Haaga-Helia
ammattikorkeakoulu Oy

Ohjelmistokehittäjän työ pienessä saksalaisessa ohjelmistoyrityksessä

Juho Rannila

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2018

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

19.5.2018

Tekijä(t)
Juho Rannila

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Ohjelmistokehittäjän työ pienessä saksalaisessa ohjelmistoyrityksessä

Sivu- ja liitesivumäärä
67 + 0

Opinnäytetyön otsikko englanniksi
Software developers work in a small German software company

Opinnäytetyö on kirjoitettu portfoliomaisessa päiväkirjamuodossa. Työssä kuvataan opiskelijan työtä ohjelmistokehittäjänä saksalaisessa ohjelmistoyrityksessä. Työ koostuu kymmenestä seurantaviikosta, jossa kirjoitetaan merkintöjä työpäivistä. Työpäivät ovat epäsäännöllisiä työsuhteen osa-aikaisuuden sekä opintojen takia. Seurantaviikkojen päivätekstien jälkeen on aina viikkoanalyysi, jossa käydään läpi päiväteksteissä mainitut tehtävät ja selitetään lähteitä käyttäen niissä tehtyjen ratkaisujen toiminnallisuus.

Opinnäytetyö suoritetaan yrityksessä, joka kehittää monipuolisia IT-ratkaisuja asiakkailleen. Tämän lisäksi yrityksellä on verkkokauppa, jossa myydään sirukortteja ja niitä hyödyntäviä tuotteita. Opiskelijalla on aiempaa kokemusta yrityksessä työskentelystä jo vuoden ajan. Työtehtäviin kuuluu pääasiassa ohjelmistojen ja WWW-sivustojen kehittäminen sekä ajoittainen asiakaspalvelu. Viikkoanalyysien tavoitteena on edistää opiskelijan työtä etsimällä tapoja kehittää parempia työtapoja ja menetelmiä. Tätä kautta myös yritys hyötyy opiskelijan ohjelmistokehittäjän työssä vaadittujen taitojen lisääntymisestä.

Seurantaviikkojen jälkeen on nähtävissä opiskelijan taitotason kehittyminen. Erilaiset työtehtävät ovat antaneet kokemusta laaja-alaisesti, sillä yrityksen pieni koko on pakottanut opiskelijan toimimaan eri tason tehtävissä. Työn tuloksena syntyy painotalolle WordPress -sisällönhallintajärjestelmällä tehty WWW-sivusto sekä AngularJS -ohjelmistokehyksellä tehty työpaikkaruokalan ruokalistaohjelma.

Asiasanat
Kehittäminen, ohjelmointi, sisällönhallinta, WWW-sivustot

Sisällys

1 Johdanto

Tämä opinnäytetyö tehdään päiväkirjamuotoisesti 10 viikon ajalta. Päiväkirjamerkinnät alkavat keskiviikkona 20.9.2017 ja viimeinen merkintä kirjataan perjantaina 24.11.2017. Päiväkirjatyyppisessä opinnäytetyössä tehdään merkintöjä jokaiselta työpäivältä määritellyn ajanjakson aikana ja viikon lopussa laaditaan viikoittainen analyysi.

Ajanjakson aikana vaadittavat työtehtävät pitää sisällään webkehittämisen ja asiakaspalvelun. Työssä perehdytään kehitystyöhön ja sen haasteisiin AngularJS -ohjelmistokehystä hyödyntäen. Tämän lisäksi tutustutaan WordPress -sisällönhallintaohjelmaan, jota käytetään työssä yksinkertaisten nettisivustojen tekemiseen. Yritys on varsin pieni, joten kehittäjältä vaaditaan motivaatiota, itseohjautuvuutta sekä täsmällisyyttä – unohtamatta ohjelmointitaitoja sekä yleisiä työpaikalla vaadittuja taitoja, kuten kommunikointia sekä halua oppia. Kehittäjä on vastuussa kaikista kehitykseen liittyvistä vaiheista.

Aloitin ohjelmoinnin vasta korkeakoulussa enkä opinnoissani päässyt tutustumaan AngularJS -kehykseen, vaan opin sen käytön yrityksessä ja vapaa-ajallani. WordPress oli samoin minulle vielä tuntematon, mutta sen käyttö ja oppiminen on tehty mahdollisimman helpoksi.
Oppiakseni uutta näistä kahdesta kehitysmenetelmästä päätin valita kaksi kumpaankin aiheeseen perehtyvää kirjaa. Professional WordPress: Design and Development -kirja on mielestäni hyvä valinta kaltaiselleni keskitason osaajalle, joka on jo tehnyt muutamia sivustoja ja tietää perusteet. WordPress tunnetaan käyttäjäystävällisyydestä, mutta kehittäjän rajoittamisesta kustomoinnin suhteen. Kirja auttaa minua pääsemään seuraavalle tasolle kehitystyössä, jotta voidaan hyödyntää WordPressin ominaisuuksia edelleen.
Node.js, MongoDB, and AngularJS Web Development -kirja tutkii AngularJS -kehyksen lisäksi myös työssä käytettyä node.js -ohjelmistokehystä ja ennen kaikkea näiden kahden välistä suhdetta. Kirja sopii hyvin tasoiselleni opiskelijalle, sillä kirjassa käydään lyhyesti läpi myös JavaScript -kielen perusteet kertauksen vuoksi. Kirja vastaa mielestäni omaa taitotasoani hyvin ja jatkaa siitä mitä jo osaan.

Yritys sijaitsee Reutlingenissa, Etelä-Saksassa ja työllistää 4 henkilöä. Henkilöstöön kuuluu yrityksen perustaja ja toimitusjohtaja, kaksi vanhempaa kehittäjää sekä yksi nuorempi kehittäjä eli minä. Yritys aloitti toimintansa valmistamalla RFID -teknologiaan (Radio Frequency IDentification) eli radiotaajuiseen etätunnistamiseen perustuvia järjestelmiä, kuten sirulla varustettuja jäsenkortteja sekä esimerkiksi ruokaloissa käytettävät maksukortit, joihin voi ladata arvoa. Nykyään sillä on nettikauppa, jossa myydään tunnistamiseen liittyviä laitteita. Sen lisäksi yritys valmistaa asiakkaille räätälöityjä webohjelmia ja sivustoja.

Osallistun pääsääntöisesti vain websovellusten kehitykseen, sillä yritys pyrkii kehittymään websovellusten sekä nettikaupan suuntaan. Sen lisäksi yritys suunnittelee laajenevansa sisätilapaikannusjärjestelmiin iBeacon -sensoreita hyödyntämällä. Olen tällä hetkellä ainoa ulkomaalainen yrityksessä ja tämän hetkinen saksan kielen taitoni on eurooppalaisen viitekehyksen tasolla A2, joka ei riitä ammattimaiseen kanssakäyntiin yritystoiminnoissa. Siispä käytän töissä muiden työntekijöiden kanssa puhuessa englantia, mutta muuten yritys keskustelee sidosryhmiensä kanssa saksaksi. Olen työskennellyt yrityksessä syyskuusta 2016 asti. Aloitin työni siellä työharjoittelun merkeissä, jonka jälkeen työsuhdettani jatkettiin osa-aikaisena.

1.1 Käsitteet

Adobe Color – Sivu, joka avustaa väriteeman valitsemisessa.

AngularJS – JavaScriptiin perustuva ohjelmistokehys.

Atom – Avoimen lähdekoodin tekstieditori.

Bootstrap – Kirjasto. Käytetään sivustojen kehittämiseen.

CMS – Lyhenne englannin sanoista Content Management System, joka suomeksi on sisällönhallintajärjestelmä.

Contact Form 7 – WordPress -liitännäinen. Käytetään kontaktilomakkeen luomiseen.

Cyberduck – Tiedonsiirto-ohjelma.

DeepL – Sivusto, jossa voi kääntää kieliä.

Express – Node.js -moduuli, jota käytetään websovellusten kehityksessä.

FileZilla – Tiedonsiirto-ohjelma. Käytetään tiedon siirtämiseen palvelimeen.

Gimp – Ilmainen kuvankäsittelyohjelma. Käytetään kuvien optimoimiseen.

Git – Yrityksen käyttämä versionhallintajärjestelmä.

Grav – Web-sisällönhallintajärjestelmä, jonka erikoisuus on kevyt toimintapinta.

GTmetrix – Nettisivun latausnopeuden testaukseen tarkoitettu sivusto.

Gzip -kompressio – Tapa ladata sivusto nopeammin. Palvelin lähettää sivuston tiedostot zip-paketteina vierailijalle.

JavaScript – Komentosarjakieli, jota käytetään websovellusten kehityksessä.

Moment.js – Node.js:n kautta ladattava moduuli. Antaa kehittäjälle keinoja muokata ja käyttää aikamääreitä ohjelmassa.

MVC – Ohjelmistoarkkitehtuuri, jonka lyhenne tulee englannin kielen sanoista Model, View ja Control. Arkkitehtuurissa puretaan ohjelmisto kolmeen tasoon.

Node.js – Avoimeen lähdekoodiin perustuva JavaScript runtime-ympäristö. Käytetään yleensä yhdessä AngularJS:n kanssa.

Npm – Node Package Manager on JavaScript -komentosarjakielelle luotu paketinhallinta-järjestelmä.

RFID – Radio Frequency Identification, eli radiotaajuinen etätunnistus. Siru, joka sisältää tietoa. Tieto saadaan viemällä siru lähelle lukijalaitetta.

Visual Composer – WordPress -liitännäinen. Yksinkertaistaa sivuston kehittämistä.

WordPress – Sisällönhallintajärjestelmä, jolla voi luoda nettisivuja ja blogeja.

WP Super Cache - Sivuston latausnopeutta parantava WordPress -liitännäinen.

W3 Total Cache – Sivuston latausnopeutta parantava WordPress -liitännäinen.

XAMPP – Ohjelma, joka antaa ohjelmoijalle työkaluja kehitystyöhön lokaalissa ympäristössä.

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työn analyysi

Yrityksen varsin pienestä koosta johtuen työtehtäväni vaihtelevat suuresti kehityksen eri vaiheista, kuten alkupään suunnittelusta aina lopullisen tuotoksen julkaisuun asiakkaan käyttöönottamiseksi. Yrityksellä on yleisesti ottaen monta projektia työn alla, joista kaikki ovat eri vaiheissa ja vaativat kehitystä.
Johtuen yrityksen pienestä koosta joudun joskus tekemään täysin itse kehityksen eri vaiheet. Tämä pitää sisällään myös suunnittelun, jossa teen yleensä asiakkaalta saadun tiedon perusteella yhteenvedon vaatimuksista ja toiveista lopullisen tuotoksen sisällön suhteen. Omissa projekteissani yrityksessä jotka joudun tekemään täysin omatoimisesti, suunnittelen yleensä kaikki osa-alueet itse. Omatoimiset projektini ovat yleisesti ottaen asiakkaille tehdyt nettisivut, jotka teen WordPress -sisällönhallintaohjelmistolla. WordPress on siitä hyvä työkalu, että pystyn tekemään nopeasti hyvännäköiset ja edustavat nettisivut asiakkaallemme, jonka vaatimukset nettisivun suhteen ovat kohtalaisen tavanomaiset. Yleensä asiakkaiden toiveena on antaa heidän omille asiakkailleen tietoa yrityksestä ja heidän tarjoamistaan palveluista. Nettisivujen pääasiallinen tehtävä on saada asiakas ottamaan yhteyttä yritykseen, joten tämä on tärkein asia pohtia ja huomioida nettisivuston ulkonäköä ja asettelua pohtiessa. Tällaista osiota nettisivussa kutsutaan nimellä ”call to action”. Kyseisessä osiossa on linkkejä nettikauppaan tai kontaktilomakkeeseen.

