SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

KIHO RAPORTOINTINÄKYMÄ

TEKIJÄ: Ilkka Rytkönen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä Ilkka Rytkönen	
Työn nimi Kiho Raportointinäkymä	
Päiväys 5.6.2019	Sivumäärä/Liitteet 34
Ohjaajat Lehtori Sami Lahti ja lehtori Keijo Kuosmanen	
Toimeksiantaja Kiho / Mastercom Oy	

Tiivistelmä

Kiho Cloud Platform on Kihon kehittämä liiketoiminta-alusta, jonka tarkoituksena on digitalisoida yrityksen manuaalinen työ. Käsin paperille tehtävät kirjaukset ovat virhealttiita ja paperit voivat hävitä. Papereille tehdyt kirjaukset joudutaan syöttämään käsin erilaisiin sähköisiin järjestelmiin ja tässä piilee virheen mahdollisuus. Tämä on myös hidasta, minkä vuoksi laskutus laahaa perässä. Hävinnyt paperi tarkoittaa pahimmassa tapauksessa sitä, että työ jää laskuttamatta.

Kiho Cloud Platform mahdollistaa yrityksen tietojen kirjauksen digitaalisesti kaikissa vaiheissa aina työaikaleimauksista kaluston seurantaan sekä kirjanpitoon ja palkanlaskentaan. Näin tiedot ovat käytettävissä reaaliajassa, mikä mahdollistaa paremmat päätökset. Laskutus ei myöskään laahaa perässä.

Järjestelmä on tarjonnut raportointia näistä toiminnoista, mutta raportit ovat olleet tähän saakka hajallaan. Tämän vuoksi oli tarpeellista kehittää raportointinäkymä, josta Kihon asiakkaat näkisivät yhdellä vilkaisulla tilaamiensa palveluiden tilanteen. Raportointinäkymässä piti olla tietoa telematiikasta, tehtävistä, työajasta, laskutuksesta ja kalustoluettelosta.

Työ lähti liikkeelle suunnitelupalaverilla, jossa päätimme toteutettavat asiat. Sain suhteellisen vapaat kädet toteuttaa työ haluamallani tavalla. Nykyinen web-sovelluspohja asetti toki omat rajoitteensa ja suuntaviivat suunnittelulle. Raportointinäkymä piti toteuttaa Vue.js:lla, joka on JavaScript-pohjainen ohjelmistokehys. Vue.js:n käyttöä helpotti se, että olin käyttänyt sitä jo aikaisemmin Kiholla työharjoittelussa ollessani. Samalla Kihon järjestelmä oli tullut jo osittain tutuksi, eli ihan nollasta ei tarvinnut lähteä liikkeelle.

Työ oli paljon Kiho Cloud Platform -liiketoiminta-alustan rajapintojen tarjoamien tietojen selvittämistä ja sen tiedon jatkojalostamista raportointikelpoiseen muotoon. Samalla jouduin opettelemaan Vue.js:n ja JavaScriptin uudempia ominaisuuksia, jotka helpottivat toteutusta. Käytin työssä myös kolmannen osapuolen JavaScript-kirjastoja, joiden avulla kehitystyö nopeutui.

Lopputulos oli onnistunut ja lähes valmis tuotantokäyttöön. Opinnäytetyö vaatii kuitenkin jatkokehitystä, jotta Kiho Raportointinäkymä saadaan latautumaan riittävän nopeasti. Tämä vaatii muutoksia tietokantaan sekä rajapintoihin, joten se jätettiin tämän opinnäytetyön ulkopuolelle.

Avainsanat
JavaScript, Vue.js, ohjelmistokehys, web-ohjelmointi

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author Ilkka Rytkönen			
Title of Thesis Kiho Dashboard			
Date	5 June 2019	Pages/Appendices	34
Supervisors Mr. Sami Lahti, Senior Lecturer, Mr. Keijo Kuosmanen, Senior Lecturer			
Client Organisation Kiho / Mastercom Oy			

Abstract

Kiho Cloud Platform is a service that digitalizes and advances information about people, locations and vehicles. It enables real-time information optimizing the business to be more productive. (Kiho, s.a.)

There have been some reporting pages in the Kiho Cloud Platform, but they have been scattered. Therefore, a new reporting view including information about telematics, tasks, worktime, invoicing and asset guard in one place was needed. The reporting view was named Kiho Dashboard.

There were some technical limitations due to the existing system. Vue.js framework was needed to build the front-end. The author was already familiar with the working procedures thanks to his earlier internship at Kiho.

First, it was necessary to find out how to get data from Kiho API endpoints. After that it was studied how to call each endpoint to get the data in the right form. Huge data parsing was needed for the chart views.

The result was satisfactory even if the dashboard loading speed was slow. The current API endpoints are not designed for real time data analysis. For further development it will be necessary to build a separate API endpoint just for the Kiho Dashboard.

Keywords
JavaScript, Vue.js, framework, front-end

ESIPUHE

Haluan kiittää Kihon ohjelmistovastaavaa Henri Hartikaista, joka järjesti minulle tämän mahdollisuuden opinnäytetyön tekemiseen. Opinnäytetyön tekeminen oli haastava ja antoisa projekti, jossa opin paljon uusia käytännönläheisiä toimintatapoja sekö ongelmanratkaisukykyä.

Kiitokset lähtevät myös Verkkokauppa.comiin, josta sain opintovapaan näiden opintojen suorittamiseksi. Opintojen aloitus vaati myös monien muiden asioiden onnistumisen, kuten talon myynnin Iissä ja muuton Kuopioon. Kaikki asiat menivät kuitenkin kuin elokuvissa ja sain opintojen pohjalle vielä sopivasti hyväksilukuja vanhoista opinnoistani.

Haluan kiittää myös vaimoani, joka on jaksanut olla vierellä lasten kanssa tässä koulukiireiden keskellä. Nyt on opinnot takana, kesä edessä ja työsopimus taskussa, joten uudelle uralle on tästä hyvä lähteä.

Kuopiossa 22.5.2019
Ilkka Rytkönen

SISÄLTÖ

LYHENTEET JA MÄÄRITELMÄT

API

Application Programming Interface. Ohjelmointirajapinta on määritelmä eri ohjelmien väliseen tiedonsiirtoon.

Back-end

Asiakas-palvelin-arkkitehtuurissa palvelinpäässä tapahtuva toiminta.

CSS

Cascaded Styled Sheet on tyylitiedosto, jolla määritellään HTML-sivujen ulkoasu.

Framework

Suomeksi ohjelmistokehys tai sovelluskehys

Front-end

Asiakas-palvelin-arkkitehtuurissa asiakaspäässä tapahtuva toiminta.

HTML

Hypertext Markup Language on kuvauskieli, jota käytetään verkkosivujen rakenteen esittämiseen.

HTTP

Hypertext Transfer Protocol on internetissä yleisesti käytettävä hypertekstin siirtoprotokolla. (Wikipedia, s.a.)

JavaScript

Ohjelmointikieli, jolla saadaan em. tekniikoiden avulla toteutetuille sivuille toiminnallisuutta ja logiikkaa. JavaScript suoritetaan tyypillisesti asiakaspäässä, eli selaimessa.

JSON

JavaScript Object Notation on Avoin tiedostomuoto tiedonvälitykseen. (Wikipedia, JSON, s.a.)

Ohjelmistokehys

Englanniksi framework. Ohjelmoinnin apuväline, jolla nopeutetaan uusien ohjelmistojen kehitystä. (Wikipedia, Ohjelmistokehys, s.a.)

PHP

PHP Hypertext Preprocessor on palvelinpäässä käytetty ohjelmointikieli. PHP:n harteille laitetaan mm. kaikki tietokantaan liittyvät tapahtumat.

REST

Repsesentational State Transfer. HTTP-protokollaan perustuva arkkitehtuurimalli rajapintojen toteuttamiseksi. (Wikipedia, s.a.)

Vue.js

Yleisesti käytössä oleva JavaScript-kielen ohjelmistokehys. Tämä nopeuttaa ja helpottaa ohjelmointia, sillä Vue.js pitää sisällään valmiina paljon testattuja toimintoja, joiden ohjelmointi suoraan JavaScriptiä käyttämällä veisi paljon aikaa.

1 JOHDANTO

1.1 Aiheen valinta

Aiheen valinta oli helppo, sillä se tuli suoraan työharjoittelupaikkani esimieheltä. Valittavanani oli kaksi erilaista työtä, joista tämä Kiho Raportointinäkymä valikoitui opinnäytetyössä työstettäväksi.

Kiho Cloud Platform -liiketoiminta-alustan nykyinen käyttöliittymä on toteutettu Vue.js:n avulla, joten oli luontevaa jatkaa saman ohjelmistokehyksen käyttöä myös opinnäytetyön käyttöliittymätoteutuksissa. Aihe oli kiinnostava, sillä halusin oppia käyttämään Vue.js-ohjelmistokehystä entistä tehokkaammin ja oppia hyödyntämään sen mahdollisuuksia monipuolisemmin.

Vue.js oli jossain määrin tuttu harjoittelussa tehdyn työn kautta, mutta sen täysimittainen hyödyntäminen vaati kuitenkin vielä opettelua. Vue.js:n lisäksi työssä joutui käyttämään myös puhdasta JavaScriptiä, HTML:ää sekä CSS:ää. Näiden lisäksi sovelluksen piti kyetä hakemaan tietoa tietokannasta rajapintojen kautta.

Opinnäytetyö oli tärkeä tulevan työni kannalta, sillä jatkoin Kiholla ohjelmistokehittäjänä opinnäytetyön valmistumisen jälkeen. Opin siis paljon tärkeää asiaa, jota pystyin hyödyntämään työtehtävissäni välittömästi. Opinnäytetyöllä oli merkitystä myös yhtiön kannalta, sillä työ on menossa tuotantokäyttöön. Opinnäytetyössä oppimani asiat eivät ole hyödyllisiä ainoastaan oman työpaikkani kannalta, vaan yleisesti käytössä olevien tekniikoiden osaaminen parantaa merkittävästi mahdollisuuksiani työmarkkinoilla.

1.2 Työn tilaaja

Työn tilaaja on Mastercom Oy, aputoiminimeltään Kiho. Työn aihe tuli suoraan Kihon vastaavalta ohjelmistosuunnittelijalta, Henri Hartikaiselta. Työ liittyy Kihon Cloud Platform liiketoiminta-alustaan kehittämiseen. Liiketoiminta-alusta koostuu erilaisista osista, joiden toimintoja halutaan seurata yhdeltä muokattavalta näkymältä. Liiketoiminta-alustan toimintoja ovat mm. työajanseuranta, ajopäiväkirja, telematiikka ja kalustovahti.

