Sami Kaskinen

Taloudellisten lukujen reaaliaikainen visualisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

19.4.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Sami Kaskinen Taloudellisten lukujen reaaliaikainen visualisointi 34 sivua 19.4.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Toimitusjohtaja Jesse Peurala Lehtori Ilkka Kylmäniemi

Insinöörityön tavoitteena oli toteuttaa asiakasyritykselle kojelauta, joka visualisoi reaaliaikaista dataa tuntikirjausjärjestelmästä. Asiakasyritykselle tärkein lisäominaisuus kojelaudassa oli tiedon reaaliaikaisuus ja kojelaudan reagointi ja varoitussignaali yritykselle, jos datan arvot vaihtelevat huonoon suuntaan. Insinöörityössä selvitettiin kojelautojen historiaa, tärkeimpiä talouden lukuja, datan analysointia ja visualisointia.

Insinöörityön kojelauta toteutettiin Dashing-ohjelmistokehyksellä ja itse ohjelmointi pääosin Ruby-ohjelmointikielellä. Dashing-kojelautaa täydennettiin käyttäjien siihen lisäämillä lisäosilla, kuten erilaisilla Googlen visualisointityökaluilla. Tietoa saatin haettua tuntikirjausjärjestelmästä, mutta sitä ei saatu lisättyä lopulliseen kojelautaan, koska tekijän Ruby-ohjelmointikielen tietämys ei riittänyt näin haastavaan ja laajaan projektiin. Näin ollen kojelauta ei tullut valmiiksi, mutta työ antoi hyödyllistä tietoa ja oppimiskokemuksia tulevaisuuden haasteita ja projekteja varten.

Insinöörityöraporttiin koottiin paljon tietoa kojelautojen toiminnasta ja visualisointien toimivuudesta. Etuna mahdollisten eri kojelautojen selvityksessä oli, että asiakasyritys voi valita niistä mieluisimman, jos haluaa jatkossa kehittää paremman kojelaudan, kuin sillä on tällä hetkellä käytössä. Lisäksi työn edetessä löytyi parempia tapoja, joilla olisi voinut suorittaa käytännön projektin. Tästä esimerkkinä Razorflow, jota olisi voitu kehittää tutummilla ohjelmointikielillä ja dokumentointia olisi löytynyt helpommin. Jotta insinöörityö olisi valmistunut onnistuneesti, olisi pitänyt valita jokin toinen vaihtoehto kojelaudan ohjelmistokehykseksi, sellainen, jonka ohjelmointikieli olisi ollut tuttu entuudestaan ja josta löytyisi paljon hyvälaatuista dokumentaatiota.

Avainsanat	data, visualisointi, kojelauta, data-analyysi

Authors Title Number of Pages Date	Sami Kaskinen Real time visualization of economic figures 34 pages 19 April 2017
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Jesse Peurala, CEO Ilkka Kylmäniemi, Principal Lecturer

The main goal of the thesis was to program a dashboard for the client that would visualize data in real time from their work hour registration system. The most important feature of the dashboard would be that the data is in real time and that the dashboard would react to and warn the company about the fluctuation of followed data. The thesis covers the history of dashboards, the most important economic figures to a company and also, how data can be analyzed and visualized.

The dashboard made for the thesis was done with the framework Dashing and the programming was done mostly on Ruby. Dashing was extended with user-created add-ons which included different visualization tools from Google. Data was fetched from the work hour registration system but it could not be implemented to the dashboard. This is because of the lack of knowledge in Ruby programming in a project this big and challenging. The thesis was not completed because of the lack of programming knowledge, but it gave useful information and learning experiences for future projects and challenges. The thesis provided a lot of information about the functions of dashboards and how to create effective visualizations.

Based on this research the client can choose one of the possible dashboard frameworks if they want to program a better dashboard than they already have. When the project progressed, better ways of completing it was found. A good example of this is a framework called Razorflow which had already familiar programming languages and better documentation. The project would have succeeded if we could have chosen another framework for the job with lots of documentation and familiar programming languages.

Keywords	data, visualization, dashboard, data analysis

Sisällys

1 Johdanto

Insinöörityön tavoitteena on toteuttaa kojelauta, joka hakee reaaliaikaista tietoa tuntikirjauspalvelusta nimeltä Toggl ja esittää sen kojelaudalla visualisointeja hyväksikäyttäen. Toteutettavien sivujen tulisi olla selkeät, ja niiden tulisi esittää dataa reaaliajassa, jotta ei tarvitsisi tutkia vanhaa tietoa. Insinöörityön tilaajana on vuonna 2012 perustettu Fraktio, joka on ohjelmistoyritys Helsingin Pasilassa. Se tarjoaa muun muassa ohjelmistokehitystä, ylläpitoa, koulutusta, liiketoiminnan kehittämistä ja käyttökokemussuunnittelua. Yrityksen toiminta perustuu paljolti avoimuuteen ja läpinäkyvyyteen monissa yrityksen asioissa, joten se jakaa muun muassa yrityksen taloudellisia lukuja koko yritykselle.

Asiakasyritykselle tärkeä työkalu on taloudellisia lukuja esittävä kojelauta. Fraktiolla on jo toimiva ja hyvä kojelauta, mutta siitä puuttuu tärkeitä ominaisuuksia, kuten muun muassa hälytykset datan heittelehtiessä, kojelaudan värien reagointi lukuihin ja se, että tietoa ei esitetä reaaliajassa. Työllä ei ole kiire, koska käytössä on toimiva kojelauta, mutta tiedon reaaliaikaisuus olisi suuri edistysaskel kojelaudassa. Näin ollen työ on tärkeää. Aihe valittiin, koska Fraktio tarjosi sitä ja tarvitsi uuden ja paremman kojelaudan.

Insinöörityössä syvennytään kojelautojen historiaan: niiden edeltäjiin, tärkeisiin tietoteknisiin ja aatteellisiin edistysaskeliin ja kokeiluihin vuosien varrella. Kojelautojen historian ymmärtäminen on hyödyllistä, jotta ymmärtää, mitkä ovat niiden tärkeimmät ominaisuudet ja mikä luokitellaan toimivaksi ja hyödylliseksi kojelaudaksi. Lisäksi tutustutaan yrityksen tärkeimpiin mittareihin, joita tutkimalla selviää, kuinka hyvin yritys pärjää. Näiden asioiden ymmärtäminen on tärkeää visualisointeja luotaessa, jotta voidaan valita oikeat visualisointityylit talouden lukuja varten.

Insinöörityössä perehdytään datan visualisointiin ja siihen, mitkä ovat hyviä käytäntöjä visualisointeja luotaessa. Samalla perehdytään siihen, kuinka ihminen sisäistää dataa ja kuinka se tapahtuu helpoiten ja tehokkaimmin. Näitä tietoja yhdistelemällä on helpompaa luoda tehokkaita ja selkeitä visualisointeja insinöörityötä varten. Lisäksi syvennytään data-analyysiin ja siihen, kuinka hankittu data tulisi hyödyntää, mikä data on tärkeää ja mitä yritykset hyötyvät datan analysoinnista. Työn suunnitteluvaiheessa raportissa perustellaan, miksi käytettiin valittuja tekniikoita ja työkaluja, sekä esitellään niiden ominaisuudet. Raportin loppuosassa pohdi-

taan missä onnistuttiin ja mitä voitaisiin tehdä tulevaisuudessa ja mitä olisi voitu tehdä paremmin.

2 Päätöksenteon tukijärjestelmät

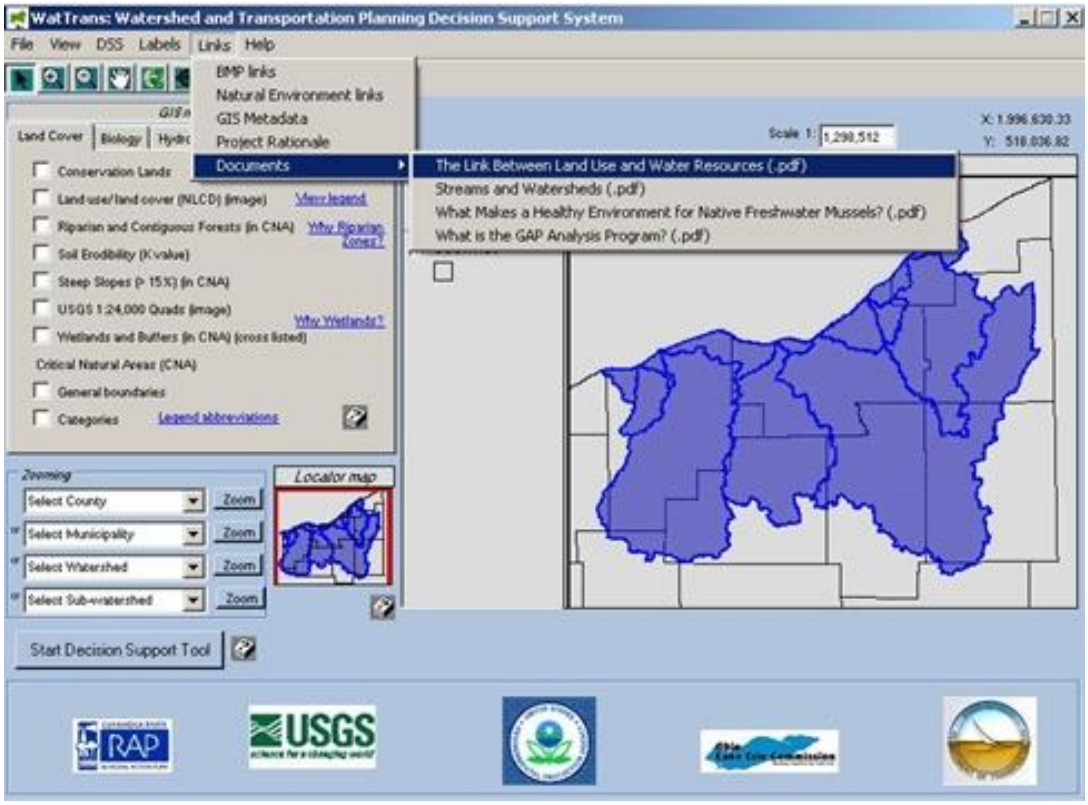
Nykyään yritetään automatisoida mahdollisimman paljon asioita, ja näihin kuuluu esimerkiksi talousraporttien laatiminen. Ne laaditaan useasti käsin tai saadaan kirjanpitojärjestelmästä. Monesti lopullinen raportti ei ole kovin informatiivinen ja sen tutkiminen ei ole kovin mielekästä, koska se voi olla vain lista tai taulukko numeroita, joiden tutkimiseen pitää käyttää paljon aikaa, jotta saisi tärkeää tietoa ymmärrettyä. Monet yritykset käyttävät vielä tätä perinteistä tapaa seurata talouttaan ja yrityksen tärkeitä lukuja, mutta uusia yrityksiä perustetaan koko ajan. Uudet yritykset kaipaavat innovaatioita ja uudistuksia vanhoihin järjestelmiin, mikä luo kysyntää eri visualisoinneille ja yrityksen sisäisille kojelaudoille. Tähän voi auttaa tiedon visualisointi tai esitys siihen suunnitellussa kojelaudassa. Kojelaudat keskittävät tärkeän tiedon yhteen paikkaan, jossa se esitetään selkeällä tavalla. Tämä helpottaa tiedon lukemista nopeammin, eikä tarvitse uppoutua vanhoihin raportteihin pitkäksi aikaa, jotta niistä saisi sisäistettyä tärkeätä tietoa.

2.1 Kojelautojen historia

Ennen vuotta 1965 isoja informaatiojärjestelmiä oli hankala kehittää, koska tietokoneet eivät olleet tarpeeksi yleisiä ja tehokkaita. Näihin aikoihin tuli markkinoille uusia tietokoneita, jotka helpottivat Management Information System (MIS) -järjestelmien kehitystä isoissa yrityksissä. MIS tarjosi johtajille jäsenneltyjä jaksottaisia raportteja, ja useimmiten tieto saatiin kirjanpito- ja rahansiirtojärjestelmistä. [1.]

1960-luvun puolivälissä alettiin kehittää Decision Support Systemiä (DSS) eli päätöksenteon tukijärjestelmää. Kuvassa 1 on esimerkki DSS-järjestelmästä. Ennen vuosikymmenen vaihtumista tehtiin paljon tutkimustyötä ja myös käytännön kokeita aiheesta. Käytännön kokeet olivat merkittäviä ja uraauurtavia DSS:n kehittämisessä. 1970-luvulla järjestelmää kehitettiin, ja se sai laajemman merkityksen koko yrityksen laajuisessa käytössä eikä pelkästään yritysten johtajien päätöksenteossa. Kehitykselle tärkeinä paikkoina toimivat konferenssit, joissa vaihdettiin ajatuksia ja kehitettiin teoriaa DSS:n takana. Powerin mukaan [1] tärkeimmiksi henki-

löiksi konferensseissa nousivat Massachusettsin teknillinen korkeakoulun professorit Peter Keen ja Michael Scott Morton. Ensimmäinen kansainvälinen DSS-konferenssi pidettiin Atlantassa Yhdysvalloissa vuonna 1981. [1; 2.]



Kuva 1. Esimerkki DSS-järjestelmästä. Tämä järjestelmä auttaa vesiresurssien hallintaan [3].

