Samlande och visualiserande av verksamhetsstyrande data för en digital marknadsförings-byrå

Jonas Kostiainen

Examensarbete
Informations- och medieteknik
2016

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	5770
Författare:	Jonas Kostiainen
Arbetets namn:	Samlande och visualiserande av verksamhetsstyrande data för en digital marknadsföringsbyrå.
Handledare (Arcada):	Magnus Westerlund
Uppdragsgivare:	

Sammandrag:

I dagens läge, med stor konkurrens mellan företag i samma bransch är det ytterst viktigt för företag att veta hur de når sina målgrupper. I detta examensarbete diskuteras vikten av verksamhetsstyrande data, hur det samlats och visualiserats tidigare jämfört med idag, och hur företag använder sig av den för att göra beslut och ändringar i sina strategier. Arbetet hänvisar till flera artiklar och intervjuer med experter inom visualisering av data, samt datadriven beslutsfattning inom företag. Målet med arbetet är att ha en fungerande nätverksapplikation, skriven i Node.js, som automatiskt sköter samlande av data från olika molntjänster ett företag använder. Utöver samlandet skall applikationen kunna utföra räkningar på data och skicka den vidare till en tjänst som visualiserar den i olika widgets på en dashboard.

Nyckelord:	Dashboard, metrik, nyckeltal, Node.js
Sidantal:	32
Språk:	Svenska
Datum för godkännande:	20.12.2016

DEGREE THESIS	
Arcada	
Degree Programme:	Informations- och medieteknik
Identification number:	5770
Author:	Jonas Kostiainen
Title:	Gathering and visualizing performance related data for a digital marketing agency.
Supervisor (Arcada):	Magnus Westerlund
Commissioned by:	

Abstract:

With increasing competition between companies within the same area, it is becoming more important than ever for companies to know how to reach their target groups. This thesis focus on the importance of performance related data, how it has been gathered and visualized earlier compared to today, and how companies use it to make data-based decisions for their strategies. Articles and interviews with experts in the fields of data visualization and data driven management are reviewed in the thesis. The goal is to have a working network application that gathers data from several online services in use by the company. In addition to gathering, the application will also perform calculations and format the data in such a way that it is easy to visualize using widgets on dashboards.

Keywords:	Dashboard, metrics, KPI, Node.js
Number of pages:	32
Language:	Swedish
Date of acceptance:	20.12.2016

INNEHÅLL

Without data you're just another person with an opinion.

- W. Edwards Deming

FÖRKORTNINGAR OCH TERMER

- API – Application programming interface
- URL – Uniform Resource Locator, webbadress
- JSON – JavaScript Object Notation
- XML – eXtensible Markup Language
- SOAP – Simple Object Access Protocol
- Dashboard – En infopanel med mätare som visar aktuell och snabbt föränderlig information
- Widgets – Mätare eller komponenter en dashboard byggs upp av

1 INTRODUKTION

I arbetet diskuteras hur viktigt data är för att företag i dagens läge skall kunna tävla med andra inom samma bransch. Ökande mängder småföretag och betydelsen av data i beslutsfattandet, tvingar företag att börja tänka om sin strategi. Läsaren ges en inblick i hur data har samlats och visualiserats de senaste årtionden och vad som har lett till hur det sköts idag. En lösning för hur data snabbt kan samlas in och hållas uppdaterad med hjälp av en nätverksapplikation tas upp, tillsammans med olika slags problem som kan uppstå på vägen. Arbetet beskriver både verktygen och metoderna använda för att uppnå detta.

1.1 Bakgrund

Ännu för 20 år sedan var det ganska ovanligt att man hade någon annan apparat kopplad till internet än en bordsdator. Idag använder människor dagligen internet med sina mobiltelefoner och andra mobila enheter för att läsa nyheter, söka information och kommunicera med varandra i sociala medier. Man behöver inte tänka efter mycket för att se mängden av data över användares vanor på internet som finns att samla ihop. Det är roligt att få veta hur många som dagligen besöker ens hemsida, men en siffra räcker inte för att effektivt få fram sitt meddelande. Gratis verktyg av kända tjänster som Google och Facebook gör det enkelt för vem som helst att följa med hur användare med olika kön, ålder eller ursprung beter sig på en tjänst man erbjuder.

En digital marknadsföringsbyrå kan ha flera kunder med synlighet i flera olika medier, därför är det ytterst viktigt att förstå hur man skapar en produkt som riktar sig exakt till den målgrupp man försöker nå. För att hålla reda på all data som samlas används tjänster för att monitorera, lagra och analysera data. Från vissa tjänster får man färdiga rapporter till sin e-post med ungefär en veckas mellanrum, men för en stor del gäller att logga in på en webbsida och själv söka fram just den siffran man är intresserad av. Att generera tabeller ur faktureringsprogram eller –tjänster är både tidskrävande och invecklat för att få fram en siffra man söker.

1.2 Syfte, mål och avgränsning

Syftet med arbetet är att väcka ett intresse för hur data av olika sorter tillsammans kan stöda ett företags verksamhet. För att ta det ett steg längre diskuteras också hur ett företag som helt baserarar sina beslut på data fungerar, och vilka hinder man kan stöta på när man rekonstruerar strategin för ett företag. I arbetet söks också svar på frågor om hur visualisering kan hjälpa en att få ordning på och tyda data. Väntat resultat för arbetet är inte att se förändringar i ett företag, utan främst en fungerande prototyp på ett system som kan hämta data från olika tjänster. För att åstadkomma detta kommer en nätverksapplikation att skapas. Applikationen skall utöver hämtandet av data också klara av att utföra uträkningar och presentera slutresultaten. Data skall visas så att datatypen passar ihop med visualiseringen, men viktigast är att den är enkel att förstå och uppdateras ofta. Tanken är att demonstrationen skall bevisa hur mycket oanvänd data företaget har och att man antagligen i långa loppet skulle tjäna på att utnyttja den.

Läsaren förväntas ha åtminstone grundläggande kunskaper i JavaScript och webbteknologier. I kapitlet om verktyg förklaras grunder för komponenterna som använts, men till exempel JavaScript i sig själv diskuteras inte. På begäran av beställaren av arbetet kommer företaget inte att nämnas med namn. Under andra kapitlet diskuteras företag i allmänhet, i tredje kapitlet syftar ”företaget” främst på beställaren.

1.3 Metoder

Arbetet har både en praktisk och en mer teoretisk del. För programmeringsdelen används främst en empirisk forskningsmetod. När man har möjlighet att använda sig av färdiga moduler i programmerandet, är koden man skriver själv den delen som kräver mest testande. Tjänsterna testas en åt gången, och när den data man sökt fås fram läggs tjänsten till i det slutliga systemet. Till programmeringsdelen hör också läsning av dokumentation för samtliga tjänster och verktyg använda under arbetet.

För delen om datadrivna företag används en mer kvalitativ metod. Flera artiklar, intervjuer, guider och möjligtvis böcker studeras för att försöka hitta teorier och åsikter från olika synpunkter. Frågor som mer direkt handlade om beställarens tjänster, och vilken

data som önskades få fram, diskuterades med personer i företaget som data var mest relevant för.

De olika delarna kräver skilda forskningsmetoder, och kommer att leda till oliks typs resultat. I slutet av arbetet presenteras bägge resultat och en sammanfattning för hur de hörs ihop lyfts fram.