1.2.1 Kiho yrityksenä

Kiho on siis Mastercom Oy:n aputoiminimi. Mastercom Oy on perustettu 2003 Siilinjärvellä ja Kiho-brändi on otettu käyttöön 2010. Mastercom Oy on tarjonnut paikannuspalveluita vuodesta 2005 lähtien. Toimisto on muuttanut myöhemmin Kuopioon ja toimintaa on myös Espoossa, Jyväskylässä ja Tampereella.

KUVA 1. Kihon Kuopion toimipiste Itkonniemellä 3.6.2019

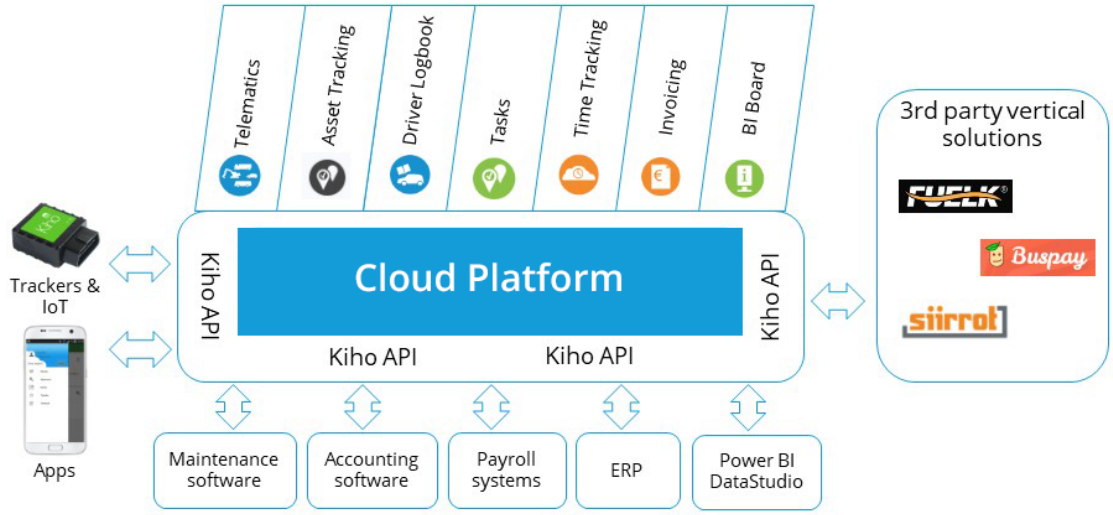
Yhtiö työllistää tällä hetkellä 32 henkilöä eri puolilla Suomea. Ohjelmistokehittäjiä on 10 ja heistä kaikki ovat Kuopiossa. Vuoden 2018 liikevaihto oli 2,9 miljoonaa euroa ja se oli lähes kaksinkertainen edellisvuoteen verrattuna. Kiholla on yli 500 yritysasiakasta, pääasiassa Suomesta.

1.2.2 Kiho Cloud Platform -liiketoiminta-alusta

Kiho Cloud Platform -liiketoiminta-alusta koostuu useista osa-alueista, jotka ovat:

- Kiho BI
- Kalustovahti
- Ajopäiväkirja
- Älykäs ajopäiväkirja
- Telematiikka
- Työajanseuranta
- Tehtävät
- Laskutus
- Ilmoitustaulu
- Pätevyydet

Kiho Cloud Platform -liiketoiminta-alustalla on tarkoitus digitalisoida yrityksen kaikki analoginen ja manuaalinen työ. Kiho-tuotteet toimivat saumattomasti yhdessä ja ne voi myös yhdistää avoimien rajapintojen kautta kolmansien osapuolten sovelluksiin. (Kiho, s.a.)



KUVA 2. Kihon liiketoiminta-alusta ja sen rajapinnat (Kiho, s.a.)

1.3 Työn tarkoitus ja tavoite

Työn tarkoituksena oli saada aikaiseksi toimiva ja valmis tuote tuotantokäyttöön. Halusin saada työstä hyötyä työnantajalleni ja heidän asiakkailleen.

Työtä tehdessä halusin oppia paljon uusia asioita ja saada rutiinia ohjelmointiin. Halusin oppia tämän työn tekemisen kautta ymmärtämään ja käyttämään entistä paremmin JavaScript-kielen ja Vue.js:n erityispiirteitä. JavaScript ohjelmistokehykset ovat tehty JavaScript-kielellä, joten JavaScript-kielen vahva osaaminen auttaa myös ohjelmistokehyksen tehokkaassa hyödyntämisessä.

Tärkein tavoitteeni oli saada työstä hyvä pohja, jolloin voisin jatkaa heti työelämään soveltamaan oppimiani taitoja. Oppiminen ei kuitenkaan lopu tähän, vaan tämä on vasta alkua. Kovan luokan osaaminen syntyy vasta pidemmän ajan kuluessa ja se vaatii jatkuvaa oppimista ja monien uusien asioiden sisäistämistä.

2 KÄYTETYT TEKNIIKAT

Aiheesta tiedetään paljon, sillä käytetyt tekniikat ovat yleisesti tunnettuja ja käytettyjä. Kyse on enemmän olemassa olevan tiedon soveltamisesta ja hyödyntämisestä parhaalla tavalla toteutettavaan järjestelmään.

HTML, CSS ja JavaScript ovat olleet web-teknologioiden ydintekniikoita vuosikaudet, eikä tähän asiaan ole muutosta näköpiirissä. Muutosta sen sijaan on tullut tapaan, jolla näitä tekniikoita käytetään. Monet JavaScript-ohjelmistokehykset ovat nousseet nopeasti suosioon, eikä ihme. Sovelluskehitystä voidaan nopeuttaa näillä melkoisesti. Ohjelmistokehyksiä käytettäessä vaarana on, että itse perustekniikan opiskelu jää liian vähälle.

2.1 HTML

HTML, eli Hypertext Markup Language on avoimesti standardoitu kuvauskieli, jolla kuvataan internet-sivuilla käytettävä rakenne. Näitä rakenteita ovat mm. erikokoiset otsikot, kappaleet, luettelot, hyperlinkit jne. HTML:ää on käytetty myös ulkoasun muokkaukseen, mutta nykyään käytössä olevan HTML5 -version myötä tästä on päästy jo pois.

KUVA 3. HTML5-logo (W3C HTML5 Logo, s.a.)

HTML on yksi web-ohjelmoinnin ydintekniikoista, vaikka HTML ei ole itsessään ohjelmointikieli. HTML on siitä huolimatta syytä osata hyvin. HTML-kuvauskieli koostuu erilaisista sisältöä kuvaavasta elementeistä. Kulmasulkeilla merkityt elementit ovat normaalisti kaksiosaisia, joiden väliin tulee varsinainen sisältö. Esimerkiksi sivun otsikko kirjoitetaan `title`-elementin sisään seuraavasti:

```
<title>Esimerkkiotsikko</title>.
```

Kauttaviiva ilmaisee lopettavan elementin. Elementit voivat olla sisäkkäin, kuten koodiesimerkki 1 osoittaa.

Joillakin elementeillä ei ole varsinaista sisältöä, kuten rivinvaihto- tai hr-elementin kohdalla. Tällöin aloittava ja lopettava elementti voidaan yhdistää seuraavan esimerkin mukaisesti:
`<hr />`. Lopettava kauttaviiva ei ole pakollinen HTML5:ssä, mutta suositeltava.

Elementeille voidaan antaa myös attribuutteja ja ne kirjoitetaan aloittavan elementin sisään. Esimerkiksi lomakkeelle syötettävä salasanaelementti näyttää seuraavalta:

`<input type=”password”>`.

Siinä `input`-tyyppinen elementti saa attribuutin `type`, jonka arvo on `password`.

Koodiesimerkki 1:ssä nähdään pienen sivun esimerkkikoodi. Kaiken aloittaa dokumenttityypin määrittely ensimmäisellä rivillä, joka HTML5:n tapauksessa on varsin lyhyt. Kaikki muu sisältö on `html`-elementtien sisällä. `Head`-elementtien sisällä olevat elementit eivät näy sisällössä, mutta sisältävät tärkeitä meta-, otsikko-, tyyli- ja skriptitietoa. (W3Schools, s.a.)

`Body`-elementtien sisällä oleva tieto on taas tarkoitettu suoraan näytettäväksi selaimessa, eli tämä paikka on varattu sisällölle. Täällä on normaalisti runsaasti erilaisia otsikoita, tekstikappaleita, taulukoita, listoja, kuvia ja muuta vastaavaa. Yleensä JavaScript-tiedosto ladataan juuri ennen lopettavaa `body`-elementtiä, jotta ohjelmakoodista voidaan viitata ennen `script`-elementtiä oleviin muihin elementteihin.

Elementit voidaan jakaa kahteen päätyyppiin, lohkotason- ja tekstitason elementteihin (Koppa, s.a.). Esimerkiksi otsikkoelementit ovat lohkotason elementtejä ja ne ottavat aina koko tilan itselleen leveyssuunnassa. Jos muuta rajoitetta ei ole, otsikko vie koko selaimessa näkyvän rivin leveyden itselleen, eikä samalla rivillä voi olla toista lohkotason elementtiä. Lohkotason elementin sisällä voi sen sijaan olla toisia tekstitason elementtejä. Esimerkiksi tekstikappaleessa voi olla normaalia tekstiä ja tekstin voi vaihtaa välillä esimerkiksi voimakkaaseen tekstiin `strong`-elementillä.

```
<!DOCTYPE html>
<html lang="fi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Sivun otsikko</title>
</head>
<body>
  <h1>Ykköstason otsikko</h1>
  <p>Tekstikappale</p>
  <script src="javascript.js"></script>
</body>
</html>
```

KOODIESIMERKKI 1. Perussivun HTML-mallipohja

Elementit voidaan kirjoittaa isoilla tai pienillä kirjaimilla ja lopputulos on sama. Suositus on kuitenkin, että kaikkien elementtien nimet kirjoitetaan aina pienillä kirjaimilla. (W3Schools, HTML Elements, s.a.)

2.2 CSS

CSS (Cascaded Styled Sheets) on nykyaikaisen web-ohjelmoinnin toinen ydintekniikka. HTML kertoo verkkosivun rakenteen, niin CSS ehdottaa sille ulkoasun. CSS:n avulla määritetään niin värit, fontit, asemoinnit, kuin muutkin ulkoasuun vaikuttavat asiat.

