LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Startup-yrityksen verkkopalveluiden infrastruktuuri

Kiviniemi, Jiri

2019 Laurea

LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Startup-yrityksen verkkopalveluiden infrastruktuuri

Jiri Kiviniemi
Tradenomi (AMK)
Opiinäytetyö
Toukokuu, 2019

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Jiri Kiviniemi

Startup-yrityksen verkkopalveluiden infrastruktuuri

Vuosi 2019 Sivumäärä 57

Opinnäytetyössä asiakasyritykselle tuotettiin kotisivut suomeksi ja englanniksi, sekä ylläpitokanava sivustojen dynaamiselle sisällölle. Toteutuksen painopisteenä oli erityisesti kokonaisuus tuotantoympäristön, backendin ja frontendin välillä. Tavoitteena on tarjota ohjenuora ja kokonaiskuva, mitä sivustokokonaisuuden tuottamiseen sisältyy, kun jokainen toimialue halutaan pitää vapaasti kontrolloitavana.

Opinnäytetyö on myös projektihallinnan näyttö tarjoten hallinnollista ohjenuoraa vastaavaksi luokiteltaviin ohjelmistokehityksen projekteihin. Projekti esittää projektiorganisaation ja toiminnan Scrum -viitekehyksen malliin, sekä asiakaslähtöisen toimintatavan tuotoksen vaatimusmäärittelyn rakentamiseen. Työssä koostetaan vaatimusmäärittelyn pohjalta Product Backlog ja toteutus puretaan tehtäväkohtaisesti toteutussykleille.

Projektin onnistumista arvioidaan asiakkaan, käyttäjien ja kehittäjien käyttökemuksen perusteella sekä vertaamalla tuotosta vaatimusmäärittelyyn. Tämän lisäksi sivustolle suoritetaan auditointi Googlen tarjoamilla työkaluilla ja analysoidaan palvelimen kerryttämää dataa Micosoftin Azure palvelusta.

Lopputuotteena projektista syntyi asiakkaalle kaivattu tuotos, sekä kokonaiskäsityksen luova malli mahdollisesta infrastruktuuri kokonaisuudesta web -palveluita toteuttaville startup yrityksille.

Asiasanat: Scrum, Azure, Kehitysjono, Sprintti

Laurea university of applied sciences **Abstract**
Degree programme in Business Information Technology
Bachelor's thesis

Jiri Kiviniemi

Web Platform Infrastructure for Startup Companies

Year 2019 Pages 57

The objective of this thesis was to create a homepage for the commissioner both in Finnish and in English. In order to manage dynamic content featured on a web page, small administration interface was also created. The thesis aims to provide an instructional example of execution for projects of this nature covering all the areas required to build a fully controlled infrastructure for web platform services.

This thesis also provides a workflow model for software development projects managed with Scrum. The product specification is created with customer oriented approach taking full advantage of empiric project organization that is the essence of Scrum.

The success of the thesis is defined based on user experiences reported by the commissioner, users and development team. The homepage is also compared to its original specification and audited with performance tools provided by Google and Azure.

The result of the thesis was a homepage for the commissioner. In addition to the homepage completed, the thesis succeeded in providing an instructional example for projects of this nature covering all the levels of Web Platform Infrastructure.

Keywords: Scrum, Azure, Product Backlog, Sprint

Sisällys

1 Johdanto

Projektin toimeksiantajana toimii Huoleti Oy. Kyseessä on suomalainen startup yritys, joka tarjoaa palveluna käyttöön mobiilisovellusta vertaistukiverkostoitumiselle. Yritys tarvitsee itselleen verkkosivustot niin suomeksi kuin englanniksi. Sivustojen on tarkoitus esitellä yritystä, sekä tarjota ohjeistusta ja tietoa sovelluksen käyttöön. Julkisten sivustojen lisäksi yritykselle tuotetaan ylläpitokäyttöliittymä, jolla mahdollistetaan sivuston dynaamisen sisällön, kuten blogin ja uutislinkkien, hallinta.

Kehitysprojektia ajetaan hyödyntämällä Scrum -viitekehystä, mikä on työkalu ohjelmistokehityksen projektihallintaan. Projektin onnistuminen määritetään tuottamalla sivuston ominaisuuksista ja toiminallisuuksista vaatimusmäärittely tuotteen kehitystä varten ja vertaamalla vaatimusmäärittelyn toteumaa valmiiseen tuotteeseen.

Kyseessä on toiminnallinen opinnäytetyö missä luodaan konkreettinen tuote nojaten ensiksi tuotettuun vaatimusmäärittelyyn. Vaatimusmäärittely tuotetaan haastattelemalla asiakasyrityksen henkilökuntaa ominaisuuksien tarpeesta ja suunnitellen asiakkaan vaateiden perusteella heille sopivimmat teknologiaratkaisut asiantuntijaorganisaation suositusten mukaisesti. Parhaan käytännön toteumaa arvioidaan myös Microsoftin ja Googlen palveluiden tuottaman datan perusteella.

Scrum -viitekehys on suosittu ohjelmistokehityksen hallintatapa. Scrum on jalostettu ketteristä kehitysmenetelmistä ja se soveltuu pienryhmien ohjelmistokehityksen hallinnalle. Yksinkertaisuudessaan Scrumissa määritetään tehtävät ja tavoitteet jokaiselle kehityssyklille sekä tavoitellulle artefaktille. Kun kehityssykli on saavutettu arvioidaan tekemättömät tehtävät artefaktin saavuttamiseksi ja näistä loogisesti seuraavaksi toteutettavat vaiheet (Schwaber, 2016).

Asiakkaan infrastruktuurin perustana tulee toimimaan Microsoftin Azure -palvelu, joka tarjoaa palvelin tilaa, käyttöliittymän palvelimen hallinnalle, sekä laajan tarjonnan web -kehityksen viitekehyksiä jättäen front- ja backend teknologioille valinnanvaraa kehittäjien preferenssin mukaan.

Markkinoilla olevista vaihtoehdoista tämä on ehdottomasti vähiten ylläpitotyötä vaativa ratkaisu, sillä sekä toimintaympäristön, että fyysisten palvelimien ylläpito on Microsoftin vastuulla palvelumaksua vastaan.

Palvelumaksun summa määräytyy käytettyjen palveluiden mukaan. Helpotukseksi Microsoft myöntää startup yrityksille BizSpark -stipendiä, joka kattaa viidelle käyttäjälle lisenssin Microsoft Visual Studio -kehitysympäristöön sekä jokaiselle tilille 130$ käyttövaraa Azuren toimintoihin. Hinnastosta mainittakoon, että keskivertosuorituskykyinen toimintaympäristö viidelle sivustokokonaisuudelle kustantaa noin 60$/kk. Käyttövarasta säästyy siis vielä merkittävä osa esimerkiksi teitokantayhteyden dataliikennepalveluihin.

Projektisuunnitelmassa artefaktin työvaiheista koostetaan Product Backlog, jonka tehtävät aikataulutetaan kronologisesti toisistaan rippuvaan järjestykseen asiantuntijaorganisaation kesken. Työtehtävät ajoitetaan sykleittäin Scrumin mallin mukaisesti.

Kehittämistehtävä on aloitettu syyskuussa 2017 ja ensimmäinen versio sivustokokonaisuudesta julkaistiin joulukuussa 2017. Käyttönoton jälkeen kerätään palautetta asiakasyrityksen henkilökunnalta ja sivuston käyttäjiltä. Sivustoa jatkokehitetään kerätyn palautteen pohjalta ja projektiryhmä arvoi artefaktin onnistumista kannalta ylläpitotoimien yhteydessä työmäärän ja teknologian riittävyyden kannalta.

Työtä arvioidessa valmista artefaktia peilataan kerättyyn käyttö- ja ylläpito palautteeseen ja näiden palautteiden pohjalta arvioidaan vaatimusmäärittelyn onnistuminen. Artefaktia tullaan vertaamaan myös suoraan vaatimusmäärittelyyn ja sen toteumaan analysoiden käytössä olevia teknologioita ja niiden tarpeellisuuteen todellisen käytön yhteydessä.

Analysoivan arvioinnin lisäksi sivustot auditoidaan Googlen sivustoauditointityökalulla, mikä antaa palautetta sivuston suorituskyvystä, parhaiden käytäntöjen toteumasta sekä esittää osa-alueita, joissa sivusto voisi suoriutua paremmin. Lopullinen onnistuminen ja jatkokehitystarve arvioidaan hyöty – seuraus suhteella yhteistoimin asiakkaan kanssa. Palvelimen suorituskyvyn arvioinnissa käytetään puolestaan Azuren tuottamaa dataa pyyntöjen käsittelyn nopeudesta, määrästä ja onnistumisesta.

2 Menetelmät ja tietoperusta

Opinnäytetyössä keskeisiä käsitteitä ovat niin projektihallinnan scrum, kuin teknologiapalvelut ja viitekehykset ojelmistokehityksessä sekä kehitystyön eettiinen toteuma. Tietoperusta pyrkii kattamaan sekä projektinhallinan menetelmät, että keskeiset teknologiaratkaisut tarjoten näin tietoperustaa ja ymmärrystä asiakkaan tuotteen kehittämisestä ja vaatimusmäärittelystä.

2.1 Scrum

Asiakasyrityksen tilaaman työn kehityksessä käytetään Scrum – ohjelmistokehityksen viitekehystä. Scrum perustuu syklittäiseen työskentelymalliin, jossa **tuotoksen** (Artifact) saavuttamiseksi vaaditut toimet hajoitetaan yksittäisiksi tehtäviksi. Kun kaikki tehtävät on suoritettu voidaan tuotos todeta valmiiksi. (Swaber, 2017)

Scrum on viitekehys monimutkaisten tuotteiden kehitykseen sekä ylläpitoon. Sen ovat kehittäneet Ken Swaber ja Jeff Sutherland. Scrum auttaa käsittelemään monitasoisia, muuttuvia ongelmia, sekä tuottamaan arvokkaita tuotoksia. Scrum on kevyt rakenteinen ja helposti ymmärrettävä, mutta haasteellinen hallita. (Swaber, 2017)

Vaikka Scrum polveutuukin ketteristä menetelmistä, se ei ole yksittäinen prosessi, tekniikka tai metodi. Se on viitekehys projektin hallintaan ja useiden erilaisten, projektin kannalta oleellisten, prosessien utilisoitiin tavoitellen tuotosta. Scrumin säännöt sitovat yhteen projektin roolit, tapahtumat, tuotoksen, sekä hallitsee näiden osioiden suhteita ja keskeistä vuorovaikutusta toisiinsa. (Swaber, 2017)

Scrumin ytimessä on **empiirinen** prosessin hallinta, missä uskotaan tiedon tulevan kokemuksesta ja päätöksen teon nojaavan hankittuun tietoperustaan. Scrumin kivijalka on kolme empiirisen prosessin periaattetta: läpinäkyvyys, tarkistelu ja mukautuminen. (Swaber, 2017)

Läpinäkyvyydellä viitataan siihen, että prosessin tulee kokonaisuudessaan olla näkyvissä sekä ymmärrettävä nille, ketkä prosessiin osallisuvat. Prosessilla tulee olla kaikille jäsenillä yhteinen kieli ja terminologia lisäksi raportoidun työn tulee olla kaikkien ymmärrettävissä. (Swaber, 2017)

Tarkistelu edellyttää, että Scrumin käyttäjät aktiivisesti tarkistelevat kehittyvää artefaktia ja havaitsevat epätoivotut muutokset. Tarkistelu ei saa kuitenkaan olla niin tiheää, että se olisi häiriöksi itse prosessin edistämsielle. (Swaber, 2017)

Mukautuessa, mikäli tarkistelussa havaitaan poikkeamia prosessin työnkulusta jotka muuttaisivat halutun tuotoksen kelvottomaksi, tulee työnkulua mukauttaa siten, että prosessi tuottaa kelvollisen tuotoksen. Mukauttaminen tulee suorittaa mahdollisimman nopeasti poikkeaman minimoimiseksi. (Swaber, 2017)

Scrumin keskeisiä **arvoja** ovat sitoutuminen, rohkeus, fokus, avoimuus ja kunnioitus. Nämä arvot ovat hyvin henkilöstökeskeisiä ja tiimin näitä noudattaen perusta onnistuneeseen Scrumiin konkretisoituu sosiaalisten vaikutusten johdosta. Projekti työt ja toteutuksen metodit ovat avointa yleistä tietoa projektissa, tiimin tulee luottaa yksilöidensä itsenäiseen työkykyyn. (Swaber, 2017)

2.1.1 Scrumin roolit

Scrum tiimiin kuuluu **tuoteomistaja** (Product Owner), **kehitystiimi** (Developement Team) ja **scrummaster (**Scrum Master). Tärkeintä tiimissä on kommunikaatio, sillä Scrum nojaa jäseniensä oma-aloitteisuuteen ja työnjakokykyyn sen sijaan, että työtehtävät määrättäisiin jonkin ulkoisen tahon toimesta. Tiimille annetaan vaatimusmäärittely tuotoksesta ja tiimi itse organisoi työnsä tavoitteen saavuttamiseksi. (Swaber, 2017)

Tuoteomistaja on vastuussa tuotteen arvon maksimoinnista, tyyli on vapaa. Tuoteomistajalle kuuluu vastuu tuotteen kehitysjonon hallinnasta. Tähän lukeutuu kirjanpito kehitysjonon osioista ja niiden vaateista sekä näiden osioiden tilaaminen kehitystiimiltä loogisimmassa mahdollisessa järjestyksessä. Tuoteomistajan tulee myös huolehtia, että tuotteen kehitysjono on saatavilla kaikille ja ymmärrettävässä muodossa. Tuoteomistaja voi myös delegoida tehtäviään, mutta delegoinnista huolimatta on vastuussa tehtävien toteumasta. (Swaber, 2017)

Kehitystiimi on tasa-arvoinen asiantuntijaorganisaatio, jonka tehtävänä on suorittaa meneillään olevan **sprintille** (Sprint) valikoidusta kehitysjonosta tehtäviä. Kehitystiimi suunnittelee itse työnsä tuottamaan kehitysjonon vaatiman **sprintin tavoitteen** (Sprint Goal). Työssä luotetaan asiantuntijoiden näkemykseen toimivimmasta ratkaisusta ja tähän ei puututa, kunhan lopputulos toiminnallisuudelle vastaa haluttua. (Swaber, 2017)

Kehitystiimi jakaa tehtävät itse osaajiensa kesken tunnistaen ja hyödyntäen jokaisen yksilön vahvuuksia tavoitteen saavuttamiseksi. Tästä huolimatta kehitystiimi on ryhmänä vastuussa tuotoksestaan ja sen toimivuudesta, eikä tiimin yksittäinen kehittäjä. Tämä kannustaa ryhmää olemaan toistensa tukena. Ideaali koko kehitystiimille on kolmesta yhdeksään jäsentä. Vähemmällä henkilöstömäärällä ei välttämättä selviä jaksottaisista suuremmista työmääristä, useampi jäsen puolestaan tekee tiimistä liian hajanaisen empiirisen prosessikulun säilyttämiseksi. (Swaber, 2017)

Scrummaster on vastuussa projektin edistämisestä ja yksilöiden tukemisesta. Käytännössä tämä ilmenee ylläpitämällä scrumin arvoja ja ideologiaa sekä auttamalla prosessin osallisia ymmärtämään Scrumin toimintamallia. Mestari toimii myös kommunikoijana projektin ulkopuolelle auttaen ymmärtämään projektin kulkua ja miten scrumin ulkopuoliset jäsenet voivat edesauttaa projektin tavoitteen saavuttamista. Mestari ei siis osallistu tuotoksen luontiin, vaan ohjaa projektin toimintaa viitekehyksen pohjalta. (Swaber, 2017)

2.1.2 Scrumin tapahtumat

Ennaltanimettyjä **tapahtumia** (Event) käytetään scrumissa luomaan järjestelmällisyyttä ja vähentämään tarvetta scrumin toimintamallista poikkeaville kokouksille. Jokaisella tapahtumalla on oma ajanjaksonsa ennakkoon rajatulla enimmäiskestolla. (Swaber, 2017)

Jokaisen tapahtuman suorittamisen ohella on määrä raportoida projektin tila koko projektiryhmälle, sekä raportoida tapahtuman aikana tapahtuneista muutoksista ja lisäyksistä projektiin. Näin ylläpitäen projektin läpinäkyvyyttä ja mahdollistaen säännöllisen arvioinnin. (Swaber, 2017)

Sprintti on scrumin keskeisin tapahtuma. Se on säännöllinen ajanjakso, kestoltaan maksimissaan kuukausi, jonka aikana on määrä saada ajanjaksolle osoitetun kehitysjonon tehtävät suoritettua. Sprintin lopputuotteena tulee syntyä inkrementti projektin tuotoksen saavuttamiseksi. Sprintin työmäärän tulee olla niin pieni, että sen saavuttaminen annetulla ajalla on mahdollista. Sprintin päätyttyä alkaa välittömästi uusi sprintti. (Swaber, 2017)

Sprintin sisäisiä tapahtumia ovat **sprintin suunnittelu** (Sprint Planning), päivittäispalaveri (Daily Scrum), **sprintin katselmointi** (Sprint Review) ja **sprintin retrospektiivi** (Sprint Retrospective). Sprintin aikana ei tehdä mitään, mikä riskeeraisi sprintin tavoitteen saavuttamisen. Työnlaatutavoitetta ei lasketa, mutta tavoitteen laajuus on uudelleen neuvoteltavissa projektin omistajan ja kehitystiimin välillä sprintin edetessä, mikäli työn suorittaminen sitä vaatii. (Swaber, 2017)

Sprinttiä voi ajatella maksimissaan kuukauden mittaisena miniprojektina. Koska Sprintin tavoite on aina tuottaa inkrementti tuotokseen, on huolellinen ja kattava suunnittelu oleellinen onnistumisen kannalta. Sprintin maksimi kesto on yksi kalenterikuukausi. Mikäli Sprintin kesto ylittäisi kuukauden on todennäköistä, että projektin kannalta merkittävä osa resursseja ja tietoa muuttuu, työtehtävät ristiävät toisistaan ja sprintin tavoitteen arvokkuus projektille kyseenalaistuu. Lyhyet sprintit mahdollistavat aktiivisemman läpinäkyvyyden ja reagointikyvyn muuttuvalle projektiympäristölle. (Swaber, 2017)

