafien suunnittelu, vaan ainoastaan niiden toteutus. Asiakkaan suunnitelman mukaan käytettäviä graafityyppejä olivat pylväskaaviot ja taulukot. Lisäksi portaali sisältäisi liiketoiminnan keskeisiä avainlukuja.

Tässä projektissa visualisointien suunnittelu ei vaatinut kehitystiimiltä suurta panostusta kahdesta syystä: Ensinnäkin asiakas oli suunnitellut graafien ulkoasua ja sisältöä jo ennen kehitystyön alkua. Toiseksi valmiin visualisointityökalun käyttö vähentää virheellisten tai huonojen graafien mahdollisuutta. Mikäli visualisoinnit tehtäisiin alusta alkaen ”käsin”, ilman erillisen työkalun apua, graafien oikeellisuuteen ja visuaaliseen ilmeeseen olisi panostettava huomattavasti enemmän resursseja.

Kirjassa *Fundamentals of Data Visualization* (Wilke 2019) annetaan ohjeita hyviin visualisointikäytänteisiin. Osa ohjeista on sovellettavissa myös tähän projektiin. Kirjassa painotetaan yhtenä keskeisimpänä visuaalisena sääntönä, että graafeissa tulisi olla tarpeeksi suuret otsikot ja selitetekstit. Usein graafit näytetään loppukäyttäjälle melko pienikokoisina, ja kun suurella ruudulla rakennettu graafi skaalataan pienemmäksi, otsikot ovatkin usein aivan liian pieniä. Useissa visualisointityökaluissa fonttikoko on asetettu oletusarvoisesti liian pieneksi. (Wilke 2019, Use Larger Axis Labels.)

Toisena suosituksena voisi pitää myös sitä, että visualisointeihin ei kannata lisätä ylimääräistä visuaalista ”hälyä”. Kaikenlaiset kehykset tekstikenttien tai koko graafin ympärillä ovat useimmiten turhia. Myös graafin taustalla olevat mitta-asteikkoviivat ja -ruudukot voivat olla häiritseviä, jos ne ovat liian tummia tai niitä on muuten liikaa. (Wilke 2019, Balance the Data and the Context.)

Raportointiportaalin tapauksessa kullekin sovelluksen sivulle sijoitettiin useita graafeja, ja graafien ympärille laitettiin kehykset erottamaan ne visuaalisesti toisistaan. Parempi vaihtoehto voisi kuitenkin olla vaihtaa graafin tausta valkoisesta johonkin toiseen neutraaliin sävyyn, joka toimisi erottavana tekijänä.

Suosituksia annetaan muutoinkin värien käytöstä. Graafeja ei pidä täyttää väreillä vain pelkän visuaalisen mielenkiinnon takia. Jos graafissa on käytetty eri värejä, niillä on oltava jokin tarkoitus. Värit eivät saa olla myöskään niin vahvoja, että ne vievät huomiota datasta ja vaikeuttavat graafin lukemista. (Wilke 2019, Common Pitfalls of Color Use.) Raportointiportaalin graafeissa käytettiin kautta linjan neutraalia sinisen sävyä, joka ei häiritse katsojan silmää.

3.3.2 Qlik Sense

Business intelligence on viime vuosien aikana painottunut entistä enemmän siihen suuntaan, että kuka tahansa voi luoda datalähtöisiä raportteja. Aiemmin raporttien luominen oli kokonaan BI-alan ammattilaisten heiniä, mutta niin sanotun self-service BI:n, eli itsepalvelu-BI:n myötä kuka tahansa voi käyttää data-analytiikan työkaluja ja luoda raportteja. Qlik Sense kehitettiin juuri tätä tarkoitusta varten.

Self-service BI:n myötä painopiste siirtyi yrityksissä vahvasti IT-puolelta bisnespuolelle. Kirjassa *Mastering Qlik Sense*, Martin Mahler ja Juan Ignacio Vitanto (2018) pitävät tätä painopisteen muutosta luontevana, sillä juuri bisnes on se yritystoiminnan osa, jota analytiikka palvelee. Mahlerin ja Vitanton mukaan Qlik Sense mukailee self-service BI:n keskeisimpiä tavoitteita: datan selkeää esitystapaa, lähdedatan helppoa käsittelyä, BI-työkalujen helppokäyttöisyyttä sekä ympäristöjen ja arkkitehtuurin helppoa hallittavuutta. (Mahler& Vitanto 2018, The self-service model.) Self-service BI:n idea on siinä, että nämä toiminnot ovat BI-asiantuntijoiden itsensä käytettävissä, ilman että väliin tarvitaan IT-asiantuntijaa.

