jamk.fi

Visualisointityökalun kehittäminen

Lauri Girsén

Opinnäytetyö
Huhtikuu 2020
Tekniikan ala
Insinööri (AMK), tieto ja -viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Girsén, Lauri	Opinnäytetyö, AMK	huhtikuu 2020
	Sivumäärä 24	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Visualisointityökalun kehittäminen

Tutkinto-ohjelma
Tieto ja -viestintätekniikka

Työn ohjaaja(t)
Kari Niemi, Esa Salmikangas

Toimeksiantaja(t)
Into-Digital Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa yrityksen sisäiseen käyttöön raportointityökalu, joka visualisoi dataa yrityksen toiminnanohjausjärjestelmästä. Toimeksiantajalla on käytössä toiminnanohjausjärjestelmä (Visma Severa), jossa projektien etenemistä voi seurata, mutta helppoa ja nopeaa tapaa seurata projekteihin kirjattuja resursseja ei ennestään ollut. Raportointityökalu listaa yrityksen kaikki keskeneräiset projektit, joista näkee niihin kirjatut tunnit selkeästi visualisoituna. Työkalun avulla voi myös seurata henkilöstön kuormitusta.

Raportointityökalu on Vue.js-sovelluskehyksellä toteutettu web-sovellus, jonka taustalla on REST-rajapinta. Rajapinta käyttää tietonaan JSON-muotoista dataa. Työkalun lopullisen implementoinnin yhteydessä luotiin käyttäjä yrityksen toiminnanohjausjärjestelmän (Visma Severan) rajapintaan, josta kaikki yrityksen projekteihin liittyvä tieto noudettiin. Opinnäytetyössä käytettiin kuitenkin valmiiksi kirjoitettua testidataa, jolla voidaan helposti simuloida työkalun lopullista toimintaa.

Opinnäytetyön tuloksena saatiin luotua yksinkertainen suunnitelman mukainen raportointinäkymä, jonka tarkoituksena on listata projektien resurssit ja henkilöstön kuormitus helposti luettavaan muotoon. Työssä hyödynnettiin erikseen kirjoitettua testidataa, sillä näkymän liittäminen toiminnanohjausjärjestelmän rajapintaan ei vielä kehitysvaiheessa ollut tarpeellista.

Tulosten avulla voitiin todeta, että näkymä tehostaa projektien resursoinnin seurantaa ja täten projektien kannattavuuden mittaamista.

Avainsanat (asiasanat)
Visualisointi, toiminnanohjausjärjestelmä, kannattavuus, REST API, Vue.js

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Girsén, Lauri	Bachelor's thesis	April 2020
		Language of publication: finnish
	Number of pages 24	Permission for web publication: x

Title of publication

Development of a visualization tool

Degree programme

Information Technology and Communication Technology

Supervisor(s)

Niemi, Kari

Assigned by

Into-Digital Oy

Abstract

The objective of this bachelor's thesis was to design and create an application that would visualize data gathered from the company's enterprise resource management system to simplify resource management and help measure the profitability of various ongoing projects within the company. An ERP (enterprise resource management system) is used within the company, but it doesn't offer a quick way of tracking project resource management. The application lists all ongoing projects and shows the current work hours that have been assigned to each project individually, with the core focus being on quick and easy access to the data and effective visualization.

The application is a simple web page created using a JavaScript framework called Vue.js. The backend is a REST API that receives JSON data. During the actual implementation of the application for the company, a user would be created for the company's enterprise resource planning system (Visma Severa), from which the data would be pulled from. In the case of this bachelor's thesis, however, premade test data was used that would effectively simulate the "real" data pulled from the ERP.

As a result of the bachelor's thesis, an application was developed that visualizes project resource data in such a way that would simplify resource management and help measure profitability. The application utilizes premade test data, as there was no need to use the interface of the enterprise resource system during development.

Keywords/tags (subjects)

Visualization, enterprise resource management system, project profitability, REST API, Vue.js

Miscellaneous (Confidential information)

Sisältö

Kuviot

1 Lähtökohdat

1.1 Toimeksiantaja

Into-Digital Oy on vuonna 2002 perustettu digitaalisen markkinoinnin yritys, joka toteuttaa asiakkaille monenlaisia digitaalisia palveluita. Into-Digital tunnetaan erityisesti verkkosivuprojektien toteuttamisesta WordPress -julkaisujärjestelmän päälle, mutta yrityksen osaamiseen kuuluvat myös muun muassa mobiilisovellukset ja display-mainokset. (Etusivu - Into-Digital n.d.)

Yrityksellä on kaksi toimistoa, joista toinen on Jyväskylässä ja toinen Helsingissä. Jyväskylän toimisto vastaa projektien teknisestä toteutuksesta, ja Helsingin toimistolla tapahtuu myyntiin ja projektinhallintaan liittyvä puoli. Yritys työllistää tällä hetkellä yhteensä 14 työntekijää, joista kolme on töissä Helsingin toimistossa. (Yhteys – Into-Digital n.d.)

1.2 Tausta

Yrityksille on tärkeää pystyä optimoimaan liiketoimintaansa. Yksi tärkeimmistä kontribuoivista tekijöistä liiketoiminnan optimointiin liittyen on projektien kannattavuuden mittaaminen, erityisesti henkilöstön resursoinnin ja kuormituksen seurannan näkökulmasta. Into-Digitalin käytössä tällä hetkellä on Visma Severa -toiminnanohjausjärjestelmä, jonka avulla projekteja voidaan resursoida, aikatauluttaa ja niiden etenemistä seurata. Tämänhetkisessä tilassaan toiminnanohjausjärjestelmä sisältää kuitenkin joitain puutteellisuuksia, jotka tekevät kannattavuuden mittaamisesta hitaampaa ja hankalampaa, kuin mitä se mahdollisesti voisi olla. Hyödyntämällä järjestelmän tarjoamaa rajapintaa ja käyttämällä järjestelmään kirjattua tietoa voidaan luoda erillinen sovellus, jonka avulla projektien resursseja voidaan visualisoida selkeämmin ja täten tehdä seurannasta yksinkertaisempaa ja nopeampaa.

1.3 Tehtävä ja tavoitteet

Opinnäytetyö on toiminnallinen kehittämistyö. Työn päätavoitteena oli luoda yrityksen sisäiseen käyttöön raportointityökalu, joka visualisoi toiminnanohjausjärjestelmästä kerättyä tietoa henkilöstön resursoinnin ja kuormituksen seurannan näkökulmasta. Käyttämällä työkalua projektin nykyisen tilanteen sekä sen historian seurantaa voitaisiin tehostaa, ja täten projektien kannattavuutta voitaisiin mitata entistä tehokkaammin, mikä puolestaan nopeuttaisi liiketoimintaa.

Opinnäytetyön toisena tavoitteena oli tutkia mahdollisia menetelmiä, jotka tekevät projektien resurssien seurannasta selkeämpää. Tähän voivat vaikuttaa monet tekijät, kuten esimerkiksi se, millä tavoin informaatiota visualisoidaan työkalun käyttäjälle.