2 DATADRIVEN STRATEGI

I detta kapitel kommer jag att ta upp hur ett företag, där besluten görs på basen av data fungerar, hur den data bestäms och vem den sist och slutligen påverkar. Kapitlet ger en bild över hur ett datadrivet företag fungerar, och hur ett sådant skiljer sig från företag där antaganden och magkänsla är stora delar av beslutsfattandet.

2.1 Visualiserande med hjälp av dashboards

2.1.1 Bakgrund

Om man bortser från hur data visualiserades innan datorer med grafiska användargränssnitt fanns, utvecklades de första systemen som i dagens läge kallas för dashboards redan under 1980-talet. Under det årtiondet kallades de för Executive information systems (EIS) och användes endast av de högsta cheferna i företag. Stephen Few beskriver i sin bok om dashboard design om hur EIS var banbrytande, men före sin tid. Varken för insamlande eller behandling av data fanns den teknik vi i dagens läge använder. Detta ledde till att data ofta var bristfällig, föråldrad eller till och med felaktig. EIS systemen blev aldrig en stor succé. Under 90-talet sattes mycket vikt på att samla, spara och integrera data. Tekniken för att åstadkomma detta utvecklades fort, men få tänkte på metoder för hur all data lätt skulle kunna användas eller dras nytta av. (Few, 2006. s. 14)

I början av 2000-talet blev det klart för aktieägare att det är viktigt att veta hur det verkligen går för företag och det satte press på företagsledare för att försäkra sig om att ingen blev lurad. Företagsledare sökte nu efter metoder för att kunna monitorera hur alla nivåer av företaget presterade, och med nu tillgänglig teknologi utvecklade företag inom affärsanalys dashboards åt sina kunder. (Few, 2006. s. 15)

Hittils har jag i arbetet endast diskuterat dashboards, men i början av 90-talet, när idén om att identifiera och använda nyckeltal som del av strategier gjordes känd av Robert Kaplan och David Norton, utvecklades något som kallas balanserade styrkort (eng. balanced scorecard). Lyndsay Wise (Wise, 2010) beskriver styrkort som ett sätt att följa med hur det går för ett företag över en längre tidsperiod, hur metrik ser ut i jämförelse

med företagets målsättningar och hjälpa göra beslut på lång sikt. Även om också dashboards kan användas för att visa förändringar i data över en längre tid, är tanken oftast att visa upp snabbt föränderliga data i realtid och hålla koll på att daglig verksamhet löper som den skall. Dashboards är dessutom ofta mer interaktiva än styrkort, man skall enkelt kunna ändra på hur data visualiseras och kunna använda en större mängd data för att få fram olika siffror. Korten och dashboards har liknande funktion och det är inte ovanligt att företag använder sig av en kombination av dem båda. Aleksey Savkin är av åsikten att styrkort är menat för nyckeltal, där som dashboards mer följer metrik. Han är övertygad om att båda är viktiga för företag, och kanske kommer att leda till en slags korsning av dem. (Savkin 2015).

2.1.2 Fördelar med dashboards

Hur kan dashboards stöda ett företags verksamhet? Kan de förutom att hjälpa ledningen med beslutsfattande, också påverka motivationen hos de anställda? Professor Hugh Watson skriver om ett gammalt ordspråk, ”what gets watched gets done”, vilket syftar på att människor ändrar sitt beteende när någon följer med eller monitorerar deras aktivitet för att resultaten ska se så bra ut som möjligt. Han berättar om ett företag som inte aktivt samlade data innan 2005, men vars omsättning steg med $50 000 per månad efter att dashboards för personalen och ledningen togs i bruk. Största delen av personalen jobbade som telefonförsäljare, den enda data som egentligen följdes var om man hade lyckats med en affär eller inte och ledare hade svårt att veta vilka försäljare som borde belönas. Den bästa försäljaren ansågs vara den med flest lyckade affärer, men många tyckte det var orättvist. Med introduktionen av dashboards fanns det plötsligt mycket mer data att följa. Ledningen för callcentret ville försöka göra säljandet till en slags tävling mellan försäljarna, och började visa upp data såsom samtal per timme, storleken på beställningar och kundnöjdhet. Alla försäljare hade en personlig vy där deras prestation visades i jämförelse med de andras. Från nya data räknades det nu ut ett index för varje anställd, och lönen bestämdes efter indexet. Det nya systemet hade den positiva inverkan man hade hoppats på; personalen tyckte om att kunna följa med sina egna prestationer och att tävla med sina medarbetare. (Watson, 2009)

Det är sällan en helt färdig-konfigurerad dashboard möter alla de krav ett företag har för sin data, och med lite extra arbete samt en klar plan kan man själv skapa en dashboard mer direkt riktad åt företaget. Enligt Alexander Chiang, direktör för konsulttjänster på Dundas Data Visualization, är största fördelen med att själv skapa sin dashboard att man får en panel som visar just de siffror som är viktiga för företaget och de anställda. (Gaffney, 2009). För att ge en klar bild över bestämda data är dashboards ofta simpla till sitt utseende och lätta att förstå. De ger en simpel introduktion till analyser och statistik, kan på ett positivt sätt påverka de anställdas motivation och är därför en bra startpunkt till att skapa en mer data- och faktabaserad miljö för beslutsfattning inom företaget.

2.1.3 Eventuella nackdelar och fallgropar

Kan dashboards ha en negativ effekt? Vad händer om man visar data som är felaktig eller irrelevant? Finns det dåliga sätt att visa data på?

Few skriver att dashboards är ett bra exempel på vad som händer när något inom teknik väldigt snabbt blir populärt bland många; alla är ivriga och vill snabbt med i händelserna, fastän få har på det klara vad det egentligen handlar om eller hur det fungerar. Flera försäljare lade i början av 2000-talet till visualiseringsmöjligheter i sina existerande program, utan att ägna ordentligt med tid för att se till att det gjordes rätt. En ordentlig dashboard skall effektivt monitorera aktuella händelser, och även om hur man visualiserar data för att tolka den är väldigt viktigt, är det inte meningen att det skall göras för att imponera på någon. (Few, 2007. s. 8-9).

När det kommer till riktighet av data lyfter professor Brian Ballou igen fram vikten av automation. För varje person som manuellt måste skriva in någonting ökar chansen att det på vägen sker något fel. Oberoende om felet händer av misstag, eller om någon med avsikt gör ändringar i datan kommer felaktig data att visas upp, och beslutsfattaren gör i värsta fall sämre beslut än han skulle ha gjort utan någon dashboard alls. Det är inte bara felaktig data, utan också data i för stora mängder, som kan leda till problem. För mycket data, för många widgets eller för många siffror leder lätt till kaos och dashboarden mister sin poäng. Ballou ber en också beakta att även om dashboards uppmuntrar till att snabbt

kunna reagera på problem är det viktigt att analysera problemet innan man gör förhastade beslut. Man måste också kunna se en helhet, i och med att dåligt planerade handlingar baserade på endast en liten mängd data lätt leder till mera problem. (Ballou, 2010)

2.2 Metrik och nyckeltal för företaget

Det finns flera siffror man måste ta i beaktande när man beskriver värderingen för ett företag. Till näst förklaras vilka dessa siffror är och hur de bestäms.

