Opinnäytetyö (AMK)

Tietojenkäsittely

2018

Tommi Hautaviita

KAAVIOIDEN HYÖDYNTÄMINEN DATAN VISUALISOINNISSA

– case Yritys X

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely

2018 | 43 sivua

Tommi Hautaviita

KAAVIOIDEN HYÖDYNTÄMINEN DATAN VISUALISOINNISSA

- case Yritys X

IT-alalla on yhä kovempi kilpailu asiakkaista. Tuotteen visuaalisella ulkoasulla on nykypäivänä suurempi merkitys kuin aikaisemmin, ja sen myötä graafisten käyttöliittymien päivittämiseen keskitetään alakohtaisesti paljon resursseja. Yritykset pyrkivät erottumaan joukosta eri tavoin. Toteutusten käyttäjäystävällisyys voi olla ratkaiseva tekijä asiakkaiden vertaillessa eri yritysten ratkaisuja.

Yritys X on turkulainen tietoturvapalveluita tarjoava yritys. Tietoturvafirmoissa datan visualisoinnilla on tärkeä rooli järjestelmän tilan ilmaisemisessa asiakkaille. Visualisointiratkaisujen tulee olla selkeitä ja helposti luettavia. Nykypäivän www-ohjelmoinnin nopea kehitystahti on tuonut mukanaan lukemattoman määrän eri JavaScript-kirjastoja datan visualisointiin.

Tämän opinnäytetyön tarkoituksena oli tutkia toimeksiantoyritykselle Chart.js-kirjaston ja Vue.js-rungon sopivuutta dataa visualisoivien kaavioiden luomiseen verkkoportaalissa, ja sen pohjalta luoda toimeksiantoyrityksen verkkoportaaliin uusia kaavioita kyseisiä teknologioita hyödyntäen. Samalla pohdittiin Vue.js-rungon toimintaperiaatteita perehtymällä tarkemmin sen modulaariseen kehitysfilosofiaan. Työssä hyödynnettiin Chart.js:n ja Vue.js:n dokumentaatiota. Samalla yritettiin hahmottaa miten kyseisiä ohjelmointikonsepteja hyödyntäen onnistuisi parhaiten selkeiden tulkittavien kaavioiden luonti.

Opinnäytetyössä rakennetut kaaviot tuli käyttöön yrityksen verkkoportaalissa, ja jatkokehitysehdotuksia kaavioille esitetään aktiivisesti. Kaaviokirjaston hyvä käytettävyys mahdollistaa uusien ratkaisujen syöttämisen järjestelmään, ja toimeksiantoyrityksen muutkin verkkoratkaisut uusitaan tarvittaessa.

ASIASANAT:

Chart.js, JavaScript, Kaaviot, Vue.js, ElasticSearch, Kibana, ElasticStack, JavaScript

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology

2018 | 43 pages

Tommi Hautaviita

THE UTILIZATION OF GRAPHS IN DATA VISUALIZATION

- case Company X

In the information security field, data visualization is an important method for communicating to the customer the status of the system and the health of its various components. The visual appearance of the product is the be-all-end-all in the modern day and, with it, more and more resources are being poured into updating graphical user interfaces. Visualization solutions have to be clear and easy to read. Companies strive to stand out in a variety of ways. User-friendliness can be a decisive factor for customers when comparing solutions from different companies.

The goal of this thesis was to update the existing data visualization methods of the commissioning company, Company X, an information security firm based in Turku, to the modern age with new graphs built using Chart.js which is a JavaScript library for data visualization and Vue.js, an open-source web development framework. The current modus operandi for data visualization within the company was also streamlined by separating the backend from the frontend using Vue.js and JavaScript and eliminating the old graphs built with Google Charts that were deemed outdated. The thesis also delves into the intricacies of both Chart.js and Vue.js through documentation and studies their efficacy for building graphs.

The new graphs built in this thesis were immediately put into use in the company's web portal, and the solutions proposed in this thesis met the demands of the commissioning company. Furthermore, the company is in the process of revamping the entire portal using new data visualization solutions, all of which are being built using the technology introduced in this thesis.

KEYWORDS:

Chart.js, Vue.js, JavaScript, Web Development, UI

SISÄLTÖ

KOODI

KUVAT

SANASTO

CSS	Cascading Style Sheets, tyyliohjeiden laji
Chart.js	JavaScript-kirjasto datan visualisointiin (Chart.js 2018)
Dashboard	Ohjauspaneeli, kojelauta
ElasticSearch	Javalla kehitetty hakukone, joka muodostaa Kibanan ja Logstashin kanssa Elastic Stackin (Elastic 2018)
ElasticStack	ElasticSearch, Logstash ja Kibana
HTML	Hypertext Markup Language, merkintäkieli (W3 2018)
JavaScript	Ohjelmointikieli (MDN 2018)
JSON	JavaScript Object Notation, avoimen standardin tiedosto-muoto, joka toimii avain- ja arvopariperiaatteella (JSON 2018)
Kibana	Avoimen lähdekoodin datan visualisointityökalu ElasticSearch alustalle (Elastic 2018)
Logstash	Lokiparsimisrajapinta joka pohjautuu Elasticiin (Elastic 2018)
SOC	Security Operations Center, keskitetty valvomo josta hoidetaan organisaation tietoturvan valvonta, uhkien seuranta sekä tietoturvapoikkeamien hallinta (Combitech Finland 2017)
Vue	Lyhennetty kutsumanimi Vue.js-kirjastolle
Vue.js	Avoimen lähdekoodin JavaScript-runko (Vue.js 2018)

1 JOHDANTO

Tietoturvayrityksien määrä Suomessa on lisääntynyt huomattavasti viimeisen vuosikymmenen aikana, ja arvioiden mukaan tietoturva-alalle voi 2022 mennessä syntyä jopa 20 000 uutta työpaikkaa (Lehto 2017). Moni alan yritys tarjoaa pitkälti samankaltaisia ratkaisuja: konsultointia, seminaareja ja SOC-palveluja. SOC eli Security Operations Center on tärkeä osa nykypäivän kyberpuolustusta, ja se tulisi löytyä jossain muodossa kaikista organisaatioista. SOC on keskitetty valvomo, joka vastaa yrityksen tietoturvan valvonnasta (Combitech Finland 2017). SOC tarkkailee yrityksen liikennettä.

SOC ei ole pelkkää valvontaa ja riskihavainnointia. Olennaisena siinä on myös kyky käsitellä uhkia ja poikkeamia reaaliajassa. Nykyaikainen SOC koostuu osaavasta henkilöstöstä ja kehittyneistä kyberturvallisuusvalvonnan työkaluista. Ajan tasalla oleva puolustusjärjestelmä, joka kykenee torjumaan älykkäimmätkin hyökkäykset ja tunkeutumisyritykset ovat nykypäivänä kaikki kaikessa, sillä tietoturvauhat kehittyvät ja muuttuvat koko ajan nopeammin. Tavallisesta virustorjunnasta, oli se sitten kuinka edistyksellinen, ei ole paljoa hyötyä kehittyneempiä tietomurtoyrityksiä vastaan (Combitech Finland 2017).

SOC:n pitää jollain tavalla kommunikoida asiakkaalle yrityksen järjestelmän tila: tämä tapahtuu useimmiten erilaisilla kaavioilla. Kaaviot visualisoivat vaikeasti tulkittavissa olevan datan sellaiseen muotoon, että kuka tahansa osaa tulkita yrityksen järjestelmän tilaa.

Opinnäytetyön toimeksiantona oli luoda Yritys X:n SOC-portaaliin uusia kaavionäkymiä. Aikaisemmat ratkaisut olivat melko karkeita, ja kaavioissa yhdisteltiin samassa koodissa frontendiä ja backendiä, joka ei ole hyvä käytäntö. Siirtymällä Vue.js-alustaan frontend käsittää vain sille ominaisen osion koodista, ja backend tekee myöskin vain sille ominaisen osan työstä eikä koodia yhdistellä samassa tiedostossa.

Edelliset kaaviot eivät olleet visuaalisesti miellyttäviä. Ne hyödynsivät Googlen omaa kaaviokirjastoa, jossa muokkaamismahdollisuudet ovat hyvin rajatut ja kaaviot karkean näköisiä. Kaaviot eivät myöskään olleet responsiivisia. Toimeksiantajayrityksen pyynnöstä olemassa olevien kaavioiden ulkoasu päivitetään, ja ne siirretään käyttämään Vue.js-alustaa, jolloin samalla taataan niiden modulaarisuus.

Opinnäytetyölle ei asetettu työsuhteen alettua syyskuun lopussa varsinaista aikarajaa, vaan sain vapaat kädet tehdä sitä omaan tahtiin syyskuusta alkaen. Lyhyen aikamäärän myötä kaikkia ominaisuuksia ei pystytty toteuttamaan, mutta alustava pohja luotiin.

Tämä opinnäytetyö on konstruktiivinen: siinä toimeksiannon mukaisesti muutetaan olemassa olevat kaaviot hyödyntämään Vue.js-runkoa ja Chart.js-kirjastoa. Visuaalista ulkoasua kehitetään pyyntöjen mukaan. Teoriaosuus muodostuu pitkälti toteutuksessa käytetyistä teknologioista ja niiden selittämisestä.

2 TEKNOLOGIAT

Tässä luvussa esitellään opinnäyteyössä hyödynnettyjä teknologioita. Vaikka tietyt käsitteet, kuten Chart.js ja Vue pohjautuvat laajasti JavaScript-kielen materiaaliin, käyn ne läpi omina alalukuinaan.

2.1 JavaScript-kieli

JavaScript on kevyt korkean tason dynaamisesti tyypitetty ohjelmointikieli, joka HTML- ja CSS-kielien kanssa muodostaa modernin www-ohjelmoinnin perustan. JavaScriptiä käytetään pitkälti tuomaan verkkosivuihin interaktiota ja dynaamista toiminnallisuutta. JavaScriptiä onkin siis yleisimmin hyödynnetty kieli verkkosivujen luomisessa, tosin sitä on hyödynnetty myös palvelinten verkko-ohjelmoinnissa ja tietyissä tapauksissa mobiilisovellusten luomisessa. JavaScript ohjelmointikielenä on melko kompakti mutta erittäin joustava. Kehittäjät ovat kirjoittaneet laajanvalikoiman työkaluja, jotka valjastavat valtavan määrän lisätoimintoja mahdollisimman vähäisellä lisätyöllä (MDN 2018). Yksinkertaisessa esimerkissä script-tagin sisään kirjoitettu koodi tulostaa keskelle sivua ”Hello World” -esimerkkilauseen (Koodi 1).

```
<html>
<body>



 <p> Ennen skriptiä.. </p>

 <script>

alert( 'Hello World!');

</script>

<p> skriptin jälkeen</p>
</body>
</html>
```

Koodi 1. Esimerkki JavaScript ohjelmointikielestä.