KUVA 4. HTML5-logo (Wikimedia Commons, s.a.)

CSS ehdottaa tyylien esitystä, mutta ei ole ehdoton. Toisin sanoen käyttäjä voi ohittaa suunnittelijan antaman tyylit omilla tyyleillään. Kuten muutkin tekniikat, CSS on myös kehittynyt vuosien varrella ja mukaan on tullut kaiken aikaa uusia ominaisuuksia. Nykyisin yleisesti käytetty versio on CSS3. (Wikipedia, s.a.)

CSS:n osaaminen tuo verkkosivujen suunnitteluun paljon joustavuutta, joten sen opettelua ei kannata laiminlyödä. CSS-tyylitiedostojen käytön helpottamiseksi on tarjolla paljon valmiita CSS-ohjelmistokehyksiä, joista käytetyin on Bootstrap. Muita vastaavia ovat esimerkiksi MaterializeCSS, Foundation, Bulma ja Semantic UI. Kaikissa näissä on tavoitteena nopeuttaa ohjelmistokehitystä ja saada verkkosivuista helposti näyttävän näköisiä. Näissä on otettu myös huomioon tyylien käyttäytyminen erilaisissa päätelaitteissa, jolloin verkkosivuista saa helposti yhteensopivat eri käyttöympäristöissä.

CSS-ohjelmistokehysten käyttö saattaa houkutella menemään sieltä, missä aita on matalin. CSS:n hyvä osaaminen auttaa ymmärtämään myös ohjelmistokehysten toimintaa sekä niiden muokkaamista oman näköisiksi. Jos kaikki käyttävät samaa ohjelmistokehystä, alkavat sivustot muistuttamaan toisiaan ulkoasun suhteen. (Traversy Media, 2018)

CSS-tyylit voidaan kirjoittaa kolmella eri tavalla. Ensimmäinen tapa on kirjoittaa tyylit suoraan HTML-elementin sisään attribuutiksi. Silloin tyyli kohdistuu vain tähän kyseiseen elementtiin. Tätä tapaa ei yleensä käytetä, sillä samaa tyyliä voidaan joutua käyttämään toisaalla. Silloin samat tyylimäärittelyt joudutaan kirjoittamaan moneen paikkaan ja sellaisen koodin luettavuus ja ylläpito on hankalaa.

Toinen ja parempi tapa tyylien kirjoittamiseen on sijoittaa ne HTML-dokumenttiin `style`-elementin sisälle. Tällöin kaikki tyylit ovat samassa paikassa ja samoja tyylejä voidaan käyttää useissa elementeissä. Koodin luettavuus ja ylläpito on myös paljon parempaa ensimmäiseen vaihtoehtoon verrat-

tuna. Tähänkin tapaan sisältyy ongelma, sillä samoja tyylejä halutaan käyttää usein monilla verkkosivuilla. Tällä menettelytavalla joudutaan samat tyylit monistamaan kaikkiin tiedostoihin, joka tekee ylläpidosta hankalaa.

Kolmas ja käytännöllisin keino tyylien kirjoittamiseen on sijoittaa ne omaan tiedostoon. Tällöin jokaisesta verkkosivusta viitataan samaan tyylitiedostoon, kuten koodiesimerkki 1:ssä on tehty `link`-elementin avulla. Tällöin riittää yhden tyylitiedoston muokkaus ja vaikutus näkyy kaikissa tiedostoissa. Tällä tavalla koodin luettavuus ja ylläpito pysyy myös hyvänä.

Jos samaan elementtiin on viitattu moneen kertaan, myöhemmin valittu tyyli tulee käyttöön. Myös sillä on väliä, onko tyyli ulkoisessa tiedostossa, `style`-elementissä, itse elementissä, palvelimella vai asiakkaan päässä ja onko `!important`-arvoa käytetty. Prioriteetit menevät suurimmat pienimpään seuraavasti:

1. Elementin sisällä oleva tyyli
2. Elementin yksilölliseen tunnisteeseen viittaava tyyli, eli ID
3. Elementin luokkaan viittaava tyyli

Tasa-arvoisissa tilanteissa viimeisin muutos jää voiman. Erityishuomio on käytettäessä !important-arvoa, sillä se ohittaa kaikki muut tyyliehdotukset. Sitä ei pitäisi normaalisi käyttää, tai sen käytössä pitää olla erityisen huolellinen.

Koodiesimerkki 2:ssa on pieni esimerkki, miten CSS-tyylejä käytetään. Aluksi kirjoitetaan valitsin, jonka jälkeen tyylien ominaisuudet ja arvot kirjoitetaan kaarisulkeiden sisään. Ominaisuuden jälkeen kaksoispiste erottelee sen arvosta. Yhdellä ominaisuudella voi olla useita arvoja ja ne erotellaan tilanteen mukaan joko pilkulla tai välilyönnillä. Rivin loppuun tulee puolipiste, joka erottelee ominaisuudet toisistaan. Tyylitiedostoissa voidaan käyttää muista ohjelmointikielistä tuttua kommentointitapaa. Merkintä `/*` aloittaa kommentin ja vastaavasti merkintä `*/` lopettaa kommentin.

Valitsimina voi käyttää mitä tahansa elementin nimeä, luokkaa tai id-tunnistetta. Valitsimia voi laittaa peräkkäin useita, jos halutaan sama tyyli vaikkapa useisiin elementteihin. Kun halutaan viitata luokkaan, käytetään valitsimen edessä pistettä. Verkkosivulla voi olla useita elementtejä, joilla on sama luokka. Samaa tyyliä voidaan siis helposti käyttää luokkien avulla niin monessa paikassa, kuin on tarvetta. Kullekin elementille voidaan antaa myös yksilöllinen tunniste ei id. Samaa id-tunnistetta ei saa käyttää samalla sivulla useaan kertaan. Kun tyyli halutaan kohdistaa tiettyyn tunnisteeseen, valitsimen eteen laitetaan #-merkki. Seuraavassa esimerkki otsikkoelementistä, johon on liitetty koodiesimerkki 2:n mukaiset tyylit:

```
<h2 id=”home-intro” class=”file-logo”>Otsikko</h2>
```

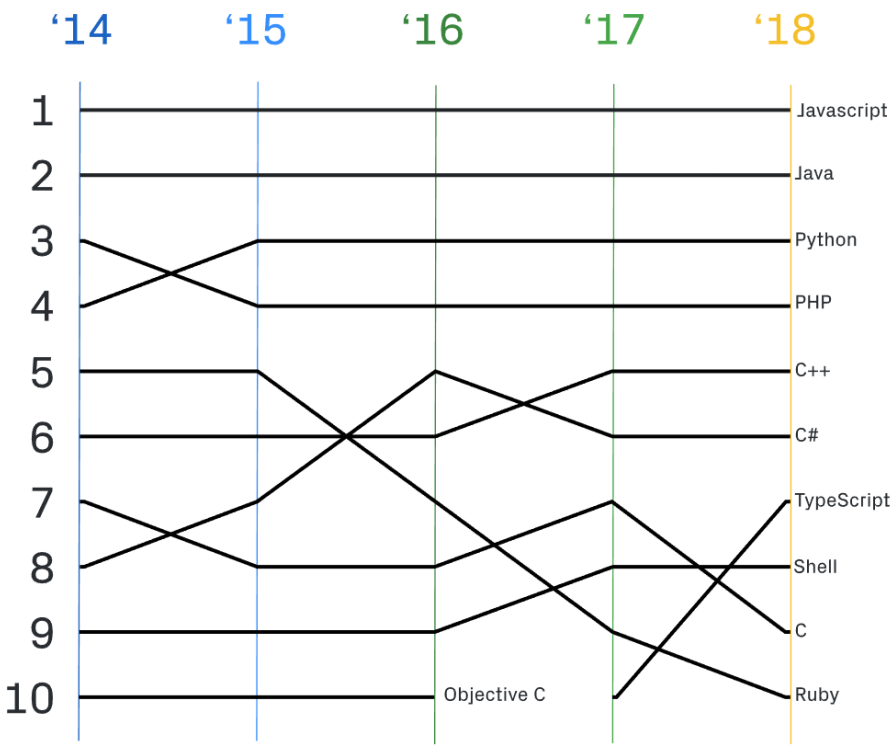
```css
body {
  /* Page background color and base font */
  background-color: beige;
  font-family: 'Crimson Text', serif;
}
h1, h2, h3, h4, h5, h6 {
  /* Color and font for all headings */
  font-family: 'Baloo Tammudu', cursive;
  color: #CC00FF;
  margin-left: 15px;
}
/* Styling links */
a {
  color: blue;
  text-decoration: none;
}
a:hover {
  color: mediumspringgreen
}
#home-intro {
  /* ID style */
  float: left;
  width: 50%;
  border: 2px dashed darkblue;
  padding: 30px;
  margin: 10px;
  font-size: 140%;
}
.file-logo {
  /* Class style for logos */
  width: 25px;
  margin-right: 5px;
  position: relative;
  top: 7px;
}
```

KOODIESIMERKKI 2. CSS-tyylitiedostoa

2.3 JavaScript

JavaScript on yksi tärkeimmistä ohjelmointikielistä ja kolmas nykyaikaisen web-sovelluskehityksen ydintekniikoista. HTML:llä ja CSS:llä verkkosivuille saadaan sisältöä ja ulkoasua, mutta kokonaisuudesta puuttuu vielä logiikka ja JavaScript tuo sen. Näillä kolmella tekniikalla voidaan toteuttaa kaikki nykyaikaisen verkkosivuston tai -sovelluksen toiminnot asiakaspäässä.

JavaScript on ollut käytetyin ohjelmointikieli jo vuosien ajan. GitHub on tehnyt Octoberverse -nimisen koosteen omasta käyttäjäkunnastaan ja heidän käyttämistään teknologioista. JavaScriptin ykkösasema ei ole uhattuna ainakaan lähitulevaisuudessa.



KUVIO 1. GitHubin suosituimmat ohjelmointikielet 2014 – 2018 (GitHub, 2018)

JavaScript on tulkattava ohjelmointikieli, eli sitä ei käännetä konekieliseksi ohjelmaksi ennen julkaisua. JavaScript-ohjelmaa suoritetaan yleensä selaimessa, eli asiakaspäässä. JavaScriptiä voidaan nykyisin suorittaa myös palvelimella, esimerkiksi Node.js:n avulla. Tällöin web-sovelluksen voi tehdä käyttäen vain yhtä ohjelmointikieltä. JavaScriptiä käytetään myös mobiilisovellusten teossa.