Työkalu on RESTful web-sovellus, joka on toteutettu käyttämällä Vue.js-sovelluskehystä. Toiminnanohjusjärjestelmän tarjoama rajapinta, johon työkalu lopullisen implementaation yhteydessä liitetään, antaa kehittäjille mahdollisuuden hyödyntää järjestelmään kirjattua dataa, kuten projektien resursseja. Rajapintaa ei kuitenkaan vielä ollut tarpeen ottaa käyttöön opinnäytetyön kehitysvaiheessa. Työkaluun voidaan lukea testidataa, jonka tarkoituksena on simuloida rajapinnasta saatua oikeaa tietoa. Työkalun varsinaisen käyttöönoton yhteydessä toiminnanohjausjärjestelmään voidaan luoda tarvittavat tunnukset ja tätä kautta implementoida oikea data työkalun luettavaksi.

2 Kannattavuus

2.1 Perusteet

Kannattavuus on yritystoiminnan käsite, joka on yksi kolmesta perustekijästä yritystoiminnan toimintaedellytyksiä mitattaessa. Sitä pidetään usein tärkeimpänä talouslaskennan mittarina, sillä heikko kannattavuus tarkoittaa pidemmän päälle toiminnan

lopettamista. Yritys tuottaa tuolloin tappiota ja joutuu käyttämään omaa pääomaansa. (Tunnuslukuopas n.d.)

2.2 Visualisointityökalu

Visualisointityökalun fokus on pääosin projektien henkilöstön resurssoinnin ja kuormituksen seurannassa. Työkalun avulla pyritään saamaan parempi kuva projektien resursoinnin tilanteesta. Oikeanlainen resursointi mahdollistaa käyttöasteen optimoinnin yrityksen kasvun ja kannattavuuden näkökulmasta. (Mitä projektien resursointi oikeastaan tarkoittaa ja miten sitä kannattaa tehdä? n.d.)

Avaintekijät tehokkaampaan kannattavuuden mittaamiseen työkalun tapauksessa ovat nopeus ja helppokäyttöisyys. Data visualisoidaan käyttäjälle siten, että projektin tilanteesta saa selkeän kuvan yhdellä silmäyksellä. Informaatio on myös koottu yhteen paikkaan, joka nopeuttaa sen saatavuutta.

3 Työkalut ja teknologiat

3.1 Visma Severa

Visma Severa on toiminnanohjausjärjestelmä, joka julkaistiin vuonna 2004. Severa tarjoaa yrityksille alustan, johon voi koota kaiken projektien hallintaan liittyvän hajanaisen tiedon yhteen selkeään paikkaan ja täten selkeyttää liiketoimintaprosesseja. Toiminnanohjausjärjestelmän päätehtävänä on integroida erilaisia tehtäviä yrityksen sisäisessä toiminnassa kuten tuotantoa, laskutusta sekä resurssointia. Järjestelmän implementaation kautta pyritään tehostamaan yrityksen liiketoimintaa sekä projektien seurantaa. (Visma Severa n.d.)

Visma Severa tarjoaa kehittäjien käytettäväksi avoimen ohjelmointirajapinnan (API), jota hyödyntämällä kehittäjillä on mahdollisuus luoda kolmannen osapuolen sovel-

luksia hakemalla ja integroimalla toiminnanohjausjärjestelmään kirjattua tietoa. Severan API käyttää SOAP-pohjaista rajapintaa, mutta siirtyy käyttämään REST-pohjaista rajapintaa vuoden 2020 aikana. (Severa API n.d.)

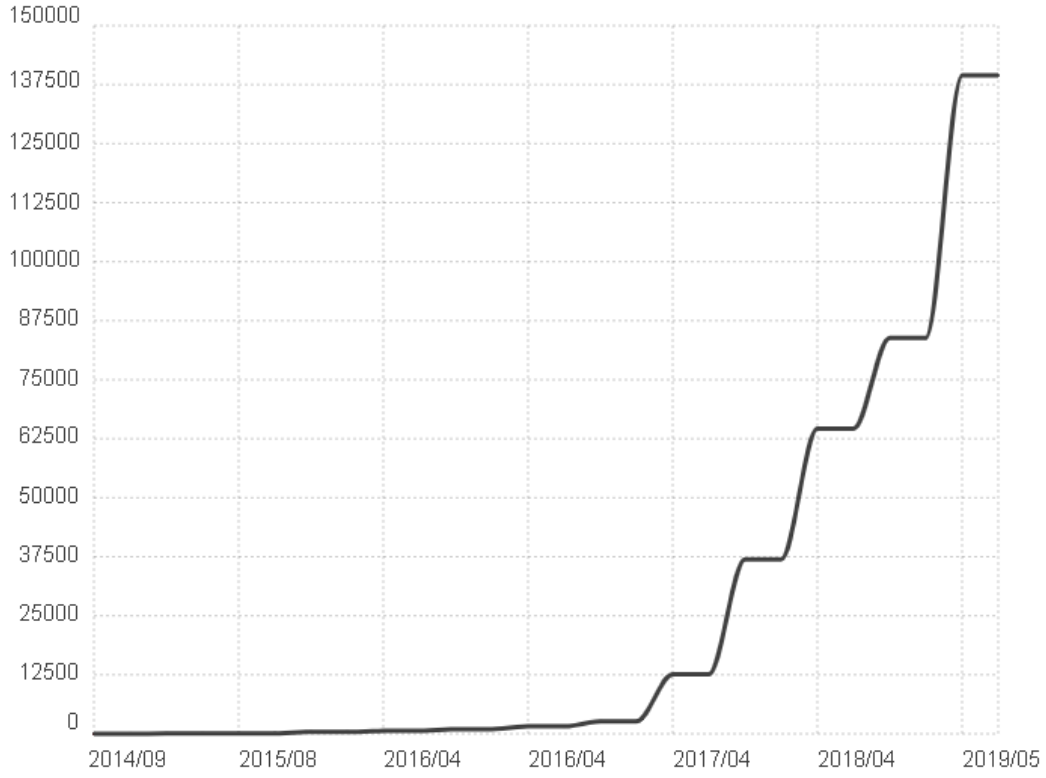
Rajapinnan voi ottaa käyttöön Severan hallintapaneelin kautta. Rajapinnan käyttäjälle tulee antaa tarvittavat käyttöoikeudet ja käyttäjän tulee luoda uniikki API-avain, jota tarvitaan palvelun käyttämiseen ja yhteyden muodostamiseen rajapintaan. (Visma Severa API n.d.)

3.2 Vue.js

3.2.1 Perusteet

Vue.js on avoimen lähdekoodin JavaScript front end -sovelluskehys, joka perustuu model-view-viewmodel -arkkitehtuurimalliin (MVVM). Sovelluskehys on Evan Youn vuonna 2014 kehittämä, ja sitä käytetään erityisesti interaktiivisten web-käyttöliittymien sekä single page -verkkosivustojen kehittämiseen**.** Vuen ydinkirjasto keskittyy päämääräisesti MVVM-mallin view-kerrokseen, joka tekee sen laajentamisesta helppoa. Tämän vuoksi Vue tunnetaan niin sanottuna progressiivisena sovelluskehyksenä, jolla tarkoitetaan sen joustavuutta ja yksinkertaisuutta integroidessa muiden sovelluskehysten ja kirjastojen kanssa. (Introduction n.d.)

Suosioltaan Vue.js oli yksi nopeimmin kasvavista sovelluskehyksistä vuonna 2018, ja sen käyttömäärä on noussut tasaisesti aina tähän päivään asti (ks. kuvio 1). Verrattuna kuitenkin muihin samankaltaisiin sovelluskehyksiin kuten Angulariin ja Reactiin, Vuen markkinaosuus on vielä huomattavasti pienempi. (Meet Vue.js – The Progressive Framework 2019.)