1980-luvulla kehitettiin järjestelmä nimeltä Excecutive Information System, jonka käyttökohde usein oli näyttää tärkeimmät taloudelliset luvut yksinkertaisen käyttöliittymän avulla. Ajatus oli aikaansa edellä ja hieno, mutta järjestelmä ei menestynyt, koska tarvittava data ei ollut helposti saatavilla ja se oli hajanaisena monessa eri paikkaa. 1990-luvulla alkoi liiketoimintatalouden hallinta, jossa keskityttiin yrityksen tietojen hallintaan, luokitteluun ja jäsentelyyn. Samalla vuosikymmenellä keksittiin Data Warehousing, joka keskittyi tietojen keruuseen ja prosessointiin. Ideana oli, että tiedot ladataan erilliseen Data warehouse -tietokantaan, joka on erikoistunut kyselyjen parantamiseen ja niiden tehokkaammaksi tekemiseen. Avuksi tuli myös OLAP eli Online Analytical Processing, tekniikka, joka järjestelee yritysten isoja tietokantoja ja tukee yrityssuunnittelua. OLAP helpottaa tiedon etsimistä ja analysointia. Vuonna 1995 saatiin internetissä toimivat DSS-järjestelmät, ja tätä myötä kehittyivät myös eri yrityksien digitaaliset koje-

laudat. Nykyaikana EIS- ja DSS-järjestelmät ovat yhä vähemmän käytössä, koska digitaaliset kojelaudat tulevat koko ajan suositummiksi. [1; 4; 5.]

2.2 Yrityksen talouden seuranta

Laskutusasteella verrataan työntekijän työaikaa laskutettavaan työhön. Laskutettavan työn ulkopuolelle kuuluvat esimerkiksi yrityksen sisäiset palaverit, virkistyspäivät ja yrityksen omat koulutukset. Mitä suurempi laskutusaste, sen tehokkaampaa yrityksen työteko on. Keskituntihinta on keskiarvo laskutettavista tunneista. Tämä helpottaa selvittämään yrityksen yleistä tuottoa, koska eri projekteista voidaan laskuttaa eri verran. Keskituntihintaa voi esimerkiksi tarkastella eri projekteista tai eri aikaväleillä, kuten kuukausittain, viikoittain tai vuosittain. [6.]

Liikevaihto on yrityksen vuotuinen myyntituotto, josta on vähennetty arvonlisävero ja myynnin oikaisuerät, joita ovat esimerkiksi asiakkaille annetut alennukset ja luottotappiot. Talouselämälehden mukaan [6] liikevoitto on yrityksen tulos ennen veroja ja korkoja ja se on yksi tärkeimpiä mittareita yrityksen kannattavuudelle. Käytännössä liikevoitto on se, kuinka paljon yrityksen myyntituotoista tehdään voittoa. [7; 8.]

Yritystä voi seurata, ja on tärkeää seurata liikevaihdon ja liikevoiton määrällä, mutta ei ole syytä unohtaa yrityksen velkoja ja varallisuutta. Tase kertoo yrityksen määrätyn päivän varallisuudet, velat ja oman pääoman. Näin ollen liikevoitto ja -vaihto ovat vain puolet todellisuudesta. Yritys voi tehdä suurta voittoa ja menestyä, mutta se voi kaatua vanhoihin velkoihin, ellei asioita hoida hyvin. Yrityksen kasvu on myös yksi mittari, joka kertoo mahdollisesta menestyksestä. Kasvun täytyy kuitenkin tapahtua niin, että yrityksen kannattavuus säilyy. [9.]

Yritykseen sijoitetun pääoman tuotto on yksi tärkeistä talouden seurannan kohteista. Siitä on erittelyt, jos haluaa selvittää oman pääoman tuoton, Return On Equity eli ROE, ja jos haluaa selvittää korollisen sijoituksen tuoton, Return Of Investment eli ROI. Pääoman tuotto lasketaan ottamalla esimerkiksi vuoden nettotulot ja jakamalla se pääoman määrällä ja kertomalla tämä tulos sadalla. Se on yksi tärkeimmistä yrityksen kannattavuuden mittareista, ja siinä tarkastellaan yrityksen kykyä pitää huolta varoista, jotka ulkoiset tai sisäiset toimijat ovat sijoittaneet yritykseen. Pääomalla tarkoitetaan siis omaa pääomaa ja korollisia velkoja. Käytännössä luku

kertoo, kuinka paljon tuottoa on kertynyt sijoittajan sijoittamalle pääomalle tilikaudella. Näin voidaan laskea vaikka vuoden ajalta pääoman tuottoprosentti. [10.]

2.3 Yrityksen datan analysointi

Dataa kerätään tänä päivänä käsittämättömiä määriä: Ruotsalaisen [10] tiedon mukaan 92 prosenttia maailman datasta on luotu vain kahden viimeisen vuoden sisällä. Jokainen kuukausi Facebook tallentaa 120 miljoonaa tietoa käyttäjistään, ja joka päivä tallennetaan 30 000 käyttökokemusta Sanoman verkoston toimesta. Nämä ovat vain pieni osa suurien yrityksien datamääristä, joita yritykset koettavat tallentaa ja hyödyntää liiketoiminnassaan. Mitä etuja saadaan datan keräyksellä ja analysoinnilla? Yrityksen dataa keräämällä ja sitä seuraamalla helpotetaan tulevaisuuden lukujen ennustusta, koska nykyisistä luvuista voi nähdä kasvun suuntaa tai yleisen tilanteen paremmin kuin ilman tärkeää dataa yrityksestä. Tämä on johtanut yrityskulttuuriin, jossa johdetaan tiedolla. Pelkkä data ei ole sinällään kovin hyödyllistä, vaan tärkeätä on, miten dataa prosessoidaan yrityksen sisällä ja mihin toimiin ryhdytään datan perusteella. Datan analysointi on tärkeää nykyään kilpailukyvyn ylläpitämiseen erityisesti digitaalisilla aloilla data-analysoinnin yleistyessä. Liiketoimintaa voidaan myös tehostaa, kun nähdään, millä yrityksen sektoreilla menee huonosti ja näiden osioiden ongelmiin voidaan keskittyä ja parantaa yrityksen toimintaa. Näin ollen datan analysointi pienentää riskejä yrityksen toiminnassa. Tärkeää on myös tutustua asiakkaiden tarpeisiin ja seurata heistä kerättyä dataa. Näin voidaan saada tärkeää tietoa, mihin osa-alueisiin kannattaa panostaa. Asiakkaille voidaan tarjota analytiikan edistyessä entistä räätälöidympiä käyttökokemuksia, koska asiakkaista voidaan kerätä tarkempaa ja parempaa dataa kuin ennen. [11.]

Nykyään dataa voidaan hyödyntää paljon aiempaa ketterämmin, älykkäämmin ja nopeammalla aikataululla, koska tekniikka on kehittynyt datan keruun ja sen esittämisen ympärillä. Ennen tehtiin suunnitelmat vuosittain, koska datan ja siitä kehitettyjen raporttien hankkiminen oli hitaampaa ja vaikeampaa. Nykyään, kun data on reaaliaikaista, voivat yritykset reagoida nopeammin virheisiin eikä yrityksien tarvitse tehdä isoja vuosittaisia suunnitelmia. Tämän sijaan ne voivat tehdä esimerkiksi kuukausittaiset suunnitelmat ja muuttaa niitä, jos data enteilee huonoa menestystä. Monet vanhat yritykset, jotka eivät edes ole digialalla, voisivat hyötyä s, jos ne keräisivät dataa ja käyttäisivät sitä parantaakseen liikemalliaan ja liiketoimintaansa. Datan määrään voi myös hukkua, jos ei päätä, mikä data on tärkeää, ja suodata sitä haluamallaan

tavalla. Lehtisen [12, s. 16] mukaan on arvioitu, että datan määrä kasvaa niin nopeasti, että muutaman vuoden päästä jopa 60 %:a datasta ei pystytä tallentamaan. Nykypäivän vastaava luku on 35 %. Haastavaa tulevaisuudessa tulee olemaan suurten datamäärien siirtäminen. Data-analyysissä suosittua on hyödyntää pilvipalveluita, koska näin yritysten ei tarvitse kuluttaa resursseja fyysisiin laitteisiin ja palveluiden lopetus on melko helppoa, jos data-analyysi ei olekaan oikea tapa parantaa yrityksen menetelmiä. Palveluita pystyy myös laajentamaan helpommin pilven kautta. [12.]

Data-analyysissä tärkeää on nopea reagointi, joten datan analyysijärjestelmään kannattaa ohjelmoida hälytysjärjestelmä, joka varoittaa yrityksen avaintyöntekijöitä esimerkiksi sähköpostitse tai tekstiviestillä, jos yrityksen reaaliaikaiset luvut ovat jostain syystä liiketoiminnan kannalta vaarallisissa arvoissa. Taulukossa 1 kerrotaan resepti analyyttiseen johtamisen tärkeimpiin elementteihin. [13.]

Taulukko 1. Resepti analyyttisen johtamisen elementteihin [11, s. 6–7]

Osaamissektori	**Olennaisimmat tekijät**
Organisaatio	• Toiminnan kontrollointi ja strateginen implementaatio • Tuloksiin vaikuttavien tekijöiden visio • Toiminnan uudelleen muotoilu ja sisällyttäminen
Ihmiset	• Analyyttisten henkilöiden hallinta • Kykyjen kohentaminen ja kartuttaminen • Asiaperusteisen yrityskulttuurin normiksi muodostuminen • Johtoportaan sitoutuminen
Teknologia	• Oikean datan ja oikean määrän kerääminen • Laadukkaat analyysisovellukset • Korkealuokkainen kerätty data

Hyvä esimerkki data-analytiikasta on Sanoma-konsernin big datan hyödyntäminen yrityksen ja palveluiden parantamisessa. Big datan määritelmä on luotu kuvaamaan vauhdikkaasti kasvavia ja monisyisiä datamääriä. Näiden datamäärien käsittelyihin eivät riittäneet enää perinteiset datankäsittelytyökalut. Ennen saatiin sirpaleista ja hajanaista dataa sieltä täältä, ja sitä oli han-

kala kerätä yhteen ja hyödyntää. Nykyään Sanoman toiminnasta saatavaa big dataa on helppo hyödyntää palveluiden parantamiseen ja muutoksiin reagointiin nopeammin. Datan keräys ja analysointi eivät kuitenkaan ole ilmaista, ja se vaatii pitkäjänteisyyttä. Suurena haasteena on myös muuttaa yrityksen johdon päätöksenteon prosessi perustumaan kerättyyn dataan eikä intuitioon tai aikaisempaan kokemukseen asioista. [14.]

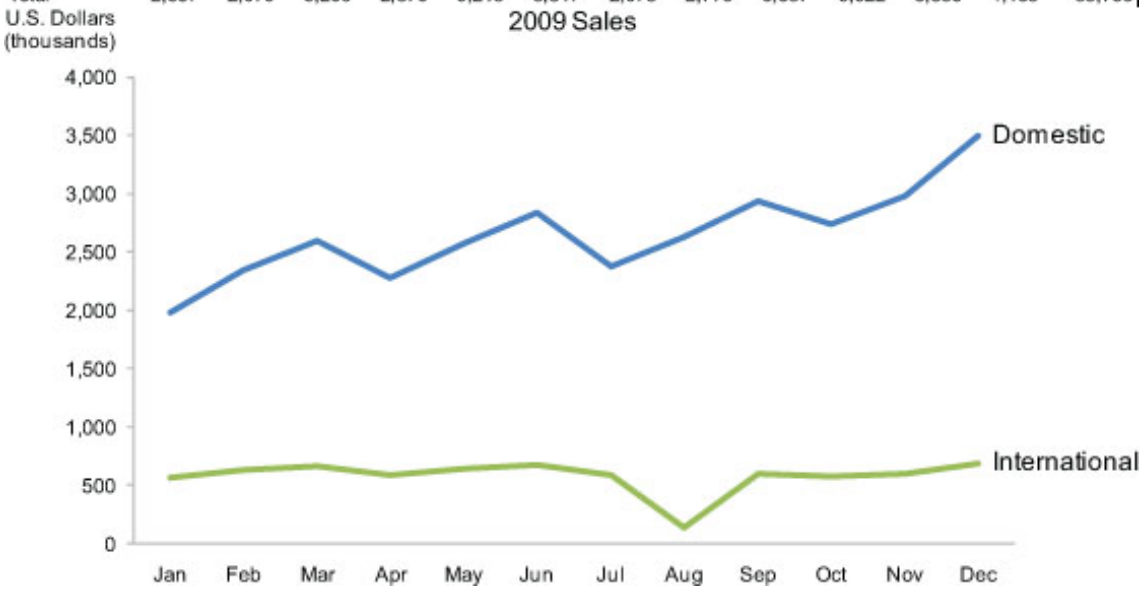
Hyvä tapa aloittaa ja parantaa data-analyysin käyttämistä on tunnistaa yrityksen sisäinen liiketoimintapula, joka saattaisi hyötyä data-analyysistä, ja sitten käyttää data-analyysiä pienemmässä rajatussa projektissa ja tutkia sen tuloksia. Jos tulokset ovat positiivisia, hyötyjä pitää dokumentoida ja analysoida ja jakaa yrityksen sisällä ja laajentaa kokeilua hieman isompaan projektiin. Yleensä yritykset lähtevät mukaan data-analytiikkaan, kun kilpailu sitä vaatii. Yritykset voisivat saada kilpailuedun muihin, jos ne siirtyisivät aikaisemmin data-analytiikkaan kuin kilpailevat yritykset.

2.4 Datan visualisointi

Datan visualisointi on minkä tahansa tiedon muuttamista visuaaliseen muotoon. Visualisoinneissa tietoa yhdistellään symboleihin, viivoihin, koordinaatteihin, pisteisiin, numeroihin, väreihin ja varjostuksiin. Datan visualisoinnista on tullut koko ajan tärkeämpää, koska kuluttamamme sisältö siirtyy digitaaliseen muotoon. Ennen oli myös kaavioita esimerkiksi uutislehdissä, mutta nykypäivänä internetissä olevassa uutisessa voidaan visualisoida esimerkiksi eri puolueiden vaalikannatusta ja kaavio voi olla interaktiivinen. Datan visualisointi nopeuttaa tärkeän tiedon sisäistämistä, tekee uutisten luvun mielekkäämmäksi ja antaa uusia kokemuksia lukijoille. Kuva kertoo enemmän kuin tuhat sanaa, ja tämä pätee myös visualisointeihin. Kuvassa 2 on esimerkki visualisoinnista. Tästä kuvasta voi selkeästi nähdä kotimaisten ja ulkomaisten lukujen eroavaisuuden ja lukujen kehittymisen, eikä itse lukuja tarvitse tarkastella pitkään ja miettiä niiden eroavaisuuksia.