Viimeisin kielen määrittely on JavaScript 1.8.5. Yksittäistä ehdot sanelevaa syntaksia JavaScriptille ei löydy. JavaScript ei nimestään huolimatta pohjaudu Java-kieleen: JavaScript ja Java ovat tietyllä tavalla samankaltaisia, mutta eroavat silti merkittävästi toisistaan. JavaScript-kieli muistuttaa Javaa, mutta siitä puuttuu Javan staattinen kirjoittaminen ja vahva tyypitys (MDN 2018). JavaScriptiä voidaan kirjoittaa tiedoston sisään joko html-sivun otsakkeeseen tai body-elementtiin, tai vaihtoehtoisesti omalle ulkoiselle tiedostolle. Koodin tulee olla script-tagien sisällä (Koodi 1).

2.2 Vue.js-runko

Vue.js on Evan Youn kehittämä avoimen lähdekoodin JavaScript-runko reaktiivisten käyttölittymien rakentamiseen (Vue.js 2018). Inspiraatio kehittää Vue.js tuli halusta saada käyttöön kevyt JavaScript-runko ilman liiallisia toimintoja ja turhia ominaisuuksia. You työskenteli aikaisemmin Googlella Angular.js:n parissa, mutta jätti projektin, koska hän halusi kehittää Angular-rungosta yksinkertaisemman vastaavan. Toisin kuin muut JavaScript-pohjaiset rungot, Vue on suunniteltu helposti muokattavaksi. Sitä onkin helppo hyödyntää olemassa olevissa verkkoprojekteissa sen kevyen ja muovattavan rakennusfilosofian myötä (Niiranen 2018). Tästä syystä se soveltuu oivallisesti tässä työssä laadittavaan visualisointiin (Koodi 2). Vue käyttää HTML-pohjaista templatointisyntaksia. Templatointisyntaksilla tarkoitetaan Vue.js:n kykyä sitoa käyttäjälle piirtyviä arvoja nettisivun runkoon dynaamisesti niin että verkkosivun HTML päivittyy automaattisesti Vue.js:n tietojen päivittyessä. Templatointisyntaksin avulla onkin siis mahdollista luoda verkkosivulle dynaamisesti uutta sisältöä Vue.js-komponentin sisällä olevan datan pohjalta (Vue.js 2018).

```
<template>
    <div><h1>
        hei
    </h1></div>
</template>
```

Koodi 2. yksinkertainen Vue-sovellus.

Vueta verrataan usein sen pääkilpailijaan Reactiin, vaikka React ei ole varsinaisesti framework vaan kirjasto. React on Facebookin kehittämä avoimen lähdekoodin JavaScript-kirjasto (Niiranen 2018). Se toimii Vuen tavoin JavaScript-runkona monissa nykypäivän verkkosovelluksissa (Mociun 2015). Vue on pitkälti yhden henkilön kehitysprojekti, jonka

takia ei voida välttämättä odottaa Vuelle löytyvän yhtä suurta määrää tukea kuin Facebookin kehittämälle Reactille (Niiranen 2018).

2.3 JSON-tiedonsiirtomuoto

JSON (JavaScript Object Notation) on kevyt, avoimen lähdekoodin tiedonsiirtomuoto (JSON 2018). JSON on JavaScriptistä riippumaton, joka tarkoittaa, että projektin ei tarvitse olla kirjoitettu JavaScript-kielellä, jotta JSONia voisi hyödyntää. Nimi pohjautuu sen JavaScript-objektien rakennetta muistuttavasta avain-arvo-pari-toimintaperiaatteesta (Koodi 3).

```
"name":"John"
```

Koodi 3. JSON-avain-arvo-parinotaatioesimerkki.

Viimeisen 15 vuoden aikana JSONista on tullut välttämätön tiedonsiirtomuoto nykypäivän verkkorajapinnoissa. JSONia tulisi käyttää sovelluksissa, jotka keskustelevat selaimen kanssa client-server-periaatteella. Muut tiedonsiirtomuodot kuten XML koetaan nykypäivän verkko-ohjelmoinnissa vanhentuneiksi (Freeman 2017).

2.4 REST API & HTTP-GET

2.4.1 REST API

REST (Representational State Transfer) on yleisesti käytetty arkkitehtuurimalli verkkorajapintojen luomiseen moderneissa client-server-verkkosovelluksissa, jotka hyödyntävät HTTP-protokollaa internetin yli kommunikoinnissa. REST-konseptin määritteli tutkija Roy Fielding vuonna 2000 julkaistussa väitöskirjassaan Architectural Styles and the Design of Network-based Software Architectures. REST tuli tunnetuksi 2000-luvulla www-buumin myötä verkkosovellusten yleistyttyä, jolloin se myöskin päihitti kilpailijansa sen yksinkertaisuudella ja helpolla luettavuudella. Konkreettisesti REST-määritelmällä tarkoitetaan resurssienhallintaa verkkosovelluksessa HTTP-metodien avulla (Kivisaari 2016).

API on lyhenne sanoista Application Programming Interface, ja se tarkoittaa lyhyesti sanottuna ohjelmointirajapintaa, jonka avulla verkkosovellukseen voidaan tehdä kutsuja sovelluksen ulkopuolisista lähteistä (Cenno 2014).

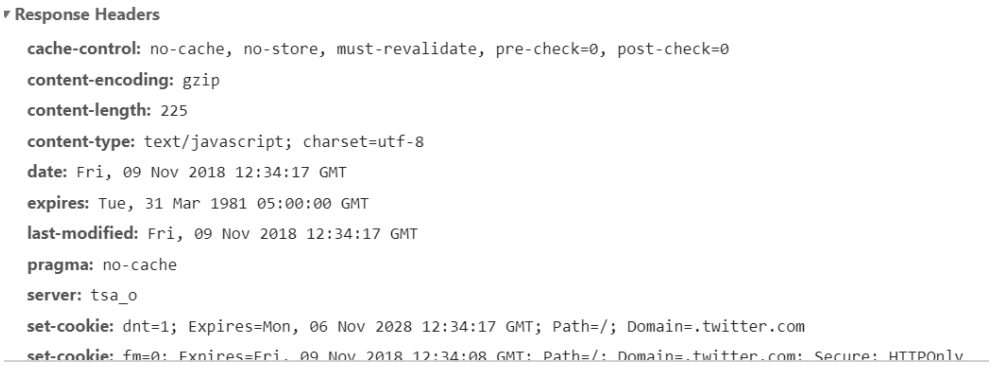
Sillä REST on rakennettu HTTP:n päälle, se on myöskin samalla tavalla tilaton. Tätä myötä myös monet HTTP-protokollan innovaatiot ja ratkaisut pätevät RESTiin, mukaan lukien HTTP:lle ominaiset piirteet kuten tilattomien palveluiden skaalautuminen, erilaisten sisältötyyppien tuki ja monipuoliset välimuistiominaisuudet (Kivisaari 2016).

2.4.2 HTTP-GET-pyyntö

GET on tietopyyntö, jolla haetaan dataa palvelimelta. Toisin kuin POST-pyynnöissä, GET-pyyntöjen mukana lähetetään vain evästeet ja osoitteeseen kirjoitetut GET-parametrit. GET-pyynnöllä ei voi tallentaa dataa tietokantaan. Selain kommunikoi palvelimen kanssa HTTP-protokollaa hyödyntäen (Korpela 2009). Chrome-selaimella konsolinäkymän Network -välilehdestä voidaan seurata palvelimen ja selaimen välistä liikennettä (Kuva 1). Lähetetystä pyynnöstä ja palvelimen vastauksesta saadaan lisätietoja Network-välilehden Headers -eli otsakkeet-osiosta. Otsakkeet tarjoavat yksityiskohtia tehdystä pyynnöstä ja palvelimen palauttamasta vastauksesta (Advanced Kittenry 2014).

Kuva 1. GET-pyynnön Headers-alakohdan esimerkki (Google Chrome 2018).

Headers-alakohdan general-osiosta selviää mihin osoitteiseen GET-pyyntö tehtiin. General-osio sisältää yleistietoa suoritetusta pyynnöstä, muun muassa pyynnön statuskoodin. Jos pyyntö tehtiin onnistuneesti, pyyntö palauttaa vastauksena statuskoodin 200. 2xx-alkuiset statuskoodit viittaavat aina onnistuneeseen pyyntöön (Advanced Kittenry 2014).



Kuva 2. Response Headers -esimerkki (Google Chrome 2018).

Response Headers -osiosta nähdään vastauksen koko ja tarkka vastaushetki. Osio käsittää kaikki palvelimen lähettämään vastaukseen liittyvät otsakkeet. Olennaisinta tässä osioissa on content-type-otsakkeen sisältämä tieto (Kuva 2). Content-type-otsakkeesta nähdään missä muodossa palvelin palauttaa dataa. Esimerkkitapauksessa text/javascript ja charset=utf-8 ilmaisee käyttäjälle, että kyseessä on JavaScript–UTF-8-tekstitiedosto.

2.5 Chart.js-kirjasto

2.5.1 Esittely

Chart.js on datan kartoittamiseen suunniteltu JavaScript-kirjasto. Se mahdollistaa HTML-tiedostoon upotettujen HTML5-kaavioiden piirtämisen sivulle. Responsiivisen kehitysfilosofian myötä se sopii myös mobiilialustoille (Chart.js 2018). Kirjastosta on tullut lyhyessä ajassa hyvin suosittu ohjelmoijien keskuudessa sen yksinkertaisen plug-n-play-toiminnallisuuden ja muihin kaaviokirjastoihin verrattuna selkeän dokumentaation ansiosta (Baaj 2017). Aikaisemmin portaalissa hyödynnettiin Google Charts -kaaviokirjastoa. Google Charts on Googlen ylläpitämä avoimen lähdekoodin kaaviokirjasto (Tutorialspoint 2018). Kyseinen kaaviokirjasto ei tyydyttänyt toimeksiantoyrityksen tarpeita: sen tarjoamat kaaviot todettiin olevan karkean näköisiä ja tönkköjä käytettävyydeltään.