KUVA 5. JavaScript-logo (Seeklogo, s.a.)

JavaScript-kieli kehittyy jatkuvasti, joten sen vuoksi myös käytettävä syntaksi muuttuu. Kieli on saanut paljon uusia ominaisuuksia, jotka helpottavat ohjelmoijan työtä ja tuovat uusia mahdollisuuksia sovellusten kehittämiseksi.

Aikaisemmin JavaScriptiä on pidetty ehkä hieman kankeana ja sen käyttöön on ollut tarjolla monenlaisia laajennuksia. Näistä yksi suosituimmista laajennuksista on ollut jQuery, jolla ohjelmoijan työtä on helpotettu. JavaScriptin kehittymisen myötä jQueryn merkitys on vähentynyt, sillä samat hommat onnistuvat nykyisin suoraan ilman laajennuksia. (Traversy Media, 2018)

Nykyaikaisessa ohjelmistotuotannossa käytetään JavaScriptin kehittymisestä huolimatta paljon valmiita kirjastoja tai kokonaisia ohjelmistokehyksiä. Näiden avulla sovelluskehitystä saadaan nopeutettua ja samalla virheiden määrä vähenee. Ohjelmistokehyksistä huolimatta sovelluskehittäjän on hyvä osata puhdas ohjelmointikieli. Tämä auttaa ymmärtämään paremmin ohjelmistokehysten toiminnan ja puhdasta JavaScriptiä tarvitaan myös ohjelmistokehyksiä käytettäessä. Joskus puhtaaseen JavaScriptiin viitataan termillä Vanilla JavaScript.

JavaScript-ohjelmointikielen yleisimmin käytetty versio lienee ES5, toiselta nimeltään ECMAScript 5, tai ECMAScript 2009. ES5 toimii hyvin kaikilla nykyisillä selaimilla, joten sen käyttöä voidaan pitää varmana valintana yhteensopivuuden suhteen. (W3Schools, What is ES5?, s.a.)

ES6 on JavaScript ohjelmointikielen ES5-versiosta kehitetty seuraava versio, toiselta nimeltään ECMAScript 6, tai ECMAScript 2015.

Nykyiset selaimet eivät tue vielä täysin ES6-versiota, joten tätä ei voida käyttää suoraan asiakaspäässä ajettavassa ohjelmassa. ES6-syntaksin mukaista koodia voidaan kyllä käyttää huoletta kehittäjän näkökulmasta, sillä ES6-muotoinen JavaScript-koodi voidaan muuntaa helposti kaikilla selaimilla ajettavaan ES5-muotoon. Muunnos tapahtuu yleensä automatisoituna osana ohjelmistotuotantoprosessia esimerkiksi Babel-kääntäjällä.

ES6 tuo mukanaan monia parannuksia edelliseen versioon verrattuna. Lista on pitkä, mutta tärkeimmät uudet ominaisuudet ovat nuolifunktiot, luokat, let- ja const-muuttujat, uudet iteraattorit, moduulit sekä promiset.

```
var fn1 = function (a) {
  return a + 5;
}
var fn2 = a => a + 5;
console.log(fn1(3));
console.log(fn2(3));
```

KOODIESIMERKKI 3. Sama funktio sekä ES5-, että ES6-syntaksilla

ES6, eli ECMASCript 2015 ei ole viimeisin JavaScript-versio, vaan versiosta ES6 lähtien standardi on päivittynyt vuosittain. Uusin versio on ECMAScript 2018, joka on julkaistu kesäkuussa 2018. Näiden lisäksi on vielä käsite ES.Next, joka viittaa parhaillaan kehityksessä olevaan ECMAScript-versioon. (Wikipedia, s.a.)

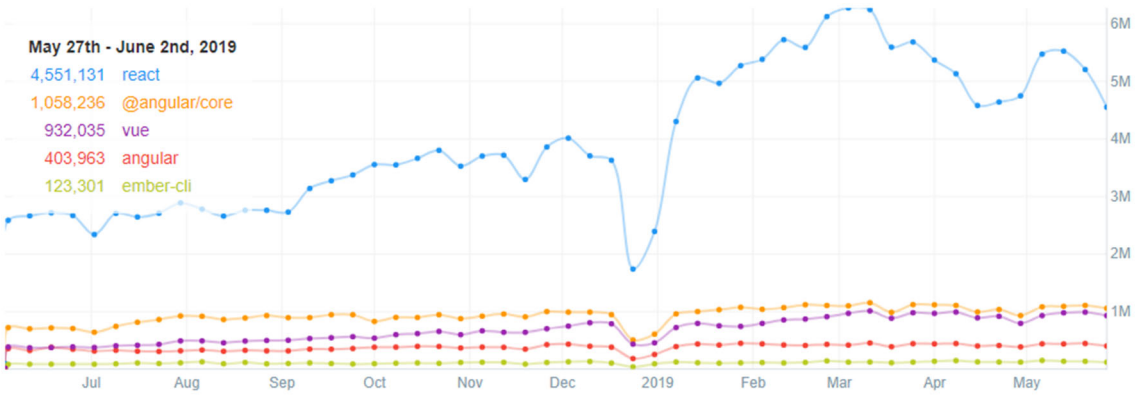
2.4 Vue.js

Asiakaspään ohjelmistokehykset ovat nousseet kovaa suosioon viime vuosina ja niitä käytetään nykyisin paljon. Ne helpottavat varsinkin suurempien verkkosovellusten tekemistä tarjoamalla valmiin paketin erilaisia testattuja ja organisoituja toimintoja mm. käyttöliittymän piirtämisestä tilanhallintaan. Näillä tehdään usein ns. yhden sivun sovelluksia, eli ne toimivat erittäin nopeasti. Palvelimelta

ladataan selaimeen kaikki tarvittava HTML-, CSS- ja JavaScript-koodi, joten ylimääräisiä HTTP-pyyntöjä ei tule.

Normaalissa verkkosivustossa kaikki linkkien painamiset tekevät aina pyynnön palvelimelle, josta tulee vastauksena uusi sivu. Tämä vie aikansa, joten käyttäjäkokemus ei ole samanlainen normaaliin työpöytäsovellukseen verrattuna. Yhden sivun sovelluksissa kaikki ohjelmakoodi on selaimen muistissa valmiina, joten sivuston sisäisten linkkien painamisesta ei tule uusia pyyntöjä ja sivulataukset ovat erittäin nopeita. Poikkeuksen tähän tekee tietokannoista haettavat tiedot, sillä niitä ei normaalisti varastoida asiakaspäähän.

Tällä hetkellä käytetään laajasti kolmea eri asiakaspään ohjelmistokehystä, jotka ovat suosituimmuusjärjestyksessä React, Angular ja Vue.js. Reactia voisi tosin luonnehtia kirjastoksi, mutta sen voi lisäpaketeilla saada vastaamaan kokonaista ohjelmistokehystä.



KUVIO 2. NPM-pakettien lataushistoriaa touko-kesäkuun vaihteesta 2019 (Npmcharts, s.a.)

Koska Kiho Cloud Platform -liiketoiminta-alustaa on kehitetty Vue.js:n avulla, oli luonnollista käyttää samaa ohjelmistokehystä myös tässä työssä. Vue.js:n taustalla ei ole isoa toimijaa, vaan sitä ylläpitää pieni ohjelmistokehittäjien joukko. Angularin ylläpidosta vastaa Google ja Reactin ylläpidosta Facebook, eli niillä on käytössä erittäin suuret resurssit ja markkinointikoneistot.

KUVA 6. Vue.js-logo (Vue.js, s.a.)

Vaikka Vue.js on pienempi, on se ohjelmistokehittäjien suosiossa ja sitä käyttävät monet suuretkin yhtiöt, kuten Alibaba ja Adobe. Vue.js:ään on otettu mukaan Angularin ja Reactin hyviä puolia ja sen

oppimiskäyrä on suhteellisen matala. Vue.js on myös suhteellisen kevyt ja erittäin suorituskykyinen, joten tämä ohjelmistokehys on tutustumisen arvoinen. (Academind, 2019)

Helpoimmillaan Vue.js:n alkuun pääsee vain liittämällä ohjelmistokehyksen koodi verkkosivun `script`-elementtiin.

```
<script src="https://unpkg.com/vue"></script>

<div id="app">
  <p>{{ teksti }}</p>
</div>
```

KOODIESIMERKKI 4. Vue.js:n käyttöönotto HTML-tiedostossa

HTML-koodin lisäksi tarvitaan JavaScript-tiedosto, joka näyttää yksinkertaisimmillaan seuraavalta:

```
new Vue({
  el: '#app',
  data: {
    teksti: Tämä teksti näytetään edellisen koodiesimerkin tekstikappaleessa'
  }
})
```

KOODIESIMERKKI 5. Vue.js-instanssin käyttöönotto erillisessä JavaScript-tiedostossa

HTML-koodissa oleva div-elementti näyttää Vue.js-instanssin kyseisen elementin tilalla. Vue.js-instanssi kytkeytyy kyseiseen elementtiin div-elementin id-attribuutin avulla. Vue.js näyttää `teksti`-muuttujassa olevan tekstin HTML-tiedostossa halutussa paikassa. Muuttujan nimi kirjoitetaan HTML:n sekaan kaksinkertaisten kaarisulkeiden sisään. Tällä tavalla voidaan näyttää myös objektien sisällä olevia arvoja.

Jos muuttuja on taulukko, voidaan se iteroida läpi HTML-tiedostossa muista ohjelmointikielistä tutulla for-silmukalla seuraavan esimerkin tapaan:

```
<ul>
  <li v-for="tuote in tuotteet">
    {{ tuote.nimi }}
  </li>
</ul>
```

KOODIESIMERKKI 6.

Edellinen koodiesimerkki käy tuotteet-taulukon läpi ja tulostaa jokaisen tuotteen nimen järjestämättömään listaan. Vue.js:ssä on käytössä myös paljon muita HTML-elementteihin liitettäviä attribuutteja, jotka tekevät ohjelmistosuunnittelusta hyvin joustavaa. Muuttujat voivat välittää tietoa kahteen suuntaan. Esimerkiksi lomakkeessa oleva tieto välitetään muuttujaan ja päinvastoin.