Kehityksen eri vaiheissa käymme keskustelua asiakkaan kanssa aina tasaisin väliajoin yhdessä pidetyissä kokouksissa. Yrityksellämi on neuvottelutila, joka on piirtoheittimellä ja valkokankaalla varustettu. Täällä käymme läpi yleiset asiat aina projektin alussa, keskivaiheella ja tarpeen vaatiessa myös lopussa. Yleensä en itse osallistu suurempien projektien kokouksiin. Tämä johtuu asemastani yrityksessä sekä tämän hetken saksan kielen taitoni tasosta joka ei ole vielä riittävä. Siispä saan yleensä asiakkaan palautteen yrityksen toimitusjohtajalta tai vanhemmalta ohjelmoijalta.

Jotkut asiakkaistamme puhuvat myös englantia, jolloin on myös minun mahdollista keskustella suoraan heidän kanssaan. Olen pitänyt itse myös kokouksia neuvotteluhuoneessa asiakkaidemme kanssa, jotka ovat olleet opettavia ja motivoivia kokemuksia. Yleensä julkaisen nettisivun jo kehitysvaiheessa, jotta asiakkaamme näkee kehityksen ja nykyisen sivuston tilan. Tällä tavoin he voivat lähettää palautetta joko sähköpostin tai Skypen kautta. Nettisivun tiedostot laitan FileZilla -tiedonsiirto-ohjelmalla yrityksen palvelimelle.

Tärkein ja tällä hetkellä yrityksen tuottoisin tuote on RFID-järjestelmään perustuva kassajärjestelmä, joka toimi alkuinspiraationa yrityksen perustamiseen. Kyseisessä kassajärjestelmässä yritys antaa asiakkaille ja/tai työntekijöille kortin, jonka voi syöttää lukijaan. Kortiin voi laittaa henkilötietoja tai vaikkapa kertyvää laskua; mitä tahansa järjestelmän ostaja haluaakaan.

Ongelmana tällä hetkellä kyseisessä toimintamallissa on sen projektimainen toimintamalli, jossa yhteen projektiin joudutaan rakentamaan melkein kaikki osat alusta loppuun, sillä asiakkaat (ruokalat, ravintolat, yökerhot jne.) tahtovat kassajärjestelmältä erilaisia ominaisuuksia. Yhteen projektiin kuluu siis pieneltä yritykseltä paljon aikaa ja resursseja eikä se lopulta tuota paljoakaan tuottoa. Sen lisäksi projektimaisessa toimintamallissa yrityksen on vaikeaa suunnitella tulevaa varten, sillä projekteja ei tule tehtäväksi tasaiseen tahtiin. Tästä syystä yrityksen uutena fokuksena on jo muutaman vuoden ajan ollut kehittää tapa tuottaa voittoa mahdollisimman pienellä vaivalla sekä ennakoitavalla tavalla. Ratkaisuksi on muodostunut nettikauppa, jonka kautta yritys voi myydä RFID-laitteita sekä Applen kehittämiä iBeaconeita. Nettikaupan avulla tuottoa kertyy tasaiseen ja ennakoitavan tahtiin, joka tuo omanlaista varmuutta yrityksen toimintaan sekä tulevaisuuteen.

Suuremmissa projekteissa joihin olen osallistunut, olemme rakentaneet verkkopohjaisia toiminnanohjausjärjestelmiä asiakkaillemme. Liittyessäni yritykseen syksyllä 2016 oli yrityksen projektina esimerkiksi pienelle investointiyritykselle rakennettu nettisivu, johon kyseisen yrityksen työntekijät kirjautuvat sisään ja näkevät kurssien vaihtelut ja muutokset. Järjestelmä on jo ollut toiminnassa vuoden 2017 alusta lähtien, mutta on edelleen jatkuvan kehityksen alla. Keväällä 2017 teimme toisenlaisen, mutta samalle pohjalle rakennetun sivuston yritykselle, joka toimittaa lounasannoksia työpaikoille. Tässä projektissa sain osakseni enemmän vastuuta, jonka otin vastaan mielelläni.

Suuremmissa projekteissa olemme käyttäneet JavaScript -komentosarjakieleen pohjautuvaa AngularJS -ohjelmistokehystä, jonka pohjalta käytämme myös node.js -ohjelmaa, joka mahdollistaa JavaScript -koodin ajamisen myös serveripuolella. Node.js:n kanssa voimme myös hyödyntää ohjelman mukana tulevaa npm -paketinhallintajärjestelmää, jonka avulla voimme asentaa useita tarvitsemiamme paketteja tukemaan lopullista tuotettamme. Yksi näistä paketeista on Express.js, joka on yksi keskeisimmistä paketeista asentaa. Se järjestää ohjelmamme MVC-arkkitehtuuriksi palvelimen puolella. Näiden lisäksi olemme käyttäneet myös HTML- ja CSS-kieliä visuaalisen puolen rakentamiseen. Yleensä käytän Atom -ohjelmistokehitysympäristöä kehitystyöhön. Olen oppinut näiden kielten ja ohjelmien käytön yrityksessä täysin, sillä opinnoissani en ole päässyt niihin vielä koskemaan.

Myös ominaisuuksien testaaminen kuuluu pienessä yrityksessä kehittäjän vastuualueisiin. Testaus tapahtuu yleensä joko omatoimisesti tai toisen työntekijän puolelta. Testauksessa emme käytä yleistä käytäntöä tai ohjelmistoa, vaan vain kokeilemalla. Virheiden löytyessä käytän Microsoftin kehittämää Visual Studio -ohjelmistokehitysympäristöä, jonka avulla voin jäljittää virheen syntymäkohdan ja seurata mikä tapahtumasarja johti virheeseen. Ohjelmalla voin laittaa koodiin pisteitä mihin ohjelman ajo keskeytyy, jotta näen esimerkiksi määriteltyjen muuttujien arvot juuri siinä kohdassa.

Ohjelmien ylläpito tapahtuu WordPress -projekteissa käytännössä päivittämällä liitännäisiä ja itse WordPressiä. Tämä on yleisesti ottaen helppoa eikä vaadi kehittäjältä hirveästi aikaa. Saan sähköpostiini aina ilmoituksen uudesta versiosta, joten päivittämistä ei tarvitse muistaa. WordPressiin voi myös ladata liitännäisiä, jotka rekisteröivät käyttäjämäärät sekä mitä sivustolla on katsottu usein. Sivustolla voi myös hyödyntää Google Analytics -teknologiaa, jolla voimme selvittää tarkemmin mitä vierailijat ovat klikanneet ja mikä on ollut suosituin osio. Joskus käymme läpi näitä tilastoja ja teemme muutoksia sivuihin. Muissa projekteissa asiakkaamme soittavat meille mahdollisista virheistä ja pyrimme korjaamaan ne mahdollisimman nopeasti. Virheiden korjaaminen jälkikäteen on häiritsevää, sillä keskittyminen nykyiseen tehtävään häiriintyy ja joudumme muuttamaan huomiomme muualle. Jotkut projekteistamme ovat jatkuvasti kehitteillä, mutta niistä ollaan aiemmin asiakkaiden kanssa sovittu niin. He maksavat yritykselle jatkuvasta jatkokehityksestä ja uusista ominaisuuksista ja aikamääreet ovat hyvin joustavia. Näitä projekteja jatkamme siis silloin, kun yritys ei ole vielä saanut uusia projekteja tehtäväksi.

Aloitin yrityksessä syyskuussa 2016 ja olen ollut töissä vaihtelevin työajoin opiskelun vuoksi. Opiskelijana en pysty sitoutumaan työhön täysillä, joten välillä tunnen kehitykseni olevan hidasta. Roolini yrityksessä on kohtalaisen vähäinen enkä esimerkiksi osallistu lainkaan yrityksen RFID -teknologiaan perustuvaan kehitykseen. Tärkein tavoitteeni olisi siten siis siirtyä WordPress -tehtävistä suurempiin projekteihin. Olen saanut paljon hyvää palautetta kyvystäni oppia nopeasti uusia toimintatapoja ja motivaatiostani. Oppiakseni uutta koen tärkeäksi saada uusia haasteita, joissa joudun parantamaan ohjelmointitaitojani. Yrityksessä ollaan oltu positiivisia työpanokseni suhteen enkä koe saaneeni koskaan vakavaa kritiikkiä työstäni. Tiedostan puutteeni ja koen, että muut työntekijät ovat huomanneet minun olevan valmis oppimaan ja kuuntelemaan neuvoja.
Ohjelmoinnin lisäksi keskustelemme usein uusista mahdollisuuksista yritykselle. Yritys on kohtalaisen nuori, mutta nettikaupan ansiosta vakaalla pohjalla. Olemme jo pitkään pohtineet iBeacon -lähitilanpaikannuslaitteeseen perustuvaa applikaatiota tai palvelua, jota voisi myydä toimijoille, jotka haluavat tarjota asiakkailleen sisätilanpaikannusjärjestelmää.

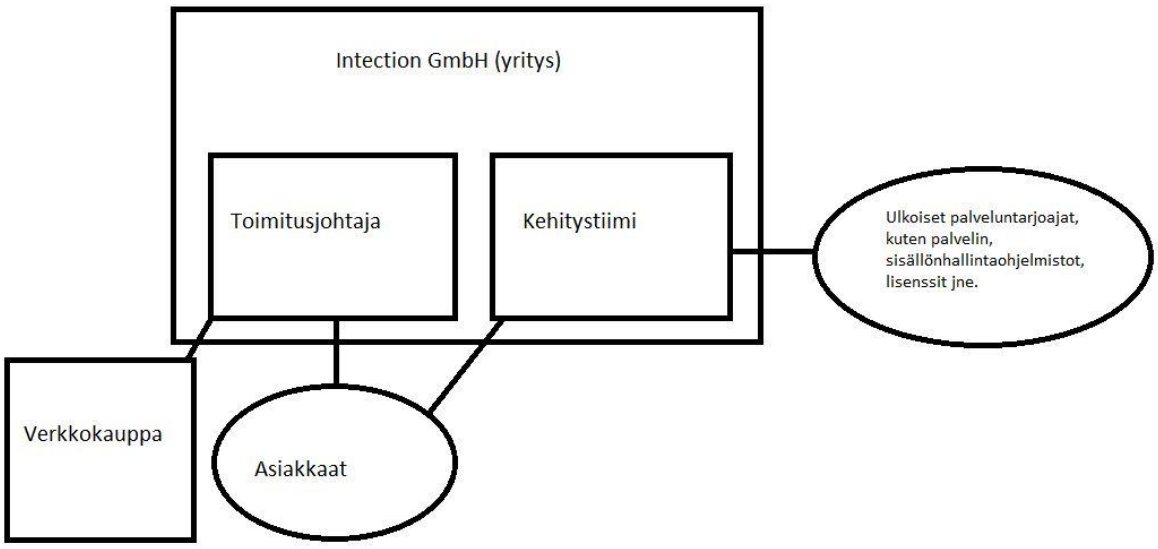
Mahdollisia käyttökohteita voisivat olla lentokentät, museot tai ostoskeskukset. Yksi mahdollisuus minulle olisi erikoistua kyseiseen teknologiaan pienessä yrityksessämme.

Tämä on ensimmäinen työpaikkani tällä alalla ja olen edelleen opiskelemassa, joten toiveenani on kokeilla monia eri alan suuntautumismahdollisuuksia. Hakiessani työharjoittelupaikkoja kohdensin hakuni pienempiin yrityksiin – enemmän työtehtäviä sekä enemmän vastuuta. Samalla myös hierarkia yrityksen sisällä on melko tasainen. Voin aina koputtaa yrityksen toimitusjohtajan ovelle keskustellakseni asioista sekä pyytää vanhempaa kehittäjää selittämään tarkemmin, mikäli jokin asia ei tunnu selvältä.

2.2 Sidosryhmät työpaikalla

Yrityksen kaikki työtehtävät tehdään samassa toimistossa aina asiakaspalvelusta kehitykseen. Toimitusjohtaja vastaa hallinnollisista asioista, markkinoinnista sekä asiakasneuvotteluista. Hän ottaa vastaan tiedustelut mahdollista IT-ratkaisuista, joita voimme tarjota asiakkaillemme sekä vastaa kaikista käytännön asioista, jotka eivät suoraan liity kehitystyöhön. Hän vastaa myös verkkokaupan toiminnasta ja tilauksien prosessoimisesta.