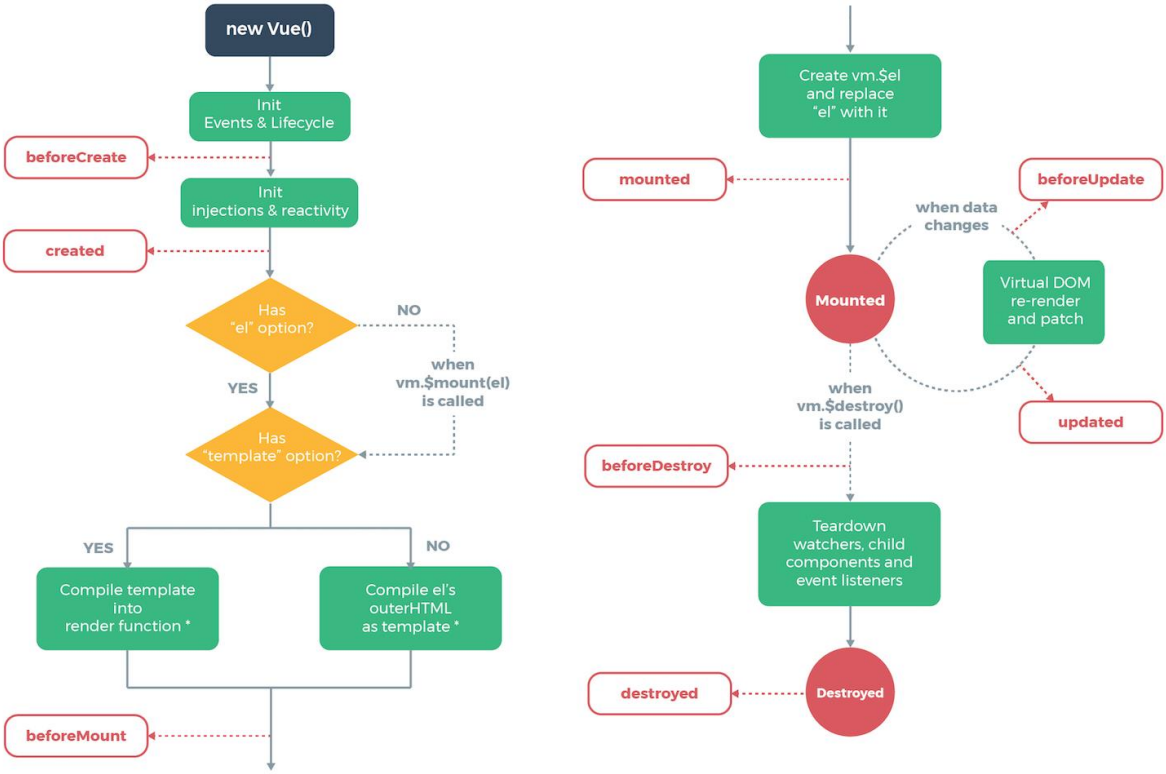
Kuvio 1. Vue.js-sovelluskehyksellä luodut verkkosivut vuosina 2014-2019 (Vue Usage Statistics 2020)

Sovelluskehysten päämääräisenä tavoitteena on antaa kehittäjälle valmis perusta, jonka pohjalta sovelluksia voi lähteä kehittämään järjestelmällisemmin. Perusta sisältää runsaasti valmiita erilaisia ominaisuuksia, joita kehittäjän ei tarvitse itse lähteä kirjoittamaan uudestaan. Useat sovelluskehykset seuraavat erilaisia arkkitehtuurimalleja (Vuen tapauksessa model-view-viewmodel), jotka ovat valmiiksi räätälöity tietynlaisia käyttötarkoituksia varten. Täten Vue, kuten myös monet muut JavaScript-sovelluskehykset, auttavat kehittäjiä luomaan web-sovelluksista helposti ylläpidettäviä ja testattavia. (What is a JavaScript Framework? n.d.)

3.2.2 Vue-instanssi ja elinkaaren tapahtumat

Jokainen vue-sovellus alkaa vue-instanssilla. Vue-instanssi käy aina läpi tietyt vaiheet sen luomisvaiheessa, joihin kuuluvat esimerkiksi template-mallin rakentaminen ja instanssin lisääminen DOM-puuhun. Näitä eri vaiheita voidaan ajatella vuen elinkaarena. Vue-instanssin elinkaaren etenemisen yhteydessä voidaan suorittaa erilaisia

elinkaaren tapahtumafunktioita, jotka mahdollistavat koodin ajamisen elinkaaren tietyissä vaiheissa. Funktioiden suoritusjärjestys on havainnollistettu kuviossa 2. (The Vue Instance n.d.)



Kuvio 2. Vue-instanssin elinkaaren tapahtumafunktiot (Gore n.d)

3.2.3 Vue CLI

Vue CLI on valmis kokoelma komponentteja Vue.js-projektien luomiseen ja hallintaan. Kokoelma on asennettavissa NPM-pakettina, ja se tarjoaa kehittäjille vaivattoman tavan luoda uusia Vue.js-projekteja ilman ylimääräistä konfiguraatiota, ja täten tehostaa projektien kehittämistä ja hallintaa. (Overview 2019.)

Paketti koostuu useasta eri komponentista. Olennaisimpiin komponentteihin kuuluvat CLI service NPM-paketti sekä komentorivityökalu, jonka avulla kehittäjä voi luoda

uuden projektin tai nopeasti testata projektia komentoriville kirjoitetuilla komennoilla. CLI Service on runtime-riippuvuussuhde, joka perustuu webpack-kehityspalvelimeen. Komponentti sisältää webpack-laajennuksen ydinominaisuuksien lisäksi sisäiset webpack-konfiguraatiot, jotka ovat jo valmiiksi optimoituja useita erilaisia sovellustyyppejä varten. (Overview 2019.)

Vue CLI sisältää myös graafisen käyttöliittymän projektien hallinnointia varten, joka mahdollistaa erilaisten tehtävien ajamisen helppokäyttöisen käyttöliittymän kautta. (Overview 2019.)