Jos Vue.js:llä ruvetaan tekemään uutta projektia, niin se aloitetaan yleensä Vue.js:n komennolla:

```
vue create projektin-nimi
```

Komento aloittaa projektin määrittelyn merkkipohjaisella käyttöliittymällä. Siinä päästään valitsemaan projektiin liitettäviä lisäosia ja muita asetuksia. Tuloksena on perusrunko uudelle Vue.js-projektille, jossa kaikki perustoiminnot on asetettu valmiiksi.

Normaalisti Vue.js-sovellukset koostetaan komponenteista. Yksi komponentti voi olla miten pieni tai suuri kokonaisuus vaan, mutta pienemmät komponentit ovat parempia ylläpidettävyyden kannalta. Yksi komponentti voi tehdä minimissään esimerkiksi vain vaakaviivan tulostamisen, mutta normaalisti komponentissa on hieman enemmän toimintoja.

Komponentit koostuvat kolmesta osasta, eli HTML-, JavaScript- ja CSS-osioista. HTML-osio on aina template-elementtien sisällä ja template saa sisältää ainoastaan yhden isäntäelementin, joka seuraavassa koodiesimerkissä on `ul`.

Varsinainen toimintalogiikka tulee `script`-elemetin sisälle, joka koostuu eri osa-alueista. Muuttujat sijoitetaan data-lohkoon, josta niitä voidaan käsitellä tavanomaiseen tapaan. Methods-lohkossa on kaikki metodit, joita tämän komponentin toiminnassa tarvitaan.

Tietoa voidaan välittään komponentilta toiselle, mutta vain isä- ja lapsikomponentin välillä. Suora tiedonsiirto sisarkomponenttien välillä ei ole mahdollista, mikä tekee tiedonvälityksestä hankalan suuremmissa ohjelmistoissa. Tieto viedään lapsikomponentille propsien avulla, kuten koodiesimerkissä 7 on tehty.

Props-muuttuja `videos` ottaa vastaan taulukon ja tätä muuttujaa voidaan lukea, mutta ei kirjoittaa. `Videos`-taulukko käydään läpi for-silmukassa ylempänä ja syötetään yksittäiset videot `VideoListItem`-komponenttiin. VideoListItem-komponentti saa videot vastaan video-nimisessä propsissa. Vienti tapahtuu kohdassa `:video="video"`. Attribuutin kaksoispiste on oleellinen asia, jolla parametrin video-muuttuja kytketään tähän attribuuttiin. Ilman kaksoispistettä välitettäisiin vain video-niminen merkkijono lapsikomponentille.

Jos tietoa halutaan välittää isäkomponentin suuntaan, tapahtuu se `$emit`-tapahtumalla. Koodiesimerkki 7:ssä `VideoListItem`-komponentti palauttaa muuttujan `@videoSelect`-tapahtumalla, joka laukaisee `onVideoSelect`-metodin. `onVideoSelect`-metodi ottaa palautuvan muuttujan kiinni ja vie sen `$emit`-tapahtumalla esimerkkikomponentin isäkomponentille.

```
<template>
    <ul class="list-group col-md-4">
        <VideoListItem
            v-for="video in videos"
            :key="video.etag"
            :video="video"
            @videoSelect="onVideoSelect"
        ></VideoListItem>
    </ul>
</template>

<script>
import VideoListItem from './VideoListItem';

export default {
    name: 'VideoList',
    components: {
        VideoListItem
    },
    data: function() {
        return {
            name: "Nimi",
            isActive: true,
            items: []
        }
    },
    props: {
        videos: Array
    },
    methods: {
        onVideoSelect(video) {
            this.$emit('videoSelect', video);
        }
    }
};
</script>

<style scoped>

</style>
```

KOODIESIMERKKI 7. Vue.js-komponenttiesimerkki (Grider, s.a.)

Edellisen koodiesimerkin lopussa on vielä `style`-elementti, johon tulisi tämän kyseisen komponentin CSS-tyylimäärittelyt. `Scoped`-attribuutti varmistaa, etteivät tyylit pääse leviämään tämän komponentin ulkopuolelle.

Komponenttien välillä voidaan tietoa siirtää myös kehittyneemmällä tavalla ja sen tekniikan nimi on Vuex. Vuex ylläpitää tilatietoa älykkäästi, niin että kaikki komponentit voivat käsitellä samaa tietoa

turvallisesti. Tilatieto pidetään muuttujissa, joihin ei ole tarkoitus koskea suoraan ulkopuolelta, vaan ainoastaan Vuex:n metodeilla.

Edelliset esimerkit ovat vain pintaraapaisua siihen, mitä Vue.js mahdollistaa. Yhden opinnäytetyön puitteissa ei ole mahdollista esittää edes kaikkea oleellista tietoa aiheesta. Lisätietoa Vue.js-ohjelmistokehyksestä on tarjolla runsaasti erilaisilla verkkosivustoilla aktiivisen kehittäjäyhteisön ansiosta.

2.5 REST API

Verkkosivustot ja web-sovellukset käsittelevät tietoa ja tieto tulee asiakaspäähän palvelimelta, yleensä tietokannasta. Tiedonvaihto tapahtuu tyypillisesti API-rajapintojen kautta, joka on sovittu tiedonvaihtotapa eri osapuolten kesken. Rajapintojen kautta päästään käsiksi niin omien ohjelmistojen, kuin myös kolmannen osapuolen tarjoamiin tietoihin.

Web-sovelluksissa rajapintojen tietoliikenne hoidetaan tyypillisesti normaalin HTTP-liikenteen kautta ja tässä kohtaa REST astuu kuvioihin. Tiedonvälitys perustuu siis tilattomaan HTTP-protokollaan, jolla hoidetaan asiakas-palvelin-arkkitehtuurin mukaista liikennettä. Tätä liikennettä voidaan käsitellä käytännössä kaikilla ohjelmointikielillä. (Traversy Media, 2017)

REST API:ssa tiedonvaihto tapahtuu tyypillisesti JSON-formaatissa, mutta myös XML on suhteellisen yleinen. Pyynnöt tapahtuvat URL:n avulla ja sopivia HTTP-otsikkotietoja käyttämällä. Asiakas tekee pyynnön ja palvelin vastaa, aina.


```
{
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
        "street": "Kulas Light",
        "suite": "Apt. 556",
        "city": "Gwenborough",
        "zipcode": "92998-3874",
        "geo": {
            "lat": "-37.3159",
            "lng": "81.1496"
        }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
        "name": "Romaguera-Crona",
        "catchPhrase": "Multi-layered client-server neural-net",
        "bs": "harness real-time e-markets"
    }
},
```


KUVA 7. JSON-muotoista testidataa kuvitteellisesta käyttäjästä (JSONPlaceholder, s.a.)

Vastauksen mukana pitäisi tulla aina myös vastauksen tilakoodi. Tilakoodi voi ilmaista onnistunutta pyyntöä, tai jonkunlaista virhettä. Onnistuneen pyynnön tilakoodi on 200. Monille tuttu koodi on varmaan 404, mikä ei ole loppukäyttäjän kannalta varmasti se selkein ilmoitus. Se tarkoittaa, ettei pyydettyä resurssia ollut olemassa. (MDN web docs, s.a.)

Tyypillisesti käytetään neljää erilaista HTTP-pyyntöä, jotka ovat GET, POST, PUT ja DELETE. Muitakin pyyntöjä on olemassa, mutta ne eivät ole yleensä välttämättömiä. GET-pyynnöllä haetaan tietoa ja se ei välttämättä tarvitse muita parametreja.

TAULUKKO 1. HTTP-pyyntöesimerkit

Pyyntö	**URL-esimerkki**	**Tapahtuma**
GET	www.saitti.fi/api/ilmoitukset	Hae kaikki ilmoitukset
GET	www.saitti.fi/api/ilmoitukset/1	Hae ilmoitus 1
POST	www.saitti.fi/api/ilmoitukset	Lisää uusi ilmoitus
PUT	www.saitti.fi/api/ilmoitukset/1	Päivitä ilmoitus 1
DELETE	www.saitti.fi/api/ilmoitukset/1	Poista ilmoitus 1

Taulukko 1:stä huomataan, että tässä tapauksessa URL pysyy samana, vaikka ne aiheuttavat eri tapahtuman. Käytettävä HTTP-pyyntö ratkaisee, mitä metodia palvelimella kutsutaan. URL voisi olla eri tyyppisissä pyynnöissä myös sellainen, että siitä näkisi samalla myös pyynnön tyypin. Kun pyyntö kohdistuu johonkin yksittäiseen resurssiin, viitataan URL:ssa normaalisti resurssin tietokannassa olevaan id-numeroon. Id-numeron sijalla voisi yhtä hyvin olla vaikkapa käyttäjätunnus.

HTTP-pyyntö voi tarvita autentikoinnin ja yleensä se on pakollinen käytettäessä PUT- tai DELETE-pyyntöjä. Esimerkiksi käyttäjän rekisteröinnissä autentikointia ei tarvita, mutta vaikka ilmoituksen jättäminen sen jo vaatii. Autentikoinnissa voidaan käyttää esimerkiksi tunnistetta, jota välitetään HTTP-otsikkotiedoissa. Otsikkotiedoissa voidaan välittää muutakin tietoa, esimerkkinä vastauksen pyytäminen JSON-muodossa.

HTTP-pyynnöt ovat turvattomia, sillä kaikki tiedot voidaan helposti väärentää. Asiakaspäässä toki kannattaa tehdä varmistuksia, jotta esimerkiksi loppukäyttäjän lomakkeen täyttäminen sujuisi juohevasti. Jos loppukäyttäjän tilalla on vihamielinen taho, voi se yrittää saada vahinkoa aikaiseksi lähettämällä virheellisen pyynnön. Tämän vuoksi kaikki pyynnöt pitää aina varmistaa palvelimella.

2.6 Git-versionhallinta

Git-versionhallinta kuuluu olennaisena osana nykyaikaista ohjelmistokehitystyötä. Usean ihmisen yhteistyö olisi haastavaa, ellei mahdotonta ilman sopivaa versionhallinnan käyttöä.

Git on hajautettu versionhallintajärjestelmä, joka pitää kirjaa tiedostoista ja niiden muutoksista projektin aikana. Sillä voidaan jäljittää tiedostoissa tapahtuneet muutokset ja koko versiohistoria on tallessa.