Siinä missä Qlik Sense ohjaa business intelligenceä kohti self-service-mallia, tässä opinnäytetyössä kuvattava projekti vie käyttäjiä jälleen hieman kauemmas

tästä mallista. Syy tähän on se, että projektin tavoite, *räätälöity* portaali, toteutettiin web-ohjelmoinnin keinoin upotuksilla. Lähdekoodiin tehtävät upotukset vaativat ohjelmointiosaamista, ja portaalin muokkaaminen ei onnistu ilman IT-osaamista. Tämä seikka oli asiakkaan huolenaiheena jo kehitystyön aikana, ja ongelmaan oli suunnitteilla ratkaisuja myöhempiä kehitysvaiheita varten. Käsittelen Qlik Sense -graafien upottamista tarkemmin kappaleessa 4.4.

Huomionarvoista on kuitenkin, että Qlik Sense palvelee self-service-ajattelutapaa myös paljon perustavanlaatuisemmalla tavalla. Datan visuaalinen esitystapa on jo itsessään seikka, joka tuo datan lähemmäs ”tavallista” käyttäjää. Kuka tahansa voi hyötyä data-analytiikasta silloin, kun data ei ole pelkkiä yksiköitä tai lukuja, vaan helposti ymmärrettäviä kaavioita. Drew Robb (2012) kuvailee artikkelissaan *Visualization Broadens Business Intelligence's Appeal* visualisointien kasvavaa suosiota. Hän jopa nimittää IT- ja BI-asiantuntijoita ”pullonkaulaksi”, joiden väliintuloa voidaan visualisoinneilla välttää. Robbin mukaan visualisoinnit auttavat käyttäjiä tunnistamaan trendejä, ongelmia, kuvioita, yhteyksiä ja poikkeuksia. Datan käsittely jätetään työkalun huoleksi.

4 RAPORTOINTIPORTAALIN RAKENTAMINEN

Tässä luvussa kuvailen tarkemmin rakennetun raportointiportaalin rakennetta ja syitä räätälöidyn portaalin rakentamiselle. Esittelen myös tiivistetysti Qlik Sensen perustoimintoja ja erilaisia integrointimahdollisuuksia.

4.1 Miksi räätälöity ratkaisu?

Qlik Sense on jo itsessään raportointiportaali. Sovelluksessa voi varsin joustavasti luoda haluamansa raportin, ja muokkausmahdollisuuksia on lukuisia. Auktorisoidut käyttäjät pääsevät katselemaan raportteja omalta päätelaitteeltaan nettiselaimen välityksellä.

Miksi siis luoda erillinen web-sovellus, kun raportteja pääsisi katsomaan myös valmiissa Qlik Sense -sovelluksessa? Ensimmäinen syy oli asiakkaan toive sovelluksen rakenteesta, johon valmis Qlik Sense -portaali ei kyennyt vastaamaan. Tämän tarpeen lisäksi projektilla oli vahva tutkiva ja selvittävä perusta. Asiakkaan toiveena oli saada käyttöönsä toimiva sovellus, mutta yhtä lailla tavoitteena oli löytää uusia hyviä toimintatapoja ja selvittää, millaisia palveluita toimeksiantajayritys voisi lisätä palvelutarjontaansa. Tavoite oli siis yhtäältä luoda toimiva tuote, toisaalta kartoittaa tulevaisuuden mahdollisuuksia raportoinnin alueella.