Kehitystiimimme vastaa ohjelmistotuotannosta sekä asiakaspalvelusta. Projektin laajuudesta riippuen laadimme suunnitelman käytetyistä kehitysmenetelmistä ja aikataulusta, jonka jälkeen tuotamme ohjelmiston. Kehityksen jälkeen vastaamme korjauksista ja mahdollisesta jatkokehityksestä. Yritys myy RFID -teknologiaan liittyviä tuotteita, kuten kortteja ja kortinlukijoita sekä iBeaconeita. Tämän lisäksi tarjoamme asiakkaille räätälöityjä IT-ratkaisuja. Asiakkaamme ovat yleensä kaltaisiamme pienyrityksiä.



Kuva 1. Yrityksen sidosryhmät nuoremman ohjelmistokehittäjän näkökulmasta

Yrityksen sisäisiin sidosryhmiin ei kuulu kovinkaan monta henkilöä tai ryhmää, kun kyseessä on näin pieni yritys. Siihen kuuluu siis toimitusjohtaja ja kehitystiimimme. Verkkokauppa toimii omalla yhtiönimellään eikä suoraan kuulu yritykseen ja on siis käytännössä sisaryhtiö.

2.3 Vuorovaikutustaidot työpaikalla

Kaikki yrityksen työntekijät ovat samassa tilassa, joka auttaa kommunikointia. Työpisteeni sijaitsee aivan vanhemman ohjelmoijan vieressä ja toimitusjohtajan huone aivan selkäni takana, joten apua löytyy aina sitä tarvitessa. Käytämme kommunikointiin ja tiedon jakoon sähköpostia ja Skypeä, sekä asiakkaiden ongelmia ratkaistaessa myös TeamViewer -etähallintaohjelmistoa. Yrityksessä kuitenkin pääsääntöisesti keskustellaan paljon suullisesti, eikä kommunikointia käydä tavallisesti koneiden kautta.

Yritys on saksalainen, joten asiakaspalvelu ja neuvottelut asioidaan saksaksi. Yhdessä puhumme englantia, jos asia koskee myös minua. Koodi kirjoitetaan yleensä englanniksi ja sivustoja tehtäessä pyydän toista työntekijää kääntämään sivuston vierailijoille näkyvän tekstin saksaksi. En voi yleisesti ottaen osallistua tekemään asiakaspalvelua, sillä saksan kielen taitoni ei ole vielä työtehtäviin sopivalla tasolla.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 38

Keskiviikko 20.9.2017

Olen työskennellyt yrityksessä jo noin vuoden ajan, joten työviikon aloitus oli pääpiirteiltään samanlainen kuin aina ennenkin. Päivän tavoitteenani oli jatkaa käynnissä olleen projektini viimeistelyä ja mahdollisesti avustaa vanhempia kehittäjiä heidän työtehtävissään. Aloitin työviikkoni keskustelemalla yrityksen toimitusjohtajan kanssa WordPress -projektin nykytilasta sekä seuraavista työvaiheista. Esittelin hänelle omia ideoitani mitä sivustolle voisi vielä lisätä tai muokata sekä keskustelimme ongelmasta, joka on ilmestynyt jo muutaman kerran. Ongelmana oli, ettei sivu välillä auennut, vaan jätti valkoisen kuvan sivulle. Tarkastelin nettiselaimeni kehittäjäkonsolista virheitä ja ongelmaksi selvisi virhekoodi 500, eli palvelimen virhe. Sivusto ei ladannut JavaScript-, CSS- tai HTML-tiedostoja eikä mitään muutakaan.

Ongelma ratkesi kuitenkin aina, kun kirjauduin WordPressin -ylläpitoikkunaan. Jostain syystä tällä tavalla sivu latautui myös muille tavalliseen tapaan. Aloin tutkimaan virheen alkamisen ajankohtaa ja selvisi, että se alkoi ilmestyä samoihin aikoihin, kun latasin liitännäisen nimeltä W3 Total Cache. Latasin kyseisen liitännäisen nopeuttaakseni sivuston latausaikaa, minkä se tekikin. W3 Total Cache on yksi suosituimmista WordPress -liitännäisistä, jota usein suositellaan lataamaan sivuston latausajan nopeuttamiseksi. Aiemmista projekteistani muistin tämän liitännäisen poiston olevan hieman työläämpi tehtävä, joten jätin sen myöhemmäksi.

Sivustosta puuttuivat edelleen kuvat ja saksakielinen teksti. Olin aiemmin laatinut ohjeet toimitusjohtajalleni tekstin muokkaamiseen ja WordPressin yleiseen käyttöön, jonka pohjalta hän on asiakkaan kanssa laatinut aineistoa sivustolle. Projekti etenee tässä vaiheessa omaa tahtiaan, sillä joudun odottamaan kuvia.
WordPress on käytännöllinen työkalu, sillä sen käyttöön ei tarvita aiempaa kokemusta webkehittämisestä tai ohjelmoinnista. Kuka tahansa voi aloittaa kehittämisen ja luoda toimivat ja kelvolliset sivut. Oppiakseni uutta haluaisin kuitenkin tehdä päästä tekemään kehitystyötä muilla työkaluilla. Tällä hetkellä pienellä yrityksellä ei kuitenkaan ole muita projekteja työn alla sekä vanhempi kehittäjä, jonka kanssa yleensä työskentelen, on lomalla.

Keskustelin esimieheni kanssa työpäivän päätteeksi sivuston nykytilasta ja päätimme pistää projektin tältä erää tauolle siihen asti, kunnes saamme kuvat. Hän kertoi uudesta si-

vustosta, jonka rakentaisin freelancerina toimivalle konsultille. Kyseinen sivusto olisi kohtalaisen pieni ja sisältäisi etusivun ja blogin, jota hän itse päivittäisi. Huomenna aloitan työni uuden sivuston parissa.

Torstai 21.9.2017

Tavoitteeni tälle päivälle on saada uusi projekti hyvälle alulle. Tehtäväni on keskustella pomon kanssa mitä asiakas haluaa sivulta ja luoda ajatuskartta halutuista ominaisuuksista. Sen jälkeen asennan WordPressin rakennusalustan. Projektin alussa tahdon rakentaa sivustoa lokaalisti turvallisuussyistä sekä työskentelyn nopeuttamiseksi. Tämä projekti ei ole yritykselle kovinkaan tärkeä, joten haluamme saada sen valmiiksi mahdollisimman nopeasti. Tästä syystä käytämme valmiiksi tehtyä pohjaa, joka nopeuttaa työtä huomattavasti.

Aloitan projektini yleensä aina lokaalisti myös siitä syystä, etten halua asiakkaan tarkastelevan keskeneräistä työtä. Käytännössä tämä tapahtuu siten, että puran WordPressin zip-tiedoston XAMPP -ohjelman htdocs -kansioon, esimerkiksi kohteeseen C:/Program Files/XAMPP/htdocs. Sen lisäksi teen MySQL -tietokantaan taulun, johon WordPress luo sarakkeet ja syöttää tarvittavat tiedot. Kuten mainitsin, käytän XAMPP -ohjelmaa lokaalissa kehittämistyössä. Sen avulla saan käyttööni Apache HTTP Serverin, joka toimii palvelimena sekä MySQL -tietokantaohjelmistoa, jota käytän tietokannan hallintaan.

Sen jälkeen asetan sivuston käyttämään tietokantaani, joka tapahtuu käytännössä muokkaamalla wp-config.php -tiedostoa. Tämä tiedosto ohjaa sivuston käyttämään säädettyä tietokantaa. Alla olevassa kuvassa näkyy kyseisen tiedoston perusasetukset.

```
20
21  // ** MySQL settings - You can get this info from your web host ** //
22  /** The name of the database for WordPress */
23  define('DB_NAME', 'database_name_here');
24
25  /** MySQL database username */
26  define('DB_USER', 'username_here');
27
28  /** MySQL database password */
29  define('DB_PASSWORD', 'password_here');
30
31  /** MySQL hostname */
32  define('DB_HOST', 'localhost');
33
34  /** Database Charset to use in creating database tables. */
35  define('DB_CHARSET', 'utf8');
36
37  /** The Database Collate type. Don't change this if in doubt. */
38  define('DB_COLLATE', '');
39
```

Kuva 2. Wp-config.php -tiedoston perusasetukset

Perusasennusten jälkeen saan haltuuni esimieheltäni valmiin pohjan, jonka päälle rakennan sivun. Valmiin teemapohjan zip -tiedoston voi lisätä WordPressin -ylläpitoikkunasta, jolloin se näkyy käytettävien teemojen listalla.

Projektin aloittamisen jälkeen keskustelen päivän lopuksi esimieheni kanssa tarkemmin yksityiskohdista ja tarpeista, joita asiakas toivoo sivuston sisältävän. Tämä pitää sisällään värit, sisällön ja kaikki sivun eri osiot. Näiden tietojen pohjalta on helpompaa jatkaa projektia ja myöhemmin tulemme tarkastamaan uudelleen sivuston tilaa.

Perjantai 22.9.2017

Aloitan päiväni valokuvaajan ohjeistamisella toivomistani kuvista entisen projektini sivustolle. Otamme muutamia kuvia erilaisista esineistä, jotka kuuluvat asiakkaan toimikuvan esineistöön. Asiakas on painotalo, joka tuottaa esimerkiksi tarroja, tikettejä ja julisteita. Tämän takia otimme kuvia heidän tuotteistaan sekä erilaisista painamiseen käytettävistä osista ja myöhemmin valokuvaaja menee asiakkaan luokse ottamaan lisäkuvia. Saan kuvat todennäköisesti jo ensi viikolla, jonka jälkeen voin jatkaa projektin loppuun saattamista.

Päätin ruveta poistamaan W3 Total Cache -liitännäistä, sillä minulla oli hieman aikaa sen tekemiseen. Tämä osoittautui kuitenkin hieman hankalammaksi tehtäväksi kuin olin odottanut, sillä liitännäinen luo useita tiedostoja ja kansioita jotka eivät kuitenkaan poistu, kun liitännäinen poistetaan. Tämän aiheuttaa hieman hankaluuksia, sillä en pystynyt poistamaan palvelimeltamme kyseisiä tiedostoja oikeuksieni ollessa riittämättömät muokkaamiseen ja poistamiseen. Vanhemmalla kehittäjällä on kyseiset oikeudet tiedostojen poistamiseen, joten joudun odottamaan hänen paluutaan työpaikalle. Tällä hetkellä vaikutti kuitenkin siltä, että sivusto toimi joka tapauksessa, joten päätin jättää tiedostojen poiston myöhemmäksi. Muuten sain liitännäisen poistetuksi ja muut sen jälkeen jättämät jäljet siivottua.

Tämä kuitenkin tarkoitti sitä, että sivuston latausaika lisääntyi. Ilman mitään välimuistiin sivuston materiaalia lataavaa liitännäistä sivuston latausaika on liian paljon. Aloin siis etsimään parasta vaihtoehtoa, joka olisi mielellään minimaalisempi ja tekisi vain sen mitä tahdon sen tekevän. W3 Total Cache oli liian iso liitännäinen, joka pyrki tekemään useita asioita samanaikaisesti. En lopulta käyttänyt suurinta osaa liitännäisen ominaisuuksista, sillä olin tietoinen niiden taipuvaisuudesta tuottaa virheitä ja ongelmia. Lopulta kävikin niin, että siitä oli enemmän haittaa kuin hyötyä.

Vaihtoehtoisista liitännäisistä löysin useita ilmaisia vaihtoehtoja, kuten WP Super Cache. Monessa lähteessä mainittiin kuitenkin WP Rocket (Agrawal 2017), joka on kuitenkin maksullinen. Yleisesti ottaen se vaikuttaa ihanteelliselta liitännäiseltä vaatimuksiini – pieni ja tekee hyvin sen muutaman asian, minkä haluan sen tekevän. Tällä tavalla minun ei tarvitse pelätä, että se sotkisi jotain sivustollani.