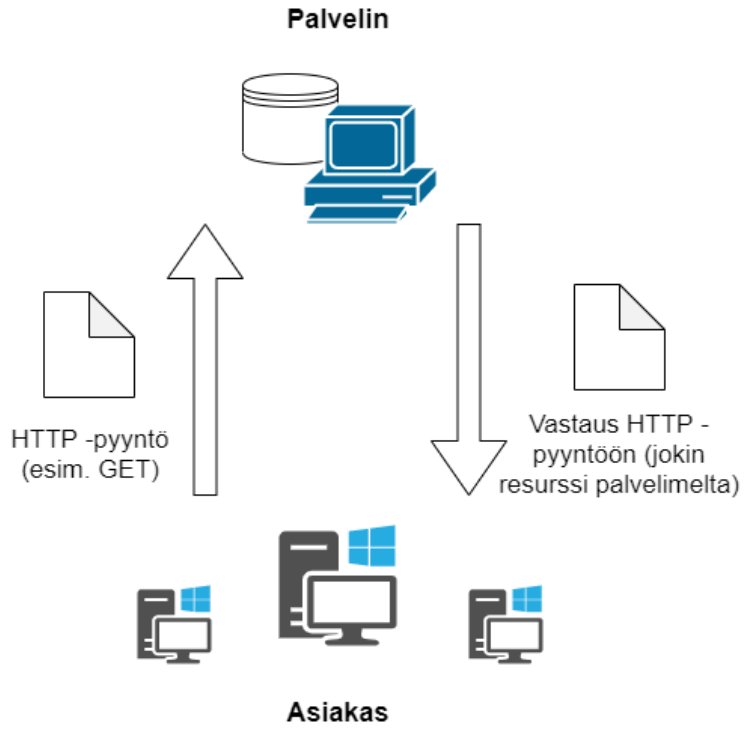
3.3 REST API

API:lla, eli ohjelmointirajapinnalla, tarkoitetaan määritelmää, jonka mukaan sovellukset voivat lähettää pyyntöjä toistensa välillä sekä vaihtaa informaatiota keskenään. REST API:lla tarkoitetaan tietynlaista ohjelmointirajapinnan määritelmää, joka perustuu representational state transfer -arkkitehtuurimalliin. Malli on kehitetty web-sovellusten luomista varten, ja se määritellään tiettyjen arkkitehtuurillisten ehtojen mukaisesti. (Rajput 2016.)

Yksi mallin tärkeimmistä määrittelevistä tekijöistä on asiakas-palvelin -malli, jolla tarkoitetaan roolien selkeää jakoa asiakkaan ja palvelimen välillä (ks. kuvio 3). Asiakas vastaa ainoastaan pyynnön tekemisestä ja palvelin puolestaan pyynnön toteuttamisesta. Palvelin- ja asiakaspuolta voidaan täten kehittää itsenäisesti, mikä parantaa koodin siirrettävyyttä sekä pitää rakenteen selkeänä. Toinen tärkeä määrittävä tekijä REST-arkkitehtuurissa on pyyntöjen tilattomuus. Jokainen palvelimelle tehty pyyntö on täysin itsenäinen ja erillään muista pyynnöistä. Tämä tarkoittaa sitä, että jokaisen yksittäisen pyynnön tulee sisältää kaikki tarvittava informaatio siihen liittyen. Palvelin ei siis säilö mitään asiakkaan nykyiseen tilaan liittyvää. (Mikkonen 2017.)

Kahden RESTful web-sovelluksen välinen kommunikaatio tapahtuu http-protokollan yli, jossa sovellukset hyödyntävät erilaisia valmiita http-metodeja välitetyn informaation kuvaamiseen toistensa välillä. Esimerkiksi GET-metodilla välitetty http-pyyntö palvelimelle palauttaa rajapinnasta resurssin, joka voi olla esimerkiksi JSON- tai XML-

tyyppisessä muodossa. Muita olennaisia metodeja ovat esimerkiksi POST, jota käyttämällä rajapintaan voidaan lisätä uutta informaatiota, ja DELETE, jota käyttämällä rajapinnasta voidaan poistaa informaatiota. Nämä useimmiten käytetyt metodit vastaavat standardin mukaisia CRUD-operaatioita (create, read, update, delete). Esimerkkejä RESTful web-sovelluksista ovat Twitter, Facebook ja Amazon. (Rajput 2016.)



Kuvio 3. Asiakas-palvelin mallin toimintaperiaate

3.4 SOAP API

SOAP API (Simple Object Access Protocol) tarkoittaa ohjelmointirajapinnan määritelmää, joka perustuu SOAP-protokollaan. Toisin kun REST, joka on ainoastaan arkkitehtuurimalli web-sovellusten luomista varten, SOAP on kokonaan oma protokollansa, joka määritellään omien standardiensa mukaisesti. Nämä standardit ovat huomattavasti tiukempia verrattuna REST-arkkitehtuurimallin määreisiin. Tietyt tekijät, kuten

tietoturvallisuus ja transaktioiden luotettavuus rajapinnassa, ovat huomattavasti tärkeämmässä asemassa SOAP-protokollaa käytettäessä. REST API tarjoaa huomattavasti enemmän joustavuutta web-palvelun kehittämisessä, kun taas SOAP API tarjoaa tarkempia ja tiukempia standardeja. (Wodehouse 2017.)

Kommunikaatio asiakkaan ja palvelimen välillä SOAP-protokollaa käyttävässä rajapinnassa tapahtuu useimmiten http-protokollan yli, mutta toisin kuin REST-rajapinnassa, SOAP voi hyödyntää kommunikaatioon myös muita siirtoprotokollia. Siinä missä RESTful web-sovelluksella tehty pyyntö palvelimelle voi palauttaa useaa erityyppistä tietoa (JSON, HTML, XML), SOAP toimii ainoastaan XML-tyypisellä tiedolla. SOAP-protokollan yli palautuva tieto on myös huomattavasti raskaampaa ja vaatii enemmän kaistaa, sillä se sisältää huomattavasti enemmän informaatiota, kuten erilaisia otsikkotietueita. Keveytensä vuoksi, REST API:n yli palautuva tieto on verrattavissa yksinkertaiseen postikorttiin, kun taas SOAP API:n tieto on verrannollinen hieman raskaampaan kirjekuoreen. (What is SOAP API? 2019.)

3.5 Axios

Axios on JavaScript-ohjelmointirajapinta, joka mahdollistaa kommunikoinnin http-protokollan yli asiakkaan ja palvelimen välillä. Rajapintaa käyttämällä selain voi tehdä http-pyynnön palvelimelle, joka palauttaa promise-objektin. Promise tarkoittaa tässä tapauksessa http-pyynnön lopputulosta, joka on saatu joko onnistuneen tai epäonnistuneen asynkronisen operaation seurauksena. Riippuen palvelimen vastauksesta tämä voi olla joko hylätty tai toteutettu. (Kelhini 2019.)

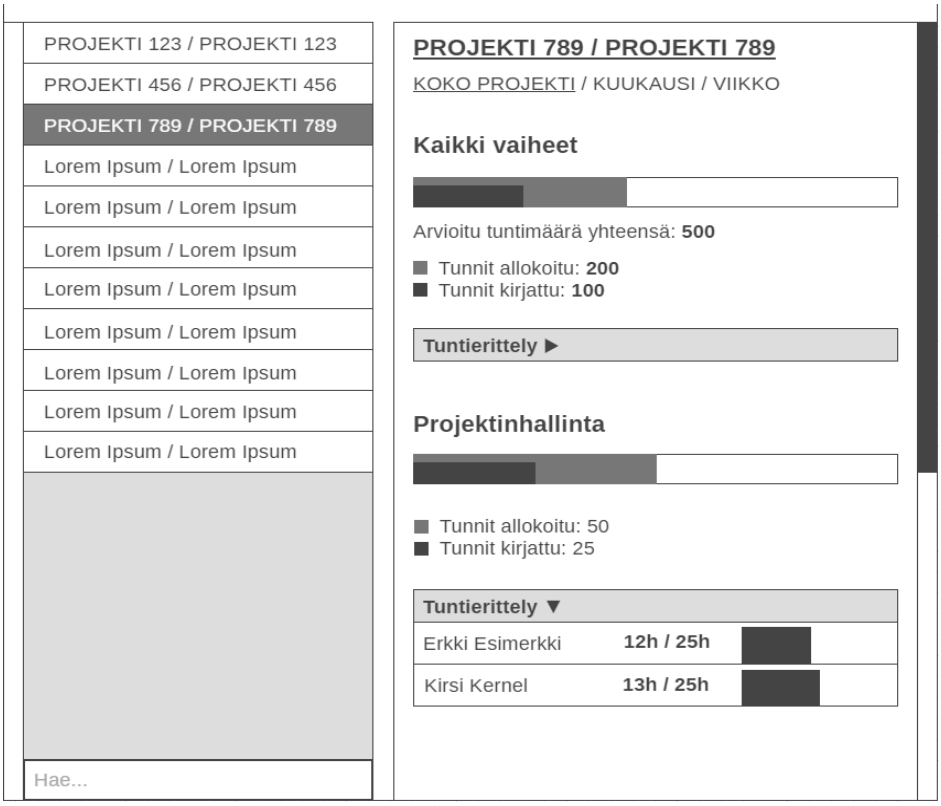
Axios pohjautuu XmlHttpRequest-rajapintaan, joka käyttää Ajax-tekniikkaa (Asynchronous JavaScript + XML). Ajax-tekniikalla tarkoitetaan http-pyyntöjen asynkronista toimintaa web-sivustoilla, eli sivusto voi siis lähettää ja vastaanottaa http-pyyntöjä “taustalla” ja päivittää sisältöä dynaamisesti päivittämättä koko sivua aina uudestaan. Tämä nopeuttaa sivuston latausaikoja sekä tekee käyttöliittymästä huomattavasti responsiivisemman. (Ajax developer guide 2020.)