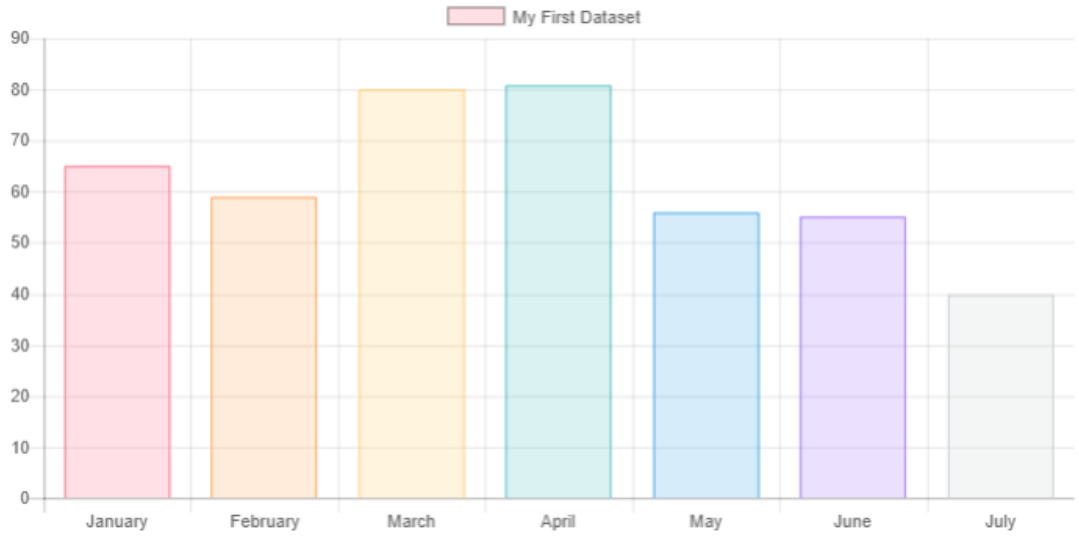
2.5.2 Käyttöönotto

Chart.js-kirjasto on helppo ottaa käyttöön olemassa olevassa verkkosivussa. Oheisessa koodissa luodaan yksinkertainen Chart.js-demo, jossa ovat tagit, joita kaavion luomiseen tarvitaan:

```
<canvas id="myChart" width="400" height="400"></canvas>
<script>
var ctx = document.getElementById("myChart").getContext('2d');
var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ["January", "February", "March", "April", "May", "June", “July”],
        datasets: [{
            label: 'My First Dataset',
            data: [12, 19, 3, 5, 2, 3],
        }]
    },
    options: {
        }
    }
});</script>
```

Koodi 4. Pylväskaavion alustaminen

Kaavan mallintamiseen vaaditaan yksi <canvas> tagi <script> tagin ohella HTML-tiedostoon upotettuna. Sen lisäksi Chart-luokka pitää alustaa: esimerkissä tämä tapahtuu ctx-muuttujan määrittelyssä. Muuttuja edustaa sitä elementtiä johon kaavio piirretään, tässä tapauksessa elementtiä myChart. Kaavion visuaaliset ominaisuudet on esimerkissä syötetty parametrina canvas -tagin sisään. Esimerkissä luodaan yksinkertainen pylväskaavio kovakoodattua datasettiä hyödyntäen. Labels-propertyn array (suom. taulukko) sisältää nimikkeet, jotka piirtyvät kuvaamaan dataa. Datasets-objekti syötetään data-propertyn sisään taulukkona. Datasets-objektilla tarkoitetaan taulukkoa, josta kaavio poimii arvot, joita hyödyntämällä kaavio piirretään. Kaavion piirtämiseen vaaditaan lukuarvoja sisältävä taulukko. Esimerkin datasets-objekti sisältää yhden taulukon, data, ja sen lisäksi labelin -eli nimikkeen, joka määrittelee datasetin nimen. Kukin datasets-objektin sisällä olevan data-taulukon numeroarvoista edustaa omaa pylvästä, ja pylväitä piirtyy yksi jokaista numeroarvoa kohden. Options-property sisältää kaavion ominaisuuksien määrittelyyn valmiudet. Se vastaanottaa datasetin ulkopuoliset visuaaliseen toteutukseen liittyvät pyynnöt (Kuva 3).



Kuva 3. Yksinkertainen pylväskaavio (Bar Chart), jossa Labels-taulukko sisältää kuukausien nimet (Chart.js 2018).

Kuvitteellisessa viivakaaviossa siihen voisi esimerkiksi syöttää pylvään kokoon liittyviä määrittelyjä (Koodi 5). BeginAtZero-komento on hyvä esimerkki Chart.js:n muovattavasta luonteesta. Se varmistaa, että kaikki pylväskaavion y-akselin elementit alkavat nollasta.

```
options: {
    scales: {
        yAxes: [{
            ticks: {
                beginAtZero:true
            }
        }]
    }
}
```

Koodi 5. Viivakaavion y-akselin parametri: beginAtZero-komento alustaa kaavion datan nollasta.

2.5.3 Rajoitukset

Chart.js-kirjasto soveltuu hyvin tilanteisiin, joissa halutut kaaviotyypit tiedetään entuudestaan ja sovelluksessa ei tarvitse olla valmiuksia uusien kaaviotyyppien luomiseen.

Jos toteutukseen tarvitaan runsas määrä lisäominaisuuksia ja asiakaan tarvitsee pystyä muokkaamaan kaavioita halutessaan, saattaa olla parempi hyödyntää muuta kaaviokirjastoa. Kirjoitushetkellä vaihtoehdot on rajattu seitsemään eri kaaviotyyppiin: line, bar, radar, doughtnut, pie, polar area, bubble ja scatter (Chart.js 2018).

2.5.4 Vue-Chart.js

Vue-Chart.js on Vuelle tarkoitettu Chart.js-wrapper, joka mahdollistaa uudelleenkäytettävien kaaviokomponenttien luomisen Vuella. Vue-chart.js tekee Chart.js-kirjaston käyttämisestä Vue -rungon sisällä helppoa (Juszczak 2018). Aloittaakseen Vue-chart.js:n käytön pitää pohjakaavio ensin tuoda Vue-tiedostoon import-lauseella, jonka jälkeen se otetaan käyttöön extends -komennolla. Kaavio piirretään RenderChart-funktiolla (Koodi 6).

```
import { Bar } from 'vue-chartjs'
export default {
  extends: Bar,
  mounted () {
    this.renderChart(data, options)
  }}
```

Koodi 6. Vue-chart.js-esimerkki.

3 TOIMEKSIANNON LÄHTÖKOHDAT

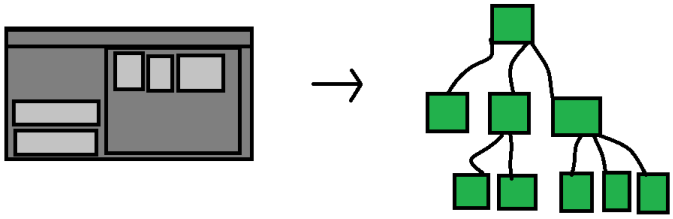
Toimeksiantoyritys halusi päivittää nykyiset olemassa olevat kaaviot käyttämään Vueta ja Chart.js -kirjastoa. Kyseisellä muutoksella pyrittiin parantamaan SOC-portaalin visuaalista ulkoasua ja yleistä toiminnallisuutta. Samalla pyritään irrottamaan frontendin backendista. Vanhassa toteutuksessa kaavioihin liittyvä koodi frontendillä oli pitkälti jQueryä. Frontendissa yhdisteltiin PHP:ta ja JavaScriptiä. Tästä syystä sitä oli aluksi vaikea ymmärtää. Yhdistely ei ole hyvä käytäntö sillä liiallinen kielien yhdistely tekee koodista vaikeaselkoista ja hidastaa jatkokehitysprosessia projektin siirryttyä uuden ohjelmoitsijan käsiin. Liika yhdistely ja koodin puukotusmainen sovittaminen olemassa olevaan koodiin voi johtaa koodispagetin syntyyn. Koodispagetilla tarkoitetaan koodia, joka on sotkuista ja vaikeasti tulkittavaa (Koodi 7). Koodispagetin syntyyn vaikuttaa monet tekijät: kokemuksen puute, ohjelmointityylisääntöjen puuttuminen järjestelmästä ja hektiset, jatkuvasti muuttuvat vaatimukset ja toivomukset ovat yleisimmät syyt koodispagetin syntyyn (Pizka 2004). Huonosti kirjoitettu koodi vaikeuttaa koodin varsinaisen tarkoituksen ymmärrettävyyttä, ja tekee olemassa olevan projektin ylläpitämisestä hankalaa. Koodispagetti voi pahimmassa tapauksessa johtaa lisäkuluihin ja työtehon heikkenemiseen, sillä ohjelmistokehittäjä joutuu pyhittämään enemmän aikaa koodin selvittämiseen, mikä hidastaa prosessin valmistumista (Lietsala 2009).

```
var taskapp.events.users = {
  initialize: function(){
    this.$tasksWrapper = $("#tasksWrapper");
      if(!this.$tasksWrapper.length) this.createWrapper();
  loadTasks: function(){
    var _this = this;
    $.load ("query.complete", function(){ _this.executeQuery(); });
      this.$tasksList
        .delegate("#btnDelete","click", function() { _this.deleteItem();
return false; });
        .delegate("#btnsave","click",   function() { _this.save(); return
false;));
        .delegate("#btnModify","click", function() { _this.modify(); re-
turn false;));
  },
  loadtasks : function() {
    $(".myTasks").validationEngine();
  },
  deleteItem: function(){
    $(this).closest('.item').remove();
  },
```

Koodi 7. JQuery-spagetti – yksinkertainen toteutus on tehty tuhannen mutkan kautta monimutkaisesti (Dugas 2012).

Refaktoroinnilla on todettu olevan koodin laatua parantava vaikutus. Refaktoroinnilla tarkoitetaan prosessia, jossa koodia muutetaan siten että alkuperäinen funktionaalisuus säilyy, mutta sisäinen rakenne muuttuu. Se parantaa koodin luettavuutta, ja sen myötä sitä on helpompi ylläpitää (Fowler 2013, 46-48). Refaktoroinnin edellytys ei ole rikkinäinen koodi. Refaktoroinnin hyöty näkyy myös nopeammassa suorituskyvyssä ja bugifiksauksen helpottumisessa (Contribyte 2018). Vue.js:n refaktorointia suosiva toimintaperiaate pyrkii ehkäisemään koodispagetin kertymistä. Siinä suositaan ominaisuuksien refaktorointia. Kaaviot hyödyntävät refaktorointia, sillä niiden toiminnallisuus on kirjoitettu useaan eri Vue komponenttiin (Kuva 4). Vue-komponentit ovat uudelleenhyödynnettäviä Vue instansseja, joilla on nimi. Niitä voidaan hyödyntää koodiin upotettuna kustomoituna HTML-elementtinä. Yhtä komponenttia voi olla upotettuna rajaton määrä (Vue.js 2018). Onkin yleistä, että sovellukset järjestellään ns. nestatuista komponenteista (nested components) koostuvaan puuhun. Nesting on ohjelmointikäsite, ja sillä lyhyesti ilmaistuna

tarkoitetaan samankaltaisten olioiden ja objektien upottamista toistensa sisään (Rouse 2005). Esimerkkitapauksessa nettisivusovellus voisi sisältää Vue-komponenteista muodostuvan sivupalkin ja navigoinnin, sekä Vue-pohjaisen ylä- ja alaotteen.



Kuva 4. Vue-instanssin refaktorointi – verkkosivu pilkotaan komponentteihin, jotka sidoksissa toisiinsa (Vue.js 2018).

Jotta komponetteja voitaisiin hyödyntää sovelluksessa, pitää ne ensin rekisteröidä. Rekisteröimällä komponentin varmistamme, että Vue tietää niiden olemassaolosta. Komponentteja voidaan rekisteröidä globaalisti tai lokaalisti. Globaalisti rekisteröityjä komponentteja voidaan hyödyntää missä tahansa Vue -instanssissa (Vue.js 2018).