Sprintin kesto on projektisuunnitelmassa ennaltamääritetty vakio, eikä sitä kuulu muuttaa. Tuoteomistaja voi kuitekin keskeyttää sprintin, mikäli sen aiheelliseksi **scrumtiimin** (Scrum Team) osalta. Todennäköisin syy sprintin keskeyttämiseen on se, että sprintin tavoite vanhentuu, eikä se enään toisi arvoa tuotokselle. Näin voi käydä esimerkiksi, mikäli projektin asiakas muuttaa vaatimumäärittelyä suunnitellulle tuotokselle. Mitä lyhyempinä sprintit pysyvät, sen todennäköisempää on, ettei niitä tarvitse keskeyttää. (Swaber, 2017)

Sprintin suunnittelu sijoittuu aikajanallisesti sprintin alkuun. Siihen osallistuu koko scrumtiimi. Kuukauden mittaisen sprintin suunnitteluun varataan maksimissaan kahdeksan tuntia aikaa. Scrummaster on vastuussa suunnittelun sprintin tavoitteen toteutumisesta. Scrummestari huolehtii, että kaikki scrumtiimin osallistuvat suunnitteluun, ymmärtävät tavoitteen ja ohjeistaa jäseniä, miten pysyä kiinni sprintin aikatavoitteessa. (Swaber, 2017)

Suunnittelun tavoitteena on arvioida ja päättää mitä tuotoksen kannalta oleellista on tuotettavissa kyseisen sprintin aikana. Sprintin tavoitteen kartoittamisen keskiössä on tuotteen kehitysjono, joka kattaa yksittäiset tehtävät tuotoksen tarvoittamiseksi. (Swaber, 2017)

Scrumtiimi suunnittelee sprintin siten, että valitut tuotteen kehitysjonon tehtävät katsoa suoritetuksi. Tuotteen kehitysjonon lisäksi suunnittelussa otetaan aina huomioon viimeisin kehitysversio ja käytettävissä olevat resurssit tulevan sprintin aikana. Sprintin työmäärä arvioidaan perustaen kehitystiimin tehokkuuteen aikaisemman Sprintin aikana. Näin kehitystiimille ei kasaudu liikaa tehtäviä. Kehitystiimi itse valitsee, mitä tehtäviä kehitysjonosta he uskovat pystyvänsä suorittamaan annetussa aikaikkunassa. Suunnittelussa sovittuja sprintin tavoitteita nimitetään sprintin tavoitteeksi. (Swaber, 2017)

Sprintin tavoitteen muodostuttua kehitystiimi sopii yhdessä, kuinka sprintin tavoitteeksi valikoidut ominaisuudet toteutetaan siten, että ne voidaan katsoa tehdyksi. Tuotteen kehitysjonosta sprintille valitut tehtävät ja niiden toteutus yhdessä muodostavat sprintin tavoitteen. (Swaber, 2017)

Suunnitelun päätteeksi kehitystiimin tulee kyetä esittämään tuoteomistajalle ja scrummasterille miten se aikoo saavuttaa sprintille asetetut tavoitteet konkreettisesti. (Swaber, 2017)

Päivttäispalaveri on kehitystiimille toistuva tapahtuma, kestoltaan noin 15 minuuttia, jossa sovitaan sprintin aikana suoritettavat tehtävät seuraavan 24 tunnin ajalle. Tämä auttaa kehitystiimiä pysymään tietoisena, missä tilanteessa tuotteen kehitys menee, mitä on tehty ja mitä tulee tehdä. Näin edistetään myös projektin läpinäkyvyyttä. Päivittäispalaveri pidetään

aina samaan aikaan samassa paikassa, esimerkiksi toimistolla kello kymmenen aamukahvin yhteydessä taukotilassa. Näin säilytetään toistuvuus, mutta rento ympäristö ja rutiinin omainen läpikäynti, jotta kehitystiimin harmonia säilyy. (Swaber, 2017)

Vaikka päivittäispalaveri onkin vapaamuotoisempi tapahtuma, tulee sen aikana jokaisen kehitystiimin jäsenen kertoa mitä hän on projektin edistämiseksi edellisen vuorokauden aikana tehnyt. Mitä jäsen aikoo tänään tehtä sprintin tavoitteen saavuttamiseksi ja onko jäsen havainnut mitään, mikä estäisi kehitystiimiä saavuttamasta sprintin tavoitetta. (Swaber, 2017)

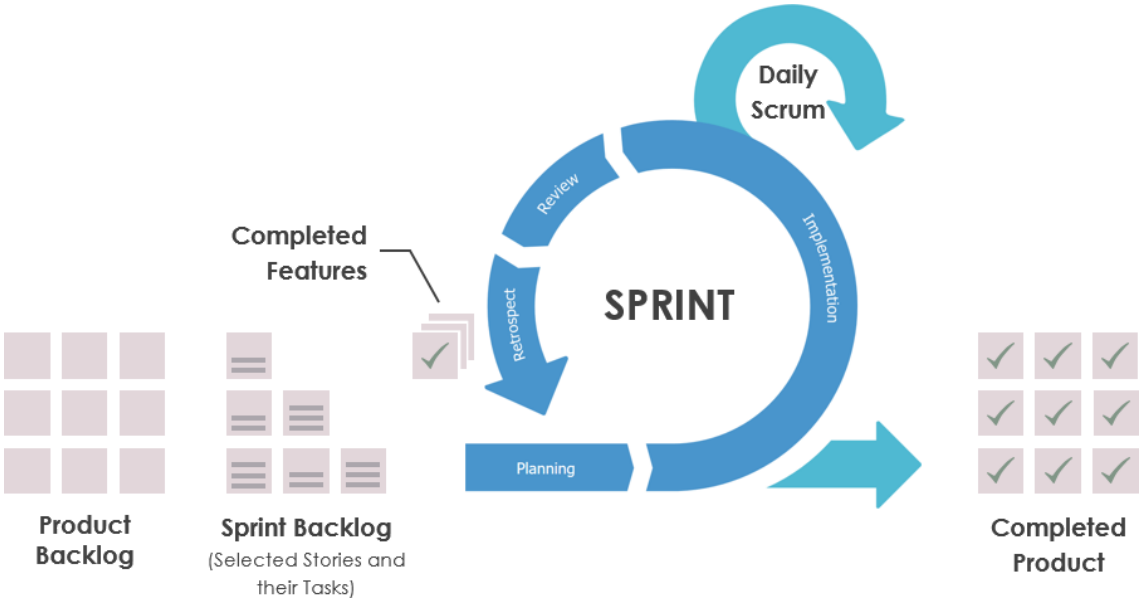
Päivittäispalaveria voi useinkin seurata virallisempi palaveri projektiin liittyen. Päivittäispalaveri on kehitystiimin sisäinen palaveri eikä tuoteomistajan tai scummasterin kuulu siihen osallistua. Mikäli päivittäispalaverin tapahtuessa mukana on muita projektiin osapuolia, huolehtii scrummaster siitä, etteivät muut osapuolet häiritse kehitystiimin palaveria. (Swaber, 2017)

Päivittäispalaverin tavoite on tuottaa harmonia ja yhteisymmärrys kehitystiimin keskuuteen, sekä poistaa tarve satunnaisille yksittäisten kehitystiimin jäsenten väliselle palavereille. (Swaber, 2017)

Sprintin katselmointi (Sprint Review) sijoittuu tapahtumana sprintin loppuun. Tällöin scrumtiimi, sekä muut projektin osalliset kuten asiakas, kokoontuvat käymään läpi sprintin inkrementtiä tuotokselle, sekä muuttamaan tuotteen kehitysjonoa, mikäli se katsotaan aiheelliseksi. Muutosten lisäksi tapahtuman osalliset päättävät, mikä on seuraavana eniten arvoa tuotokselle luova inkrementti. Sprintin katselmointi ei ole tilannepalaveri, vaan informaatiivinen tapahtuma kaikille projektin osallisille projektin tilanteesta. (Swaber, 2017)

Sprintin retrospektiivi antaa scrumtiimille mahdollisuuden tutkia itseään ja löytää parannettavaa toiminnassaan seuraavalle sprintille. Tapahtumana sprintin retrospektiivi ajoittuu kahden sprintin vaihteesen sprintin katselmoinnin jälkeen ennen uuden sprintin aloittamista. Tapahtuman maksimikesto on kolme tuntia. (Swaber, 2017)

Scrummaster on vastuussa tapahtumasta ja jäsenten osallistumisesta. Tapahtuman tulee olla positiivinen ja tuottava, tässä ei siis kartoiteta menneen sprintin virheitä, vaan se kuuluu sprintin katselmoinnissa läpikäytäviin asioihin. (Swaber, 2017)



Kuva 1: Scrumin vaiheet (Visual Paradigm, 2019)

2.1.3 Scrumin esineet

Scrumin määrittelemä **tuotos** (Artifact) on lopputuote, jonka pohjatieto on koko tiimille avointa ja auttaen näin luomaan yhteisymmärryksen tuotoksesta. Tuotos on siis se, minkä scrumtiimi yhteisymmärrtyksessä kokee tuotokseksi. (Swaber, 2017)

Tuotteen kehitysjono (Product Backlog) on jäsennetty lista kaikesta, mitä tiedetään tarvittavan tuotoksen saavuttamiseksi. Tuoteomistaja on vastuussa tuotteen kehitysjonon sisällöstä, muokkauksesta sekä jäsennyksestä. Projektin alkaessa tuotteen kehitysjono kattaa ns. rautalankamallin sovelluksesta: asiat mitä tiedetään vähintäänkin tarvittavan. Lisäyksiä listaan syntyy projektin etenemisen mukaan. (Swaber, 2017)

Tuotteen kehitysjonon tehtäviä ovat ominaisuudet, toiminnallisuudet, vaatimukset, kehityskohteet ja korjaukset, jotka vaikuttavat tuotoksen valmistumiseen. Listan tehtäville suositellaan myös kirjoittamaan pieni kuvaus tehtävän sisällöstä, jotta ymmärrettävyys säilyy. Tehtävän kuvausta käytetään myös todentamaan tehtävä valmiiksi. (Swaber, 2017)

Sprintin kehitysjono (Sprint backlog) on kooste kyseiselle sprintille valituista tuotteen kehitysjonon tehtävistä, sekä suunnitelmasta näiden tehtävien toteutukselle. Sprintin kehitysjono konkretisoi vaaditun työn sprintin tavoitteiden täyttymiseksi. Koska jokaisen sprintin tavoite on edistää projektia, tulle jokaisen sprintin kehitysjonossa esiintyä ainakin yksi korkean priotiteetin tehtävä. (Swaber, 2017)

2.2 Etiikka

Projektin etiikka on aina oleellista huomioida. Tietotekniikka-alalla on kehittäjän käsissä paljon tietoa ja resursseja, jotka ovat asiakkaan yksityisomaisuutta. Hyvän asiakaskokemuksen nimissä asiakkaan tulee tuntea olonsa turvalliseksi ja luottavaiseksi luovuttaa näitä tietoja ja oikeuksia kehittäjälle. Asiantuntijana kehittäjällä on myös eettinen vastuu asiakkaalle aiheutettavista kuluista työskentelyyn vaadittujen resurssien myötä, sekä toimivan tuotteen toimittamisesta.

Illinoin- sekä SANS:in teknoligiaininstituutit ovat tuottaneet etiikan ohjenuorat IT alan käytännöistä. Näihin perehtyneenä olen vapaasti yhdistellyt ja suomentanut lähteitä tuottaakseni allaolevan ohjenuoran projektin eettiseksi pohjaksi. Tiivistelmä ei ole täydellinen mutta pyrin tuomaan esiin painavimmat aspektit.

IT alan etiikka voidaan hajoittaa oman ammattitaidon ylläpitoon ja osaamisen jakamiseen, alan parhaiden käytäntöjen toteutukseen ja noudatukseen sekä ammatin, aseman ja artefaktin kunnioitukseen.

Omaa ammattitaitoa tulee ylläpitää. Omaa sekä kollegojen työtä kohtaan tulee olla kriittinen ja tunnistaa ongelmat, löytää kehitettävää ja kehittää omaa asiantuntijuutta tämän myötä. Omaa ammattitaitoa tulee kartuttaa ja hioa jatkuvasti. Oma osaaminen tulee jakaa kollegoiden kanssa, mikäli siinä on potentiaalia tuottaa toimivampi tuote tai se kehittää työyhteistön työskentelytapoja. (SANS teknologiaininstituutti, 2004), (Illinoin teknologiainstituutti, 2011).

Asiantuntijan tulee seurata alan kehitystä ja tunnistaa omaa työtä varten merkittävien uusien teknologioiden hyöty, perehtyä näiden käyttöönottoon ja edesajaa tehokkaammaksi todetun käytännön käyttöönottoa työyhteisössä. (SANS teknologiaininstituutti, 2004), (Illinoin teknologiainstituutti, 2011).

Asiantuntija on vastuussa siitä, että toteutetun ohjelmiston laatu vastaa aina asiantuntijan parasta osaamista ja parhaiden käytäntöjen mukaisia metodeja: tärkeimpinä mainittakoon mahdollisimman selkeästi tuotettu sisältö, mahdollisimman kompaktisti toteutettu sisältö, koodin monikäyttöisyys sekä suorituskyvyn takaaminen. (SANS teknologiaininstituutti, 2004), (Illinoin teknologiainstituutti, 2011).

Asiantuntijan tulee kunnioittaa oman asemansa suomia käyttöoikeuksia. Järjestelmästä haetun tiedon tulee olla työn suorituksen kannalta oleellista ja edesauttaa työtehtävän etenemistä. Tietoa ei saa koskaan hakea henkilökohtaisessa tarkoituksessa. Mitään tekinjäinoikeuksiin tai yhtiön salaiseksi kehittämää toiminallisuutta ei saa jakaa eteenpäin minkäänlaisissa

tarkoituksissa, poikkeuksena yhtiön oma toive tämän tapahtumisesta. (SANS teknologiainstituutti, 2004), (Illinoin teknologiainstituutti, 2011).

Asiantuntijan tulee kunnioittaa omaa ammattiaan. Asiantuntija ei saa väärinkäyttää tietoasemaansa teeskennellen olemattomia tarpeita päättävälle henkilölle omien tarkoitusperien tai viihteen saavuttamiseksi. Kaikkien asiantuntijan esittämien tarpeiden tulee edesauttaa ja hyödyttää asiakasta. (SANS teknologiainstituutti, 2004), (Illinoin teknologiainstituutti, 2011).

Asiantuntijan tulee kunnioittaa itsentä tuottamaa, sekä aiemmin tuotettuja tuotoksia tai niiden inkrementtejä. Tuotteita ei saa tarkoituksen mukaisesti vioittaa tai tuhota ja niiden ylläpitoon tulee perehtyä. Kehittäessä tai ylläpitäessä tuotoksia tulee tuottaa toimivia pitkäkestoisia ratkaisuja asiantuntijan oman parhaan ammattitaidon mukaan. (SANS teknologiainstituutti, 2004), (Illinoin teknologiainstituutti, 2011).

2.3 Projektin työkalut & ympäristöt

Valmis sivustokokonaisuus on kolmitasoinen toteutus: vaaditaan palvelinympäristö, backend sekä frontend. Palvelinmpäristöllä tarkoitetaan alustaa, joka tarjoaa palvelut/toiminnallisuudet sivuston hostaamiselle, tietokannan käyttöön sekä työkaluja ja/tai viitekehyspalveluita palvelimen ohjelmointiin. Backend tarkoittaa palvelimen käyttämää koodia, jolla palvelin käsittelee saamiaan kutsuja käyttäjän selaimista. Frontend kattaa käyttäjälle näkyvän www-sivun ja sen kautta käyttäjälle paljastetut toiminnot.

2.3.1 Ympäristöt

Web sivustoa/-sovellusta kehittäessä yleisin käytäntö on tuottaa sivusto kehitysympäristössä ja tuotteen valmistuttua kokonaisuus siirretään toimintaan tuotantoympäristössä.

Projektissa kehitysympäristönä käytetään lokaalisti käynnistettävää kehityspalvelinta joka suorittaa samaa koodia, mitä tuotantoympäristössä tullaan käyttämään. Kehitysympäristössä dokumentinhallintaan ja palveluiden ajoon käytetään Microsoftin kehittämää avoimen lähdekoodin projektin tuotetta Visual Studio Code. Tämä työkalu mahdollistaa niin frontend- kuin backend koodien samanaikaisen näytön, muokkauksen, käytön ja monitoroinnin käytön aikana.

Tuotantoympäristönä projektissa käytetään Microsoftin pilvipalvelun tarjoamaa Azure -pilvipalvelua, joka on kattava palvelinympäristö tarjoten niin puhdasta palvelintilaa, kuin useita suosittuja palvelin- ja tietokantaviitekehyksiä tukevia palveluita.

2.3.2 Microsoft Azure

Microsoft Azure on palvelukokonaisuutena niin kattava, että opinnäytetyössä joudutaan rajaamaan sen tarjoamat palvelut ainoastaan projektin kannalta oleellisiin palveluihin.

Azuren **App service** -palvelu tarjoaa mahdollisuuden julkaista oma web -sovellus käyttäen erilaisia yleisimpiä palvelinviitekehyksiä/ohjelmia, mukaan lukien tässä projektissa alustana toimiva Node.js.

App service on palvelinkonetta ohjaileva käyttöliittymä Azuressa. Erona puhtaaseen palvelinkoneeseen käyttötarkoitukseen soveltuvalla käyttöjärjestelmällä, App service on jo itsessään mahdollisimman varmis palvelinohjelmisto -palvelu, jolle kehittäjän olisi tarkoitus luovuttaa omasta palvelinkoodista vain sovelluksen toiminnallisuuskia ajavat koodi. Palvelimille tyypilliset toiminnallisuudet on tarkoitettu App servicessa käytettäävksi Azuren käyttöliittymästä ja antaa microsoftin palvelun toteuttaa itseään esimerkiksi sovelluksen skaalaamisessa suurille käyttäjämäärille. (Microsoft, 2019)

Tuotteen julkaisu **AppService** -palveluun mahdollistettu usealla tavalla. Tässä projektissa tuotoksen koodi kokonaisuudessaan julkaistaan App Servicen käyttöön suoraan versiohallinnan kautta. Palvelu tarjoaa kattavasti statistiikkaa sivuston kävijöistä, suorituskyvystä ja dataliikenteestä mahdollistaen tuotannossa olevan sivuston monitoroinnin.