Kukin ohjelmistokehitysprojektiin osallistuva lataa koko projektin omalle koneelleen paikallisesti, jolloin kaikki versionhallinnan tieto on käytettävissä myös ilman verkkoyhteyttä. Kukin kehittäjä voi kehittää ohjelman osia eri haaroissa (branch). Haaroja voi olla useita ja eri haaroissa olevat tiedostot pysyvät versionhallinnan avulla erillään, vaikka työskennellään samassa hakemistossa.

3 TEKEMISEN VAIHEET

3.1 Suunnittelu

Opinnäytetyön tekeminen alkoi palaverilla ohjaajan ja toimeksiantajan kesken. Myöhemmin pidimme toisen palaverin toimeksiantajan kanssa, jossa työn sisältö alkoi hahmottua tarkemmin. Teimme muistion työhön haluttavista asioista ja ominaisuuksista. Muistio toimi ohjelmaa kirjoittaessa tuotteen kehitysjonona, tai tässä tapauksessa sprintin kehitysjonona.

Sain erilliseltä käyttöliittymäsuunnittelijalta hahmotelman, millaiselta lopputulos voisi näyttää. Siinä oli mietitty valmiiksi mm. kaavioiden asettelua, kuvakkeita ja värejä. Tämä teki työn aloittamisesta helpompaa, kun oli jokin selkeä tavoite.

Kiho Raportointinäkymän piti noudattaa sivuston muuta ulkoasua ja logiikkaa. Tämän vuoksi Kihon käyttöliittymäkomponentteja pyrittiin kierrättämään. Kaikki tarvittavat painikkeet ja muut käyttöliittymäkomponentit löytyivät valmiina. Mikäli työssä olisi tarvittu uusia käyttöliittymäkomponentteja, olisi niistä pitänyt tehdä nykyisten toimintatapojen ja logiikan mukaisia.

3.2 Toteutus

Työn tekemisessä oli suhteellisen vapaat kädet toteuttaa työtä halutulla tavalla. Kehitys tapahtui nopeassa tahdissa uusia prototyyppejä luoden ja niitä edelleen kehittäen.

3.2.1 Tiedon visualisointikirjasto

Työn toteutus alkoi sopivan JavaScript-pohjaisen tiedon visualisointikirjaston etsimisellä. Tällaisia avoimen lähdekoodin kirjastoja on tarjolla useita, joten aluksi piti tehdä pientä vertailua näiden välillä. Osa oli tehty pelkästään React-ohjelmointikehykselle, joten tällaiset jäivät laskuista heti pois.

Muutama visualisointikirjasto tarjosi suoraan yhteensopivuuden Vue.js-ohjelmistokehyksen kanssa, joten huomio kiinnittyi näihin. Aluksi tehtiin testikaaviot kullakin vaihtoehdolla ja päädyttiin käyttämään ApexChars-kirjastoa.

3.2.2 Visualisoitavan tiedon hakeminen ja muokkaaminen

Aluksi piti selvittää olemassa olevien rajapintojen kautta saatavaa tietoa ja sen soveltuvuutta haluttujen asioiden visualisointiin. Samalla piti tutustua eri rajapintakutsujen parametreihin mahdollisimman valmiin tiedon saamiseksi. Palautuva JSON-muotoinen tieto ei kuitenkaan sellaisenaan ollut valmista visualisoitavaksi, vaan tiedosta piti hakea oikeat asiat ja muokata sopivaan muotoon.

Tämä osa työstä oli työläin, sillä tiedot olivat jossain määrin pirstaleisia ja kaikkien saatavilla olevien tietojen tarkoitus ei ollut ilmiselvää. Samalla joutui selvittämään Kihon liiketoiminta-alustassa käytettävien termien tarkoitusta ja niiden englanninkielisiä vastineita.

Kun liiketoiminta-alustan toiminnot olivat selvillä, piti miettiä tapaa saada tiedot sopivaan esitysmuotoon. Raakatietoa jouduttiin tässä vaiheessa muokkaamaan runsaasti ja lopputuloksena tiedoista saatiin aikaiseksi yksinkertaiset JavaScript-taulukot.

3.2.3 Vue.js-komponenttien kirjoittaminen

Kun tarvittava tieto oli muokattu käyttötarkoitusta vastaavaan muotoon, oli aika ruveta kirjoittamaan käyttöliittymässä käytettäviä Vue.js-komponentteja. Vue.js:n käyttö oli tuttua jo työharjoittelusta, mutta tässä opinnäytetyössä aihetta piti opetella lisää.

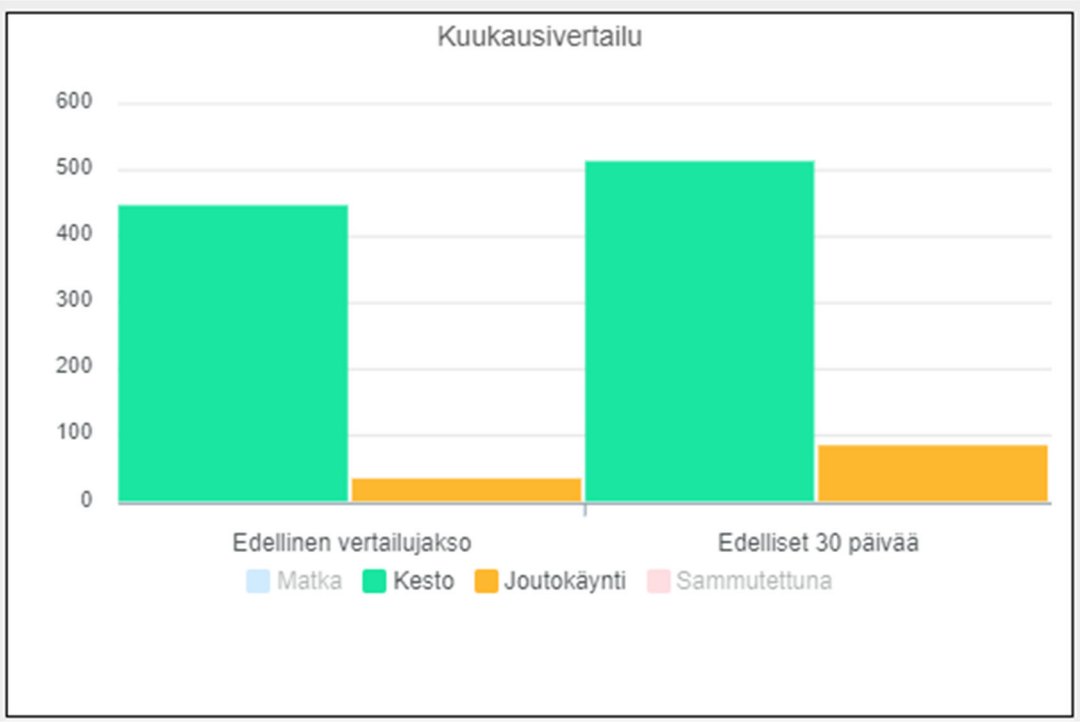
ApexCharts-kaaviokirjasto tarjosi kaiken tarvittavan tämän työn toteuttamiseksi, mutta toiminnot piti paketoida sopivaan, uudelleenkäytettävään Vue.js-komponenttiin. Näytti siltä, että kaikkien kaavioiden näyttämiseksi tarvitaan vain yksi ainoa komponentti. Aluksi olikin näin ja vain komponentin attribuutteja muuttelemalla homma alkoi toimia.

Myöhemmin käyttöön otettu Vue-grid-layout-kirjasto aiheutti kuitenkin ongelmia, joten jokainen esitettävä graafi piti laittaa omaan komponenttiinsa. Erilliskomponentit käyttivät kuitenkin isäntänä tätä aikaisemmin tehtyä pääkomponenttia. Jos kaikki graafit yritti esittää yhdellä komponentilla, niiden paikkojen vaihtamisen aikana ruudunpäivitysnopeus tippui huonolle tasolle. Erilliskomponenteilla päivitysnopeus oli puolestaan hyvällä tasolla, vaikka muutos ei näennäisesti muuttanut koodirakennetta mihinkään suuntaan.

```
<template>
    <div id="chart">
        <apexchart
            :type="type"
            :height="height"
            :width="width"
            :options="chartOptions"
            :series="series" />
    </div>
</template>
```

KOODIESIMERKKI 8. Vue.js:llä tehdyn kaaviokomponentin HTML-osuus

Aluksi tehtiin yksinkertaisen prototyyppisivusto, jonne kerättiin erilaisia kaavioita. Tässä hyödynnettiin aiemmin haettua ja muokattua tietoa eri tavoin. Uusia kaavioita lisätessä jouduttiin alkuperäistä Vue.js-komponenttia päivittämään erilaisten ApexCharts-ominaisuuksien käyttöön ottamiseksi.



KUVA 8. Esimerkkikaavio telematiikkatiedoista

3.2.4 Muokattavan käyttöliittymän tekeminen

Käyttöliittymäsuunnittelussa minulla oli apuna erillinen käyttöliittymäsuunnittelija, joka hahmotteli mahdollista käyttöliittymää, ulkoasua sekä värimaailmaa. Käyttöliittymäksi ei käy pelkästään satunnaiseen järjestykseen laitetut kaaviot. Käyttöliittymäsuunnittelussa käytin hyväkseni olemassa olevia komponentteja mahdollisuuksien mukaan. Käyttöliittymästä haluttiin muokattava, eli käyttäjän tulisi pystyä muuttamaan kaavioiden asetuksia, niiden järjestystä ja kokoa.