2009 Sales (thousands of U.S. $)

Region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
Domestic	1,983	2,343	2,593	2,283	2,574	2,838	2,382	2,634	2,938	2,739	2,983	3,493	31,783
International	574	636	673	593	644	679	593	139	599	583	602	690	7,005
Total	2,557	2,979	3,266	2,876	3,218	3,517	2,975	2,773	3,537	3,322	3,585	4,183	38,788



Kuva 2. Kaavion yläpuolella olevan taulukon tiedot voi sisäistää paljon helpommin, kun ne on visualisoitu [15].

Tiedon visualisoinnissa hyödynnetään ihmisen tapaa sisäistää tietoa: kaavion eri väreistä, muodoista, koosta ja paikasta katsoja voi sisäistää tietoa erittäin nopeasti. Tässä hyödynnetään niin sanottua esitietoista tarkkaavaisuutta. Ihminen kerää alitajuntaisesti tietoa ja erottelee tärkeän ja turhan tiedon ja käyttää siitä tarvitsemansa. Esimerkiksi, jos ihminen näkee listan lukuja, jotka ovat kaikki samaa väriä, hän joutuu käymään listaa tarkkaavaisesti ja ajatuksen kanssa läpi, ennen kuin ymmärtää listan merkityksen. Jos listalla olevat huonommat luvut olisivat punaisella värillä merkittyjä, ne huomaisi heti ja prosessi nopeutuisi ja siitä tulisi mielekkäämpää. [16, s. 3–7.]

Tiedon visualisoinnissa täytyy kuitenkin välttää tiedon vääristelyä, koska eri asteikkojen suhteet voivat välittää käyttäjälle samasta datasta aivan erilaiset tulokset. Myös liian monimutkainen kaavio voi hukuttaa datan ja tehdä kaavion käyttäjälle luotaantyöntäväksi. Tärkeää on, että kaaviota ei koristella turhilla elementeillä, jotka vievät huomiota pois itse tärkeästä datasta ja visualisoinnista. Hienon näköinen kaavio voi olla ylpeydenaihe sen tekijälle kaikkine hienoine ominaisuuksineen, mutta sen käyttötarkoitus ei tule täytettyä, jos kaavio ei ole tarpeeksi selkeälukuinen loppukäyttäjälle. Melko iso virhe on käyttää yksittäistä kohtaa datan visuali-

soinnissa, koska ei ole mitään vertailukohdetta muuhun dataan, joten visualisoinnista ei voi päätellä, välittääkö visualisointi hyvää vai huonoa lukemaa.

Tiedon visualisoinnit voivat nopeuttaa virheiden korjausta yrityksissä, antaa yrityksille tietoa, jotta kaikki yrityksestä olisivat samoilla toimintalinjoilla, helpottaa tulevaisuuden hahmottamista ja analysointia, näyttää datan, jota ei mahdollisesti tekstimuodossa huomattaisi, antaa yrityksille paremman ymmärryksen asiakkaiden käyttäytymisestä ja tarpeista ja kertoa tärkeät osa-alueet, joihin kannattaa panostaa. Nämä ja muut mahdolliset syyt tekevät datan visualisoinnin tärkeäksi työkaluiksi monille eri tahoille. Nykyiset selaimet ovat kehittyneet niin paljon, että ne pystyvät luomaan reaaliaikaisesti kaksiulotteisia ja jopa kolmiulotteisia visualisointeja ilman, että käyttäjän tarvitsee asentaa mitään lisäosia tai liitännäisiä. Ongelmana ovat erilaiset selaimet, sillä kaikki selaimet eivät tue samoja visualisointitekniikoita ja saattavat jättää visualisoinnin näyttämättä. [17, 18, 19, s. 5–6.]

Jos ydinvoimalaitoksessa jokin pettää ja koko tehdas on lähellä tuhoutua, moni varmasti haluaisi tästä tiedon isoin punaisin välkkyvin valoin. Jos vaaratilanteen tieto näkyisi pienenä tekstinä ydinvoimalan ohjauspaneelissa, vaaratilanteisiin ei useimmiten voitaisi puuttua ajoissa. Tämä on kärjistetty esimerkki siitä, mitä yrityksissäkin voi tapahtua, jos ei olla tarkkana tai käytetä oikeita työkaluja tärkeän datan esittämiseen. Tässä tapauksessa tärkeätä on datan reaaliaikaisuus, jotta kriittisiin asioihin pystyttäisiin reagoimaan ajoissa. Myös vanhaa dataa voidaan haluta esittää, kuten uutisissa esimerkiksi eri paikkakuntien väkilukua, ikäjakaumaa tai syntyvyyttä. Reaaliaikaisuuden tarve riippuu näin ollen visualisoinnin käyttökohteesta. Visualisoinneilla on jopa autettu pelastamaan henkiä. Tästä hyvä esimerkki on Niemisen mukaan [18, s. 2], kun John Snow merkitsi Lontoon kartalle 1854 vuoden kolerakuolemat ja kartan avulla yhdistettiin, että taudin levittäjä oli vesipumppu, jonka sulkemisen jälkeen tartunnat loppuivat ja ihmisiä pelastui. Tällä tapaa saatiin kumottua väärä usko, että tautia levittäisi haju. [18, s. 2.]

3 Kojelaudan suunnittelu

3.1 Lähtötilanne

Fraktio on yli kahdenkymmenen henkilön ohjelmistoyritys, joka tarjoaa asiakkailleen ohjelmistokehitystä, koulutusta, liiketoiminnan kehitystä, ylläpitoa ja käyttökokemussuunnittelua. Frak-

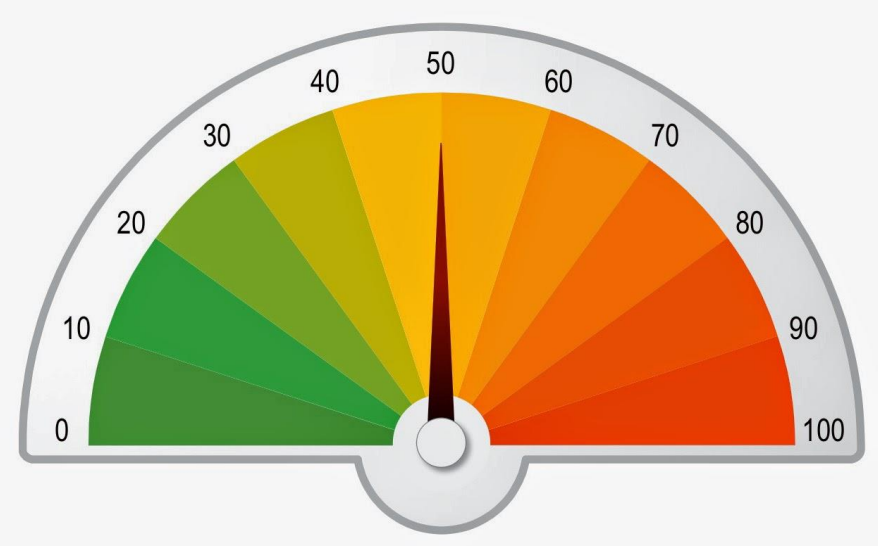
tion toimintatapoihin kuuluu avoimuus ja läpinäkyvyys. Sille tärkeätä on seurata yrityksen lukuja reaaliajassa, jotta tiedetään, kuinka yrityksellä menee. Näihin lukuihin kuuluvat muun muassa laskutusaste, liikevoitto ja liikevaihto. Tällä hetkellä Fraktiolla on käytössä selaimessa toimiva kojelauta nimeltä Geckoboard, joka visualisoi yritykselle tärkeitä tietoja. Se hakee tiedot ohjelmointikieli Scalalla Fraktion tuntikirjausjärjestelmästä lukuja, kuten kuukauden laskutusaste ja keskihinta. Tämä malli toimii aika hyvin, koska tiedon haku ja itse kojelauta ovat erillään, mutta ne yhdistetään, kun kojelaudan pienoisohjelmaan, esimerkiksi viivadiagrammiin, syötetään halutun datan URL-osoite. Ongelmana on, että mahdollisuudet räätälöidä kojelautaa omien mieltymysten mukaisiksi ovat melko pienet. Pienoisohjelmia ei voi ohjelmoida reagoimaan tiedonlähteisiin. Geckoboardin pienoisohjelmien taustaväriä ei saa vaihdettua esimerkiksi punaiseksi tai vihreäksi riippuen yrityksen menestyksestä. Geckoboard ei ole myöskään suunniteltu talouden näkökulmasta, vaan enemmänkin sosiaalisten medioiden aktiivisuuden mittaamiseen ja muiden metriikoiden seuraamiseen. Geckoboard on kehitetty Yhdysvalloissa, joten siitä ei löydy helppoa data integraatiota pohjoismaisiin kirjanpitojärjestelmiin eikä muihin pohjoismaisiin palveluihin. Geckoboard ei anna mitään ilmoitusta, jos jokin seurattavan metriikan arvo laskee nopeasti, ja se voitaisiin korjata, jos kojelauta näyttäisi ruudulla ilmoituksen, antaisi äänimerkin tai väläyttäisi ruutua.

Useat yritykset käyttävät talouden seurantaan Microsoftin Excel-taulukoita, mutta ne tehdään käsin ja se on työläs ja hidas prosessi eikä lopputulos ole kovin informatiivinen. Fraktio haluaa automatisoida tämän prosessin hakemalla tiedot kirjanpitojärjestelmästä ja tuntikirjauspalvelusta automaattisesti. Tämän jälkeen tiedot esitetään visuaalisesti selaimessa toimivassa kojelaudassa. Ongelmana aikaisemmin oli myös, että talouden lukuja seuratessa joutuu koko ajan katsomaan vähintään kuukauden vanhoja kirjanpitoraportteja, jotka eivät kerro nykytilanteesta mitään. Näin mahdollisiin virheisiin reagointi tulee liian myöhään, mikä voi aiheuttaa vaikeita tilanteita.

3.2 Käyttöliittymä

Työn tavoite oli luoda Fraktiolle kojelauta, joka esittäisi selkeällä tavalla yrityksen tietoja Toggl-tuntikirjausjärjestelmästä. Suunnittelu lähti käyntiin miettimällä, mitä tietoja halutaan esittää. Keskihinta, keskihinta projektia kohti, laskutettavat tunnit verrattuna yleiskustannuksiin sekä liikevaihto ja -voitto osoittautuivat tärkeiksi luvuiksi. Keskihintaa ja muita lukuja voi kuvata vain

luvuilla, joiden alla näkyisi prosentteina vertailu edellisen viikon tai kuukauden keskihintaan. Tämä vaatisi lukujen vertailua algoritmein. Eri kategorioiden tuntien määrän vertailuun sopisi hyvin auton polttoainemittarin kaltainen visualisointi. Se on esitetty kuvassa 3. Liikevaihdon muutosta olisi hyvä vertailla viivadiagrammilla tai pylväsdiagrammilla. Kulujen jakautumista voisi vertailla piirakkadiagrammilla. Helppoa olisi näyttää tiedot yhdessä välilehdessä, eikä jakaa tietoja moneen välilehteen kojelaudalla, joten samalla voisi nähdä kaiken tiedon selaimen näkymää alas tai ylös vierittämällä. Tarkoituksena olisi kuitenkin, että ei tule liikaa visualisointeja kojelaudalle, jotta kojelauta säilyisi informatiivisena ja kiinnostavana eikä vaikuttaisi pitkästyttävältä.



Kuva 3. Hyvä visualisointityyli laskutusastetta varten [20].

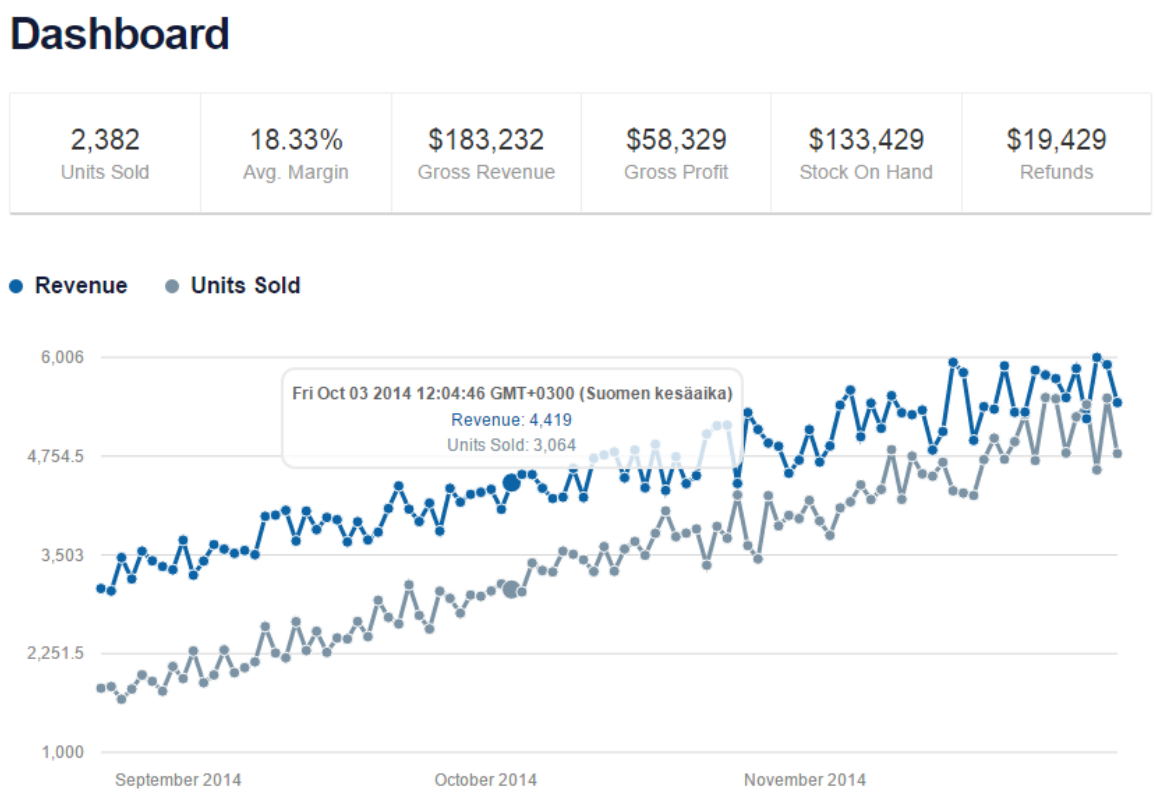
Monet Dashing-kojelaudan omat alkuperäiset pienoisohjelmat eivät ole kovin informatiivisia tai selkeitä, joten päätin asentaa käyttäjien luomia visualisointityökaluja. Hyvänä esimerkkinä ovat Googlen eri visualisoinnit, joita voi integroida Dashing-kojelautaan. Näitä ovat esimerkiksi Googlen viiva-, pylväs- ja piirakkadiagrammit, jotka ovat selkeitä eivätkä tutkimani mukaan liian hankalia lisätä Dashing-kojelaudalle.