Loppupäivän vietin uuden projektin parissa. Lisäsin materiaalia sivulle ja muokkasin väriteemaa halutuksi. Lisäsin Contact Form 7 -liitännäisen, jonka avulla voin helposti luoda kontaktilomakkeen. Lisäsin asiakkaan jo kirjoittamia artikkeleita blogiin sekä muokkasin sivustolta kommentit pois. Ensi viikolla jatkaisin nykyisten projektien parissa vielä ja toivottavasti sitä seuraavalla viikolla voisin jo aloittaa muiden kehitystyökalujen parissa, kun yrityksen vanhempi kehittäjä palaa lomiltaan. WordPress on tehty tarkoituksella helposti lähestyttäväksi työkaluksi, mutta nuorempi kehittäjä ei opi sen avulla oikein mitään uutta.

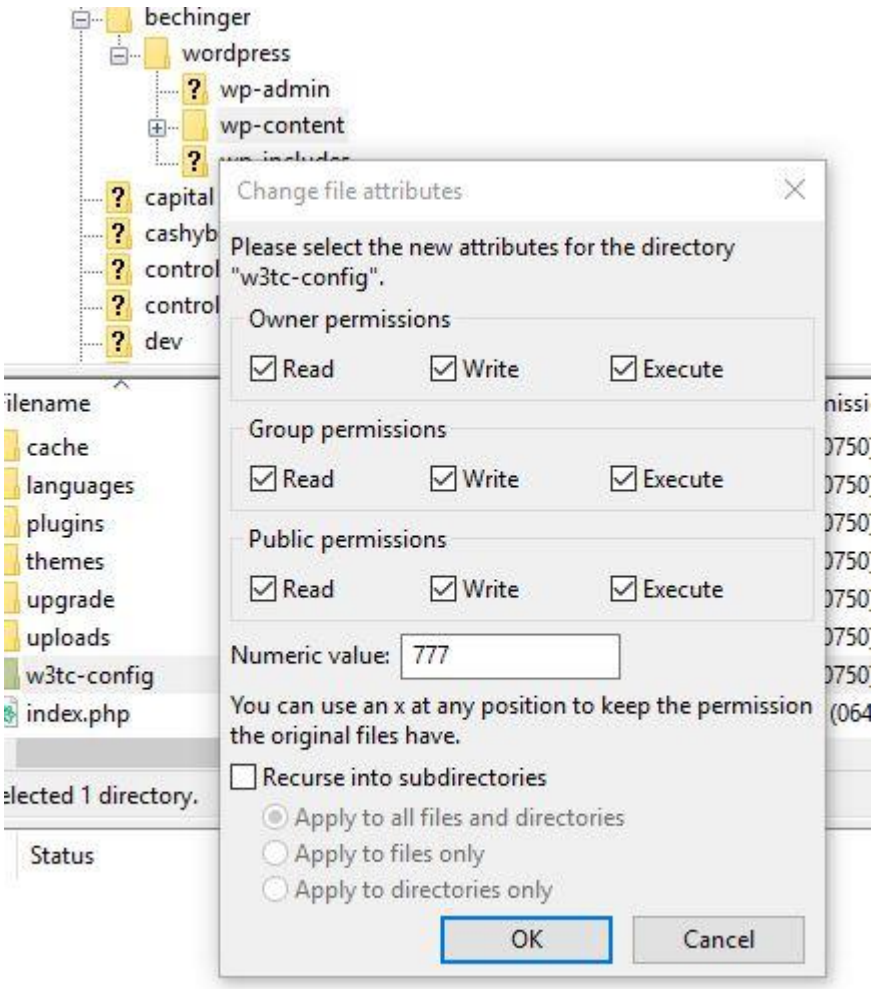
Viikkoanalyysi

Ensimmäisen raportointiviikkoni tavoite oli viimeistellä käynnissä ollut WordPress -projekti. Tavoitteena oli, että kuvia ja tekstiä lukuun ottamatta sivusto olisi toimivassa ja käyttöönotettavassa kunnossa. Keskustelimme paljon asiakkaan kanssa sivuston tavoitteista ja sisällöstä ja tämänhetkiseen versioon oltiin tyytyväisiä. Näimme sivuston olevan julkaisukelpoinen, kun olemme saaneet lisättyä kuvat ja tekstit. Omalta osaltani kyseinen projekti on täten tauolla kuvien saapumiseen asti.

Sivuston suurin ongelma oli W3 Total Cache -liitännäinen, joka aiheutti sivuston ”500 Internal Server” -virheilmoituksen eikä sivustoa siten voinut ladata. Asensin kyseisen liitännäisen, sillä halusin sivuston latautuvan nopeammin ja sitä kautta parantavan sivuston hakukoneoptimointia. Nopealla latausajalla vähennetään myös sitä mahdollisuutta, että sivuston vierailija turhautuu latausaikaan eikä tutustu sivustoon. W3 Total Cache on yksi suosituimmista WordPress -liitännäisistä, jota on ladattu yli miljoona kertaa. Sitä ei kuitenkaan päivitetä kovinkaan usein (kirjoitushetkellä (23.9.2017) liitännäistä oli viimeksi päivitetty 5 kuukautta sitten), mikä johtaa virheisiin usein, kun WordPressiin tulee uusi päivitys.

Pitkän harkinnan jälkeen päätin poistaa liitännäisen käytöstä, mikä osoittautui kohtalaisen vaikeaksi tehtäväksi. Liitännäisen poisto Wordpressin ylläpitäjän käyttöliittymästä ei riittänyt, vaan jouduin tarkistamaan, että liitännäisen luomat muokkaukset .htaccess -tiedostoon olivat todella poistuneet. (Agrawal 2016) Kyseinen tiedosto on yksi tärkeimmistä tiedostoista WordPress -projekteissa. Sen pääasiallinen tehtävä on luoda hakukoneille ystävällisimpiä linkkejä sekä uudelleenohjata vanhat linkit uusiin. Tämän lisäksi .htaccess

-tiedosto määrittelee suurimman ladattavan tiedoston koon WordPress -sivustolle ja PHP-tiedostoihin käytetyn muistin määrän. Kyseisellä tiedostolla voidaan myös suojata sivustoa antamalla lupa vain tietyille IP-osoitteille vierailla sivulla tai päästä sivuston ylläpitonäkymään. (Williams, Damstra & Stern, 38)
Liitännäinen jätti myös muutamia tiedostoja palvelimelle, joita en saanut poistettua. Yritin muuttaa oikeuteni FileZilla -ohjelmassa arvoon 777, jolla voisin muokata ja poistaa kyseisiä tiedostoja.



Kuva 3. FileZilla -ohjelmassa tiedoston muokkausoikeuksien muuttaminen

Ohjelma antaa kuitenkin vastauskoodin 550, joka tarkoittaa tarvittavien oikeuksien puuttumisen käyttäjällä. Ainut tapa poistaa tiedostot on kirjautua sisään FTP-ohjelmaan korkeammilla käyttöoikeuksilla varustetulla käyttäjätilillä, jonka omaa yrityksen vanhempi kehittäjä, joka ei ollut tällä viikolla paikan päällä. Siitä huolimatta sivusto toimii, joten voin jättää aiheen myöhemmäksi.

Uuden projektin aloittaminen oli minulle jo entuudestaan tuttua. Projektin alkuvaiheessa haluan rakentaa sivua lokaalisti ja vasta myöhemmin siirtää kehitystyön palvelimelle. Siinä

vaiheessa sivusto on tarpeeksi edustavassa kunnossa, että asiakas voi tarkkailla kehitystä. Käytän lokaalisti rakentamiseen XAMPP -ohjelmaa, jossa saan käyttööni samalla Apache HTTP Server -palvelinohjelman sekä MySQL -tietokantaohjelmiston. XAMPP on hyvä työkalu kehitykseen lokaalissa työympäristössä, jonka takia olen päätynyt käyttämään sitä.
Yleisesti ottaen voidaan katsoa kehitystyön tekemisen lokaalissa työympäristössä olevan paras ratkaisu. Tämä johtuu kahdesta syystä. Ensinnäkin kehittäjä voi kokeilla ja testata asioita ja ominaisuuksia rikkomatta jo julkaistua sivustoa. Kehittäjien ei tulisi tehdä kokeiluja jo käytössä olevilla sivuilla. Toinen hyöty on yksityisyys. Kehitystyötä voi tehdä rauhassa omalla työasemalla ja hallitsemalla täysin kaikkea kehitykseen liittyen. Muutokset eivät näy kaikille sivuston kävijöille ja vaikka pääsyä nettisivulle voi rajoittaa, voi potentiaalinen yleisö olla maailmanlaajuinen. (Williams, Damstra & Stern, 44.)

Ensimmäinen raportointiviikko kului suurilta osin jo osattujen asioiden äärellä. WordPress -sisällönhallintaohjelma alkaa olla minulle jo erittäin tuttu työväline eikä sivuston rakentaminen tuota oikeastaan mitään suurempia haasteita. Mario Peshev kirjoittaa kolumnissaan (2017), kuinka WordPressillä kehittäminen ei ole oikeaa kehitystyötä. Jaan osittain mielipiteen hänen kanssaan, sillä kyseisellä ohjelmalla sivuston rakentaminen on tehty niin helpoksi, ettei käyttäjän tarvitse osata käytännössä mitään ohjelmointikieltä toimivan sivuston rakentamiseen. Tämä on tietysti tulosten saamiseksi ehkä hyvä asia, mutta oppimisen kannalta mielestäni huono.
Tällä hetkellä pienessä yrityksessä vallitsee jonkinlainen suvantovaihe, jossa käynnissä ei ole kuin muutamia pieniä projekteja ja kehitämme edelleen erästä jo valmistunutta projektia, jota jatkamme aina ajan salliessa ja asiakkaan pyytäessä lisäominaisuuksia. Tästä syystä en ole päässyt kokeilemaan taitojani haastavimmissa työtehtävissä ja toivon, että jatkossa pääsisin jälleen työskentelemään AngularJS:n tai jonkin uuden parissa. Kenties voisin jatkossa pyrkiä kehittämään jotain iBeacon -lähitilapaikannuslaitteen parissa joita yritys myy, muttei ole vielä kehittänyt mitään ohjelmistoa.

3.2 Seurantaviikko 39

Maanantai 25.9.2017

Aloitan työviikkoni keskustelemalla johtajan kanssa painotalolle rakentamani sivuston nykytilasta. Sivusto on siis melkein valmis ja siitä puuttuu enää vain asianmukaiset kuvat sekä saksankielinen teksti. Valokuvaaja on tämän viikon ottamassa kuvia sivustolle asiakkaan luona ja saamme kuvat sieltä todennäköisesti myöhemmin seuraavalla viikolla. Viikonlopun aikana olimme saaneet muutamia kuvia tuotteista, joita valokuvattiin viime viikol-

la. Tekstin muokkaavat yhdessä asiakas ja esimieheni sekä tavanomaiset tekstit esimieheni täyttää jo tänään. Minun tehtäväni on korjata mahdollisesti esiintyvät virheet, lisätä uusia kuvia asiakkaan tuotteista sekä lisätä sivustolle uusi alasivu, johon tulee tietoa asiakkaan saamasta DIN EN ISO 9001:2008-sertifikaatista.

Olin aikaisemmin jo lisännyt sivustolle galleria-alueen, jossa tuotekuvat olisivat. Kuvien vaihtaminen esimerkkikuvista oikeisiin oli yksinkertainen tehtävä, johon ei kulunut kauaa aikaa. Samoin asetin muutaman kuvista toimimaan linkkinä asiakkaan aikaisempiin projekteihin. Sivustolla on oma osionsa asiakkaan referenssiprojekteille, joka on aseteltu Essential Grid -liitännäisen avulla kuvaruudukoksi, jossa jokainen kuva on linkki eri referenssiprojektin alasivulle.