4 YRITYKSEN RAJAPINNAT JA TYÖKALUT

Toimeksiantoyrityksen käytössä on lukuisia työkaluja, jotka helpottavat datan mallinnusprosessia. Tässä osiossa perehdytään yrityksen käyttämään Elastic-ohjelmistopinoon, joka koostuu kolmesta ohjelmasta: ElasticSearch, Logstash ja Kibana. Pinoon viitataan yleisesti termillä ElasticStack. Opinnäytetyössä hyödynnetään ElasticSearchia kaavioiden tietojen noutamisessa, Kibanaa käyttöliittymien rakentamiseen ja Logstashia tiedon keräämiseen ja suodattamiseen.

4.1 ElasticSearch

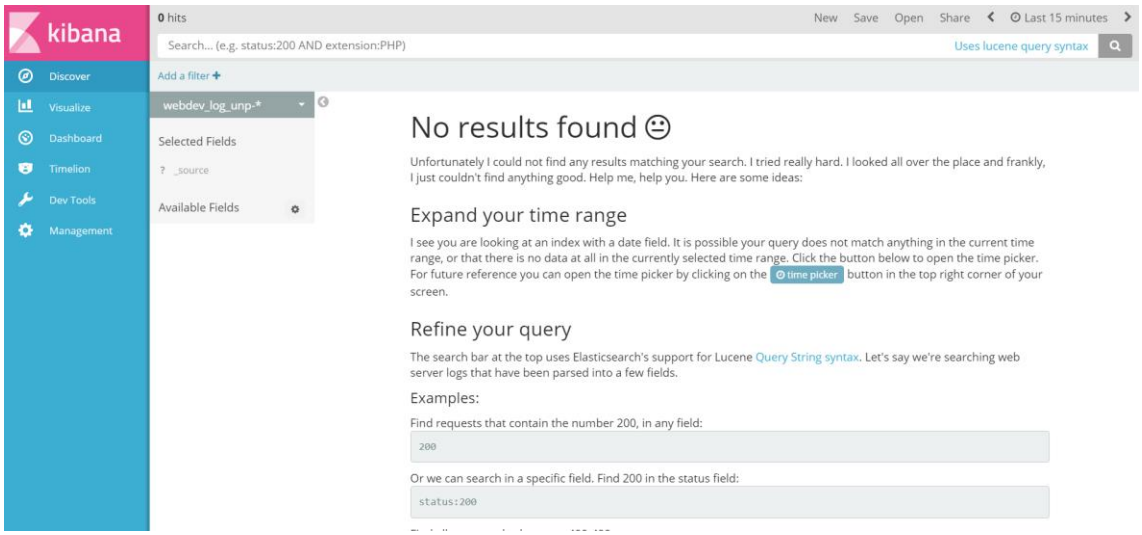
ElasticSearch on avoimen lähdekoodin hakumoottori, joka mahdollistaa datan etsimisen ja analysoinnin reaaliajassa. Avoimeen lähdekoodiin pohjautuvan ohjelmointifilosofian myötä sitä voidaan hyödyntää ilmaiseksi eri projekteissa. Sen ominaisuuksiin lukeutuu täysi tekstihaku, rakenteellinen haku ja data-analyysi. Se on hyvin skaalautuva, ja soveltuu moneen eri tarkoitukseen (Zucchetto 2018). Useat suuryritykset hyödyntävät tuotteissaan ElasticSearchia, mutta se soveltuu myös pienempiin käyttötarkoituksiin, muun muassa Wikipedian hakukoneessa hyödynnetään ElasticSearchia artikkeliehdotus ja tekstihakuominaisuuksissa. Monet pienyritykset ja verkkokaupat hyödyntävät ElasticSearchia sivujensa hakukoneissa pienemmässä mittakaavassa (Gormley & Tong 2015, 1-4).

Ohjelma on rakennettu Apache Lucenen päälle, joka on täyden tekstihaun hakukonekirjasto. Lucene itsessään on pelkkä kirjasto. Sitä voidaan hyödyntää ilman ElasticSearchia integroimalla Java-kielen avulla se omaan sovellukseen, mutta valjastamisprosessi itsessään voi olla hyvin monimutkainen ilman asianomaista osaamista (Gormley & Tong 2015, 5-7). ElasticSearch on myöskin kirjoitettu Java-kielellä, ja se hyödyntää Lucenea sisäisesti indeksoinnissa ja hakuprosesseissa, valjastaen muutoin monimutkaisen Lucenen voimat helppokäyttöiseen ElasticStack ohjelmointirajapintaan. Ohjelmaa voidaan käyttää monella eri ohjelmointikielellä (Gormey & Tong 2015, 7). Opinnäytetyössä sitä käytetään PHP & JavaScript -pohjaisessa verkkoportaalissa. ElasticStackia käytetään toimeksiantoyrityksessä hälytyksien lokihallintajärjestelmän hakumoottorina. Se poimii ja käsittelee sensoreista poimittua hälytysdataa, ja tulkkaa sen oikeaan muotoon, jolloin

sitä voidaan helposti hyödyntää visualisointialustoissa ja yrityksen sisäisissä hakukoneissa. Lokijärjestelmän tarkoitus on kerätä tietoa, jotta sitä voidaan hyödyntää verkkopohjaisesta käyttöliittymästä.

4.2 Kibana

Kibana on ilmainen avoimen lähdekoodin analytiikka- ja visualisointialusta, joka on suunniteltu toimimaan ElasticSearchin kanssa (Kuva 5). ElasticSearchin dataa voidaan hyödyntää Kibanan hallintapaneelissa. Kibana on kirjoitettu JavaScriptillä. Kibanaa ja ElasticSearchia yhdistäen voidaan luoda kaavioita hakukonedatan pohjalta. Kaavioita voidaan vapaasti yhdistellä raahaa ja pudota -toimintaperiaatteella, mikä mahdollistaa käyttöliittymien luomisen Kibanan sisällä. Kibana toimii yhteistyössä ElasticSearchin kanssa RESTful-periaatteeseen pohjautuvan ohjelmointirajapinnan avulla. Hakukonedataa voidaan hyödyntää reaaliajassa Kibanan visualisointityökaluissa (Kibana 2018).



Kuva 5. Kibana-alustan etusivunäkymä (Kibana 2018).

Kibana tukee yleisimmin käytettyjä kaaviotyyppejä: käyttäjä voi luoda lokidatan ja hakutulosten pohjalta esim. viiva- ja pylväskaavioita, jotka päivittyvät reaaliajassa uuden datan ilmestyessä lokiin. Käyttäjä voi vapaasti päättää y- ja x-akselien arvot ja muuttujat, ja

lisätä omia muuntajia tulkitsemaan ElasticSearchin palauttamia arvoja halutulla tavalla (Kibana 2018).

4.3 Logstash

Logstash on ilmainen, helposti pystytettävä avoimeen lähdekoodiin pohjautuva lokikäsittelyohjelma. Logstashin toimintaperiaate on kolmiosainen: ensin Logstash vastaanottaa syötteen lähteestä, jonka jälkeen Logstash suodattaa lokidatan käyttäjän itse määrittämän suodattimen perusteella haluttuun tiedostomuotoon. Kun suodatusprosessi on valmis, Logstash tulostaa lokidatan helposti tulkittavaan muotoon. Logstashin dataa voidaan hyödyntää ulkoisissa analysointityökaluissa kuten ElasticSearchissa. Logstash tukee monia eri lähteitä: TCP/UCP, Microsoft Windows EventLogs, Syslog ja STDIN ovat kaikki tuettuja lähdemuotoja Logstash-alustassa. Tämän lisäksi Elasticin mukana tulee lokisyöteavustusohjelma Beats, joka mahdollista lokitiedon keräämisen hajanaisemmistakin lähteistä. Logstash on kirjoitettu JRuby-ohjelmointikielellä. Logstash pyörii Java Virtual Machine -alustalla (Turnbull 2016, 5-9).

5 TOTEUTUS

Opinnäytetyön alussa perehdyttiin erilaisiin kaaviokirjastoihin ja vertailtiin mahdollisia toteutustapoja, kunnes lähtökohdat olivat selvillä. Toimeksiantoyrityksen kanssa päästiin yhteisymmärrykseen toteutussuunnitelmasta. Suunnitelman mukaan toteutuksessa tulisi käyttää Chart.js-visualisointikirjastoa jotta voitaisiin tuoda ElasticStackilla rakennetut kaaviot SOC-verkkoportaaliin. Kaavioiden tarkoitus on tuoda asiakkaille Kibanan virtaviivaisuutta ja helposti tulkittavia visualisointiratkaisuja myös verkkoportaaliin asiakkaiden nähtäväksi. Soveltavassa osiossa hyödynnetään seitsemän päivän sekuntikeskiarvokaaviota mallikappaleena ja dokumentoidaan sen tekoprosessi: samalla valottuu miten kyseisen kaavion pohjalta voidaan luoda vähäisellä vaivalla uusia kaavioita.

5.1 Vue-chart.js:n asennus ja käyttöönotto

Ennen kaavioiden suunnittelua tulee vue-chart.js-wrapper asentaa portaaliin ja alustaa. Wrapperilla tarkoitetaan funktiota, jonka sisään on upotettu muita funktioita, joita se kutsuu tarpeen tullen. Wrapperin on tarkoitus virtaviivaistaa ohjelmointiprosessia delegoiden tiettyjä toimintoja yhteen helposti käyttöönotettavaan kokonaisuuteen. Wrapperin asennus tapahtuu projektissa npm install -komennolla (Koodi 8). NPM eli Node Package Manager on JavaScript-kielellä kirjoitetuille järjestelmille suunniteltu pakettienhallintaohjelma. Se on Node.js JavaScript Runtime -ympäristön oletus riippuvuuksien- ja pakettien hallintaohjelma (Node.js 2018). Asennusprosessin valmistuttua otetaan käyttöön Chart.js-wrapper Vue-komponentissa. Kirjaston toimivuuden saa upotettua komponenttiin asianmukaisella import-lauseella (Koodi 9).

```
npm install vue-chartjs chart.js –save
```

Koodi 8. Vue-chart.js wrapperin asennuskomento.

```
import VueCharts from 'vue-chartjs'
```

Koodi 9. Import-lause wrapperin käyttöönottoon.

5.2 Kaavioiden alustaminen

Verkkoportaalin kaaviot hyödyntävät parent-to-child-toimintaperiaatetta. Kaavion visuaaliset ominaisuudet määritellään pohjakomponentissa, johon kaaviokomponetti tuo dataa ominaisuus-määritelmillä (englanniksi prop). Verkkoportaalin kaaviosivu koostuu useasta kaaviokomponentista. Jokainen kaaviokomponentti sisältää komponentin HTML:ään upotetun asianmukaisen kaavioelementin, joka piirtää kaaviokomponettiin itse kaavion (Koodi 10). Kaavioita piirtyy kaaviokomponenttiin saman verran kuin kaavioelementtejä löytyy komponentin HTML:stä.

```
<line-chart
 :width="850" v-if="loaded" :myDataSet="datasetValue"
:chartLabels="this.labels2" ></line-chart>
```

Koodi 10. Kaavioelementti koodin sisällä.

