jamk.fi

Google Analytics Suiten hyödyntäminen asiakaskyselyissä

Joonas Tuttavainen

Opinnäytetyö
Tammikuu 2019
Tekniikan ja liikenteen ala
Insinööri (AMK), Mediatekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Tuttavainen, Joonas	Opinnäytetyö, AMK	28.01.2019
	Sivumäärä 59	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Google Analytics Suiten hyödyntäminen asiakaskyselyissä

Tutkinto-ohjelma
Mediatekniikka

Työn ohjaaja(t)
Pasi Manninen, Kari Niemi

Toimeksiantaja(t)
-

Tiivistelmä

Opinnäytetyön tavoitteena oli tutkia Google Analytics Suitea ja voisiko tätä hyödyntää asiakaskyselyjen toteuttamisessa. Ennakkoedellytyksenä oli, että Google Analyticsin täytyy tukea verkkosovelluksen ohjelmakoodin etäajoa, jonka avulla haluttiin helpottaa sovellusten toimitusprosessia asiakkaan ja ohjelmoijan välillä.

Lisäksi käytiin läpi web-sovellusten suunnitteluun liittyviä käytänteitä alkaen responsiivisuudesta HTML-lomakkeiden tietoturvaan ”prepared statement” -lausekkeiden avulla sekä hyviä ohjelmointiperiaatteita kuten nimeämisristiriitojen välttämistä. Opinnäytetyötä tehdessä responsiivisuus sekä tietoturva haluttiin heijastaa Analytics Suiteen, jotta niiden sovellettavuus Analytics Suitessa sekä käytännössä tulisi todistetuksi.

Google Analytics Suiten etäajo-ominaisuuksia testattiin ohjelmoimalla ja lataamalla JavaScriptin ja PHP:n avulla toteutettu NPS-asiakaskysely kuvitteellisen asiakkaan sivuille. Tiedostot, joista kysely koostui, ladattiin halutulle sivustolle Analyticsin kautta Ubuntu-verkkopalvelimelta.

Juuri luodun NPS-asiakaskyselyn toiminta varmistettiin vastaamalla kyselyyn ja tallentamalla tulokset MySQL-tietokantaan testikäyttäjänä. Tulokset näytettiin pylväsdiagrammina annettujen vastausten kokonaismääränä Analytics Suiten sisällä. Joidenkin ohjelmistokehysten, kuten Bootstrapin, todettiin häiritsevän kyselyn ulkoasua ja CSS-tyylejä. Vika kuitenkin kohdennettiin kyselyn ohjelmointilogiikkaan Google Analyticsin sijaan, vaikkei tarkempi syy selvinnyt.

Onnistuneesti Analyticsiin haettujen vastausten perusteella tultiin kuitenkin johtopäätökseen, että Analytics Suitea voidaan hyödyntää asiakaskyselyjen ohjelmoinnissa. Lisäksi sen osoitettiin soveltuvan responsiivisten ja tietoturvallisten verkkosovellusten etäajoon.

Avainsanat (asiasanat)

Google Analytics, Tag Manager, Data Studio, Asiakaskysely

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Tuttavainen, Joonas	Bachelor's thesis	28.01.2019
		Language of publication: Finnish
	Number of pages 59	Permission for web publication: x

Title of publication

Utilization of Google Analytics Suite in customer survey execution

Degree programme

Media Engineering

Supervisor(s)

Manninen, Pasi; Niemi, Kari

Assigned by

-

Abstract

The objective of the thesis was to explore Google Analytics Suite and whether it can be used in executing customer surveys. The prerequisite was that Google Analytics must support remote execution of web program code in order to facilitate the delivery process of web applications between the client and the programmer.

In addition, practices regarding web software design were covered, ranging from responsiveness to HTML form security through prepared statements and good programming principles such as avoiding variable name collisions. During the creation of the thesis, responsiveness and data security were reflected on Analytics Suite in order to prove their applicability with Analytics Suite and in practice.

Google Analytic Suite's remote execution capabilities were tested by programming an NPS customer survey using JavaScript and PHP, which was then loaded onto a page of an imaginary customer. The survey and the files that make it up were loaded onto the desired page through Analytics from an Ubuntu web server.

The functionality of the recently created NPS customer survey was confirmed by replying to the survey as a test user and saving the results into a MySQL database, after which they were then displayed as a bar chart inside Analytics Suite. It was found out that some frameworks such as Bootstrap may interfere with its layout and CSS-styling. However, the problem was caused by the survey's programming logic and it was not related to Google Analytics.

It was, nevertheless, concluded that after successfully fetching the survey results into Analytics Suite, it can be utilized to program a customer survey. It was also shown that it can be used to remotely execute web applications made to be both responsive and secure.

Keywords/tags (subjects)

Google Analytics, Tag Manager, Data Studio, Customer Survey

Miscellaneous

Sisältö

Kuviot

Taulukot

1 Työn lähtökohdat

1.1 Asiakaskyselyjen merkitys yritysmaailmassa

1.1.1 Mihin asikaskyselyjä tarvitaan?

Tämän opinnäytetyön taustana toimii työelämälähtöinen tarve järjestelmälle, jonka avulla on mahdollista tarjota helppokäyttöistä asiakastyytyväisyyttä mittaavaa sovellusta yritysasiakkaille. Yrityksen liiketoiminnallisen menestyksen kannalta asiakastyytyväisyys on vain yksi tekijä, mutta se on tärkeää ei pelkästään taloudellisessa mielessä vaan myös yrityksen brändin kannalta. (Kierczak n.d.)

Yksi tehokkaimmista asiakastyytyväisyyden mittaustavoista on suoran palauteen kerääminen asiakaskyselyn avulla. Kyselyn avulla on helppo selvittää, korreloivatko esimerkiksi laskeva myynti ja asiakkaiden mielikuva yrityksen palveluista keskenään, vai ovatko taustalla muut kuten taloudelliset syyt. Vastaavaa tietoa ei ole mahdollista saada pelkän tekoälyn keräämään asiakasdatan avulla, vaan tähän tarvitaan suora asiakaskontakti. Tekoälyä voidaan kuitenkin hyödyntää muuhun tekniseen mittaukseen, kuten siihen, kokevatko kuluttaja-asiakkaat yrityksen verkkosivujen käytön hankalaksi, ts. löytävätkö nämä etsimänsä asian tai tuotteen sivuilta. Tämä heijastuu yleensä suurina poistumisprosentteina tietyssä kohtaa sivua.

Keskiarvoisesti tyytyväinen ja palaava asiakas tulee käyttämään tulevaisuudessa rahaa 10 kertaa enemmän yrityksen palveluihin, kuin mikä oli asiakkaan ensimmäisen ostoksen arvo. Vastavuoroisesti on 6-7 kertaa kalliimpaa hankkia uusi asiakas, kuin menettää pidempiaikainen asiakas tyytymättömyyden vuoksi. Suora asiakaskysely on nopein tapa vastata asiakkaan tarpeeseen ja korjata tämän kokemat epäkohdat. (Kierczak n.d.)

1.1.2 Onko markkinoilla tarvetta uudelle asikaskyselylle?

Koska asiakastyytyväisyyden mittaaminen on selvästi tärkeää, haluttiin tämän vuoksi lähteä rakentamaan omaa asiakaskyselyä, jota voitaisiin lähteä myymään yritysasiakkaille liiketoiminnan päätöksenteon tueksi. Erilaisia asiakaskyselyjen luontiin tarkoitettuja alustoja on kuitenkin jo olemassa, joista esimerkkinä jo kohta 20 vuotta toiminut SurveyMonkey-niminen asiakaskyselyihin erikoistunut yritys.

Tarkoituksena ei kuitenkaan ollut kilpailla suoraan jo olemassa olevien ratkaisujen kanssa, vaan tuoda mukaan jotain uutta. SurveyMonkeyn avulla voidaan helposti luoda asiakaskyselyjä valmiista kyselypohjista. Niiden heikkoutena on, että luodut kyselyt sijaitsevat kiinteästi yrityksen omilla palvelimilla. Tämän vuoksi asiakas joudutaan ohjaamaan aina erilliselle sivulle, jos tämän halutaan vastaavan kyselyyn.

Ratkaisuksi lähdettiin suunnittelemaan kyselyä, joka voidaan toteuttaa ponnahdusikkunan kautta aukeavana kyselynä yritysasiakkaan omilla sivuilla. Tällöin esimerkiksi verkkokaupassa vieraileva asiakas voi vastata kyselyyn mistä kohtaa sivua tahansa, eikä tämä vaadi asiakkaan ohjaamista pois sivustolta. Tämä ei kuitenkaan ole mitään uutta, vaikka onkin kiinteästi tietylle sivulle asetettua kyselyä joustavampi ratkaisu.

Uutena ominaisuutena kyselystä haluttiin sellainen, että sen ohjelmakoodi on mahdollista etäajaa sivuilla, joilla kysely on käytössä. Asiakkaan kannalta tällä tavoiteltiin kyselysovelluksen käyttöönoton vaivattomuutta. Tällöin sovellusta tulevaisuudessa päivittäessä voidaan välttää turhaa päivitystiedostojen siirtoa tai lataamista kehittäjän palvelimelta asiakkaalle, ja lisäksi olisi vain yksi keskitetty palvelin, josta ladattava koodi päivittyisi automaattisesti kaikille kyselyn ostaneille asiakkaille.

Samalla haluttiin päästä eroon ylimääräisestä käyttäjäoikeuksien määrittämisestä sovelluskehittäjää varten, jotta ostettu kyselysovellus tai sen päivitys saataisiin toimitettua asiakkaan palvelimelle. Tätä pidettiin erityisen tärkeänä, koska kaikilla asiakkailla ei ole tähän vaadittavaa tietoteknistä osaamista tai halua. Keskitetyn palvelimen avulla voitaisiin lisäksi tarjota yksittäiselle asiakkaan personoituja ominaisuuksia normaalin kyselyn toiminallisuuden lisäksi. Tällöin etäajon mahdollistavan järjestelmän täytyisi tunnistaa asiakas esimerkiksi IP- tai verkko-osoitteen perusteella.

1.2 Tavoitteet

Alustavan tutkimustyön perusteella todettiin, että sopivin työkaluehdokas asiakaskyselyn toteuttamiseksi olisi Google Analytics Suite. Työkalu näytti tukevan etäajettavan HTML-, CSS-, JavaScript- ja Ajax-koodin suoritusta. Tämän perusteella voitaisiin toteuttaa asiakaskysely, jonka ohjelmakoodi voidaan etäajaa ja muokata ”lennosta” ilman palvelintunnuksia upotettavien tagien avulla. Lisäksi voitaisiin vaikuttaa kokonaisiin sivustoihin tai vain haluttuun osaan sivua.

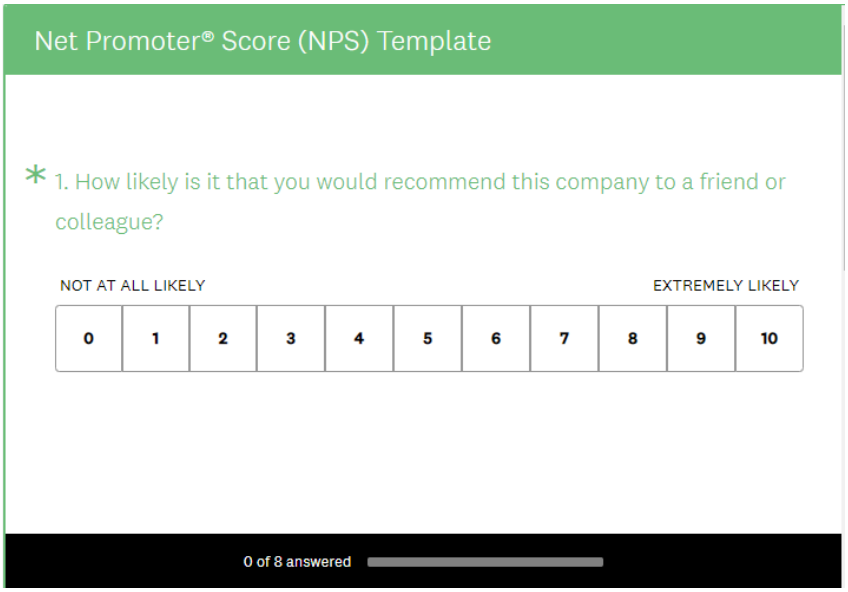
Analytics Suiten käytännöllisyyden selvittämiseksi lähdettiin tutkimaan, soveltuuko se todella etäajettavan NPS-asiakaskyselyn toteuttamiseen. Ensisilmäyksellä sovellus vaikutti olevan kohdennettu enemmän markkinointi- kuin ohjelmistokehityskäyttöön.

Testiä varten haluttiin nostaa esille myös hyviä ohjelmointitapoja ja tekniikkoja, joiden avulla web-sovelluksista saadaan tietoturvallisempia sekä yhteensopivampia eri laitteiden ja selainten kanssa. Näihin kuuluvat esimerkiksi responsiivisuus sekä SQL-injektioiden estäminen HTML-lomakkeen kautta. Tämän jälkeen niitä sovellettaisiin Analytics Suiten avulla toteutettavassa NPS-asiakaskyselyssä, jotta nähtäisiin toimivatko ne myös Analyticsin kanssa yhtä hyvin kuin teoriassa.

2 NPS-kysely ja Google Analytics Suite

2.1 NPS-kysely yleisesti

NPS-kysely (ks. kuvio 1) eli Net Promoter Score on mittaustapa, jonka avulla seurataan asiakastyytyväisyyttä. Kyselyn avulla on mahdollista löytää ne asiakkaat, jotka mahdollisesti suosittelisivat käyttämäänsä palvelua tai sivustoa. NPS-tunnetaan yleisesti käytännöllisyydestään, koska kyselyn tuloksia on helppo tulkita ja vastaaminen on nopeaa. Kyselyssä vastataan yhteen kysymykseen asteikolla 0-10, joka kuuluu seuraavasti: ”Asteikolla 0-10, kuinka todennäköisesti suosittelisit yritystämme kollegallesi tai kaverillesi”. (Net Promoter Score eli NPS n.d.)



Kuvio 1. Esimerkki NPS-kyselystä (Net Promoter Score (NPS) Survey Template n.d.)

Kyselyssä vastaukset on jaettu kolmeen ryhmään, jossa vastaukset asteikolla 10-9 ovat suosittelijoita, 8-7 neutraaleja ja 6-0 arvostelijoita. Suosittelijoille on jäänyt selvästi positiivinen kuva sivustosta tai palvelusta, kun taas arvostelijat ovat tyytymättömiä. Neutraalit äänet tarkoittavat, ettei vastaaja osaa päättää, kallistuuko mielipiteessään enemmän positiiviselle vai negatiiviselle puolelle. Syy myös voi olla se, ettei vastaaja jaksa miettiä omaa vastaustaan ja valitsee helpoimman keskimmäisen arvosanan. Tämän vuoksi kyselyyn suositelleen lisättäväksi myös valinnainen palautelaatikko pelkän arvosanan lisäksi. (Net Promoter Score eli NPS n.d.)

NPS-pisteet lasketaan ottamalla suosittelijoista sekä arvostelijoista prosentuaalinen osuus kaikista vastaajista, ja vähentämällä lopuksi arvostelijoiden prosentuaalinen osuus suosittelijoiden prosentuaalisesta osuudesta. NPS-kysely on monipuolinen, ja sillä voi mitata yrityksen kokonaisvaltaisen toimivuuden lisäksi esimerkiksi yksittäistä tuotetta tai tuoteryhmää. (Mt.)

NPS-kysely on mahdollista toteuttaa joko sähköpostikyselynä tai ponnahdusikkunan kautta suoraan verkkosivuilla. Monipuolisuus, helppo mittaustapa sekä toimiminen myös ponnahdusikkunana vaikuttivat NPS-kyselyn valintaan. Ponnahdusikkuna on lisäksi helppo ladata millä tahansa sivulla vaikuttamatta sivuston omaan ulkoasuun.

2.2 Analytics Suite

2.2.1 Yleistä

Google Analytics Suite, joka on viralliselta kokonimeltään Google Analytics 360 Suite, on markkinointidatan keruuseen ja analysointiin tarkoitettu ohjelmistokokoelma yrityksille. Analytics Suite on Googlen ylläpitämä ja vuonna 2005 aloittama projekti, joka koostuu kuudesta eri tuotteesta, joilla kaikilla on eri käyttötarkoitus ja jotka kuuluvat Analytics Suite -nimellä Google Marketing Platform -brändin alle. Google Analytics Suiteen kuuluvat Analytics, Tag Manager, Optimizer, Data Studio, Surveys ja Attribution. (Google Analytics is 10 years old – What's changed n.d; Moore n.d.)

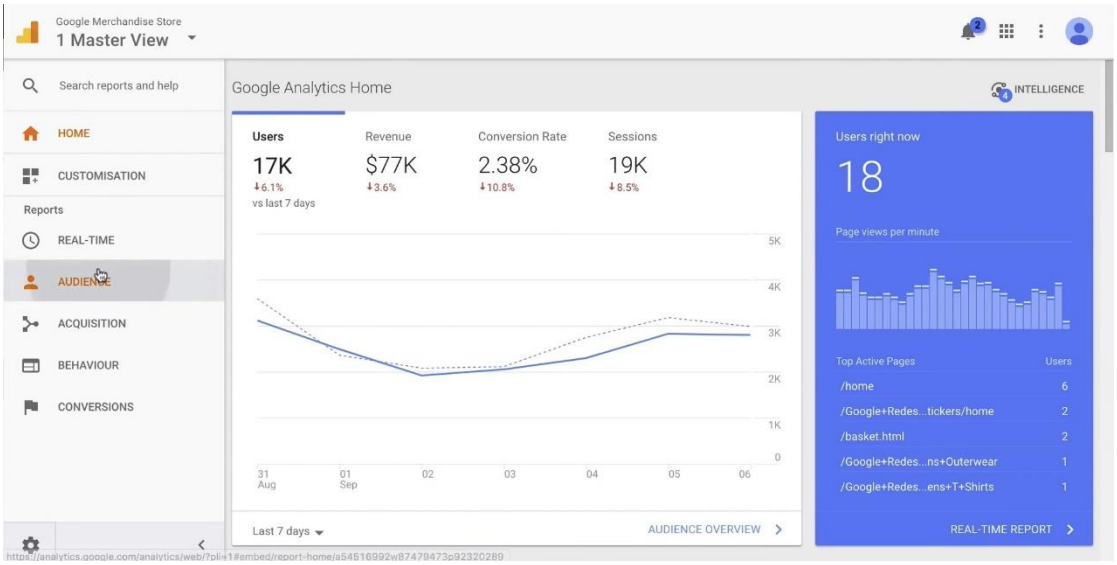
Opinnäytetyön tavoitteiden saavuttamisen kannalta oleellisia ovat vain Tag Manager ja Data Studio, joten muita sovelluksia Analyticsia lukuun ottamatta ei tulla tässä opinnäytetyössä esittelemään.