4 CASE: Into-Digital

4.1 Näkymän suunnittelu

Työ alkoi raportointinäkymän suunnittelulla. Ennen työn käytännön toteutusta oli tärkeää ottaa huomioon mitä tietoa rajapinnasta varsinaisesti tarvittaisiin, ja miten sitä tultaisiin esittämään näkymässä.

Koska opinnäytetyön tavoitteena oli visualisoida projektien resursointia sekä henkilöstön kuormitusta, rajapinnasta tarvittiin kaikki projektin resursointiin liittyvä informaatio: projektien nimet, yksittäisen projektin kaikki vaiheet, projektiin kirjatut tunnit vaiheittain ja kaikki projektissa mukana olleet työntekijät, sekä työntekijöiden merkkaamat tunnit vaiheittain. Lopullisesta näkymästä luotiin rautalankamalli (ks. kuvio 4).



Kuvio 4. Näkymän rautalankamalli

Työkalu suunniteltiin siten, että se olisi mahdollisimman helppokäyttöinen. Informaatio on nopeasti löydettävissä yhdellä silmäyksellä, ja helppokäyttöisyyden ansiosta käyttäjä voi seurata projektien resursointia ja henkilöstön kuormitusta tehokkaammin. Resurssien tilannetta voi myös seurata eri aikavälein. Oletuksena näkymä näyttää resurssit koko projektin ajalta, mutta tarjoaa myös mahdollisuuden rajata aikaikkunaa edeltävään viikkoon tai kuukauteen.

Näkymän vasemmanpuoleiseen laitaan muodostuu lista kaikista yrityksen aktiivisista projekteista. Painamalla projektin nimeä listasta näkymän oikeanpuoleiseen laitaan ilmestyvät näkyviin projektin resurssit. Resurssit on eritelty projektin eri vaiheiden mukaisesti ja kunkin vaiheen alla on erikseen taulukko, josta löytää kunkin projektiin osallistuneen työntekijän kirjaamat resurssit.

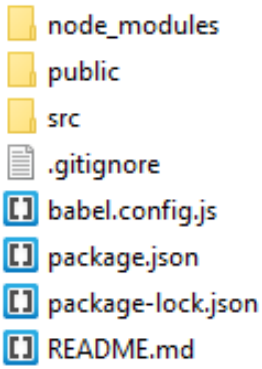
Koska opinnäytetyön puitteissa ei vielä ollut tarvetta käyttää toiminnanohjausjärjestelmän rajapintaa, visualisointia varten luotiin testidataa, jonka tarkoituksena oli simuloida rajapinnasta noudettua dataa. Tämän avulla saatiin jo kehitysvaiheessa hyvä kuva siitä, miltä näkymä tulisi näyttämään lopullisen implementaation yhteydessä.

4.2 Näkymän kehittäminen

4.2.1 Projektin luominen

Ennen kehitysvaiheen aloittamista, kehitystä varten tarpeelliset ohjelmistot tuli asentaa. Tärkeimmät ohjelmistot tämän työn tapauksessa olivat Axios ja vue-cli. Paketit asennettiin käyttämällä Node.js:n paketinhallintatyökalua (NPM), komennoilla `npm install @vue/cli` ja `npm install axios`. Asennusten jälkeen voitiin luoda uusi Vue.js projekti käyttämällä vue-cli -komentorivityökalua.

Uusi projekti luotiin komennolla `vue create <projektinimi>`. Ennen luomista työkalu tarjosi mahdollisuuden valita projektiin erilaisia riippuvuussuhteita (kuten babel ja ESLint) joko automaattisesti tai manuaalisesti. Oheisilla riippuvuussuhteilla ei ollut varsinaisesti mitään merkitystä opinnäytetyön kannalta, joten kaikki oletusriippuvuudet asennettiin automaattisesti. Komento loi kaikki projektin tiedostot automaattisesti uuteen kansioon (ks. kuvio 5).



Kuvio 5. Vue-projektin kansiorakenne

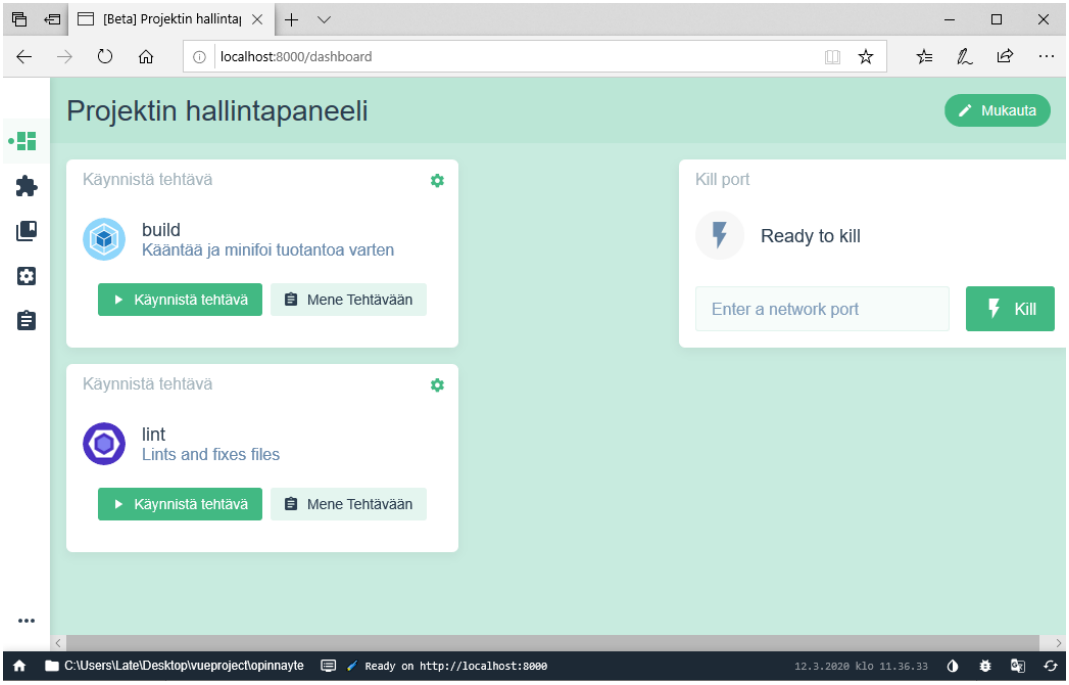
Public-kansio sisältää projektin staattiset sisällöt sekä index.html-tiedoston, joka toimii pohjana projektia rakentaessa. Projektin jakeluversion rakentamisen yhteydessä webpack prosessoi ja rakentaa tiedoston uudelleen lisäten kaikki tarvittavat linkit. Public-kansioon asetettuja muita staattisia sisältöjä ei ajeta webpack-laajennuksen läpi. Kehittäjällä ei ole tarvetta koskea kansion sisältöön lainkaan kehitysvaiheessa.