Metrik är ofta siffror som är direkt knutna till företagets verksamhet. Med rätt definierad metrik kan man skapa ett ramverk med data för vidare analyser. Till exempel säger en siffra om ett företags vinst under ett år egentligen väldigt lite, men tillsammans med andra siffror om företagets omsättning och satsat kapital är det lättare att analysera och dra slutsatser om varför året var lyckat. Av det insamlade data kan man skapa så kallade nyckeltal, eller KPI (key performance indicators), som till exempel mäter effektivitet och resultat i ett företag. Man kan tänka sig metrik som flera olika faktorer som påverkar en kunds vilja att besöka ett företags webbsida, så som användarvänlighet, mängden text, bilder eller tid det tar att ladda sidan. Alla dessa siffror bidrar till ett nyckeltal; hur många besökare har webbsidan totalt över en viss tid. Ledningen för ett företag har i bästa fall klart för sig vilka nyckeltalen och de största målen för ett företag är, men ofta händer det att också tidigare förbisedd metrik leder till nya idéer och beslut. (Halper, 2011).

2.3 Migrering till datadriven beslutsfattning

Pat Saporito, chefsdirektör för SAP Global Center of Excellence for Analytics, är av den åsikten att företag som redan använder sig av data, och lyckas använda metrik för beslutsfattande i framtiden, kommer att ha ett stort försprång över företag som inte har den möjligheten. (Saporito, 2008). Hennes påstående stöds av resultaten i en undersökning från 2011. I undersökningen skickades frågeformulär gällande data-användning till IT-företag. Bland det viktigaste man var ute efter, var svar på om företaget hade data tillgängligt för

att stöda dess beslutsfattning eller produktutveckling. Efter uträkningar och analyser visade det sig att datadriven strategi har en direkt, och positiv, inverkan på företagets verksamhet (Brynjolfsson, 2011).

För att ett företag skall kunna ändra sina beslut till att vara datadrivna, krävs det att personer högre upp i företaget är färdiga att göra ändringar, och förstår hurdana ändringar som krävs av företaget. Detta kan vara lättare för mindre och agila företag, då förändringarna i hur företaget fungerar inte nödvändigtvis behöver vara lika stora eller inte påverkar lika många anställda. Förändringen går inte att göra i en handvändning, utan kräver också testande av nya metoder. När man har en klar bild över vad man strävar till att uppnå, vet man också vad är viktigt att testa och utveckla vidare. I varje delmoment av processen (Figur 1) behövs det en eller flera personer som förstår sig på de olika områdena. En hypotes läggs fram och arbetet för verkställande av planen sätts igång. Beroende på arten av ändring samlas data, för att efter en bestämd tidsperiod bli analyserad. Hypotesen kan komma från ledningen, planen och verkställande sköts av projektledare och anställda inom den involverade avdelningen. När data har samlats tas arbetet över av någon som analyserar och jämför data med tidigare resultat. Analysen borde ge ett klart resultat för hypotesen och leda till nästa hypotes, eller eventuellt också ändringar i företagets strategi.

Figur 1: Arbetsflödet i ett datadrivet företag (Sten, 2016)

För alla anställda är de största målen i företagets strategi inte alltid klart framställda, men för att kunna utnyttja alla sina resurser så bra som möjligt borde man sträva efter det. Varje anställd har något de är bra på och någon form av data de arbetar med varje dag. När de anställda vet vad de gemensamma målen är har de enklare att bidra till beslutsfattandet. När frågor som ”vad baserar du dina åsikter på?” kommer fram, styrs en till att i framtiden alltid ha data bakom sina påståenden. Således skapas också en naturlig förbindelse, där information flödar åt båda hållen, mellan ledning och avdelningar. (Rouke, 2015).

För att kunna göra beslut på basen av data måste företaget se till att data hela tiden samlas, hålls uppdaterad och framför allt att den är korrekt. Manuell generering av rapporter för en tjänst kan fungera, men när det handlar om flera tjänster där data samlas från blir mängden arbete snabbt för stor. På basen av data görs uträkningar, dessutom integreras data med andra tjänster eller delas vidare. Det enda förnuftiga sättet att sköta rapporteringen på är automatiskt.

3 VERKTYG UNDER UTVECKLINGEN AV NÄTVERKSAPPLIKATIONEN

I detta kapitel kommer behandlas vilka verktyg som användes under arbetet och bakomliggande orsaker till varför de valdes.

3.1 Mjukvara och operativsystem

All mjukvara som behövdes i utvecklingen av arbetet fanns färdigt installerat på arbetsdatorn då projektet inleddes. Som utvecklingsmiljö användes PhpStorm, som trots namnet också är menat för webbutveckling. För att hålla koll på versionshanteringen användes Git klienten SourceTree. Koden utvecklades och testades främst i en lokal miljö på OSX 10.11.5, men körs när arbetet är klart i Raspbian, ett linuxbaserat operativsystem för Raspberry Pi -datorer.

3.2 Programmeringsspråk och bibliotek

3.2.1 Geckoboard

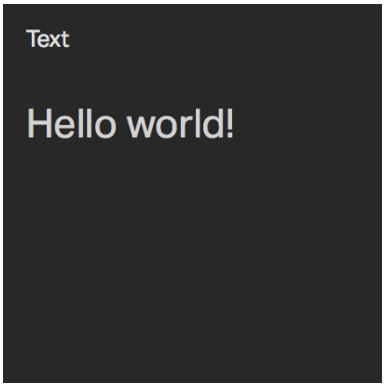
Geckoboard erbjuder dashboards som en tjänst åt sina användare, och utnyttjades under detta arbete. Geckoboard beskrivs på den officiella webbsidan som ett program för att försnabba tillväxten i företag och hjälpa dem hitta de nyckeltal som verkligen har den största inverkan på företagen (Geckobard, 2016). De har satt stor vikt på att dashboards skall vara synliga och tillgängliga för alla i ett företag, och har både förslag på maskiner att köra dem på och olika skärmar att visa upp dem på. Bland deras dokumentation hittar man videoklipp och guider med allt från hur man väljer sina nyckeltal till hur man använder dem för att öka tillväxten och ta steget till en mer datadriven arbetsmodell.

Ett alternativ till Geckoboard hade varit freeboard, en dashboard lösning med öppen källkod. Med freeboard hade man haft mer frihet att styra och ställa med widgets och stilar,

men Geckoboards färdiga stöd för en del tjänster och bättre dokumentation sågs som en större fördel.

3.2.2 Widgets

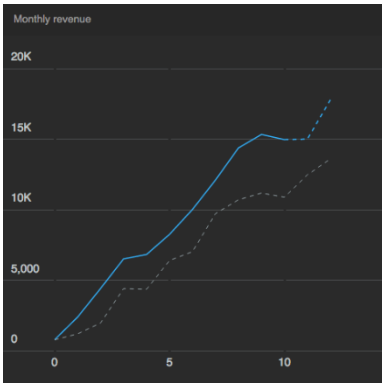
Geckoboard har flera olika typer av färdiga widgets att använda, varav en tabell med en beskrivning av de väsentligaste visas nedan (Figur 2-7).