Analyticsin valinta perustui työkalun ilmaisuuteen, joka pätee 5 miljoonaan kuukausittaiseen sivustolla kävijään asti per Google-tili. Tämän pitäisi kuitenkin olla enemmän kuin tarpeeksi, sillä alle viiden henkilön yrityksissä 5000 - 6000 hengen kuukausittaiset kävijämäärät ovat realistisia lukuja. Näin varsinkin, jos liiketoiminta on maantieteellisesti rajoitettu vain Suomeen tai Pohjoismaihin. (Is Google analytics free of charge 2015.)

Kaikki käyttäjät, joilla on Google-tili, on automaattisesti käytössä kaikki Analytics Suiten ominaisuudet. Tili luodaan aina mihin tahansa Googlen palveluun rekisteröityessä, kuten Gmail tai YouTube. Analytics Suite mahdollistaa navigoinnin sarjan muiden tuotteiden välillä (ks. liite 1), ja tämä tapahtuu klikkaamalla aina kunkin palvelun käyttöliittymän valikosta ”muut tilit” -nappia.

2.2.2 Analytics

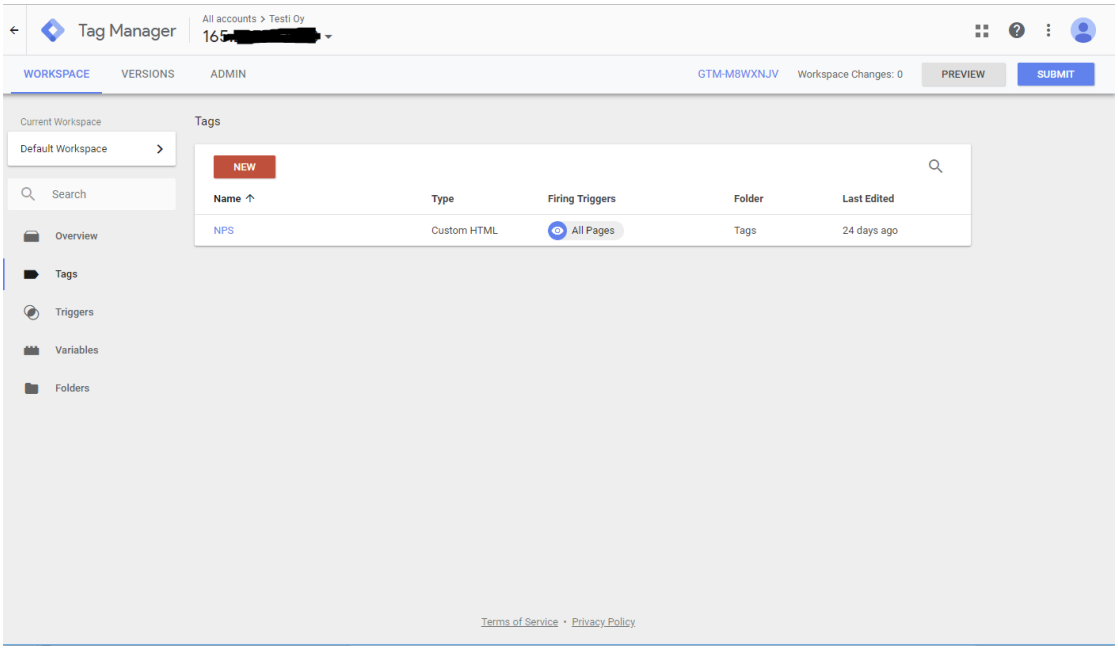
Analytics on Analytics Suiten yksi käytetyimmistä työkalauista ja sisältää web-analytiikan työkaluja hakukoneoptimoinnin ja markkinoinnin tarpeisiin (ks. kuvio 2). Sivuilta kerättyä kävijätietoa on mahdollista tuoda ja visualisoida taulukoihin ja kaavioihin eri ajanjaksoille. Joitain tietoja voidaan myös jakaa alajoukkoihin, kuten esimerkiksi millä laitteilla sivua on selattu ja maan mukaan eroteltuna. (Rouse 2011.)



Kuvio 2. Analyticsin käyttöliittymä

2.2.3 Tag Manager

Tag Managerin avulla voidaan asentaa tageja verkkosivuille, jotka itsessään ovat lyhyitä JavaScript-koodinpätkiä (ks. kuvio 3). Tagien ideana on kerätä käyttäjädataa ja lähettää tämä eteenpäin johonkin kolmannen osapuolen alustaan jatkoanalyysiä varten. Lähetettävistä tiedoista voidaan suodattaa ei-toivotut osumat pois esimerkiksi silloin, jos tuloksista halutaan ottaa talteen vain tietyn maan IP-osoitteista tehdyt sivustovierailut. (Gant n.d; Long 2016.)



Kuvio 3. Tag Managerin käyttöliittymä

Nopealla vilkaisulla Tag Manager ja Analytics voivat vaikuttaa toimintaperiaatteiltaan hyvin samankaltaisilta, sillä myös Analytics hyödyntää sivulle asetutteja tageja tiedon keruuseen. Tag Manager ei kuitenkaan itsessään sovellu tiedon analysointiin, vaan se on tarkoitettu käytettäväksi Analyticsin tai jonkin muun tuetun sovelluksen kanssa. Googlen tuotteena Tag Manager sisältää suoran tuen Analyticsille. (Gant n.d.)

Tag Managerin vahvuus verrattuna Analyticsiin tulee siitä, että Tag Manager voidaan ohjelmoida lähettämään sivustovierailijasta kerättyä dataa vain silloin, kun tämä suorittaa jonkin toiminnon (ks. kuvio 4). Toiminnoksi voidaan laskea esimerkiksi minkä tahansa tekstikappaleen maalaaminen 10 kertaa peräkkäin.

Kuvio 4. Tag managerin toimintaperiaate (Gant n.d.)

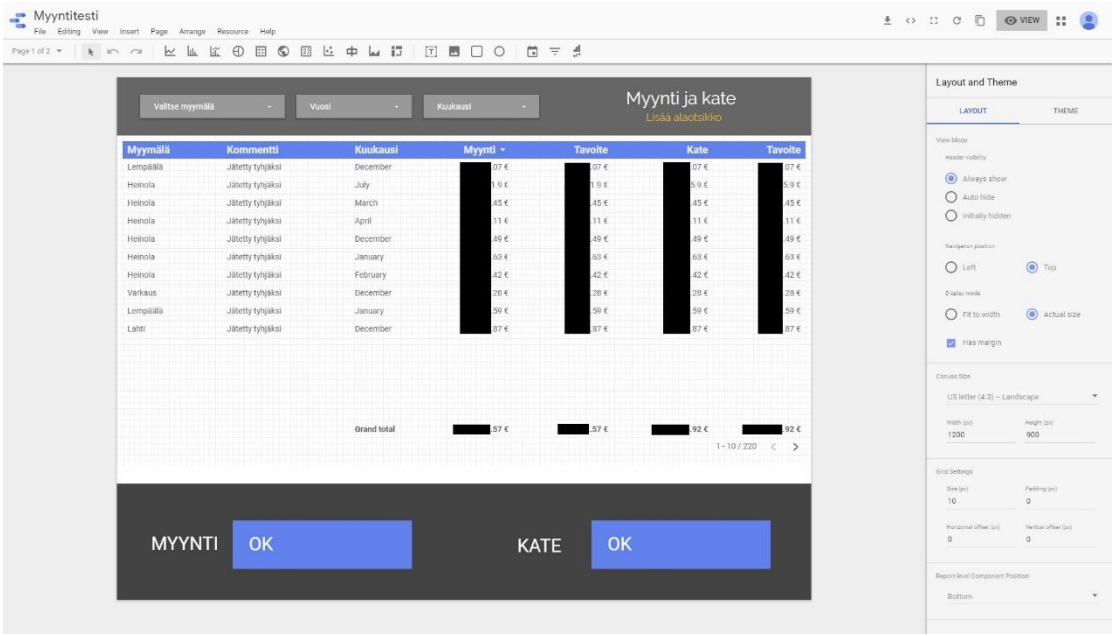
Kerätyn datan lähettäminen toiseen sovellukseen on mahdollista myös Analyticsin puolella, mutta on tämän suhteen Tag Manageria paljon rajoittuneempi. Yhdistämällä Tag Manager Analyticsiin voidaan määrittää paljon monipuolisemmin, minkälaista kävijädataa Analyticsiin lähetetään, kuin että käytettäisiin pelkästään Analyticsin omia seurantatageja. Tag Managerin yhdistäminen Analyticsiin tai johonkin muuhun kolmannen osapuolen asiakasdatan analysointiohjelmaan ei kuitenkaan ole pakollista. (Gant n.d.)

Tag Manager sisältää erilaisia tagityyppejä, joista yhtenä vaihtoehtona ovat Analytics -tagi tai oma tagi (eng. custom tag). Oman tagin avulla voidaan ohjelmoida tuki sellaisille kolmannen osapuolen analysointiohjelmille, joille Tag Managerissa ei ole vielä omaa tukea. (Custom tags 2018; Deploy Google Analytics with Tag Manager 2018; Harmon 2016.)

Tätä ominaisuutta tullaan rikkomaan ja hyödyntämään opinnäytetyössä etänä ladattavan asiakaskyselyn toteuttamisessa. Tätä varten luodaan oma tagi, joka vain suorittaa asiakaskyselyn JavaScript- tai HTML-koodin, muttei lähetä kävijädataa minnekään.

2.2.4 Data Studio

Data studio (ks. kuvio 5) on raportointityökalu, jolla käyttäjän tuomaa dataa voidaan visualisoida vapaasti muokattavissa olevien raporttien avulla. Data Studio muistuttaa ulkoisesti hyvin paljon Analyticsia Tag Managerin tapaan, mutta suurimmat erot ovat raporttien tiedonkeruutavoissa ja niiden muokattavuudessa. Analyticsissa raporttien tiedot haetaan seurantatagien avulla sivuston kävijäliikenteen seasta, kun taas Data Studio hyödyntää raporttiin yhdistettäviä tietolähteitä (eng. data sources). (What is Google Data Studio 2017.)



Kuvio 5. Data Studion käyttöliittymä

Tietolähteitä ovat kaikki Data Studion tukemat tietokokoelmat, joista raportti voidaan koostaa. Näitä ovat esimerkiksi MySQL-tietokannat tai Google Sheets (Microsoft Excelin kaltainen web-pohjainen taulukkolaskentasovellus). Data Studio tarjoaa lisäksi Analyticsia enemmän työkaluja raportin ulkoasun muokkaamiseen, ja valmis tuotos on mahdollista jakaa muiden käyttäjien katseltavaksi tai muokattavaksi alkuperäisenä tai kopiona. Analytics sisältää vain yhden näkymän, johon käyttäjille on mahdollista myöntää käyttöoikeuksia. (What is Google Data Studio 2017.)

Data Studiota käytetään Tag Managerin avulla toteutettavan asiakaskyselyn tulosten hakemiseen. Tällä tavoin voidaan varmistaa kyselyn toiminta ja palautteen läpimeno käytännössä.

3 Verkkosovelluskehityksen teoriaa ja käytäntöjä

3.1 Verkkosovelluksen testaus

Web-sovelluskehityksen yhteydessä testauksella tarkoitetaan useimmiten ohjelman yhdenmukaisen toiminnan varmistamista eri selaimilla, selainversioilla ja laitteilla. Selaimia ja eri selainversioita on useita. Tämän vuoksi hyvin usein törmätään tilanteeseen, jossa kaikki selaimet eivät tue ohjelmakoodissa käytettyjä JavaScript-funktioita ja CSS-määrityksiä. Usein eroavaisuudet estävät koko ohjelman toiminnan, tai vaikuttavat negatiivisesti sivun käytettävyyteen.

Esimerkiksi CSS3-version mediakyselyt ovat yleisin tapa toteuttaa sivun responsiivisuus, mutta ne eivät toimi kaikilla selaimilla, kuten Internet Explorer 8 tai Opera 10. Kun selain ei tunnista käytössä olevaa CSS-määritystä, on ratkaisuna korvata määritys JavaScriptin avulla. Pääsääntöisesti yhteensopivuusongelmat kasvavat siirtyessä vanhempiin selaimiin ja näitä ovat erityisesti ennen vuotta 2010 julkaistut selaimet. NPS-asiakaskyselyn toteuttamisessa hyödynnettiin paljon mediakyselyjä, jotta kyselyn visuaalinen ilme saatiin säilytettyä eri näyttökoilla. (Can I use css3. n.d.)

3.2 Yleisimmät ongelmat verkkosovelluskehityksessä

3.2.1 Järjestelmien sulauttaminen

Stangarone (2015) kertoo blogissaan, että yksi merkittävistä ongelmista sovelluskehityspuolella on eri järjestelmien sulauttaminen toisiinsa toimivaksi kokonaisuudeksi. Web-kehittäjän haaste ei ole vain sitä, että sovelluksesta löytyy jokainen asiakkaan haluama ominaisuus, vaan yhtä tärkeää on saumaton toiminta eri järjestelmien välillä.

Web-ohjelmat ovat nykyisellään enää harvoin itsenäisiä ohjelmiston toimittajan tai asiakkaan palvelimella sijaitsevia kokonaisuuksia. Ne muodostuvat useista eri ohjel-

mistotoimittajien komponentista ja järjestelmistä, eivätkä edes välttämättä sijaitse maantieteellisesti samalla alueella. Tällöin puhutaan SaaS-järjestelmästä (Software as a Service), jonka ideana on tarjota koko sovellus tai sen osia hajautetusti verkkoyhteyden yli. (Rouse 2016.)

Web-sovelluksen tai kokonaisen verkkosivuston toiminnan kannalta oleellisia osia voivat olla taulukkolaskentaohjelmistot, tietokannat tai muut tietovarastot, pilvipalvelut, raportointityökalut sekä yrityksen jonkin liiketoiminnallisen osan seuraamiseen tarkoitetut edistysraportit – dashboardit eli ”kojelaudat” (Hillsberg 2018).

3.2.2 Ylituotanto ja tietoturva

Integraatio-ongelmien lisäksi usein törmätään ylituotantoon liittyviin kysymyksiin. Stangaronen (2015) mukaan on tärkeää pyrkiä erottumaan massasta, koska pelkkiä mobiilisovelluksiakin on tarjolla jo miljoonia. Tämän suuren valinnanvaran vuoksi yritysasiakkaat ja toisinaan myös yksityisasiakkaat vaativat sovelluksilta enemmänkin kuin sen, että sen avulla ratkaistaan vain tietty yksittäinen tarve tai ongelma. Pelkän sovelluksen perustoiminnallisuuden lisäksi halutaan näyttävyyttä ja nopeutta.

Stangarone (2015) nostaa blogissaan esille myös amerikkalaisen Trustwave Holding-tietoturvayhtiön vuonna 2014 julkaiseman tutkimuksen, jonka mukaan 96 % prosenttia kaikista heidän vuonna 2013 testaamistaan sovelluksista sisälsi vakavan tietoturva-aukon. Haavoittuvuuksien mediaani oli jopa 14 per sovellus, joiden joukosta löytyi myös opinnäytetyössä käsitelty SQL-injektio. (Application Vulnerability Trends Report 2014; Stangarone 2015.)

3.2.3 Toimituksen ja ylläpidon haasteet

Perinteisesti kun asiakkaan tilaama sovellus tai komponentti on valmis, siirretään ohjelman lähdetiedostot tämän osoittamalle web-palvelimelle FTP-yhteyden yli. Vaihtoehtoisesti asiakas voi siirtää tiedostot itse saatuaan ne ensin sähköpostin liitetiedostona tai jollain massamuistilla toimitettuna. Yhteistyö ei kuteinkaan aina lopu tähän, sillä myöhemmässä vaiheessa voi syntyä tarve uusille ominaisuuksille, joista ei oltu sovittu alkuperäistä ohjelmaa toimittaessa.

Edellä kuvattu sovelluksen toimitus- ja ylläpitoprosessi voi muodostua ongelmalliseksi. Kuvitellaan tilanne, jossa asiakas on tilannut upotettavan palautelomakkeen sivuilleen. Sivuja asiakas päivittää itse jonkin CMS-järjestelmän, kuten WordPressin kautta. Asiakas antaa sovelluksen toimittajalle väliaikaiset ja rajatut FTP- ja WordPress-tunnukset, jotta palautelomaketta päästään asentamaan ja testaamaan. Myöhemmin kuitenkin selviää, että lomakkeeseen on jäänyt tietoturva-aukko, joka mahdollistaa lomakkeen hyödyntämisen roskapostittamiseen asiakkaan verkko-osoitteesta. Palautelomakkeen suunnitellut taho korjaa virheen, mutta joutuu pyytämään tätä varten uudet palvelintunnukset.

Vaikka asiakas on jo maksanut tilaamastaan tuotteesta, joutuu tämä näkemään joka kerta erikseen vaivaa, jos lomakkeeseen halutaan tehdä mitä tahansa muutoksia. Toimitetun sovelluksen päivittäminen ja uusien ominaisuuksien lisääminen tulisi onnistua taustalla niin, ettei asiakkaan tarvitsee itse olla prosessissa mukana millään tavalla. Tätä varten tarvittiin järjestelmä, joka Analytics Suiten tapaan mahdollistaa ohjelmakoodin etäajon ja muokkaamisen ”lennosta” ilman palvelintunnuksia ja jolla voidaan vaikuttaa koko sivuun tai vain haluttuun osaan sivua.

3.3 Muuttujien ja funktioiden suojaaminen

Ennen mitä tahansa ohjelmointiprojektia on hyvä olla perillä edes joistain käytänteistä, joiden avulla voidaan ennaltaehkäistä tilanteita, joissa ohjelmakoodi ei enää toimikkaan silloin, kun tätä yritetään ajaa muualla kuin sen alkuperäisessä kehitysympäristössä. Näin voi käydä esimerkiksi silloin, kun yhden asiakkaan sivuilla toimivaksi todettu sovellus ei toimikkaan toisella.

Ohjelmalogiikan pettämisen yksi yleinen syy on, että uudet sivut sisältävät samannimisiä globaaleja muuttujia ja funktioita kuin sivuille ladattava sovellus. Jos muuttujien sisältö ja funktioiden toimintaperiaate on eri, niin tällöin kyseessä on nimeämisristiriita.

JavaScriptissä muuttujat ovat näkyvyysalueeltaan (eng. scope) joko globaaleja tai lokaaleja. Lokaalit muuttujat näkyvät if-lauseiden tai funktion sisällä, jolloin ne ovat käytettävissä vain kyseisessä koodilohkossa. Koodilohkojen ulkopuolella olevat muuttujat ovat globaaleja ja ne ovat käytettävissä mistä tahansa kohtaa koodia. Globaalin

ja lokaalin muuttujan erottaa siitä, että se on määritelty hakasulkeiden sisään. Globaalien muuttujien käyttöä kannattaa välttää mahdollisuuksien mukaan kokonaan, koska tämä kasvattaa nimeämisristiriidan riskiä. (block 2018; JavaScript Scope n.d.)