Src-kansio sisältää kaiken kehityksen kannalta olennaisen sisällön, kuten vue-komponentit, tyylitiedostot, skriptit sekä projektin mediatiedostot. Tärkein tiedosto kansion sisällä on App.vue. Tämä toimii vue-sovelluksen runkona, johon kaikki projektin komponentit sisällytetään ja kasataan.

4.2.2 Kehitysympäristö

Komennolla `vue-cli-service serve` käynnistettiin paikallinen kehityspalvelin, jota käyttämällä sovellusta voitiin testata. Palvelin avautui oletuksena porttiin 8080, joten projekti saatiin näkyviin selaimessa osoitteessa localhost:8080. Palvelin perustuu webpack-server -laajennukseen.

Vue-cli sisältää myös graafisen hallintapaneelin, jonka avulla kehittäjä voi helposti hallinnoida ja ylläpitää projekteja. Hallintapaneelin voi käynnistää komennolla `vue ui`. Komento luo ympäristön oletuksena porttiin 8000, jonka jälkeen sen saa auki selaimessa osoitteessa localhost:8000 (ks. kuvio 6).



Kuvio 6. Vue-projektin graafinen hallintapaneeli

Hallintapaneelin avulla kehittäjä voi ajaa erilaisia komentoja projektin sisällä, tarkastella projektin riippuvuuksia sekä hallinnoida uusia projekteja. Ensimmäiseen näkymään voi lisätä erilaisia widgetejä, jotka sisältävät olennaisimpia projektiin liittyviä tehtäviä, kuten tärkeitä ajettavia komentoja.

4.2.3 Komponentit ja rakenne

Näkymä jaettiin kahteen erilliseen komponenttiin: projektilistaan (ProjectList) sekä projektin resurssitietoihin (ProjectInfo). Molemmat komponentit sisällytettiin yhteen päätiedostoon (App.vue), joka sisältää projektin varsinaisen rungon. Tämä jako tekee projektin rakenteesta selkeämmän ja helpottaa sen jatkokehittämistä (ks. kuvio 7).

```
<template>
<div id="app">
  <ProjectList/>
  <ProjectInfo/>
  </div>
</template>

<script>
  import ProjectList from './components/ProjectList'
  import ProjectInfo from './components/ProjectInfo'
  import axios from 'axios'
  window.axios = axios
  export default {
    name: 'App',
    components: {
      ProjectList,
      ProjectInfo
    },
```

Kuvio 7. Komponenttien tuominen App.vue-tiedostoon

Sivun latautuessa näkymä noutaa testidatan palvelimelta. ProjectList-komponentin mounted()-funktiossa tehdään http-pyyntö palvelimelle (ks. kuvio 8). Mounted() on elinkaaritapahtuma, joka ajetaan heti DOM-puun renderöinnin jälkeen. Http-pyynnön onnistuttua data asetetaan objektiin, joka sisältää kaiken tarvittavan visualisointia varten. Komponentti hakee projektien nimet objektista ja esittää ne näkymässä for-silmukalla. Projektin nimeä painamalla, data esitetään näkymän ProjectInfo-komponentissa.

```
mounted() {
  axios.get('api/data.json')
    .then(response => {
    this.projects = response.data.projects
  })
},
```

Kuvio 8. Http-pyyntö komponentin mounted()-elinkaaritapahtumassa

4.2.4 Datan välittäminen kahden komponentin välillä

Jokaiseen yksittäiseen elementtiin listan sisällä on sidottu funktio, joka ajetaan elementtiä painaessa. Kullakin elementillä on myös uniikki indeksiarvo, joka välitetään parametrina funktiolle. Tämän arvon avulla sovellus osaa hakea testidatasta oikean projektin datan.

Koska resurssien listaus tapahtuu kuitenkin erillisessä komponentissa, data täytyy välittää ProjectInfo-komponentille käyttämällä niin sanottua event bus -tapahtumaa. Event bus -tapahtuma on tässä tapauksessa yksinkertaisuudessaan uusi vue-instanssi, jota käyttämällä dataa voidaan liikuttaa komponentista toiseen. Instanssi luodaan main.js-tiedostossa, jossa se saadaan vaikuttamaan globaalisti. Muuttuja, joka sisältää instanssin tuodaan ProjectList-komponenttiin, jonka jälkeen sitä voidaan hyödyntää. Funktio, jossa event bus -tapahtumaa hyödynnetään, on esitetty kuviossa 9.

```
methods: {
  openProject: function(e) {
    this.selected = e;
    bus.$emit('updateinfo', this.projects[e])
  }
}
```

Kuvio 9. Datan välittäminen komponenttien välillä event bus -tapahtumalla

ProjectInfo-komponentti vastaanottaa datan tapahtumakuuntelijan kautta, joka on asetettu komponentin created()-elinkaaritapahtuman sisään. Projektin vaiheet käydään läpi for-silmukalla ja data asetetaan muuttujien sisään (ks. Kuvio 10).

Tapahtumakuuntelijan sisältämä koodi ajetaan aina, kun uutta projektia painetaan. Tämän vuoksi ennen for-silmukkaa kunkin muuttujan arvo alustetaan nollaksi, jonka jälkeen uuden projektin arvot asetetaan ja ne voidaan esittää näkymässä.