Suunnittelun edetessä päätettiin keskittyä vain Toggl-tuntikirjausjärjestelmään, koska kirjanpitojärjestelmä Netvisorin ohjelmointirajapinta on vaikeampi käyttää ja tietoihin on hankalampi päästä käsiksi. Työstä tulisi myös liian laaja, jos alkaisi hakea tietoa molemmista lähteistä, vaikka lopputulos olisi hieno ja kattava kahden eri palvelun tietojen yhdistäminen kojelaudalle.

3.3 Ohjelmistokehysvaihtoehdot

Flakes

Flakes on Kumail Hunaidin [21] suunnittelema ja kehittämä ilmainen ohjelmistokehys, joka on rakennettu JavaScript ja CSS-kirjastoilla ja muilla tyylitiedostoilla. Se tarjoaa monia mahdollisuuksia visualisointeihin ja ensi kädessä taloudellisia sovelluksia varten. Se tarjoaa monia erilaisia komponentteja sovellusten rakennukseen, ja niitä voi vaihdella haluamansa mukaan. Flakes painottaa, että se on hyvin suunniteltu ja tarkasti harkittu ohjelmistokehys. Siltä vaikuttaa, koska saatavia elementtejä ei ole liikaa ja kaikki esimerkit Flakes-ohjelmistokehyksen käytöstä ovat selkeitä ja visuaalisesti miellyttäviä. Flakes-kojelaudan [21] viivadiagrammin ja kojelautanäkymän voi nähdä kuvassa 4.



Kuva 4. Esimerkki kojelautanäkymästä, joka on toteutettu Flakes-ohjelmistokehyksellä. Hiiren pitäminen viivadiagrammin päällä esittää lisätietoa kuplassa [22].

Flakes-ohjelmistokehyksen tarkoituksena on tarjota kattava ohjelmistokehys tekemättä siitä liian raskasta ja hidasta. Se on kuitenkin niin kevyt, että sen tarkoitus ei ole Hunaidin [21] mu-

kaan olla kaiken kattava paketti, vaan se on aloituspiste, jonka saa nopeasti toimimaan ja jota voi itse laajentaa halutessaan. Flakes tarjoaa esimerkiksi viivadiagrammin, hakutyökalun, lomakkeita, painikkeita, taulukoita, valmiita valikoita ja listojen muokkauksen, ja tuotteita voi lisätä omaan tietokantaan selaimesta käsin. Hyvä puoli Flakes-ohjelmistokehyksessä on, että sen CSS-ruudukkojärjestelmä on hyvin samankaltainen Bootstrap-ohjelmistokehyksen kanssa. Bootstrap-ohjelmistokehystä käytetään ympäri maailmaa, ja se oli minulle entuudestaan tuttu.

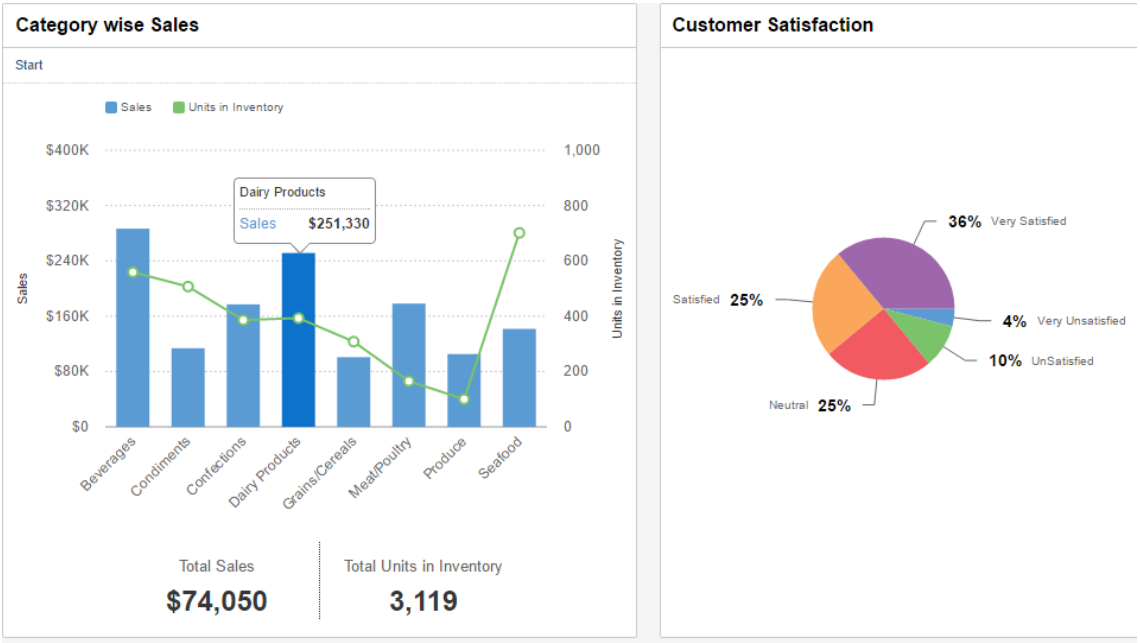
Kojelaudan rakentamisen aloittaminen on helppoa, koska Flakes-ohjelmistokehyksen asennuspaketti sisältää valmiin HTML-sivun tiedoston, jossa on useimmat Flakes-ohjelmistokehyksen tarjoamat elementit. Elementtejä voi alkaa vain muokata haluamansa näköisiksi ja poistaa turhat elementit. Vaikka Flakes-ohjelmistokehys vaikutti hyvältä vaihtoehdolta kojelaudan toteutukseen, se ei ole käyttäjien keskuudessa kovin suosittu eikä yleinen. Dokumentaatiota Flakes-järjestelmän omista ominaisuuksista saisi olla enemmän. Esimerkkisivustot on rakennettu HTML- ja CSS-ohjelmointikieliä hyväksikäyttäen, joten ne eivät ole kovin teknisiä, mutta silti dokumentaatiota voisi olla lisää. Varsinkin lisädokumentaatio oman datan lisäämiseen esimerkiksi viivadiagrammiin olisi erittäin hyödyllinen tieto. [21.]

Razorflow

Razorflow on avoimen lähdekoodin ilmainen kojelautaohjelmointikehys, jota voi kehittää PHP-, HTML- ja JavaScript-ohjelmointikielillä. Se sisältää paljon valmiita pienoisohjelmia tiedon visualisointiin, kuten viivadiagrammi, ympyrädiagrammi, donitsidiagrammi, pylväsdiagrammi, aluediagrammi, taulukko ja muita hyödyllisiä pienoisohjelmia. Näistä suurimman osan voi nähdä kuvassa 5. Tietoa voi sijoittaa joko vertikaalisesti tai eri välilehtiin. Tiedon löytöä helpottaa hakutoiminto, jolla voi suodattaa ei-toivotut tiedot pois diagrammista ja näyttää vain halutut tiedot visualisoinnissa. Tietoa voi myös suodattaa päivämäärien avulla. Erittäin hyödyllistä oli, että pylväsdiagrammin pylväitä klikattaessa tai pitämällä hiirtä diagrammin päällä sai lisätietoa. Näin ollen diagrammit voivat olla monitasoisia. Esimerkkinä voi olla pylväsdiagrammi eri kuntien myyntitiedoista ja kunnan pylvästä painamalla aukeaa kunnan kaupungin myyntitiedot ja kaupungin pylvästä painamalla näkee eri tuotekategorioiden myynnit. Tämä mahdollistaa monimutkaisten tietojen esittämisen ilman, että näytöllä on liikaa asioita. [23.]

Razorflow näyttää hyvältä ja toimii hyvin erikokoisilla laitteilla, jonka voi nähdä hyvin kuvasta 5. Mobiililaitteilla pienoisohjelmat pienentyvät, ja ne voi avata painalluksella, joten pienen

ruudun tila voidaan käyttää tehokkaasti ja käyttäjä voi vaikuttaa siihen, mitä näytöllensä haluaa. Razorflow'n omien sivujen mukaan [23] ensimmäisen kojelaudan voi saada valmiiksi jo tunnissa. Razorflow sisältää eri teemoja, joiden avulla kojelaudan ulkonäköä voi muuttaa helposti. Tämä ohjelmointikehys tukee reaaliaikaista päivitystä tiedoille, eli jos haettava data muuttuu, se päivittyy välittömästi kojelaudalla ilman, että sivua tarvitsee ladata uudelleen.



Kuva 5. Esimerkkikojelauta, joka on toteutettu Razorflow-ohjelmistokehyksellä [24].

Razorflow-kojelautaa voi tarkastella mistä vain, koska se ei tarvitse erillistä sovellusta, vaan HTML5-tekniikkaa tukevan selaimen, joten kojelautaa voi tarkastella esimerkiksi tabletilta, älypuhelimelta tai omalta tietokoneelta. Razorflow-kojelautaa ei voi kehittää ohjelmistokehittäjä, jolla ei ole tietotaitoa ohjelmoinnista, koska kojelauta rakennetaan PHP-, HTML- ja JavaScript-ohjelmointikielillä. Tämä on haittapuoli, jos vertaa esimerkiksi Neatly.io-kojelautaan. [23.]

AppInsights

AppInsights on maksullinen selaimella toimiva kojelauta. Ennen tuotteen ostamista tuotetta saa kuitenkin kokeilla kahden viikon ajan. Tämän kojelaudan avulla voi seurata tärkeitä tietojaan reaaliajassa. Pienoisohjelmiin voi ainoastaan vaikuttaa selaimessa, joten kojelaudan ylläpitoon ei tarvita kovinkaan tietotaitoista henkilöä. AppInsightsin etuna on myös, että käyttöön ei

tarvita asennusta, vaan palveluun täytyy vain kirjautua sisään ja pääsee käsiksi kaikkiin tarvittaviin tietoihin mistä vain selaimen ääreltä.

AppInsightsin etuna muihin kojelautoihin verrattuna on, että kojelaudan datalle voi määritellä rajoja, joiden ylittyessä saa sähköpostiin ilmoituksen. Näin lukuja ei tarvitse seurata koko aikaa, vaan kojelauta pitää huolta, että käyttäjä tietää isoista muutoksista ilmoitusten avulla. AppInsights voi myös lähettää sähköpostiin päivittäisiä tai viikoittaisia tai oman määrittelyn aikavälein tapahtuvia yhteenvetoja tärkeistä luvuista. Palveluun saa luotua eritasoisia käyttäjiä, joilla on eritasoisia oikeuksia muokata kojelautaa ja sen ominaisuuksia, ja kojelaudan visualisointeja ja dataa voi kommentoida. Kojelaudan visualisointeja voi jakaa muille, jotka voivat näin nähdä reaaliajassa olevaa dataa eikä vanhoja lukuja kuvina. AppInsights-kojelaudan [25] elementtejä voi raahata hiirellä ja näin vaihtaa elementtien paikkoja, ja elementtien kokoa voi muuttaa. AppInsights-kojelaudan elementtejä voi tarkastella kuvasta 6.

Kuva 6. Esimerkki AppInsights-kojelaudasta [25].