Funktiot ja muuttujat on mahdollista suojata nimiristiriitojen varalta käärimällä ne itsensä ajavan funktion sisään (eng. self-executing function). Funktion sisällä olevat muuttujat ja funktiot ovat suojassa ulkopuolella olevilta saman nimisiltä muuttujilta ja funktioita, koska itsensä ajavan funktion sisällä on oma lokaali näkyvyysalueensa. Itsensä ajavan funktion toiminnastaan nähdä esimerkki kuviossa 6. Kyseistä suojausta tullaan hyödyntämään NPS-kyselyn toteutuksessa. (Herszak 2015.)

Punaisessa laatikossa asiakkaan sivuille on ladattu kuvitteellinen nps-kysely, joka on tallennetty survey-nimiseen objektiin. Sivuilla kuitenkin on jo valmiiksi survey-niminen muuttuja, joka liittyy aivan eri asianyhteyteen. Kyselyn lataaminen muuttaa tuota arvoa, vaikkei tämä olisi tarkoitus.

```
<script>
var survey = false;

var survey = {get : "npsKysely()"};
console.log(survey.get); // Tuloste -> npsKysely()

if(survey){ // Tulostaa sanan survey, vaikkei se ollut ohjelman alkuperäinen tarkoitus
    console.log('survey');
}
</script>
```

Vihreässä laatikossa oleva asiakkaan sivuille ladattu kuvitteellinen nps-kysely ei ylikirjoita sivun omia muuttujia, koska ne on kääritty itsensä ajavan funktion sisään, ja ovat siksi "piilossa" pääohjelmalta.

```
<script>
var survey = false;

(function(){
    var survey = {get : "npsKysely()"};
    console.log(survey.get); // Tuloste -> npsKysely()
})();


if(survey){ // Ei tulosta mitään
    console.log('survey');
}
</script>
```

Kuvio 6. Itsensä ajava funktio

4 Responsiivisuus

4.1 Responsiivisuus yleisesti

Hyvän tietoturvan ja integraation helppouden lisäksi web-sovelluskehityksessä painottuu responsiivisuus, joka vielä toistaiseksi on varsin huonosti tiedostettu ongelma (Stangarone 2015).

Responsiivisuus tarkoittaa sivustoa, jonka elementit ja ulkoasu mukautuvat käyttäjän laitteen resoluutioon. Tällä tarkoitetaan sulavaa siirtymistä työpöytäresoluutiosta mobiiliresoluutioon. 1024 pikselin vaakaresoluution ylittävien laitteiden resoluutiosta käytetään toisinaan nimitystä työpöytäresoluutio ja kaikki muut lasketaan kuuluvan mobiiliresoluution puolelle. Eri laitteiden näyttökoot ja resoluutiot ovat kuitenkin ajan mittaan kasvaneet ja tämän vuoksi todellisen työpöytä- ja mobiiliresoluution raja on joskus kiistanalainen. Uudemmaksi vaakaresoluution rajaksi on ehdotettu 1366 pikseliä. (Gregory 2018; Lardinois 2011; Popular Screen Resolutions: Designing for All 2018.)

Monien puhelimien resoluutio ylittää virallisen mobiiliresoluution rajan, mutta ne käyttävät usein skaalattua sivunäkymää eli ”viewporttia”. Tällä pyritään parantamaan sivujen luettavuutta, sillä varsinkin alle 5 tuuman teräväpiirtonäytöillä (teräväpiirto = pystyresoluutio yli 720 pikseliä, kun puhelinta pidetään sivuttain kädessä) sivujen luettavuus olisi pienen koon vuoksi huono. Esimerkiksi vuonna 2014 julkaistun iPhone 6 Plus -puhelimen resoluutio on 1080 x 1920 pikseliä, mutta viewport-resoluutio on vain 414 x 736. Tämä olisi noin 70 % pienimmästä työpöytäympäristön vaakaresoluutiosta (1024 pikseliä). (Popular Screen Resolutions: Designing for All 2018.)

Responsiivisuuden tarkoitus on lisätä sivun käyttömukavuutta ja vähentää tätä kautta sivuille tulevan liikenteen välittömän tai nopean poistumisen määrää. Vaikka web-kehitysprojektin lopputuote olisikin kohdennettu pelkästään työpöytäkäyttäjille, pitäisi responsiivisuuteen tästä huolimatta panostaa. Googlen hakukonealgoritmit mittaavat myös sivuston skaalautumista eri laitteilla ja tiputtavat huonosti skaalautuvat sivut alemmaksi. (Gregory 2018.)

4.2 Layout-tyypit

Erilaisia sommittelu- eli layout-tyyppejä on kolme erilaista: fixed, fluid ja adaptive. Paras mahdollinen responsiivisuus saavutetaan, kun yhdistetään kaksi viimeistä tekniikkaa eli fluid ja adaptive. (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016.)

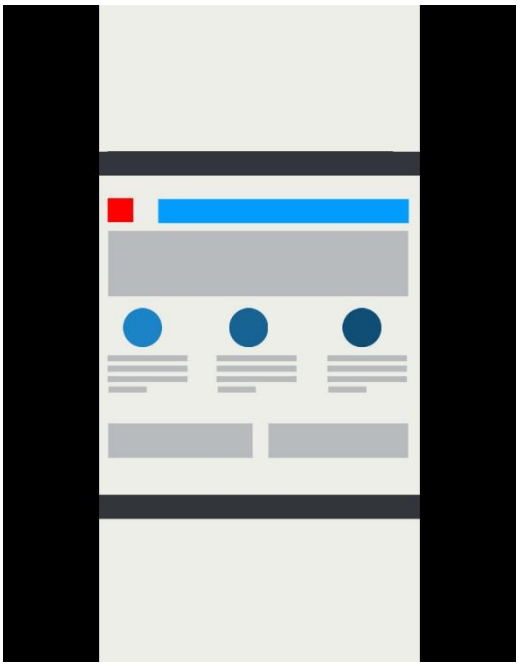
Fixed-layout

Fixed-layout on kolmesta sommittelutyypistä kaikkein yksinkertaisin (ks. kuvio 7). Tässä mallissa sivu ei reagoi laitteen resoluution muutokseen millään tavalla. Kohteet, jotka eivät mahdu selainikkunan sisään, rajautuvat kokonaan pois, koska sivun leveys on määritetty kiinteästi tiettyyn pikselikokoon. (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016.)

Kuvio 7. Alkuperäinen layout vasemmalla ja fixed-layout oikealla (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016, muokattu)

Fluid-layout

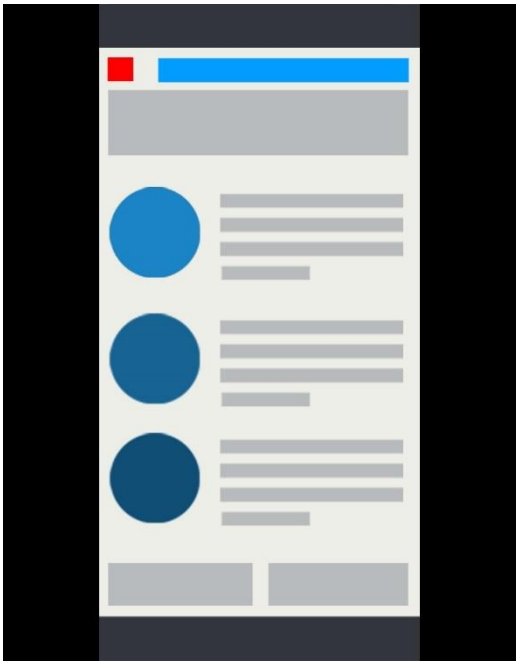
Fluid-layoutissa elementtien kokoa ei määritellä pikseleinä vaan prosentteina (ks. kuvio 8). Sivun koko muuttuu suhteessa käytettävissä olevaan tilaan nähden. Pienillä näytöillä sivun sisältö saattaa kuitenkin mennä liian pieneksi, jolloin tekstin ja kuvien erottaminen on hankalaa. Tämä sommittelutyyppi on kuitenkin parempi vaihtoehto sille, kuin että osa sivun sisällöstä rajautusi pois. (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016.)



Kuvio 8. Fluid-layout (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016, muokattu)

Adaptive-layout

Adaptive-layout eli mukautuva sommittelu sisältää useamman fixed-layoutin samalla sivulla (ks. kuvio 9). Käyttäjälle näytetään aina portaittain erilaista sisältöä sitä mukaa, kun käytettävän laitteen selainikkunan koko muuttuu. (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016.)



Kuvio 9. Adaptive-layout (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016, muokattu)

Responsive-layout

Verkkosivu on sommittelultaan responsiivinen, kun sen elementit pystyvät mukautumaan mahdollisimman moneen resoluutioon. Yhdistämällä fluid- ja adaptive-layout päästään parhaaseen mahdolliseen lopputulokseen. Fluid-mallin avulla sisältö ei koskaan rajaudu kuvan alkupuolelle ja adaptive-mallin avulla koko sivuston asettelua voidaan muuttaa tietyn vaakaresoluution alittuessa niin, että oleellinen sisältö ei pienene liikaa ja mahtuu näytölle. (What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care 2016.)

Mobiililaitteille, kuten tableteille ja älypuhelimille kohdennettu sisältö ei kuitenkaan aina toimi suoraan kaikissa tapauksissa. Näihin kuuluvat muun muassa puettaviksi laitteiksi luettavat älykellot. Tämä ei johdu riittämättömästä resoluutiosta, vaan esteenä on laitteiden pieni fyysinen koko. Esimerkkinä vuonna 2017 julkaistuissa Apple iPhone 8 -puhelimessa (4,7 tuumaa) ja Apple Watch Series 3 -älykellossa (1,65 tuumaa) ruudun kokoero on lähes kolminkertainen. (Apple iPhone 8 n.d; Apple Watch Series 3 n.d.)

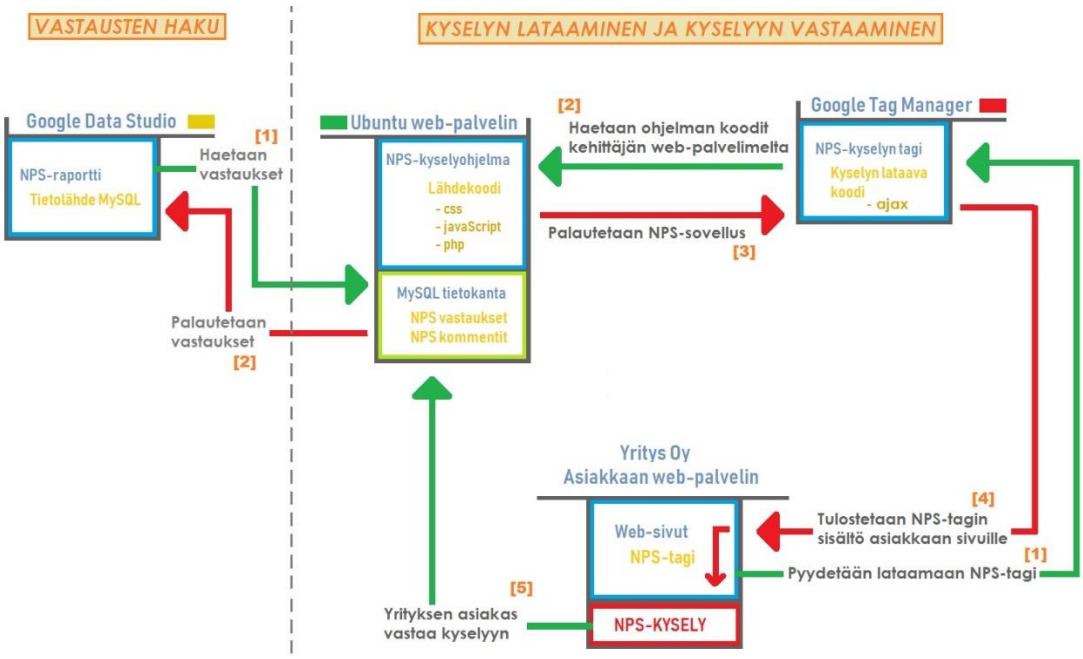
Oman näkemykseni mukaan puettavien älylaitteiden kohdalla on käytännöllisempää suunnitella näille kokonaan omaa personoitua sisältöä, kuin yrittää sovittaa työpöytäresoluution sovellusta ”postimerkkikokoon”. Tällöin kehitteillä oleva web-sovellus tulisi haarauttaa kahteen itsenäiseen versioon, joista toinen on suunnattu työpöytä- ja mobiililaitteille, ja toinen keskittyisi pelkästään puettaviin laitteisiin. Yleensä kehittäjän ja asiakkaan resurssit ovat rajalliset. Tämän vuoksi tuettavien versioiden määrän sanelee käytännössä lähes aina sovelluksen kohdeyleisöstä se enemmistö, joka vielä toistaiseksi suosii suurikokoisemmille laitteille kehitettyä mobiili- tai työpöytäkäyttöliittymää.

Tilanne saattaa kuitenkin tulevaisuudessa muuttua, sillä vuodesta 2014 vuoteen 2017 älykellojen myynti nousi viisitoistakertaiseksi 5 miljoonasta kellosta 75 miljoonaan kelloon. Älypuhelimiin verrattuna luvut ovat kuitenkin vielä toistaiseksi varsin pieniä, sillä maailmanlaajuisesti pelkästään vuoden 2017 suurinten puhelinvalmistajien myynti oli yhteensä noin 1,516 miljardia laitetta. Tästä voidaan päätellä, että tuona vuonna noin viisi sadasta puhelimen ostajasta (4.94 %) hankki myös älykellon. Vaikka puettavien älylaitteiden myynti onkin selvästi nousussa, rajoittuvat tässä opinnäytetyössä käytettävät tekniikat pelkästään perinteisiin työpöytä- ja mobiilisovelluksiin. (Sharmila 2018; Smartwatch unit sales worldwide from 2014 to 2018 n.d.)

5 NPS-kyselyn toteutus Analyticsin avulla

5.1 Toimintakaavion laatiminen

Google Analyticsin demoamisprojekti asiakaskyselyn toteutusalustana aloitettiin toimintakaaviota laatimisella (ks. kuvio 10), joka kuvaa niiden komponenttien ja tekniikoiden välisiä suhteita, joiden avulla NPS-kysely tullaan toteuttamaan. Jonkinlaisen vapaamuotoisen toimintakaavion mallintaminen olisi suotavaa minkä tahansa ohjelmointiprojektin yhteydessä, koska se hahmottaa ohjelman toimintaa kirjallista dokumentointia paremmin.



Kuvio 10. NPS-kyselyohjelman toimintakaavio

Toimintamalli on jaettu kahteen osaan, joista vasen puoli kuvaa kyselyn vastaustulosten hakemista, ja oikea puoli kyselyn lataamista yritysasiakkaan verkkosivuille. Kyselyn vastaukset haetaan MySQL-tietokannasta Data Studioon itseensä sisäänrakennetun logiikan avulla, joka on täysin itsenäinen kyselyohjelman toiminnasta.

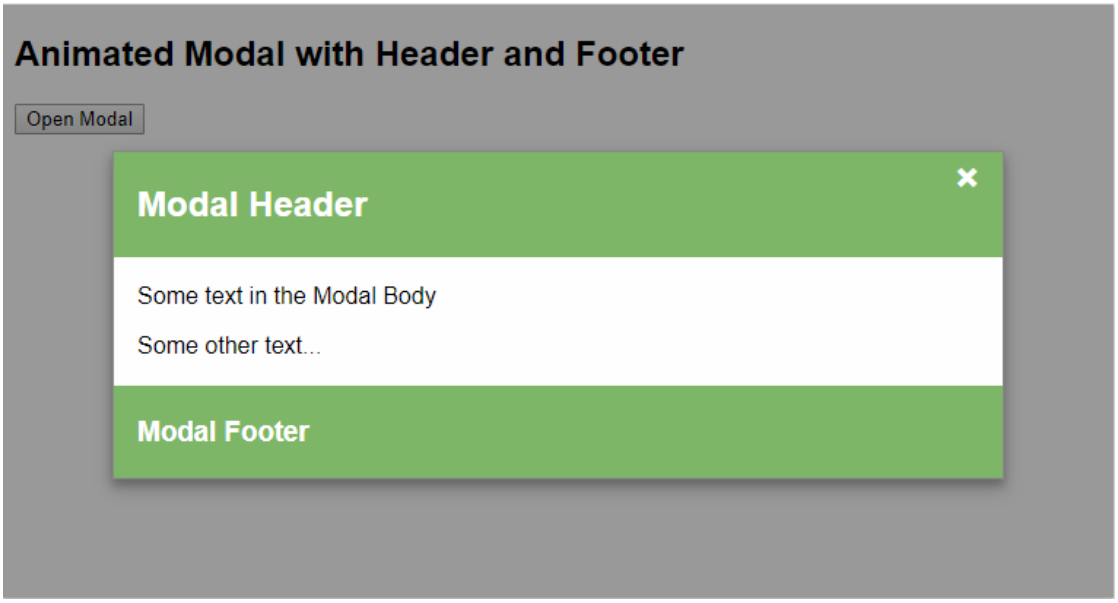
Nuolet osoittavan datan kulkusuunnan ohjelman eri osien välillä, joista vihreät kuvaavat käyttäjän syöttämää dataa, ja punaiset tämän perusteella tulevaa vastausta.

Datan kulkusuunnan lisäksi eri vaiheet kyselyn lataamiseksi on numeroitu siitä alkaen (oranssilla hakasulkeissa), kun käyttäjä avaa kyselyn sisältämän sivun.

5.2 Ohjelmointivaihe

5.2.1 Modal Box NPS-kyselyn mallipohjana

Toimintamallin laatimisen jälkeen lähdettiin suunnittelemaan NPS-kyselyn HTML-runkoa ja CSS-tyylejä. Kysely haluttiin sivun avautuessa aina päällimmäiseksi. Lisäksi sen täytyisi pysyä kiinteästi (eng. fixed-positioning) selainikkunan kohdassa, mihin se oli sivun avautuessa asetettu. Tämän avulla kysely saadaan pysymään aina näkyvillä, vaikka sivuilla olisi vierityspalkit. Mallipohjaksi (ks. kuvio 11) haettiin vapaan lähdekoodin W3Schools-sivuston CSS-muotoilukielellä toteutettu kyselyikkuna (modal box), jota muokkaamalla haettiin haluttua visuaalista tyyliä.