Kuvien lisäämisen jälkeen rakensin uuden alasivun edellä mainitsemalleni sertifikaatille, jonka asiakas oli toivonut lisättävän sivustolle. Alasivun tulisi sisältää tietoa ja lähteitä sertifioijasta sekä linkin ladattavaan pdf-tiedostoon asiakkaan sertifikaatista. Sain asiakkaalta kuvia sekä kyseisen pdf-tiedoston, joita sain hyödyntää alasivua tehdessä. Sivuston rakentaminen osoittautui nopeaksi ja yksinkertaiseksi tehtäväksi, sillä rakenne tulisi olemaan samanlainen kuin muilla alasivuilla. Asiakas oli myös kertonut aiemmin, mitä asioita hän toivoi sivustolla mainittavan ja korostettavan. Toiveena oli, että korostetaan sertifikaatin vaatimuksia, jotka asiakkaamme oli hyväksytysti läpäissyt. Sivun rakentaminen on nopeaa, kun suunniteltu rakenne ja aineisto on selkeä sekä materiaali on helposti saatavilla.

Päivän viimeisenä tehtävänä päätin tutkia, miten saisin kontaktilomakkeen tekstin käännettyä saksan kielelle. WordPressin käyttöliittymä ei sallinut haluttujen kohtien kielen vaihtamista, joten minun oli pakko löytää jokin toinen keino. Kontakilomake oli teeman luojan tekemä ratkaisu, joka oli ollut siihen asti toimiva ratkaisu, mutta muokkausmahdollisuuksien puutteen vuoksi päädyin asentamaan jälleen Contact Form 7 -liitännäisen. Sen avulla korvasin vanhan kontaktilomakkeen ja sain enemmän vaihtoehtoja muokkaamiseen.

Tiistai 26.9.2017

Päivän aluksi esimieheni ilmoitti olevansa työpäivän muualla, joten päätin käyttää työpäivän muutamien pienten virheiden korjaamiseen uudessa projektissa. Eilen sain tehtyä kaikki tehtäväni, joten tälle päivälle ei asettunut mitään varsinaisia tavoitteita.

Kontaktilomakkeen lähetysnappi oli hieman liian alhaalla viivan päällä. Samoin uutiskirjeen ilmoittautumislomakkeen lähetysnappi oli hieman sivussa. Kumpikin virhe johtui span -elementistä, jolle oli asetettu CSS-tiedostossa säännöt *”visibility: hidden;”* sekä *”display:*

inline-block;”. Tämä tarkoitti sitä, että vaikkei elementtiä nähnyt, se vei silti fyysisesti oman tilansa sivustolla. Jos sääntönä olisi ollut *”display: none;”*, olisi elementti toiminut oikein. Kyseessä oli span -elementti, jolle oli annettu luokka ”ajax-loader”. Se oli siis elementti, joka näytti pienen latausikonin, kun käyttäjä painaa nappia. Päätin sen olevan kokonaisuuden kannalta melko turha ja piilotin sen.

Päätin poistaa uuden sivuston blogista muutamia pieniä ominaisuuksia sekä muuttaa blogia saksankieliseksi. Blogiin tulee oletuksena useita ominaisuuksia, joita ei lopulta yleensä tarvitse. Tässä projektissa sellaisia ominaisuuksia olivat esimerkiksi monet sosiaalisessa mediassa jakamiseen käytetyt linkit, kommentit sekä tekijän nimi. Olen käyttänyt useissa aiemmissa WordPress -projekteissani Disable Comments -liitännäistä, joka tehokkaasti ja luotettavasti poistaa kommentit blogiosioista. Muut osat sain poistettua joko käyttöliittymän valinnoista tai kirjoittamalla aiemmin mainitsemani CSS -säännön.

Loppupäivän vietin tekemällä sivustolle oman sijaislogon sekä muutamia muita ehdotuksia logosta, joita esimieheni oli pyytänyt tekemään. En lopulta päivän aikana tehnyt merkittäviä muutoksia, sillä olen omalta osaltani saanut tehtyä tehtäväni. Tällä hetkellä odotan vain uuden projektin siirtämistä palvelimelle sekä kuvien ja tekstin saamista vanhan projektin loppuunsaattamiseksi.

Keskiviikko 27.9.2017

Keskiviikko alkoi uudella tehtävällä siirtää uusi sivusto palvelimeen, jotta esimieheni voi tarkkailla työn edistymistä sekä saada uusia kehitysideoita. Sain tunnukset palvelimelle tiedostojen siirtämistä sekä tietokannan luontia varten. Päivän tavoitteena oli saada sivusto siirrettyä palvelimelle ja korjata tiedostojen polut toimimaan palvelimella.

Ensimmäinen haasteeni oli käyttää yrityksen hankkimaa saksankielistä palvelinta. Alkeellisella saksankielentaidollani kykenin kuitenkin navigoimaan palvelimen nettisivuilla tarpeeksi hyvin. Onnistuin luomaan palvelimen tarjoamalla phpMyAdmin -tietokantaohjelmalla taulun, johon siirsin lokaalista projektista importoimani tietokannan. Tietokannan siirtäminen osoittautui melko helpoksi ja nopeaksi tehtäväksi. Vaikeinta tehtävässä olikin lopulta löytää palvelimen palveluntarjoajan sivustolta oikea paikka.

Lounaalla keskustelin esimieheni kanssa sivustosta ja parhaasta käytännöstä muokata sen sisältöä saksankieliseksi. Aikaisemmissa projekteissani aineiston käännöstyö on osoittautunut yllättävän työlääksi prosessiksi. Tein sivuston englanniksi, jonka jälkeen toinen yrityksen työntekijä tekee käännöstyön. Ongelmaksi muodostui kyseisessä toimin-

tatavassa käännöstyön jälkeen tapahtuneet muokkaukset ja lisäykset sekä mahdolliset ”pop-up” -viestit, jotka olivat jääneet epähuomiossa kääntämättä.
Ehdotin, että opetan hänelle huomenna WordPress -sivuston muokkauksen alkeet sekä kirjoitan tämän työpäivän lopuksi hänelle perusohjeet sisäänkirjautumiseen sekä sivustojen ja eri osioiden muokkaamiseen. Hänellä ei ole aikaisempaa kokemusta ohjelmoinnista tai webkehityksestä, mutta WordPressin tärkein ominaisuus on sen lähestyttävyys. Ehdotus oli hänen mielestään hyvä ja sovimme oppitunnista huomiseksi.

Loppupäivästä siirsin sivuston tiedostot palvelimelle FileZilla --tiedonsiirto-ohjelmalla, joka myös oli yksinkertainen prosessi. Siirtäessä ainoa haaste itselleni oli olla tarkkana siirrettävien tiedostojen ja kohteen kanssa. Siirron jälkeen sivusto näytti odotetusti virhettä, joka johtui vääristä poluista tiedostoissa. Tietokannassa minun täytyi muuttaa ”wp_options” -taulun ”optionamme” -sarakkeen riviä, jossa arvo oli ”siteurl”. Kyseisellä rivillä oli ”options_value” -sarakkeessa arvona lokaalisen sivuston osoite. Tämä arvo tuli vaihtaa selaimen www-alkuiseen osoitteeseen.

Tämän jälkeen muokkasin wp_options -tiedostoa. Vaihdoin myös siellä vanhan lokaalin osoitteen uuteen ja määritin käytettävän tietokannan osoitteen ja salasanan, jota WordPress tulee käyttämään. Viimeisenä vaiheena muutin kuvien ja linkkien polut vastaamaan uutta palvelimessa toimivaa sivustoa. Tein myös sen palvelimen tietokannassa phpMyAdminissa komennolla

”UPDATE wp_posts SET post_content = REPLACE(post_content, 'localhost/lokaalisivu/', 'www.nettisivu.de/');”.

Päivän lopuksi tein esimiehelleni tekstitiedostona ohjeet WordPress -sivuston adminnäkymään sisäänkirjautumiseen, sivujen muokkaamiseen ja muutoksien tallentamiseen. Huomenna aamupäivästä käymme läpi WordPressin perusteet ja toivottavasti sen jälkeen saamme nopeutettua ja selkeytettyä käännöstyötä.

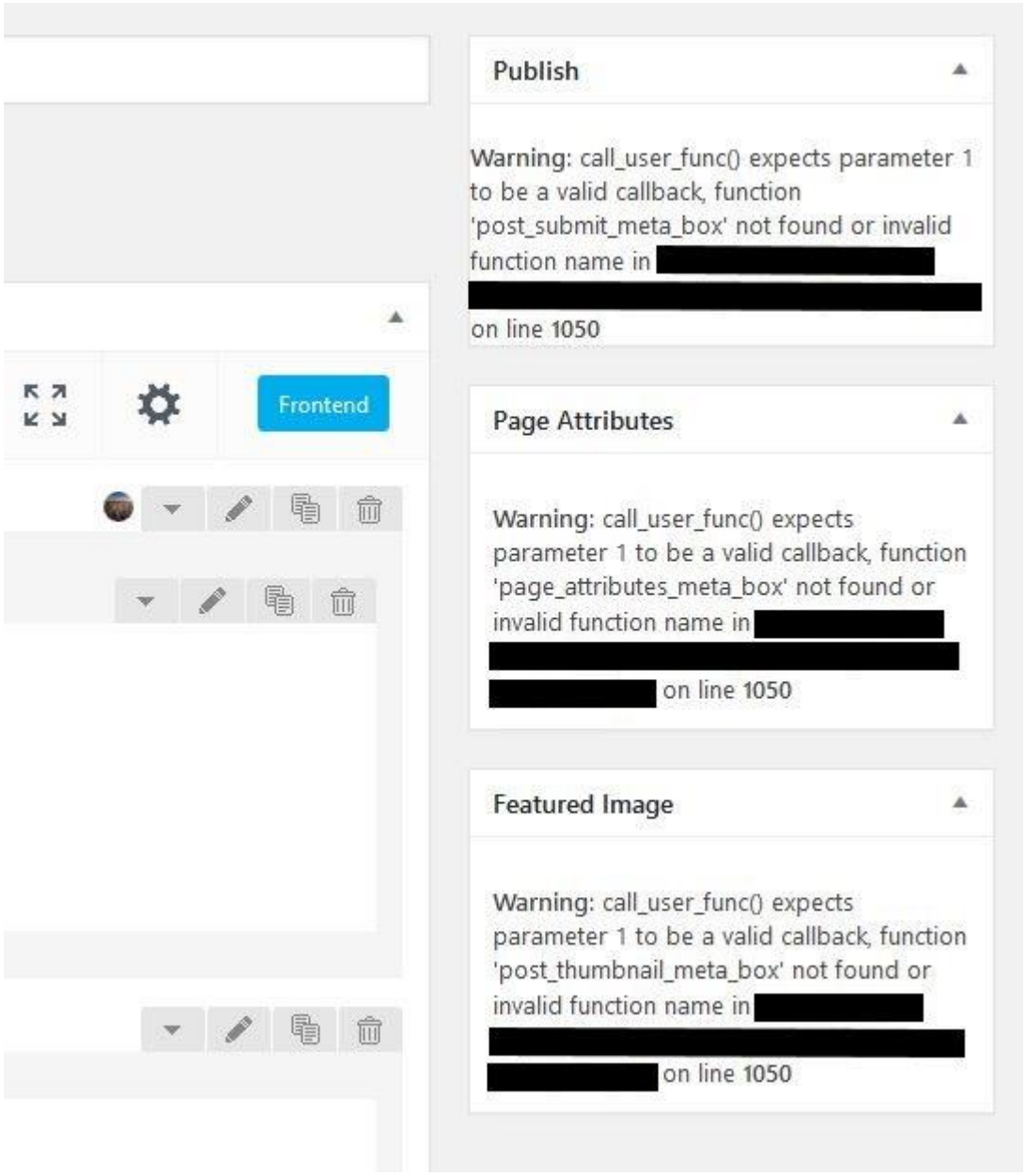
Torstai 28.9.2017

Tehtäväni työviikkoni viimeiselle päivälle oli opettaa esimiehelleni WordPress -sisällönhallintajärjestelmän perusteet. Tavoitteenamme oli, että hän kykenisi itsenäisesti muokkaamaan sivuston sisältöä ja kääntämään tekstit saksan kielelle. Eilen tein valmiiksi käyttöohjeet sivustomme muokkaamiseen WordPressillä, joihin hän oli tutustunut viime illan aikana. Ohjeissa kerroin yksityiskohtaisemmin jokaisen vaiheen aina sisäänkirjautumisesta tekstin muokkaamiseen.