HTML-elementtiin voi suoraan tehdä määritelmiä kaavion ulkoasusta. Kaavion korkeutta voi muuttaa height-propilla, ja width-prop muuttaa kaavion leveyttä. Esimerkkielementissä myDataSet-prop vastaanottaa datasetin, jonka pohjalta kaavio piirretään. Chartlabels-prop taas määrittää datan kuvaamiseen käytettävät arvot (Koodi 11). Projektin alussa alustamme kaaviokomponenttien pohjakaaviot myöhempään käyttöön. Jokaiselle kaaviokomponentilla tulee olla pohjakomponentti, jota kaaviokomponentti lainaa toiminnassaan (Kuva 6).

Kuva 6. Toteutuksen toimintaperiaate.

Pohjakomponentin kaaviotyyppi määritellään asianmukaisella import-lauseella (Koodi 11). Lauseen muuttuja, tässä tapauksessa Bar -eli pylväs, vaihtuu riippuen minkälaisen kaavion käyttäjä haluaa pohjakomponentin piirtävän. Jos pylväskaavion sijaan haluttaisiin pohjakomponentin piirtävän sivulle esimerkiksi viivakaavio, bar-määritelmä korvattaisiin line-määritelmällä (suomeksi viiva) (Koodi 12).

```
import { Bar } from 'vue-chartjs'
```

Koodi 11. Pylväskaavion tuominen pohjakomponenttiin

Koodi 12. Pylväskaavion pohjakomponentin koodi (Visual Studio 2018).

Pohjakomponentit on nimetty komponentin edustaman kaaviotyypin perusteella, ja ne noudattavat KaavioTyyppi-nimeämiskäytäntöä. Pohjakomponentin ominaisuuksiin -eli propseihin on määritelty missä muodossa pohjakomponentti hyväksyy dataa, ja mitä dataa se hyväksyy. Type-objekti määrittelee missä muodossa data tulee ja required-objekti taas määrittää onko kenttä pakollinen (Koodi 12). Jokainen Vue-instanssi käy läpi rykelmän eri alustamisfunktioita luomisen ja käynnistyksen yhteydessä. Instanssin tarvitsee määrittää siinä hyödynnettävä tieto, koota Vue-malli, ja liittää se DOMiin -eli Document Object Modeliin, eli lyhyesti sanottuna linkittää Vue-instanssi verkkosivun HTML-

pohjaan, jotta Vue instanssi näkyy nettisivulla. Alustusmatkan varrella käydään läpi erä lifecycle -eli elinkaarimetodeja -tai koukkuja (englanniksi lifecycle hook). Lifecycle-hookit mahdollistavat toimintojen suorittamisen ajoittamisen -tai ketjuttamisen komponentin elinkaaren eri vaiheisiin. Ne sitoutuvat automaattisesti komponenttiin, jolloin voidaan edelleen hyödyntää komponentin tilaa ja menetelmiä muissa toiminnoissa. Elinkaaren menetelmät voidaan jakaa neljään luokkaan: luonti, asennus, päivitys ja tuhoaminen. Luontimenetelmiä käytetään tekemään toimia komponenttiisi ennen kuin se lisätään DOMiin. Asennettu -eli mounted lifecycle -koukku poikkeaa muista elinkaarikoukuista siinä mielessä, että sitä kutsutaan vasta silloin kun sivun luontitoimenpiteet on saatu hoidettua (Vue 2018). Kun alustustoimenpiteet on saatu tehtyä, renderChart-funktio piirtää mounted lifecycle -koukkuun upotettuna annetulla datalla kaavion DOMiin (Koodi 13).

```
mounted () {


      this.renderChart({
        labels: this.chartLabels,

        datasets: this.myDataSet,

      }, this.options)



    },
```

Koodi 13. RenderChart-funktio.

RenderChart-funktio tuodaan chart.js-kirjastosta. Se piirtää kaavion sivulle hyödyntäen sille annettua dataa. Tässä tapauksessa labels-parametri vastaanottaa dynaamisesti käyttäjän asettaman chartLabels-taulukon ja datasets-parametri myDataSet-taulukon. ChartLabels-taulukko koostuu tekstiarvoista, jotka tulevat x-akselille kuvaamaan dataa. MyDataSet-taulukko taas koostuu yksittäisistä numeroarvoista (Koodi 13).

5.3 Kojelauta

Pohjakomponenttien määrittelyn jälkeen aletaan tarkastelemaan kaaviokomponentteja: kaaviokomponentti muodostuu html-pohjasta, joka sisältää kaavioelementin, ja

kaaviossa käytettävästä datasta. Kaaviokomponentti sidotaan Vue-instanssiin, jonka myötä se piirtyy myös verkkoportaaliin tavanomaisena html-elementtinä (Koodi 14).

```
<template>
    <div>


            <template v-for="(charts, i) in charts">
                <chart component displays here on the web page>
            </template>
        </div>

    </div>
</template>
```

Koodi 14. Yksinkertaistettu esimerkki Vue-instanssin toimintaperiaatteesta.

Vue-instanssi muodostaa sivun, jolle kaaviot piirtyvät. For-loop tulostaa portaaliin niin monta kaaviota kuin sivulle on niitä aseteltu. Vue-instanssi on kojelauta, johon käyttäjä voi tiputella kaavioita mielensä mukaan. Kojelaudan toiminnallisuuden rakensi ulkoistettu turkulainen ohjelmointifirma.

5.4 Kaavion luomisprosessi

Aloitin kaavioiden luomisen määrittelemällä mistä data tulee. Tällä hetkellä yrityksen Kibanasta löytyy lukuisia kaavioita, jotka hyödyntävät logstashin suodattamaa dataa. Dataa voidaan elasticSearchin avulla lajitella ja tarkastella. Kibanan sisäiset kaaviot käyttävät sen sisällä tehtyjen ElasticSearch-queryen -eli kyselyjen tulostamaa dataa. Käyttäjä voi luoda kaavioita Kibanan tarjoamassa editorissa tai kirjoittaa omia kyselyjä kehittäjäikkunassa (Kuva 7). Kibana palauttaa kyselyn vastauksen JSONina. JSONia tullaan pilkkomaan koodissa kaavioita luodessa.

Kuva 7. Kibana-kehittäjäikkunan hakukyselyesimerkki (Kibana 2018).

Tarkkailemalla verkkoliikennettä selaimen Network-välilehdessä Kibanan sisällä saadaan käyttöön JSON, joka käsittää kyselyn vastauksen. JSON-pätkää tarkastelemalla selventyy mitä arvoja se palauttaa, ja missä muodossa data tulee (Kuva 8). Samalla selviää miltä Kibanan GET-hakupyyntö näyttää. Pienten yksityiskohtien tietäminen saattaa vaikuttaa tässä vaiheessa mitättömältä, mutta kaavioita luodessa JSONin muodon tietäminen perin kohtaisesti helpottaa elämää huomattavasti. Esimerkkikyselyssä haetaan seitsemän päivän aikaväliltä kaikki arvot yhden päivän intervallilla, jolloin arvoja piirtyy, jos niitä löytyy, seitsemän eli yksi jokaiselle viikonpäivälle. Tämän lisäksi 1-bucket-objekti ilmaisee kyselyyn liittyvän jonkinlainen aggregaatio -eli kasautuva kysely, jolla tarkoitetaan kaikkien arvojen pohjalta luotua yksittäistä kokonaislukua. Yleisimmin käytetty 1-bucket kysely on tavallisen kyselyn palauttamien arvojen pohjalta luotu keskiarvoluku (Kuva 7).

Kuva 8. Kibanan hakupyyntö (Google Chrome 2018).

Kaaviot luodaan pyynnön hakeman JSON-datan pohjalta. Kaavio tekee latautuessa hakupyynnön kibanan avulla palvelimelle haluamilla hakuehdoilla. Pyyntöön saadaan vastauksena pätkä JSONia. JSONiin on kätketty objektien ja taulukoiden sisälle arvoja, jotka heijastavat kyselyn tuloksia (Kuva 8).


```
{
  "took": 3,
  "timed_out": false,
  "_shards": {
    "total": 98,
    "successful": 98,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 0,
    "max_score": 0,
    "hits": []
  },
  "aggregations": {
    "2": {
      "value": null
    },
    "1-bucket": {
      "buckets": []
    }
  }
}
```


Kuva 9. Kibana-queryn tulos (Kibana 2018).

Jos kyselyyn ei löydy vastauksia valitulta ajanjaksolta, palauttaa aggregations-objekti arvon 0. Toinen olennainen objekti esimerkkikyselyn vastauksessa on 1-bucket-objekti: kuten aikaisemmin mainittiin, 1-bucket-objektista löytyy ylimääräisen aggregaatiofunktion tulos. Esimerkkikyselyssä 1-bucket-objekti kerää aggregations-objektin sisällä olevien arvojen pohjalta avg-bucket -eli keskiarvon kokonaislukuna 1-bucket-objektiin kokonaislukuna, jolloin sitä voidaan hyödyntää asianmukaisissa kaavioissa. Aggregations-olion tietojäsen 2 kerää kaikki kyselyn vastauksena tulevat arvot. Sillä tietojäsen 2 on esimerkkitapauksessa ei sisällä yhtäkään lukuarvoa, voidaan todeta, että hakupyyntö ei tuottanut tuloksia: 1-bucket objektin buckets-taulukko on tästä syystä myöskin tyhjä (Kuva 9).

5.5 Events-per-second 7-päivän keskiarvokaavio

5.5.1 Alustus

Uuteen kojelautaan tarvittiin toimeksiantoyrityksen puolesta muun muassa keskiarvokaavioita, jotka ilmaisevat tapahtumien kokonaismäärän helposti luettavassa muodossa asiakkaalle, ja myöskin SOC-työntekijöille. Samalla kaaviot toimivat eräänlaisena mainoksena asiakkaille toimeksiantoyrityksen tarjoamien palvelujen olennaisuudesta. Korkea tapahtumien määrä ilmaistu värikkäällä tavalla varmistaa asiakkaan palvelun tärkeydestä, ja pitää hänet ajan tasalla järjestelmän tilasta.

Työn aloittamiseksi luodaan dashboard-kansion components-alakansioon widgetEventsPerSecondWeek-niminen Vue-komponentti, joka käsittää kyseisen kaavion kaikki osa-alueet, ja toimii kaaviokomponenttina (Kuva 10).

Kuva 10. Kaaviokomponentti components-alakansiossa (Visual Studio 2018).

Kaaviokomponenttiin lisätään asianmukainen HTML-pohja kaaviolle. Aikaisemmin määritelty pohjakomponentti tulee käyttöön kaaviokomponentissa upottamalla sen kaaviokomponentin HTML-pohjaan. Se käyttäytyy komponentin sisällä normaalin HTML-elementin tavoin (Koodi 15).

```
<line-chart></line-chart>
```

Koodi 15. Viivakaavio kaaviokomponentin sisällä.

Tarvitaan jonkinlainen error-elementti, jos kaaviokutsu epäonnistuu. Error message -elementti aktivoituu, kun showError-niminen boolean palauttaa totuusarvon true (Koodi 16).

```
<div class="error-message" v-if="showError">
    {{ errorMessage }}
</div>
```

Koodi 16. Error-elementti.

Elementin sisällä oleva errorMessage Vue-objekti ilmoittaa käyttäjälle, että kaaviokutsu on epäonnistunut (Koodi 17).

```
  errorMessage: 'Fetch query failed'
```

Koodi 17. Error-viesti.