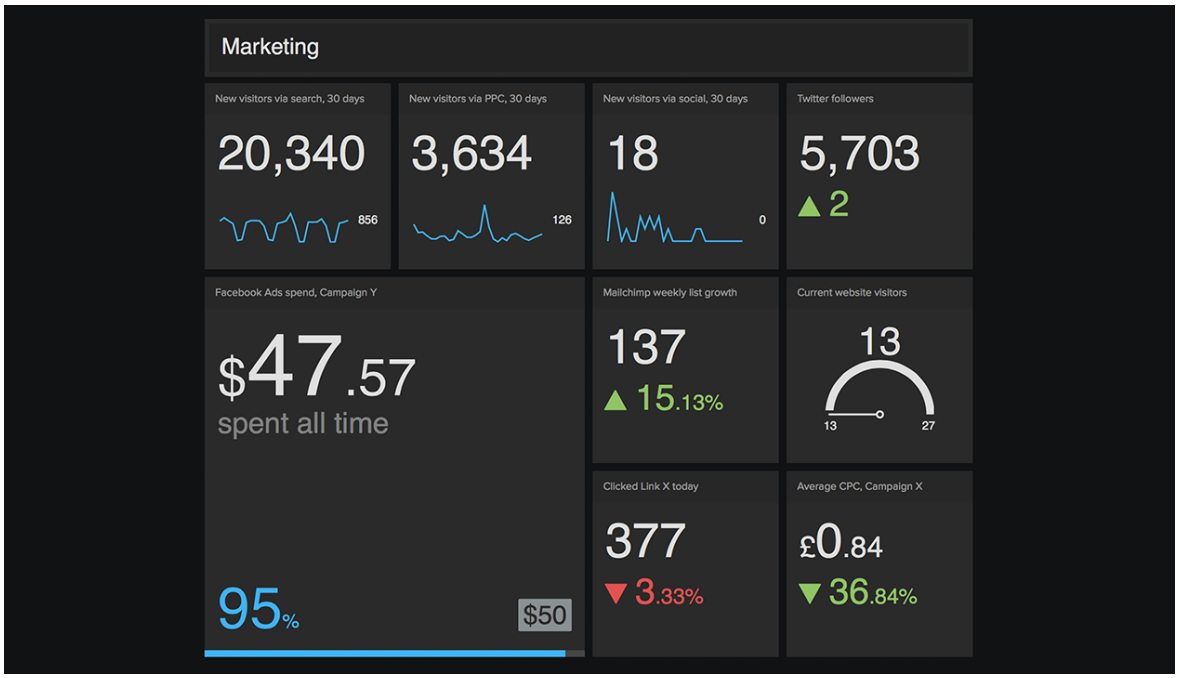
AppInsights tarjoaa helposti yhteydet 70 eri suosittuun internetpalveluun, joista esimerkkeinä Facebook, GitHub, Google Analytics, Google Sheets, Instagram, MySQL, RSS-syötteet, Twitter, YouTube, ja muihinkin palveluihin. AppInsightsin dataintegraation luulisi olevan melko helppoa, ja sivuilla mainostetaan, että prosessi ei ole tietoteknisesti haastavaa. Esimerkiksi, jos haetaan tietoa MySQL-tietokannasta, tietokannan tiedonhaun käskyä voi muokata sovellukses-

ta suoraan ja sen voi yhdistää omiin visualisointeihin. AppInsights mahdollistaa myös oman tiedon lisäämisen kojelaudalle omasta data warehouse -tietokantajärjestelmästä. Palvelu tarjoaa myös ohjelmointikielen pätkiä, joilla voi hakea tietoa ulkopuolisten palveluiden ohjelmointirajapinnoista. AppInsightsia käyttävät seuraavat yritykset: Cisco, GameStop, Greenpeace, Lacoste, Live Nation, Red Bull, Redfox Media ja The Daily Gazette. [25.]

Geckoboard

Geckoboard.io on maksullinen selaimella ja mobiilisovelluksena toimiva kojelauta. Palvelua voi kokeilla 30 päivää ilmaiseksi ennen sen ostamista. Se tarjoaa pienoisohjelmia esimerkiksi viivadiagrammia, pylväsdiagrammia ja karttaa, joilla voi esittää dataa. Se sisältää ohjelmointirajapintavalmiuksia näyttää dataa esimerkiksi seuraavista palveluista: Facebook, GitHub, Google Analytics, Google Sheets, Linkedin, Jenkins, PayPal, RSS-syötteet, Twitter, Tumblr, Vimeo, Trello, Instagram ja muista tärkeistä palveluista. Tämä tekee kehittämisen mahdolliseksi myös niille, joilla ei ole teknistä tietämystä. Lisäksi Geckoboard antaa mahdollisuuden käyttää QR-koodeja. Geckoboardia käyttävät seuraavat suuret yritykset: Netflix, Slack, Airbnb, Marketo, Skyscanner ja Hootsuite. [26.]

Geckoboard on suunniteltu näyttämään hyvältä kaikenkokoisilla laitteilla. Mutta jos tilaa on vähän, Geckoboard voi vaihdella määritellyin aikavälein eri kojelautojen välillä. Kojelaudan värejä voi muokata, ja käyttäjä voi lisätä oman logonsa kojelaudalle, jotta se sopisi yrityksen ulkonäköön. Pienoisohjelmien taustaväriä ei saa muokattua, jos vaikka haluaisi hyvän tuloksen olevan vihreällä pohjalla ja huonon tuloksen olevan punaisella pohjalla. Tämän voi nähdä hyvin kuvassa 7. Pienoisohjelmia voi raahata jopa montakin pienoisohjelmaa samanaikaisesti hiirellä haluttuihin paikkoihin. Kojelaudan voi jakaa muille myös yrityksen ulkopuolisille ihmisille. Pienoisohjelmiin voi myös asettaa tavoitteita, jotka näkyvät kojelaudalla. Geckoboardin mobiilisovelluksella kojelautaa ei voi kuitenkaan muokata tai rakentaa, mikä on yksi huonoista puolista, ja itse muokattuja teemoja ei voi nähdä mobiilisovelluksella vaan näkyvillä on ainoastaan perusnäkymä. Mobiilisovellus on saatavilla ainoastaan Applen laitteisiin eli ei Androidille tai Windows-puhelimille.



Kuva 7. Geckoboardin markkinointikojelauta [27].

Geckoboard on hyvä kojelauta, ja se antaa paljon hyviä valmiita ohjelmointirajapintoja ja valmiita pienoisohjelmia datan esittämiseen. Ongelmaksi muodostuu, että valmiita pienoisohjelmia ei voi muokata paljoa ja omien pienoisohjelmien luonti alusta asti ei ole kovin helppoa. Huonona puolena Geckoboard ei anna minkäänlaisia ilmoituksia käyttäjälle, jos data muuttuu tai muuta tärkeää tapahtuu.

Dashing

Dashing on Shopify-nimisen yrityksen kehittämä ohjelmistokehys, joka on tehty Sinatra-ohjelmointikielellä, joka pohjautuu Ruby-ohjelmointikieleen. Dashing tarjoaa pienoisohjelmia datan esittämiseen, ja se toimii selaimella. Se antaa vapaammat kädet omalle tekemiselle, koska pienoisohjelmia voi luoda itse ja niitä voi muokata enemmän kuin muiden kojelautojen pienoisohjelmia. Dashing-ohjelmistokehyksen käyttäjäyhteisö on ollut aktiivisempi kuin itse kehittäjät ja laajentanut Dashingin mahdollisuuksia roimasti. Dashing-ohjelmistokehyksen dokumentaatio tarjoaa myös linkkejä käyttäjäyhteisön luomiin pienoisohjelmiin, joita on tarjolla runsaasti. Niitä ovat esimerkiksi Googlen kaaviot ja kalenteri, HSL:n pysäkkiaikataulut, RSS-syötteet ja musiikkisoitin. Dashing-kojelaudan sivujen [28] mukaan pienoisohjelmien järjestystä voi vaihtaa raahaamalla niitä ja uuden järjestyksen voi tallentaa. Suurena huonona puolena Dashingissa on, että se ei mukaudu erikokoisille näytöille. Löysin kyllä tietoa, millä kojelautaa

voisi muokata responsiivisemmaksi, mutta mobiililaitteiden yleistyessä luulisi, että responsiivisuus olisi ilman omia muokauksia Dashingin kojelautaohjelmistokehyksessä. Pienoisohjelmat näyttävät myös hyviltä, mutta mielestäni viivadiagrammin esitys on epäselvä. Tämän voi nähdä keltaisella pohjalla kuvassa 8, jossa lukee Convergence. Lisäksi toivoisin, että valmiita pienohjelmia olisi enemmän, mutta onneksi käyttäjäyhteisö on luonut omia pienoisohjelmiaan, jotka antavat lisäarvoa kojelaudalle. [28.]

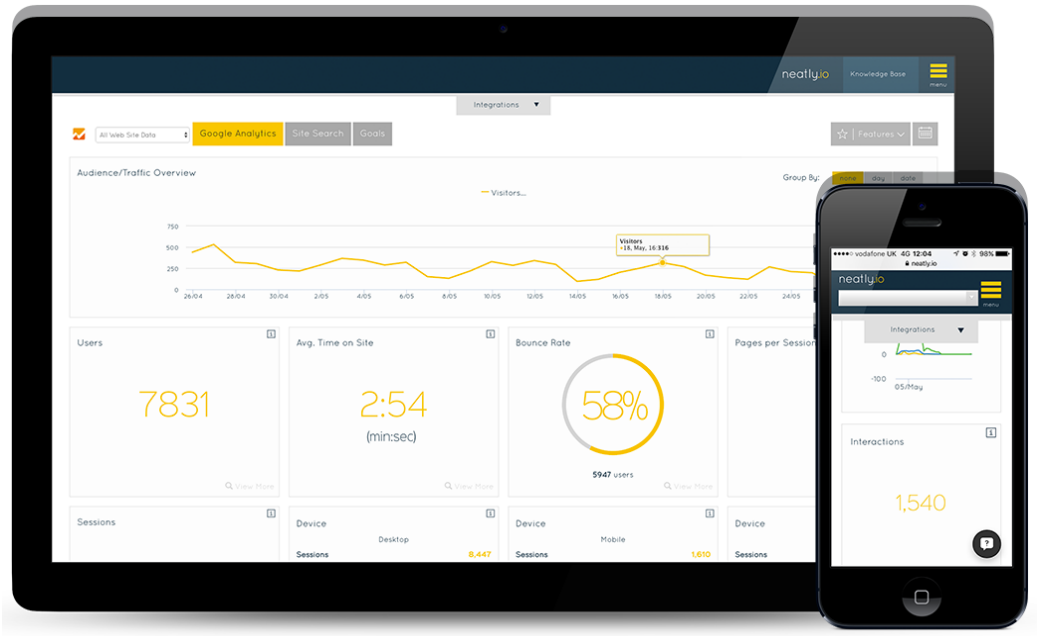
Kuva 8. Dashingin kojelauta [29].

Pienoisohjelmia luodaan ohjelmointikielillä HTML, SCSS ja CoffeeScript. Pienoisohjelmien luonti on suuri etu verrattuna muihin kojelautoihin. Heikkoutena ovat entuudestaan tuntemattomat ohjelmointikielet, mutta projektin edetessä työ helpottuu, kun ohjelmointikieliä alkaa ymmärtää. [30.]

Neatly.io

Neatly.io on iResources-nimisen yrityksen tekemä ilmainen kojelauta, joka tarjoaa yli 45 valmista ohjelmointirajapintaa palveluihin, kuten esimerkiksi Bitly, Google Adwords, Google Analytics, Magento, LinkedIn, Instagram, Facebook, Eventbrite, Paypal, Pinterest, Slack, Youtube ja Twitter. Kojelauta tarjoaa valmiita pienohjelmia datan esittämiseen, ja palvelun käyttö helppoa. [31.]

Etuna on datan integraation helppous: käyttäjän pitää vain syöttää kirjautumistietonsa esimerkiksi Instagram-kuvapalveluun ja kojelauta näyttää heti käyttäjän seuraajien määrän ja käyttäjän viimeisimmät kuvat. Tämän voisi tehdä kuka vain, joten tietoteknistä osaamista ei välttämättä tarvita. Neatly.io-sivujen [31] mukaan kojelaudan voi saada toimimaan jopa alle viidessä minuutissa, mikä kertoo palvelun käytön helppoudesta. Kuvassa 9 voidaan nähdä, miltä Neatly.io-kojelauta näyttää tabletilla ja mobiilipuhelimella. Vanhaa dataa voi tarkastella, ja aikavälin, jonka tiedot haluaa nähdä, voi määritellä. Kojelaudan lukemille voi asettaa tavoitteita, joiden edistymisestä kojelauta kertoo. Neatly.io-palveluun voi luoda useita yrityksiä ja yrityksien kojelaudoille käyttäjiä, joiden avulla eri yritysten seuraaminen helpottuu ja eriarvoiset käyttäjät pääsevät tarkastelemaan kojelautaa. [31.]

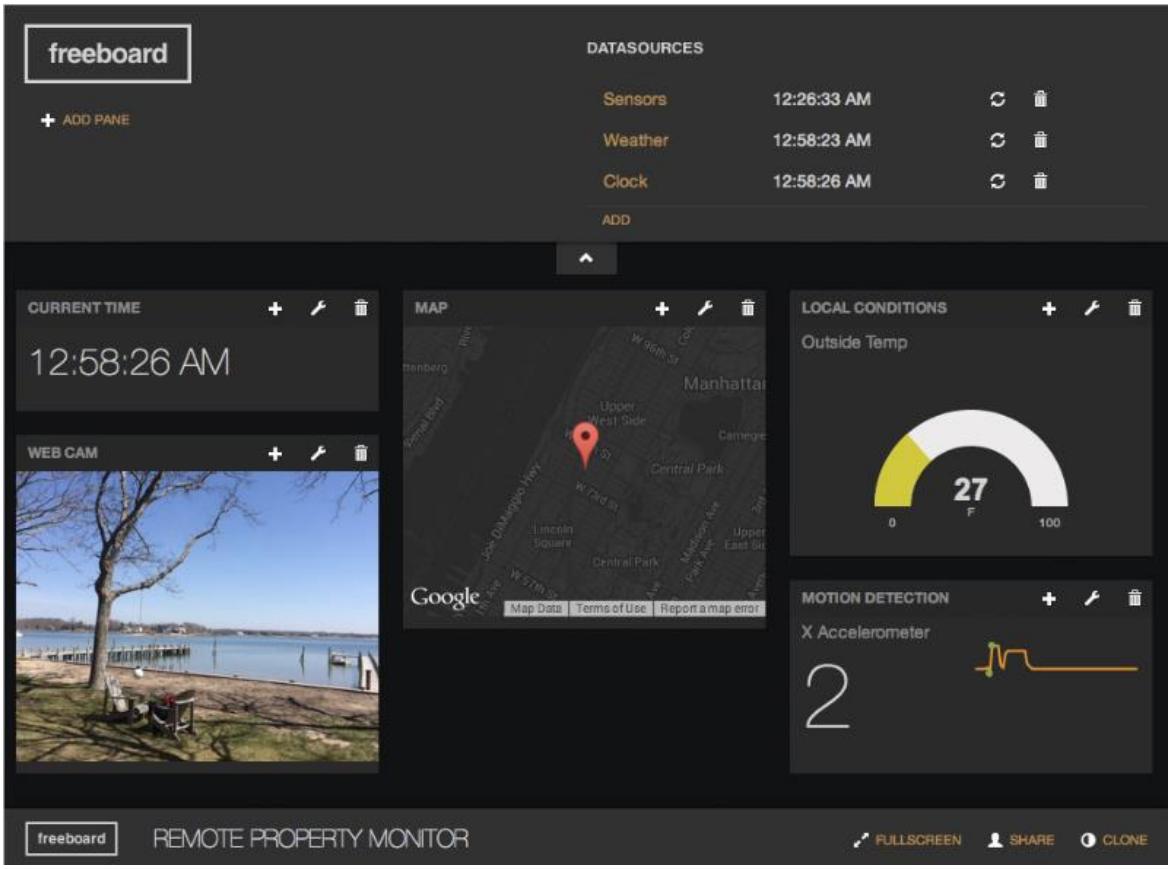


Kuva 9. Neatly.io:n näkymä tabletilla ja puhelimella [32].