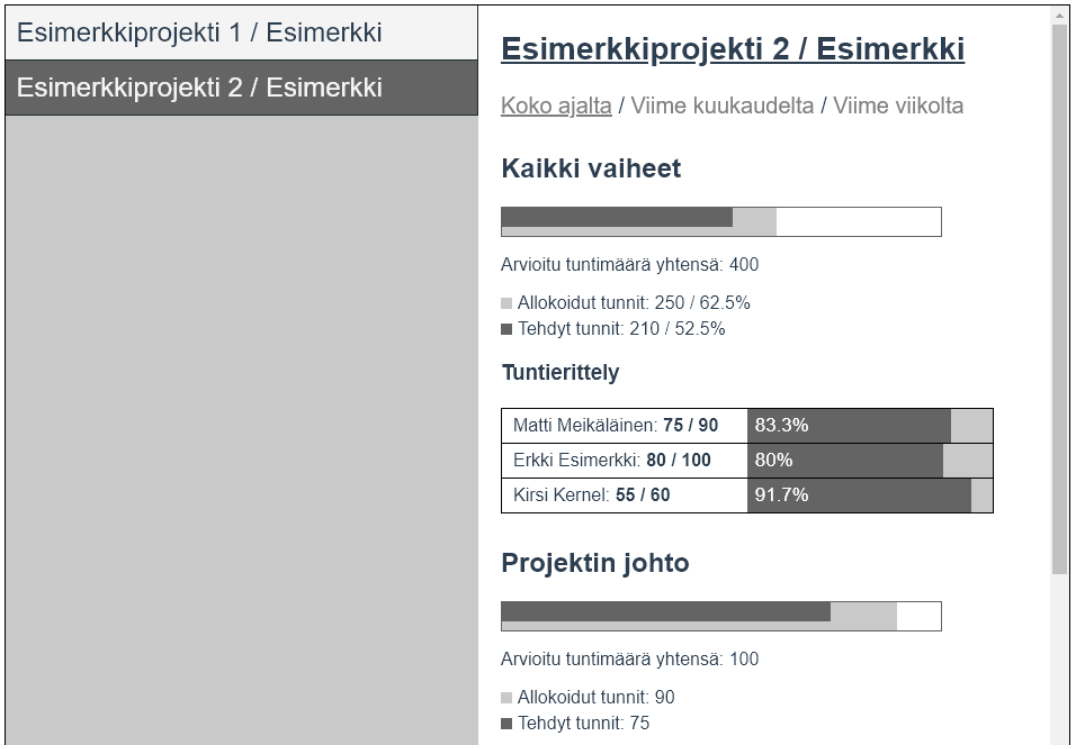
```
created () {
  bus.$on('updateinfo', (data) => {
    this.projectname = data.name
    this.steps = data.steps

    // states for data under all steps
    this.all_hours = this.alloc_hours = this.done_hours = this.monthly_alloc_hours =
    this.monthly_done_hours = this.weekly_alloc_hours = this.weekly_done_hours = 0;
    data.steps.forEach((step, i) => {
      this.all_hours += step.all_h
      this.alloc_hours += step.all.allocated_h
      this.done_hours += step.all.done_h

      this.monthly_alloc_hours += step.monthly.allocated_h
      this.monthly_done_hours += step.monthly.done_h

      this.weekly_alloc_hours += step.weekly.allocated_h
      this.weekly_done_hours += step.weekly.done_h

      this.step_alloc_percentage[i] = (step.all.allocated_h / step.all_h * 100)
      this.step_done_percentage[i] = (step.all.done_h / step.all_h * 100)
    })
    this.all_alloc_percentage = (this.alloc_hours / this.all_hours * 100)
    this.all_done_percentage = (this.done_hours / this.all_hours * 100)
  })
},
```

Kuvio 10. Tapahtumakuuntelija ProjectInfo-komponentin created()-elinkaaritapahtumassa

4.2.5 Datan visualisoiminen

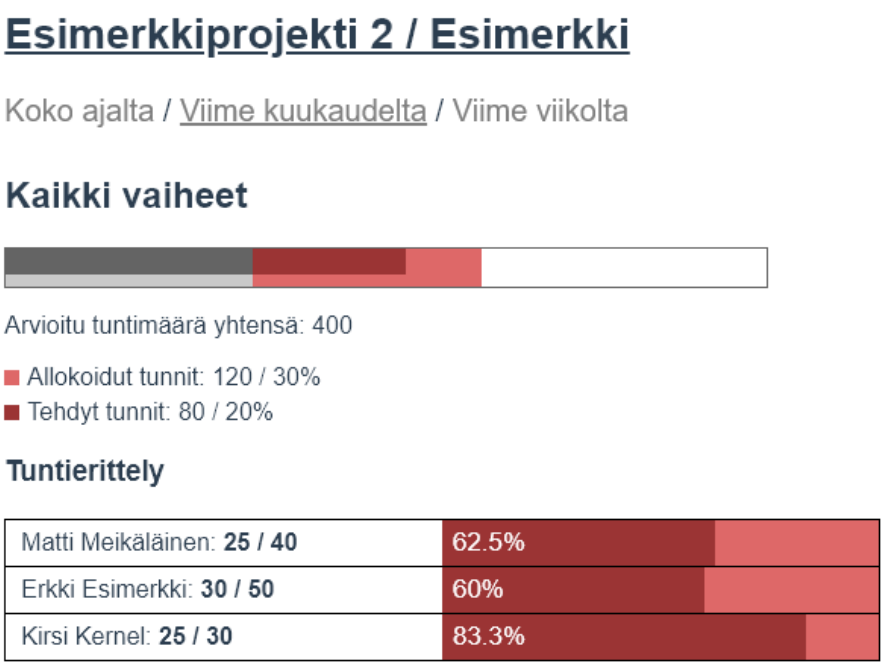
Datan visualisointi tapahtuu ProjectInfo-komponentissa siten, että kaikista ylimpänä esitetään koko projektin yhteinen resurssitilanne. Tämän alapuolella resurssit esite-

tään vaiheittain. Henkilöstön kuormitus esitetään taulukkona yhteensä koko projektilta sekä kunkin yksittäisen vaiheen alla erikseen. Data voidaan esittää näkymässä kolmella eri tavalla: kaikilta ajoilta, viime kuukaudelta ja viime viikolta (ks. kuvio 11).



Kuvio 11. Data visualisoituna näkymässä

Data esitetään näkymässä oletuksena kaikilta ajoilta. Painamalla otsikon alapuolella sijaitsevia linkkejä viime kuukauden tai viime viikon data piirtyy eri värisenä kokonaistuntien päälle, jonka tavoitteena on auttaa hahmottamaan tietyn ajanjakson tuntimäärää suhteessa koko projektiin. Henkilöstön kuormitusta kuvaava tuntierittelytaulukko päivittyy myös riippuen valinnasta (ks. kuvio 12).



Kuvio 12. Projektin resurssien listaaminen viime kuukauden tarkkuudella

Palkit, jotka visualisoivat tunteja ovat html-elementtejä, joille annetaan leveys css-tyylimääreenä. Tyylimääreen arvo on muuttuja, joka on sidottu kullekin elementille käyttämällä v-bind:style -ominaisuutta. Tämä mahdollistaa tyylin arvon muuttumisen dynaamisesti, eli aina muuttujan arvon vaihtuessa (ks. kuvio 13). Painettaessa eri ajanjakson linkkiä kutsutaan funktiota, joka päivittää muuttujan arvon. Tämän seurauksena palkki piirtyy välittömästi näkyviin halutulla tavalla.

```
<h2 class="step-name">Kaikki vaiheet</h2>
<div class="prog-all">
  <div v-bind:style="{width: all_alloc_percentage + '%'}" cla
  <div v-bind:style="{width: all_done_percentage + '%'}" clas

  <div v-bind:style="{width: monthly_alloc_percentage + '%',
  <div v-bind:style="{width: monthly_done_percentage + '%', r

  <div v-bind:style="{width: weekly_alloc_percentage + '%', r
  <div v-bind:style="{width: weekly_done_percentage + '%', ri
</div>
```

Kuvio 13. Inline-tyylin sitominen html-elementtiin

5 Tulokset ja pohdinta

5.1 Lopputulos

Opinnäytetyön lopputuloksena saatiin kehitettyä helppokäyttöinen pohja visualisointityökalulle, jonka avulla projektien etenemistä sekä henkilöstön kuormitusta voidaan seurata. Käyttämällä testidataa työkalu saatiin luotua suunnitelmien mukaisesti.