Microsoft tukee startup yrityksiä monin tarjoamalla kuukausittaista käyttövaraa Azure -palveluihin myöntämänsä **Biz Spark** -stipendin muodossa. Näillä varoilla statupit voivat ottaa käyttöön omat sivustonsa, palvelunsa, tietokantansa tai minkä tahansa yritykselle tarpeelliseksi todetun Azuren tarjoaman palvelun. Biz spark stipendi kattaa yleisimmät kulut ensimmäiseen 130$/kk saakka. Tästä ylittyvä summa veloiteaan yrityksen ilmoittamalta maksukortilta. Veloitus on myös mahdollista lukita azuren myöntämän käyttövaran rajoihin jolloin palvelut sammuvat, mikäli palvelukustannukset per kuukausi tulevat täyteen. (Microsoft, 2019)

Azure Cosmos DB on Azuren tarjoama tietokantapalvelu. Cosmos DB on poikkeuksellinen palvelu tietokannoille, sillä se tarjoaa käyttäjälle itselleen mahdollisuuden päättää viitekehyksen, jolla tietokanta operoi. Valittavissa on niin SQL kuin noSQL -viitekehyksiä, mukaan lukien tässä tuotoksessa käytetty noSQL viitekehys MongoDB. (Microsoft, 2019)

CosmosDB:stä on mahdollista jakaa tietoa rajattomasti useisiin lähteisiin. Reunaehtona on, että tietokantaan yhdistävän sivuston on toimittava suojatun yhteyden (SSL) kautta, tietoturvan takaamiseksi. Lisäksi kantaan yhdistävä palvelin tulee lisätä Azuressa sallittujen yhteyksien listaan. Tietokannan käytön hinnoittelu on suhteessa dataliikenteen määrään.

App service -palvelun lisätuotteena Azure tarjoaa **App Service Certificate** -palvelua, missä Microsoft välittää GoDaddy – palvelun tarjoaman SSL -sertifikaatin ja mahdollistaa sertifikaatin mutkattoman asennuksen käytössä oleville sertifikaatin piirissä oleville AppServiceille. (Microsoft Azure, 2018)

SSL -sertifikaatti on tietoturvanormi, joka on nykypäivänä enemmänkin oletus kuin suositus. Tuttavallisemmin SSL -yhteys tarkoittaa https:// -alkuista www-osoitetta. Kun sivusto käyttää SLL -yhteyttä, suojataan palvelimen ja selaimen välinen dataliikenne kryptaamalla liikenteen sisältö lukemattomaan muotoon käyttämällä sertifikaattiin sisältyvää Public -avainta ja tulkitsemalla liikenteen sisältö palvelimella käyttämällä sertifikaatin Private -avainta. Avaimet ovaat uniikkeja eikä mikään muun lisenssin avainta voida käyttää tulkitsemaan toisen lisenssin alaista liikennettä. Ainoa avain kryptauksen avaamiseksi on kyseisen lisenssin Private -avain. (Microsoft, 2018)

2.3.3 Back-End

Tuotteen Back-End rakenteeseen kuuluvat palvelin, jota Azuren App service palvelee, sekä tietokannan viitekehys, joka on asennettu Azure Cosmos DB -ympäristöön. Palvelin toimii välikätenä tuotantoympäristön, tietokannan ja Front endin välillä.

Palvelimena projektissa käytetään Node.js -viitekehystä, mikä on Javascriptilla ohjelmoitava runtime-hakemisto, suunniteltu erityisesti monitasoisten verkkosivustojen / -sovellusten tuotantoon. Node.js:n suurin etu on sen asynkrooninen toteutus: Missä perinteiset palvelimet pystyvät käsittelemään pyynnön kerrallaan ja aloittamaan uuden, kuin edellinen on käsitelty, pystyy Node.js ottamaan uusia pyyntöjä vastaan odottamatta edellisen valmistumista. (Node.js Foundation, 2019)

Palvelimen toimintalogiikan puolesta se on ideaali startup -yrityksen sivustolle, minne tavoitellaan mahdollisimman paljon kävijöitä ja minne kävijät tulevat suurina aaltoina yleensä pitchaus tilaisuuksien saattelemana.

Palvelimen yksinkertaistamiseksi sen ohjelmointiin käytetään express.js -viitekehystä, joka on minimalistinen ja kevyt funktiokirjasto yleisimmistä palvelimen toiminnallisuuksista mahdollistaen mahdollisimman kattavan rajapinnan sivuston Front-Endille. (Node.js Foundation, 2019)

Verkkosivustojen tietokanta kattaa sivuston dynaamisen sisällön, kuten blogi postaukset ja uutislinkit. Sivustojen tiedot on eristeetty omaan tietokantaansa sovelluksen henkilörekisterin turvallisuuden nimissä.

MongoDB on noSQL -tietokannan viitekehys, joka käsittele dataa JSON -formaatissa. Tämä mahdollistaa jatkuvat muutokset tietomalliin vaikuttamatta aiemmin tallennettuun tietoon. Node.js:n kanssa JSON -tyyppiset noSQL tietokannat ovat täydellinen yhdistelmä, sillä JSON on tiedon säilöntään jalostettu javascript -formaatti. Näin javascriptia suorittavan Node.js:n ja JSON:a käsittelevän MongoDB:n välillä ei tarvitse tulkita ja kääntää tietoa toistensa ymmärtämään muotoon, toisin kun perinteisten SQL tietokantojen kanssa. (Wikipedia, 2019)

Vaadittu HTML -koodi sivuston esitykseen selaimessa tuotetaan EJS -viitekehyksellä. Kyseessä on dokumenttigeneraattori, jonka avulla palvelin lukee sivustoa varten tuotettuja sapluunoita ja parsii näiden mukaisen HTML dokumentin. EJS mahdollistaa JSON -datan parsimisen suoraan sapluunaan, näin samaa sapluunaa voi käyttää useaan tarkoitukseen vaihtamalla vain luettevan teidon. Tietojen tulee jakaa sama rakenne tälläisissä tapauksissa. (EJS, 2019)

2.3.4 Front-End

Web-kehityksessä Frontend -toteutus koostuu pohjimmiltaan kolmesta kielestä: HTML, CSS ja Javascript. Näille kielille on merkittävän paljon erilaisia viitekehyksiä helpottaamaan ja automatisoimaan tavanomaisimpia toiminnallisuuskia valmiin lopputuloksen tuottamiseksi. Massiivisesta viitekehystarjonnasta huolimatta web-sivu koostuu loppupeleissä aina näiden kolmen kielen voimin.

Hypertext Markup Language (HTML) on standardi kieli websivustoille. HTML -dokumentti määrittää sivuston rakenteen semanttisesti: dokumentti esitetetään selaimessa siinä määritetyn rakenteen mukaisesti. HTML dokumentti koostuu erilaisista sivustorakenteelle osoitetuista elementeistä, kuten kuvista, lomakkeista, linkeistä ja lausekkeista. (MDN web docs, 2019)

HTML doumentti on itsessään standardin mukainen staattisen esitys dokumentin sisällöstä. Dokumentin ulkoasua voidaan muuttaa CSS -kielellä ja sen toiminnallisuutta voidaan dynamisoida upottamalla dokumenttiin Javascriptillä ohjelmoituja toiminnallisuuksia. (MDN web docs, 2019)

Cascading Style Sheets (CSS) on tyylittelykieli dokumenttien ulkoasun määrittämiseen. Vaikka CSS on eniten käytetty HTML dokumenttien ulkoasun määrityksissä, voidaa sitä käyttää minkä tahansa XML pohjaisen dokumentin tyylittelyyn. CSS käytetään niin visuaalisesti vetoaviin verkkosivustojen ulkoasun toteutukseen, kuin erilaisten web- ja mobiilisovellusten ulkoasun määrittelyyn. (MDN web docs, 2019)

CSS on suunniteltu mahdollistamaan dokumentin tyylittely erillään itse dokumentista. Sillä voidaan erottaa mikä tahansa tyyli dokumenttirakenteesta ja luoda sille sääntö, jota sääntöön

kohdistuvat elementit noudattavat oletus säännön sijasta. Kun tyylit on eriytetty HTML -dokumentitsta erilliseen CSS tiedostoon voivat useat sivut käyttää samaa tiedostoa tyylittelyn lähteenä. (MDN web docs, 2019)

Javascript (JS) on ohjelmointikieli ensisijaisesti dynaamisten web sivustojen, selaimessa toimivien sovellusten tai pelien tuottamiseksi. Valtaosa web sivuista käyttää Javascriptiä ja kaikki modernit selaimet tukevat tätä ohjelmointikieltä natiivisti selaimeen rakennetulla javascript -moottorilla. Vaikka kaikkien modernien selainten mottorit tukevat javasciptiä, eivät kaikki tue uusinta versiota tai kaikkia ominaisuuksia. (MDN web docs, 2019)

Javascript mahdollistaa niin tapahtuma-, funktio-, objekti- ja prototyyppi pohjaisen ohjelmoinnin. Kieli tarjoaa rajapinnan lukuisien muuttujatyyppien käsittelyyn ja HTML dokumentin manipulointiin. (MDN web docs, 2019)

Vaikka Javascript on alun perin suunniteltu selaimen ajettavaksi websivustoilla on sen käyttö laajentunut palvelinten, tietokantojen ja täysin selaimesta riippumattomien ohjelmien tuotantoon runtime-hakemistojen avulla, jotka mahdollistavat tietokoneelle Javascriptin käytön selaimen ulkopuolella. (MDN web docs, 2019)

3 Vaatimusmäärittely ja tavoitteet

Asiakkaan projekti tuotetaan Scrum -ohjelmistotuotannon viitekehyksellä. Projektin käynnistyessä koostetaan tuotokselle kehitysjono tuotteen vaatimusmäärittelyn pohjalta, näin selventäen tuotoksen saavuttamiseen vaaditut tehtävät.

Tuotoksen kehitysjonoa varten on asiakasta haastateltu heidän sivustoa koskevien vaatimustensa kartoittamiseksi. Haastattelun perusteella suunnitellaan ja esitetään asiakkaalle parhaiten tarpeen täyttävä teknologiakokonaisuus kokonaisuus ja tuotantoympäristö sekä muut valmiin tuotteen saavuttamiseksi vaaditut edellytykset.

3.1 Asiakkaan vaatimukset

Asiakas tarvitsee yritykselle kotisivut suomeksi ja englanniksi. Näille asikas on jo ostanut domainit huoleti.fi ja huoleti.com. Kotisivuilla asiakas tahtoo esittää tietoa sovelluksesta ja yrityksestä itsestään. Asiakas myös painottaa visuaalisuuden tärkeyttä, sivuston tulee olla käytettävyydeltään sovelluksen tuntuinen ja olla teemassa yrityksen muun materiaalin kanssa. Sivuston tulisi myös avautua selaimessa nopeasti ja kyetä vastaamaan nopeasti kävijäpiikkienkin aikana.

Etusivulla esitetään yrityksen missio ja arvot: mitä sovellus tarjoaa. Sisältö on vakio eikä asiakas itse tarvitse kanavaa muokata tätä. Asiakas luovuttaa tekstisisällön tälle sivustolle erillisessä tiedostossa.

Sivustolla tulee olla jatkuvasti esillä yrityksen yhteystiedot sekä linkit sovelluskauppoihin ja tuotevideoon. Yhteystiedoissa tarjotaan niin yrityksen osoite, kuin CEOn yhteystiedot sekä yrityksen virallinen yhteydenotto osoite ja linkit käytössäoleviin somekanaviin. Asiakas toimittaa tarkemman sisällön näihin erillisessä dokumentissa.

Sivustolle tehdään **Tuki** -alasivu myös sovelluksen käyttö- ja käyttöönottotuelle. Sivun sisältö tulee olemaan vakio eikä asiakas tarvitse mahdollisuutta muokata sisältöä. Asiakas luovuttaa tarkemman tekstisisällön kehityksen haltuun. Sivulla tulee esittää askeleet sovelluksen käyttöönotolle, listata usein kysytyt kysymykset ja näiden vastaukset sekä tarjota käyttäjälle mahdollisuus hakea oman aiheensa kysymyksiä. Lisäksi tukisivulla tulee olla linkki Henkilötietolain mukaiseen rekisteriselosteen ja mahdollisuus ladata printattava versio sovelluksen käyttöönottoohjeesta.

Huoleti -sovellus tarjoitaa vertaistukiverkostoa moniin huoliin. Sovelluksen sisällä huolet, kuten erilaiset sairaudet ja elämäntilanteet ovat jaoteltu 20 pääryhmään, joilla on omia alaryhmiään. Nämä **huolet** halutaan esittää omana ala-sivunaan auttamaan käyttäjiä löytämään oman vertaistukiryhmänsä jonne hakeutua sovellusta käyttöönottaessa. Tiedot tulevat olemaan vakioita eikä asiakas tarvitse kanavaa näiden tietojen muuttamiseen.

Koska kyseessä on henkisesti raskaita asioita käsittelevä verkosto, ovat käyttäytymis- sekä käyttösäännöt oleellisia. Sivustolle halutaan **Yhteisönormit** -alasivu. Sivulla esitetään yleistä termistöä ja käyttäytymissääntöjä sovelluksen käyttöön, avataan sovelluksen toimintaa ja sen yksityisyyden suojaa sekä tarjotaan tietoa vapaaehtoisena toimimisesta.

Huoleti haluaa sivuilleen kanavan bloggaamiseen, sekä itseään koskevien uutisten esille nostamiseen. Sivustolle tehdään **blogi**-, sekä **uutiset & some** -alasivut, jota kautta yritys voi suorittaa tiedonvälitystään. Lisäksi **uutiset & some -**sivulle lisätään upotukset yrityksen käytössäolevien sosiaalisen median rajapinnoille. Yritys kokee jatkuvan tiedonvälityksen olevan olellinen osa sijoittajien tavoitteluun ja kiinnostuksen ylläpitoon. Blogisisältöä ja nostettavia uutisartikkeleja on sekä suomeksi että englanniksi, suomenkielisellä sivulla tahdotaan näyttää sekä suomen-, että englanninkielinen sisältö ja englanninkielisellä sivulla näytetään ainoastaan englanninkielinen sisältö.

3.2 Asiakkaan vaatimusten synnyttämät vaatimukset tuotoksen rakenteelle

Koska sivustokokonaisuuden pääasiallinen tarkoitus on informoida kävijää yrityksen ja sen tarjoaman sovelluksen eri osa-aluieista eikä sivustolla tarjota varsinaista sovellustoimintaa, on sivuston Frontend toteutus toiminnalisuuksiltaan yksinkertainen. Tämä mahdollistaa sivuston koostamisen palvelimella, jolloin käyttäjälle saadaan toimitettua mahdollisimman valmis dokumentti ja näin tarjottua nopea käyttökokemus.

Node.js:n tapahtumapohjainen reagointi mahdollistaa kutsujen vastaantoton, käsittelyn sekä vastauksen reaaliajassa monien samanaikaisesten käyttäjien kesken, missä perinteiset palvelintekniikat käsittelevät yhtä kutsua kunnes sen vastaus on valmis ja aloittavat vasta vastauken jälkeen seuraavan kutsuun käsittelyn. Tälläisella palvelintekniikalla saavutetaan mahdollisuus monien samanaikaisten kävijöiden sivuston käytölle sen vaikuttamatta suorituskykyyn.

Tuottamalla palvelimen Node.js -viitekehyksellä saadaan käyttoon express.js ja ejs -kirjastot. Express.js on funktiokirjasto palvelintoiminnoille, mikä mahdollistaa suoraviivaisen ja selkeän palvelinkoodin tuotannon sekä keskittymisen aisiakkaan vaatimiin toiminnallisuuksiin yleisten toiminnallisuuksien tuottamisen sijasta. Ejs -kirjasto on puolestaan dokumenttigeneraattori, jossa ilmoitetaan dokumentin erinäiset osat, rakenne, sisällön toistosäännöt sekä sivustoon

upotettava data palvelimelta. Palvelin käsittelee dokumentin parsimisen dokumenttimoottorille annettujen sääntöjen mukaan ja tuottaa selaimelle kokonaisen HTML -dokumentin näytettäväksi, samalla minimoiden käyttäjän puolen selaimen työt.

Koska sivuston dynaaminen sisältö halutaan näyttää osittain englanninkielisellä sivstolla ja kaikki sisältö suomenkielisellä sivustolla, on kannattavinta sijoittaa kaikki data yhteiseen tietokantaan. Sisältö voidaan hakea ja erotella kieli -parametrilla palvelimella pyyntöä käsitellessä ja kyselyyn sopiva data sijoittaa osaksi dokumenttigeneraattorin vastausta palvelinkutsuun. Blogien ja uutislinkkien hallintaa varten tuotetaan erillinen ylläpitokäyttöliittymä samoilla työkaluilla joita sivusto jo käyttää, josta kutsutaan ylläpitoon vaadittuja muokkaus toimintoja.

Nopea suoriutuminen latausajoista ja sivun latauksesta saavutetaan tuottamalla mahdollisimman kevyt kokonaisuus palvelimen lähetettäväksi sivustoa kutsuvalle selaimelle. Huomioiden, että erilaiset ulkoasu -kirjastot,kuten Material Design Lite tai Bootstrap, ovat kooltaan useita kilotavuja on nopeuden sekä sivuston databudjetin kannalta taloudellisempaa tuottaa kaikki tyylit ja toiminnallisuudet ilman kirjastoja. Sivusto tulee nojaamaan myös paljon visuaaliseen sisältöön. Kaikki sivustolle lisättävät valokuvat web-optimoidaan mahdollisiman nopean latauksen takaamiseksi.

3.3 Rakenteen synnyttämät toteutussympäristö vaateet

Asiakkaan palvelin tulee olemaan perustoiminnallisuudet kattava Node.js pohjainen sovellus tietokanta yhteydellä. Sovelluksen kannalta on siis kannattavampaa asentaa kokonaisuus enemmän kontrolloituun, mutta enemmän automatisoituun ympäristöön.

Yrityksellä on käytössään Microsoftin startup yrityksille myöntämä Biz Spark -stipendi yhtiön Azure -palveluihin, jolla yritys saa tuetusti käyttönsä palvelinympäristön verkkosovelluksen hallinalle ja palvelulle. Vaihtoehtoja on perustaa palvelinasennus etätyöpöytä -ympäristöön tai käyttöö Azuren automatisoidumpaa App service -palvelua.

App Service mahdollistaa mm. SLL -suojauksen asettamisen ja domainien hallinnan sivustolle suoraan Azuren käyttöliittymästä vähentäen näin palvelimelle tuotettavan koodin määrää ja ylläpidon vaivallisuutta. Yleisten ylläpitotehtävien jättö Azuren käyttöympäristöön Node.js -palvelimen sijasta helpottaa myös ylläpitotyötä eikä vaadi ylläpitäjältä koodaustaitoa käytetystä viitekehyksestä, vaan ainoastaan tuntemusta Azuren ympäristöstä.