Kojelauta tarjoaa myös valmiiksi rakennettuja kojelautoja ja viikoittaisia ja kuukausittaisia PDF-raportteja yrityksen luvuista. Kojelaudan ulkonäköä voi muokata raahaamalla pienoisohjelmia halutulle paikalle. Muuten kojelaudan ulkonäköä ei voi muokata. Lisäksi kojelauta ei näytä hyvältä pienemmillä laitteilla. Näiden puutteiden vuoksi Neatly.io-ohjelmistokehystä ei voinut valita projektin toteutukseen, mikä on sääli, koska tiedon integraatio kojelautaan oli niin helppoa.

Freeboard.io

Freeboard.io on avoimen lähdekoodin ilmainen selaimella toimiva kojelauta. Sen etuna on, että se on helppo ja nopea ottaa käyttöön. Se mahdollistaa myös fyysisten laitteiden lisäyksen tietoihin, jotka tuovat dataa esimerkiksi sääaseman antureilta Freeboard.io-kojelaudalle. Tästä on esimerkki kuvassa 10. Se antaa samankaltaisia pienoisohjelmia datan esittämiseen kuin muutkin kojelaudat ja mahdollisuuden syöttää esimerkiksi XML- tai JSON-dataa sisältävän URL-osoitteen, jonka tiedot Freeboard.io voi esittää. Omia pienoisohjelmia voi luoda JavaScript-ohjelmointikielellä, ja Freeboard-ohjelmistokehyksen GitHub-sivuilla on hyvä dokumentaatio prosessiin. Vaatimuksena on, että JavaScript-tiedosto sijaitsee URL-osoitteessa, joka syötetään Freeboard.io:n järjestelmään. [33.]



Kuva 10. Freeboard.io:n kojelauta [33].

Tässäkin kojelaudassa on mahdollisuus järjestää pienoisohjelmia raahaamalla. Kojelauta näyttää hyvältä kaikenkokoisilla näytöillä. Ongelmana on, että kojelaudan ilmettä ei voi muokata, joten pienoisohjelmat ovat vain tummia eikä värejä voi vaihtaa yritykselle sopivaksi. Tämä on

suuri puute, koska asiakasyritys haluaisi pienoisohjelmien reagoivan eri väreillä erilaisiin tuloksiin. Pienoisohjelmien kokoakaan ei saa muutettua, joten tästä ohjelmistokehyksestä puuttuu paljon tärkeitä ohjelmia. [33.]

4 Kojelaudan toteutus

4.1 Valitut tekniikat

Kojelaudan kehitysalustaksi valittiin Linux, koska Windows-käyttöjärjestelmällä kehityksen aloitus olisi ollut paljon vaikeampaa ja kehitysohjelmistoni ovat paremmat Linuxilla. Neuvon etsiminen internetistä on helpompaa Linuxilla, koska Linuxin ja Applen komentorivikäskyjä käytetään eniten. Applen komentorivikäskyt ovat samanlaisia Linuxin kanssa, joten asia helpottuu Linuxia käyttämällä. Minulla oli Linuxilla jo olemassa oleva kehitysympäristö laadukkaine työkaluineen, jotka on aikaisemmin saatu toimimaan, joten niitä oli hyvä hyödyntää insinöörityössä.

Kojelaudan ohjelmistokehykseksi valittiin Dashing sen räätälöintimahdollisuuksiensa vuoksi. Dashing ei toimi vain selaimessa, kuten useimmat löytämäni kojelautaohjelmistokehykset, vaan omalla palvelimella, josta sen saa selaimeen. Dashing antaa paljon enemmän mahdollisuuksia luoda omia pienoisohjelmia ja laajentaa kojelaudan ominaisuuksia, koska itse ohjelmointikieltä pääsee muokkaamaan eikä pienoisohjelmia tarvitse muokata ainoastaan selaimessa. Valinnan syynä oli myös aktiivinen käyttäjäyhteisö, joka on luonut paljon pienoisohjelmia, joilla voi parantaa tämän ohjelmistokehyksen ominaisuuksia ja säästää aikaa, jotta kaikkea ei tarvitse luoda alusta asti. Huonona puolena Dashingissa ovat sen uudet ohjelmointikielet, joihin en ollut vielä tutustunut, mutta työn edetessä asioiden voi ajatella helpottuvan. Dashing-kojelaudassa ei ole mobiilisovellusta, mikä on melko pieni puute. Tärkeimmiksi vaatimuksiksi kojelaudalle muodostuivat seuraavat asiat:

- Kojelaudan täytyy esittää tärkeät tiedot ja luvut visuaalisesti ja yksinkertaisesti.
- Luvut täytyy esittää reaaliajassa, jotta ei joutuisi seuraamaan vanhaa dataa.
- Kojelaudan pienoisohjelmien värit vaihtuvat tiedon mukana.
- Kojelaudan täytyy varoittaa käyttäjää datan heilahteluista.

4.2 Ruby-ohjelmointikieli

Ruby on Yukihiro “Matz” Matsumoton kehittämä ohjelmointikieli, joka julkaistiin vuonna 1995. Rubyä kehittäessään Matsumoto keskittyi siihen, että ohjelmointi olisi helppoa ja nopeaa. Siinä on vaikutteita monista ohjelmointikielistä, kuten Perl, Smalltalk, Eiffel, Ada ja Lisp. Sen syntaksi tekee C- ja Java-ohjelmontikieliä käyttäneille Rubyn opettelun helpoksi. Ruby on dynaaminen ja oliopohjainen, ja sen kaikki elementit ovat objekteja. Ruby-ohjelmointikieltä ja sen toimintaa voi myös muokata, jos haluaa koodin toimivan eri tavalla. [34; 35, s. 2.]

Kuvan 11 esimerkissä luodaan ohjelma, joka kysyy käyttäjän nimeä ja tervehtii tätä. Itse ohjelmointikieli nähdään kuvassa 11 valkealla pohjalla ja konsolin tulosteet ovat harmaalla pohjalla. Katkoviivalla ympäröity nimi on käyttäjän syöttämää tekstiä. Esimerkin ensimmäinen rivi tulostaa konsoliin rivin “Hello, what’s your name?” Seuraavalla rivillä tallennetaan käyttäjän syöttämä tieto muuttujaan nimeltä name. Rubyn gets-metodilla voi ottaa talteen käyttäjän syöttämää dataa. Chomp-metodilla poistetaan rivinvaihto. Kolmannella rivillä tulostetaan konsoliin ”Hello” ja käyttäjän syöttämä nimi. Seuraavalla rivillä verrataan, onko käyttäjän syöttämä nimi Chris. Jos näin on, seuraava rivi tulostaa konsoliin rivin “What a lovely name!” Jos käyttäjän nimi ei ole Chris, tämä vaihe ohitetaan eikä konsolille tulosteta mitään.

```
puts 'Hello, what\'s your name?'
name = gets.chomp
puts 'Hello, ' + name + '.'
if name == 'Chris'
  puts 'What a lovely name!'
end
```

```
Hello, what's your name?
Chris
Hello, Chris.
What a lovely name!
```

Kuva 11. Ruby-esimerkki [36].

4.3 Sinatra-ohjelmointikieli

Sinatra on internetohjelmistokehys, jonka on kehittänyt Blake Mizerany vuonna 2007. Se on niin sanottu Domain Specific Language eli DSL, mikä tarkoittaa sitä, että se on täsmäkieli tiettyyn käytön alaan. Sinatra on kirjoitettu Ruby-ohjelmointikielellä. Se on nimetty laulaja Frank Sinatran mukaan, ja se on pienempi vaihtoehto muille vastaaville ohjelmistokehyksille. Näitä ovat esimerkiksi Ruby on Rails, Nitro, Camping ja Merb. Se on tarkoitettu sisältämään vain tarvittavat elementit, ja sen on tarkoitus olla yksinkertainen ja nopea luomaan pieniä web-sovelluksia. Tästä on esimerkki kuvassa 12. Helpon ”Hello world!” -esimerkkisovelluksen rakentamiseen ei tarvita kuin pari riviä ohjelmointikieltä. Sen saamiseksi toimintaan menee vähemmän aikaa, kuin esimerkiksi Ruby on Rails -ohjelmointikielessä. Sinatra on hyvä prototyyppien kehityksessä ja ideoiden kokeilussa, kun ei tarvita lopullista versiota vaan kokeellinen versio. Tyypillisesti Sinatra-web-sovellukset kirjoitetaan vain yhteen tiedostoon. Ikävä kyllä Sinatra-ohjelmointikielestä on aika vähän dokumentaatiota, koska se ei ole niin suosittu ohjelmointikieli, kuten esimerkiksi Ruby. Hyvänä puolena on, että Sinatra tukee reititystä, jossa voi määritellä eri URL-osoitteita, joilla Sinatra-sovellus näyttää eri sisältöä. Tätä samaa tekniikkaa tukee myös AngularJS-ohjelmointikieli. Huonona puolena Sinatrassa on, että se ei sovellu kovin suuriin web-sovelluksiin, koska se on tarkoitettu vain pienille ja yksinkertaisille projekteille. Isommissa projekteissa sen suorituskyky ei vain yksinkertaisesti riitä. [37; 38; 39; 40.]

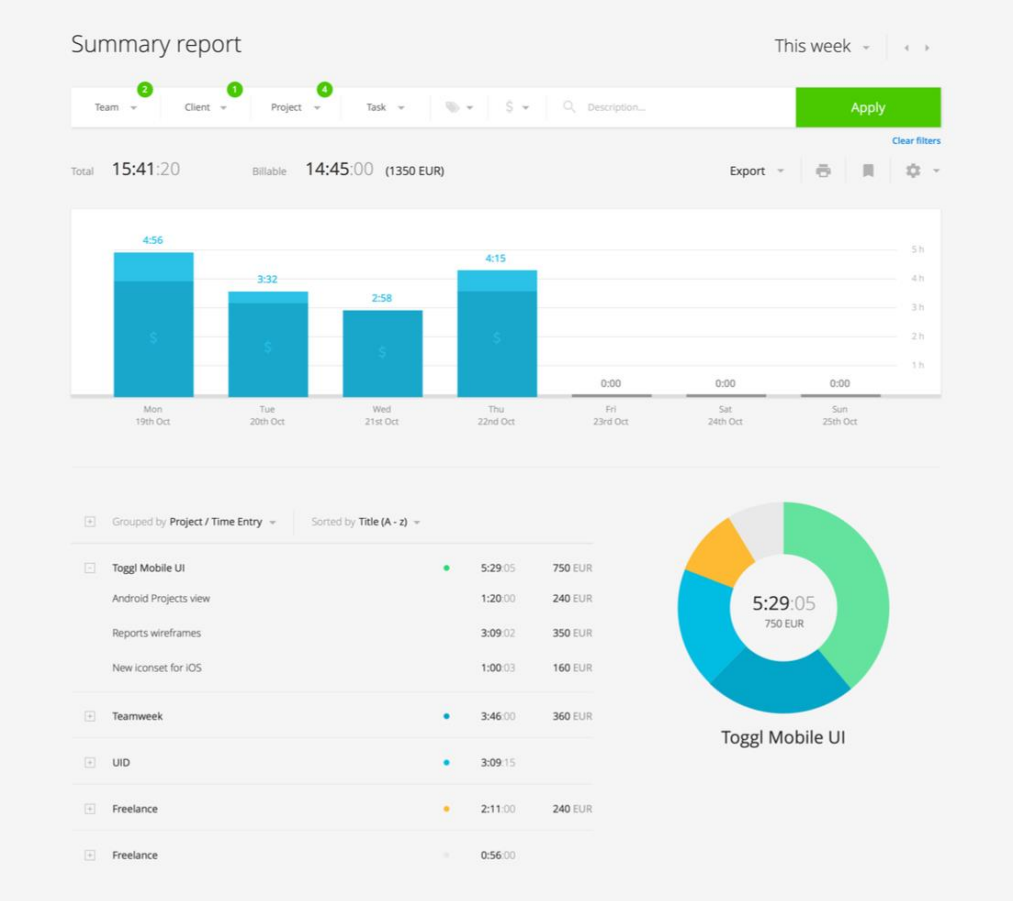
```ruby
# myapp.rb
require 'sinatra'

get '/' do
  'Hello world!'
end
```

Kuva 12. Klassinen “Hello world!” -esimerkki Sinatra-ohjelmointikielellä rakennettuna [41].

4.4 Toggl-tuntikirjausjärjestelmä

Toggl on virolaisen Toggl OÜ -nimisen yrityksen kehittämä tuntikirjauspalvelu. Se on maksuton palvelu, jos yrityksessä on vain viisi ihmistä, muuten palvelu maksaa 10–59 dollaria kuukaudessa käyttäjää kohden. Palvelua voi kokeilla 30 päivää ennen ostamista. Toggl-järjestelmää voi käyttää mobiililaitteilla sovelluksesta Androidilla, Windows-puhelimilla, Applen laitteilla tai selaimelta tietokoneilla. Toggl voi kehittää viikoittaisia tai kuukausittaisia visualisoituja raportteja ajankäyttöluvuista, joita se voi vaikka lähettää sähköpostiin tai niitä voi tulostaa. Kuvan 13 raportissa voidaan nähdä Toggl-palvelun kehittämä esimerkkiraportti. Eri projektien värejä voi muuttaa, projekteille voi luoda alaprojekteja, ja tiimejä voi luoda. Toggl osaa myös viedä raportteja Excel-, CSV- tai PDF-muotoon. Toggl-järjestelmään työtunteja syöttäessä voi joko käyttää desimaalilukuja tai antaa järjestelmän pyöristää luvut. Toggl antaa myös mahdollisuuden tallentaa työajat silloinkin, kun ei ole internetyhteyttä saatavilla. Tuntitiedot siirtyvät järjestelmään, kunhan yhteys palaa takaisin.