4.2 Portaalin rakenne

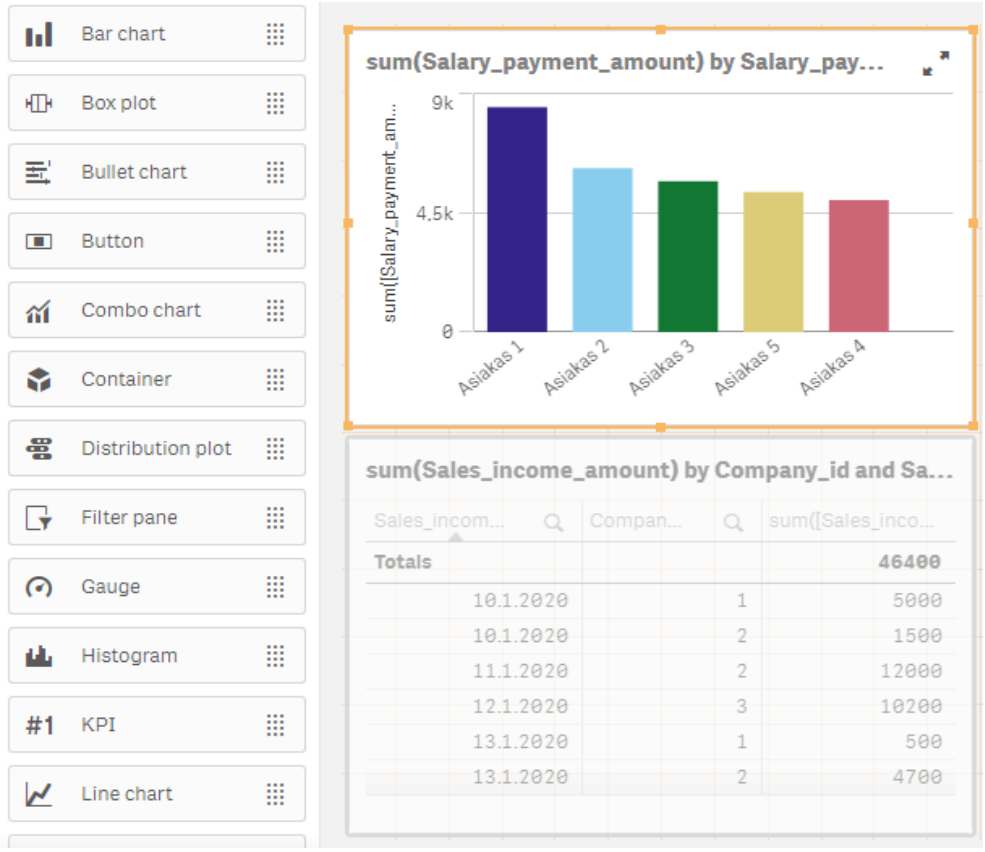
Kuten sanottua, keskeinen syy räätälöidyn portaalin rakentamiselle oli asiakkaan toive spesifisestä sivurakenteesta. Sovelluksen sisäisen navigoinnin haluttiin olevan ”kerroksellinen”. Asiakkaan toiveen mukaan käyttäjä kirjautuisi sisään portaaliin ja näkisi juuri ne tiedot, jotka ovat hänelle itselleen relevantteja. Sivuilla olevia graafeja napauttamalla käyttäjä pääsisi yhä syvemmälle kohti yksityiskohtaisempaa tietoa. Yhtenä esimerkkinä voisi mainita esimerkiksi seuraavanlaisen polun, jossa jonkin tiimin esimies etenisi sivulta toiselle:
koko yrityksen data > yksikön data > tietyn asiakkaan data > asiakkaan projektin data.

4.3 Graafeja Qlik Sense Enterprise -sovelluksella

Käyttäjän on mahdollista valita sopivin kolmesta eri Qlik Sense -versiosta: Qlik Sense Desktop, Cloud ja Enterprise. Qlik Sense Desktop sopii yksittäiselle käyttäjälle, ja se ladataan paikallisesti käyttäjän omalle tietokoneelle. Desktop-versiosta on saatavilla ilmainen kokeilukuukausi, ja myös tässä projektissa ensimmäiset Qlik Sense -testigraafit tehtiin paikallisesti desktop-versiolla. Qlik Sensen pilviversio, Qlik Sense Cloud on web-pohjainen versio Qlik Sensestä. Erillistä asennusta ei tarvita, ja versio sopii hieman suuremmalle käyttäjämäärälle kuin desktop-versio.

Tässä projektissa käyttöön valittiin kuitenkin Qlik Sense Enterprise, joka tarjoaa mainituista kolmesta versiosta laajimmat mahdollisuudet. Qlik Sense Enterprise asennetaan erilliselle palvelimelle, ja sitä voidaan hallinnoida suoraan palvelimella tai sisäverkossa nettiselaimella. Enterprise-versioon on ostettava maksullisia käyttäjälisenssejä.