4 Projektin toteutussuunitelma

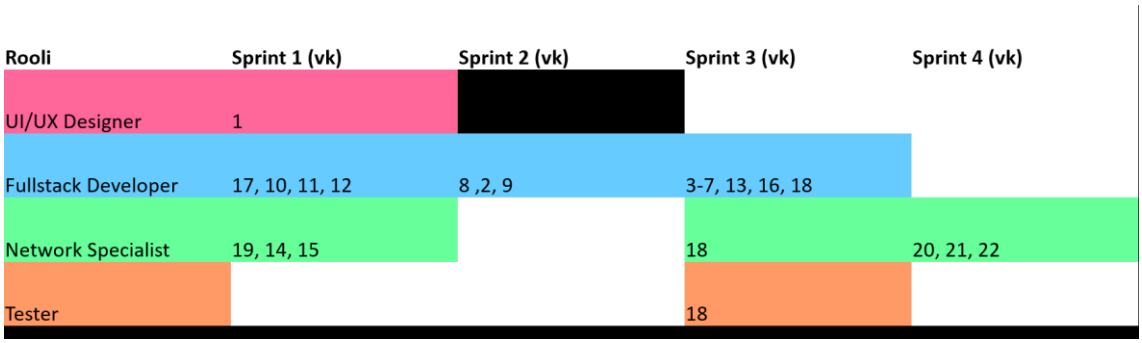
Projektissa toteutetaan yhteensä kolme sivustoa. Asiakasyritykselle kotisivut suomeksi ja englanniksi, sekä ylläpitokäyttöliittymä asiakkaalle sivustojen dynaamisen sisällön hallinnalle. Projekti hallinnoidaan Scrum -ohjelmistotuotannon viitekehyksellä jonka pohjalta on sunniteltu aikataulu ja koostesttu projektiorganisaatio. Projektin toteutukseen valitut teknologiat on suunniteltu asiakasta haastattelemalla ja sitten haastattelua analysoimalla sekä valintojen synnyttämien resurssien pohjalta.

Sivustot isännöidään Microsoftin Azure -palveluympäristöpalvelusta, jonne sivuston palvelin koodataan Node.js -viitekehyksen ja sen lisämoduulien avulla. Sivuston käyttöliittymä tuotetaan natiiveilla kielillä HTML, JS ja CSS ja HTML -dokumentti koostetaan palvelimella minimoidaksemme käyttäjän selaimen suorituskykyvaatimuksen.

4.1 Aikataulu & työmääräarvio

Projekti toteutetaan Scrum -projektihallintaviitekehyksen mukaisesti sykleissä eli sprintteinä. Sprintin kestoksi on sovittu projektiosallisten kesken viikko jonka aikana maksimityömäärä henkilöä kohden on 38h. Osallisten tulee olla tavoitettavissa klo 8-16 välillä.

Projekti suunnitellaan ja valmistellaan tuotantovalmiuteen vuoden 2017 loppuun mennessä. Kehitystyö toteutetaan vuoden 2018 tammikuun aikana. Työaikamäärällisesti projektin on arvioitu kehitystiimin osalta onnistuvan neljässä sprintissä. Tuoteomistaja on kehitystiimin kanssa yhteistyössä tuottanut suunnitteluvaiheessa tuotoksen kehitysjonon.



Kuva 2: Tuotoksen kehitysjonon tehtävien jakautuminen sprinteittäin

Sprinttien aikana järjestetään Scrumin käytännön mukaisesti sprintin suunnittelu ja -tavoite palaverit sekä loppu reflektointi. Projektin edistyminen käydään läpi päivittäispalavereissa. Tehtävien toisistaan riippuvuus on esitetty tuotoksen kehitysjonossa, josta valitaan tehtävät suoritettavan sprintin kehitysjonoon.

4.2 Jäsenet & roolit

Poiketen suositellusta scrumin miehityksestä, projekti on yksittäisen tekijän työ. Projektin tekijän vastuulla on useita rooleja projektin toteutuksessa. Projektiraportissa sekä -suunnitelmassa tullaan viittamaan monissamäärin rooleihin jäsenten sijasta, jotta olisi mahdollista tarjota selkeämpi käsitys projektin rakenteesta.

Tuoteomistaja ylläpitää ja hallinnoi tuotoksen kehitysjonoa, sekä tilaa toiminnallisuuksia kehitystiimiltä. Tuoteomistajan tärkein tehtävä on pysyä kartalla projektin etenemisestä kokonaiskuvallisesti, delegoida ja aikatauluttaa tehtäviä sekä kommunikoida scrummasterin kanssa, jotta tämä voi tukea Scrumin ideologian toteutumaa projektissa.

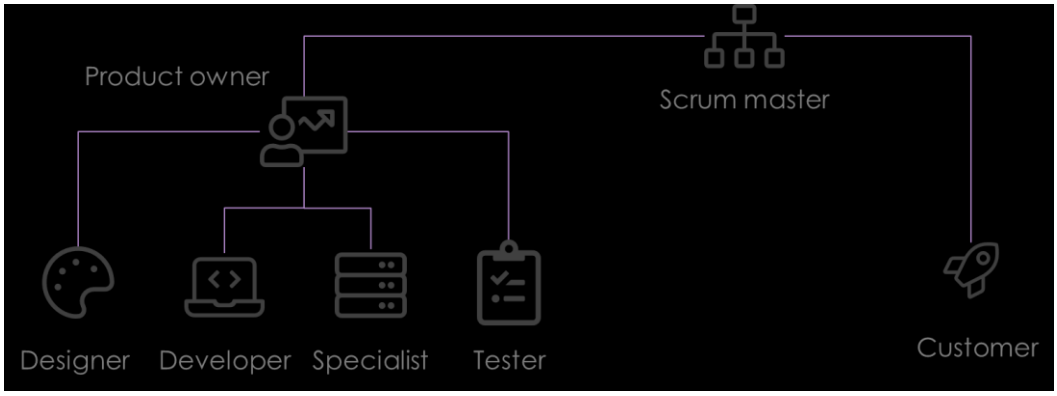
Scrummaster kommunikoi tuoteomistajan ja asiakkaan välillä tulkaten toiveita ja etenemistä kumpaankin suuntaan.

Kehitystiimin designer vastaa käyttöliittymän ja käyttäjäkokemuksen suunnittelusta. Hänen vastuullaan on yrityksen imagon ja sanoman näkyminen sivuston ulkoasusta, kuin myös vaatimusmäärittelyn tuottaminen asiakasta haastattelemalla ja käytettävyyden toteuma sivustojen käyttöliittymissä.

Developer tuottaa tarvittavan koodin sekä palvelinta että käyttöliittymää varten designerin tuottaman suunnitelman pohjalta. On erityisen tärkeää mukailla suunnittelijan toteutusta, jotta käytettävyys ja viestintä eivät kärsi.

Specialist vastaa tässä projektissa tuontantoympäristön palvelimien ja tietokannan asennuksesta Microsoftin Azure -ympäristöön, sekä SSL sertifikaattien ja DNS -tietueiden ohajuksista ja käyttöönotoista. Ympäristön tulee myös tarjota kehittäjälle julkaisukanava sekä backendin, että frontendin koodille.

Testaaja testaa develperin tuottamaa käyttöliittymää ja toiminnallisuutta designerin suunnitelmaa vasten ja raportoi mahdollisita poikkeamista ja rikkeistä, jotta toteutus pysyy suunnitelman mukaisena. Tässä projektissa testaaja myös testaa sivuston suorituskykyä Googlen tarjoamaa sivuston suorituskykyauditointi -palvelua vasten ja raportoi parannusehdotuksia kehitykselle.



Kuva 3: Havainne kaavio projektin rooleista ja hierarkiasta

4.3 SWOT -analyysi

Projekti kokonaisuus tulee reflektoida ennen toteutusta, jotta on mahdollista tiedostaa projektin vahvuudet ja heikkoudet. Analyysiin voidaan käyttää SWOT -viitekehystä jossa analysoidaan toteutusta sekä sisäisten, että ulkoisten tekijäiden kannalta. SWOT -nimitys tulee englannin kielen sanoista **strengths** (vahvuudet), **weaknessess** (heikkoudet), **oppotunities** (mahdollisuudet) ja **threats** (uhat). Vahvuudet ja heikkoudet ovat kuvastavat organisaation sisäisiä tekijöitä mahdollisuudet ja uhat puolestaan organisaation ulkopuolisia tekijöitä. Vahvuudet ha mahdollisuudet edistävät tavoitteiden saavuttamista missä heikkoudet ja uhat ovat mahdollisia riskitekijöitä. (Opetushallitus, 2019)

Vahvuudet (S)	**Heikkoudet (W)**
- Tiivis organisaatio mahdollistaa dynaamisemman ympäristön kommunikoinnin ja kehityksen kannalta - Projektin työvaiheet on kartoitettu selkeästi ja kattavasti - Kehittäjillä on tuoretta kokemusta toteutuuttavista tehtävistä	- Kehittäjillä on osaamisesti huolimatta vain vähän kokemusta vastaavista töistä - Aikataulutettu työ on suuresti keskenään riippuvaista: yksittäisen tehtävän viivästyminen viivästyttää myös suurilta osin muita tehtäviä.
Mahdollisuudet (O)	**Uhat (T)**
- Merkittävät oppimismahdollisuudet, kokonaisymmärryksen kartoitus ja sen tarjoaminen tiedoksi muille. - Uusien teknologiaratkaisujen käyttö jättää mahdollisuuden teknologian kehittymisen myötä syntyville toiminnallisuukille käyttökanavan.	- Käytettävien palveluiden viitekehys ja tuntemattomuus saattaa aiehuttaa turhaa työtä ja kustannuksia

- Kattavan infrastruktuurin puolesta toiminnallisuudet ovat skaalattavissa tarpeen mukaan	

4.4 Resurssit

Jotta tuotoksessa voidaan mukailla asiakkaan sovelluksen käyttökokemusta mahdollisimman kattavasti, on asiakasta pyydetty toimittamaan lähdemateriaalia projektia varten. Lähdemateriaalin lisäksi asiakas toimittaa sivustolla esitettävät tekstit sekä sivuston tiedon, mitä dynaamista sisältöä kopioidaan uuteen tietokantaan. Dynaaminen sisältö on asiakkaan itse jatkuvasti päivitettävää dataa kuten blogi postauksia.

Asiakasta pyydetään toimittamaan seuraavat resurssit projektin toteutusta varten:

Design resurssit	- Väriarvo koodit sivuston väriteemaa varten joko HEX tai RGB muodossa - Fonttiperheiden nimet, joita sovelluksessa käytetään - Kuvamateriaalia sovelluksesta - Sivustolle tarkoitetut taustakuvat
Sivuston data	- Tekstisisältö jokaiselle sivulle - Tieto uuteen kantaan kopioitavista blogeista ja uutislinkeistä - Listan huolista huolet -sivua varten
Käyttäjätunnukset	- Korvattavan wordpress sivuston käyttöliittymään - Azure portaalin tunnutkset, joille on aktivoita Biz spark -stipendi

5 Projektin toteutus

Vaatimusmäärittelyn pohjalta on kehitystiimille syntynyt käsitys teknologiakokonaisuudesta, jolla asiakkaan projekti toteutetaan. Teknologiakokonaisuus kattaa niin kehitys, - kuin tuotanto ympäristön palvelut ja alustat, palvelin ja käyttöliittymä teknologiat.

Scrumin ideologiaan perustuen projektissa tavoitellaan parhaimman mahdollisen toteutuksen mukaista DevOps toimintamallia: Tuotteen lähdekoodi on saatavilla kaikkille viimeisimpänä versiona suoraan GIT -versionhallinnasta, josta operoidaan myös tuotteen päivitystä tuotantoympäristöön.

Tuotantoympäristönä toimii Microsoft Azure -alusta. Kyseisen alustan palveluista palvelinympäristö toteutetaan App service -palvelulla, jonne voidaan julkaista suoraan versiohallinnan kautta haluttu tuoteversio. Palvelinympäristöön tulee kohdentaa nimipalvelun DNS -osoitteet ja asentaa käyttöön SLL-suojaus App Service Sertifiacte -palvelun kautta. Sivustojen dynaaminen sisältö palvelaan Azure Cosmos DB alustalle asennetusta Mondo DB- tietokannasta.

Palvelin tuotetaan Node.js -viitekehyksellä ja julkaistaan versiohallinnan kauttaa Azuren App serviceen. Nodejs -palvelimen avuksi käytetään Express.js, EJS ja MongoDB viitekehyksiä, jotka yksinkertaistavat palvelinkokonaisuuden toteutusta. Palvelin koostaa HTML -dokumentin valmiiksi selaimelle, josta se tarjoillaan käyttäjälle vaatien mahdollisimman vähän käyttäjän päätelaitteelta.

Selaimeen lähetetty html -rakenne koostetaan palvelimella ja sivuston tyylit (css) on mahdutettu yhteen tiedostoon. Näin minimoidaan resurssien lataamiseen vaadittujen pyyntöjen määrä. Javascriptiä sivusto tarvitsee vain valikkojen avaamiseen, ja dynaamisen sisällön selaamiseen.

5.1 Tuotoksen kehitysjono

Tuotoksen kehitysjono on kooste tehtävistä, jotka vaaditaan valmiin tutoksen savuttamiseksi. Kehitysjonon pohjan on suunnitellut projectin tuoteomistaja, joka on vaihestanut työn alustavasti kronologiseen järjestykseen tuotoksen tiedettyjen vaiheiden pohjalta. Tämän jälkeen kehitystiimi on käynyt läpi tuotoksen kehitysjonon tehtävät tuoteomistajan kanssa, jolloin tehtävälistausta on muokattu työn toteutuksen kannalta konkreettisempiin tehtäviin.

Kehitysjono kattaa tehtäväkohtaisen listan kaikista vaiheista tuotoksen saavuttamiseksi. Tuotoksen kehtiysjono on luettavissa opinnäytetön liitteenä.

5.2 Sprint

Projektin tehtävät suoritetaan neljässä sprintissä. Sprintti alkaa aina suunnittelulla, missä dokumentoidaan tiivisti tulevan sprintin aikana suoritettavat tehtävät. Tehtävien riippuvuus toisistaan käydään myös läpi ja kehitystiimi sopii yhdessä työjärjestyksen ja tavoiteajat tehtäväkohtaisesti, jotta sprintin tavoitteisiin ylletään.

Kehitysjonon tehtävien suoritus on dokumentoitu ikään kuin päivittäispalaverissa läpikäyden. Tavoitteena on tarjota yleiskuva tehtävän kulusta ja toteumasta kaikille ymmärrettävällä tasolla. Tehtävä on numeroitu kehitysjonon numeroinnin mukaisesti.

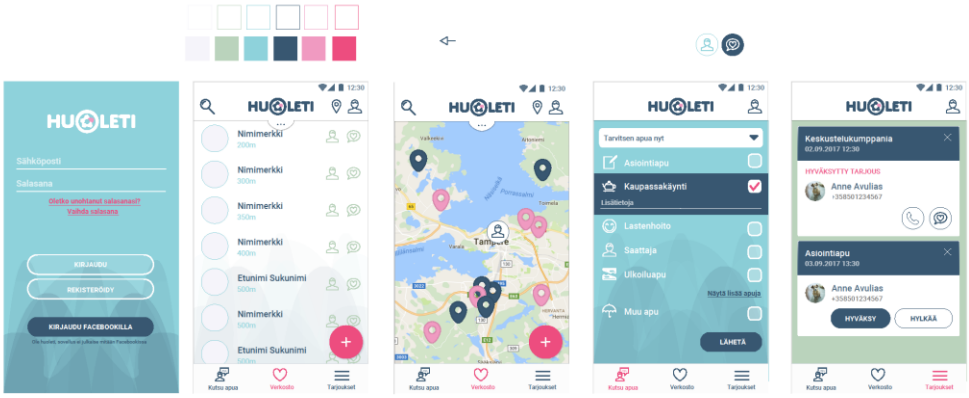
Sprintin päätyttyä koostetaan lyhyt lausunto sprintin katselmoinnista, sekä retrospektiivistä. Katselmoinnin aikana arvioidaan sprintistä suoriutumista ja suunnitellaan suoritettavat tehtävät tulevalle sprintille.

5.2.1 Sprint 1/4

Projektin ensimmäisen sprintin tavoite on saada toteutettua sivuston **ulkoasusuunnitelma** (1), tuotantoympäristön **palvelinympäristö** (19) ja **tietokantaympäristö** (14), **tietokannankopiointi** (15), **versiohallinta** (17), **palvelin** (10) sekä **palvelinkirjastojen implementointi** (11, 12).

Sprintin tehtävät jakautuvat kehitystiimistä designerin, developerin ja specialistin kesken. Ensimmäisen sprintin tehtävistä numero 14 (specialist) toteutus on riippuvainen tehtävän 10 (seveloper) suorittamisesta, kehitystiimin jäseniä suositellaan huomioimaan tämä suunnitelmissaan ja sopimaan yhdessä tavoiteajan etenemistä estävän työn suorittamiseksi. Töiden tulee valmistua sprintin aikana aikataulun toteutumiseksi.

Suunnittele sivuston teema (Tuotoksen kehitysjono: tehtävä 1)
Teeman suunnittelua varten asiakas on toimittanut kuvamateriaalia sovelluksen uudistetusta ulkosusta, kuva liitteenä. Asiakas on myös maininnut sivustoja kuten haltu.fi ja reaktor.fi visuaaliseksi inspiraation lähteeksi.



Kuva 4: Asiakkaan sovelluksen design

Tuotettu teema on käyty läpi asiakkaan kanssa ja tehty muutamia asettelu muutoksia. Asiakas on hyväksynyt designin ja ulkoasu voidaan toteuttaa sen pohjalta.

Perusta App servicet (Tuotoksen kehitysjono: tehtävä 19)
Asiakkaalle on perustettu Azureen App Service Plan. App Service Plan on Azuren resussi, jonka kautta määritetään mm. palvelimien suorituskykyä, App serviceitten maksimimäärä ja palveluiden kattavuutta.

App Service Planin alle on perustettu Azuren Resource group -resussi, jonka sisään voi ryhmittää palveluita. Saman Resource groupin sisäiset resurssit voivat myös jakaa yhteisiä Azuren sisäisiä palveluita, kuten SSL suojaus lisenssin olettaen, että lisenssi on alidomainelle vapaasti laajennettava.