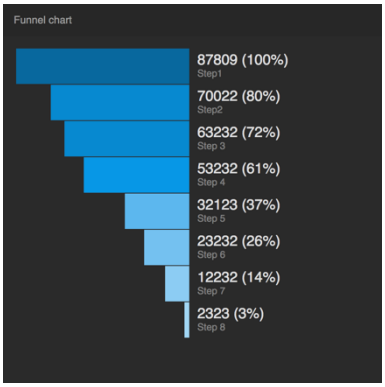
Figur 2: Textwidget - Används för att visa text.

Figur 3: Numberwidget - Visar en siffra som uppdateras varje gång nya data fås in. Kan också visa förändringen över en viss tid.

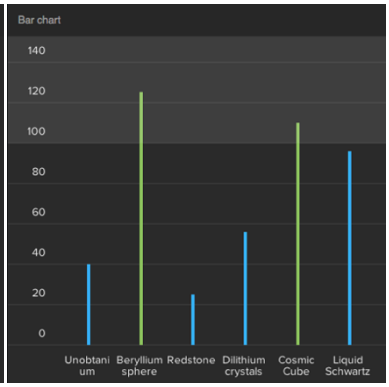
Figur 4: Leaderboard - Visar nyckel-värde data i stigande eller sjunkande ordning enligt värde.



Figur 5: Line graph - Visar förändring i data över en längre tidsperiod. Kan jämföra eller visa flera värden samtidigt.



Figur 6: Funnel graph - Visar hur olika skeden av något framskrider.



Figur 7: Bar chart - Används främst för att visa data uppdelad i olika kategorier.

3.2.3 Färdiga integrationer och dataset

Geckoboard har ca 60 färdigt integrerade tjänster och källor för data. Genom att rakt från sin dashboard kunna lägga in sin inloggningsinformation, eller API nycklar, för diverse tjänster har de gjort det så enkelt som möjligt för sina användare att komma åt sin data. Varje tjänst man kan ansluta till via dashboarden har egna och ett begränsat antal widgets att välja mellan, vilka tyvärr i vissa fall är väldigt bristfälliga. Även om man via de färdiga integreringarna har möjlighet att få allt data man behöver, kunde man inte alltid fritt plocka vad man ville visa.

Geckoboard har också en API för dataset, som gör det möjligt för användarna att skapa sina egna integreringar med dashboarden. Till en början betyder det mera arbete för utvecklaren, man måste själv lyckas kontakta tjänstens API, få fram, behandla och strukturera den data man behöver så att den känns igen av Geckoboard. All data skickas i JSON –format och alla dataset kan i Geckoboard granskas som en tabell. När man dock fått automatiseringen i skick och fått data över till Geckoboard är det enkelt att välja vilken slags widget man vill visa datan i. Med bra strukturerad data kan man dessutom laga flera olika widgets av samma dataset.

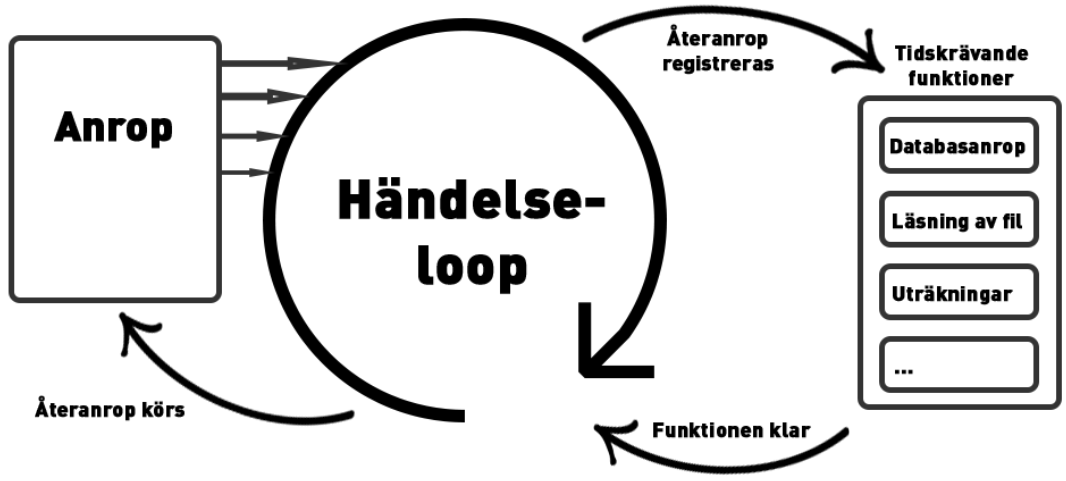
Som ett exempel för båda fallen kan tjänsten Trello, en tjänst för att underlätta projekthantering genom att dela upp det i mindre uppgifter, som Geckoboard erbjuder en färdigt gjord integrering till nämnas. Geckoboard kan visa en fin funnel graph –visualisering av den totala mängden uppgifter i olika stadier för samtliga projekt, men för att kunna visa mängden uppgifter var och en av de anställda har ogjorda var man tvungen att själv skapa ett dataset och skicka det till Geckoboard.

3.2.4 Node.js roll i arbetet

Node.js (Node) är en händelsedriven plattform skriven i JavaScript. Istället för att köras i en besökares webbläsare körs Node på serversidan, i Googles JavaScriptmotor V8, och används till att skapa skalbara nätverksapplikationer (Rauch, 2012). Då man till exempel kör PHP på en Apache server skapar Apache en ny tråd eller process för varje anrop som skickas till tjänsten. Varje sådan tråd belastar och kräver minne av servern den körs på.

När man använder PHP för att hämta något ur en databas står resten av programmet stilla tills anropet är gjort och man har fått tillbaka ett svar. Detta kallas för synkron programmering. Problemet med synkron programmering är att om det tar en lång tid för databasen att svara händer inget på en längre tid och programmet eller tjänsten upplevs som trög.

Till skillnad från Apache har Node endast en tråd den arbetar med, men använder istället en händelseloop. Node bryr sig inte om att vänta på att anropet till databasen har fått ett svar, utan kan under tiden börja arbeta på nästa anrop i händelseloopen. När svaret från databasen slutligen kommer fram använder sig Node av återanrop, en funktion som behandlar data man fått som svar. Node baserar sig långt på återanrop och använder dem, tillsammans med hanterande av felmeddelanden, för att se till att servern inte hamnar i ett dödläge. Ett bra exempel på skillnaderna är hur beställningar tas emot på en restaurang. (Stackoverflow, 2015). Servitören kan ta emot en beställning åt gången, föra beställningen åt kocken, och vänta på att maten blir klar, för att sedan föra ut den åt kunden. Alternativt kan servitören fortsätta ta emot beställningar under tiden kocken tillreder den första. Då kocken har första beställningen färdig meddelar han servitören, som under tiden tagit emot fler beställningar, att den första rätten är klar och kan tas till bordet.



Figur 8: Visualisering av händelseloopen i Node

Node har en pakethanterare, node package manager (npm), som gör det lätt att installera moduler och bibliotek för en applikation. Största orsaken till att använda Node i arbetet var att Geckoboard har ett eget Node bibliotek för att enkelt kunna lägga upp en applikation som kommunicerar med dashboarden. Färdigt skrivna bibliotek finns tillgängliga till en stor del av tjänsterna data kommer att samlas från, så beslutet att använda sig av Node var lätt.