Visualisointityökalu toimii niin sanottuna proof of conceptina. Työssä luotua näkymää hyödyntämällä ja tarvittavalla jatkokehityksellä saadaan tehokas menetelmä projektien resursoinnin seurantaan. Avaintekijöitä tehokkuudessa ovat näkymän helppokäyttöisyys sekä nopeus, jolla käyttäjä saa kokonaisvaltaisen kuvan projektin resursoinnin tilanteesta.

5.2 Jatkokehitys

Visualisointityökalulla on runsaasti potentiaalia jatkokehitykselle. Koska työkalu ei ole millään tavoin yhteydessä yrityksen toiminnanohjausjärjestelmään, sen varsinainen käyttöönotto yrityksessä vaatii jatkokehitystä. Jotta työkalu voisi lukea dataa oikealla tavalla tulee luoda ylimääräinen skripti, joka ottaa yhteyden Severan rajapintaan. Skriptin tulisi noutaa kaikki tarvittava data rajapinnasta ja tallentaa se projektissa käytetyn testidatan muotoon.

Näkymän visuaalinen puoli tarjoaa myös mahdollisuuksia jatkokehitykselle. Datasta voitaisiin muun muassa piirtää kuvaajia, jotka entisestään helpottaisivat datan silmäiltävyyttä.

Työkaluun oli alunperin suunnitelmissa luoda myös toinen välilehti, joka listaisi yrityksen henkilöstön ja näyttäisi kunkin työntekijän kirjaamat työtunnit. Ominaisuus kuitenkin jätettiin pois lopullisesta opinnäytetyöstä ja lisätään mahdollisesti jatkokehityksen myötä.

5.3 Pohdinta

Opinnäytetyön tavoitteena oli luoda yritykselle työkalu, joka visualisoi yrityksen toiminnanohjausjärjestelmästä haettua projektien resursointiin liittyvää dataa. Vaikka yhteyttä järjestelmän rajapintaan ei kehitysvaiheessa luotu, työn aikana saatiin toteutettua suunnitelman mukainen ja toimiva visualisointityökalu. Jatkokehityksen myötä, näkymästä voidaan räätälöidä entistä tehokkaampi ja siihen voidaan lisätä lisää ominaisuuksia, jotka jäivät opinnäytetyöstä pois.

Opinnäytetyön suunnittelu alkoi syksyllä 2019 ja varsinainen kehitysvaihe alkoi heti vuodenvaihteessa. Työ eteni pääosin suunnitelmien mukaisesti ilman viivästymisiä. Vaikka toiminnanohjausjärjestelmää ei työssä hyödynnetty, sen rajapinnan tutkimiseen käytettiin aikaa. Tämä ei ollut kuitenkaan turhaa, sillä informaatiolle tulee olemaan käyttöä projektin jatkokehitystä ajatellen.

Vue.js-sovelluskehys ei ollut opinnäytetyön tekijälle entuudestaan tuttu, joten sen käyttämisessä oli oppimisen kannalta erityisen paljon hyötyä. Työ auttoi tekijää ymmärtämään myös useita muita teknologioita paremmin, kuten REST-rajapintaa sekä Severa-toiminnanohjausjärjestelmää.

Lähteet

Ajax developer guide. 2020. Ajax-dokumentaatiosivu. Päivitetty 12.1.2020. Viitattu 13.1.2020. https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

Introduction. N.d. Vue.js-sovelluskehyksen dokumentaatio. Viitattu 20.11.2019. https://vuejs.org/v2/guide

Kelhini, F. 2019. How to make HTTP requests like a pro with Axios. Julkaisu LogRocket-verkkosivustolla. Päivitetty 2.7.2019. Viitattu 13.1.2020. https://blog.logrocket.com/how-to-make-http-requests-like-a-pro-with-axios

Gore, A. N.d. Lifecycle hooks. Julkaisu O'Reilly-verkkosivustolla. Viitattu 24.3.2020. https://www.oreilly.com/library/view/full-stack-vuejs-2/9781788299589/c3bb3fdf-3850-4518-99d7-a281b4732c46.xhtml

Meet Vue.js – The Progressive Framework. 2019. Julkaisu Medium-verkkosivustolla. Päivitetty 12.3.2019. Viitattu 17.1.2020. https://medium.com/we-are-minds/meet-vue-js-the-progressive-framework-8f9a2c55acb7

Mikkonen, J. 2017. Rest on nettipalveluiden yhteinen kieli. Julkaisu tivi.fi-verkkosivustolla. Päivitetty 27.5.2017. Viitattu 12.1.2020. https://www.tivi.fi/uutiset/rest-on-nettipalveluiden-yhteinen-kieli/23703ab5-dd19-383e-a422-ebfc3d910583

Mitä projektien resursointi oikeastaan tarkoittaa ja miten sitä kannattaa tehdä?. 2018. Blogijulkaisu Visman verkkosivustolla. Päivitetty 9.6.2018. Viitattu 24.3.2020. https://psa.visma.fi/blog/mita-projektien-resursointi-oikeastaan-tarkoittaa-ja-miten-sita-kannattaa-tehda/

Morris, S. N.d. What is a JavaScript framework?. Blogijulkaisu Skillcrush-verkkosivustolla. Viitattu 20.11.2019. https://skillcrush.com/2018/07/23/what-is-a-javascript-framework

Overview. 2019. Vue.js-sovelluskehyksen dokumentaatio. Päivitetty 10.7.2019. Viitattu 24.3.2020. https://cli.vuejs.org/guide/

Palvelumme - Into-Digital. N.d. Viitattu 3.2.2020. Into-Digitalin verkkosivusto. https://into-digital.fi/palvelumme/

Rajput, D. 2016. What is REST? REST architecture and constraints. Julkaisu Dinesh on Java-verkkosivustolla. Päivitetty 15.8.2016. Viitattu 11.12.2019. https://www.dineshonjava.com/what-is-rest-and-rest-architecture-and-rest-constraints

Severa API. N.d. Visma Severa API:n verkkosivusto. Viitattu 17.1.2020. https://psa.visma.fi/api-severa/

The Vue Instance. N.d. Vue.js-sovelluskehyksen dokumentaatio. Viitattu 24.3.2020. https://vuejs.org/v2/guide/instance.html

Tunnuslukuopas. N.d. Artikkeli Alma Talent-verkkosivujen tietopalvelussa. Viitattu 31.1.2020. https://www.almatalent.fi/tietopalvelut/tunnuslukuopas

Visma Severa. N.d. Visma Severan verkkosivusto. Viitattu 17.1.2020. https://psa.visma.fi/

Visma Severa API. N.d. Visma Severa API:n dokumentaatio. Viitattu 17.1.2020. https://support.severa.com/hc/en-us/articles/215091363

Vue Usage Statistics. 2020. Vue.js-sovelluskehyksen käyttömäärät builtwith-verkkosivustolla. Viitattu 17.1.2020. https://trends.builtwith.com/javascript/Vue

What is SOAP API?. 2019. Julkaisu Altexsoft-verkkosivustolla. Päivitetty 20.8.2019. Viitattu 18.1.2020. https://www.altexsoft.com/blog/engineering/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest

Wodehouse, C. 2017. SOAP vs. REST: A Look at The Two Different API Styles. Julkaisu Upwork-verkkosivustolla. Päivitetty 19.4.2017. Viitattu 18.1.2020. https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis

Yhteys – Into-Digital. N.d. Into-Digitalin verkkosivusto. Viitattu 3.2.2020. https://into-digital.fi/ota-yhteytta/