Kaikissa kolmessa versiossa graafien luominen tapahtuu samaan tapaan. Käyttöliittymä on eri versioissa samankaltainen ja melko intuitiivinen. Uusien graafien luominen tapahtuu yksinkertaisesti raahaamalla graafipohjia visualisointinäkymään. Kuva 1 näyttää osan Qlik Sensen hallintapaneelista. Vasemmalla näkyy lista valittavista graafityypeistä, oikealla kaksi esimerkkidataa esittävää graafia.



KUVA 1. Osa Qlik Sensen hallintapaneelista

Tässä työssä ei perehdytä tarkemmin Qlik Sensen peruskäyttöön, sillä painotus on graafien upottamisessa sekä web-ohjelmoinnissa. Hyvän pohjan Qlik Sensen käytöstä voi saada esimerkiksi Qlik Sensen omista dokumentaatioista (Qlik Help 2020a).

4.4 Integrointimahdollisuuksia

Qlik Sense tarjoaa muutamia erilaisia keinoja graafien upottamiseksi ulkoisille web-sivuille. Kenties yksinkertaisin keino on yksittäisen graafin upottaminen koodiin iframe-komponentin avulla. Qlik Sense käyttää tästä tekniikasta nimeä *single integration*. Iframe on eräs HTML-kielen elementti, jonka avulla voidaan upottaa jokin verkkosivu kokonaan toiselle sivulle (Mozilla Developer Network 2021b). Käytännössä tämä tarkoittaa sitä, että käyttäjä voi selata jotakin verkkosivua toisen sivun sisällä, ilman että selain ohjautuu sivulta toiselle.

Iframen tapauksessa upotettava elementti on kuitenkin vain ”ikkuna” siihen, mitä jollakin toisella sivulla tapahtuu, eikä upotettava elementti ole suorassa yhteydessä sivuun, jolle se on upotettu. Tässä projektissa asiakkaan toiveena oli luoda vahvempi integraatio upotuksien ja varsinaisen web-sivun välille, ja siksi single integration -tekniikka hylättiin jo aivan projektin alussa.

Toinen yleinen integrointitekniikka on valmiin mashup-sivun luominen. Qlik Sense Desktop- sekä Enterprise-versiot tarjoavat Dev Hub -työkalun, jonka avulla käyttäjä voi koota Qlik Sense -graafeista valmiin verkkosivun. Mashup-sivun luominen ei välttämättä vaadi koodaustaitoa, sillä käyttäjä voi raahata valmiiseen pohjaan haluamansa graafit. Dev Hub generoi käyttäjälle valmiin paketin, johon kuuluvat HTML- ja CSS-sivut, Javascript-tiedosto sekä tarvittavat konfiguraatiot. Ratkaisu ei kuitenkaan ollut tarpeeksi joustava tämän projektin tarpeisiin. Lisäksi teknologiaksi oli valikoitunut Javascriptin React-kirjasto, ja projektiin haluttiin nimenomaan React-kirjaston kanssa yhteensopiva ratkaisu.

Näiden kriteerien pohjalta upotusratkaisuksi valikoitui koodillisesti monimutkaisempi ratkaisu, Qlik Sensen omien ohjelmointirajapintojen (API) käyttö. *Capability API* -rajapinnat ovat Qlik Sensen omia Javascript-rajapintoja, joiden avulla graafeja voidaan upottaa web-sivulle. Niiden käyttö on kuitenkin suhteellisen monimutkaista, joten projektissa päädyttiin käyttämään Qlik Sense -graafien upottamiseen tarkoitettuja frontend-koodikomponentteja, niin kutsuttuja *Qdt-komponentteja*. Qdt-komponentit hyödyntävät Capability API -rajapintoja, mutta tekevät niiden käytöstä yksinkertaisempaa.

4.4.1 Qdt-komponentit

Qdt-komponentit ovat suhteellisen uusi upotusratkaisu, sillä ne julkaistiin ensimmäisen kerran vuonna 2018. Kehittäjien mukaan tarkoituksena oli auttaa käyttäjiä hyödyntämään Qlik API -rajapintoja ja kehittämään mielenkiintoisia mashup-verkkosivuja. Qdt-komponentit eivät ole riippuvaisia mistään tietystä kirjastosta. (Qlik Design Blog 2020.)