5.5.2 Tarkempi kuvaus

Keskiarviokaaviot kuvaavat sekunnin tarkkuudella, kuinka monta tapahtumaa palvelussa esiintyy. Sekuntikeskiarvot lasketaan yhden päivän kokonaisdatan pohjalta. Ensin valmistetaan asianmukainen kibana-kysely, joka noutaa kaikki tapahtumat palvelusta jonkin avainsanan perusteella. Avainsanan tulee löytyä jokaisen tapahtuman tiedoista: tästä syystä kyselyssä hyödynnetään timestamp-avainsanaa, sillä jokainen tapahtuma sisältää aikaleiman.

5.5.3 Päivämäärien haku

Palvelin ja ElasticSearch palauttavat aikaleimat millisekunteissa. Samoin tavoin Kibanakyselyissä palvelin hyväksyy ainoastaan millisekunteissa olevat aikaleimat, jonka takia kaaviokyselyt tulee ensin muuttaa sopiviksi. Tämä onnistuu koodissa parhaiten moment.js-kirjastolla. Moment.js on helppokäyttöinen JavaScript-kirjasto päivämäärien manipuloimiseen ja generointiin (Moment.js 2018).

Kaaviokyselyn tulee hakea seitsemän päivän aikahaarukalla kaikki arvot. Viikon takaisen tarkan päivämäärän saa luotua moment.js-kirjastolla dynaamisesti aina kun kaaviokysely tehdään palvelimelle. Startdate on kaaviokomponentin data-objekti, joka sisältää viikon takaisen päivämäärän. Siinä vähennetään tämän hetkisen ajankohdan päivämäärästä seitsemän päivää substract-funktiolla, jolloin moment.js palauttaa viikontakaisen date-arvon formaatissa MM-DD-YYYY (Koodi 18).

```
startdate: moment().subtract(7, 'd').format("MM/DD/YYYY")
```

Koodi 18. Viikon takaisen päivämäärän generointi mm/dd/yyyy-formaatissa.

Date-arvo pitää jollain keinolla saada sellaiseen muotoon, että backend ja ElasticStack osaavat tulkita sitä. Lisäämme kaaviokomponenttiin computed-propertyn, joka muuntaa startdate-objektin mm/dd/yy-muodossa olevan päivämääräpätkän millisekunteiksi. Computed propertyillä tarkoitetaan Vue propertyjä, joihin on sidottu jonkinlainen toiminto (Koodi 19).

```
periodStart() {
                return moment(this.startdate, "MM/DD/YYYY").valueOf()
            },
```

Koodi 19. PeriodStart computed property.

PeriodStart niminen computed property palauttaa startdate-objektin pohjalta millisekunteissa siihen syötetyn aikaleiman. ValueOf-funktio muuntaa päivämäärän millisekunttilukuarvoksi (Koodi 19). Kun kyselyn alkupäivämäärä on saatu sopivaan muotoon, on aika luoda aikahaarukalle millisekunteissa oleva loppupäivämäärä. Loppupäivämäärän saa generoitua käyttämällä JavaScriptin date-funktiota tämänhetkisen ajankohdan aikaleiman luomiseen, joka sitten muutetaan moment.js-kirjaston apufunktioita hyödyntäen oikeaan muotoon (Koodi 20).

```
                enddate: moment(new Date(), "MM/DD/YYYY").valueOf(),
```

Koodi 20. Enddate-objekti käsittää kyselyn loppupäivämäärän.

Kun kaaviokyselyn aikaleimaparametrit on generoitu, on aika alkaa tarkastelemaan varsinaista kaaviokyselytoteutusta.

5.5.4 Kaaviokyselyn backend-valmius

Kaaviokyselyt hyödyntävät ElasticStackia datan noutamisessa, vaikka kyselyt eivät varsinaisesti suoraan ole yhteydessä Kibanaan. Kaaviokysely tehdään järjestelmässä ennalta määriteltyyn osoitteeseen, joka toimii Kibanan haun tavoin kyselyalustana. Osoitteeseen /chartdata tehty GET-pyyntö palauttaa asianmukaiset lukuarvot, joita voidaan hyödyntää kaavioissa. Kyselyn backendillä hyödynnetään yksinkertaista switch-lausetta: switch-lause on JavaScript-konsepti, jossa riippuen siitä mikä avainsana- tai arvo on kyseessä, eri koodipätkä suoritetaan. Switch-lause sisältää valitun määrän eri kaaviokyselyille luotuja tapauksia, jotka toimivat avainsanaperiaatteella. Jokainen tapaus sisältää kovakoodatun, kyseistä kaaviota koskevan Elastic-kyselyn. Riippuen siitä mikä avainsana GET-pyyntöön on liitetty query stringinä, kaavio tekee avainsanan tunnistaessa avainsanan oman tapauksen pohjalta GET-pyynnön hyödyntäen tapaukseen kovakoodattua Kibana kyselylausetta. Kyselylauseessa esiintyy äsken luotu dynaamisesti muuttuvat, käyttäjän itse määrittelemät aikahaarukkaparametrit php-muuttujina epoch_days ja epoch. Kyselylauseke ei poikkea muutoin normaalista Kibanan sisällä tehdystä

kyselylausekkeesta. Tapauksen määrittelemisen myötä backendistä löytyy valmius halutun kaavion luomiseen (Koodi 21).

```
            case "example_last_7_days":
                $query = '{"index":'.$RAND.',"ignore_unavaila-
ble":true,"preference":'.$preference.'}
          {"query":{"bool":{"must":[{"query_string":{"query":"*","ana-
lyze_wild-
card":true}},{"range":{"@timestamp":{"gte":'.$epoch_days.',"lte":'.$epoch
.',"format":"epoch_mil-
lis"}}}],"must_not":[]'.$RAND'}},"size":0,"_source":{"ex-
cludes":[]},"aggs":{"2":{"date_histogram":{"field":"@timestamp","inter-
val":"'.$interval.'","time_zone":"Europe/Hel-
sinki","min_doc_count":1},"aggs":{"3":{"terms":{"field":"_type","size":5,
"order":{"_count":"desc"}}}}}}}
            ';
```

Koodi 21. Tapaus-esimerkki (eng. case).

Kun kaaviokyselyyn on tehty backend-valmius, voidaan tarkastella itse GET-pyynnön yksityiskohtia.

5.5.5 Kaaviokyselyn GET-pyyntö

Kuten viime kappaleessa mainittiin, GET-pyyntö tehdään osoitteeseen /chartdata, jolloin se palauttaa JSON-muodossa olennaiset arvot sisältävän vastauksen. Hyödynnetään GET-pyynnön tekemisessä axios-kirjastoa. Axios on selaimelle ja node.js-rungolle luotu promise -pohjainen HTTP-kirjasto, joka virtaviivaistaa HTTP-pyyntöjen tekemisprosessia verkkosovelluksessa (Axios Github 2018). Promise on JavaScript-käsite, jossa funktiot suoritetaan asynkronisesti pyyntöketjussa: funktio suoritetaan, jos sitä edeltävä pyyntö valmistui onnistuneesti. GET-pyyntö muodostuu axios.get-lauseesta, jossa axiosin get-funktio ottaa argumenttina osoitteen eli URLin johon pyyntö tehdään (Koodi 22).

```
axios.get(`/chartdata?days=8&startepo=${this.period-
Start}&endepo=${this.enddate}&que=example_last_7_days&index=*`)
.then(response => {
    this.doc_count = response.data.responses[0].aggregations['1-buck-
et'].buckets.map(documents => documents.doc_count)
    this.labels = response.data.responses[0].aggregations['1-buck-
et'].buckets.map(documents => documents.key)
}
```

Koodi 22. GET-kutsu & takaisinkutsuoperaatio.

URL sisältää kaavioon liittyviä query-stringejä, jotka toimivat pyynnön parametreina. Query-stringit alkavat kysymysmerkillä. Startepo-query-string käsittää kaaviokyselyn aikahaarukan alkamispäivän. Annetaan sille parametrina aikaisemmin määritelty PeriodStart- computed property. Samalla tavalla toimitaan endepo-query-stringin kanssa, joka käsittää aikahaarukan päättymispäivämäärän. Tämän lisäksi que-query-string vastaanottaa kaaviotyypin avainsanan, jonka backend heijastaa sitä vastaavaan tapaukseen, ja tekee tapauslausekkeeseen kovakoodattua kyselydataa hyödyntäen kaaviokyselyn. Kun pyyntö on suoritettu onnistuneesti, then-metodi palauttaa promisen, joka sisältää argumenttina pyynnön vastauksena palauttaman datan response-muodossa. Vastausta käsitellään then-metodin sisällä (Koodi 22).

Tarkastellaan palvelimen palauttamaa vastausta. Network -välilehdestä poimittu vastaus sisältää suuren objektin, joka käsittää GET-pyynnön vastauksen (Kuva 11).

Kuva 11. Kyselyn palauttama JSON-vastaus (Google Chrome 2018).

Vastausta pilkkomalla kaaviot saavat dataa oikeisiin parametreihin. JSONin tarkastelusta saa selville, että ElasticStack palauttaa aikaleimat millisekunteina key-objektin sisällä, ja varsinaiset kaavion numeroarvot doc_count-objektin sisällä. Key ja doc_count -objektit löytyvät kummatkin buckets-taulukosta, joka taas on responses-objektin sisällä. Response-objekti on data-objektin sisällä, joka taas kuuluu response-objektiin. Ketjuttamalla nämä objektit pääsevät käsiksi haluttuihin doc_count ja key -objekteihin. Doc_count ja key -objekteille määritellään asianmukaiset muuttujat. Luomalla taulukon doc_count-arvoista map-funktion avulla kaavio saa käyttöön datan oikeassa muodossa, samoin key-arvojen kanssa. This.doc_count-muuttuja käsittää siis taulukon, joka sisältää map-funktion avulla tehdyn poiminnan seurauksena kaavion numeroarvot. Muuttujassa hyödynnetty funktio poimii JSON-vastauksesta doc_count-arvot, ja tekee arvoista taulukon -eli arrayn. This.labels-muuttuja toimii samalla periaatteella, ja käsittää taulukon kaikista aikaleima -eli key-arvoista. This.labels-muuttuja on taulukko, joka käsittää x-akselille piirtyvät aikaleima-arvot (Koodi 23). Tällä hetkellä aikaleimat ovat alkuperäisessä millisekunttimuodossa.

```
.then(response => {
                        this.doc_count = response.data.responses[0].aggre-
gations['1-bucket'].buckets.map(documents => documents.doc_count)
                         this.labels = response.data.responses[0].aggrega-
tions['1-bucket'].buckets.map(documents => documents.key)
```

Koodi 23. Kaavion määrittävät taulukkomuuttujat doc_count ja labels.