Resource groupin alle on perustettu App service -palvelu. App service kanavoi Microsoft IIS:n kautta käskyjä joko Azuren rajapintaan tai App servicen ajamaan palvelinkoodiin. IIS toimii Azuren palveluiden rajapintana esimerkiksi SSL ja DNS konfiguraatioissa, mutta kaikki kutsut joihin app servicesta löytyy vastaus, käsitellään sitä kautta.

Back- ja fontend koodin GIT julkaisua varten App serviceen tarvittavan osoitteen ja tunnukset löytää Azuresta App servicen properties -välilehdellä. Julkaisua varsten Azuren repository liitetään yhdeksi remote branchiksi, jonne valmiit versiot totoksesta julkaistaan. Julkaisun saatuaan Azure kääntää automaattisesti koodin toimintaan App servicen päällä.

Versiohallinnan perustaminen (Tuotoksen kehitysjono: tehtävä 17)
Githubiin on luotu uusi repository ja kaikki projektin osalliset on kutsuttu sen jäseniksi. Tämä versiohallinta on kaikille avoin ja kopioitavissa lokaaliksi.

Toiminnallisuudet toteututetaan totuetushaaroittain ja haarat yhdistetään päähaaraan haaran valmistuttua. Kun oma toteutus on valmis ja testattu lokaaliympäristössä voi sen yhdistää päähaaraan ja julkaista uutena versiona.

Tuota Node.js palvelin (Tuotoksen kehitysjono: tehtävä 10)
Node.js palvelinta kehittäessä kaikilla kehittäjillä tulee olla sama Node.js versio asennettuna. Projektissa käytetään viimeisintä vakaata versiota joka on tarkistettavissa versiohallinnasta.

Palvelimen riippuvuudet on listattuna package.json tiedostossa, itse riippuvuudet on poistettu versiohallinnan alaisuudesta, jottei versiohallinnan koko kasvaisi kohtuuttomaksi. Node.js projekteissa viimeiseimmät versiot ilmoitetusta riippuvuuksista voi asentaa automaattisesti ”npm install” -komennolla.

Toivottu Hello World -toteutus on valmis. Koodi on otettu suoraan Node.js -kotisivulta. Tämä toiminnallisuus on todettu valmiiksi ja julkaistu versiohallintaan.

Perusta tietokanta (Tuotoksen kehitysjono: tehtävä 14)
Tietokanta on asennettu Azuren tarjoaman Cosmos DB -tietokantapalvelun päälle. Cosmos DB:n päälle voi asentaa noSQL -tietokanta viitekehyksiä. Tietokanta on perustettu MongoDB viitekehyksellä. Tietokannan CRUD toiminnallisuuksia on mahdollista toteuttaa sekä kantaan yhdistetyltä palvelimelta, että Azuren portaalista.

Palvelimelle on asennettu MondoDB riippuvuus ja palvelin on ohjemoitu yhdistämään tietokantaan viitekehyksen dokumentaation mukaisesti. Muutos on julkaistu versiohallintaan.

Vanhan tietokannan kopiointi (Tuotoksen kehitysjono: tehtävä 12)
Asiakkaan vanhalle wordpress sivustolle on asiakas tuottanu dynaamista dataa blogin ja uutismainintojen linkkausten muodossa. Wordpressin käytössäoleva datarakenne ei soveltunut suunniteltuun MongoDB rakenteeseen. Koska sisältöä ei ollut kuin muutaman blogipostauksen ja noin 20 uutislinkin verran, oli toivotun toteutuksen tavoittamiseksi nopeinta kopioida tiedot käsin uuden datamallin mukaisesti.

Lisää Express -kirjasto palvelimen käyttöön (Tuotoksen kehitysjono: tehtävä 11)
Palvelimelle on asennettu Express.js -palvelinfunktiokirjasto yksinkertaistamaan palvelinlogiikkaa ja säästämään aikaa. Palvelinkoodi on päivitetty käyttämään express kirjastoa Hello World -toiminnallisuden saavuttamiseksi.

Lisää EJS -kirjasto palvelimen käyttöön (Tuotoksen kehitysjono: tehtävä 12)
Palvelimelle on asennettu EJS -dokumentingenerointikirjasto. Dokumenttipohjille on luotu kansiorakenne ja malli tiedostot. Hello World -toiminnallisuutta on muutettu tuottamaan dokumentti EJS:n sapluunalla normaalin tekstin sijasta.

Sprint review
Kaikki suunnitellut tehtävät on ehditty toteuttamaan ja toteamaan toimiviksi. Yhteenvetona on onnistuttu tuottamaan toimiva kehitys- ja tuotantoympäristö versiohallinnan alaisuuteen. Palvelin on valmis Frontend -kehitykselle ja sivujen dokumenttien tuotantoon. Tietokantaan voi aloittaa CRUD toiminnallisuuksien funktioita palvelimen koodiin.

Tulevan sprintin aikana oleellisinta olisi toteuttaa tuotoksen kehitysjonon tehtävät 8,2 ja 9, jotta projekti edistyy aikataulussaan ja loppujen tehtävien toteutuksille aukeaa kanava.

Sprint retrospektiivi
Menneellä sprintillä tiimin toisistaan riippuvainen toiminta onnistui mallikkaasti ja tehtävät oli ajoitettu helposti toteutettaviksi. Tehtäville oli varattu ehkä liikaakin aikaa. Tämä kuitenkin vahvisti käsitystä kehitystiimin työtahdista ja jatkossa tehtäviä uskaltaa kartoittaa nopeammalla toteutustahdilla.

5.2.2 Sprint 2/4

Ensimmäisen Sprintin aikana onnistuttiin toteuttamaan sivustokokonaisuuden infrastruktuurin perustqa tarvittuihin ympäristöihin ja implementoimaan perustoiminnallisuudet kuten palvelin, joka vastaa kutsuihin.

Sivuston ulkoasusuunnitelman valmistuttua kehittäjä pääsee tuottamaan teeman rakenteenmukaisen **aloitussivun** (2) ja sille **tyylit** (8), jotka palvelevat pohjana myös muita, myöhemmin luotavia sivuja. Kun perustyyli on valmis, tehdään asiakkaan toivoma funktionaalinen toiminnallisuus, mikä totetuttaa **tekstilohkon animoitumisen näytölle** (9).

Tällä sprintillä developerin työmäärä on suuri missä muilla osallisilla ei ole heille vartavasten osoitettuja tehtäviä. Muulta kehitystiimiltä odotetaan siis aktiivista developerin tukemista, mikäli se suinkin on mahdollista.

Aloitussivu (Tuotoksen kehitysjono: tehtävä 2) ja sivuston **teema** (Tuotoksen kehitysjono: tehtävä 8) toteutettiin keskenään samanaikaisesti, koska ne ovat suuresti toisistaan riippuvaisia tehtäviä. Sivustolle on luotu vakiosegmentit EJS -sapluunoina. Sivuston rakeneen rinnalle on luotu rakennetta mukaileva CSS temaa, jota toistamalla sivuston rakenne pysyy yhtenäisenä.

CSS teemaan luotu tyylit myös vakio komponenteille; kuten napeille, valikoille, linkeille, työkalupalkeille ja sivusto segmenteille. Lisäksi sovelluksen värimaailma ja fontit ovat osa tuotettua teemaa.

Aloitussivu on tuotettu rakenteeltaan ja ulkoasultaan designerin tuottaman layoutin mukaan. Tekstisisältö on toistaiseksi kirjoitettu suoraan HTML -dokumenttiin. Mikäli sivustoa halutaan muuttaa tulevaisuudessa monikieliseksi vaihdetaan, EJS sapluunat käyttämään sisältönään kielitiedoston mukaista dataa.

Animaatio funktio (Tuotoksen kehitysjono: tehtävä 9)
Animaatio toteutettiin javascriptillä tarkkailemma sivuston ”scroll” -interaktiota, eli tilannetta jolloin sivua selataan ylös tai alas. Kun sivua selataan, tarkistetaan onko ruudun alueella tekstilohkoja, joiden kuuluisi olla näkyvissä. Mikäli tekstilohkoja löytyy annetaan niille CSS -luokka, joka animoi tekstilohkot näkyviin. Animointi on kohdennettu erilliseen CSS -luokkaan, joten mikäli jokin segmentti halutaan jättää animoimatta, jätetään vain animoinnin mahdollistava luokka pois.

Sprint review
Suunnitellut tehtävät ehdittiin toteuttaa. Vaikkakin kaikki tehtävät olivat developerin toteutettavana ei työmäärä ollut kohtuuton. Sprintin totoksena on onnistuttu luomaan helposti toistettava rakenne, joka toteuttaa sivustolle suunnitellun teeman.

Tulevan sprintin tavoitea on työmäärällisesti merkittävästi päättyvää kattavampi. Tulevalla sprintillä toteutetaan sivustokokonaisuuden alasivut (3-7), kohdennetaan palvelin käyttämään niitä ja vastaamaan tarvittaessa ”sivua ei löydy” -sivulla (13). Kun sivustojen perusrakanne on valmis, tuodaan vielä dynaaminen sisältö sitä käytäville sivuille (16) ja testataan kokonaisuus (18).

Sprint retrospective
Menneen sprintin aikana työ oli developer keskeistä eikä yhteistyötä vaativia tehtäviä ollut. Tehtävä määrä oli pieni mutta tehtävät suuria. Ajallisesti ne asettuivat hyvin sprintin jaksolle. Animoinnin toteutukselle olisi monia vaihtoehtoisia toiminnallisuuskia, on harmillista ettei ole toista developeria konsultoimassa.

5.2.3 Sprint 3/4

Toisen sprintin tuotoksena syntyi suunnitellun teeman mukainen, toistettava rakenne koko sivustolle, jota voidaan nyt alkavan kolmannen sprintin aikana kopioida tuotettavien alasivujen rakenteeseen.

Sprintin aikana on tavoite toteuttaa tuotoksen kehitysjonossa esitetyt sivsuton **alasivut** (3-7). Varotoimena, mikäli kyseistä alasivua ei löydy, ohjataan käyttäjät **sivua ei löydy** -sivulle (13). Sivuston blogi-, sekä uutiset -sivuja tuottaessa ohjelmoidaan sivut **näyttämään oikeaa tietokannan dataa** (16). Kun tehtävät on suoritettu **testataan kokonaisuus** (18) ja todetaan toimivaksi mikäli tämä suinkin on mahdollista.

Valvatosa sprintin tavoitteen tehtävistä kuuluu jälleen developerilla. Kokonaisuuden testaus on pääasiallisesti testaajan tehtävä, mutta minkä tahansa vian ilmetessä tulee tämä työllistämään joko developeria tai specialistia. Tämän johdosta sprinttiin on varattu resurssiksi developer, specialist ja testaaja.

Alasivustoja (Tuotoksen kehitysjono: tehtävät 3-7) lähdettiin totetuttamaan sivuston yleisrakenteen pohjalta. Kaikilla sivustoilla on tarkoitus vaihtaa sivustorakenteeseen oikea tekstisisältö ja lisäksi toteutta sivukohtaisiksi määritettyjä muokkauksia tai lisäyksiä. Mikäli sivulla ei ollut muuta muokattavaa kuin tekstirakenne, ei sitä ole alla erikseen mainittu.

Tuki -sivulle (Tuotoksen kehitysjono: tehtävä 3) tuotettiin perusmuokkausten lisäksi ”haitari” toiminnallisuus **Usein kysytyt kysymykset** -osioon. Haitari toiminnallisuudella on tarkoitus näyttää vain kysymykset ja kysymystä klikatessa tuoda näkyviin kyseisen kysymyksen vastaus. Kysymyksiä pitää myös pystyä suodattamaan hakutermin perusteella.

Nämä toiminnallisuudet toteutettiin kumpikin samalla periaatteella: Javascriptillä annetaan oikeassa tilanteessa oikealle kysymykselle, tai kysymyksille, CSS -luokka, jolla niitä joko tuodaan näkyviin tai piiloitetaan näkyvistä. Hakua tehdessä piiloitetaan kaikki kysymykset, mitkä eivät otsikko- tai tekstitasolla sisällä hakukenttään syötettyä tekstin pätkää.

Huolet -sivulle (Tuotoksen kehitysjono: tehtävä 4) tuotettiin perusmuokkausten lisäksi ”kääntökortti” -toiminnallisuus. Kyseisellä sivulla on listattuna kaikki huolet, joihin käyttäjä voi itsensä ilmoittaa osalliseksi. Sivulla on tarkoitus auttaa käyttäjää löytämään itselleen sopivin ryhmä. Jokainen kortti kattaa yhden huoliryhmän. Kortin kannessa on kerrottu huoliryhmän nimi, kortin kääntöpuolella on puolestaan listattu kaikki ryhmään kuuluvat huolet.

Korttia klikatessa kortti kääntyy ympäri ja näyttää huoli -ryhmän sisällön. Ohjelmoinnillisesti korttia klikatessa asetetaan sille javascriptillä CSS -luokka joka kertoo kummin päin kortti on. Kortin käännön lisäksi kortteja tulee voida suodataa samalla lailla, kuin **Tuki -sivun** kysymyksiä. Suodatukseen käytetään samaa toiminnallisuutta, kuin tukisivulla.

Blogi -sivulle (Tuotoksen kehitysjono: tehtävä 5) tuodaan asiakkaan tietokannasta blogiksi merkittyjä tekstejä, jotka ovat **julkaistu** -tilassa. Näiden lisäksi tuotuja tekstejä rajoitetaan kieli -parametrin mukaan: suomen kielisellä sivulla näyteään suomen- ja englannin kieliset sisällöt, huoleti.com sivulla vain englannin kieliset sisällöt. Nämä järjestetään ajallisesti uusimmasta vanhimpaan.

Itse sivulla jokaisesta postauksesta näytetään otsikko, johdanto ja kuva. Lisäksi tarjotaan linkki lukemaan teksti kokonaisuudessaan. Sivu, jolla näytetään yksittäinen kirjoitus saa tiedon kyseisen tekstin ID -arvosta. Sivulle navigoidessa hakee palvelin kaikki kyseisen kirjoituksen tiedot ilmoitetun ID:n perusteella, mikäli ID:tä ei löydy tai se on väärän mallinen ohjataan **sivua ei löydy** -sivulle.

Uutiset -sivulle (Tuotoksen kehitysjono: tehtävä 6) tuodaan asiakkaan tietokannasta uutislinkeiksi merkittyjä tekstejä, muutoin tominnallisuus vastaa täysin blogisivulla toteutettua kirjoitusten listausta. Uutislinkit ovat linkkejä uutisiin, joissa asiakas on mainittu. Asiakas voi itse linkin sisäksi syöttää uutislähteen, otsikon ja julkaisupäivän.

Koska uutislinkkejä on paljon, ja tulee olemaan vielä enemmän, näytetään uutisista maksimissaan kuusi kerrallaan. Loput uutiset löytyvät sivutuksen takaa kuuden ryhmissä. Aina sivustusta vaihtaessa noudetaan AJAX -kutsulla seuraavat kuusi uutislinkkiä ja parsitaan ne näkyviin javascriptillä. Lisäksi **Uutiset** -sivulle upotetaan syötteet asiakkaan facebookista ja twitteristä.

Sivua ei löydy -sivulle (Tuotoksen kehitysjono: tehtävä 13) ohjataan aina, mikäli mikään palvelimelta pyydetty osoite tai tiedosto vastaa pyyntöä. Näin saadaan sekä informoitua käyttäjää, että estettyä vihreelliset pyynnöt palvelimelle tarjoamalla mihinkä tahansa pyyntöön jo-

kin vastaus. Sivu näyttää pyydetyn osoitteen, ja ilmoittaa ettei tälläistä alasivua ole sivustolla. Sivu on sivuston muun teeman mukainen, joten käyttökokemus ei katkea vaikka virheelliseen linkkiin navigoisikin.

Sprint rewiev
Suunnitellut tehtävät ehdittiin toteuttaa ja lopputuloksena saatiin toimiva kokonaisuus. Sivustolla on luotu kaikki vaaditut alasivut ja niille suunnitellut lisätoiminnallisuudet.Toteutuksessa ei syntynyt ongelmia ja lopputulokseen ollaan tyytyväisiä.

Toisella sprintillä tuotettu teema oli helposti toistettavissa ja ala-sivujen luonti onnistui helposti, mikä säästi paljon arvokastaaikaa. Tulevan sprintin aikana julkaistaan sivustokokonaisuus kehitysympäristöstä tuotantoympäristöön, asetetaan sivustolle SSL suojaus ja käännetään asiakkaan virallinen DNS osoittaamaan App serviceen.

Sprint retrospective
Tehtäviä oli paljon, mutta hyvintehdyn pohjatyön ansiosta tehtävistä suoriuduttiin nopeasti. Kehitystiimin keskeinen kommunikaatio toimi hyvin ja tehtävissä, joissa yhdistettiin useamman tekijän aikaansaannoksia oli työn dokumentointi helposti ymmärrettävää.

5.2.4 Sprint 4/4

Kolmannen sprintin aikana sivuston kokonaisuus saatiin testiympäristössä valmiiksi ja se voidaan luovuttaa testaajalle **testattavaksi**(18) kokonaisuutena. Sivustolla on oikeat alasivut sekä dynaaminen sisältö ja sivukohtaiset lisätoiminnot. Mikäli sisältöä ei ole, on sillekkin toteutettu ratkaisu ilmoittaamaan virheellisestä osoitteesta.

Testaaja raportoi mahdollisista virheistä tai vioista muille kehitystiimin jäsenille virheen laadun mukaan. Kun testaaja vahvistaa kokonaisuuden toimivaksi voidaan se **julkaista tuotantoympäristöön** (20). Kun sivusto on julkaistu azuren tuotantoympäristöön pitää oikeat **DNS -tietueet** (21) kohdistaa oikeaan App Serviceen. Lisäksi sivustolle tulee asettaa **SSL suojaus** (22). Varsinaisesti selaimen ja palvelimen välillä ei liiku tällä sivustolla dataa mitä pitäisi kryptata, mutta SSL -suojaus on oletettu standardi ja asiakkaan uskottavuuden kannalta suotavaa toteuttaa.

Sivustokokonaisutta **testatessa** (Tuotoksen kehitysjono: tehtävä 18) on oleellista muistaa toiminnallisuuksien lisäksi testata sivustoa kaikilla laitteilla, joita sivusto haluaa tukea. Asiakkaan tapauksessa tämä kattaisi Android, iOS, Windows käyttöjärjestelmillä varustetut älypuhelimet ja tabletit sekä tietokoneissa yleisimmin käytety selmainet: Google Chrome, Safari, Mozilla Firefox, Microsoft Edge ja Opera. Internet Explorer tuki olisi myös suotavaa ainakin kotisivustoille, sillä mitään niin funktionaalista toiminnallisuutta ei sivustolla toteuta, etteikö

vanhempikin selain siitä selviytyisi. Ylläpitokäyttöliittymän tuki vanhemmalle selain sukupolvelle ei ole tarpeellinen.