Kuva 13. Toggl-palvelun kehittämä raportti kirjatuista tunneista [42].

Aikakirjauksia ei tarvitse muokata yksi kerrallaan, vaan Toggl tarjoaa monen aikakirjauksen muokkaamisen samanaikaisesti. Tärkeä ominaisuus Toggl-palvelussa on, että eri aikakirjauksille voi antaa eri nimikkeitä, joilla niitä voi jälkeenpäin hakea kirjattujen tuntien seasta. Tuntikirjauksia voi ryhmitellä, jotta niitä olisi helpompi käsitellä, ja Toggl ryhmittelee identtisiä tuntikirjauksia automaattisesti. Toggl-palvelun tuntirjauksia voi määritellä joko yleiskustannuksiin tai laskutettaviin, ja tuntikirjauksia voi erotella määrittelemällä niille eri asiakkaat.

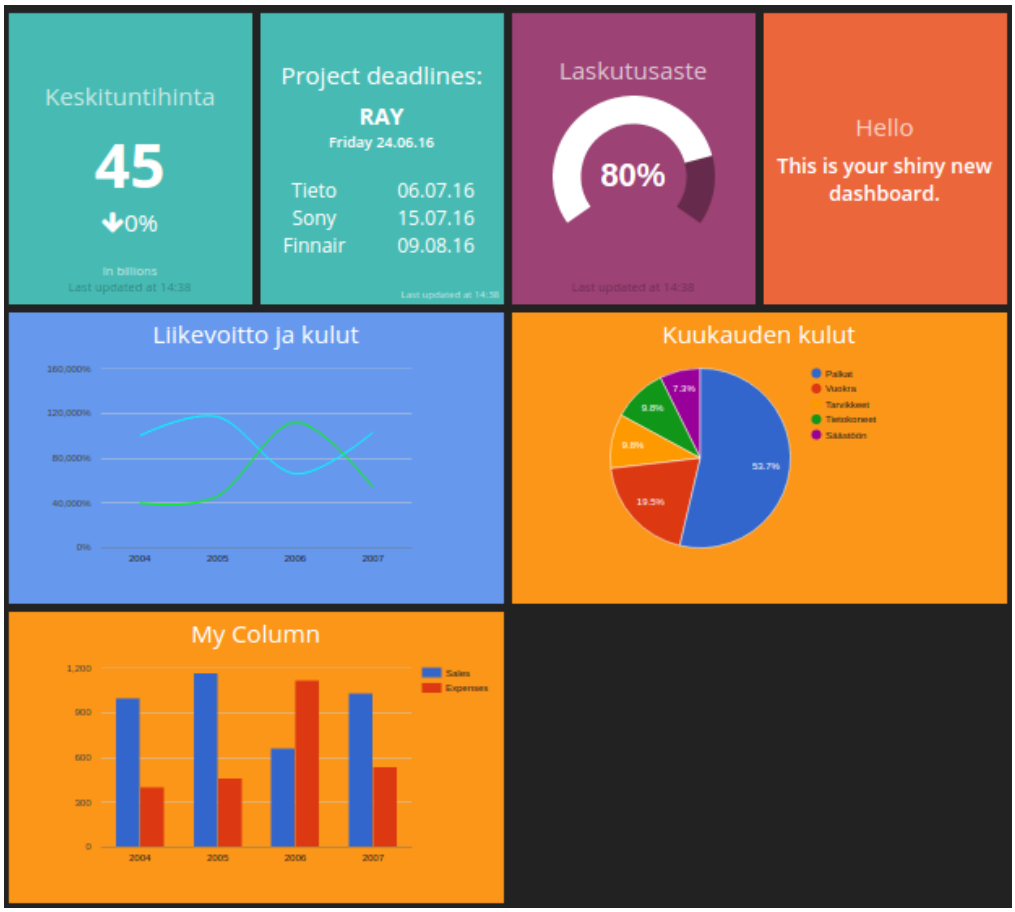
Toggl toimii yhdessä seuraavien palveluiden kanssa: Basecamp, FreshBooks cloud accounting, Teamweek, asana ja GitHub. Teamweek-palvelusta voidaan noutaa käyttäjiä, projekteja ja tehtävänimikkeitä käytettäväksi Toggl-järjestelmässä. Freshbooks-palvelusta voidaan tuoda Toggleen projekteja, tehtävänimikkeitä ja tuntikirjauksia. Toggl-palvelusta voi hakea tietoa ohjelmointirajapinnalla, joka hyväksyy pyyntöjä vain JSON-muodossa. Ohjelmistokehittäjät ovat kuitenkin toteuttaneet ohjelmointirajapintoja Toggl-palveluun seuraavilla ohjelmointikielillä: Java, Python, Ruby, JavaScript, Node.js, C++, .Net, Scala, Php, Go, Elixir ja Perl. Trello-palvelusta on mahdollista aloittaa Toggl-tuntikirjaus. Trello on suosittu palvelu projektien päämäärien ja tietojen järjestelyyn. Tuntikirjauksen voi myös aloittaa Google Calendar -kalenteripalvelusta, kunhan vain asentaa siihen tarkoitetun lisäosan. [43; 44.]

4.5 Kojelaudan toteutus

Kojelaudan toteutus aloitettiin asentamalla Linux-virtuaalikone, jolla tehtäisiin kaikki työ, koska Windows-käyttöjärjestelmä Dashing-kojelaudan kehitysalustana ei ollut hyvä tähän projektiin. Linuxin komentorivikomentoihin tutustuttiin tämän jälkeen melko pikaisesti, mutta kattavasti. Seuraavaksi alkoi pitkä tutustuminen Dashing-kojelautaohjelmistokehykseen, mikä oli melko haastavaa sen Ruby-ohjelmointikielen vuoksi. Dokumentointia Dashing-kojelaudasta on melko vähän. Eri pienoisohjelmien käytöstä olisi kaivattu lisää dokumentointia, jotta ne olisi ollut helpompi saada toimimaan. [28.]

Dashingin asennus sujui melko ongelmitta: ensiksi piti asentaa NPM:llä muun muassa Ruby, Bundler ja Nodejs. Kun ne oli saatu asennettua komentorivin avulla, piti asentaa itse ohjelmistokehys Dashing. Rubyn lisäosat eli gemit asennetaan erikseen, ja kun lisäosia haluaa käyttää pitää asennetut lisäosat punoa yhteen Bundler-lisäosalla. Sitten Dashing-palvelimen voi käynnistää ja pääsee tutkimaan itse kojelautaa. [45.]

Tutkintaa jatkettiin Dashingin valmiisiin pienohjelmiin. Niihin kuuluvat kello, erilaiset listat, kaaviot ja luvut. Lisäksi on mahdollista ladata käyttäjien tekemiä pienoisohjelmia, joita alettiin seuraavaksi tutkia. Niistä hyödyllisimpiä olivat Googlen viiva- ja pylväsdiagrammi, piirakkakaavio ja kalenteri, jotka nähdään kuvassa 14. Ne antavat selkeämmät ja paremmat työkalut datan esittämiseen kuin Dashing-kojelaudan alkuperäiset pienoisohjelmat.



Kuva 14. Insinöörityön aikana valmistunut kojelauta.

Seuraavaksi alettiin tutkia, kuinka saisi tietoa Toggl-tuntikirjausjärjestelmästä Dashing-kojelaudalle. Aloitin kokeiluni komentoriviltä työkalulla nimellä Curl. Se mahdollistaa tiedon hakemisen erilaisista tietokannoista ja eri tekniikoilla. Onnistuin saamaan tuntikirjauksia tulostettua komentoriville ja luotua uusia tuntikirjauksia ja projekteja Toggl-palveluun. Tämä helpotti tiedonhaun ymmärtämistä, koska selvisi, missä muodossa data tulee takaisin ja miten dataa voi suodattaa. Kokeilut olivat onnistuneita ja hyödyllisiä.

Seuraavaksi siirryttiin etsimään lopullista ratkaisua tiedonhakuun, josta voitaisiin siirtää tiedot kojelaudan visualisointeihin. Löysin Toggl-ohjelmointirajapinnan dokumentaatiosta Ruby-ohjelmointikielellä rajapinnan, jolla saisi tietoa hankittua. Yritin saada järjestelmän toimimaan, mutta edes esimerkkikoodi, joka annettiin, ei toiminut. Ei ollut muuta vaihtoehtoa, kuin vaihtaa rajapintaa. Tämä on harmillista, koska rajapinta oli laajakäyttöinen ja tarjosi paljon eri mahdollisuuksia ja metodeja hakea tietoa Toggl-järjestelmästä. Lisäksi ohjelmointirajapinnasta oli melko hyvin dokumentaatiota. Sitten löysin GitHub-koodipalvelusta käyttäjän wkang kehittämän yksinkertaisemman rajapinnan. Tällä järjestelmällä voi hakea nykyisen päivän kaikkien tuntikirjauksien tiedot, tuntikirjaukset tietyllä aikavälillä ja luoda myös uusia tuntikirjauksia. Tämä ohjelmointirajapinta osoittautui toimivaksi ja hyväksi käyttää insinöörityössä. Huonona puolena ohjelmointirajapinnassa oli, että sen metodit hakea tietoa Toggl-tuntikirjausjärjestelmästä olivat paljon suppeampia kuin aikaisemmassa ohjelmointirajapinnassa. Sain tietoa haettua Toggl-tuntikirjausjärjestelmästä, mutta en saanut sitä kojelaudalle sen monimutkaisen järjestelmänsä vuoksi. Jatkoin Ruby-ohjelmointikielen opettelua, jotta koodi ymmärtäisin, miten Dashing-kojelaudan järjestelmä toimii.

Pian selvisi, että projekti olisi liian vaativa ja monimutkainen toteutettavaksi nykyisillä tietotaidoilla ja projekti tulisi epäonnistumaan. Tätä ennen opeteltiin Rubyä melko laajasti, mutta tärkeintä tietoa ei löydetty juuri tiedonsiirrosta, joten insinöörityön teko pysähtyi tähän. Tässä vaiheessa löydettiin parempi vaihtoehto Dashing-kojelaudalle: Razorflow. Sen ohjelmointikielet JavaScript ja PHP olisivat olleet tuttuja entuudestaan, joten työ olisi ollut sujuvampaa ja nopeampaa. Näistä ohjelmointikielistä olisi ollut paremmin dokumentaatiota kuin Ruby-ohjelmointikielestä. Olisin saanut asiakasyritykseltä helpommin apua projektissa, koska sillä on erittäin taitavia JavaScript-osaajia. Väärä ohjelmistokehys Dashing tuli valittua, koska vaihtoehtojen tutkimusvaiheessa ei ajateltu olevan parempaa vaihtoehtoa, koska sitä ei ollut vielä löydetty ja haluttiin päästä toteuttamaan insinöörityötä käytännössä.

Jotta projekti olisi onnistunut, olisi pitänyt valita Razorflow ja tutkia mahdollisia JavaScript- ja PHP-ohjelmointirajapintoja, jotta olisin saanut vastaanotettua tarvittavan datan Toggl-järjestelmästä ja siirrettyä sen Razorflow-kojelaudalle. Tietenkin olisi jouduttu tutkimaan, kuinka Razorflow toimii ja kuinka sillä voi rakentaa kojelautoja. Yhtä tärkeää olisi ollut kanssa tutkia JavaScripti- ja Php-ohjelmointikieliä, koska näitä taitoja ei ollut käytetty vähään aikaan. Uskon, että olisin saanut paljon paremmin projektin käyntiin ja jopa valmiiksi, koska olisin saanut helpommin neuvoja näille ohjelmointikielille. Razorflow tarjoaa myös valmiiksi tarvittavat visuali-

sointityökalut, jotta projekti onnistuisi ja olisi tarpeeksi informatiivinen. En usko, että olisin joutunut itse lisäämään ylimääräisiä pienoisohjelmia ja visualisointityökaluja. Vaikka olisi joutunutkin lisäämään, se olisi ollut paljon mutkattomampaa ja helpompaa kuin Dashingin ja Rubyn kanssa.

Insinöörityön tekemistä olisi jatkettu Razorflow-ohjelmointikehyksellä, jos aika ei olisi loppunut kesken. Onneksi lopputyön tilannut asiakasyritys ymmärsi, että projekteja voi epäonnistua ja varsinkin tässä tilanteessa, kun projektia aletaan tehdä ohjelmointikielellä, joka ei ole entuudestaan tuttu. Insinöörityön lopputuotos on toimiva Dashing-kojelauta, joka osaa hakea päivämääriä ja dataa Googlen kalenteripalvelusta ja esittää mallidataa Googlen viiva-, pylväs- ja ympyrädiagrammeilla. Lisäksi dataa voidaan hakea Toggl-järjestelmästä, mutta sitä ei vielä voida lisätä kojelaudalle, ellei tehdä jatkotutkimuksia ja toteuteta uuden tiedon avulla insinöörityön päämäärää loppuun asti.