This.labels-taulukon aikaleima-arvot tulee muuttaa millisekunneista helposti tulkittavaan YY-MM-DD-muotoon. Näin toimimalla varmistetaan, että kaavio on selkeä ja helposti ymmärrettävä: on päivänselvää, että päivämäärää on vuosi-kuukausi-päivä-muodossa helpompi lukea kuin että päivämäärä olisi millisekunneissa. Muuttamisprosessia varten luodaan uusi funktio nimeltä getFormattedTimeStamp. GetFormattedTimeStamp muuttaa sille argumenttina annetun numeroarvon DD-MM-YY-muotoon. Funktiossa hyödynnetään moment.js-kirjastoa. Uusi taulukko tulee käsittämään this.labels-taulukon arvot YY-MM-DD-muodossa (Koodi 24).

```
function getFormattedTimeStamp(x) {
     return moment.unix(x / 1000).format("DD MMM YYYY")

   }
```

Koodi 24. GetFormattedTimeStamp-funktio, joka paluttaa syötetyn millisekuntiarvon DD-MM-YY-muodossa.

Tähän tarkoitukseen määritellään uusi muuttuja labelsFormatted. LabelsFormatted-muuttujassa map-funktio kutsuu getFormattedTimeStamp-funktiota jokaisessa this.labels-taulukon elementissä, ja luo getFormattedTimeStamp-funktion läpi käyneistä elementeistä uuden taulukon. LabelsFormatted-muuttuja sisältää siis this.labels-taulukon millisekunti muodossa olleet numeroarvot YY-MM-DD-muodossa (Koodi 25).

```
 this.labelsFormatted = this.labels.map(x => getFormattedTimeStamp(x))
```

Koodi 25. LabelsFormatted-taulukko, joka käsittää aikaleimat DD-YY-MM.

This.labelsFormatted-taulukon luomisen jälkeen keskitetään huomio doc_count-taulukkoon. Tällä hetkellä taulukko sisältää päivittäiset kokonaistapahtumamäärät viikon ajalta. Doc_count-taulukko ei siis toimi nykymuodossa sillä kaavion tulee näyttää tapahtumien määrän sekuntikeskiarvo, events-per-second, kokonaistapahtumamäärän sijaan. Sen sijaan että muokkaisimme olemassa olevaa kaaviokutsua, keskiarvon voi poimia suhteellisen vaivattomasti tällä hetkellä saatavilla olevista päivittäisistä kokonaistapahtumamääräarvoista.

Sekuntikeskiarvon saa jakamalla yhden päivän kokonaistapahtumamäärän 86400:lla. Jakolaskun vastaukset pyöristetään myöskin kokonaisluvuiksi. Luodaan funktio GetSecondAverageFromDay koko jakoprosessia varten. Pyöristämisessä käytetään JavaScriptin ToFixed-funktiota. GetSecondAverageFromDay noudattaa samaa toimintaperiaatetta kuin getFormattedTimeStamp-funktio. Se ottaa argumenttina yhden numeroarvon, ja palauttaa uuden numeroarvon. Kuten aikaisemmat funktiot, se myöskin toimii parhaiten yhteistyössä map-funktion kanssa (Koodi 26).

```
function getSecondAverageFromDay(x) {
                            return (x / 86400).toFixed()
                        }
```

Koodi 26. GetSecondAverageaFromDay-funktio palauttaa sekuntikeskiarvon yhden päivän datan pohjalta.

Uutta taulukkoa varten luodaan uusi muuttuja doc_countFixed. Sen sisäinen funktio toimii lähes täsmälleen samalla periaattella kuin labelsFormatted-muuttujassa esiintynyt funktio. Funktion lopputuloksena doc_countFixed käsittää yhden taulukon. Taulukko sisältää seitsemän päivän ajalta yhden päivän intervallilla tapahtumamäärien sekuntikeskiarvot (Koodi 27).

```
 this.doc_countFixed = this.doc_count.map(x => getSecondAverage-
FromDay(x))
```

Koodi 27. Doc_countFixed-taulukko, joka käsittää sekuntikeskiarvo -lukuarvot.

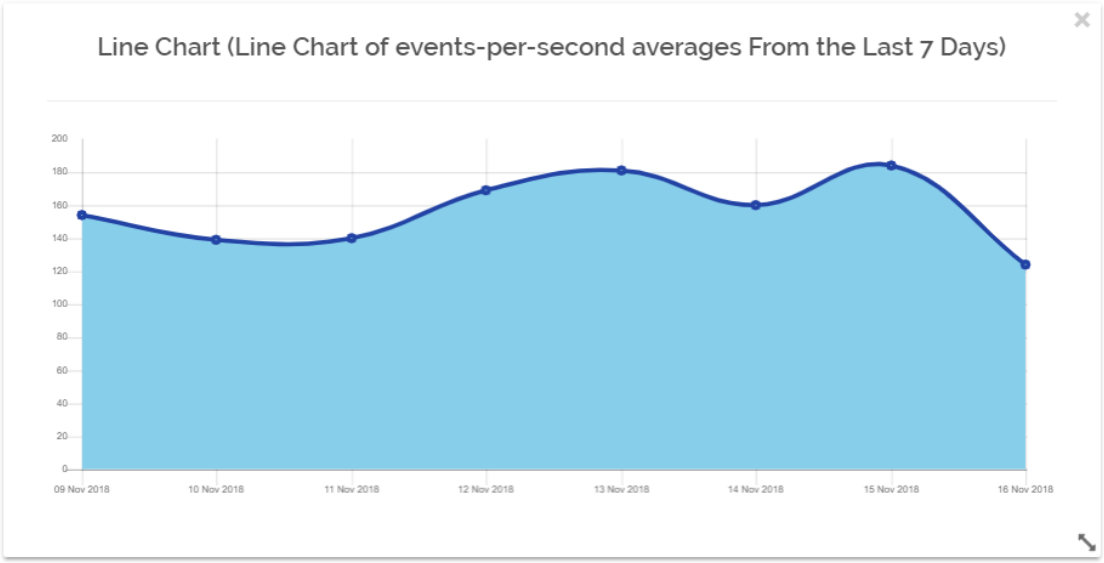
labelsFormatted ja doc_countFixed -taulukot syötetään argumentteina kaaviokomponentin kaavioelementtiin. Esimerkkikaavioelementtin chartLabels -ja myDateArray parametrit vastaanottavat arvot, ja luovat niiden pohjalta asianmukaisen kaavion verkkoportaaliin (Koodi 28).

```
 <line-chart ref="chart" :width="950" :height="260": chartLabels="la-
belsFormatted" :myDataArray="doc_countFixed"></line-chart>
```

Koodi 28. Kaavio-elementti, johon on syötetty äsken luodut taulukot argumentteina.

Events-per-second averages from the last 7 days -kaavio on yksinkertainen viivakaavio, joka käsittää yhden viivan. Viiva ilmaisee järjestelmään tulevan datan määrän sekuntikeskiarvon. X-akseli näyttää aikaleiman, joka kertoo miltä päivältä kyseinen keskiarvo on (Kuva 12). Kaavio toimii pohjana tulevaisuudessa käytettäville kaavioille. Muuttamalla

aikahaarukka-arvoja GET-pyynnossä saamme viikon aikahaarukan sijaan esim. kuukauden ajalta dataa. Myöskin kaaviotyyppiä saa helposti vaihdettua haluttuun muotoon. Viivakaaviosta saadaan piirakkakaavio vaihtamalla kaavio-elementin nimeä: pie-chart piirtää sivulle viivakaavion sijaan piirakkakaavion samaa dataa hyödyntäen.



Kuva 12. Viivakaavio events-per-second keskiarvoista seitsemän päivän ajalta (Chart.js 2018).

5.5.6 Viikkokaavion uudelleenkäytettävyys

Kuten aikaisemmin mainittiin, viikkokaavio toimii pohjana muille kaavioille. Sitä hyödyntäen saa helposti tehtyä uusia kaavioita aikahaarukasta riippumatta. Myöskin kaaviotyyppiä voidaan muuttaa muuttamalla HTML-elementin nimeä, ja korvaamalla import-lausekkeesta line esimerkiksi bar-määritelmällä (Koodi 29).

```
import BarChart from './BarChart'
```

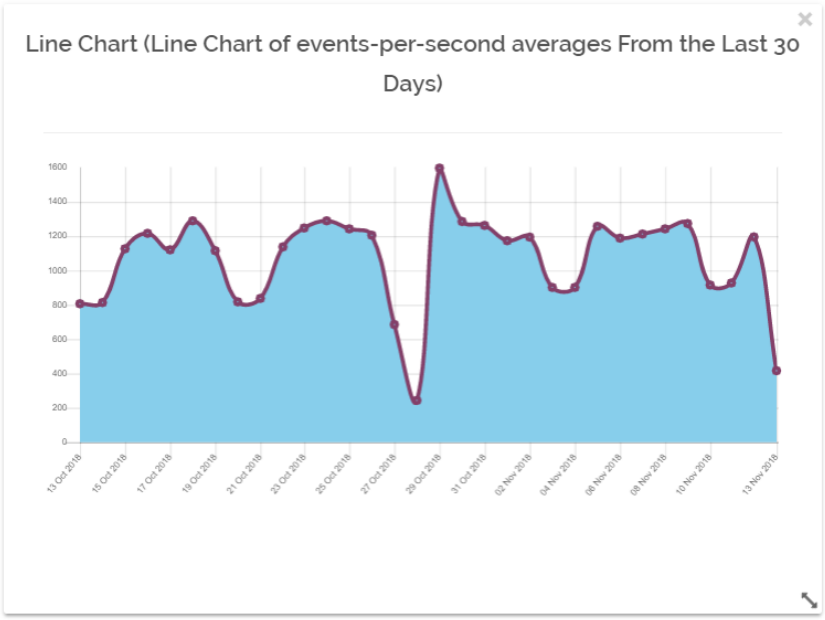
Koodi 29. Barchart-import-lause, joka tuo pylväskaavion komponenttiin.

Viikkokaavion pohjalta saa luotua kuukausikaavion melko vähäisellä vaivalla. Kuukausikaavio toimii lähes täsmälleen samalla tavalla kuin viikkokaavio. Ainoa ero näiden kahden välillä on startdate-muuttujassa käytetyn substract-funktion arvo: sen sijaan että vähennettäisiin vain viikko, nykyisestä päivämäärästä vähennetään kuukausi (Koodi 30).

```
 startdate: moment().subtract(1, 'months').format("MM/DD/YYYY")
```

Koodi 30. Kuukausikaavion aikahaarukkamuuttuja.

Kaavio piirtää viikon sijaan sekuntikeskiarvot kuukauden ajalta. Kuukausikaavio toimii parhaiten viivakaaviopohjalla kuten viikkokaavio, sillä kun aikaleima heijastetaan x-akselilla päiväarvot erottuvat parhaiten (Kuva 13).



Kuva 13. Kuukausikaavio (Chart.js 2018).