Erona mobiililaitteella ja tietokoneella on se, että tietokoneella selaimet saavat käsitellä sivut omien moottoriensa kautta, missä mobiililaitteissa sovellukset saatetaan rajoittaa käyttämään järjestelmän tukemaa sivustomoottoria. Esimerkiksi iOS laitteilla, riippumatta selaimesta, sivustot käsitellään applen tukeman webkit moottorin kautta, eikä selainten omalla moottorilla joita tietokoneohjelmat puolestaan hyödyntävät. Google ei tätä puolestaan vaadi, mutta muut selaimet eivät androidilla ole saavuttaneet niin suurta käyttäjäkuntaa, että niitä voisi pitää valtavirran käyttäminä. (Steven J. Vaughan-Nichols, 2017)

Mobiililaitteilla testastaan siis järjestelmän oletusselaimella, tietokoneilla yleisimmillä morderneilla selaimilla. Selaimen tulee myös olla päivitettynä viimeisimpään ohjelmistoversioon.

Testatessa sivustoa ilmeni, ettei käytössäoleville CSS muuttujille oltu tehty fallback -arvoja niitä tukemettomille selaimille, kuten Internet Explorerille. Tämä raportoitiin developerille ja fallback arvot lisättiin. Muutoin sivuston toiminnallisuudessa ei havaittu ongelmia.

Sivusto julkaistaan (Tuotoksen kehitysjono: tehtävä 20) Git -versiohallinnasta suoraan Azuren App Serviceen. Versiohallintaan lisätään etähaara App Servicen tiedoista saatavissa olevaan osoitteeseen ja sille asetetuilla julkaisutunnuksilla. (Microsoft Azure, 2018)

Kun etähaara on lisätty voidaan, toiminnallisuudelle tehdä komentorivikehoite uuden version julkaisuun App servicen etähaaraan. Versiohallinnan etähaara Azuren julkaisua varten on käytettävissä myös App servicen tietojen kautta Azuren käyttöliittymässä.

Sivusto on julkaistu, mutta se näkyy vain App Servicen oletusosoitteesta: **appservicennimi**.azurewebsites.net. Tämä tulee muuttaa siten, että asiakkaan domainia pyytäessä pystytään avaamaan sivu tuohon oikeaan osoitteeseen. Tämä tehdään asettamalla oikeat tietueet nimipalveluun, mitkä yhdistävät App servicen palvelin sijainnin pyydettävään DNS osoitteeseen.

Asiakkaan DNS (Tuotoksen kehitysjono: tehtävä 21) ohjataan osoittamaan oikeaan palvelimeen nimipalveluiden hallintapaneelista. Nimipalveluiden hallintaan lisätään A -tietue sekä TXT -tietue. A -tietueessa ilmoitetaan App servicen tiedoista löytyvä IP -osoite tietueen IP -osoitteeksi ja alidoimainiksi ilmoitetaan sekä pelkistetty- , että villikortti- domain. Asiakkaan tapauksessa osoitteet ovat huoleti.com ja *.huoleti.com. Tämän jälkeen otetaan käyttöön vielä CNAME -tietue etuliitteellisille alidomainelle, malliltaan esimerkiksi www.huoleti.com.

Poiketen A -tietueista CNAME -tietueelle annetaan IP -osoitteen sijasta kohdeosoite. Kun sivustoa palvellaan Azuren App servicesta, ilmoitetaan App servicen oletus domain: **huoleti.azurewebsites.net** CNAME -tietueessa. Kun nimipalvelussa on asetettu halutut tietueet oikein, App servicen asetuksissa ilmoitetaan mitä domainia halutaan käyttää, jolloin Azure tarkistaa saatavuuden ja luo yhteyden domainin ja palvelimen välille, jos tämä on mahdollista.

Sivuston SSL -suojaus (Tuotoksen kehitysjono: tehtävä 22) asennetaan suoraan Azuren sisäisesti käyttäen Azuren tarjoamaan App Service Sertificate palvelua. App Serice Certificate on mikropavelu App servicen käyttäjille, missä Azure toimii välikätenä SLL -sertifikaatin hankinnassa ja antaa sertifikaatin käyttöön sille osoitetun Resource Groupin App serviceissä yksinkertaistaen näin sertifikaatin asennusta merkittävästi. Kun sertifikaatti on lunastettu ja turvattu dokumentaation mukaisesti voi sen asettaa sivustolle App servicen asetusnäkymästä. (Microsoft Azure, 2018)

Sprint review

Sivuston voidaan todeta kokonaisuudessaan julkaistuksi ja toimivaksi kokonaisuudeksti mikä vastaa oikeista osoitteista oikealla sisällöllä suojatun yhteyden läpi. Tuote on esitetty myös asiakkaalle tässä tilassa ja asiakas hyväksyy työn valmiiksi ja on tyytyväinen työn jälkeen.

Ongelmia tuotantoympäristöön julkaisussa syntyi ensiksi oikean Node.js -versioinnin kanssa: Azuren App servicen huomattiin käyttävän eri versiota, kun millä projekti on rakennettu. Ongelmaan löytyi kuitenkin korjaus ja palvelinta käsitellään nyt oikealla ohjelmistoversiolla.

Vielä puuttuva ylläpitokäyttöliittymä tuotetetaan ylimääräisenä sprinttinä tämän sprintin jälkeen. Ylläpitokäyttöliittymä tuotetaan hyödyntämällä mahdollisimman paljon asiakkaan julkisivustoon tuotettua koodia. Ylläpitokäyttöliittymälle asennetaan oma App service julkisivuston rinnalle.

Sprint retrospective

Sprintin ja koko scrumin eteneminen sujui ongelmitta, tehtävät oli mitoitettu oikeille henkilöille ja suunniteltu loogiseen järjestykseen. Tiiviin toteutusaikataulun johdosta ei asiakkaan kanssa päädyttiin siihen, että ylläpitokäyttöliittymä toteutetaan viikon mittaisella projektin lisäjaksolla.

5.3 Extra sprint: Ylläpitokäyttöliittymä

Sprintin suunnitteluissa ei otettu huomioon ylläpitokäyttöliittymän luontiin liittyviä toiminnallisuuksia, sillä asiakkaan toiveesta haluuttiin ensiksi keskittyä julkisiin sivustoihin. Ylläpito käyttöliittymän työvaiheet on kuitenkin suunniteltu tuotoksen kehitysjonon jatkoksi.

Tällä lisäsprintillä tuotetaan julkisivustoprojektin pohjalta ylläpitokäyttöliittymä hyödyntäen mahdollisimman paljon projektin olemassaolevaa koodia: **Julkisivuston versiohallinnasta valitaan mahdollisiman lähellä haluttua vaihetta** (23) oleva versio ja kopioidaan se oman GIT -versiohallinnan alle, jonka jälkeen **poistetaan turhat resusrssit** (23). Kun projekti on perustettu tuotetaan tietokannan Read -funktioiden rinnalle loputkin vaaditut **CRUD toiminnot** (24) palvelimelle. Kun palvelimella on tuotettu toiminnot, luodaan ylläpitopaneelin **käyttöliittymään lomakkeet** (25) CRUD toiminnallisuuksien ajamiseen. CRUD -lomakkeiden lisäksi tuotetaan ylläpitopaneeliin **kirjautumis sivu** (26) ja koodataan palvelimelle **kirjautuneen käyttäjän istunto** (27). CRUD toiminnallisuuksien ajo rajataan mahdolliseksi vain kirjautuneille käyttäjille. Lopuksi kokonaisuus **testataan** (28) ja julkaistaan omana App servicenään julkisuvistolle luodun Resource Groupin sisään kotisivujen rinnalle.

Ylläpitopaneelin projektin pohjaksi **kopioidaan julkisivusto -projektin** (Tuotoksen kehitysjono: tehtävä 23) versiohallinnasta sopivin versio. Näin kehitystiimi tekee vähiten tupla työtä. Ylläpitopaneeli ei käytä julkisivuston ulkoasuja, joten rakennepohjaa sivustolle ei tarvitse kopioida. Ylläpitopaneelin käyttöliittymä tulee olemaan todella siisti ja yksinkertainen. Tärkeimmät kopioitavat toiminnallisuudet olivat mahdollisimman valmis palvelin tietokantayhteyksillä. Ylläpitopaneelille perustettiin oma versiohallinta ja sopivimmasta julkisivuston versiosta otettiin kopio.

Kun pavelin tietokannan lukuoikeuksilla on valmiina, voidaan lukuoikeuksien rinnalle lisätä muita **CRUD*** (Tuotoksen kehitysjono: tehtävä 24) toiminnallisuuksia. Ylläpitopaneelissa tulee pystyä hallinnoimaan sekä blogi päivityksiä, että uutislinkkejä, joita asiakas esittää julkisivustollaan. MongoDB tarjoaa suoraan toiminnallisuudet niin tietojen päivitykseen, luontiin kuin poistoonkin. Dataa muokatessa tai poistaessa kohdistetaan toiminto oikeaan tietoon tämän ID -arvon perusteella. (* lyhenne sanoista **Create - Read - Update – Destroy**, millä tarkoitetaan tietokantaa vastaan suoritettavia tietoja käsitteleviä toimintoja).

Palvelimen CRUD toinnallisuuksien käyttämiseksi ylläpitopaneelin käyttöliittymästä tuotetaan **lomakkeet** (Tuotoksen kehitysjono: tehtävä 25) jokaiselle funktiolle. Muokkaus ja poisto lomakkeissa pyydetään ensiksi tieto, mitä dataa halutaan muokata, jonka jälkeen lomake esitäytetään valitulla tiedolla. Valtaosa tiedoista on natiiveja teksti- tai valinta -kenttiä. Itse blogitekstin kirjoitusta varten otettiin käyttöön kolmannen osapuolen työkalu: Quill -teksti

editori selaimille. Näin mahdollisestaan tekstille perus muotoilu mahdollisuudet missä natiivi tekstikenttä ei tätä tarjoa.

Lomake -sivujen lisäksi paneeliin tuotettiin **kirjautumis -sivu** (Tuotoksen kehitysjono: tehtävä 26), kun käyttäjä ei ole kirjautuneena näytetään kirjautumis lomake. Kirjautuneena näytetään linkit hallintapaneelin lomakkeisiin. Kirjautumissivun vastakappaleeksi palvelimelle tuotetaan käsky **sisään- ja uloskirjaumiseen** (Tuotoksen kehitysjono: tehtävä 27) sekä rajoitetaan sisältö vain kirjautuneelle käyttäjälle. Käyttäjän kirjautuessa kirjoitetaan eväste palvelimen toimesta selaimen välimuistiin ja kirjautuminen on voimassa kunnes käyttäjä kirjautuu ulos. Käyttäjätunnuksia on vain yksi ja se on kovakoodattu palvelimen muuttujaksi.

Kun käyttöliittymä ja toiminnallisuudet ovat valmiit **testattiin** (Tuotoksen kehitysjono: tehtävä 28) ylläpitopaneelin toiminnallisuudet. Huomattiin, että blogitekstiä päivittäessä kaikki tiedot eivät tallentuneet oikein. Tämä johtui kirjoitusvirheestä koodissa. Vika korjattiin.

Ylläpitopaneelia varten perustettiin oma App service julkisivuston käyttämän Resource Groupin alle. Sivustolle luotiin nimitietue admin.huoleti.com, jota vasten ylläpitopaneelin App service ohjattiin aukeamaan.

Sprint review
Ylläpitopaneeli saatiin luotua kiitettävän nopeasti hyödyntäen mahdollisimman paljon asiakkaan julkisivustoon tuotettua koodia. Aisiakas on tästä erittäin tyytyväinen ja pitää käyttöliittymää suoraviiveisena ja selkeänä.

Sprint retrospective
Ylläpitopaneelin toiminnallisuuksien tuottaminen sujui jopa yllättävän kivuttomasti. Lisäksi on hienoa, että palvelimella on API -kutsut toimintojen ajamiseen. Tällöin on mahdollista tulevaisuudessa vaihtaa pelkkä Front End.

5.4 Ylläpito

Sivuston valmistuttua asiakkaan kanssa on sovittu ylläpitosopimus tulevia lisätöitä muokkauksia varten. Todennäköisiä ylläpitotöitä ovat sivuston tekstien muuttaaminen tai lisä sivujen tekeminen esimerkiksi jotain tiettyä käyttäjäkampanjaa varten.

Asiakas on ensisijaisesti velvoitettu ottaamaan yhteyttä sivuston tuottaneeseen tahoon ylläpitotöissä. Normaaleihin työpyyntöihin, jotka eivät ole kiireellisiä on määritetty työlle palvelu taso kaksi. Palvelutaso kakkosen töihin reagoidaan 48h sisällä asiakkaan yhteydenotosta ja sovitaan toteutusaikataulu tulevan kahden viikon ajalle, näistä töistä laskutetaan sovitun tuntihinnaston mukaisesti.

Kiireellisiä vikakorjauksia tai muutoksia varten on asiakkaan kanssa sovittu palvelutaso ykkösestä. Näissä tapauksissa työpyyntöön reagoidaan heti kuin mahdollista, mutta viimeistään 24h kuluessa yhteydenodosta. Työn vaatimat toimet suunnitellaan ja toteutetaan heti. Palvelutaso ykkösen töistä laskutetaan tuplahinta palvelutaso kakkosen töihiin nähden työn kiireellisyyden nojalla.

Tähän mennessä asiakkaalle on tehty ylläpitotöinä tervetulo -sivuja käyttöönottokampanijoiden tueksi niin sovelluksen pilotointiin, kuin yhteistyöjärjestöjen kanssa. Sivuilla tarjotaan lyhyt esittely järjestöstä ja yhteistyöstä, sekä tarjotaan käyttöönottokoodi ja latauslinkki sovelluskauppaan. Kampanja sivustojen lisäksi asiakkaan englanninkieliselle sivustolle on päivitetty lisää sisältöä sitä mukaan, kun asiakas kokee ne tarpeelliseks ja toimittaa käännettyä sisältöä.

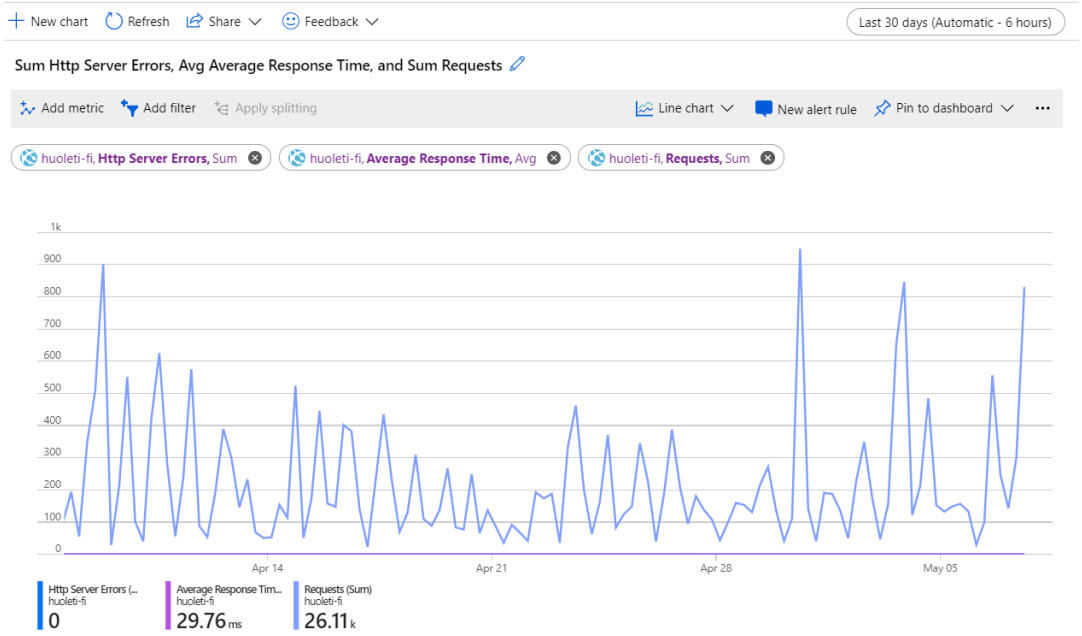
Asiakkaalle tuotetiin myös sivuston ylläpitokäyttöliittymän tavoin hallintapaneeli yhteistyöjärjestöjen tietojen-, oikeusien- sekä asiakaan sovellukssa järjestöjen kautta toimivien vapaaehtoishenkilöihen hallintaan. Suunnitteilla on myös monikielinen sivusto ja teeman uusiminen sovelluksen ulkoasun uudistuessa, mutta todennäköisesti nämä toteutukset kasvavat omiksi projekteikseen.

5.5 Sivuston suorituskyky

Sivuston suorituskyky ja nopeus todettiin tärkeäksi niin asiakkaan puolesta kuin myös kehittäjäeettiseltä pohjalta. Tämä mielessä, päädyttiin käyttäjän selaimelle mahdollisimman kevyeeseen toteutukseen: Sivusto koostetaan palvelimella valmiiksi dokumentiksi, eikä frontend kehitykseen sisällytettäisi Javascript -viitekehyksiä, sillä javascriptin tarve sivustolle on minimaalista. Näin käyttäjän selaimen avaama sivu olisi mahdollisimman valmis jo pyyntö hetkellä ja ladattavat tiedostot sivun näyttämäseksi olisivat mahdollisimman kompakteja vaatien mahdollisimman vähän suorituskykyä päätelaitteelta ja verkkoyhteydeltä.

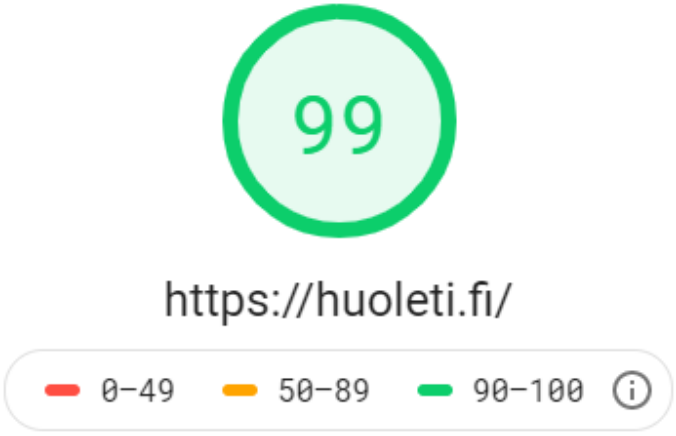
App servicen etuna suorituskyvyn toteumalle verraten pelkkään palvelin koneeseen, joka koodiamme palvelisi, App service koostaa automaattisesti sisäisen infratruktuurinsa palvelemaan sivustoa suurille yleisöille varmentaen näin suorituskykyä entisestään.