Qdt-komponentit vaativat frontend-ohjelmointiosaamista, mutta niiden käyttö ei normaalitapauksessa ole monimutkaista. Kehitystyössä haasteita toi jonkin verran juuri se seikka, että Qdt-komponentit ovat melko uusi ratkaisu. Ongelmatilanteisiin oli haastavaa löytää vastauksia, sillä virallisia dokumentaatioita oli suppeasti tarjolla, ja käyttäjäfoorumeillakin ongelmista oli ehditty keskustella niukasti. Qdt-komponenteista on tarjolla mallikoodit, jotka toimivat useimmissa tapauksissa ongelmitta.

Rakentamamme raportointiportaalin tapauksessa haasteeksi nousi se seikka, että tekemämme verkkosovellus ja Qlik Sense -ohjelmisto sijaitsivat eri palvelimilla. Tämän takia sovelluskoodissa oli huomioitava Qlik Sense -sovellukseen autentikoituminen, ennen kuin graafeja voitiin näyttää. Kerron tarkemmin näistä erityishaasteista kappaleessa 5.3.

5 RAPORTOINTIPORTAALI: WEB-OHJELMOINTI

Tässä luvussa syvennyn projektin käytännönläheisempään osioon, varsinaiseen ohjelmointityöhön. Kerroin edellisessä luvussa, kuinka Qlik Sense -upotusten toteuttamiseksi valittiin valmis React-komponentti, Qdt Components. Tässä luvussa kerron lyhyesti web-ohjelmoinnista sekä perehdyn tarkemmin Qdt-komponentteihin ja niiden keskeiseen toimintaperiaatteeseen, API-rajapintoihin. Kerron myös tarkemmin Qdt-komponenttien toiminnasta ja niiden sovellusmahdollisuuksista.

Rakentamamme raportointiportaali toteutettiin verkkosovelluksena, joten graafien rakentamisen lisäksi perinteinen web-ohjelmointi oli osa projektin työnkuvaa. Sovelluksen sivurakenne ja navigointi sekä graafien ulkopuolelle jäävä muu ulkoasu rakennettiin web-ohjelmoinnin keinoin. Web-ohjelmoinnin piiriin asettuu myös graafien upottaminen, eli mainittujen Qdt-komponenttien käyttö.

5.1 Web-ohjelmointi

Verkkosovellusten kenties tärkein ominaispiirre on client-server-yhteys, eli palvelimien ja internet-yhteydellä varustettujen laitteiden yhteistoiminta. Asiakaslaite lähettää palvelimelle pyynnön, ja palvelin vastaa lähettämällä jonkin resurssin, esimerkiksi verkkosivun. Asiakaslaitteet ja palvelimet käyttävät http-protokollaa eräänlaisena ”kielenä”, jonka avulla pyyntöjä ja vastauksia lähetetään.

Raportointiportaalin tapauksessa asiakas-palvelin-yhteyksiä oli kaksi, sillä projektiympäristöön kuuluvat sovelluspalvelin, jossa varsinaisen sovelluksen koodi sijaitsee, sekä Qlik-palvelin, joka ”hostaa” Qlik-sovellusta ja rakennettuja Qlik-graafeja. Toisin sanoen sovelluspalvelin vastaa pyyntöihin, jotka pyytävät sovelluskoodia, Qlik-palvelin pyyntöihin, jotka pyytävät graafeja ja muita Qlik-resursseja.

Sovelluskoodi kirjoitettiin Javascript-kielellä, ja ohjelmoinnin apuna käytettiin React-kirjastoa. React on tarkoitettu nimenomaan käyttöliittymien rakentamiseen, eli juuri siihen, mitä käyttäjä näkee avatessaan sovelluksen. Reactia voi

pitää rakenteellisesti melko yksinkertaisena kirjastona. Peruslogiikka toimii niin, että koodi sisältää React-komponentteja, loogisia koodikokonaisuuksia, jotka vastaanottavat dataa ja huolehtivat siitä, että haluttu sisältö saatetaan näkyville web-sovellukseen. (Boduch ym. 2020, Why React.)