Med Node installerat använde jag npm för att installera moduler och hämta bibliotek, i den ordningen jag behövde dem, under arbetets gång. Exempel på hur paket installeras visas i Figur 9. (npm docs).

```
> npm install <package_name>
```

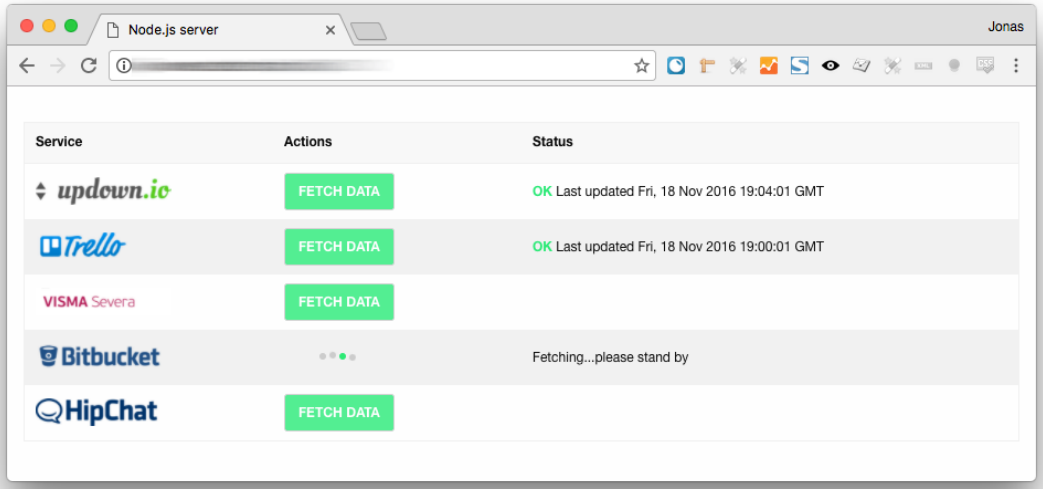
Figur 9: Hur moduler installeras med npm

3.2.5 Express.js och Socket.IO

För att få igång en webbserver med Node, installerade jag modulen Express.js (Express). Express möjliggör att utvecklaren, med endast ett få antal rader kod, kan starta en webbserver som lyssnar på inkommande trafik (Bilaga 1). Tanken med Node servern var att hämtande, formaterande och uppdaterande av data skulle skötas automatiskt. Till en början kunde jag bara testa hämtandet av data från en tjänst åt gången genom komandoraden, och måste göra ändringar i min kod för att byta till en annan tjänst. Jag behövde ett bättre sätt att välja vilken tjänst data hämtas från, utan att vid varje anrop starta om servern. Jag ville se en lista över tjänsterna data hämtas från, och med en knapptryckning kunna köra en funktion på servern för att göra ett nytt API anrop.

Efter en del undersökande i ämnet bestämde jag mig för att prova Socket.IO (Socket). Modulen gör det enkelt att hantera kommunikation mellan flera klienter och servern. Nästan all data som behandlas i projektet skickas i JSON-format, då också Socket använder samma JSON-format var det väldigt lätt att implementera modulen. Nu hade jag en möjlighet att enkelt hämta data, och dessutom skriva ut den i ett sådant format att jag enklare kunde se vilka specifika delar jag behövde av data för respektive tjänst. I ett senare skede

visades status och tidpunkten för när data från varje tjänst senast hämtats, istället för bara rå data.



Figur 9: Simpel webbklient som tillåter manuellt hämtande av data

3.2.6 API:er använda i projektet

Tjänsterna som data samlas ifrån har API:er som alla i stort sätt fungerar på liknande sätt. Samtliga tjänster kräver inloggning för att komma åt att använda API gränssnittet, men istället för att logga in på tjänstens nätsida skickar man sitt användarnamn och lösenord till en viss ändpunkt. För att göra det ännu lättare att autentisera sig över API har de flesta tjänsterna, för varje konto eller användare av tjänsten, en API nyckel som används istället för kombinationen av användarnamn och lösenord. (Nilsson, 2013)

Färdiga Node moduler fanns för alla de utvalda tjänsterna förutom en, vilket möjliggör en lätt kommunikation med tjänsterna via ett API. Den tjänsten som saknar stöd för Node använder sig av protokollet SOAP, där största skillnaden är att meddelanden skickas i XML format istället för JSON. Dokumentationen ger endast ett fåtal exempel om hur man anropar tjänsten med C#, Visual Basic eller PHP. För att testa om önskad data kunde nås, bestämdes att ett av de färdiga exemplen skrivna i PHP skulle fungera som botten. En Node server kan inte köra PHP kod utan extra moduler, så istället lades PHP skriptet upp

på en skild Apache server, och sedan anropades det från Node servern. PHP skriptet skötte om att samla data, plocka ut metriken som behövdes och skickade vidare nyckeltalen till Node servern. API:n var rätt dåligt dokumenterad och talen man sökte efter nåddes sällan med färre än två anrop, det senare baserat på data från det tidigare. Detta, kombinerat med det faktum att XML tar längre att parsa än JSON, ledde till att hämtande av data från just denna tjänst var överlägset mest tidskrävande. Tillgängligt fanns också Node moduler för att lägga upp SOAP klienter, och tanken var att vid tillfälle flytta över anropen från Apache servern till Node servern, så att alla anrop kunde köras från samma ställe.

3.2.7 Webhooks

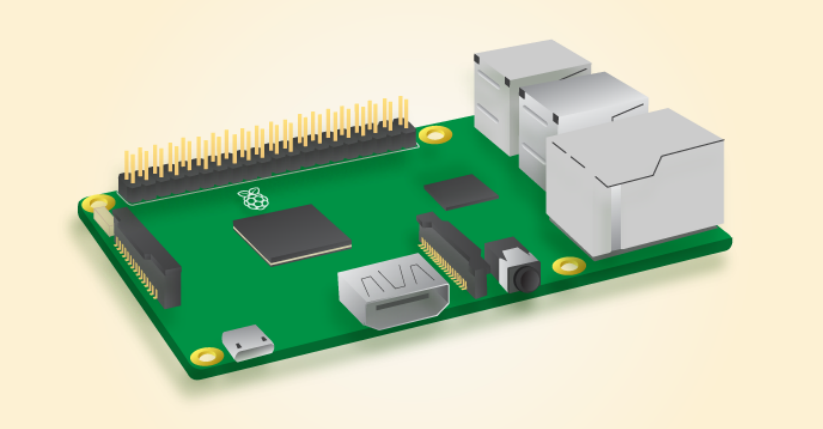
Om man använder tjänster som monitorerar upptiden av servrar, och vill veta när en server går ner, är det inte lönsamt att väldigt ofta skicka en fråga om någon server är nere – idealt är servrarna igång dygnet runt. När det ändå händer att en server går ner vill man få veta om det så fort som möjligt. Webhooks är data som skickas till en bestämd URL efter bestämda händelser (Brown, P.). URLen, som i detta fall är en URL på Node servern, tar emot datan och meddelar avsändaren att paketet kom fram. Med hjälp av webhooks på monitoreringstjänsten får man ett meddelande om när en server går ner, och Node servern ser till att meddelandet visas på dashboarden utan större fördröjning.