Suorituskykyä on mitattu Googlen tarjoamalla työkalulla, joka vertaa sivuston suorituskykyä yhtiön parhaiden käytäntöjen toteumaan, kuin myös tarkisteltu Azuren App servicen tuottaman statistiikan kautta positiivisin tuloksin. Azuresta tulostettu palvelimen vastaustatistiikka osoittaa, että vaihtelevasta pyyntömäärästä huolimatta palvelin on alkanut reagoida pyyntöihin keskimäärällisesti 30 millisekunnin kuluessa pyynnön lähetyksestä.



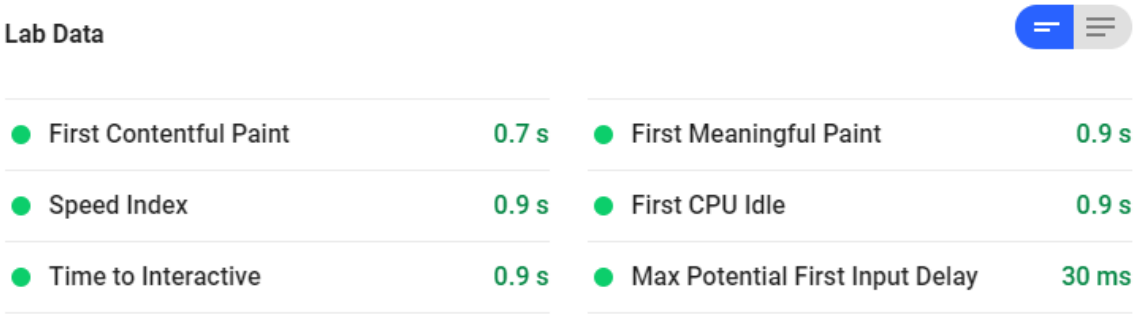
Kuva 5: huoleti.fi App Servicen vastausstatistiikkaa Azuresta

Googlen sivustonopeustesti analysoi annetulla sivustolla toteutuvat, googlen parhaaksi käytännöksi katsomat toimet ja toteumat. Pisteytys 90-100 välille on ehdoton vaatimus projektissa, jossa painotetaan sivuston nopeutta. Raskaissa sovelluksissa voidaan tietoisesti nopeudesta karsia yhteisymmärryksessä asiakkaan kanssa.



Kuva 6: huoleti.fi pisteytys Googlen sivustotestissä

Pisteytys koostuu parhaiden käytäntöjen toteuman lisäksi palvelimen suorituskyvystä. Kuvan 7 taulukossa esitetään kuusi tärkeää ominaisuutta sivuston nopean vastauksen puolesta.



Kuva 7: huoleti.fi lataus statistiikkaa Googlen sivustotestissä

First Contentful Paint eli kauanko kestää, että sivustolla näytetään lainkaan sisältöä. Sivuston latautuessa on sisällön esittämisen kannalta tärkeää pystyä näyttämään sisältöä jo ennen, kuin sivusto on kokonaan latautunut.

First Meaningful Paint eli kauanko kestää, että valtaosa sivuston näkyvästä sisällöstä on esitettävissä. Missä sisällön näyttäminen on heti on oleellista, täytyy sivuston myös latautua nopeasti kokonaisuutena pitääkseen käyttäjän mielenkiinnon yllä.

Speed Index kertoo kauanko sivustolla kokonaisuudessaan kestää latautua.

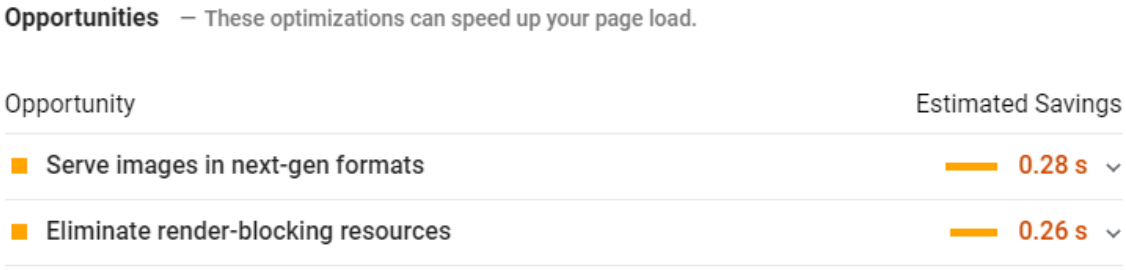
First CPU Idle keroo kauanko kestää, että päätelaitteen prosessorin säie on ensimmäisen kerran vapaana vastaanottamaan käyttäjän interaktiota sivuston kanssa.

Time to Interactive kertoo puolestaa, kauanko kestää että sivu on kokonaisuudessaan valmis käyttäjän interaktiolle. Ennen tätä tapahtuvat interaktiot näkyvät käyttäjälle reagoimattomana käyttöliittymänä.

Max Potential First Input Delay kertoo pahimman mahdollisen viiveen keston, mitä käyttäjä mahdollisesti kokee. Käyttäjät huomaavat yli 100 millisekuntia kestävät viiveet, joten kaikki sen alittava on hyväksyttävää.

Yksi sivuston tavoitteista oli saada latausaika alle kolmeen sekuntiin. Google muiden internetjättien mukana suosittelee sivun latausajaksi maksimissaan kahta tai kolmea sekuntia. Asiaa tutkiessa on huomattu, että yli kahden sekunnin latausajalla puolet käyttäjistä lakkaavat odottamasta ja kolmen sekunnin ylittyessä jopa 80% käyttäjistä luopuu sivulla vierailusta. (Hosting Manual, 2018)

Missä edellämainittu kuvan 7 data on aina vakio ja kaikille sivuille yhteinen tavoite, tarjoaa googlen sivustotesti myös sivustokohtaisia kehitysehdotuksia. Alla olevassa kuvassa on listattuna analyysin mielestä kriittisimmät korjauskohteet sekä arvio, paljonko toimenpide nopeuttaisi sivua.



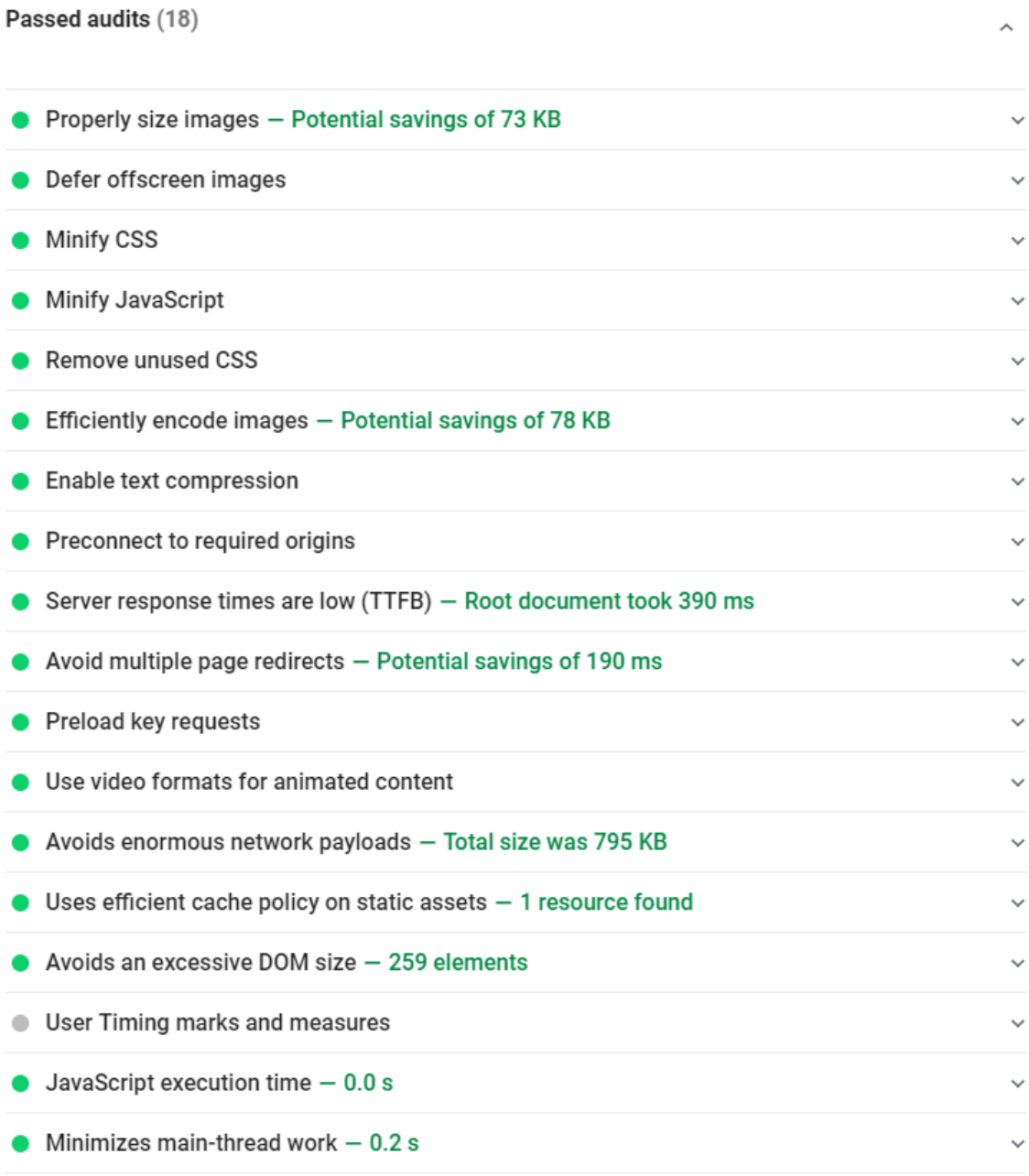
Kuva 8: huoleti.fi kehitysehdotuksia Googlen sivustotestissä

Asiakkaan sivusto saa ensimmäiseksi kehitysehdotukseksi sivustolla käytettävien kuvien tiedostomuodon päivittimisen verkkoon sopivampaan formaattiin. Tällä hetkellä sivusto käyttää kompaktiksi pakattuja JPG formaatin kuvia. Uuden sukupolven kuvaformaatit ovat huomattavasti paremmin pakattuja ja optimoitu säilyttämään kuvanlaatua kompressoinnista huolimatta.

Tuki uudemman sukupolven formaateille on kuitenkin valitettavan selain kohtaista, jotta sivusto olisi edes kaikilla moderneilla selaimilla yhtenäinen tulisi kuvia kääntää perusmuodon lisäksi ainakin kolmelle eri formaatille, jonka jälkeen tuottaa toiminnallisuus lataamaan kuva oikeassa formaatissa selaimen perusteella. Tämä olisi täysin tehtävissä, mutta työmäärää suhteessa saavutettuun hyötyyn ei ainakaan toistaiseksi koeta tarpeeksi arvokkaaksi kehityksen eikä asiakkaan puolesta.

Alempi suositus huomattaa, että sivustolle ladataan ulkoisia resursseja ennen sivuston rakennetta. Ulkoisen sisällön lataaminen ennen rakenteen esittämistä on lähtökohtaisesti huono käytäntö, mutta yleensä ennen rakennetta ladatavat rerussit ovat tyyli- tai fonttitiedostoja.

Pisteytyksen puolesta olisi helppo siirtää ladattavat tyylit sivuston loppuun muiden resurssien rinnalle, mutta varsinkin asiakkaan tapauksessa tyylit koetaan erittäin oleellisena osana sivustoa ja niiden hetkellinen puuttuminen aiheuttaisi suttuisen ensikokemuksen sivustosta. Kun rakenne ladataan ennen tyylejä, näkyy sivusto tyylittelemättömänä html -dokumenttina. Koska tämäkään huomio ei säästä merkittävästi sivuston latausaikaa ei optimointitarvetta koeta.



Kuva 9: huoleti.fi toteutettuja optimointeja Googlen sivustotestissä

Googlen sivustotesti kiittää myös sivustolla toteutuvista optimoinneista ja parhaankäytännön toteumista. Missä testin ehdottamat kehitysehdotukset auttavat kehittäjä tuottamaan parhaan mahdollisen tuotteen, auttaa tämä onnistumisien lista asiakasta käsittämään työnmäärää, mitä parhaiten optimoidun sivuston tuottaminen on vaatinut ja mistä kaikesta hän maksaa. Asiakkaan ei tietenkään tarvitse ymmärtää mitä jokainen kohta tarkoittaa, mutta nähdessään kokonaiskuvan sivuston laadusta luo se luottamusta asiakkaan ja kehittäjän välille.

6 Loppuanalyysi

Työtä analysoidessa tärkeiksi tekijöiksi rajoitetaan: Asiakkaan kokemukset työn kulusta ja toteumasta, itse sivustosta saatu mahdollinen käyttäjäpalaute, projektiryhmän kokemukset työn kulusta ja onnistumisesta sekä tuotetun artefaktin toteuma vaatimusmäärittelyyn nähden.

6.1 Kokemukset

Kun sivusto oli ollut tuotantokäytössä noin kuukauden, käytiin asiakkaan kanssa läpi sivustoa kokonaisuutena ja miten se on vastannut haluttua. Sivuston toiminnallisuuksissa tai ulkoasussa ei ole koettu puutteita. Joitakin koko- tai fontti asetuksia on muuteltu ylläpitötöiden ohessa ja mobiilissa esitettävää navigaatiota on muokattu hieman silmään iskevämmäksi ja sivustosta erottuvammaksi.

Ylläpitotyöt eivät ole aiheuttaneet kohtuuttomia laskuja, vaan oikeastaan odotetun minimin verran. Toistettava rakenne ja teema ovat tässä isossa roolissa ja sivuston suunnitteluun investointi selvästikkin kannatti.

Myös dynaamisen sisällön tuotanto on ollut helppoa ja vaivatonta. Blogi postausten ja uutislinkkien määrä kasvaa kuitenkin kokoajan ja näiden hallintointi varsinkin muokkausnäkymässä voi tulevaisuudessa muotoutua haastavaksi. Mikäli tähän syntyy muun kehitystyön ohella ehdotuksia, otetaan niitä mielellään vastaan.

Sivustosta on tullut käyttäjäpalautteen muodossa pieniä, mutta oleellisia huomioita. Esimerkiksi ”Lataa sovellus” -linkit olivat vain normaalina tekstintä muun sisällön tavoin. Dominoiva otsikko aiheutti ymmärrysongelmia käyttäjien keskuudessa ja herätti hämmenystä, kun otsikkoa klikatessa mitään ei tapahtunukkaan.

Ratkaisuksi tähän päädyttiin korostamaan ulkoisiin sijainteihin viittaavia linkkejä kohotetun näppäimen tyylin mukaiseksi. Nyt linkit erottuvat huomattavasti selkeämmin eikä sekavuudesta ole saatu enempää palautetta.

Kehitystiimi on myös kokenut sivuston koodin helposti ylläpidetteväksi sekä teeman ja rakenteen helposti toistettavaksi. Ylläpitotöiden yhteydessä on kuitenkin huomattu sivuston tyylitiedoston paisuvan hankalasti ylläpidettäväksi läjäksi sääntöjä. Tulevaisuuden kannalta olisi ollut ylläpidon kannalta selkeämpää käyttää mahdollisesti SASS – kielilaajennusta CSS koodissa ja sen käyttöön suositeltua 7-1 rakennetta, missä tyylit hajoitetaan erillisiin kansioihin käyttötarkoituksen mukaisesti omiksi tiedostoikseen. (Hugo Giraudel, 2018)

6.2 Artefaktin ja vaatimusmäärittely

Vaatimusmäärittely koosteettiin kolmivaiheisesti pohjaten teknologiavalinnat asiakaslähtöisesti suunniteltuun kokonaisuuteen ja tämän jälkeen sivuston palvelu asiakkaalle sopivimpaan valittuja teknologioita tukevaan ympäristöön.

Asiakasta haastateltiin aiheesta, mitä tietoa sivuston tulisi tarjota ja miltä sivuston tulisi näyttää. Lisäksi asiakkaalta tiedusteltiin heille oletettuja tyypillisiä skenaarioita sivuston käyttöön liittyen.

Sivustosta haluttiin mahdollisimman paljon asiakkaan sovelluksen ulkoasun mukainen yhteinäisen käyttökokemuksen laajentamiseksi. Sivuston tuli kyetä palvelmaan sisältöä myös suurien kävijäpiikkien aikaan ja aueta selaimessa mahdollisimman nopeasti, jotta kaikki asiakkaan palvelusta kinnostuneet pääsevät halutessaan siihen tutustumaan.

Taiteellisten seikkojen toteuman arviointi vaatimusmäärittelyyn nähden ei ole kovinkaan mahdollista, vaan sovelluksen ilmeen ja teeman toteumaa käsitellään tarkemmin asiakkaan antaman palautteen kautta. Faktapohjaisesti sovelluksen teeman ja tuntuman mukailu voidaan kuitenkin todeta onnistuneeksi, sillä sivusto mukailee asiakkaan sovellusta ja muuta materiaalia niin typografisesti, kuin värien ja kuvienkin osalta.

Teknoloagiallisesti työ toteutettiin juuri niin, kuin vaatimusmäärittelyssä oli suuniteltu. Mistään toiminnallisuudesta ei karsittu, mutta on pohtimisen arvoista, onko toteutuksessa mahdollisesti käytetty käyttötarkoituksen nähden turhaa teknologiaa.

Sivustosta haluttiin tehdä mahdollisimman helposti laajennettava ja jatkokehitettävä. Tämän johdosta päädyttiin käyttämään mm. oikeaa tietotakantaa ja koodaamaan itse palvelin. Vaikka näille olisi ollut vaihtoehtoisia, ehkäpä kevyelle sivustolle mittasuhteeltaan sopivampia vaihtoehtoja.

Sivustoa ylläpitäessä itse koodattu palvelin on osoittautunut eduksi. Node.js varaan rakennettu Express.js -viitekehystä hyödyntävä palvelin on logiikaltaan todella yksinkertainen ja mahdollistaa nopeasti monimutkaistenkin asioiden tuottamisen palvelimelle sekä helpon hallinnoinnin sivuston polkujen nimistä ja sisällöistä.