5.2 Qdt-komponenttien peruskäyttö

Mainitut React-komponentit ovat erittäin keskeinen osa React-ohjelmointia, ja tämä näkyy myös tavassa, jolla Qdt-komponentit lisätään React-pohjaiseen sovellukseen. Sovelluskoodiin luodaan uusi komponentti, joka hyödyntää valmista qdt-components-pakkausta (package). Pakkaus ladataan osaksi koodiprojektia ajamalla komentorivillä komento *npm install --save qdt-components*. Seuraava mallikoodi näyttä, millainen koodikomponentti projektiin luodaan. Konfiguraatioon (config) lisätään Qlik-palvelimen osoite sekä Qlik-projektin tunnus (appId).

```
import React, {useEffect, useRef} from 'react';
import PropTypes from 'prop-types';
import QdtComponents from 'qdt-components';


const options = {
 config: {
  host: "yourdomain.com",
  secure: true,
  port: 443,
  prefix: "",
  appId: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
 },
 connections: {
  vizApi: true,
  engineApi: true,
 },
};

const qdtComponent = new QdtComponents(options.config, options.connections);

function QdtComponent(props) {

 const node = useRef(null);
 const { type, props: qProps } = props;

 useEffect(() => {
  qdtComponent.render(type, qProps, node.current);
 }, []);

 return (
  <div ref={node} />
 );
}

QdtComponent.propTypes = {
 type: PropTypes.string.isRequired,
 props: PropTypes.object.isRequired,
};

export default QdtComponent;
```

KUVA 2. Qdt-koodikomponentti

Komponentti lisätään verkkosivulle haluttuun kohtaan seuraavalla tavalla:

```
<QdtComponent type={viz.type} props={viz.props} />
```

KUVA 3. Qdt-komponentti koodissa

Komponentille täytyy tätä ennen kuitenkin ”kertoa”, mikä graafi komponentissa halutaan esittää. Tämä kerrotaan seuraavanlaisen konfiguraation avulla:

```
const viz = {
  type: 'QdtViz',
  props: {
    type: 'table', id: 'xxxxxx', width: 'auto', height: '400px',
  },
};
```

KUVA 4. Qdt-komponentin konfiguraatio

Graafin tunniste kopioidaan Qlik-sovelluksesta ja lisätään konfiguraation kohtaan ”id”. Lisäksi komponentille kerrotaan graafin tyyppi sekä haluttu leveys ja korkeus.

5.3 Qlik Sense -autentikointi

Qlik Sense tarjoaa muutamia erilaisia autentikointitapoja käyttäjähallintaa varten. Tässä projektissa autentikointitavaksi valikoitui JWT eli JSON Web Token. JWT on yksinkertaisuudessaan merkkijono, jonka avulla käyttäjä voi todentaa olevansa oikeutettu johonkin sisältöön.

Qdt-komponenttien kannalta autentikointi osoittautui erityisen merkitykselliseksi, sillä sovelluskoodin sijaitessa eri palvelimella kuin Qlik Sense, autentikointi oli suoritettava koodissa ennen kuin Qlik-resursseja voitiin noutaa. Sovelluskoodissa autentikointi suoritettiin tekemällä axios-kutsu Qlik-palvelimelle. Jotta autentikointi onnistui, axios-kutsuun oli lisättävä authorization-header, eli auktorisoinnin mahdollistava otsake. Otsakkeen arvoksi annettiin JWT-merkkijono. Lisäksi kutsu vaati boolean-arvon *withCredentials: true*.

Autentikointi oli myös sijoitettava koodissa oikeaan kohtaan. Yleensä uusia moduuleita tuodaan projektiin aivan kooditiedoston alussa komennolla *import name from "module-name";*. Autentikoinnin onnistumisen kannalta moduulin tuominen tiedostoon oli kuitenkin tehtävä vasta axios-kutsun jälkeen.

5.4 Capability API

Aluksi lienee tarpeen pohjustaa yleisellä tasolla API-rajapintojen tarkoitusta ja käyttöä. APIt (application programming interface) ovat monimutkaista koodia, jota eri ohjelmointikielet tarjoavat yksinkertaisemmassa muodossa ohjelmoijien käyttöön. APIt ”piilottavat” monimutkaisen koodin niin, että ohjelmoijan tarvitsee vain lisätä jokin yksinkertainen koodilohko projektiinsa, ja tämä yksinkertainen koodi tuo koko monimutkaisen toiminnon ohjelman käyttöön. (Mozilla Developer Network 2020a.)