Kuvio 11. NPS-kyselyn mallipohja

Kyselyyn haluttiin lisäksi uusi avaamisefekti, koska sen oletuksena mukana tuleva avausanimaatio oli mahdollista toteuttaa yksinkertaisemmin. Avaava animaatio oltiin toteutettu CSS:n @keyframe-määrityksen avulla. Määrityksen sisälle luotavien avainkuvien (eng. keyframes) avulla voidaan HTML-elementtiin animoida vaihteleva muo-

don, koon ja sijainnin muutos halutulle ajanjaksolle (ks. kuvio 12). (CSS Animations. n.d.)

```
<style>
/* HTML ELEMENTTI */
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: animation_changeColor;
    animation-duration: 1s;
}

/* LISÄTTÄVÄ ANIMAATIO */
@keyframes animation_changeColor {
    from {background-color: red;}
    to {background-color: yellow;}
}
</style>
```

Kuvio 12. Kappaleen värin muuttaminen CSS-animaation avulla

Avainkuvat päätettiin korvata siirtymillä eli transitioneilla. Transitionien perusidea on sama kuin CSS-animaatiossa, joka on hidastaa HTML-elementin muutos tilasta toiseen sen sijaan, että muutos tapahtuisi välittömästi, mikä on yleensä selaimen oletusasetus. Transitionit ovat CSS-animaatioita yksinkertaisempia, sillä niissä määritetään vain muutoksen kesto ja kiihtyvyys kahden pisteen välillä.

CSS-animaatiossa on mahdollista määrittää useampi piste ja näiden välinen muutos. Tämä olisi kuitenkin ollut tarpeetonta kyselyikkunan suoraviivaisessa liikkeessä. Esimerkki transitioneiden käytöstä nähdään kuviossa 13. (CSS Animations. n.d; Ivanov 2015.)

```
/* Active selection indicator. */
#scoreBox-scoreSelected {
    position: absolute;
    display: none;
    width: 63.4px;
    height: 40px;
    left: 10px;
    top: 10px;
    background-color:#3f3f3f;
    -webkit-transition: left 1.5s cubic-bezier(0.165, 0.84, 0.44, 1);
    -moz-transition: left 1.5s cubic-bezier(0.165, 0.84, 0.44, 1);
    -ms-transition: left 1.5s cubic-bezier(0.165, 0.84, 0.44, 1);
    -o-transition: left 1.5s cubic-bezier(0.165, 0.84, 0.44, 1);
    transition: left 1.5s cubic-bezier(0.165, 0.84, 0.44, 1);
    -webkit-transform: scale(1.2);
    -moz-transform: scale(1.2);
    -o-transform: scale(1.2);
    -ms-transform: scale(1.2);
    transform: scale(1.2);
}
```

Animoitava arvo left (ts. kun elementti liikuu tai sitä liikutetaan x-suunnassa.)

Transition eli siirtymä. Kertoo, minkä arvon muutos halutaan animoida, sekä kuinka "sulava" tämä on.

Kuvio 13. HTML-elementin animointi tansition-arvon avulla

5.2.2 Kyselyn vastauspainikkeet

Kun kyselyn mallipohjana toimiva ”modal box” oli visuaalisesti sekä avausanimaationsa osalta valmis, lähdettiin tämän päälle tekemään käyttöliittymää eli lomaketta, joka mahdollistaa kyselyyn vastaamisen. Tätä varten kyselyikkunaan lisätiin HTML-vastauspainikkeet (eng. radio button input), sekä lisäpalautteen antamiseksi tekstilaatikko (eng. textarea), jotka asetettiin form-tagien sisään.

Lomakkeen valintapainikkeet koostuvat label-kentistä, sekä label-kentän sisältämästä painikkeesta (eng. input). Valintanappulan arvo sekä tämän yksilöivä nimi säilöttiin nappulan ”name” ja ”value” -attribuutteihin (ks. kuvio 14).

```
<label id="scoreBox-score-4" class="scoreBox-score detractors-color"><input type="radio" name="score" value="4">
<span class="score-top noselect">4</span></label>
```

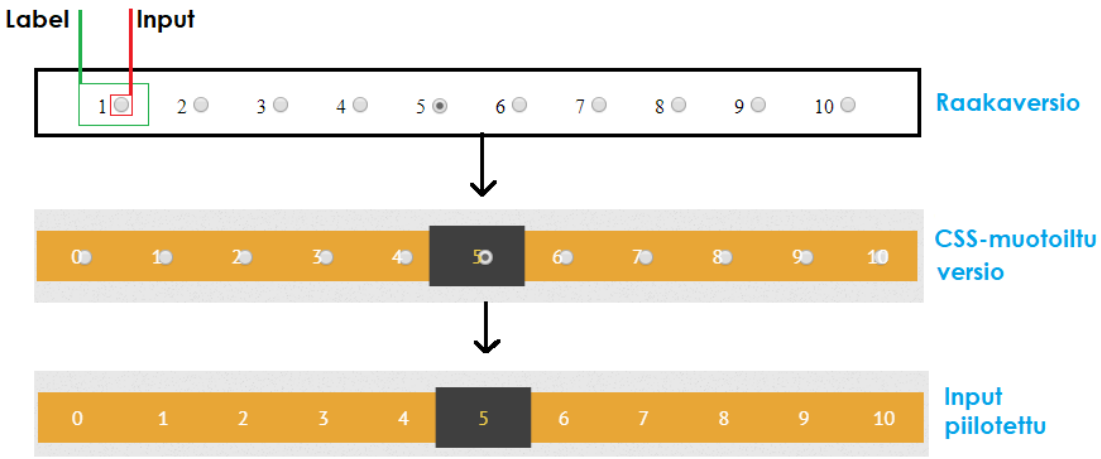
Vastaustulokset käsittelevä ohjelma osaa hakea annetun arvosanan tähän viittaavalla nimellä (score).

Lähetettävä arvosana (4).

Kuvio 14. Arvosanan hakeminen nappulasta

Lopuksi arvosanan valintanappuloille tehtiin CSS-tyylimäärittelyt. Input-tagia ei kuitenkaan ole mahdollista muokata CSS-tyyleillä, sillä niiden ulkoasu määräytyy suoraan tämän type-attribuutin perusteella (esim. radiopainike on pyöreä). Ratkaisuna

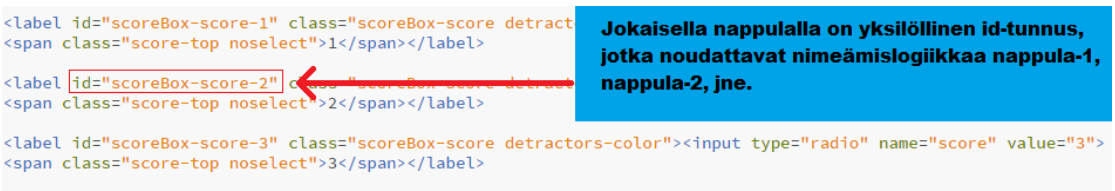
hyödynnettiin label-tagin muokattavuutta, jonka sisällä valintanappula on ja joka ei itsessään vaikuta sivuston ulkoasuun. Sen ympärillä on valinta-alue, jota klikkaamalla voidaan valita sisempänä oleva nappula aktiiviseksi. Tämän jälkeen alkuperäinen nappula piilotettiin (ks. kuvio 15). (HTML <label> Tag. n.d.)



Kuvio 15. Arvosanan valintanappulat NPS-kyselyssä

Kun arvosanan valintalaatikko oli valmis tyylimäärittelyjen osalta, haluttiin tähänkin tuoda animaatiota NPS-kyselyn avausefektin rinnalle. Tässä vaiheessa, kun käyttäjä klikkasi mitä tahansa arvosanaa, osoitettiin aktiivisena olevaa valintaa tähän ilmestyvällä mustalla laatikolla.

Musta valintailmaisin haluttiin siirtää aina liu'uttamalla seuraavaksi klikattavan label-valinta-alueen kohdalle sen sijaan, että valinta siirtyisi suoraan klikatun arvon päälle. Tämä oli mahdollista antamalle jokaiselle labelille (nappulalle) yksilöllinen, juokse-vassa numerojärjestyksessä annettu id-tunnus (ks. kuvio 16).



Kuvio 16. Arvosanan sisältävien nappuloiden yksilöllinen id-tunniste

Tämän jälkeen kirjoitettiin getScoreIds-funktio, jota kutsumalla voitiin palauttaa for-looppia ja juoksevan id-tunnisteen loppuosaa hyödyntäen kaikkien nappuloiden id-tunnisteet peräkkäin ja pilkulla erotettuna. Funktion palauttamat tunnisteet vietiin argumenttina JavaScript-kieleen sisäänrakennetulle querySelectorAll-metodille. Metodille argumenttina viety lista CSS-valitsimista (id- tai luokkavalitsin) palauttaa kaikki ne HTML-elementit, joista kyseinen valitsin löytyy. Id-tunnisteen perusteella haetut nappulat säilöttiin ScoreElements-nimiseen muuttujaan.

Seuraavaksi luotiin uusi for-looppi, jonka sisällä kutsuttiin selectScore-nimistä funktiota. For-loopin ja funktion avulla käytiin läpi jokainen ScoreElements-muuttujan sisältämä nappula, jonka yhteydessä kuhunkin asetettiin erikseen tapahtumakuuntelija (eng. eventlistener). Click-tyyppisen tapahtumakuuntelijan avulla klikattu nappula laitettiin palauttamaan oma sijaintinsa x- ja y-suunnassa aina, kun käyttäjä klikkaa kyseistä nappulaa.

Klikatusta nappulasta saadut koordinaatit vietiin sillä hetkellä aktiivisena olevalle valintailmaisimelle, jonka nykyiset koordinaatit korvattiin klikatun nappulan koordinaateilla. Valintailmaisin saatiin siirtymään animaation kautta uuteen sijaintiin transitionien avulla.

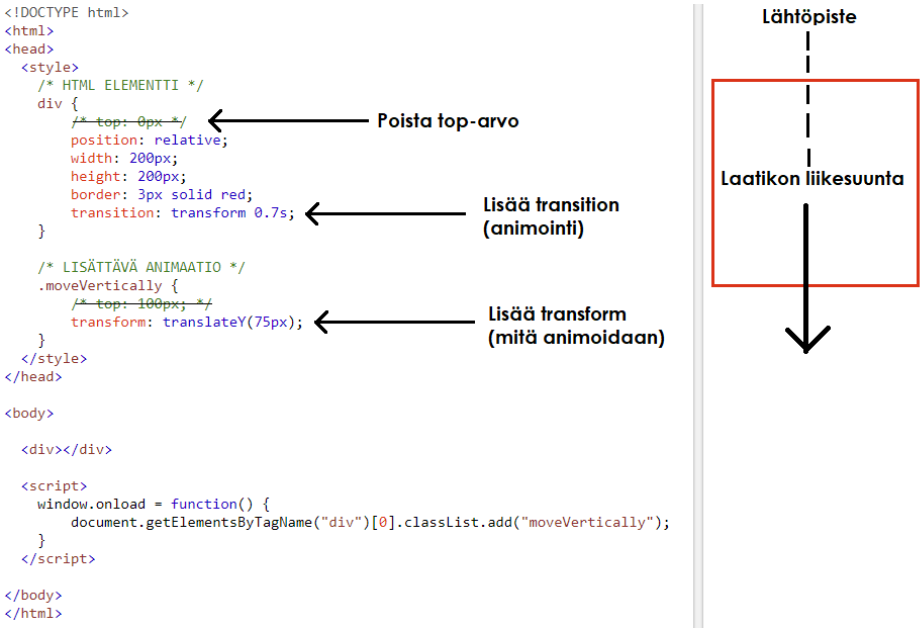
5.2.3 Animaatioiden optimointi

Valmiissa kyselyikkunan mallipohjassa oli hyödynnetty HTML-elementtien sisältämää top-arvoa, joka kertoo, kuinka kaukana elementin yläreuna sijaitsee emoelementin (eng. parent element) yläreunasta y-suunnassa. Top-arvon avulla HTML-elementti voidaan animoida liikkumaan pystysuunnassa haluttuun kohtaan. (CSS top Property n.d.)

Tämä ei kuitenkaan osoittautunut parhaaksi ratkaisuksi, sillä top-arvon animointi saattoi selaimesta riippuen näyttää epäsulavalta tai nykivältä. Selaimen on laskettava animoitavan kappaleen koko ja sijainti jokaista avainkuvaa kohden uudelleen (eng. reflow), sekä piirrettävä niiden välissä muuttuva sivu tietokoneen näytölle (eng. repaint). Koko sivun uudelleen piirtäminen tekee animaation toistamisesta raskasta. (Chikuyonok 2016.)

Ratkaisuna oli käyttää top-arvon ja CSS-animaation sijasta CSS:n transform-ominaisuudelle annettua translate-arvoa, jonka avulla HTML-elementtiä voidaan myös liikuttaa ja joka tapahtuu pystysuunnassa antamalla sen arvoksi translateY. Kuviosta 17 näkyy top-arvon animointi transform-ominaisuuden avulla. Transformin liikkeen muutos saadaan animoitua erillisen transitionin avulla, joka kertoo, kauan kappaleessa tapahtuva muutos kestää. CSS-animaatiossa tämä tehtäisiin ”animation-duration” -määrityksellä.

Transform saa selaimen siirtämään liikutettavan kappaleen ja verkkosivun kahteen eri kerrokseen (eng. compositing layers), jonka jälkeen ne tallennetaan välimuistiin. Enää koko sivua ei tarvitse piirtää uudelleen avainkuvien välissä, vaan ne ladataan valmiiksi piirrettynä muistista. Tämän avulla niitä voidaan vain liikuttaa toisistaan poispäin piirtämättä kappaleita aina uudelleen (Chikuyonok 2016.)




```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* HTML ELEMENTTI */
    div {
        /* top: 0px */
        position: relative;
        width: 200px;
        height: 200px;
        border: 3px solid red;
        transition: transform 0.7s;
    }

    /* LISÄTTÄVÄ ANIMAATIO */
    .moveVertically {
        /* top: 100px; */
        transform: translateY(75px);
    }
  </style>
</head>

<body>

  <div></div>

  <script>
    window.onload = function() {
        document.getElementsByTagName("div")[0].classList.add("moveVertically");
    }
  </script>

</body>
</html>
```

Kuvio 17. CSS-animaation optimointi

Chikuyonokin (2016) mukaan kyseistä tekniikkaa käyttäessä voi sivu kuitenkin joskus välkkyä ja kaatuilla. Tätä tapahtuu erityisesti silloin, kun liikkuva kohde on usean liikkumattoman kohteen alla. Tällöin jokainen liikkumatonkin kohde on siirrettävä ja tallennettava muistiin omiksi kerroksikseen (eng. implicit compositing), joka puolestaan saattaa kasvattaa muistin käyttöä liikaa. Tässä projektissa NPS-laatikko oli kui-

tenkin tarpeeksi yksinkertainen, jotta kyseisestä optimointikeinosta oli enemmän hyötyä kuin haittaa. (Chikuyonok 2016.)

Aina transform-ominaisuuden avulla suoritettu optimointi ei kuitenkaan ole välttämätöntä, jos animoitava kappale on pieni ja liikkeeltään yksinkertainen, eikä sen päälle ole staattisia objekteja. Aiemmin left-arvoa oli käytetty NPS-kyselyn arvosanan valintailmaisimen animoinnissa, vaikka tämä olisi voitu korvata myös translateX-arvolla. Tällöin animaatiota kannattaa kuitenkin testata useammalla laitteella ja varmistaa, ettei se nyi tai kaada selainta.

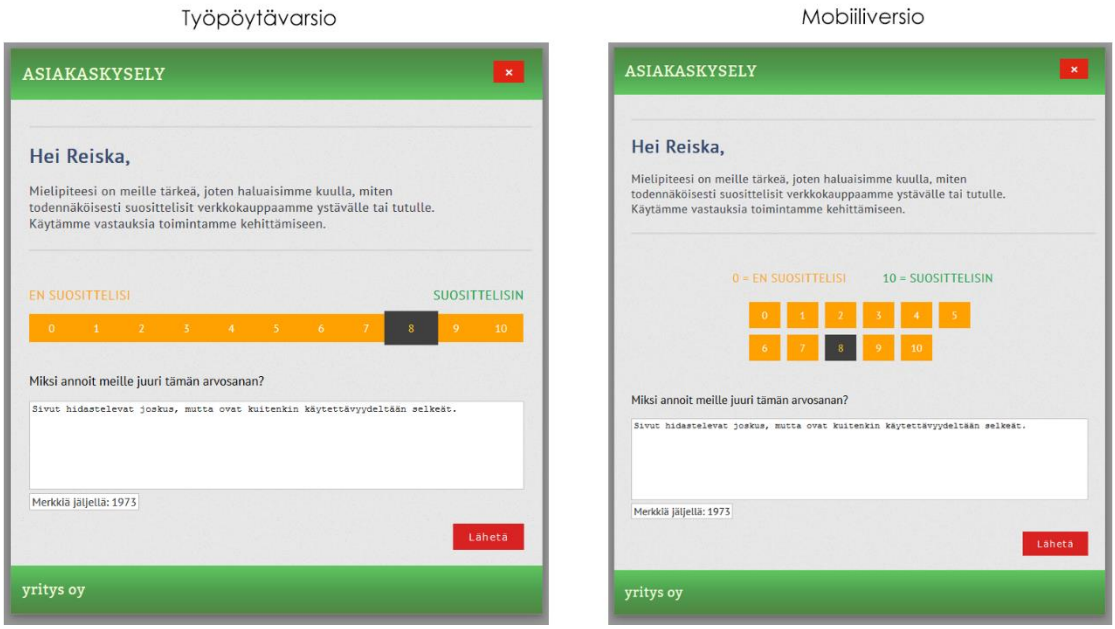
5.2.4 Responsiivisuus

NPS-kyselyn CSS-tyylien määrittelyn ja animaatioiden lisäämisen jälkeen lähdettiin toteuttamaan responsiivisuutta, hyödyntäen fluid- ja adaptive-layout -suunnittelua. Adaptive-layout toteutettiin CSS-muotoilukielen omilla media query -määrittelyillä, joiden avulla kyselystä luotiin yksi versio työpöytä- ja mobiilikäyttöön. Media queryt on tarkoitettu sivun sisällön ja ulkoasun muokkaamisen jonkin tietyn ehdon perusteella. (Responsive Web Design - Media Queries n.d.)

Kyselyn mobiilikäyttöä pyrittiin helpottamaan arvosanan valinnan osalta lisäämällä valintanappuloiden ympärille tyhjää tilaa eli marginia silloin, kun laitteen vaakaresoluutio on alle 900 pikseliä. Tämä helpottaa arvosanan antamista pienellä kosketusnäytöllä, ja erityisesti jos kyseessä on isosorminen henkilö. Alla oleva kuvio 18 esittää media queryn toimintaa käytännössä. Kuviosta 19 voidaan nähdä, miltä lopullisen NPS-kyselyn työpöytä- ja mobiiliversiot näyttävät.

```
@media only screen and (max-width: 900px) {
  .scorebox-score {
    width: 18px;
    margin: 5px;
  }
}
```

Kuvio 18. Media Query



Kuvio 19. NPS-kyselyn työpöytä- ja mobiiliversiot vierekkäin

Jotta lopputulosta voitiin kutsua täysin responsiiviseksi, otettiin alle 830 pikselin vaakaresoluution alittuessa käyttöön fluid-layout. Tämä tehtiin määrittämällä kyselyn leveydeksi 90% koko käytössä olevan selainikkunan leveydestä. Näin kyselylaatikon reunoille jäi 10% verran tyhjää tilaa ja se kapenisi samalla kun selainikkuna pienenee.