5 Yhteenveto

Jos insinöörityö voitaisiin aloittaa uudelleen, ei työhön valittaisi Dashing-ohjelmointikehystä. Oli hankala ymmärtää, miten ohjelmointikehys toimii, kun ei ollut tietämystä ohjelmointikielestä, jolla se on ohjelmoitu, ja ohjelmistokehyksessä oli paljon ohjelmointikieltä, jota tutkia. Ohjelmointikieli oli myös erittäin edistynyttä, joten asioiden ymmärtäminen hankaloitui entisestään. Käytännössä projektin alussa tartuttiin vasta-alkajan taidoilla erittäin vaativan työhön ymmärtämättä työn laajuutta ja tulevaisuuden pahimpia haasteita. Asiakasyrityksessä ehdotettiin käytettäväksi JavaScript-ohjelmointikieltä, jonka avulla voisi alkaa kehittää kojelautaa ja sen tiedonhakujärjestelmää. Tämä olisi ollut helpompaa, koska kieli oli jo entuudestaan tuttu. Ohjaavan opettajan neuvo käyttää Razorflow-ohjelmistokehystä olisi ollut myös paljon parempi vaihtoehto, ja uskon, että insinöörityö olisi valmistunut nopeammin ja tehokkaammin tällä ohjelmointikehyksellä.

Insinöörityössä saatiin kuitenkin aikaan keskeneräinen lopputuotos. Saatiin rakennettua toimiva kojelauta Dashing-ohjelmointikehyksellä, johon lisättiin Googlen visualisointityökaluja. Lopullinen tuote on käyttöliittymältään selkeä ja informatiivinen, eikä mielestäni siihen olisi vaadittu paljoa parantamista. Tietoa saatiin haettua Toggl-tuntikirjausjärjestelmästä parilla eri tapaa, ja sitä saatiin rajattua päivämäärän ja kuvauksen mukaan. Myönteinen yllätys oli, että

projektista saatiin näinkin paljon valmiiksi, koska työkokemusta Linux-käyttöjärjestelmällä ei ollut entuudestaan paljoa, joten käyttöjärjestelmästä opittiin paljon projektin edetessä. Ruby-ohjelmointikielestä ei tiedetty paljoakaan, kun insinöörityön teko alkoi, mutta siitäkin opittiin melko paljon, vaikka taidot ovat vieläkin aloittelijan tasolla tämän ohjelmointikielen kehityksessä. Insinöörityö kertoo myös, kuinka hyvin ohjelmointikehykset, joita käytettiin, oli rakennettu ja kuinka helppo niitä oli käyttää yksinkertaisiin projekteihin. Ohjelmointikehyksistä ja rajapinnoista oli melko tarpeeksi dokumentaatiota, joten elementit saatiin toimimaan useimmiten.

Tulevaisuudessa voitaisiin toteuttaa kojelauta Razorflow-ohjelmistokehyksellä, mutta työssä olisi voitu tutkia lisää Toggl-tuntikirjauspalvelun ohjelmointirajapintoja. Uskon, että JavaScript-ohjelmointikielen vaihtoehtoja Toggl-järjestelmän ohjelmointirajapinnalle on enemmän ja ne ovat laadukkaampia kuin Ruby-ohjelmointikielellä tehdyt. Lisäksi tulevaisuudessa voidaan tutkia asiakasyrityksen kirjanpitojärjestelmää ja sen tiedonhakumahdollisuuksia. Se jätettiin tällä kertaa tutkimatta, koska insinöörityötä aloittaessa tiedettiin, että tiedon hakeminen kirjanpitojärjestelmästä olisi paljon vaikeampaa kuin Toggl-järjestelmästä. Lopputuloksesta voisi tulla melko laaja, ja se yhdistäisi tietoja kahdesta eri palvelusta ja esittäisi ne kojelaudalla. Datan kehityksen ennusteita voisi tehdä, ja datasta voisi tehdä algoritmeja, jotka tekevät ennustuksia tulevaisuuden tuloksista ja luvuista. Algoritmien tutkiminen olisi ollut hyödyllistä visualisointien parantamiseksi. Olisi voitu laskea, kuinka luvut ovat kehittyneet aikaisemmista viikoista, ja olisi voitu ohjelmoida tavoitteita luvuille ja laskea, mikä on lukujen erotus. Algoritmein olisi voitu laskea esimerkiksi keskihintaa, laskutusastetta, liikevoittoa ja muita tärkeitä talouden mittareita.

Insinööriraporttia ja insinöörityötä tehdessä opittiin paljon Ruby-ohjelmoinnista, Linux-käyttöjärjestelmän käytöstä, kojelautojen historiasta, visualisoinneista, datasta ja sen analysoinnista, taloudesta sekä yleisesti paljon projektien teosta ja dokumentoinnista. Tiedonkeruuta tuli väistämättä tehtyä niin paljon, että siitäkin opittiin uusia puolia. Oli mielenkiintoista tutkia Toggl-tuntikirjausjärjestelmän toimintaa ja sitä, kuinka sieltä voi noutaa dataa ja missä muodossa sen saa. Oli myönteistä, että saatiin haettua dataa järjestelmästä, mutta ikävä kyllä sitä ei saatu lisättyä lopulliseen kojelautaan. Uskon, että jos sovellusta olisi kehitetty esimerkiksi JavaScript- tai PHP-ohjelmointikielillä, olisi datan integraatiossa onnistuttu ja saatu insinöörityö valmiiksi asiakasyritystä varten. Olisi pitänyt tehdä enemmän tutkimusta eri kojelautojen mahdollisuuksista ja ominaisuuksista, ennen kuin insinöörityötä alettiin tehdä. Jos

Razorflow-ohjelmistokehys olisi löydetty aikaisemmassa vaiheessa, insinöörityö olisi todennäköisesti valmistunut. Insinöörityön lopputulos on kuitenkin hieno saavutus ja prosessissa oppi paljon, joten se ei mennyt hukkaan.

Lähteet

1 Power, Daniel. 2003. A Brief History of Decision Support Systems. Verkkodokumentti. DSSResources.com. <http://dssresources.com/history/dsshistoryv28.html>. 31.5.2003. Luettu 5.8.2016.

2 Power, Dan. Types of Decision Support Systems (DSS). Verkkodokumentti. The Global Development Research Center. <http://www.gdrc.org/decision/dss-types.html>. Luettu 24.8.2016.

3 Cote, Don. 2006. U.S. Department of Transportation Federal Highway Administration. Verkkodokumentti. <https://www.fhwa.dot.gov/resourcecenter/teams/environment/eq_7_4.jpg>. 10.2006. Luettu 14.1.2017.

4 Shekar, Chandra. 2014. Executive information system. Verkkodokumentti. Slideshare. <http://www.slideshare.net/shivakrishnashekar/executive-information-system>. 24.7.2014. Luettu 5.8.2016.

5 OLAP (Online Analytical Processing) -yleiskatsaus. Verkkodokumentti. Microsoft. <https://support.office.com/fi-fi/article/OLAP-Online-Analytical-Processing-yleiskatsaus-15d2cdde-f70b-4277-b009-ed732b75fdd6>. Luettu 8.12.2016.

6 Fredman, Janne. 2016. Tilitoimistojen tunnusluvut. Verkkodokumentti. ValueFrame. <http://www.valueframe.fi/blogi/tilitoimistojen-tunnusluvut/>. 13.8.2016. Luettu 12.9.2016.

7 Näin Talouselämä laskee tunnusluvut. 2015. Verkkodokumentti. Talouselämä. <http://www.talouselama.fi/uutiset/nain-talouselama-laskee-tunnusluvut-3386449>. 29.1.2015. Luettu 12.9.2016.

8 Yrityksen tunnusluvut ja sijoittaminen. 2014. Verkkodokumentti. Taloussuomi. <http://www.taloussuomi.fi/sijoitus/yrityksen-tunnusluvut-ja-sijoittaminen>. 14.11.2014. Luettu 12.9.2016.

9 Siipola, Sami. 2014. Mikä on tase ja mitä se kertoo?. Verkkodokumentti. Talousvideo. <http://www.talousverkko.fi/mika-on-tase-ja-mita-se-kertoo/>. 11.7.2014. Luettu 12.9.2016.

10 Oman pääoman tuotto-% (ROE). Verkkodokumentti. Balance consulting. <http://www.balanceconsulting.fi/tunnusluvut/oman_paaoman_tuotto>. Luettu 9.12.2016.

11 Ruotsalainen, Roope. 2013. Analytiikka ja tiedolla johtaminen. Diasarja. Dagmar. <http://www.dagmar.fi/blogit/analytiikka-ja-tiedolla-johtaminen>. 19.3.2013. Luettu. 15.1.2017.

12 Lehtinen, Jaakko. 2015. Analytiikan hyödyntäminen pk-yrityksen johtamisessa. Opinnäytetyö. Metropolia.

13 Data-analytiikka. Verkkodokumentti. Pwc. <http://www.pwc.fi/fi/palvelut/riskienhallinta/data-analytiikka.html>. Luettu 14.1.2017.

14 Big Data ja analytiikka vauhdittavat Sanoman verkkopalveluiden kehittämistä. Verkkodokumentti. Solita. <https://www.solita.fi/asiakkaat/big-data-ja-analytiikka/>. Luettu 14.1.2017.

15 Few, Stephen. Data Visualization for Human Perception. Verkkodokumentti. Interactive Design Foundation. <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/data-visualization-for-human-perception>. Luettu 6.12.2016.

16 Few, Stephen. 2004. Tapping the Power of Visual Perception. Verkkodokumentti. Perceptual edge. <http://www.perceptualedge.com/articles/ie/visual_perception.pdf>. 4.9.2004. Luettu 7.12.2016.

17 Data Visualization What it is and why it matters. Verkkodokumentti. SAS. <http://www.sas.com/en_sg/insights/big-data/data-visualization.html>. Luettu 16.9.2016.

18 Nieminen, Joonas. 2014. Avoimen datan visualisointi verkkoselaimessa. Opinnäytetyö. Metropolia.

19 Tuominen, Markus. 2016. Datan visualisointi D3-javascriptkirjastolla. Opinnäytetyö. Tampereen ammattikorkeakoulu.

20 Gauge_0086. 2015. Verkkodokumentti. GetMyGraphics. <http://getmygraphics.com/files/2015/07/Gauge_0086.jpg>. Heinäkuu 2015. Luettu 18.1.2017.

21 Hunaid, Kumail. Build better business software a lot quicker with flakes. Verkkodokumentti. Flakes. <http://getflakes.com/>. Luettu 9.3.2017.

22 Hunaid, Kumail. Dashboard. Verkkodokumentti. Flakes. <http://getflakes.com/preview/dashboard.html>. Luettu 9.3.2017.

23 Open Source HTML5 Dashboard Framework. 2014. Verkkodokumentti. Razorflow. <https://www.razorflow.com/>. 2014. Luettu 9.11.2016.

24 Tour. 2014. Verkkodokumentti. Razorflow. <https://www.razorflow.com/tour/>. 2014. Luettu 9.11.2016.

25 Make Intelligent Business Decisions. Verkkodokumentti. AppInsights. <https://www.appinsights.com/>. Luettu 23.11.2016.

26 Integrations. Verkkodokumentti. Geckoboard <https://www.geckoboard.com/integrations/> Luettu 1.7.2016.

27 Marketing dashboard example. Verkkodokumentti. Geckoboard. <https://www.geckoboard.com/learn/dashboard-examples/marketing-dashboard-example/>. Luettu 1.7.2016.

28 Dashing. 2015. Verkkodokumentti. Dashing <http://dashing.io/>. 2015. Luettu 1.7.2016.

29 Hello This is your shiny new Dashboard. Verkkodokumentti. Dashing. <http://dashingdemo.herokuapp.com/sample>. Luettu 1.7.2016.

30 McGeary, Ryan. 2016. Sass. Verkkodokumentti. Github. <https://github.com/sass/sass>. 12.12.2016. Luettu 5.8.2016.

31 How neatly works. Verkkodokumentti. Neatly.io. <https://neatly.io/features>. Luettu 1.7.2016.

32 Neatly.io. Verkkodokumentti. <https://static.neatly.io/img/guest/home/neatly-responsive-devices.png>. Luettu 1.7.2016.

33 Visualize the Internet of Things. Verkkodokumentti. Freeboard. <https://freeboard.io/>. Luettu 8.7.2016.

34 About Ruby. Verkkodokumentti. Ruby-lang. <https://www.ruby-lang.org/en/about/>. Luettu 26.10.2016.

35 Flanagan, David. Matsumoto, Yukihiro. 2008. The Ruby Programming Language: Everything You Need to Know. O' Reilly Media.

36 Pine, Chris. 2009. Flow Control. Verkkodokumentti. Learn to program. <https://pine.fm/LearnToProgram/chap_06.html>. 4.1.2009. Luettu 1.11.2016.

37 Morin, Michael. 2016. Hello, Sinatra! Using Sinatra in Ruby. Verkkodokumentti. about tech. <http://ruby.about.com/od/sinatra/a/sinatra1.htm>. 26.5.2016. Luettu 2.11.2016.

38 About. Verkkodokumentti. Sinatrarb. <http://www.sinatrarb.com/about.html>. Luettu 2.11.2016.

39 Hagerty, PJ. 2014. Rails vs. Sinatra, Verkkodokumentti. Engine yard. <https://blog.engineyard.com/2014/rails-vs-sinatra>. 20.2.2014. Luettu 22.1.2017.

40 Building tiny Web-applications in Ruby using Sinatra. 2009. Verkkodokumentti. Packt Publishing Limited. <https://www.packtpub.com/books/content/building-tiny-web-applications-ruby-using-sinatra>. 08.2009. Luettu 22.1.2017.

41 Getting started. Verkkodokumentti. Sinatrarb. <http://www.sinatrarb.com/intro.html>. Luettu 22.1.2017.

42 R__webapp. Verkkodokumentti. Toggl. <https://toggl.com/images/landing-pages/r__webapp.png>. Luettu 20.1.2017.

43 Instant productivity boosts. Verkkodokumentti. Toggl. <https://toggl.com/features>. Luettu 20.1.2017.

44 Bonde, Bjarne. 2017. Toggl API documentation. Verkkodokumentti. Github. <https://github.com/toggl/toggl_api_docs>. 18.01.2017. Luettu 20.1.2017.

45 Cai, Larry. 2014. Learn Dashing Widget in 90 minutes. Verkkodokumentti. Slideshare. <http://www.slideshare.net/larrycai/learn-dashing-widget-in-90-minutes>. 5.4.2014. Luettu 1.7.2016.