Exempelvis så använder företaget ett chatprogram, för att skicka meddelanden mellan de anställda. Chatprogrammet låter sina användare dra nytta av deras API för att registrera webhooks för en stor mängd händelser, i arbetet utnyttjades detta för att kunna visa meddelanden, från ett visst så kallat chatrum, på dashboarden. Chatprogrammet registrerar när någon skriver ett nytt meddelande, informerar Node servern om detta, och servern skickar meddelandet vidare till en specifik widget i Geckoboard.

3.3 Produktionsmiljö

All kod är skriven och testad på en MacBook Pro. När arbetet är färdigt körs Node servern på en Raspberry Pi –dator med operativsystemet Raspbian. Förutom Node, tillsammans med en del moduler för att köra servern, behövdes endast en webbläsare i fullskärmsläge

för att visa upp dashboarden. Datorn kopplades med HDMI-kabel fast i en TV-skärm. Raspberry Pi är en dator designad för att vara billig, lätt att använda och klara av det mesta man kan göra med en bordsdator. Den har ungefär samma storlek som ett kreditkort och lämpar sig bra för projektarbeten. Modellen som används i detta arbete är en Raspberry Pi 3 B (Figur 1). Datorn valdes till arbetet för att den var billig, lätt att sätta upp och har tillräckligt med minne för att både köra Node servern och visa upp dashboarden på en skärm. För att enkelt kunna styra datorn skaffades också ett trådlöst tangentbord med inbyggd styrplatta.



Figur 10: Raspberry Pi 3 B (Raspberry Pi)

4 IMPLEMENTATION OCH RESULTAT

I följande kapitel beskrivs noggrannare hur verktygen användes för att skapa nätverkapplikationen. Dessutom presenteras resultatet av programmeringsdelen och ett exempel på hur en färdig dashboard kan se ut i Geckoboard.

4.1 Strukturering av koden

4.1.1 Laddning av moduler med require

Funktioner eller variabler inskrivna i en .js-fil kan i Node inte rakt användas i någon annan fil. För att inte behöva skriva ett helt program i endast en fil använder man sig istället av require, en modul som kommer färdiginstallerad med Node. Modulen require låter en ladda in andra filer som moduler för att komma åt deras funktioner (Figur 12).

calculator.js

```
// Fil med funktion som skall laddas

exports.add = function(a,b) {    // Gör funktionen add() tillgänglig för andra filer

        console.log(a+b);        //Skriver ut värdet för a+b i konsolen
}
```

main.js

```
// Fil som laddar in en funktion från calculator.js

var calc = require('calculator');        // Laddar in calculator.js som en modul med require()
calc.add(2,4);                           // Kör add() med värden 2 och 4

// 6                                     // Resultatet 6 skrivs ut i konsolen
```

Figur 11: Exempel på hur moduler laddas med require()

Filen med koden för att starta Node servern kallade jag för server.js. Efter att med npm ha installerat paketen för tjänsterna som data skulle samlas från, skapade jag, för att hålla någon slags struktur, en egen .js fil för var och en av dem. Med de egna funktionerna inskrivna laddade jag in dem i server.js, där det bestämdes när de skulle köras.

4.1.2 Schemaläggning med node-cron

En del av tjänsterna har data som ofta ändras, eller är viktig att granskas ofta. Andra tjänster har data där det inte sker förändringar i på en längre tid. Det snabbt klart för mig att det inte var effektivt att hämta data, från alla tjänster, med lika långa pauser emellan. Cron är ett program för schemaläggning av uppgifter skapat för UNIX-baserade system, och med paketet node-cron får man funktionaliteten till Node. Med node-cron kan man lägga till egna funktioner, att köras med valfria mellanrum, i händelseloopen i Node (Figur 13).

```
var cron = require('node-cron');                    // Laddar cron-modulen
var exampleservice = require('services/example');   // Laddar in tjänsten data hämtas från

cron.schedule('0 */15 * * * *', function(){      // Cron lägger till funktionen i händelse-
    exampleservice.getData();                    // loopen med 15 minuters mellanrum
});
```

Figur 12: Exempel på hur funktioner läggs till i händelseloopen med node-cron

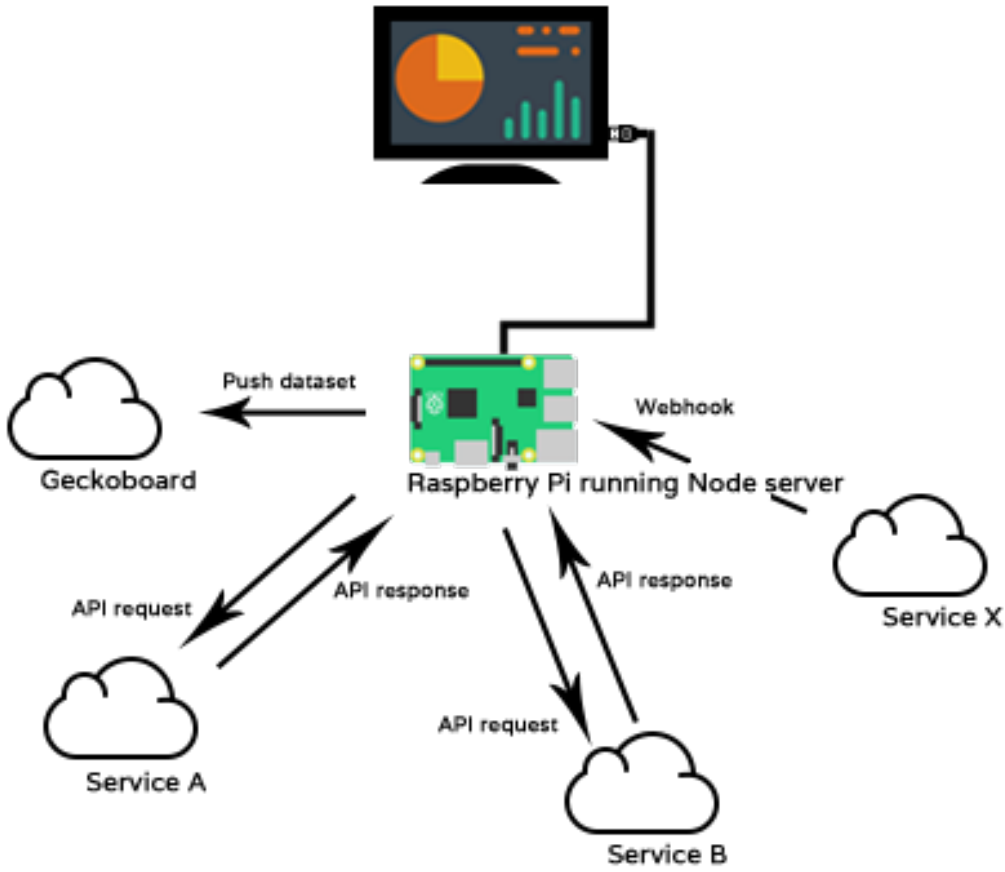
4.1.3 Data till Geckoboard

Det finns flera sätt att få in sin data i Geckoboard, men i detta arbete användes främst deras dataset. Skapandet och uppdaterande av dataset fungerade fint med hjälp av Geckoboards Node modul. När ett dataset skapas ges det ett id, och typen av data definieras (Bilaga 3). Geckoboard märker när ett dataset uppdaterats, och visar förändringarna på dashboarden automatiskt. Se Bilaga 4 för en exempel dashboard på geckoboard.com.