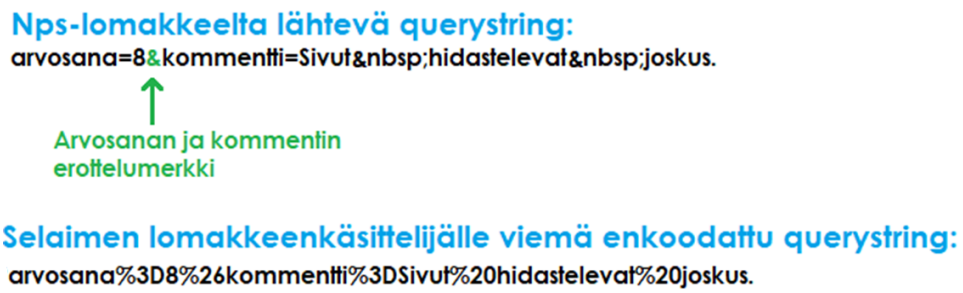
Kaikkein pienimmille laitteille määritettiin 600 pikselin kynnysarvo. Tämän jälkeen kyselyn kaventaminen ei enää ollut mahdollista, koska tekstin luettavuus olisi kärsinyt liian kapean tilan vuoksi. Ratkaisuna oli määrittää leveys uudelleen kiinteästi 600 pikseliin prosenttien sijaan. Lisäksi otettiin käyttöön JavaScriptillä kirjoitettu scale-funktio, jonka avulla kyselylaatikkoa alettiin kutistaa kaventamisen sijaan aina, kun selainikkunan koko on pysty- tai vaakatasossa vähemmän kuin kyselyikkunan leveys tai korkeus.

5.2.5 Kyselyvastausten lähetys

Kun vastaaja täyttää NPS-kyselyn HTML-palautelomakkeen, lähettää selain lomakkeeseen täytetyt tiedot sen action-attribuutissa määritellyn lomakkeenkäsittelijän osoitteeseen. Käsittelijä saa POST-metodin kautta lähetetyt tiedot querystring-merkkijonona, johon lomakkeelta tulevat tiedot on säilötty avain-arvopareina. (Rouse 2006.)

Lomakkeen lähetyksen jälkeen selain hakee querystring-merkkisarjan lomakkeenkäsittelijälle lähetetyn HTTP-pyynnön HTTP-otsakkeiden (eng. HTTP headers) perästä. Avain-arvot erotetaan toisistaan ”=”-merkillä, ja useampi peräkkäinen avain-arvopari erotetaan toisistaan ”&”-merkillä. (Rouse 2006; Guzel 2009.)

Kuvio 20 hahmottaa NPS-lomakkeen käsittelijälle lähetettävän querystringin toimintaa, jossa palvelun arvosanaksi on annettu 8 ja viestiksi ”Sivut hidastelevat joskus”. Kannattaa huomata, että HTML-kielessä tietyt erikoismerkit voidaan kirjoittaa vaihtoehtoisessa esitysmuodossa, joita kutsutaan englanniksi nimellä ”HTML Entities”. Esimerkiksi lomakkeen kommenttikentän välilyönnit voidaan käsitellä merkkisarjana ” ”, jotka HTML-tulkki ymmärtää välilyönteinä. (HTML Entities n.d.)



Kuvio 20. Querystringin enkoodaus

Kun querystring on muodostettu, täytyy lähetettävä merkkijono vielä enkoodata (eng. URL Encoding). Koodauksen tarkoituksena on muuntaa kaikki erikoismerkit sellaisiksi merkkijonoiksi, joilla ei ole mitään erikoismerkitystä. Edeltävästä kuviosta 20 nähtiin, että esimerkiksi välilyöntimerkit ( ) on enkoodattava muotoon ”%20”, koska muutoin selain luulisi sen sisältämää ”&”-merkkiä querystringin erottelumerkiksi. (HTML URL Encoding Reference n.d.)

Querystringin lähettämisen jälkeen selain ohjaa (eng. redirect) lomakkeen täyttäjän samalle sivulle kuin missä lomakkeenkäsittelijä sijaitsee, eli aiemman kuvan esimerkissä sivu.fi/kasittelija.php. Uudelleenohjaus haluttiin estää, koska tässä tapauksessa kyse oli mille tahansa sivulle upotettavasta asiakaskyselystä. Olisi epäjohdonmukaista ohjata lomakkeen täyttäjä sivuilta pois, jos tämä olisi esimerkiksi tekemässä tilausta verkkokaupassa, ja kysely halutaan näyttää kesken ostotapahtuman.

Lomakkeen uudelleenohjauksen esto toteutettiin jQuery-nimisen JavaScript-kirjaston avulla, jonka tarkoituksena oli lyhentää kirjoitettavan koodin määrää, sekä tarjota tähän jo valmiiksi rakennettua selainyhteensopivuutta. Eston toiminnan varmistaminen kaikilla selaimilla olisi ollut työlästä, jos tämän olisi joutunut tekemään kokonaan itse. Kuviossa 21 nähdään ote lomakkeen lähetyksen estävästä jQuery-koodista.

```
$form.submit(function () {
    $.post($(this).attr('action'), $(this).serialize(), 'json');
    var cookie = cookies("nps");
    if (cookie === "") { cookies_Tridea.set("nps", true, 30); }
    return false;
});
```

Kuvio 21. Lomakkeen lähetyksen esto jQueryn avulla

Ensimmäisellä rivillä nähdään ”$form”-nimisen muuttujan sisään tallennettu lomakeobjekti, joka luotiin välittämällä jQuery-objektille lomakkeen id-tunniste. Tämän jälkeen voitiin kutsua lomakeobjektin jQuery-objektin sisäisen logiikan kautta perimää submit-metodia, jonka avulla voidaan ajaa haluttu JavaScript-koodi lomaketta lähettäessä.

Lähetyksen yhteydessä ajettiin anonyymifunktio, joka kutsuu jQuery-kirjaston omaa post-metodia, jota tarvitaan varsinaiseen lomakkeen lähetykseen. Metodille välitettiin lähetettävän lomakkeen lomakkeenkäsittelijän osoite (this-objekti) sekä toisena argumenttina lomakkeen sisältö querystring-muodossa enkoodattuna.

Viimeisellä rivillä palautettava false-arvo keskeyttää lomakkeen lähetyksen, koska JavaScript-tulkki hyppää tuolloin koko submit-metodista pois. (jQuery serialize Method n.d.)

5.3 Lomakkeenkäsittelijä

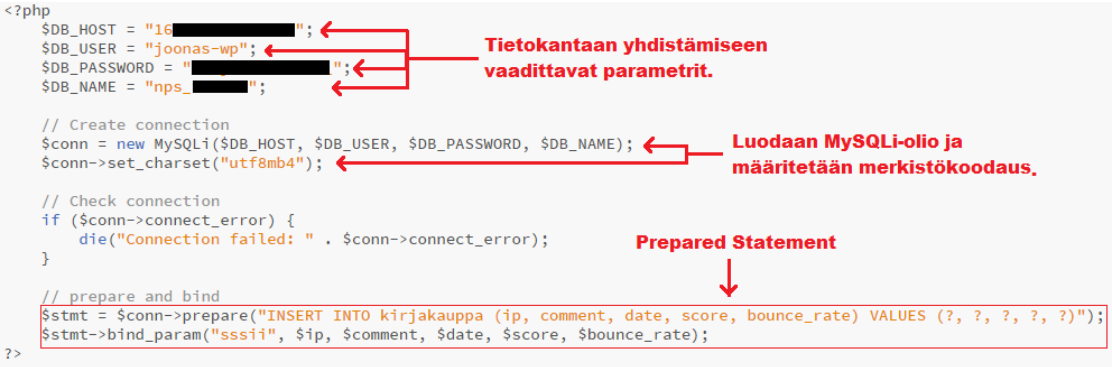
5.3.1 Toimintaperiaate ja käyttötarkoitus

NPS-kyselyn vastaukset vietiin PHP-lomakkeenkäsittelijälle, jossa varsinainen kyselytulosten vastaanotto ja tallentaminen tapahtui MySQL-tietokantaan. Käsittelijä pää-

tettiin jakaa kahdeksi erilliseksi tiedostoksi: ”getFeedback.php” ja ”connect_db.php”. Connect_db-ohjelma suorittaa MySQL-tietokantaan yhdistämisen, jonka jälkeen getFeedback tallentaa vastauksen tätä varten luotuun NPS-nimiseen tietokantaan kirjakauppa-tauluun. Tiedostojen sijainti palvelimella voidaan nähdä liitteessä 2.

Tietokantayhteys luotiin hyödyntäen MySQLi-luokkaa, josta luoduilla olioilla on mahdollista tehdä hakuja ja kutsuja MySQL-tietokantoihin. Oliolle syötettiin argumentteina palvelimen osoite sekä käyttäjätunnus ja tietokannan nimi. Tämän jälkeen kutsuttiin set_charset-metodia, jonka avulla annetut palautteet voidaan tallentaa UTF8-merkistökoodattuna. (PHP 5 MySQLi Functions n.d.)

On kuitenkin hyvä tietää, että MySQL:än tapauksessa oikean UTF8-merkistökoodauksen tyyppinimi on historiallisista syistä ”UTF8MB4” eikä loogisesti ”UTF8”. Jotkin erikoismerkit, kuten hymiöt, eivät näy oikein väärää UTF-merkistökoodausta käytettäessä. Tämä täytyy ottaa huomioon myös tietokantaa luodessa ja asettaa sen tyypiksi UTF8MB4. Kuviosta 22 voidaan nähdä käytännön esimerkkinä, miten tietokantaan yhdistäminen ja käytettävän merkistökoodauksen määritys toteutettiin. (Hooper 2016.)



Kuvio 22. Tietokantaan yhdistäminen ja merkistökoodauksen määrittäminen

5.3.2 Lomakkeenkäsittelijän suojaaminen SQL-injektiolta

Lähes kaikki verkossa olevat käyttäjän täytettävissä olevat suojaamattomat lomakkeet ovat alttiita SQL-injektioille, ja tämän vuoksi se on yksi yleisimmistä verkon hakkerointikeinoista. Injektio tapahtuu antamalla SQL-kielisiä lausekkeita verkkosivuilla

oleviin syötekenttiin, joilla on lomakkeen arvot käsittelevässä ohjelmassa jonkin erikoismerkitys. (SQL Injection n.d.)

Pahimmassa tapauksessa hakkeri voi saada käsiinsä esimerkiksi listan kaikkien tietokantaan lisättyjen käyttäjien sähköpostiosoitteista ja salasanoista (yleensä kuitenkin salatussa muodossa), tai poistaa halutessaan koko sähköpostisarakkeen. Edellinen on kuitenkin yleensä mahdollista vain, jos lomakkeenkäsittelijä on ohjelmoitu oletuksena suorittamaan select-lausekkeita, jonka avulla voidaan hakea valittujen sarakkeiden tietyt rivit. Oikein käytettynä select-lausetta voidaan hyödyntää myös sarakkeiden poistamiseen, koska useat tietokannat mahdollistavat käskyjen ketjuttamisen. Esimerkiksi ”SELECT * FROM Käyttäjät; DROP TABLE Ylläpitäjät” hakisi kaikki käyttäjät, mutta poistaisi samalla ylläpitäjät-nimisen taulun. (Naik 2012; SQL Injection n.d.)

NPS-kyselyn lomakkeenkäsittelijässä ei kuitenkaan ollut käytössä select, vaan insert-lause, jolla tietokantaan voidaan tallentaa tietoa. Insert-lausekkeiden avulla toteutetut hyökkäykset ovat select-lauseisiin nähden harvinaisempia ja ne toimivat vain, jos käyttäjä saa minkäänlaista näkyvää palautetta insert-lausekkeen lopputuloksesta. (Naik 2012.)

Taulukosta 1 nähdään miten insert-lause mahdollistaa SQL-injektion suorittamisen. Esimerkissä käyttäjän on mahdollista antaa arvosana jostain palvelusta sekä valinnainen kommentti ja oma nimensä. Tämän jälkeen syötetyt arvot tallennetaan tietokantaan ja käyttäjälle näytetään yhteenvetosivu syöttämistään arvoista. Insert-lausetta voidaan kuitenkin muokata niin, että yhteenvetosivulle tulostuu palautteen antajan nimen sijaan palvelimen käytössä olevan MySQL-tietokantaohjelmiston versionumero, jota voidaan käyttää mahdollisten haavoittuvuuksien etsimiseen.

Taulukko 1. SQL-injektio (Naik 2012, muokattu)

Käyttäjäsyöte	SQL-komento	Tuloste
HTML-lomake: Normaali syöte		
Arvosana: 1 **Kommentti:** hei! **Nimi:** Matti	**INSERT INTO** nps (arvosana, kommentti, nimi) **VALUES** (‘1’,’hei!’,’Matti’);	Hei, Matti! Annoit arvosanaksi ”1” ja kommentiksi ”hei!”.
HTML-lomake: SQL-injektio		
Arvosana: testi’, (select version()), ’testi2’) -- - **Kommentti:** abc **Nimi:** abc	**INSERT INTO** nps (arvosana, kommentti, nimi) **VALUES** (’testi’, (select version()), ’testi2’) -- -,’abc’,’abc’);	Hei, 5.6.34-Ubuntu14.04! Annoit arvosanaksi ”testi” ja kommentiksi ”testi2”.

Koska NPS-kyselyn lomakkeenkäsittelijä ei anna palautetta käyttäjän syöttämistä arvoista eikä se sisällä select-lauseita, on mahdollisen SQL-injektion todennäköisyys normaalia pienempi. Haavoittuvuus on kuitenkin mahdollista estää helposti, eikä tästä ole minkäänlaista haittaa. Koska injektion riskiä on sataprosenttisella varmuudella hankala ennustaa, olisi kaikki verkkolomakkeet suositeltavaa suojata injektiota vastaan.

Yksi suosittu ja opinnäytetyössäkin käytetty tapa estää SQL-injektio on prepared statement. Prepared statementin ideana on lähettää pelkkä SQL-kyselypohja tietokantaan ilman välitettäviä parametreja. Kyselypohjassa parametrit on korvattu ”?”-merkeillä, josta nähtiin insert-tyyppinen esimerkki aiemmin esitetyssä kuviossa 22.

SQL-kyselypohjan lähettämisen jälkeen web-palvelin kääntää kyselyn konekieliseksi koodiksi, jonka jälkeen sille suoritetaan kyselyoptimointi, jonka tarkoituksena on nopeuttaa tietokantaan suoritettuja lisäis-, päivitys- ja hakuoperaatioita. Lopuksi optimoitu SQL-kysely säilötään, mutta kuitenkaan suorittamatta sitä. Säilöttyä kyselypohjaa voidaan myöhemmin käyttää kutsumalla sitä bind_param-metodin avulla, jonka yhteydessä tapahtuu varsinainen parametrien syöttö ja SQL-kyselyn suoritus. (PHP Prepared Statements n.d.)

Prepared statementin suojausmekanismi perustuu siihen, että erillinen käyttäjän syöttämät arvot lähettävä protokolla pakottaa palvelimen tulkitsemaan ne osana syötettyjä parametreja, eikä osana SQL-käskyä. (How does a PreparedStatement avoid or prevent SQL injection 2017; PHP Prepared Statements n.d.)

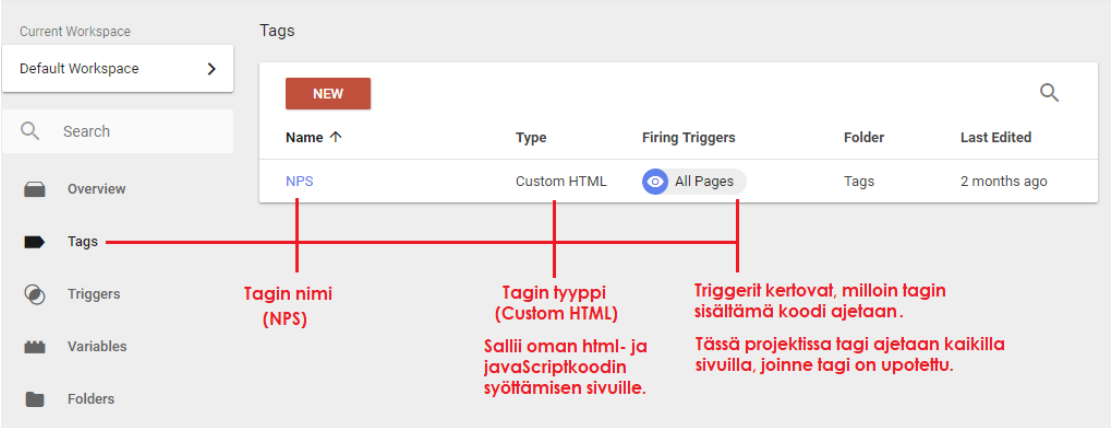
Prepared statementteja kannattaa käyttää myös silloin, kun niistä ei koeta olevan tietoturvan kannalta hyötyä. Propared statement säästää kaistaa, koska SQL-kyselypohja täytyy lähettää tietokantapalvelimelle vain kerran, jonka jälkeen voidaan lähettää vain parametrien arvoja. Normaalissa SQL-kyselyissä kyselyä ei tallenneta muistiin, joten koko kysely on lähetettävä parametreineen joka kertaa uudestaan. Lisäksi kyselyn vastaukset saadaan nopeammin, koska kyselyn optimointi ja kääntö konekielelle tehdään myös vain kertaalleen. (PHP Prepared Statements n.d.)

5.4 NPS-kyselyn vieminen Tag Manageriin

NPS-kyselyohjelman valmistuttua lähdettiin sen lähdekoodia siirtämään Tag Manageriin. Tag Manageriin yhteydessä olevia upotettavia tageja ohjataan säiliöiden (eng. container) kautta, joille jokaiselle annetaan yksilöllinen säiliötunniste (eng. container id). Jokainen säiliö on ennen käyttöönottoa luotava, eikä niiden määrää ole rajoitettu.