WordPress -sivustoa muokataan sivustoon liitetyssä alasivussa ”/wp-admin/”, johon käyttäjän tulee kirjautua sisään. Sisäänkirjautumisen jälkeen käyttäjälle avautuu selaimen kautta toimiva käyttöliittymä. Näytin esimiehelleni, kuinka ylläpitoikkunaan päästään sekä näytin yleisesti mitä mahdollisuuksia WordPress tarjoaa kehittäjälle. Tämän jälkeen opastin häntä tekemään saman hänen omalla työkoneellaan. Hänellä ei ole aiempaa kokemusta kehitystyöstä, joten WordPressin käyttö viehätti häntä suuresti. Sen avulla hän kykeni luomaan aineistoa itsenäisesti, johon hän ei ollut aiemmin pystynyt.

Opetushetken jälkeen aloin tutkimaan konsulttisivun kuntoa palvelimeen siirtymisen jälkeen. Aloitin tekemällä pieniä muutoksia ja muutoksia tallentaessani huomasin virheen, joka esti tallennuksen. Tallennuspainike oli poissa ja sen paikalla näkyi vain alla oleva virheilmoitus. Ongelma oli korjattava mahdollisimman pian, sillä sivuston muokkaaminen olisi muuten mahdotonta. Aloin täten tutkimaan ongelmaa ja nopeasti selvisi, että virhe syntyi palvelimelle siirtymisen jälkeen.

Ensimmäiseksi kokeilin poistaa kaikki liitännäiset käytöstä ja selvittää, jos jokin niistä aiheutti konfliktin. Virhe esiintyi kuitenkin edelleen, joten varasin hieman enemmän aikaa ongelman selvittämiseksi. Virheilmoituksessa kerrottiin virheen esiintyvän eräässä wp-admin -kansiossa olevassa php -tiedostossa. Tarkastelin ilmoituksessa esiintyvää riviä ja kaikki vaikutti olevan mielestäni kunnossa. Vertasin sitä myös vanhempaan versioon, jota käytin kehittäessäni sivustoa lokaalisti. Lopulta huomasin tiedostossa olevan viittaus toiseen php -tiedostoon. Avatessani tiedoston selvisi, että se oli täysin tyhjä. Päätin korvata palvelimella olevan tiedoston lokaalilla versiolla, joka sisälsi koodia. Tämä korjasi virheen, joka oli siis ilmeisesti aiheutunut tiedostojen siirtämisestä palvelimelle. Jostain syystä kyseinen tiedosto oli tyhjentynyt migraation aikana.



Kuva 4. WordPress -käyttöliittymän virheviesti palvelimeen siirtymisen jälkeen

Päivän lopussa huomasin etusivulla olevan virheen. Se sijaitsi ensimmäisessä näkymässä, jonka sivuston vierailija näkee. Näkymässä oli yrityksen nimi, yrityksen motto, linkki kontaktilomakkeeseen sekä ”polyline” -elementti, joka piirtää viivan sivulle. Kyseinen elementti ei myöskään piirtynyt enää sivulle palvelimelle siirtymisen jälkeen. Yritin korjata elementtiä muokkaamalla WordPressin käyttöliittymän valikossa, mutta kaiken mukaan viivan pitäisi piirtyä entiseen tapaan myös tässä versiossa. Päätin tämän jälkeen tarkastella Firefox -selaimen kehittäjänäkymässä sivuston elementtien CSS -tyyliasetuksia tarkemmin. Huomasin, että ”polyline” -viivalle oli jostain syystä asettunut sääntö elementin piilottamiselle. Tämä tarkoittaa sitä, ettei kyseistä elementtiä näytetä lainkaan. Asetin uuden CSS -säännön elementille ja asetin sen tärkeämmäksi asettamalla ”!important” -

asetuksen säännölle. Lisäämällä ”!important” -komennon CSS -säännön perään voin ohittaa aiemmin asetetut säännöt.

Viikkoanalyysi

Viikon pääteemoina oli konsulttisivuston siirto palvelimelle, Wordpressin perusteiden opettaminen esimiehelleni sekä uusien kuvien syöttäminen painotalon sivustolle. Työskentelin koko viikon Wordpressin parissa, josta suuri osa työajasta kului myös keskusteluihin asiakkaan ja esimieheni kanssa. Keskustelimme paljon sisällönhallintaohjelmistojen hyödyistä ja haitoista sekä käyttömahdollisuuksista.

Esimiehelläni ei ole aiempaa kokemusta ohjelmointityöstä kehitystasolla, joten kävimme osittain hieman kiivaankin keskustelun yrityksen sisällä siitä, voisiko WordPressiä käyttää laajemmin muissakin projekteissa. Esimieheni oli aina ollut hyvin kiinnostunut oppimaan lisää WordPressistä ja halusi laajentaa sen käyttöä, kun taas vanhemmat kehittäjät sekä minä olimme olleet yleisesti ottaen sitä vastaan. Yritys rakentaa esimerkiksi verkkokauppoja ja verkossa toimivia ERP -järjestelmiä, joita on hankalaa rakentaa WordPressin avulla. White (2014) kirjoittaa, kuinka WordPress luotiin alun perin bloggaamiseen ja vasta myöhemmin siitä muodostui työkalu nettisivujen rakentamiseen. Useat kehittäjät alkoivat rakentamaan liitännäisiä, teemoja ja muita työkaluja, jolla sivustosta saatiin rakennettua myös muihin tarkoituksiin tarkoitettuja sivustoja.

Aiemmin sivuston siirtäminen palvelimelle ja nettiin lokaalista on ollut vanhemman kehittäjän tehtävä. Useimmissa projekteissa olemme työskennelleet yhdessä, joten hänellä on ollut suurin vastuu projektin etenemisestä. Tämän takia hän on myös halunnut tehdä palvelimelle siirtymisen itse. Tässä projektissa pääsin tekemään tämän vaiheen kuitenkin itse, joka osoittautui mielenkiintoiseksi haasteeksi. Ensimmäinen haasteista oli tietysti saksan kieliseen palvelimeen siirto, joka liittyy työnkuvaan Saksassa työskennellessä. Käytin myPhpAdmin -tietokannan hallintatyökalua sivustossa käytettävän tietokannan siirtämiseen palvelimelle. Työkalun avulla voin luoda .sql -loppuisen tiedoston, jonka voin siirtää uuteen tietokantaan. Sen jälkeen kirjauduin sisään palvelimen tiedostokantaan FileZilla -ohjelmalla ja siirsin tiedostot uuteen kansioon. Siirron jälkeen huomasin muutamia virheitä, jotka sain onneksi korjattua nopeasti ja helposti.

Williams, Damstra ja Stern (sivu 44) kertovat kirjassaan, että yksi hyvä käytäntö kehitystyöhön on kolmiasteinen työprosessi. Ensimmäisessä vaiheessa rakennetaan sivustoa paikallisesti omalla koneella. Toisen vaiheen tarkoitus on testata sivuston toimivuutta testipalvelimella turvallisessa ympäristössä sekä varmistaa kaikkien kehittäjien operoinnin

samalla sivuston versiolla. Yrityksessä teen yleensä itsenäisesti WordPress -sivustoja, joten tämä vaihe jää omalta kohdaltani melko lyhyeksi. Kolmas ja viimeinen vaihe on sivuston siirtäminen palvelimelle ja siten myös näkyväksi Internetissä. Kolmivaiheisen kehitystyön yksi tavoite on sovittaa yhteen kehittäjän ja palvelimen käyttämät mahdollisesti erilaiset käyttöjärjestelmät ja varmistaa, että sivusto toimii myös palvelimelle siirtyessä.

Taulukko 1. WordPress -sivuston kolmen askeleen kehityskaari lokaalissa kehitystyössä

1	”Development” Kehitys lokaalisti	Kehitystyö tapahtuu omalla koneella ja sivusto näkyy paikallisella palvelimella, esimerkiksi Apache HTTP Server.
2	”Staging” Alustus ja testaus	Sivusto siirretään testipalvelimelle, joka vastaa ominaisuuksiltaan kehityspalvelinta esimerkiksi käyttöjärjestelmän osalta. Tällä tavalla saadaan turvallisesti varmistettua, että ohjelma tai sivusto toimii oikealla palvelimella aiheuttamatta mitään vahinkoa.
3	”Production” Kehitys palvelimessa	Sivusto siirretään lopuksi palvelimelle, jonka kautta se näkyy Internetissä. Tässä vaiheessa ei tulisi ilmentyä virheitä, ainakaan suuria, sillä ne oltaisiin huomattu aiemmissa vaiheissa.

En ole työssäni tehnyt kirjassa tehtyä toista vaihetta, jonka mukaan siis sivustoa tulisi ensin kokeilla testipalvelimella ennen sen siirtämistä oikeaan, netissä näkyvään palvelimeen. Olen pohtinut, miten sen voisin tehdä, sillä yrityksellä ei ole testipalvelimia jossa voisin ajaa sivustoa erilaisilla asetuksilla. Kirjassa (sivu 56) annetaankin vaihtoehdoksi ”virtual machines” -ratkaisu, jossa sivustoa voi yrittää ajaa erilaisilla käyttöjärjestelmillä, palvelimilla ja työasemilla sekä eri asetusten kanssa. He ehdottavat kehitysalustaa nimeltään ” Varying Vagrant Vagrants ”, joka on luotu nimenomaan WordPress -kehitykseen. Kehitysalustan nimi tulee ”Vagrant” -ohjelmasta, jonka avulla se luo virtuaaliympäristön VirtualBoxin avulla (Varying Vagrant Vagrants). Sen avulla kehittäjä voi luoda virtuaalisia kehitysympäristöjä sekä profiloida koodia ja korjata virheitä (Williams, Damstra & Stern, 57).

Toivon, että saan ensi viikolla muita tehtäviä. Vanhempi kehittäjä, jonka kanssa olen usein työskennellyt samoissa projekteissa palaa lomalta. Esimieheni kertoi, että hänellä on jo ajatus projektista, jonka hän voisi delegoida minulle. En vielä tiedä mistä projektissa on kyse, mutta tiedän että pääsen tekemään sitä AngularJS:n kanssa.

3.3 Seurantaviikko 40

Torstai 5.10.2017

Työviikkoni alkaa vasta torstaina Saksan yhtenäisyyspäivän ja koulun takia, joten minun täytyy taas orientoitua työhön. Viikko alkaa siitä huolimatta hyvällä uutisella, sillä yritys on saanut uuden projektin asiakkaalta. Aiemmin viikolla esimieheni ja vanhemmat kehittäjät olivat keskustelleet asiakkaan kanssa tarkemmin yksityiskohdista. Uudessa projektissa tulen rakentamaan vanhan tuotteemme päälle uuden osion, jota asiakas oli toivonut. Kyseessä on yrityksen rakentama ERP -järjestelmä kanttiineille, ruokaloille ja ravintoloille, jota yritys on myynyt jo useammalle yritykselle. Koska kyseessä on vanha tuote, voimme rakentaa sen päälle ja kehittää sitä edelleen asiakkaan toiveiden mukaiseksi. Pääsen pitämään taukoa Wordpress -sivustojen tekemisestä ja uudessa tehtävässäni tulen työskentelemään jälleen AngularJS:n parissa.

Yrityksen rakentaman ERP -järjestelmän avulla asiakas voi hallinnoida esimerkiksi työntekijöitä, toimituksia ja ruoka-annoksia. Vanhojen ominaisuuksien lisäksi he olivat esittäneet toiveen uudesta ominaisuudesta, jonka kehittäminen tulee olemaan minun vastuullani. He halusivat saada ruokalaan kosketusnäytöllisen listan viikon päivittäisistä ruoka-annoksista. Heidän asiakkaansa voisivat katsoa nykyisen ja seuraavan viikon tarjonnan ruokalassa. Menussa tulisi olla myös ”admin” -näkymä, jonka kautta sisäänkirjautuneet työntekijät voivat muokata listaa useammankin viikon päähän.