Nämä ominaisuudet ovat joka ylläpitotyön yhteydessä osoittautuneet erittäin hyödylliseksi ja ovat säästäneet työtunteja merkittävästi. Lisäetuna kaikki muokkaukset löytyvät myös saman versiohallinnan alta, joten mahdollisten ongelmien puolesta palauttaminen on hyvin suoraviivaista. App Servicelle on suoraan Azuren kautta mahdollista ohjelmoida useita toimintoja,

mutta kokonaisuuden hajoittaminen usealle tasolle koettiin vaivalloisesi ja ylläpidolle haastavaksi.

Sivustoa varten tuotettu tietokanta, jota isännöitään Azure Cosmos DBssä MongoDB -viitekehyksen kautta, on taas koettu mahdollisesti tarpeettomaksi. Tietokantayhteydet aiheuttavat kukauden aikana keskimäärin noin 50€ juoksevia kulujua. Vaikka kyseessä on vertaillen pieni summa yhteyskuluksi on se sivuston kokonaiskustannuksia ajatellen turhan iso osa, sillä asiakkaan App Serviceitä palveleva Azuren palvelin itsessään kustantaa asiakkaalle saman verran.

Koska asiakkaan sivuston tietokannassa ei säilötä yksityistä dataa tai henkilötietoja, vaan pelkästään sivustolla esitettävää julkista dataa, on perusteltua pitää tätä kuluerää turhana. Tietokanta toiminnallisuus olisi mahdollista korvata palvelimen toiminnallisuudella lukea ja kirjoittaa tiedostoja .json -formaattiin, jota tietokantakin käyttää. Tällöin tiedoston luku voitaisiin säilyttää palvelimella muiden tiedostojen ohella aiheuttamatta tietokannan lukukustannuksia. Asia on tuotu asiakkaan tietoisuuteen ja muutoksen tarpeellisuutta arvioidaan tulevien töiden yhteyteen.

7 Lähteet

Ken Schwaber, Jeff Sutherland, The Scrum Guide, 2016. Tulostettu 3.12.2017. https://www.scrum.org/re-sources/scrum-guide

SANS teknologiainstituutti: IT Code of Ethics, 24.4.2004. Tulostettu 19.11.2017. https://www.sans.org/security-resources/ethics

Illinoin teknologiainstituutti: Code of Ethics for IT professionals, 24.10.2011. Tulostettu 19.11.2017. http://ethics.iit.edu/ecodes/node/3115

O. Nykänen, A. Tervakar, Tampere university of technology, Verkkosisällön saavutetavuus ohjeet. 16.02.2011. Tulostettu 3.12.2017. https://www.w3.org/Translations/WCAG20-fi/

Microsoft Azure, Web Apps, 2019. Tulostettu 13.5.2019. https://azure.microsoft.com/en-us/services/app-service/web/

Microsoft Azure, Azure Cosmos DB, 2019. Tulostettu 13.5.2019. https://azure.microsoft.com/en-us/services/cosmos-db/

Microsoft Azure, Monthly Azure credit for Visual Studio Enterprise (BizSpark) subscribers, 2019. Tulostettu 13.5.2019. https://azure.microsoft.com/en-us/offers/ms-azr-0064p/

Microsoft Azure, Use an SSL certificate in your application code in Azure App Service, 12.1.2017. Tulostettu 13.5.2019 https://docs.microsoft.com/en-us/azure/app-service/app-service-web-ssl-cert-load

Microsoft Azure, Local Git Deployment to Azure App Service, 5.6.2018. Tulostettu 13.5.2019. https://docs.microsoft.com/en-us/azure/app-service/deploy-local-git

Microsoft Azure, Tutorial: Map an existing custom DNS name to Azure App Service, 18.6.2018. Tulostettu 13.5.2019. https://docs.microsoft.com/en-us/azure/app-service/app-service-web-tutorial-custom-domain

Microsoft Azure, Buy and configure an SSL certificate for Azure App Service, 16.10.2018. Tulostettu 13.5.2019. https://docs.microsoft.com/en-us/azure/app-service/web-sites-purchase-ssl-web-site

Microsoft, Description of the Secure Sockets Layer (SSL) Handshake, 17.5.2018. Tulostettu 13.5.2019. https://support.microsoft.com/en-us/help/257591/description-of-the-secure-sockets-layer-ssl-handshake

Node.js Foundation, About Node.js®.Tulostettu 13.5.2019. https://nodejs.org/en/about/

Node.js Foundation. Fast, unopinionated, minimalist web framework for Node.js.Tulostettu 13.5.2019. https://expressjs.com/

EJS, What is EJS? Tulostettu 13.5.2019. http://ejs.co/#about

MDN web docs, HTML: Hypertext Markup Language, 18.5.2019 . Tulostetu 28.5.2019. https://developer.mozilla.org/en-US/docs/Web/HTML

MDN web docs, JavaScript, 21.5.2019. Tulostettu 28.5.2019. https://developer.mozilla.org/en-US/docs/Web/JavaScript

MDN web docs, CSS, 28.5.2019. Tulostettu 28.5.2019. https://developer.mozilla.org/en-US/docs/Web/CSS

Jeremy Wagner, Google Developers, Why performance matters, 1.5.2019. Tulostettu 14.5.2019. https://developers.google.com/web/fundamentals/performance/why-performance-matters/

Anoop Raveendran, Medium, JavaScript Event Loop Explained, 27.12.2017. Tulostettu 14.5.2019. https://medium.com/front-end-weekly/javascript-event-loop-explained-4cd26af121d4

Opetushallitus, SWOT-analyysi. Tulostettu 14.5.2019. https://www.oph.fi/saadokset_ja_ohjeet/laadunhallinnan_tuki/wbl-toi/menetelmia_ja_tyovalineita/swot-analyysi

Visual Paradigm, What is a Sprint in Scrum? Tulostettu 14.5.2019. https://www.visual-paradigm.com/scrum/what-is-sprint-in-scrum/

Steven J. Vaughan-Nichols, ZDNet, Edge in name only: Android and iOS Edge web browser, 8.10.2017. Tulostettu 14.5.2019. https://www.zdnet.com/article/edge-in-name-only-android-and-ios-web-browser/

Hosting manual, 3 Seconds: How Website Speed Impacts Visitors and Sales, 4.9.2018. Tulostettu 14.5.2019. https://www.hostingmanual.net/3-seconds-how-website-speed-impacts-visitors-sales/

Google Developers, Serve Images in Next-Gen Formats, 1.3.2019. Tulostettu 14.5.2019. https://developers.google.com/web/tools/lighthouse/audits/webp?utm_source=lighthouse&utm_medium=unknown

Hugo Giraudel, Architecture, The 7-1 Pattern, 2018. Tulostettu 14.5.2019. https://sass-guidelin.es/#the-7-1-pattern

Arto Eskelinen, Tommi Kemppi, Maarit Laanti, Lare Lekman, Jukka Lindström, Karoliina Luoto, Towo Toivola, Pentti Virtanen ja Martin von Weissenberg. Suomenkielinen scrum-sanasto, 2018. Tulostettu 13.5.2019. https://scrumwell.files.wordpress.com/2018/03/suomenkielinen-scrum-sanasto-2018-v1-0.pdf

Wikipedia, JSON, 7.3.2019. Tulostettu 15.5.2019. https://en.wikipedia.org/wiki/JSON

8 Kuvat

9 Liitteet

Liite 1: Tuotoksen tilausjono

1. Suunnittele sivuston teema: Sivustokokonaisuuden teeman tulee olla sovelluksen, sekä yrityksen esittelymateriaalin mukainen.
   Vaaditut valmiit: <tyhjä>

2. Luo aloittussivu: Tuota suunnitellun teeman mukainen aloittussivu, jossa sivuston teema ja komponentit ovat rakennettu toistokelpoiseksi muille sivuille.
   Vaaditut valmiit: 1, 8, 12

3. Tuota /tuki -sivu: Sivustolle kopioidaan aloitissivun rakenne ja päivitetään tekstisiältö oikeaksi. Lisäksi tuota haitari -komponentti usein kysytyt kysymykset -osiolle.
   Vaaditut valmiit: 2

4. Tuota /huolet -sivu: Sivustolle kopioidaan aloitissivun rakenne ja päivitetään tekstisiältö oikeaksi. Lisäksu tuotetaan ”kääntökortti” -näkymä huoli -listaukselle. Kortin oletuspuolella näytetään huoliryhmä, kääntöpuolella kaikki ryhmään kuuluvat huolet.
   Vaaditut valmiit: 2

5. Tuota /yhteisosomit -sivu: Sivustolle kopioidaan aloitissivun rakenne ja päivitetään tekstisiältö oikeaksi.
   Vaaditut valmiit: 2

6. Tuota /blogi -sivu: Sivustolle kopioidaan aloitussivun rakenne. Sisällöksi tuodaan lista tietokantaan tallennetuista blogi -teksteistä järjestettynä ajallisesti uusimmasta vanhimpaan jokaisesti postauksesta näytetään otsikko, johdanto ja kuva. Lisäksi tuotetaan /blogi/<id> -sivu, jolle voi avata kokonaisuudessaan yksittäisen blogi postauksen.
   Vaaditut valmiit: 2,16

7. Tuota /uutiset -sivu. Sivustolle kopioidaan aloitussivun rakenne. Sisällöksi tuodaan lista tietokantaan tallennetuista uutislinkeistä ja listataan ne ajallisesti uusimmasta vanhimpaan. Uutislinkkejä näytetään kerralla maksimissaan kuusi kappaletta. Loput sivutetaan. Uutislinkkien lisäksi sivulla näytetään yrityksen facebook ja twitter syötteet.
   Vaaditut valmiit: 2,16

8. Tuota sivustolle CSS teema Designerin toimittaman ulkoasun pohjalta.
   Vaaditut valmiit: 1

9. Tuota funktio, mikä animoi yksittäisen tekstilohkon ilmestymisen ruudulle sivustoa selatessa artikkelin tasolle. Animoi teksti liukumaan hieman ylöspäin ja läpinäkyvästä näkyväksi. Toiminnallisuus toistaiseksi kaikkiin tekstilohkoihin.
   Vaaditut valmiit: 2

10. Tuota node.js -palvelin. Tuota perus toiminnallinen Node.js palvelin, johon voidaan lisätä NPM kirjastoja. Anna palvelimelle portti, josta sitä voi kutsua ja lähetä "hello world" -vastaus selaimelle.
    Vaaditut valmiit: 17

11. Lisää express -kirjasto palvelimen käyttöön. Muuta "hello world" -vastausta antava komento käyttämään express -kirjaston funktioita.
    Vaaditut valmit: 10

12. Lisää EJS -kirjasto palvelimen käyttöön. Tee kirjaston dokumentaation mukaiset kansiorakennemuutokset palvelimen juureen ja tuota esimerkkisivu dokumenttimoottorilla.
    Vaaditut valmiit: 10,11

13. Päivitä palvelin tuottamaan halutut sivut kohdista 2-7 EJS -moottorilla kun palvelimeen osoitettu polku vastaa suunniteltua. Tee ”sivua ei löydy” sivu ja aseta palvelin vastaamaan se, jollei haluttua sivua löydy.
    Vaaditut valmiit: 12, 2-7

14. Perusta MongoDB -tietokanta Azuren Cosmos DB -ympäristöön. Aseta tietokantaa testidata ja ohjelmoi palvelin käyttämään tietokantayhteyttä.
    Vaaditut valmiit: 10

15. Lisää MongoDB -tietokantaan asiakkaan olemassaoleva data blogista ja uutislinkeistä parhaaksi katsomallasi tavalla: Wordpress export tai käsin kopiointi.
    Vaaditut valmiit: 14

16. Tuo blogi- ja uutisdata tietokannasta niille kuuluville sivuille. Järjestä data uusimmasta vanhimoaan ja erottele kielen mukaan: EN -sivulle englannin kieliset, FI -sivulle suomen kieliset
    Vaaditut valmiit: 15, 13
17. Perusta kohdekansio projektille ja aseta kansio GIT -versiohallinnan alaiseksi.
    Vaaditut valmiit: none

18. Testaa sivustokokonaisuus kehitysympäristössä
    Vaaditut valmiit: 1-17

19. Perusta AppServicet FI, EN ja Admin sivustoille asiakkaan tunnuksien alle Azure portaalissa ja aktivoi GIT julkaisu mahdollisuus.
    Vaaditut valmiit: none

20. Julkaise toimivaksi todettu sivustokokonaisuus sille kuuluvaan AppServiceen
    Vaaditut valmiit: 18, 19

21. Ohjaa asiakaan ostamat DNS nimet osoittaamaan sille kuuluvaan App Serviceen Azure -ympäristössä.
    Vaaditut valmiit: 19

22. Ota käyttöön SSL -suojaus. Hanki Azure -portaalin kautta App Service Sertificate ja osoita se sille kuuluvalle App Servicelle.
    Vaaditut valmiit: 19,21

23. Kopioi projektin pohja ylläpitopaneelia varten versiohallinasta mahdollisimman läheltä minivaatimusten toteutumista. Poista turhat resurssit.
    Vaaditut valmiit: 12, 14

24. Tuota Create -, Update ja Destroy -toiminnallisuudet palvelimelle sekä blogipostausten että uutislinkkien hallintaan. Read toiminto on toteutettu jo julkisivusto vaiheessa.
    Vaaditut valmiit: 23

25. Tuota ylläpitopaneelin käyttöliittymään lomakkeet blogin ja uutislinkkien CRUD toiminnallisuuksien kutsumiselle.
    Vaaditut valmiit: 24

26. Tuota kirjautumis -sivu ylläpitopaneelin käyttöliittymään.
    Vaaditut valmiit: 29

27. Ohjelmoi ylläpitopaneelin palvelin vaatimaan kirjautuminen kaikkien CRUD toimintojen kohdalla. Käyttäjätunnus ja salasana voidaan kovakoodata palvelimen muuttujiksi. Kirjautuminen vermennetaan evästeillä.
    Vaaditut valmiit: 23

28. Testaa ylläpitopaneelin toiminnallisuudet ja julkaise se sille osoitettuun App Serviceen kohtien 19-22 ohjenuoraa noudattaen.
    Vaaditut valmiit: 25-28

Liite 2: Asiakkaan vaatimusmäärittely haastattelu

1. Mitä sivustokokonaisuuden tulee kattaa?
   Huoleti tarvitsee sekä suomen-, että englannin kieliset kotisivut, jossa esitellään yritys ja tarjotaan ajankohtaista tietoa sekä yrityksestä, että sovelluksesta. Näiden lisäksi tarvitaan yhteystyökumppaneille sivusto, jossa tarjotaan esimerkiksi kumppanikohtaisia käyttöönottokoodeja sovellukselle.

2. Mitä kotisivujen tule pitää sisällään?
   Meillä on tästä speksi, jonka pohjalta haluttaisiin seuraavat sivut:

   Koti -sivulla näytettään yrityksen viisi ”pää tekstiä” mitkä esittelevät sovellusta pääpiirteittäin sekä yhteystiedot, linkki tuotevideoon ja sovelluksen latauslinkit. Itseasiassa olisi hyvä, jos yhteystiedot ja sen jälkeen mainitut asiat olisivat näkyvillä, vaikka joka sivulla. Tekstit näihin pääartikkeleihin näkyy erikseen sivuston speksissä.

   Tuki -sivulla tarjotaan käyttötukea sovelluksen käyttäjille ja käyttöönotolle: Kolmen kohdan käyttöönotto malli, sekä usein kysyttyjä kysymyksiä. Laitetaan sivulle myös ladattavaksi printattava käyttöönotto-opas, rekisteliseloste sekä meidän yrityksen tuen yhteystiedot.

   Huolet -sivulla autetaan sovelluksen käyttäjää löytämään oma huoli-ryhmänsä ja selvennetään mitä sovellusta käyttäessä on ”huoli”. Huolista on meillä toistaiseksi 20 yläkategoriaa ja niissä 1-10 alaluokkaa. Sivulla tulee olla listattuna kaikki huoliryhmät ja niiden huolet mutta oma huoli tulee löytyä helposti.

   Yhteisönormit -sivulla esitetään sovelluksen käyttökieltä ja -sääntöjä: Toisen kunnioitusta, rauhaa ja rehellisyyttä. Lisäksi annetaan tietoa vapaaehtoisena toimimisesta. Blogi -sivulle halutaan pystyä kirjoittelemaan uutisia ja blogeja yrityksestä samaan tapaan, kuin nykyisellä wordpress -sivustolla.

   Uutiset ja Some -sivulla halutaan listata uutisia muista lähteistä, joissa huoleti on mainittu. Lisäksi esitetään yrityksen Twitter ja Facebook feedit.

   Sivustosta on erikseen tarkempi speksi sisällöstä. On erityisen tärkeää, että sivusto on näyttävä ja sovelluksen tyylilinjan mukainen. Vastaava sisältö tulee englanninkieliselle sivustolle, mutta kaikki sisältö ei vielä ole valmiina, joten englannin sisältöä julkaistaan lisää sitä mukaan kuin sitä tulee.

Sivuja varten on ostettuna huoleti.fi ja huoleti.com -domainit, jotka voidaan kääntää osoittamaan sivustoon kun nämä ovat valmiit.

3. Miten haluatte jaotella sivustolla esitetyn tiedon? Mitä tuodaan tietokannasta ja mitä on sivulla valmiiksi?

   Näin aluksi valtaosa tekstistä voisi olla staattisena suoraan html -tiedostoon kirjoitettuna. Haluamme itse mahdollisuuden kirjoittaa blogitekstejä ja lisätä uutislinkkejä, mutta muut tekstit tulevat pysymään kutakuinkin vakioina.

4. Minkälaisen ulkoasun tahtoisitte sivustollenne?
   Sivuston ulkoasun suhteen ei ole varsinaista suunnitelmaa. Meillä on postereita ja flyereita tulossa sovelluksen käyttöönottoon käyttäjille jaettavaksi. Olisi upeaa, jos näillä olisi jokin yhteinen ulkoasuteema.

   Esimerkiksi haltu.fi ja fastmonkeys.fi sivuilla on kivat yksinkertaisen kaupalliset ulkoasut. Voisitteko suunnitella jotain tämän mukaista sivustolle?

5. Millainen sivusto yhteistyökumppaneita varten tulee?
   Sovellusta tarjotaan käyttöön esimerkiksi sairaaloiden / kaupunkien kautta, joiden kanssa on yhteistyösopimus. Tarvitsemme sivuston, jossa tarjotaan ainakin näin aluksi, kumppanikohtainen käyttöönottokoodi, käyttöönotto ohjeet ja tietoa loppukäyttäjälle.