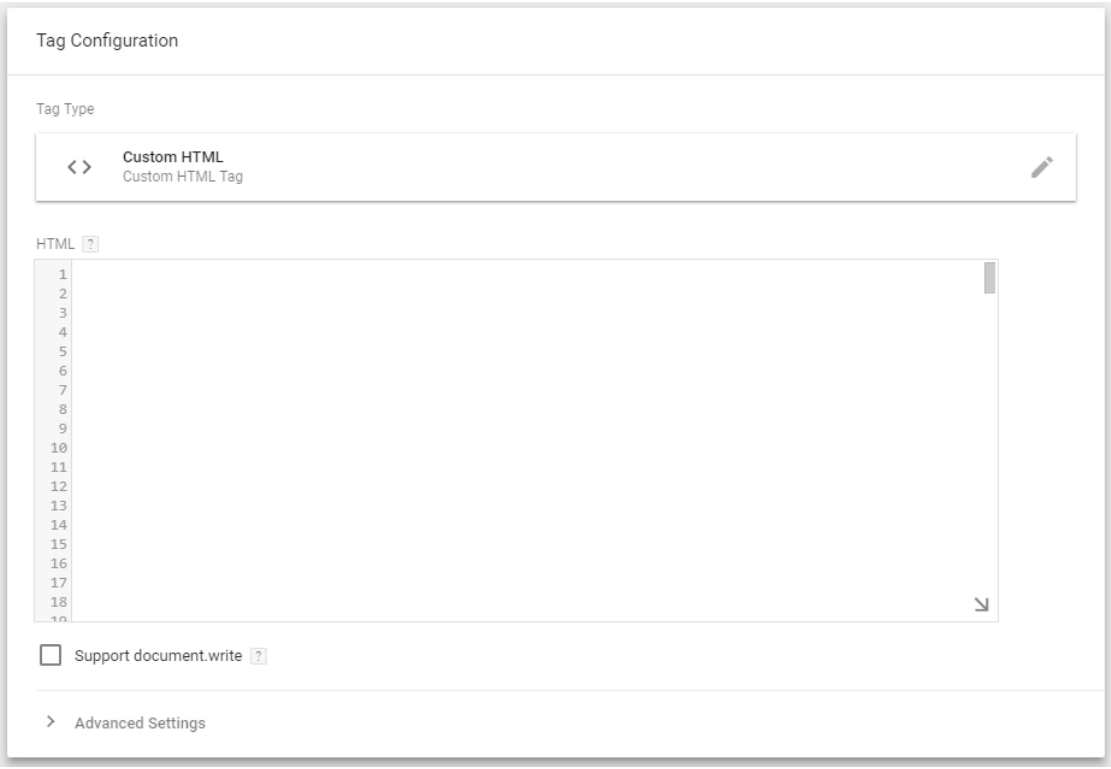
Normaalisti säiliöitä luodaan yksi per sivusto, mutta halutessa yhtä säiliötä voidaan käyttää monella eri sivustolla. Säiliön luomisen jälkeen saatiin tagi, jonka sisältö voidaan ladata kaikilla sivuilla, jonne tagi on asetettu. Liitteessä 3 havainnollistetaan, miltä säiliö sekä sisällön lataava tagi näyttävät. Tagin asentamiseksi tarvitaan säiliötunniste sekä Googlen antamien script- ja noscript-koodien lisäämiset HTML-rungon head- ja body-tagien sisään.

Seuraavaksi määriteltiin, mitä tagin lataamisen jälkeen tapahtuu. Tämä toteutettiin luomalla ”tagisäiliö” edellä luodun ”pääsäiliön” sisälle. Tagin luontivaiheessa kysytään tagin nimeä, tyyppiä sekä milloin tagin sisältö (HTML- ja JavaScript-koodi) ladataan. Kuvio 23 esittää tagisäiliön luomiseen vaadittavia parametreja.



Kuvio 23. Tagin luominen Tag Managerissa

Luotua tagia päästiin muokkaamaan klikkaamalla sen nimeä. Tämän jälkeen avautui normaalilta HTML-editorilta näyttävä kuvion 24 mukainen tyhjä sivu. Kyselyikkunan HTML-pohjan koko rakenne kopioitiin sellaisenaan tagieditoriin, minkä jälkeen tagia yritettiin ladata kirjakauppa.com-sivustolla.



Kuvio 24. Tagieditori

5.5 NPS-kyselyn toiminnan testaaminen

NPS-kyselyn toimivuus haluttiin varmistaa luomalla kyselyvastauksista visualisoitu Data Studio raportti, jonka arvot haettiin MySQL-palvelimelta. Raporttia varten luotiin tyhjä raportti, jonka tietolähteeksi valittiin MySQL (ks. liite 4).

Seuraavaksi suoritettiin tietokantaan yhdistäminen (ks. liite 5), jota varten annettiin SQL-käyttäjätunnukset ja tietokannan nimi. Raportti ja tietolähde nimettiin ”NPS-Demoraportiksi”, jonka jälkeen tietokannassa olevat sarakkeet näkyivät Data Studiossa (ks. liite 6).

NPS-kyselyn arvosanojen hakemiseksi siirryttiin raportin editointitilaan, jossa luotiin pylväskaavio ”Lisää”-välilehdeltä. Kyselyn kautta annetut arvosanat asetettiin näkymään annetun arvosanan mukaan lajiteltuna ja määrällisesti yhteenlaskettuna. Kyseessä ei siis ole summa, vaan kuinka monta kertaa tietty arvosana tietokannassa esiintyy (ks. liite 7). Liitteen oikeasta alanurkasta voidaan nähdä raakadatana, miltä tietokannan sarakkeisiin tallennetut arvot näyttävät. Liitteestä 8 taas nähdään, miten NPS-kyselyssä annettu arvosana päivittyy reaaliajassa Data Studioon.

6 Analyticsin käyttöönottossa ilmenneet ongelmat ja niiden ratkaisut

6.1 Mixed content

Ladatessa NPS-kyselyä ensimmäistä kertaa Tag Managerin kautta kirjakauppa.com-sivustolla, ilmestyi selaimen konsoliin ”mixed content warning” -virheilmoitus (suom. ”sekoitetun sisällön varoitus”). Virhe huomattiin, koska kyselyn fontit renderöityivät verkkoselaimen oletusfontteina, vaikka käytössä olivat erikseen asennetut Bitter- ja PT Sans -Google-fontit.

Virheilmoitus ei kuitenkaan johtunut Tag Managerista. Ilmoitus tarkoitti, että palvelimella oleva tagi yrittää ladata kohdepalvelimen tiedostoja suojaamattoman HTTP-protokollan yli, vaikka tiedostoja pyytävä kirjakauppa.com-palvelin on määritetty käyttämään kaikkeen tiedostonsiirtoon suojattua HTTPS-protokollaa. (Mixed Content 2018.)

Tag Manager siis haki NPS-kyselyn JavaScript-, CSS- ja fontti-tiedostot suojaamattomana opinnäytetyötä varten pystytetyltä Ubuntu-web-palvelimelta. Ongelma korjaantui lisäämällä kaikkien Tag Managerin kautta ladattavien tiedostojen osoitteiden perään HTTPS-pääte. Tämän avulla saatiin muodostettua kirjakauppa.com-sivuston vaatima suojattu HTTPS-yhteys Tag Managerin sekä kirjakaupan ja Ubuntu-palvelimen välille, jonka jälkeen selain ei enää varoittanut suojatun sekä suojaamattoman sisällön sekoittamisesta kirjakaupan sivuilla.

6.2 JavaScript- ja tyylitiedostojen sijainti DOM-puussa

Tag Managerin HTML-editorilta näyttävä tagieditori eli ”tagisäiliö” poikkesi toiminnaltaan perinteisistä HTML-editoreista, sillä sen avulla ei voi kuvata verkkosivun rakennetta yhtä täsmällisesti kuin HTML-kielen avulla yleensä.

Tag Manager rajoittaa sellaisten tagien lisäystä ja muokkaamista, jotka ovat osa HTML-rungon perusrakennetta. Näitä ovat esimerkiksi head- ja body-tagit. Tämän vuoksi alun perin head-tagin sisällä ladatut skriptitiedostot ladataan vasta ennen sulkevaa body-tagia niillä sivustoilla, joilla Tag Manager -tagi on käytössä. Body-tagi noudattaa samaa periaatetta, joten kaikki body-osioon kirjoitettu HTML-koodi ladataan ennen sulkevaa body-tagia. Tämän avulla Tag Manager estää useamman body- ja head-tagin lisäämisen samalle sivulle ja hakee niiden sisään kirjoitetun koodin kohdesivuston oman body-tagin loppuun.

Tag Managerissa jotkin HTML-tagit ovat lisäksi vain Tag Managerin sisäistä metadataa, eivätkä kuvaa varsinaista kohdesivuston rakennetta. Html- ja script-tagien avulla Tag Manager osaa hakea tagieditorissa lisätyn koodin oikeaan paikkaan ennen sulkevaa body-tagia sen perusteella, onko lisätty teksti HTML- vai JavaScript-koodia.

Tag Managerin koodieditorin toimintalogiikan vuoksi osa NPS-kyselyn JavaScript-koodeista latautui body-tagin sisään, vaikka niiden kuului olla head-tagin sisällä. Lisäksi CSS-tyylitiedostot lataava link-tagi ei virallisen HTML-dokumentaation mukaan kuulu body-tagien sisälle ollenkaan, vaan niiden tulisi aina sijaita head-tagin sisällä. (Salvia 2013.)

Head-tagin sisälle haluttiin muun muassa jQuery-kirjaston lataus, jonka avulla oltiin toteutettu verkkolomakkeen uudelleenohjauksen esto. Head-tagin sisällä ladattua

jQuery-kirjastoa voidaan kutsua mistä tahansa kohtaa sivun koodia. Muutoin kutsu voidaan tehdä vasta body-tagin ja jQueryn lataavan script-tagin välissä, mikä ei ollut tarkoitus epäkäytännöllisyyden vuoksi. NPS-kyselyn koodi haluttiin myös HTML-validaattorista läpi, joten täytyi löytää keino, millä CSS-tyylitiedostot hakeva link-tagi saadaan ladattu jo head-tagin sisällä.

Ratkaisuna NPS-kyselyn koodia jouduttiin muokkaamaan Tag Managerin tagieditorin puolelta niin, että head-tagi poistettiin kokonaan. Seuraavaksi kirjoitettiin loadFile-funktio JavaScriptillä, jonka avulla ulkoinen CSS- tai JavaScript-tiedosto voitiin pakottaa latautumaan kohdesivustolla joko head- tai body-tagin sisään. Funtiolle voidaan syöttää argumentteina ladattavan tiedoston url-osoite, tiedoston tyyppi, sekä halutaanko se ladata head- vai body-tagissa.

6.3 JavaScript- ja tyylitiedostojen kansiopolun määrittäminen

NPS-kyselylle oltiin luotu liitteen 2 mukainen kansiorakenne, jonka tarkoitus oli erottaa palvelin- ja selainpuolen tiedostot omiksi loogisiksi ryhmikseen. Tämä oli tarpeen, koska lopullinen kysely koostui useammasta pienemmästä tiedostosta, joiden löytäminen tarpeen mukaan olisi muutoin ollut hidasta.

Kyselyn ydintiedostot sisältävän ”nps-min”-nimisen kansion ja sieltä löytyvien tiedostojen sijainti täytyi määritellä käsin Tag Managerissa absoluuttisella urlilla, jotka sitten ladattiin alaluvussa 6.2 esitellyn loadFile-funktion avulla. Jos kyselytiedostot sisältävän kansion sijaintia haluttiin muuttaa, niin tällöin kaikkien kansiossa olevien tiedostojen url-osoite täytyi päivittää Tag Managerin HTML-editorin koodiin.

NPS-kysely koostuu yhteensä 12 pienestä JavaScript- ja PHP-tiedostosta. Tag Managerin testausvaiheessa kansioiden nimiä ja sijaintia jouduttiin muuttamaan useamman kerran, joten uuden tiedostopolun määrittäminen kaikille erikseen osoittautui työlääksi.

Ratkaisuksi kirjoitettiin pieni getUrl-niminen PHP-ohjelma, joka asetettiin samaan kansioon NPS-kyselyn kooditiedostojen kanssa (ks. liite 9). Ohjelma tulostaa JSON-objektimuodossa kyselyn kooditiedostot sisältävän kansion absoluuttisen url-osoitteen. Ohjelma hakee tiedon ”$_SERVER”-nimisen taulukon sisältä, joka luodaan

kaikilla web-palvelimilla automaattisesti. Oletuksena taulukkoon säilötään ajossa olevan tiedoston urlin lisäksi HTTP-headereihin liittyvää tietoa. ($_SERVER n.d.)

Seuraavaksi Tag Managerin puolelle kirjoitettiin loadUrl-funktio, joka osaa Ajaxin avulla hakea getUrl-ohjelman tulostaman JSON-objektin output-nimiseen muuttujaan. LoadUrl-funktiota käytettiin välittämällä tämän sisään argumenttina toinen, success-niminen funktio. Kyseessä on callback-funktio, jonka sisällä lueteltiin kaikkien NPS-kyselyn kooditiedostojen alku- ja loppuosoitteet loadFile-funktiolle välitettävinä argumentteina. LoadFile-funktiolle välitettävät alku- ja loppuosoitteet olivat muotoa ”output + /kansio/tiedosto.pääte”, josta output on JSON-objektilta saadun kansion absoluuttinen polku.

Success-funktio odottaa, että loadUrl-funktio on ajanut itsensä ja hakenut URL-osoitteen sisältämän JSON-objektin, jonka jälkeen se ajaa itsensä loadUrl-funktion sisällä. Lopuksi ajetaan success-funktion itsensä sisällä oleva loadFile-funktio, jossa vasta varsinainen kyselytiedostojen haku tehdään.

JSON-objekti ja tämän sisältämä url näkyvät loadFile-funktiolle, koska se sijaitsee loadUrl-funktiolle välitetyn toisen funktion (success) sisällä. Jos loadFile-funktiota olisi kutsuttu suoraan loadUrl-funktion sisällä, niin output-muuttuja olisi tyhjä, koska Ajax-käskyn suorittamisessa on pieni viive. Tämän jälkeen ei enää tarvinnut määritellä kuin yksi absoluuttinen url-osoite: PHP-tiedosto, josta NPS-kyselyn kansiopolku haettiin.

6.4 CORS

Tag Manager -tagi ei enää toiminut sen jälkeen, kun NPS-kysely asetettiin lataamaan ydintiedostonsa automaattisesti edellisessä luvussa 6.3 esitellyn getUrl.php-ohjelman avulla. Selaimen konsoli antoi CORS eli ”Cross-Origin Request Blocked”-virheilmoituksen. CORS on tietoturvaan liittyvä standardi, jonka avulla rajoitetaan, voiko paikallisella palvelimella (Ubuntu-web-palvelin) olevia ohjelmia tai tiedostoja ajaa toisen palvelimen pyynnöstä.

NPS-kyselyn hyödyntämää PHP-tiedostoa yritettiin ladata kirjakauppa.com-sivustolla Googlen palvelimien (Tag Managerin) kautta, johon tällä ei ollut tarvittavaa ajo- tai suorituslupaa. (CORS errors 2018.)

Virheilmoituksesta pääsee eroon kutsumalla ilmoituksen aiheuttamassa PHP-tiedostossa header-funktiota, joka lähettää tiedoston pyytäjälle eli selaimeen käyttöoikeudet HTTP-otsakkeena.

Tässä tapauksessa täytyi lähettää kaksi otsaketta: ”Access-Control-Allow-Origin: https://www.kirjakauppa.com” sekä ”Access-Control-Allow-Methods: GET, POST, OPTIONS”. Ensimmäinen antaa luvan käyttää Ubuntu-palvelimella olevia tiedostoja kirjakauppa.com-osoitteesta CORSia hyödyntäen ja jälkimmäinen mahdollistaa GET-metodin, jota käytetään muun muassa Ajax-kyselyjen tekemiseen. (Access-Control-Allow-Methods 2017; Access-Control-Allow-Origin 2018.)

6.5 Koodikirjastoa ei ole määritelty

NPS-kysely oltiin saatu latautumaan kirjakauppa.com-sivustolla ja menemään HTML-validaattorista läpi, mutta jQyery-kirjastolla toteutettu HTML-lomakkeen uudelleenohjauksen esto ei toiminut. Selain antoi ilmoituksen, jonka mukaan jQueryä ei ole määritelty, eli se ei ole latautunut sivuille ollenkaan. jQyery kuitenkin löytyi oikealta paikaltaan head-tagin sisällä ladattuna.

Tämä tarkoitti, että jQueryä yritettiin kutsua ennen kuin se oli vielä kerennyt latautua. Syy löytyi jQueryn lataavasta ”load_jquery.js”-nimisestä JavaScript-tiedosto, jolla NPS-kyselyn tarvitsema kirjasto ladattiin sivun head-tagissa. Normaalisti head-tagin sisältö ladataan aina ennen body-tagia, jonka sisällä jQuerystä riippuvainen lähetyksen estävä ”jquery_submitForm” JavaScript-tiedosto oli.

jQueryn lataaminen oltiin toteutettu niin, että se haettiin head-tagin sisään ohjelmallisesti JavaScriptin avulla. Tämä loi tiedoston lataamiseen viivettä, jonka takia body-tagin sisällä olevat riippuvuustiedostot latautuivat ennen jQueryä. Ongelma ratkaistiin rekursiivisesti itseään enintään 5 sekunnin ajan kutsuvaa waitjQuery-funktiota hyödyntäen.

Funktion avulla mitään NPS-kyselyn skripti- tai tyylitiedostoja ei ladattaisi ennen kuin jQuery on latautunut. Käytännössä tämä tarkoittaa, että kyselyä ei tällöin ladata ollenkaan. Tämän jälkeen virheilmoitusta jQueryn puuttumisesta ei enää tullut. Funktion toiminta näkyy kuviosta 25.

```
var waitjQuery = function () {
  "use strict";
  time_ms = time_ms + 500;
  if (time_ms <= 5000) {
    if (typeof jQuery === "undefined") {
      window.setTimeout(waitjQuery, 500);
    } else {
      // Lataa nps- kysely
    }
  } else {
    console.log("NPS: Unable to load jQuery");
  }
};
waitjQuery();
```

Kuvio 25. jQueryn lataava funktio

6.6 Fouc

Fouc (flash of Unstyled Content) tarkoittaa sivua tai sivun osaa, jonka CSS-tyylimärittelyt eivät näy sivua ladatessa muutamaan sekuntiin tai jopa minuuttiin. Joskus fouc voi tapahtua myös sellaisilla sivuilla, joiden tyylit on toteutettu JavaScriptin tai jQueryn avulla. Boudreaux (2013) kertoo artikkelissaan, että yleisin tapa estää fouc on lisätä tyylimäärittelyt mahdollisuuksien mukaan sivun head-tagin sisään. Suurimalle osalle web-kehittäjistä tämä on kuitenkin jo lähes itsestäänselvyys, koska head-tagin ulkopuolella määritellyt tyylit ovat virallisen HTML-standardin vastaisia. (Boudreaux 2013.)

NPS-kyselyssä tyylitiedostot oli määritetty head-tagiin, eikä käytössä ollut JavaSciptin avulla toteutettuja tyylimuutoksia (lukuun ottamatta animaatioita). Tag Managerin kautta ladattu kysely kuitenkin välähti sivua ladatessa tyylittömänä. Samaa ongelmaa ei ilmennyt ajettaessa kysely suoraan opinnäytetyön Ubuntu-palvelimella ilman Tag Manageria.

Yhdeksi syyksi epäiltiin Tag Manager -toteutukseen lisättyä CSS- ja JavaScript-tiedostoja lataavaa loadFile-funktiota. CSS-tiedostojen hakeminen funktion avulla saattoi viedä sen verran aikaa, että sivulla oleva HTML-sisältö sisältö latautui reilusti ennen tyylimäärittelyjä. Välkkyminen korjattiin piilottamalla kysely JavaScriptin avulla siksi aikaa, kunnes sivu on latautunut kokonaan. Tämä tehtiin asettamalla kyselyikkunan display-arvoksi ”none”. Boudreaux (2013) suosittelee artikkelissaan käyttämään samaa tekniikkaa, joten tämä vahvistaa ratkaisun oikeellisuuden.