Järjestelmä on rakennettu AngularJS:n, node.js:n, HTML:n ja CSS:n avulla ja se näkyy netissä. Asiakas oli toivonut linkkiä heidän etusivultaan nyt rakennettavaan ruokalistaan, jotta ruokalan asiakkaat voivat tarkastella viikon tarjontaa myös netissä. Tämä tarkoittaa siis sitä, että joudun pohtimaan, rakennanko eri sivun ruokalistan muokkaamiseen, joka on piilotettu sisäänkirjautumisen taakse vai rakennanko lisäys, muokkaus ja poisto -valikot samaan sivuun. Päätän keskustella asiasta tarkemmin vanhemman kehittäjän kanssa, joka hetken tutkimisen jälkeen kertoo kannattavansa yhden sivun mallia, jossa ohjelma rakentaa tietyt elementit sivulle vain, jos sivuston vierailija on kirjautunut sisään.

Päivä kuluu lopulta uuden projektin suunnitteluun ja siitä keskusteluun. Päivän lopussa ehdimme asetella projektin alkuun ja saan alustan, jonka päälle tulemme rakentamaan sivustoa. Versionhallintaan tulemme käyttämään git -versionhallintaohjelmistoa. Useimmilta ominaisuuksiltaan se tulee olemaan samankaltainen kuin yrityksen aiemmissa kanttiineille rakennetuissa ERP -järjestelmissä, joten projektin pitäisi valmistua melko nopeasti.

Perjantai 6.10.2017

Päivän tehtäväni oli aloittaa ruokalan menu -listan rakentamien. Olimme vanhemman kehittäjän kanssa eilen keskustelleet lopullisessa tuotteessa olevista yksityiskohdista ja tänään minun tuli päättää miten rakennan visuaalisen puolen yleisen rungon. Aloitin ensin piirtämällä paperille paperilankaversion, jossa näkyy viikonpäivät, napit edelliseen ja seuraavaan viikon tarjontaan siirtymiseen sekä ylläpitäjälle näkyvä menun lisäysvalikon avaava nappi.

Näkymän rakentamiseen päätin käyttää Bootstrap -kirjastoa, jolla saan käyttööni HTML-, CSS- ja JavaScript -sääntöjä, joiden avulla elementtien asettelu sivulla on varsin helppoa. Kirjastosta on myös erittäin paljon apua kehitettäessä sivustoa sopivaksi erikokoisille näytöille ja responsiiviseksi. Bootstrap perustuu grid -järjestelmään. Tämä tarkoittaa käytännössä sivustolla leveyssuuntaan laajenevaa ruudukkoa, jossa on 12 ruutua. Sivun kaikki elementit sijoitetaan ruudukkojen sisälle ja niitä voi myös yhdistää – esimerkiksi yhdelle kuvalle voi varata tilaa kuusi ruudukkoa vasemmalta ja tekstille saman verran oikealta. Sivulla kuva ja tekstialue ovat samankokoisia ja samalla rivillä.

Taulukko 2. Esimerkki Bootstrap -kirjaston grid -järjestelmästä

			col-6		
col-2	col-2	col-2	col-2	col-2	col-2
col-2	col-6		col-4		

Päätin asetella sivun seuraavalla tavalla: napit viikon vaihtamiseen ovat pieniä, joten annan molemmille yhden ruudun sivun oikealle ja vasemmalle puolelle. Jokaiselle viikonpäivälle annan kaksi ruutua. Menun lisäysnappi tulee sijaitsemaan ruokalistan yläpuolella, jota painamalla käyttäjälle avautuu Bootstrapin modal -ponnahdusikkuna. Ponnahdusikkunassa tulee olemaan pudotusvalikko, josta käyttäjä voi valita yhden tietokannassa olevista ruoka-annoksista sekä kalenterinäkymä päivän valitsemiseen. Myöhemmin lisään mahdollisesti muitakin valikoita ponnahdusikkunaan, mutta alkuvaiheessa on tärkeintä vain nähdä, että yhteys tietokannan ja sivun välillä toimii. Lisäominaisuuksia ja valikoita on melko yksinkertaista lisätä kopioimalla jo toimivia ominaisuuksia.

Koska rakennamme sivustoa node.js:n ja Angularin kanssa, pääsemme käsiksi myös npm -paketinhallintajärjestelmään. Sen avulla voimme asentaa paketteja, kuten datepicker -kalenterin päivän valitsemiseen ruoka-annokselle. Yleisimpiä paketteja ovat kuitenkin Express ja socket.io, jotka mekin lataamme ensin projektin alkaessa.

Päivä kuluu siis osaltani ruokalistan runkoa rakentaessa ja suunnitellessa, sekä mahdollisten käytettävien pakettien tutkimiseen. Mielestäni on hyvä olla selkeä suunnitelma jo

ennen sivun rakentamisen aloittamista kaikista tarvittavista ominaisuuksista ja keinoista sen rakentamiseen. Tulen tarvitsemaan esimerkiksi keinon päivämäärien hallitsemiseen ja siirtymiseen viikosta toiseen.

Viikkoanalyysi

Uuden projektin takia vajaa viikkoni kului enimmäkseen tulevien tehtävien kartoittamiseen ja alustamiseen. Sain esimieheltäni listan ominaisuuksista, joita asiakas oli toivonut lopulliseen tuotteeseen ja vanhemman ohjelmoijan kanssa keskustelimme siitä, mikä olisi paras ratkaisu rakentaa eri ominaisuudet sivulle. Tämä osuus projektista tulisi olemaan täysin minun vastuullani, joka on uusi tilaisuus tehdä sivu alusta loppuun itse. Aiemmissa Angularilla tehdyissä projekteissamme olemme työskennelleet yhdessä samojen ominaisuuksien parissa. Tällä kertaa pääsen suunnittelemaan, rakentamaan ja testaamaan tuotteen ja saan nähdä sivun asiakkaan käytössä, kun myöhemmin menemme asentamaan järjestelmän ruokalaan.

Projektin alussa on tärkeää määritellä tarkkaan mitä tuotteelta halutaan: mitkä ovat sen vaatimukset, mihin tarkoitukseen tai tarpeeseen se on rakennettu ja ketkä tulevat sitä käyttämään. Aiemmissa projekteissani olen alkanut rakentamaan sivustoa ilman tarkkaa suunnitelmaa siitä, mitä sivustolla halutaan edes tarkkaan ottaen saavuttaa. Tämä on tuottanut paljon haaskattua aikaa ominaisuuksien rakentamiseen, jotka olen myöhemmin joutunut poistamaan. Näiden kokemuksien pohjalta olen alkanut käyttää enemmän aikaa paremman ja tarkan suunnitelman tekemiseen nykyisissä projekteissa. Selene M. Bowby (2014) kirjoittaa blogissaan ”6 Phases of the Web Site Design and Development Process” sivuston rakentamiseen kuuluvan kuusi eri vaihetta. Varsinainen kehitysvaihe ja sivuston rakennus on vasta neljäs vaihe, kun taas ensimmäinen on tiedon kerääminen, toinen suunnittelu ja kolmas mallintaminen. Haaga-Helia korkeakoulun opinnoissa on myös painotettu aina alkuvaiheiden tärkeyttä opiskelijoille ja nyt työelämässä ymmärrän paremmin niiden tärkeyden.

Bowbyn blogissa olleet kuusi vaihetta nettisivuston suunnittelemiseen ja rakentamiseen ovat seuraavat:

1. Tiedon kerääminen. Tämä on prosessin ensimmäinen vaihe, jossa pyritään vastaamaan kysymyksiin kuten mikä on sivuston tarkoitus, mitkä ovat sivuston tavoitteet, ketkä ovat sivuston vierailijoita ja minkälaista sisältöä sivustolle halutaan laittaa.
2. Suunnittelu. Mitkä ovat sivuston pääaiheita, mitä sivuja ja alasivuja sivustolle tulee, mitä tulee näkymään ensimmäisessä näkymässä vierailijalle, miten sivusto tullaan

rakentamaan – nämä ovat muutamia kysymyksiä, joita pohditaan tässä vaiheessa. Toisin sanoen tavoitteena on koota aiemmin kerätty tieto ja jäsentää se tärkeysjärjestykseen.

3. Mallintaminen. Sivusto mallinnetaan joko kuvallisesti tai rakentamalla kevyt alkuversio. Tässä vaiheessa yleensä näytetään suunnitelma asiakkaalle ja visualisaatio tulevasta sivustosta auttaa asiakasta päättämään toivotuista muutoksista etukäteen.
4. Kehittäminen. Vasta neljännessä vaiheessa aletaan kehittämään sivustoa käytännön tasolla. Kehitystyössä on hyvä tapa rakentaa sellaisella tavalla, joka antaa projektin omistajan tarkastella sivuston rakentumista ja jonka pohjalta hän voi toivoa lisämuutoksia ja korjauksia.
5. Testaaminen ja julkaisu. Nettisivuston kaikki ominaisuudet, kuten esimerkiksi kontaktilomakkeet, linkit ja animaatiot testataan eri resoluutioilla ja selaimilla. Tämän jälkeen sivusto ladataan palvelimelle ja testataan uudelleen. Tässä vaiheessa sivusto optimoidaan hakukoneita varten, jotta ne löytävät sivuston helpommin.
6. Ylläpitäminen. Joskus sivuston omistaja toivoo päivityksiä ja uutta sisältöä, joten tässä viimeisessä vaiheessa voidaan jälkikäteen tehdä muutoksia sivuston sisältöön.

Viikon aika tein kaksi ensimmäistä vaihetta ja osittain kolmannen. Sain tiedot esimieheltäni sivun tavoitteista ja mihin tarkoitukseen se tulee, jonka jälkeen keskustelin vanhemman kehittäjän kanssa, miten alkaisin rakentamaan sivustoa ja mitkä ovat ensimmäiset askeleet kehitystyössä. Päätimme rakentaa sivuston AngularJS -ohjelmistokehyksellä ja node.js -ajoympäristöllä. AngularJS auttaa meitä rakentamaan sivuston MVC -ohjelmistoarkkitehtuurin mukaisesti. Brad Dayleyn kirjassa Node.js, MongoDB, and AngularJS Web Development (2014, 4) hän listaa AngularJS:n kuusi eri hyötyä, joista yksi on koodin selkeys. Tästä on ollut jo aiemmin paljon hyötyä tässä pienessä yrityksessä, sillä usein joudumme työskentelemään eri projekteissa samaan aikaan. Kun rakenne on selkeä ja koodi on helposti ymmärrettävää jokaisessa projektissa, ei kehittäjällä kulu aikaa tutustua ohjelman toimintamenetelmään.

Ensi viikolla pääsen aloittamaan kehitystyön. Todennäköisesti tulen myös työskentelemään WordPress -sivustojen parissa. Olen kuitenkin tottunut siirtymään projektista toiseen, sillä pienessä yrityksessä on vaikeaa varata pieniä resursseja vain yhteen projektiin. Vain suurissa projekteissa siirtyy kehittäjä tekemään vain yhtä asiaa. Suurimmissa projekteissa siirtyvät kaikki kehittäjät työskentelemään yhdessä, jolloin kaikki muu siirtyy myöhemmäksi.

3.4 Seurantaviikko 41

Maanantai 9.10.2017

Viime viikolla sain suunniteltua projektin alkuaskeleet ja tiedän jo viikon alussa mitä minun tulee tehdä. Viime viikon raportissa kerroin, kuinka sivuston ulkoasu tullaan rakentamaan Bootstrapin pohjalle. Ehdin aloittaa jo hieman projektin alustamista ja ulkoasun rakenteiden luomista. Tänään tavoitteenani on saada projekti hyvään alkuun, jonka pohjalle voin ruveta rakentamaan itse ruokalistan ominaisuuksia. Selkeän rakenteen kanssa on helppoa lisätä ja poistaa ominaisuuksia.