Niin kutsutussa client-side Javascriptissa, eli asiakaspuolen Javascript-koodissa APIen käyttö on hyvin yleistä. Ne voidaan jakaa kahteen kategoriaan, selain-API-rajapintoihin sekä kolmannen osapuolen API-rajapintoihin. Selain-APIt ovat nettiselaimiin rakennettuja koodikokonaisuuksia. Niiden avulla ohjelmoija voi käsitellä selaimesta tai laitteesta saatavaa dataa. Yhtenä esimerkkinä voisi antaa vaikkapa selaimen äänet, joita voi säätä tietyn APIn avulla. Kolmannen osapuolen APIt sen sijaan täytyy erikseen noutaa internetistä. Ne eivät siis oletusarvoisesti löydy käyttäjän selaimesta. (Mozilla Developer Network 2020a.)

APIen toiminta perustuu siihen, että ohjelmoijan luoma koodi on yhteydessä APIin Javascript-objektin avulla. Objekti sisältää metodeita, joiden avulla APIn toiminnallisuudet saadaan ohjelman käyttöön. Qdt-komponenttien osalta voimme tarkastella tätä menetelmää tutustumalla Qdt-komponenttien lähdekoodiin kuvan 5 avulla.

```
return new Promise((resolve) => {
  if (globals.qlik) {
    const app = globals.qlik.openApp(config.appId, { ...config, isSecure: config.secure, prefix });
    // apply theme set in QSE
    app.theme.get().then((theme) => {
      theme.apply();
    });
    resolve(app);
  } else {
    window.require(['js/qlik'], (q) => {
      globals.qlik = q;
      globals.resize = () => {
        q.resize();
      };
      const app = q.openApp(config.appId, { ...config, isSecure: config.secure, prefix });
      resolve(app);
    });
  }
});
```

KUVA 5. Qdt-komponentin lähdekoodia (Qlik Demo Team 2020)

Kolmannella koodirivillä hyödynnetään Qlikin omaa Root API -rajapintaa. Se on ulkoinen rajapinta Qlik Senseen, ja sen avulla voidaan esimerkiksi avata yhteys haluttuun Qlik-sovellukseen (Qlik Help 2020b). Yllä olevassa koodissa käytetään openApp-metodia, joka antaa uuden Javascript-objektin ohjelmoijan käyttöön. Tämä objekti hyödyntää App API -rajapintaa, joka avaa jälleen uusia metodeita Qlik-sovelluksen hyödyntämiseksi koodissa.

Rakentamassamme raportointiportaalissa hyödynnettiin Capability API-rajapintoja Qdt-komponenttien lisäksi suoraan. Kun Qdt-komponentit olivat jo käytössä, app-objektin hakeminen koodiin onnistui Qdt-komponentin kautta.
Sovelluksessa käytettiin esimerkiksi App API -rajapinnan metodia clearAll, jolla voidaan tyhjentää loppukäyttäjän tekemiä rajauksia. Graafeja sisältäville sivuille lisättiin Poista valinnat -painike, jolla pystyi palauttamaan graafit takaisin alkutilaan.

Qdt-komponenttien käyttö tuo parhaimmillaan paljon joustavuutta web-sovellukseen. Tarpeen tullen uusia graafeja voi luoda ”lennosta” web-sovelluksen puo-

lella, ilman että graafi täytyy ensin luoda Qlik Sense -sovelluksessa. Tällä ominaisuudella olisi potentiaalia tuoda räätälöity web-ratkaisu jälleen lähemmän self-service-mallia, sillä loppukäyttäjä voisi luoda osan graafeista itse web-sovelluksessa. Tämä vaatisi kuitenkin lisää kehitystyötä, sillä tällä hetkellä uusien graafien luominen Qdt-komponenttien avulla vaatii käyttäjältä ohjelmointiosaamista. Yksi mahdollisuus voisi olla, että web-sovellus ottaisi käyttäjän syötteitä vastaan vaikkapa tekstikentän kautta, ja generoisi koodissa uusia graafeja. Tätä ominaisuutta ei kuitenkaan tämän projektin puitteissa toteutettu.

6 POHDINTA

Tässä opinnäytetyössä etsittiin uusia ratkaisuja raportoinnin tarpeisiin. Ensisijainen tavoite oli löytää ratkaisuja web-sovelluksen ja visualisointityökalun (tässä tapauksessa Qlik Sense) integrointiin. Projektin aikana erilaisia integrointimahdollisuuksia pohdittiin ja myös testattiin, mutta lopullinen valinta olivat Qlik Sense -graafien upottamiseen tarkoitetut Qdt-komponentit.