7 Tulokset ja pohdinta

Opinnäytetyön yksi tavoitteista oli selvittää, soveltuuko verkkoliikenteen analysointiin tarkoitettu Google Analytics Suite -ohjelmistokokoelma etäajettavan NPS-kyselyn toteuttamiseen, sekä voitaisiinko tällä helpottaa kyselyn toimitusta ja jatkokehitystä asiakkaan kannalta. Opinnäytetyötä lähdettiin alun perin tekemään työelämän tarpeiden pohjalta, mutta alkuperäisen NPS-kyselyn ja Analytics Suite -tutkimuksen tilaaja päätettiin kuitenkin häivyttää lopullisesta työstä pois.

Lopputuloksena saatiin aikaiseksi NPS-asiakaskyselyohjelma, joka oli Google Analyticsin kautta mahdollista etäajaa millä tahansa palvelimella. Asiakaskyselyohjelmaan jäi kuitenkin muutamia bugeja, kuten ikkunan rajautuminen osittain näytön ulkopuolelle tietyillä resoluutioilla. Lisäksi jotkin ohjelmistokehykset kuten Bootstrap sekoittivat kyselyn omat tyylimäärittelyt, joten jatkokehitystä tämän suhteen vielä kaivataan. Tämän vuoksi NPS-kyselyä ei päästy vielä testaamaan oikealla asiakkaalla.

Matkan varrella oli lisäksi useita jo opinnäytetyössä mainittuja haasteita, kuten CORS, fouc ja Mixed-Content. Analytics todettiin kuitenkin ongelmien selättämisen jälkeen toimivaksi alustaksi verkkosovelluksen toteuttamiseen. Etäajon ansiosta kyselyn lähdekoodia on mahdollista päivittää taustalla ilman FTP-tunnuksia tai verkkopalvelinoikeuksia. Tämä on selkeä etu asiakkaan kannalta verrattuna perinteisiin ratkaisuihin, joissa etäajo ei ole mahdollista, vaan kooditiedostot on toimitettava asiakkaalle aina erikseen.

Opinnäytetyössä käytiin läpi myös verkkosovelluskehitykseen liittyvää tietoturvanäkökulmaa, hyviä ohjelmointikäytänteitä, responsiivisuutta ja käytettävyyttä, joita heijastettiin Google Analyticsin avulla toteutetussa NPS-sovelluksessa. Kysely skaa-

lautui sulavasti monelle eri resoluutioille, animaatiot olivat optimoinnin ansiosta sulavia, eivätkä mahdolliset SQL-injektiot pääse kyselyn läpi. Näin tultiin todistaneeksi, ettei Google Analytics asettanut mitään esteitä esimerkiksi Prepared Statementtien tai muuttujien suojaamisen suhteen.

Eteen tulleine ongelmineen opinnäytetyö osoittautui suhteellisen haasteelliseksi. Suurin osa ajasta meni Google Analyticsiin tutustuessa, joka ei kuitenkaan sinällään haittaa, koska tämä oli koko opinnäytetyön tärkeintä antia. Lisäksi se voitiin tyhjentävästi todeta kelvolliseksi työkaluksi niin verkkoliikenteen analysointiin, kuin etäajettavan asiakaskyselyn toteuttamiseen.

Lähteet

Access-Control-Allow-Methods. 2017. Mozillan HTTP-manuaali. Viitattu 29.12.2018. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Methods

Access-Control-Allow-Origin. 2018. Mozillan HTTP-manuaali. Viitattu 29.12.2018. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin

Apple iPhone 8. N.d. Tietokanta älypuhelinten teknisistä ominaisuuksista. Viitattu 22.10.2018. https://www.gsmarena.com/apple_iphone_8-8573.php

Apple Watch Series 3. N.d. Tietokanta älypuhelinten teknisistä ominaisuuksista. Viitattu 22.10.2018. https://www.gsmarena.com/apple_watch_series_3-8860.php

Application Vulnerability Trends Report. 2014. Viitattu 19.10.2018. https://www.infopoint-security.de/medien/cenzic-vulnerability-report-2014.pdf

Block. 2018. JavaScript-opetusmanuaali. Viitattu 12.12.2018. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block

Boudreaux, R. 2013. How to prevent Flash of Unstyled Content on your websites. Viitattu 21.12.2018. https://www.techrepublic.com/blog/web-designer/how-to-prevent-flash-of-unstyled-content-on-your-websites/

Bowlby, S. 2014. 6 Phases of the Web Site Design and Development Process. Blogiteksti. Viitattu 29.10.2018. https://www.idesignstudios.com/web-design/phases-web-design-development-process/

Can I use css3. N.d. Tietokanta eri selainten tukemista ominaisuuksista. Viitattu 29.10.2018. https://caniuse.com/#search=css3

Chikuyonok, S. 2016. CSS GPU Animation: Doing It Right. Viitattu 04.12.2018. https://www.smashingmagazine.com/2016/12/gpu-animation-doing-it-right/

CORS errors. 2018. Mozillan HTTP-manuaali. Viitattu 29.12.2018. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors

CSS Animations. N.d. Ohjeivusto CSS-kielen käytöstä. Viitattu 02.12.2018. https://www.w3schools.com/css/css3_animations.asp

CSS top Property. N.d. CSS-manuaali. Viitattu 03.12.2018. https://www.w3schools.com/cssref/pr_pos_top.asp

Custom tags. 2018. Tag Manager käyttöopas. Viitattu 19.11.2018. https://support.google.com/tagmanager/answer/6107167

Deploy Google Analytics with Tag Manager. 2018. Tag Manager käyttöopas. Viitattu 19.11.2018. https://support.google.com/tagmanager/answer/6107124?hl=en

Gant, A. N.d. What is Google Tag Manager and why use it. Viitattu 18.11.2018. https://www.orbitmedia.com/blog/what-is-google-tag-manager-and-why-use-it/

Google Analytics is 10 years old – What's changed. N.d. Blogteksti. Viitattu 16.11.2018. https://neilpatel.com/blog/google-analytics-360/

Gregory, S. 2018. Why Responsive Design is Important and Google Approved. Blogiteksti. Viitattu 21.10.2018. https://freshsparks.com/why-responsive-design-is-important/

Guzel, B. 2009. HTTP Headers for Dummies. Ohjesivusto web-kehitykseen liittyen. https://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039

Harmon, K. 2016. What Is Google Tag Manager? (And How Does It Work With Google Analytics?). Viitattu 10.12.2018. https://www.bounteous.com/insights/2016/02/15/what-is-google-tag-manager/

Herszak, M. 2015. JavaScript — A Sneak Peak into the Essentials (Part 3). Viitattu 13.12.2018. https://hackernoon.com/javascript-a-sneak-peak-into-the-essentials-part-3-c97a9c2e933f

Hillsberg, A. 2018. Business intelligence software. Viitattu 17.10.2018. https://business-intelligence.financesonline.com/

Hooper, A. 2016. In MySQL, never use "utf8". Viitattu 29.12.2018. https://medium.com/@adamhooper/in-mysql-never-use-utf8-use-utf8mb4-11761243e434

How does a PreparedStatement avoid or prevent SQL injection. 2017. Keskutelufoorumi. Viitattu 30.12.2018. https://stackoverflow.com/questions/1582161/how-does-a-preparedstatement-avoid-or-prevent-sql-injection

HTML Entities. N.d. Ohjesivusto HTML-kielen käytöstä. Viitattu 12.12.2018. https://www.w3schools.com/html/html_entities.asp

HTML <label> Tag. N.d. HTML-manuaali. Viitattu 05.12.2018. https://www.w3schools.com/tags/tag_label.asp

HTML URL Encoding Reference. N.d. HTML-manuaali. Viitattu 16.12.2018. https://www.w3schools.com/tags/ref_urlencode.asp

Is Google analytics free of charge. 2015. Keskustelufoorumi. Viitattu 06.11.2018. https://webmasters.stackexchange.com/questions/81084/is-google-analytics-free-of-charge

Ivanov, D. 2015. CSS Transitions. Viitattu 02.12.2018. https://zinoui.com/blog/css-transitions

JavaScript Scope. N.d. Ohjesivusto JavaScript-kielen käytöstä. Viitattu 12.12.2018. https://www.w3schools.com/js/js_scope.asp

jQuery serialize() Method. N.d. Ohjesivusto jQuery JavaScript-kirjaston käytöstä. Viitattu 16.12.2018. https://www.w3schools.com/jquery/ajax_serialize.asp

Kierczak, L. N.d. 5 Reasons Why Customer Satisfaction Is Important. Blogikirjoitus asiakastyytyväisyyden merkityksestä. Viitattu 17.01.2019. https://survicate.com/customer-satisfaction/importance-customer-satisfaction/

Lardinois, R. 2011. Move Over 1024x768: The Most Popular Screen Resolution On The Web Is Now 1366x768. Viitattu 09.12.2018. https://techcrunch.com/2012/04/11/move-over-1024x768-the-most-popular-screen-resolution-on-the-web-is-now-1366x768/

Long, L. 2016. Marketing Tags and Pixels - What They Are and How They Work. Viitattu 18.11.2018. https://taginspector.com/marketing-tags-and-pixels-form-and-function/

Mixed Content. 2018. Verkkoturvallisuuteen liittyvä manuaali. Viitattu 17.12.2018. https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content

Moore, K. N.d. Google Analytics 360 Suite: What You Need to Know. Viitattu 16.11.2018. https://neilpatel.com/blog/google-analytics-360/

Naik, A. SQL Injection in INSERT Query. 2012. Blogiteksti. Viitattu 30.12.2018. http://amolnaik4.blogspot.com/2012/02/sql-injection-in-insert-query.html

Net Promoter Score eli NPS. N.d. Tietosivu NPS-kyselyn toiminnasta. Viitattu 24.11.2018. https://www.surveypal.com/fi/net-promoter-score

Net Promoter Score (NPS) Survey Template. N.d. NPS-kyselyn mallipohjaa esittelevä sivu. Viitattu 24.11.2018. https://www.surveymonkey.com/mp/net-promoter-score-survey-template/

PHP 5 MySQLi Functions. N.d. PHP-manuaali. Viitattu 29.12.2018. https://www.w3schools.com/php/php_ref_mysqli.asp

PHP Prepared Statements. N.d. PHP-manuaali. Viitattu 30.12.2018. https://www.w3schools.com/php/php_mysql_prepared_statements.asp

Popular Screen Resolutions: Designing for All. 2018. Blogiteksti. Viitattu 19.10.2018. https://mediag.com/news/popular-screen-resolutions-designing-for-all/

Responsive Web Design - Media Queries. N.d. CSS-manuaali responsiivisuudesta. Viitattu 15.12.2018. https://www.w3schools.com/css/css_rwd_mediaqueries.asp

Rouse, M. 2006. querystring. Viitattu 15.12.2018. https://whatis.techtarget.com/definition/querystring

Rouse, M. 2011. Google Analytics. Viitattu 16.11.2018. https://searchbusinessanalytics.techtarget.com/definition/Google-Analytics

Rouse, M. 2016. Software as a Service (SaaS). Viitattu 08.12.2018. https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service

Salvia, C. Using <style> tags in the <body> with other HTML. 2013. Viitattu 20.12.2018. https://stackoverflow.com/questions/2830296/using-style-tags-in-the-body-with-other-html

$_SERVER. N.d. PHP-manuaali. Viitattu 28.12.2018. http://php.net/manual/en/reserved.variables.server.php

Sharmila, N. Gartner: 1.5 billion smartphones were sold in 2017. 2018. Uutinen älypuhelinten myyntitilastoista. Viitattu 28.10.2018.

https://www.thestar.com.my/tech/tech-news/2018/02/26/1point5-billion-smartphones-were-sold-in-2017/

Smartwatch unit sales worldwide from 2014 to 2018. N.d. Pylväsdiagrammi älykellojen myynnistä. Viitattu 28.10.2018. https://www.statista.com/statistics/538237/global-smartwatch-unit-sales/

SQL Injection. N.d. SQL-manuaali. Viitattu 29.12.2018. https://www.w3schools.com/sql/sql_injection.asp

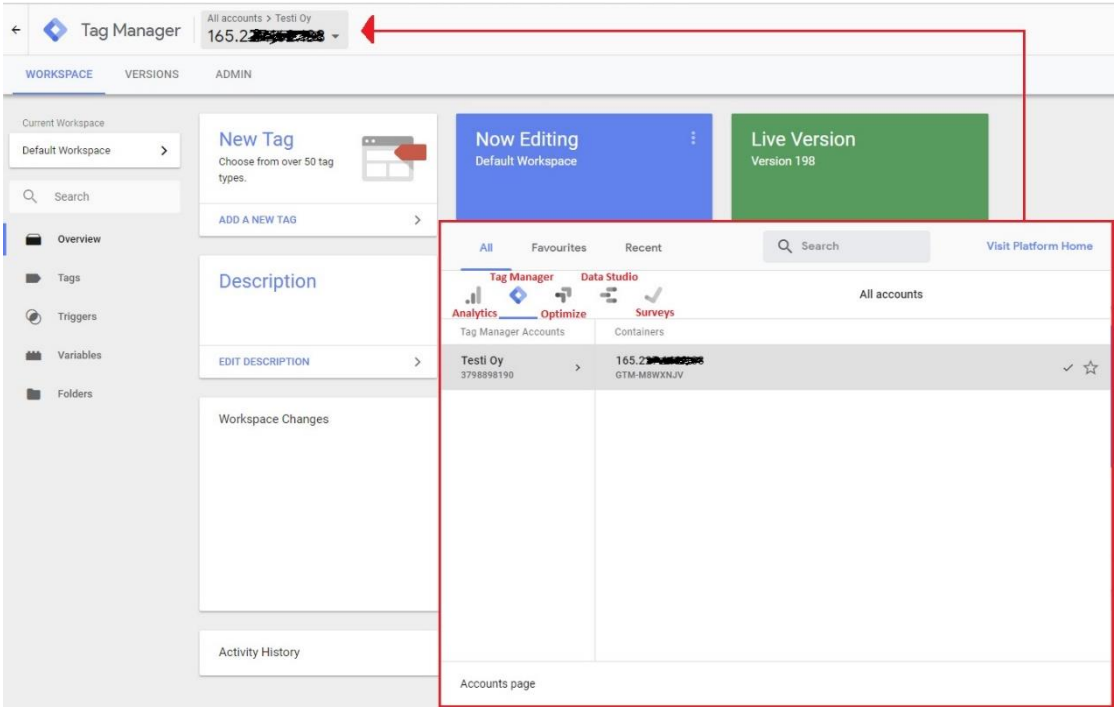
Stangarone, J. 2015. 7 web development challenges you can't ignore. Blogiteksti. Viivattu 17.10.2018. https://www.mrc-productivity.com/blog/2015/05/7-web-development-challenges-you-cant-ignore/

What is Google Data Studio. 2017. Blogiteksti. Viitattu 24.11.2018. https://www.seerinteractive.com/blog/google-data-studio-whats-working-whats-missing/

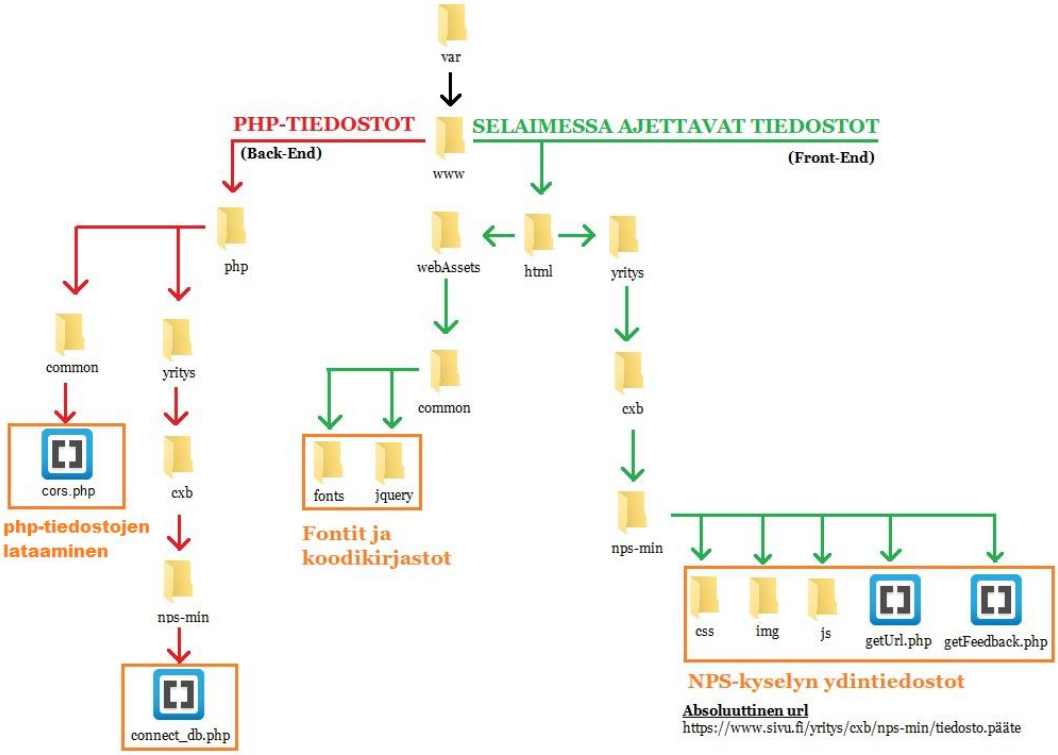
What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care. 2016. Blogiteksti responsiivisuudesta. Viitattu 21.10.2018. https://medium.com/@space.alpaca/so-what-exactly-is-the-difference-between-fixed-fluid-adaptive-and-responsive-layouts-and-why-3773272d8481