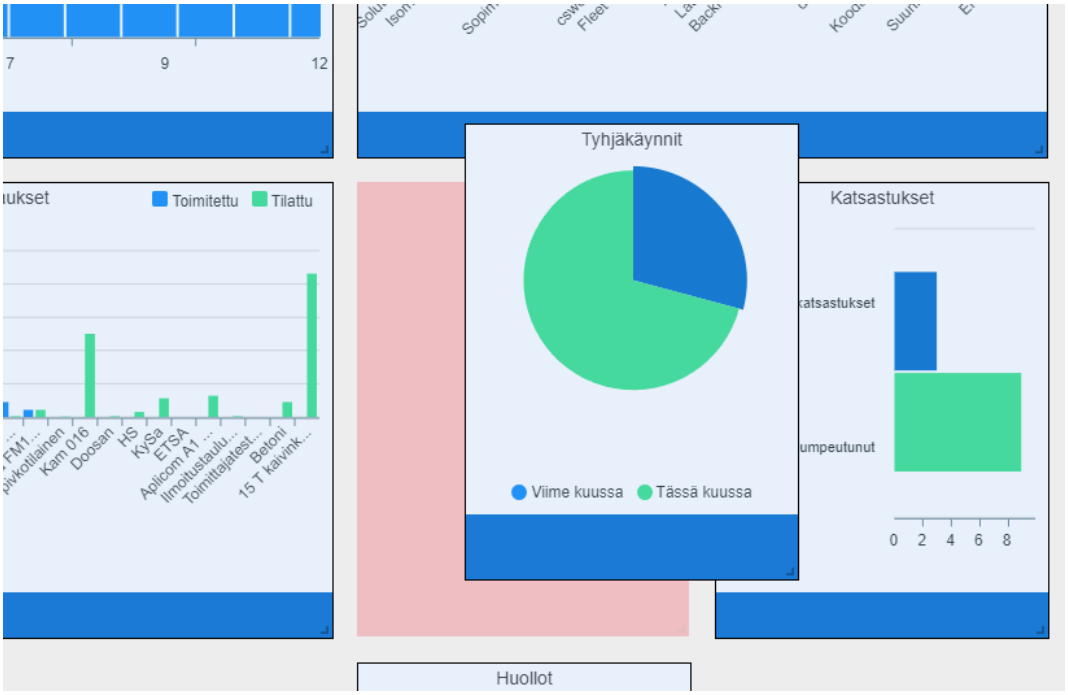
Tällaisen muokattavan ulkoasun mahdollistavia JavaScript-kirjastoja on valmiina, joten katse kohdistui niiden tutkimiseen. Lupaavia vaihtoehtoja oli useita, joita oli vaivatonta päästä kokeilemaan. Kirjastojen asennus tapahtui NPM-paketinhallintasovelluksella. Kahden ensimmäiseksi kokeillun kirjaston käytössä ilmeni ongelmia, sillä ne eivät olleet yhteensopivia Kihon liiketoiminta-alustan kanssa.

Ongelmaan oli törmätty aiemmin, eikä sitä ollut silloin saatu ratkaistua. Sen vuoksi jatkoin etsintää ja löysin Interact.js-nimisen kirjaston, jonka kanssa ei ollut yhteensopivuusongelmaa. Tämä kirjasto oli vaatimattomampi, kuin aiemmin kokeillut. Sen käyttäminen tietäisi paljon omaa koodausta, jotta kaaviot saataisiin ruudukkonäkymään ja niiden paikkatiedot tallennettua.

Myöhemmässä palaverissa kävi ilmi, että yhteensopivuusongelmaan tartutaan ja se yritetään ratkaista. Näin kävi ja Kiho Raportointinäkymään voitiin ottaa käyttöön monipuolisia, käyttöliittymän komponenttien liikuttelun mahdollistavia kirjastoja.

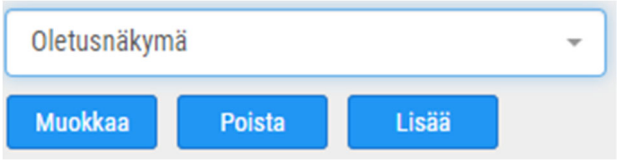
Vue-grid-layout-niminen kirjasto tuntui olevan näistä paras vaihtoehto. Kuten jo nimestä pystyy arvaamaan, on tämä kirjasto suunniteltu Vue.js:n kanssa yhteensopivaksi. Sen avulla kaaviot saatiin

liikkumaan hiiren avulla paikasta toiseen. Toiset kaaviot osaavat väistyä alta pois ja mennä seuraaviin vapaisiin paikkoihin. Kirjasto on myös mobiiliyhteensopiva, eli kaavioita voi liikutella sormella kosketusnäyttöisissä laitteissa. Oletuksena kaaviot eivät ole tosin mukaudu pieniin näyttöihin, mutta asia korjaantuu Vue-grid-layoutin asetuksia muuttamalla.



KUVA 9. Tyhjäkäynnit-kaavion paikkaa ollaan muuttamassa

Vue-grid-layoutin tekemä käyttöliittymä on jaettu 12 sarakkeeseen, eli enimmillään samalla rivillä voi olla 12 erilaista komponenttia. Tällöin se vie kaiken käytettävissä olevan tilan leveyssuunnassa. Komponentille voi antaa myös enemmän tilaa, vaikkapa kaikki 12 saraketta. Tällöin tämä elementti vie yksin kaiken käytettävissä olevan tilan. Samaan tilaan voi laittaa myös kaksi kappaletta kuuden sarakkeen levyistä elementtiä. Vue-grid-layoutin pohjalla on todennäköisesti CSS:n grid layout.



KUVA 10. Käyttöliittymän painikkeiden ja alasvetovalikon perustila

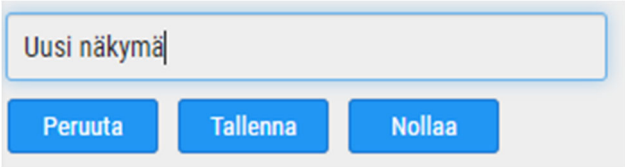
Käyttöliittymään haluttiin erillinen muokkaustila, etteivät kaaviot lähde vahingossa eri paikkoihin hiiren mukana. Vue-grid-layoutissa oli tähän valmis asetus, samoin kuin komponenttien koon muutokselle. Muokkaa-nappia painamalla päästään editointitilaan ja samassa yhteydessä tallennetaan sen hetkinen tilanne. Jos käyttäjä muutoksien jälkeen haluaakin palata lähtötilanteeseen, onnistuu se tämän avulla helposti.

Vue-grid-layout pitää yllä tietoa komponenttien sijainnista ja koosta JavaScript-taulukossa, jossa on puolestaan jokaiselle komponentille oma objekti. Objekteissa on x- ja y-koordinaatit, komponentin leveys sekä korkeus ja viimeisimpänä komponentin yksilöllinen id-numero.

```
layout:
[
    { 'x': 0, 'y': 0, 'w': 2, 'h': 1, 'i': '0' },
    { 'x': 2, 'y': 0, 'w': 2, 'h': 1, 'i': '1' },
    { 'x': 4, 'y': 0, 'w': 2, 'h': 1, 'i': '2' },
    { 'x': 0, 'y': 1, 'w': 2, 'h': 1, 'i': '3' },
    { 'x': 2, 'y': 1, 'w': 2, 'h': 1, 'i': '4' },
    { 'x': 4, 'y': 1, 'w': 2, 'h': 1, 'i': '5' },
    { 'x': 0, 'y': 2, 'w': 2, 'h': 1, 'i': '6' },
    { 'x': 2, 'y': 2, 'w': 2, 'h': 1, 'i': '7' },
    { 'x': 4, 'y': 2, 'w': 2, 'h': 1, 'i': '8' },
    { 'x': 0, 'y': 3, 'w': 2, 'h': 1, 'i': '9' },
    { 'x': 4, 'y': 3, 'w': 1, 'h': 1, 'i': '10' },
    { 'x': 5, 'y': 3, 'w': 1, 'h': 1, 'i': '11' },
    { 'x': 0, 'y': 4, 'w': 1, 'h': 1, 'i': '12' },
    { 'x': 1, 'y': 4, 'w': 1, 'h': 1, 'i': '13' },
    { 'x': 2, 'y': 4, 'w': 1, 'h': 1, 'i': '14' }
]
```

KOODIESIMERKKI 9. Vue-grid-layoutin asetukset JavaScript-taulukossa

Vue-grid-layoutin asetustiedostoa muokattiin hieman ja asetuksille annettiin nimi, sekä tunniste. Tämän jälkeen oli mahdollista lisätä useita näkymiä, joilla kaikilla voi olla yksilölliset asetukset. Samassa yhteydessä tehtiin myös näkymän poisto ja näkymän muuttaminen oletustilaan.

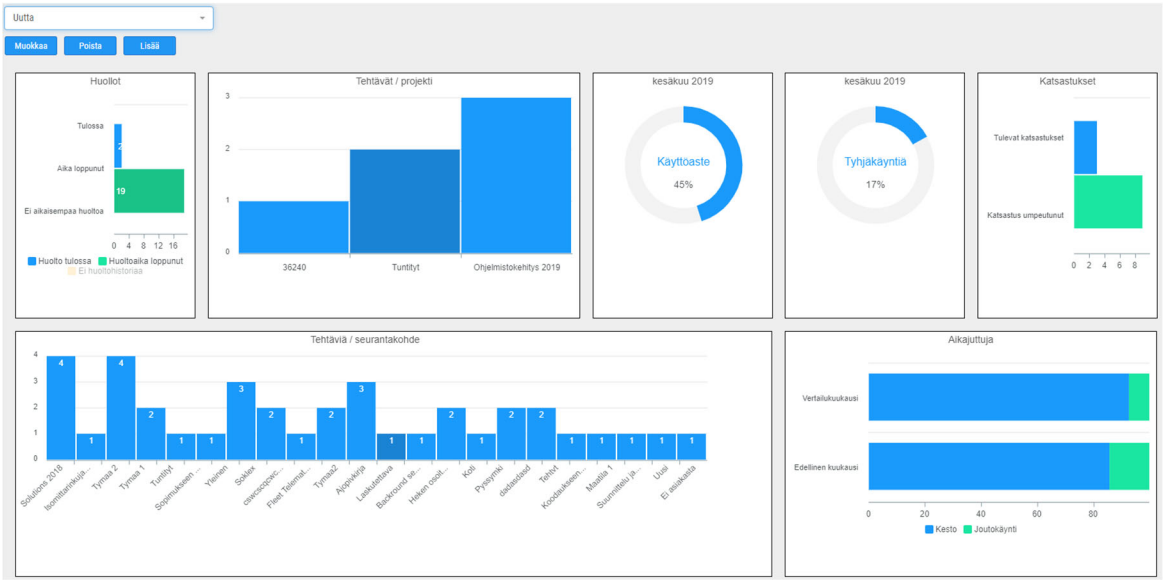


KUVA 11. Käyttöliittymän painikkeiden tila muokkausnäkymässä

3.3 Viimeistely

Kun kokonaisuus alkoi olla kasassa, piti työtä vielä viimeistellä. Tämä tarkoitti käyttöliittymän pieniä puutteiden muokkausta lähemmäksi alkuperäistä suunnitelmaa. Samalla koodia tuli siistittyä ja sieltä tuli poistettua kehitystyön aikaisia ohjelmalohkoja, kommentteja sekä tulostuksia.