Pidän Qdt-komponentteja erittäin hyvänä työkaluna räätälöityjen raportointiratkaisujen luomisessa. Qdt-komponentit avasivat paljon kattavammat mahdollisuudet kuin esimerkiksi yksinkertaiset iframe-upotukset. Qdt-komponenttien avulla on esimerkiksi mahdollista luoda graafeja ”lennosta” sovelluskoodin puolella, mikä tuo uusia mahdollisuuksia integrointeihin. Qdt-komponentit helpottivat myös Capability API -rajapintojen käyttöönottoa, mikä omalta osaltaan laajensi entisestään Qlik Sensen sovellusmahdollisuuksia. Qlik Sense itsessään on erittäin käytetty työkalu visualisointien ja raportointiratkaisujen rakentamiseen, joten dokumentaatiot ja käyttäjäfoorumien keskustelut auttavat monissa ongelmatilanteissa.

Tämän opinnäytetyöprojektin puitteissa ei ollut mahdollisuutta jatkaa pidemmälle räätälöidyn portaalin self-service-puolta. Alkuperäinen toive siitä, että loppukäyttäjät voisivat vaivattomasti muokata portaalia ja rakentaa uusia visualisointeja, jäi toteutumatta. Tämä olisi erittäin hyödyllinen ja mielenkiintoinen aihe jatkokehittelylle.

On todennäköistä, että liiketoimintatiedon merkitys vain kasvaa entisestään, ja visualisointeja halutaan hyödyntää yhä laajemmin myös web-ohjelmoinnissa. Erilaisille integraatioratkaisuille on selvää kysyntää, mutta trendiä jarruttaa juuri teknisen osaamisen puute. Nähtäväksi jää, pystyvätkö suuret BI-työkaluyritykset (Qlik, Microsoft Power BI, Tableau) vastaamaan tähän haasteeseen ja luomaan ratkaisuja, jotka eivät vaadi erityistä teknistä harjaantuneisuutta. Siihen asti raportointiratkaisujen piirissä on selvää tarvetta henkilöille, joilla on sekä BI- että ohjelmointiosaamista.

LÄHTEET

Boduch 2020. React and React Native

Davenport, T. H., Harris, J. G., Paalosalo, M. 2007. Analysoi ja voita: kilpailun uusi tiede. Helsinki: Talentum.

Domes, S. 2017. Progressive web apps with React: create lightning fast web apps with native power using React and Firebase. Birmingham: Pact Publishing.

Labbe, P., Hand, P., Kharpate, N. 2018. Qlik Sense Cookbook - Second Edition. Birmingham: Packt Publishing.

Mozilla Developer Network. 2020a. Introduction to web APIs. Luettu 12.1.2020 https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Clientside_web_APIs/Introduction.

Mozilla Developer Network. 2021b. <iframe>: The Inline Frame element. Luettu 10.1.2021 https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

Qlik Demo Team. 2020. QdtCapabilityApp.js-lähdekoodi GitHubissa. Luettu: 10.1.2021 https://github.com/qlik-demo-team/qdt-components/blob/master/src/qdtCapabilityApp.js

Qlik Design Blog. 2020. Qdt-components 2020 Update. Blogi-kirjoitus. Luettu 10.1.2021 https://community.qlik.com/t5/Qlik-Design-Blog/Qdt-components-2020-Update/ba-p/1663426

Qlik Help. 2020a. Luettu 11.1.2021. https://help.qlik.com/en-US/sense

Qlik Help. 2020b. Capability APIs. Luettu 11.1.2021. https://help.qlik.com/en-US/sense-developer/November2020/Subsystems/APIs/Content/Sense_ClientAPIs/capability-apis-reference.htm

Robb, D. 2012. Visualization Broadens Business Intelligence's Appeal. Luettu: 16.10.2020 http://www.cbpp.uaa.alaska.edu/afef/visualization_broadens_business.htm

Sahay, A. 2017. Data visualization. Volume 1, Recent trends and applications using conventional and big data. New York: Business Expert Press.

Sherif, A. 2016. Practical Business Intelligence. Birmingham: Packt Publishing

Wilke, C. O. 2019. Fundamentals of Data Visualization. Sebastopol: O’Reilly Media, Inc.