Liitteet

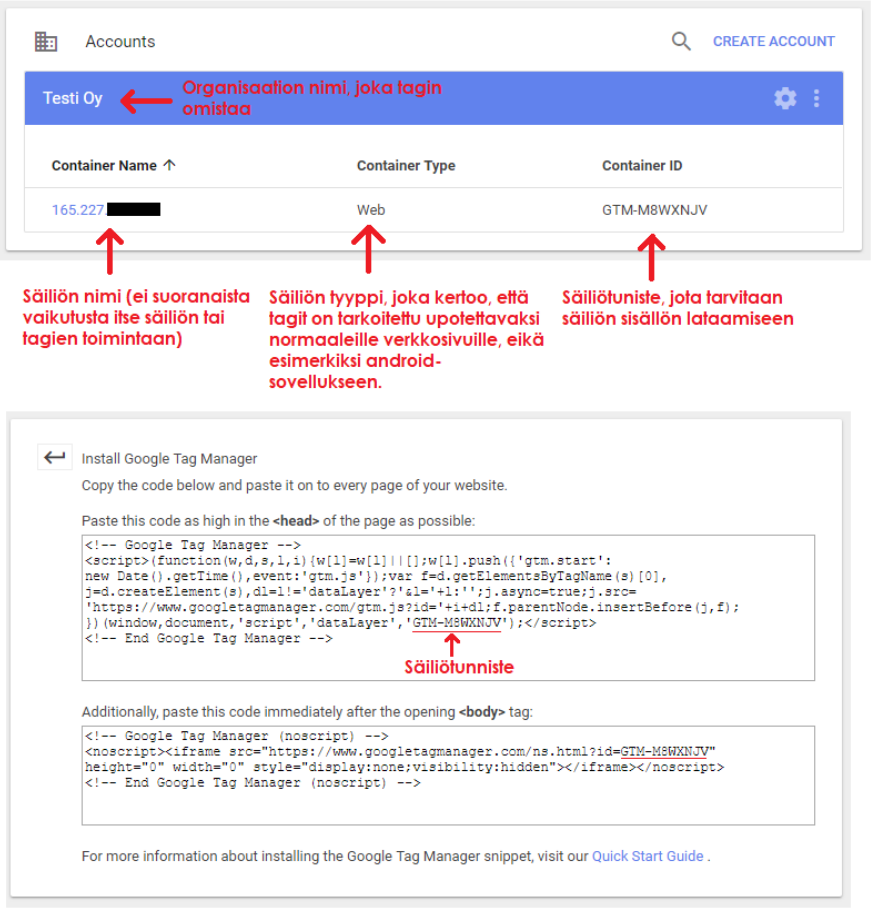
Liite 1. Sovelluksen valinta Analytics Suitessa



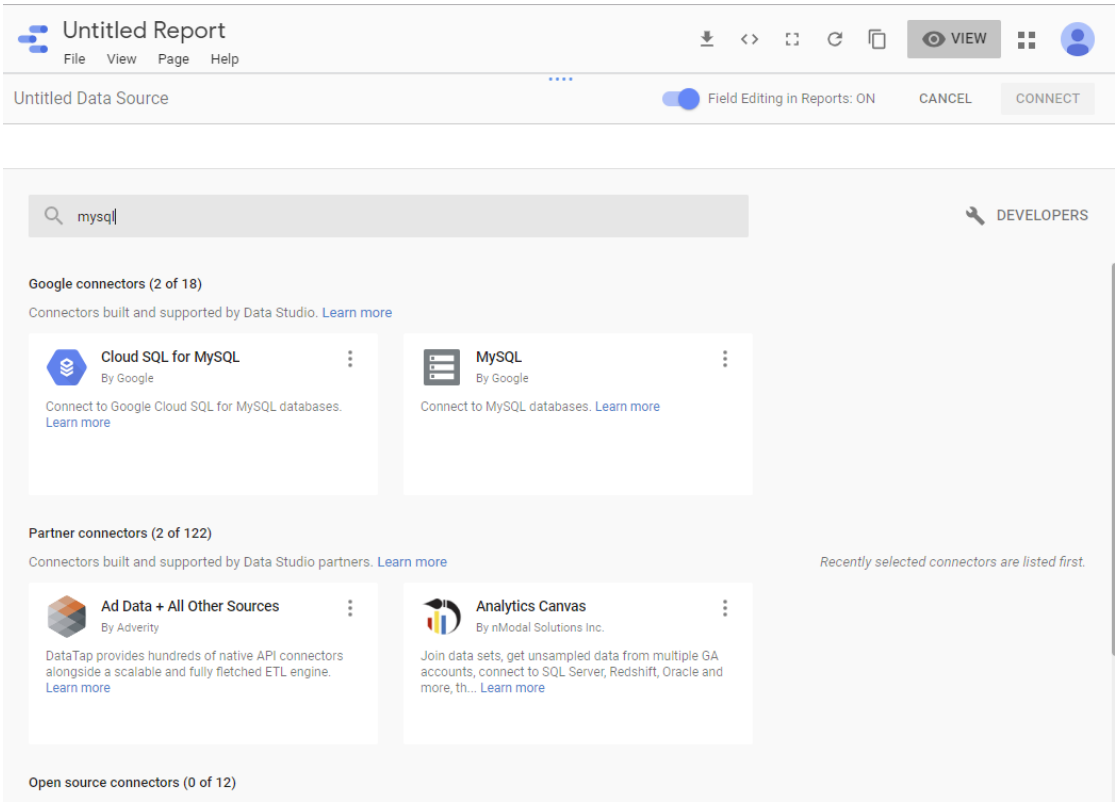
Liite 2. NPS-kyselyohjelman kansiorakenne



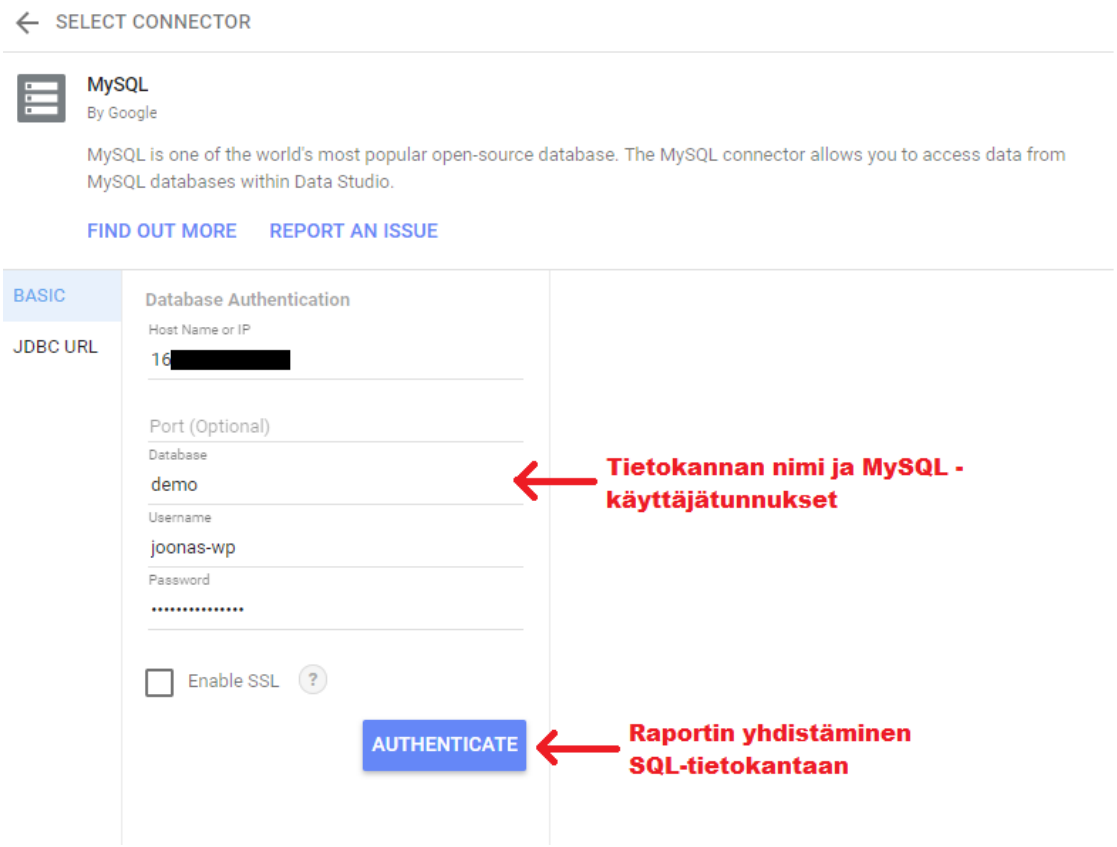
Liite 3. Tag Managerin asentaminen



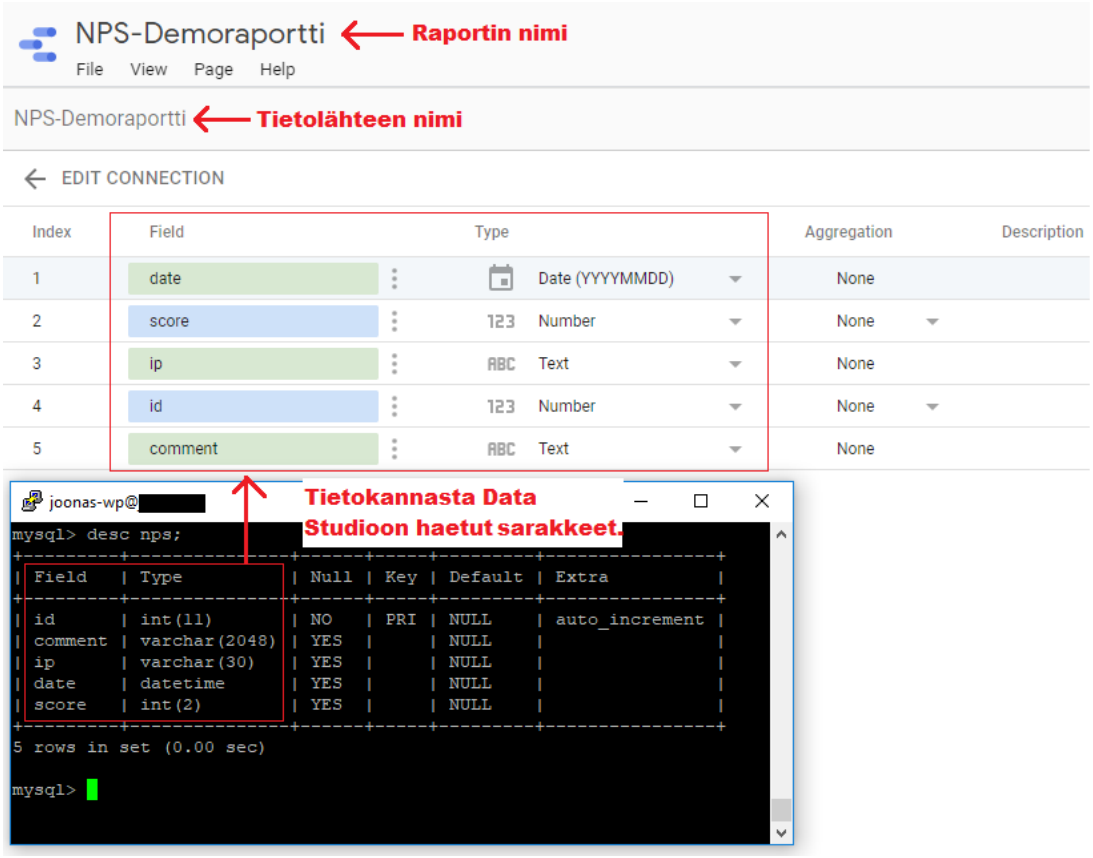
Liite 4. Data Studio - Tietolähteen valitseminen



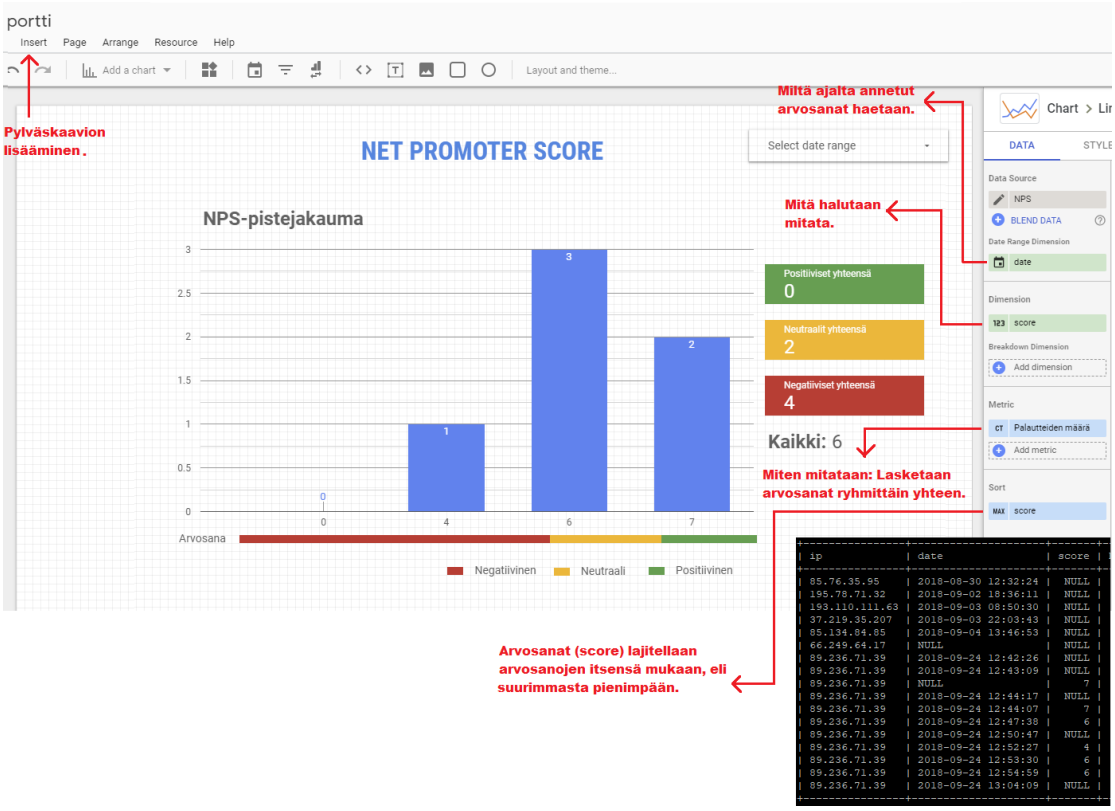
Liite 5. Data Studio - Raportin yhdistäminen tietokantaan



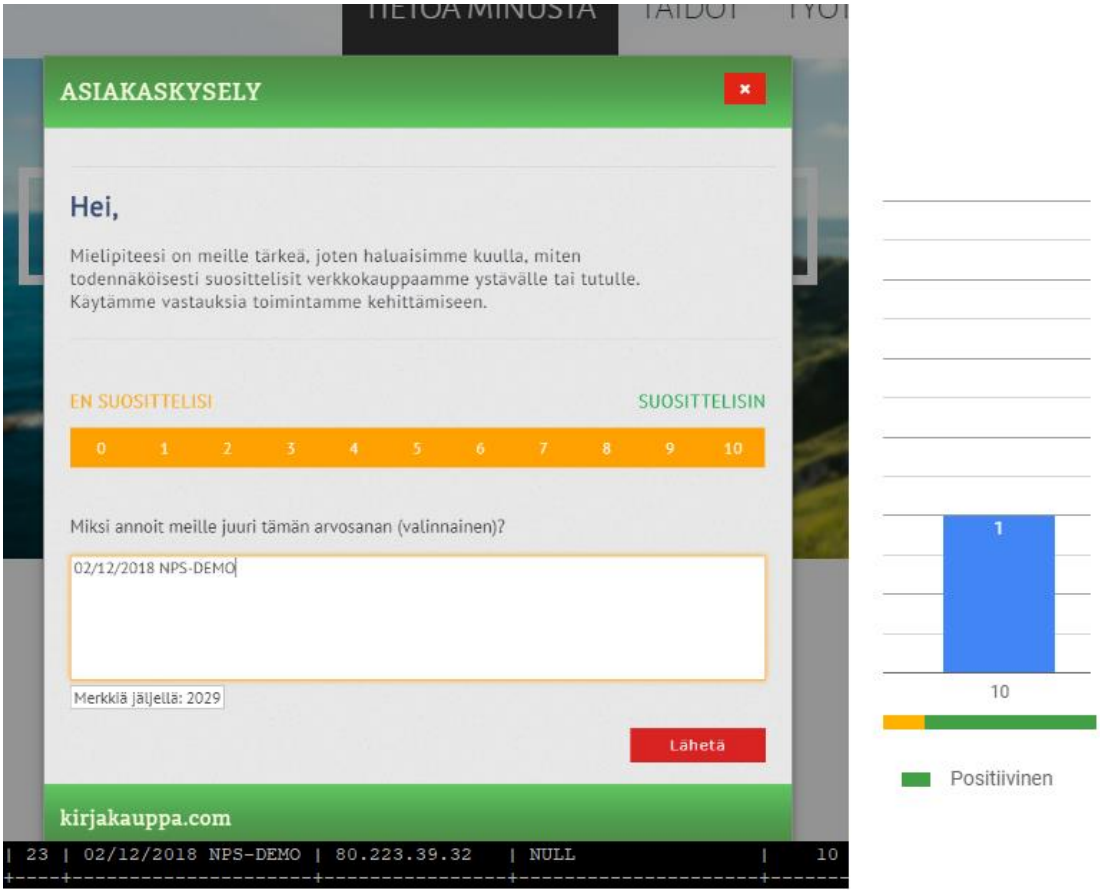
Liite 6. Data Studio - Sarakkeiden haku



Liite 7. NPS-kyselyn vastaustuloksien haku



Liite 8. Arvosanan antaminen NPS-kyselyssä



Liite 9. Tiedostojen automaattinen urlin hakeminen

URLIN HAKEVA FUNKTIO

```
function loadURL(success, error) {
  var xhr = new XMLHttpRequest();
  xhr.open("POST", "https://[redacted].fi/[redacted]/cxb/nps-min/getUrl.php", true);
  xhr.setRequestHeader('Content-Type', 'application/json');
  xhr.onreadystatechange = function() {
    if(xhr.readyState === XMLHttpRequest.DONE) {
      if(xhr.status == 200 && success) {
        var data = JSON.parse(xhr.responseText);
        var url = data.url;
        success(url);
      } else {
        var data = 'Error: ' + xhr.status;
        error(data);
      }
    }
  };
  xhr.send();
}
```

getUrl.php -tiedoston tulostama JSON-objekti

{"url":"https://[redacted].fi/[redacted]/cxb/nps-min"}

CALLBACK -FUNKTIO

```
loadURL(
  (function success(output) {
    var frm_close = document.getElementById("scoreBox-closeBtn-submit"),
        frm_send = document.getElementById("scoreBox-feedback");

    if(frm_close) {
      frm_close.action = output + "/getFeedback.php";
      frm_send.action = output + "/getFeedback.php";
    }

    loadFile(output + "/js/dateTime-min.js", "js", "body");
    loadFile(output + "/js/setCookie-min.js", "js", "body");
    loadFile(output + "/js/scoreBox-min.js", "js", "body");
    loadFile(output + "/js/modal-min.js", "js", "body");
    loadFile(output + "/js/jquery_submitForm-min.js", "js", "body");
    loadFile(output + "/js/detect-ie-min.js", "js", "body");

    loadFile(output + "/css/google-fonts-min.css", "css", "head");
    loadFile(output + "/css/modal-min.css", "css", "head");
    loadFile(output + "/css/alertBox-min.css", "css", "head");
    loadFile(output + "/css/scoreBox-min.css", "css", "head");

    window.addEventListener("DOMContentLoaded", function() {
      document.getElementsByClassName("scoreBox")[0].style.display = "block";
    });
  }),
  (function error(output) { console.log(output); })
);
```