CSS-tyylejä ei tässä työssä tarvinnut paljon kirjoitella, sillä käytössä olevat tyylit hoitivat hommansa hyvin. Tyylitiedostoihin piti tehdä ainoastaan pientä hienosäätöä, jolla saatiin aikaan viimeinen silaus.



KUVA 12. Kiho Raportointinäkymä opinnäytetyön valmistumisen aikaan

Raportointinäkymän asetukset oli tarkoitus tallentaa tietokantaan, mutta tätä vaihetta ei ehditty toteuttaa ajan puutteen vuoksi. Sen sijaan asetukset tallennetaan paikallisesti selaimen muistiin, eli local storageen. Huono puoli tässä on, että asetukset eivät seuraa käyttäjää laitteelta tai selaimelta toiselle. Tietokantaan tallennus ei sinällään olisi enää ison työn takana, sillä tieto serialisoidaan JSON-muotoon paikalliseen muistiin tallennettaessa.

4 JATKOKEHITYS

Opinnäytetyön aikataulun puitteissa saatiin aikaan suhteellisen valmis tuote. Raportointinäkymään jäi kuitenkin puutteita, jotka pitää hoitaa kuntoon.

Tärkeimpänä parannuskohteena on erillisen rajapinnan tekeminen vain raportointikäyttöä ajatellen. Tällöin kaikki tarvittava tieto saadaan yhdellä kutsulla, kun nykyisin kutsuja on yli kymmenen. Jokainen kutsu vie aikaa, joten yhden kutsun käyttämisellä saadaan selvää nopeushyötyä.

Tarvittavaa nopeutta ei ole mahdollista saavuttaa pelkästään tarkoitusta varten tehdyllä rajapinnalla, vaan pitää mennä syvemmälle järjestelmään. Näytettävät tiedot vaativat paljon prosessointia, varsinkin jos yrityksellä on paljon työntekijöitä, kalustoa, työmaita, kustannuspaikkoja yms.

Kaikista näistä tiedoista koosteen tekemiseen menee oma aikansa, joten olisi järkevämpää laskea kaikki nämä tiedot jo ennakkoon yhteen tietokannan tauluun. Kun kaikki tarvittava tieto on valmiiksi laskettuna, niin sen hakeminen ja näyttäminen on nopeaa. Tiedon ei tarvitse olla reaaliaikaista, vaan viive voisi olla ehkä muutaman tunnin luokkaa.

Asetukset pitää saada myös tietokantaan, mutta se on tässä vaiheessa toteutettavassa hyvin pienellä vaivalla.

Koodin rakennetta voisi myös miettiä hieman uudelleen. Tällä hetkellä raportointinäkymän Vue.js-pääkomponentin koko on turhankin iso, mikä haittaa mm. koodin muokkaamista sekä luettavuutta. Ohjelmakoodin pitäisi pilkkoa pienempiin ja helpommin hallittaviin osiin. Tämä helpottaisi Kiho Raportointinäkymän jatkokehitystä riippumatta siitä, onko asialla alkuperäinen tai uusi työntekijä.

Yksittäisiin kaavioihin pitää saada myös piilotustoiminto, jos käyttäjä ei halua yhdelle näytölle kaikkia tarjolla olevia kaavioita. Raportoinnin jaksot pitäisi pystyä myös valitsemaan kalenterinäkymästä. Tällöin tarjolle pitäisi tulla automaattisesti yhtä pitkä vertailujakso.

5 YHTEENVETO

Työn tavoitteena oli tehdä selkeä ja havainnollinen raportointinäkymä Kiho Cloud Platform -liiketoiminta-alustaan. Raportointinäkymään piti saada tietoja visuaalisessa muodossa telematiikasta, tehtävistä, työajasta, laskutuksesta ja kalustoluettelosta. Opinnäytetyöhön varatussa ajassa raportointinäkymästä ei saatu vielä aivan tuotantokäyttöön soveltuvaa versiota aikaiseksi, mutta lähelle päästiin. Asiakassovellus tuli tehtyä melkein valmiiksi pieniä hienosäätöjä lukuun ottamatta. Raportointinäkymän sujuvaan toimintaan vaadittavat tietokanta- ja rajapintamuutokset jätimme opinnäytetyön ulkopuolelle. Näiden pariin palaan heti tämän opinnäytetyön jälkeen.

Työllä on merkitystä toimeksiantajalle siinä mielessä, että Kihon liiketoiminta-alusta saa uusia asiakkaiden toivomia ominaisuuksia. Asiakkaat puolestaan hyötyvät näistä uusista toiminnoista hallitessaan omia liiketoimintaprosessejaan. Kun asiakkaat saavat tilaamallaan tuotteella lisäarvoa, auttaa se toimeksiantajaa toimittamaan tuotteita yhä useammille asiakkaille.

Työn tekijänä sain itse tästä paljon hyödyllistä oppia ja tietoutta. Opin käyttämään ennestään jossain määrin tuttua tekniikkaa paremmin ja tehokkaammin. Opin samalla myös kokonaan uusia asioita ja toimintatapoja. Ammattitaito siis kasvoi ja olen työnantajan kannalta entistä houkuttelevampi työntekijä. Olen päässyt entistä paremmin sisään Kihon järjestelmään, joten se helpottaa tulevia projekteja. Tulevat projektit ovat suoraan jatkoa opinnäytetyölle, sillä aloitan Kiholla toistaiseksi voimassa olevan työsuhteen nykyisen sopimuksen päättymisen jälkeen.

LÄHTEET JA TUOTETUT AINEISTOT

Academind. (10. 4. 2019). *Vue.js in 2019 & Beyond. YouTube*. Noudettu osoitteesta https://www.youtube.com/watch?v=SC74AXdOCh4

ApexCharts. (s.a.). Haettu 2. 6. 2019 osoitteesta https://apexcharts.com/

ESLint. (27. 1. 2016). *GitHub*. Haettu 2. 6. 2019 osoitteesta https://github.com/eslint/eslint/issues/5085

GitHub. (2018). *Octoverse 2018*. Haettu 4. 5. 2019 osoitteesta https://octoverse.github.com/projects

Grider, S. (s.a.). *Vue JS Essentials with Vuex and Vue Router. Udemy*. Haettu 4. 6. 2019 osoitteesta https://www.udemy.com/vue-js-course/

JSONPlaceholder. (s.a.). *Fake Online REST API for Testing and Prototyping*. Haettu 4. 6. 2019 osoitteesta https://jsonplaceholder.typicode.com/

Kiho. (s.a.). Haettu 26. 5. 2019 osoitteesta https://www.kiho.fi/en/

Kiho. (s.a.). *Integraatiot*. Haettu 17. 5. 2019 osoitteesta https://www.kiho.fi/integraatiot/.

Koppa. (s.a.). *XHTML*. Haettu 2. 6. 2019 osoitteesta https://koppa.jyu.fi/avoimet/hum/tvt/www/xhtml

MDN web docs. (s.a.). *HTTP response status codes*. Haettu 4. 6. 2019 osoitteesta https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Npmcharts. (s.a.). Haettu 3. 6. 2019 osoitteesta https://npmcharts.com/

Seeklogo. (s.a.). *JavaScript Logo Vector*. Haettu 2. 6. 2019 osoitteesta https://seeklogo.com/vector-logo/255387/javascript

Traversy Media. (20. 2. 2017). *What Is A RESTful API? Explanation of REST & HTTP. YouTube*. Noudettu osoitteesta https://www.youtube.com/watch?v=Q-BpqyOT3a8

Traversy Media. (2. 4. 2018). *Is jQuery Still Relevant in 2018? YouTube*. Noudettu osoitteesta https://youtu.be/pk3tsynNZ0w

Traversy Media. (9. 9. 2018). *Web Development Framework Pros & Cons. YouTube*. Noudettu osoitteesta https://www.youtube.com/watch?v=KoQY8f3bepk&t=947s

W3C HTML5 Logo. (s.a.). Haettu 1. 6. 2019 osoitteesta https://www.w3.org/html/logo/

W3Schools. (s.a.). *CSS Specificity*. Haettu 3. .6 2019 osoitteesta https://www.w3schools.com/css/css_specificity.asp

W3Schools. (s.a.). *HTML <head> Tag*. Haettu 2. 6. 2019 osoitteesta https://www.w3schools.com/tags/tag_head.asp

W3Schools. (s.a.). *HTML Elements*. Haettu 2. 6. 2019 osoitteesta https://www.w3schools.com/html/html_elements.asp

W3Schools. (s.a.). *What is ES5?* Haettu 29. 5. 2019 osoitteesta https://www.w3schools.com/whatis/whatis_es5.asp

Webpack. (s.a.). Haettu 13. 5. 2019 osoitteesta https://webpack.js.org/

Wikimedia Commons. (s.a.). *File:Babel Logo.svg*. Haettu 1. 6. 2019 osoitteesta https://commons.wikimedia.org/wiki/File:Babel_Logo.svg

Wikimedia Commons. (s.a.). *File:CSS3 and HTML5 logos and wordmarks.svg*. Haettu 2. 6. 2019 osoitteesta https://commons.wikimedia.org/wiki/File:CSS3_and_HTML5_logos_and_wordmarks.svg

Wikipedia. (s.a.). *Cascading Style Sheets*. Haettu 2. 6. 2019 osoitteesta https://fi.wikipedia.org/wiki/Cascading_Style_Sheets

Wikipedia. (s.a.). *Comparison of JavaScript frameworks*. Haettu 22. 4. 2019 osoitteesta https://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

Wikipedia. (s.a.). *Devops*. Haettu 4. 6. 2019 osoitteesta https://fi.wikipedia.org/wiki/Devops

Wikipedia. (s.a.). *ECMAScript*. Haettu 4. 6. 2019 osoitteesta https://en.wikipedia.org/wiki/ECMAScript

Wikipedia. (s.a.). *HTTP*. Haettu 4. 6. 2019 osoitteesta https://fi.wikipedia.org/wiki/HTTP

Wikipedia. (s.a.). *JSON*. Haettu 4. 6. 2019 osoitteesta https://fi.wikipedia.org/wiki/JSON

Wikipedia. (s.a.). *Ohjelmistokehys*. Haettu 29. 5. 2019 osoitteesta https://fi.wikipedia.org/wiki/Ohjelmistokehys

Wikipedia. (s.a.). *REST*. Haettu 4. 6. 2019 osoitteesta https://fi.wikipedia.org/wiki/REST

Vue.js. (s.a.). *The Progressive JavaScript Framework*. Haettu 2. 6. 2019 osoitteesta https://vuejs.org/