Kummassakin kaaviossa hyödynnetään samoja ulkoasuun liittyviä arvoja. Chart.js mahdollistaa kaavioiden ulkoasun muokkaamisen eri komennoilla. BorderColor muuttaa viivakaavioissa viivan väriä. Kummassakin kaaviossa viivakaavion väri generoidaan apufunktiolla. Apufunktio luo kolme satunnaisesti generoitua numeroa yhden ja 255:n väliltä, ja syöttää ne RGB-väriarvoina argumenttilauseen sisään. RGB on värimalli, jossa luodaan eri värejä yhdistelemällä sinisen, vihreän ja punaisen väriarvoja. Ensimmäinen luku kertoo punaisen arvon, toinen vihreän arvon ja kolmas sinisen arvon. 0 on minimiarvo ja 255 on maksimiarvo, kun arvo ilmoitetaan numeroina (Salakapakka 2012). Numero generoidaan Math.random-funktiolla. Math.floor tekee numerosta kokonaisluvun (Koodi 31).

```
borderColor: "rgb(" + Math.floor(Math.random() * 255) + "," +
Math.floor(Math.random() * 255) + "," + Math.floor(Math.random() * 255) +
")",
```

Koodi 31. Värin generoiva muuttuja.

BackgroundColor-parametri vastaanottaa kaavion täyttövärin. Kaavion täyttämisessä käytetään vaaleansinistä väriä, jonka hex-arvo on #87CEEB (Koodi 32).

```
        backgroundColor: "#87CEEB"
```
Koodi 32. Taustavärimuuttuja.

Viivakaavion voi vaihtoehtoisesti jättää täyttämättä niin että täyttöalue ei olisi väritetty. Fill on boolean, joka määrittää väritetäänkö viivakaavion alapuolella oleva alue vai ei. Esimerkkikaavioissa täyttöalue väritetään, joten fill saa arvon true (Koodi 33).

```
fill: true
```
Koodi 33. Fill-boolean.

5.5.7 Piirakka- ja pylväskaavio

Piirakkakaavioissa import-lauseessa tuomisen kohteena on viivakaavio-pohjakomponentin sijaan piirakkakaavio-pohjakomponentti (Koodi 34).

```
  Import PieChart from './PieChart'
```
Koodi 34. Piirakkakaavion pohjakomponentin import-lause.

Piirakka- ja pylväskaaviot toimivat täsmälleen samalla periaatteella kuin viivakaaviot, jonka takia en kokenut niiden tekoprosessin tarkempaa dokumentointia tarpeelliseksi. Asianmukaisten import-lauseiden liittämisen jälkeen HTML-elementti upotetaan kaaviokomponentin koodiin samalla tavalla kuin viivakaavioesimerkissä (Koodi 35).

```
          <pie-chart
  :height="200":chart-data="doc_count" :chart-labels="labels" :getCol-
ors="getColors"></pie-chart>
```
Koodi 35. Piirakkakaavio.

Pylväskaavioissa sama periaate pätee. Pylväskaavion HTML-elementissä toimii samat parametrit kuin aikaisemmissa esimerkeissä. Niissä hyödynnetään samalla tavalla doc_count- ja labels-pohjaisia taulukkoja (Koodi 36).

```
<bar-chart
  :height="200":chart-data="doc_count" :chart-labels="labels" :getCol-
ors="getColors"></bar-chart>
```
Koodi 36. Pylväskaavio.

6 YHTEENVETO

Opinnäytetyön ja toimeksiannon tavoitteena oli korvata yrityksen järjestelmän karkeat vanhentuneet Google Charts -visualisointiratkaisut uusilla, helposti tulkittavissa olevilla kaavioilla. Kaaviot päivitettiin hyödyntämään Vue.js-runkoa ja Chart.js-kaaviokirjastoa. Verkkoportaalin sivut uudistettiin toteutuksen ratkaisuilla. Vanhojen kaavioiden tilalle tehtiin uusia ja uudet ratkaisut tuli käyttöön verkkoportaalissa.

Projektin alussa tutkittiin eri kaaviokirjastoja ja vertailtiin mahdollisia ratkaisuja, kunnes toimeksiantajan kanssa päästiin yhteisymmärrykseen toteutustavasta. Aikaisempaa kokemusta Vuesta tai Chart.js-kirjastosta ei ollut, joten ymmärrettävästi opinnäytetyöprosessin alussa piti hankkia vaadittava osaaminen.

Uudet kaaviot ovat värikkäitä ja visuaalisesti miellyttäviä, sekä myöskin responsiivisia Chart.js-kirjaston käyttöönoton myötä. Front- ja backend onnistuttiin tämän lisäksi irrottamaan kaavioissa toisistaan: kaaviot käsittävät vain omat niille kuuluvat Vue-komponentit, eikä Vue-komponenttien sisällä sekoiteta PHP:ta ja JavaScriptiä keskenään.

Projektin ajankohta oli hektinen, sillä aikaraja koulun puolesta opinnäytetyölle oli marraskuun loppu. Kiireisen aikataulun myötä työn toteuttamiseen ei jäänyt hirveästi aikaa. Tästä syystä koodi ja konseptien visualisointi saattavat ajoittain olla hieman epäselviä. Aloitusajankohta oli syyskuun puoliväli, ja visualisointiratkaisut saatiin valmiiksi noin marraskuun puolivälissä.

Loppuratkaisu vastasi odotuksia. Kaaviokirjastoratkaisut tulivat viralliseen käyttöön yrityksen järjestelmissä, ja jatkokehitysehdotuksia esitetään aktiivisesti. Muun muassa kaavion interaktiivisuutta on pohdittu. Kaaviokirjaston hyvä käytettävyys mahdollistaa uusien ratkaisujen syöttämisen järjestelmään, ja toimeksiantoyrityksen muutkin verkkoratkaisut uusitaan tarvittaessa.

LÄHTEET

Advanced Kittenry 2014. Web-sovelluksien toiminta. Advanced Kittenry – tietokantasovellusohjeet. Viitattu 17.10.2018 https://advancedkittenry.github.io/web-sovelluksista.html

Baaj A. 2017. Compare the best JavaScript Chart libraries. Medium. Viitattu 20.10.2018 https://blog.sicara.com/compare-best-javascript-chart-libraries-2017-89fbe8cb112d

Cenno 2014. Mikä on API ja miksi se on SaaS ohjelmistossa niin tärkeä?. Cenno Software Oy. Viitattu 11.11.2018 https://www.cennoapp.com/blog/2014/05/mika-on-api-ja-miksi-se-on-saas-ohjelmistossa-niin-tarkea/

Contribyte 2018. If it works, fix it! – Refaktorointi kannattaa aina. Contribyte. Viitattu 25.10.2018 https://contribyte.fi/2018/05/08/if-it-works-fix-it/

Chajed, S. 2015. Learning ELK Stack. Birmingham: Packt Publishing Ltd. Viitattu 20.10.2018

Combitech Finland 2017. Mikä on SOC ja miksi sellainen tarvitaan?. Medium. Viitattu 16.10.2018 https://medium.com/@combitech/mik%C3%A4-on-soc-ja-miksi-sellainen-tarvitaan-a0df93609118

Dugas C. 2012. jQuery Spaghetti! Tips and tricks for cleaner code. Cakemail. Viitattu 20.11.2018 https://www.cakemail.com/blog/jquery-spaghetti-tips-and-tricks-for-cleaner-code/

Fowler M. 2018. Refactoring: Improving the design of existing code. Addison Wesley Professional. Viitattu 20.11.2018

Freeman J. What is JSON? JavaScript Object Notation explained. InfoWorld. Viitattu 5.12.2018 https://www.infoworld.com/article/3222851/javascript/what-is-json-javascript-object-notation-explained.html

JSON 2018. Introducing JSON. Viitattu 05.11.2018 https://www.json.org/

Juszczak J. 2018. vue-charjs – Easy and beautiful charts with Chart.js and Vue.js. Viitattu 20.11.2018 https://vue-chartjs.org/

Kibana 2018. Introduction. Kibana UserGuide [6.4]. Viitattu 30.10.2018 https://www.elastic.co/guide/en/kibana/current/introduction.html

Kivisaari T. 2016. API:t ovat modernin integraatiostrategian ydin. Digi arjessa. Viitattu 11.11.2018 http://blog.digia.com/rest-api

Korpela J. 2009. Web-julkaisemisen opas - HTTP. Datatekniikka ja viestintä. Viitattu 20.10.2018 http://jkorpela.fi/webjulk/

Lehto, T. 2017. Tietoturva-ala vaatii suosimaan suomalaista - ”tällä alalla työpaikat eivät siirry halpatyömaihin”. Tekniikka & Talous. Viitattu 16.10.2018 https://www.tekniikkatalous.fi/tekniikka/ict/tietoturva-ala-vaatii-suosimaan-suomalaista-talla-alalla-tyopaikat-eivat-siirry-halpaty-omaihin-6681081

Lietsala K. 2009. PHP:n koodispagetti aiheuttaa turhia kuluja. Gemilo. Viitattu 20.11.2018 http://www.gemilo.com/yritysblogi/phpn-koodispagetti-aiheuttaa-turhia-kuluja

Mikkonen J 2017. Rest on nettipalveluiden yhteinen kieli. Tivi. Viitattu 17.10.2018 https://www.tivi.fi/Kaikki_uutiset/rest-on-nettipalveluiden-yhteinen-kieli-6652501

Mociun W. 2015. Big Names Using React.js. React Kung Fu. Viitattu 16.10.2018 https://reactkungfu.com/2015/07/big-names-using-react-js/

Moment.js 2018. Introduction. Moment.js. Viitattu 13.11.2018 https://momentjs.com/

Node.js 2018. About Node.js. Linux Foundation. Viitattu 1.11.2018 https://nodejs.org/en/

Niiranen E. 2017. Vauhtia fronttiin Vuella. Codento. Viitattu 17.10.2018 https://www.codento.fi/2018/03/vauhtia-fronttiin-vuella/

Pizka M. 2004. Straightening Spaghetti-code with Refactoring?. Nevada: ResearchGate. Viitattu 15.11.2018 https://www.researchgate.net/publication/221610982_Straightening_Spaghetti-Code_with_Refactoring

Rouse M. 2005. Nested-definition - “what is nesting?”. TechTarget. Viitattu 30.10.2018 https://whatis.techtarget.com/definition/nested

Turnbull, J. 2016. The Logstash Book. Brooklyn: James Turnbull. Viitattu 30.10.2018

TutorialPoint 2018. Google Charts – Overview. Viitattu 2.12.2018 https://www.tutorialspoint.com/googlecharts/googlecharts_overview.htm

Salakapakka 2012. HTML- ja CSS-opas. Apek. Viitattu 20.11.2018 http://salakapakka.net/oppaat/html-ja-css-opas/html_opas_varit.php

Visual Studio Code 2018. Code editing redefined. Microsoft. Viitattu 15.11.2018 https://code.visualstudio.com/

Vue.js 2018. What is Vue.js? Vue.js. Viitattu 24.10.2018 https://vuejs.org/v2/guide/

W3 org. 2018. HTML Introduction. W3. Viitattu 16.10.2018 https://www.w3.org/html/

Ymon 2017. SIEM-järjestelmän monet hyödyt. Ymon. Viitattu 14.11.2018 https://blogi.ymon.fi/fi/siem-jarjestelman-monet-hyodyt

Zucchetto J. 2018. ElasticSearch: Getting Started. Elastic. Viitattu 23.10.2018 https://www.elastic.co/webinars/getting-started-elasticsearch?elektra=home&storm=sub