Ett sätt att skapa textwidgets ur dataset var ännu inte möjligt. För att få fram meddelanden från HipChat till dashboarden måste istället en metod, där man använder textwidgetens unika URL för att posta data rakt till den, användas.

4.2 En fungerande applikation

Med att använda metoder och verktyg beskrivna i arbetet skapades en sådan nätverksapplikation som var planerad. Förutom de viktigaste, tidigare nämnda moduler, användes också ett fåtal andra för att underlätta parsande och granskande av data. Servern var igång på Raspberry Pi -datorn och cron såg till att data hämtades med bestämda mellanrum. API-anropen fungerade, data behandlades och skickades vidare till Geckoboard. Geckoboard i sin tur plockade fram bestämda data ur dataseten och visade upp den i olika widgets på dashboarden (Figur 14). Webbklienten, som monitorerade tiderna för när data senast hämtats för var och en av tjänsterna, sattes upp med Socket, HTML och JavaScript. Med hjälp av webbklienten var det enkelt att följa med när det hade skett något problem, dessutom skrevs alla felmeddelanden ner på en loggfil på servern.



Figur 13: Illustration över hur data rör sig i applikationen

En del problem uppstod naturligtvis under arbetets gång. Speciellt när hämtande av data testades, gjordes en stor mängd API-anrop, något de flesta tjänster har satt begränsningar på. Detta resulterade i att inga anrop kunde utföras på en längre tid, så man hamnade tillfälligt testa någon annan tjänst eller utveckla någon annan del av projektet. I början av arbetet orsakade händelseloopen en del bekymmer, men efter att man studerat dokumentationen och förstod hur återanrop i Node fungerar fick man ett smidigt fungerande system. Så länge man fångade alla felmeddelanden var det alltid relativt lätt att hitta orsaken till problemen.

5 SLUTSATSER

Data har, sedan lämpliga verktyg och teknologier funnits, samlats och sparats. Företag har sparat resultat av försäljning, inköp, eller andra affärer, för att kunna se en historia över företaget. Med alla företag som idag tävlar om uppmärksamhet på marknaden räcker det inte länge med ett ”keep it up” –tankesätt när det ser ut att gå bra. För att hållas med i trenderna och fortsätta nå sina målgrupper är det viktigt att ta till vara all information man kan och vara färdig att anpassa sig. Anpassningen till en datadriven strategi gäller inte bara dem som fattar beslut, utan betyder ofta att hela företaget bör struktureras om. Innan metoder för hur samlad data kan användas för att få en blick framåt, istället för bakåt, sattes inte heller så mycket tanke på hur data presenterades visuellt.

Det viktigaste med visualiseringen är inte att den är i 3D, har mycket färger, eller använder en snygg font. När man visar upp data skall man snabbt kunna se den informationen man söker utan andra distraktioner. All data kan inte heller visas i en likadan graf, resultatet måste avspegla typen av data. Genom att visa samma data på flera olika sätt kan man dessutom visa ytterligare detaljer över viss data. Dashboards är inte längre ett verktyg som endast används av företagsledare, utan leder med stilenlig implementation till positiva resultat. Dashboards med data anknuten till de anställdas arbetsuppgifter har setts öka motivationen, och ger dem dessutom en bättre chans att bidra med sina kunskaper.

För att sköta automatisk hämtande och visualiserande av data skapades en nätverksapplikation skriven i Node.js. Målet med applikationen var att eliminera behovet för en anställd att manuellt hämta rapporter eller tabeller från de tjänster företag har i användning. Med den mänskliga faktorn avlägsnad kan man försnabba processen o minska fel. Förutom att hämta data gör applikationen uträkningar och formaterar data till sådan form att dashboardtjänsten Geckoboard har lätt att visualisera den. Mängden kod som sist och slutligen krävdes för att köra nätverksapplikationen var förvånansvärt liten. Med Node var det väldigt enkelt att arbeta med tjänsternas API:er, dessutom fungerade installationen av samtliga verktyg smärtfritt i de olika miljöerna. Fungerande struktur och enkel laddning av moduler lämnade ännu rum för framtida förbättringar. En modul för SOAP-anropen och koppling till en egen databas för bättre lagring av data skulle kunna vara en bra uppdatering.

I stort sett var jag väldigt med nöjd med valet att använda Geckoboard som tjänst för dashboarden. Node modulen fungerade problemfritt och dokumentationen var både tydlig och utförlig. Det gladde mig att se att tjänsten kom med nya egenskaper och funktioner ganska ofta under hela tiden applikationen utvecklades. Med mera tid skulle jag gärna tagit en närmare titt på andra tjänster som erbjuder dashboards med öppen källkod. Det skulle öppna möjligheter för ännu noggrannare anpassningar och visualiseringar.

På basen av andras undersökningar och mina egna observationer är jag fullt övertygad om att data kommer att fortsätta vara en viktig faktor för att omsätta strategier till operativa mål i företag. Dashboards är trevliga att jobba med, påverkar positivt sin omgivning när de används rätt, och kan utan problem få sin data från en server skriven i Node på en Raspberry Pi.

KÄLLOR

Ballou, B. Heitger, L. Donnell, L. 2010. *Creating Effective Dashboards. Strategic Fi nance* (Mar 2010). USA: Institute of Management Accountants. ISSN 1524833X

Brown, P. 2014. *What are webhooks?* Tillgänglig: http://culttt.com/2014/01/22/web hooks/ Hämtad 18.11.2016.

Brynjolfsson, E. 2011. Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance? Tillgänglig på SSRN: https://ssrn.com/abstract=1819486 Hämtad 5.12.2016

Few, S. 2006. *Information Dashboard Design: the effective visual communication of data*. USA: O'Reilly Media. 223 s. ISBN 0596100167

Few, S. 2007. *Data Visualization: Past, present and future.* Cognos Innovation Center for Performance Management 10.01.2007. Tillgänglig: https://www.perceptua-ledge.com/articles/Whitepapers/Data_Visualization.pdf Hämtad: 15.11.2016

Geckoboard. 2016. Tillgänglig: http://wwww.geckoboard.com Hämtad 14.11.2016

Gaffney, K. *One on one with Alexander Chiang. Media Relations, Dashboard Insight,* 16.04.2009. Tillgänglig: http://www.dashboardinsight.com/articles/one-on-one-with-dashboard-insight/one-on-one-with-alexander-chiang.aspx Hämtad 20.11.2016.

Halper, F. 2011. *How To Build A Metrics-Driven Company*. Tillgänglig: http://www.dashboardinsight.com/articles/business-performance-manage-ment/how-to-build-a-metrics-driven-company.aspx Hämtad: 15.11.2016

IT-ord. 2016. Tillgänglig: http://it-ord.idg.se/
Hämtad 10.11.2016

Nilsson, M. 2013. *Vad är API:er och hur använder man dem?* Tillgänglig: http://www.portablamedia.se/vad-ar-api-er-och-hur-anvander-man-dem/ Hämtad 16.11.2016

Node.js. 2016. Tillgänglig: https://nodejs.org/en/about/
Hämtad: 16.11.2016.

npm. 2016. *Getting Started, Installing npm packages locally.* Tillgänglig: https://docs.npmjs.com Hämtad 16.11.2016.

Raspberry Pi. 2016. Tillgänglig: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ Hämtad: 16.12.2016

Rauch, G. 2012. *Smashing Node.js: JavaScript Everywhere.* Storbritannien: John Wiley and Sons Ltd. 320 s. ISBN 978-1-119-96259-5

Rouke, P. 2015. *Five characteristics of businesses ready to grow through data driven optimisation.* Tillgänglig: https://econsultancy.com/blog/66339-five-characteristics-of-businesses-ready-to-grow-through-data-driven-optimisation/ Hämtad: 15.11.2016

Saporito, P. 2008. *Metrics Management. Best's Review* 02.2008. USA: A.M. Best Company. ISSN 15275914

Savkin A. 2015. *What's the Difference Between a Dashboard and a Balanced Scorecard? BSC Designer*, 11.05.2015. Tillgänglig: http://www.bscdesigner.com/dashboard-vs-balanced-scorecard.htm Hämtad 27.11.2016.

Stackoverflow, 2015. *How does Asynchronous programming work in a single threaded programming model?* Tillgänglig: http://stackoverflow.com/a/8989174 Hämtad 16.11.2016

Sten, P. 2016. *Det här är datadriven strategi* Tillgänglig: http://www.rebeland bird.com/datadriven-strategi/ Hämtad: 16.11.2016

Watson, H J. Hill, J. 2009. *What Gets Watched Gets Done: How Metrics Can Motivate. Business Intelligence Journal* 14.3. USA: Data Warehousing Institute ISSN 1547-2825

Wise, L. 2010. *A Closer Look At Scorecards And Dashboards. Dashboard Insight,* 27.04.2010. Tillgänglig: http://www.dashboardinsight.com/articles/digital-dash-boards/fundamentals/a-closer-look-at-scorecards-and-dashboards.aspx Hämtad 27.11.2016.

BILAGOR

```
var express = require('express')
var app = express()


app.get('/', function (req, res) {
  res.send('Hello World!')
})


app.listen(3000, function () {
  console.log('Example app listening on port 3000!')
})
```

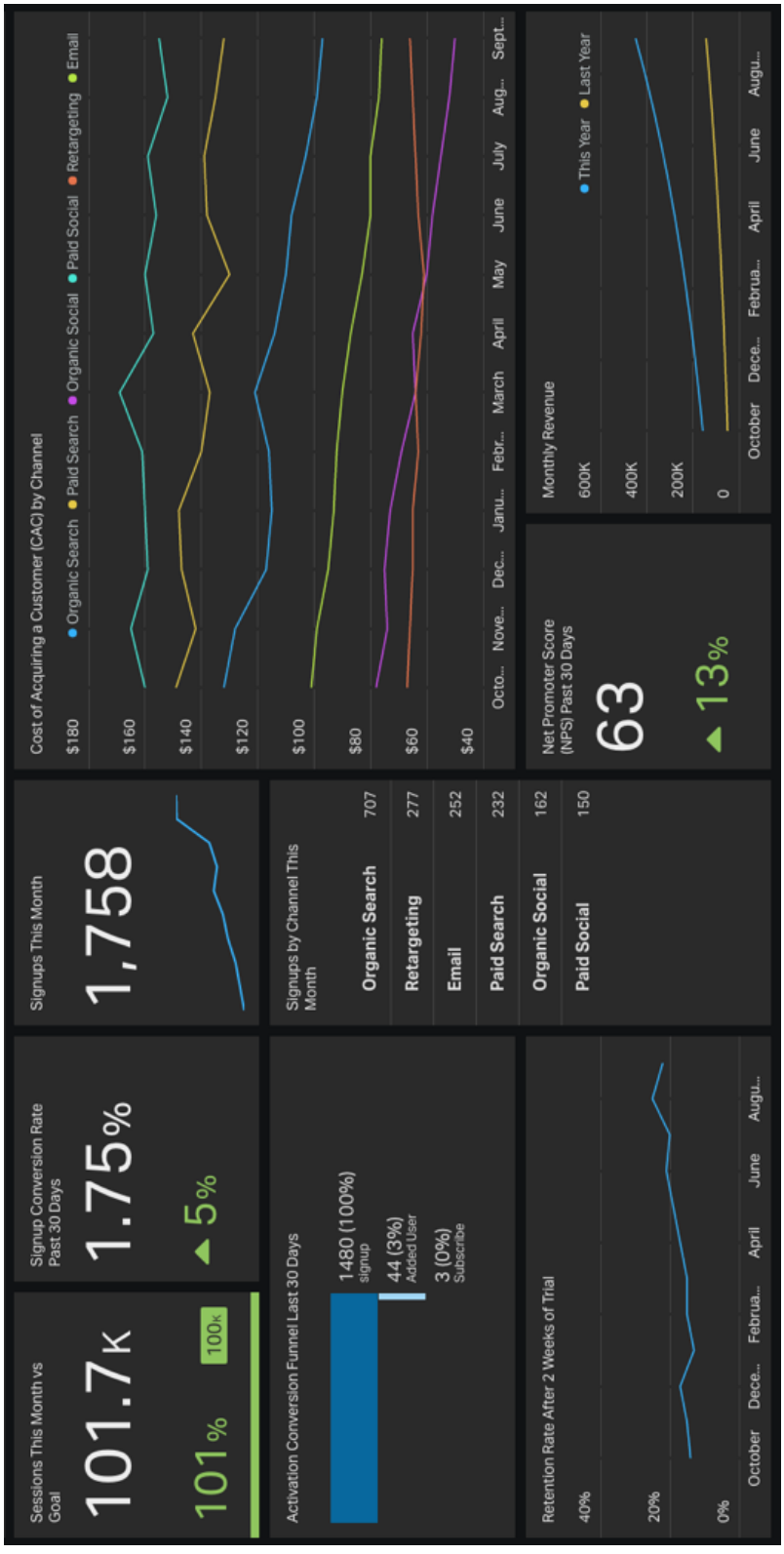
Bilaga 1: Kod för att starta en webbservice med Node och Express

```
app.all('/announce', function(req, res){

        var announcer = req.body.item.message.from.name;
        var message = req.body.item.message.message;

        message = '<span style="font-size:16px;font-weight:bold;color:#90c564;">'+an-
        nouncer+'</span><br />'+message;

        announcement = {
                "item": [
                        {
                                "text" : message,
                                "type" : 0
                        }
                ]
        };

        gecko.announceToGecko(announcement);

      res.status(200);
      res.send();

});
```

Bilaga 2: Hur ett webhook meddelande tas emot och behandlas på Node servern.

```
exports.dataToGecko = function(data) {
    gecko.datasets.findOrCreate(
        {
            id: 'exampleset.closed_deals',
            fields: {
                date: {
                    type: 'date',
                    name: 'date'
                },
                name: {
                    type: 'string',
                    name: 'Customer'
                }
                […]
            }
        },
        function (err, dataset) {
            if (err) {
                console.error(err);
                return;
            }

            dataset.put(
                data,
                function (err) {
                    if (err) {
                        console.error(err);
                        return;
                    }
                    console.log('Data successfully sent');
                }
            );
        }
    )
};
```

Bilaga 2: Hur ett dataset skapas och data skickas till Geckoboard



Bilaga 4: Exempel på en Geckoboard dashboard med olika widgets