Tulemme rakentamaan projektia MVC -arkkitehtuurin pohjalle. Koska suurin osa projektista kopioidaan yrityksen aiemmasta projektista, ei meidän tarvitse rakentaa ohjelmaa alusta asti uudelleen. Sivullani tulen hakemaan ruokalistat SQL -tietokannasta GET -komennolla sekä admin -näkymään sisäänkirjautunut henkilö voi lisätä, muokata ja poistaa ruoka-annoksia päivälle. Käytännössä tämä tulee tapahtumaan siten, että kun käyttäjä tekee MVC -arkkitehtuurin mukaan määritellyllä V-kirjainta edustavalla ”view”-sivulla jotain - kuten siirtyy viikkojen välillä, tekee muutoksia ruokalistaan tai jotain muuta - aktivoituu silloin C-kirjainta edustavassa ”controller”-tiedostossa siihen tapahtumaan määritelty komento.
Tämä voi olla esimerkiksi seuraavanlainen tapahtuma: ruokamenu.html -tiedostossa on rivi:

<td class="menuitem" ng-repeat="artikkeli in tavarat>,

joka AngularJS:n ng-repeat -komennolla antaa käskyn controllerille tuoda kaikki artikkelit taulukosta viikonpäivät. ”Controller”-tasolla taas käsittelemme käskyn seuraavalla komennolla:

```
$scope.tavarat = [];
$http.get('/menu/getKaikkiTavarat').then(function(response){
 $scope.tavarat = response.data;
});
```

Tämän jälkeen käsky siirtyy tasolle, jota kutsumme yrityksessä nimellä ”route”. Tämä tiedosto siirtää käskyn edelleen ”model”-tasolle, jossa lähetämme kyselyn tietokantaan ja saamme vastauksen, joka kulkeutuu samaa tietä pitkin takaisin käyttäjätasolle. Route-tasolla ohjaamme edellä mainitun pyynnön tasolle ”model” tällä komennolla:

```
var router = require('express').Router();
var menu = require('./../models/menu');
router.get('/getKaikkiTavarat', function (req, res) {
    menu.lataaKaikkiTavarat(function (result) {
        res.json(result);
    });
});
```

Kun olemme lähettäneet pyynnön viimeiselle tasolle model, voimme laatia esimerkiksi seuraavan pyynnön:

```
var connection = require('./db');
module.exports.lataaKaikkiTavarat = function(callback) {
    connection.query("SELECT * FROM tavarat", function(err, rows) {
        callback(rows);
    });
};
```

Käytännössä kaikki pyynnöt ja syötteet mitä haluamme laittaa tietokantaan ohjelman kautta, kulkeutuu samaa reittiä pitkin. Tämä tekee MVC -arkkitehtuurista selkeän ja auttaa kehittäjiä ohjelman toimintatavan ymmärtämisessä. Sen avulla esimerkiksi uuteen projektiin siirtyminen on helpompaa, sillä arkkitehtuuri on toimintatavaltaan jo entuudestaan tunnettu. Myös mahdollisten virheiden tunnistaminen on helpompaa, sillä tieto kulkeutuu aina samalla tavalla.

Rakennan MVC -mallin mukaiset tiedostot nimeltään ”weeklymenu” projektin kansioihin, joille olemme antaneet nimet controllers, routes, models ja templates. Kirjoitan tiedostoihin perusrungon, johon voin myöhemmin ruveta kirjoittamaan koodia. Main.js -tiedostoon kirjoitan rivit

var weeklymenu = require('./routes/weeklymenu'); sekä *app.use('/weeklymenu', weeklymenu);.*

Tämän avulla express tietää ohjata käyttäjän oikealle sivulle ja ohjaa menevän tiedon oikein.

Tiistai 10.10.2017

Tavoitteeni päivälle on saada kuljetettua tietoa tietokannasta, sillä vielä tässä vaiheessa ei minun tarvitse viedä mitään sinne. Olen tyytyväinen, jos saan tänään tai huomenna tuotua jotain sivulle näkyviin. Eilen sain rakennettua tiedostorakenteen, joka on yhteneväinen muun projektin kanssa. Tiedon kulkeutuminen toimii samalla tavalla kuin sivuston muissa kohteissa, josta voin katsoa mallia, mikäli sitä tarvitsen.

Ensin haluan varmistua, että saan luotua jotain kontrollerissa joka myös näkyisi sivulla. Teen ensin listan päivistä kirjoittamalla rivin, joka luo listan saksan kielisistä viikonpäivistä:

$scope.workweekdays = ['Montag','Dienstag','Mittwoch','Donnerstag','Freitag'];.

Sen jälkeen tulostan listan sivulle antamalla div -elementille ng-repeat -attribuutin ja sille arvon ”day in workweekdays”. Muutamien korjauksien jälkeen sain listan näkymään sivulla, joten yhteys näiden kahden asteen välillä toimi.

Seuraavaksi haluan rakentaa pienen tietokannan lokaalissa kehitysympäristössä. Teen pienen tietokantataulun XAMPP -ohjelman mukana tulevalla phpMyAdmin -tietokannan hallintatyökalulla, johon laitan eri ruoka-annoksia. Asetan myös ohjelman hakemaan tietokannan db.js -tiedostossa node.js:n mysql -liitännäisen avulla. Sen jälkeen kirjoitan models -kansiossa olevaan tiedostoon tekstin koodia, jossa käytän ”query” -komentoa. Kyseisessä koodissa lähetän tietokantaan seuraavan kyselyn:

*SELECT * FROM projekti.ateriat.*

Tällä tavalla pyydän tietokannasta kaikki ateriat. Kyselyn palattua se siirtyy luomaani if -lausekkeeseen, joka tarkistaa onko tapahtunut virhettä. Kysely toimii haluamallani tavalla ja se tulostaa konsoliin kyselyn tuloksen.

Seuraavana työpäivänä pyrin saamaan tiedon kulkeutumaan eteenpäin ohjelmassa. Olen kuitenkin tyytyväinen siihen, että saan tuotua tietokannasta viestin ohjelmaan ja nyt haasteena on siis vain kuljettaa tieto sivulle ja näkyviin käyttäjälle.

Perjantai 13.10.2017

Viikon aikana olin saanut sähköpostiini viestin painoyritykseltä, jolle olen rakentanut WordPress -sivustoa. He olivat yrityksen sisällä tutkineet sivua tarkemmin ja keskustelleet sivuston kuvista. Viestissä he esittivät toiveen, että voisimme ottaa lisää kuvia yrityksestä,

henkilöstöstä ja tuotteista. Keskustelin esimieheni kanssa ja hän asetti kuvien päivittämisen päivän tehtäväkseni. Tällä kertaa tulen ottamaan kuvat itse ja muokkaamaan ne tarvittaessa Gimp -kuvanmuokkausohjelmalla. Avukseni saan käyttööni yrityksen kameran sekä valkokankaan ja kaksi kohdevalaisinta.

Olin aiemmin saanut yrityksestä kuvia, mutta tarkasteltuaan sivustoa tarkemmin he toivoivat lisää kuvia. Asiakasyritys sijaitsee Stuttgartissa. Matkani sinne kestää Reutlingenista noin tunnin verran. Saavuttuani yrityksen johtaja antoi listan kuvattavista kohteista ja kävimme lyhyesti läpi sivustoa ja minkälaisen kuvan he haluavat eri osioihin. Tämän johdosta tiedän muokata kuvan resoluutiota. Pyrin ottamaan mahdollisimman monta kuvaa ja lopulta tehtävä oli helppo tarkkojen ohjeiden ansiosta. Muutaman tunnin jälkeen sain valmiiksi kuvien ottamisen yrityksessä ja lähdin lounaan jälkeen takaisin Reutlingeniin.

Sain mukaani muutaman tuotteen, joista otin myöhemmin kuvia valkokangasta vasten. Sivuston väriteemana on tummanoranssi, vaaleanvihreä, tummansininen ja taustavärinä valkoinen. Tämän takia pyrin ottamaan kuvat tuotteista mahdollisimman valkoista taustaa vasten valkokankaan ja kohdevalaisinten avulla, jotka myöhemmin muokkasin edelleen valkoisemmaksi. Olin aiemmin valinnut väriteeman yrityksen logon innoittamana, joka oli vaalean vihreä, musta ja valkoinen. Valitsemiseen olin käyttänyt Adobe Color Wheel -sivustoa, joka auttaa löytämään toisiinsa sopivat värit.

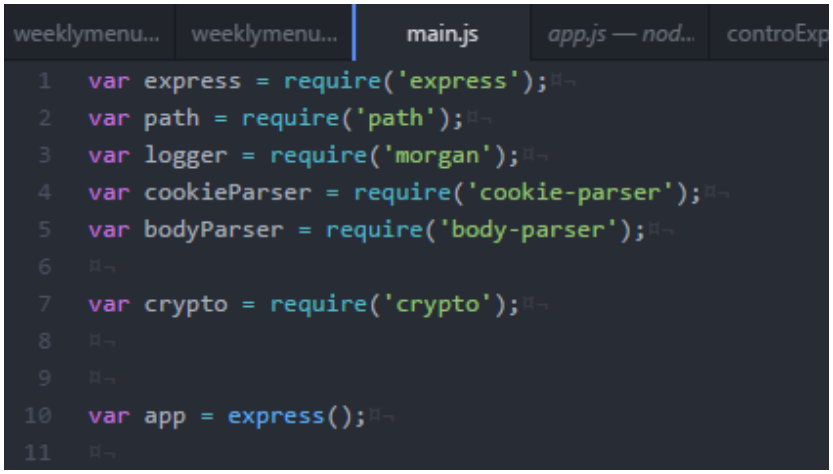
Loppupäivän tarkastelin ja valitsin käytettävät kuvat. Muokkasin käytettäviä tuotekuvia Gimpin avulla, jotta ne sulautuisivat paremmin sivun valkoisen taustavärin kanssa.

Viikkoanalyysi

Tavoitteeni viikolle oli saada uusi projekti mahdollisimman hyvin alkuun ja saada perustoiminnallisuudet rakennettua. Alkuaskeleista tärkein on mielestäni saada tieto kulkeutumaan käyttöliittymästä tietokantaan ja samoin toisinpäin. Tämän jälkeen on yksinkertaisempaa lisätä ja poistaa ominaisuuksia, sillä käytännössä kaikki tieto liikkuu ohjelmassa samalla tavalla. Tämä selkeyttää huomattavasti sivuston toimintaa ja ohjelmoimista. Viikon viimeisenä päivänä näkyi yrityksen pienen koon vaikutus kehittäjien työssä, sillä jouduin vaihtamaan toisenlaiseen työtehtävään nopeasti ja keskittymään siihen täysin. Olen jo ehtinyt tottumaan tähän työtapaan ja oppinut, että on tärkeää pystyä keskittämään huomio uuteen tehtävään. On myös yleistä, että joudun palaamaan vanhoihin työtehtäviini myöhemmin, jolloin olen jo ehtinyt unohtaa ohjelman rakenteen ja toimintatavan. Tällöin on hyödyllistä, jos on tehnyt aiemmista projekteista muistiinpanoja ja kaavioita itselleen, joka tekee siirtymävaiheesta helpomman.

Kuten mainitsin viikon ensimmäisessä merkinnässä, käytämme sivustoissamme MVC -ohjelmistoarkkitehtuurimallia. Yrityksen toiminta helpottuu huomattavasti, kun käytämme kyseistä arkkitehtuuria, siihen liittyvää tiedostorakennetta ja tiedon liikkumista ohjelmassa. Yritys koki pienen kriisivaiheen viime vuoden alussa, kun yksi vanhempi kehittäjä päätti vaihtaa työpaikkaa lyhyellä varoitusajalla. Hänen lähtönsä tuotti kaksi ongelmaa: hänen työpanoksensa tuli korvata uudella työntekijällä sekä hänen mukanaan lähti myös paljon tietoa hänen rakentamistaan ohjelmista. Edelleen joudumme yrityksessä tutkimaan kuinka esimerkiksi hänen rakentamansa RFID -kortinlukijatekniikkaan pohjautuvat ohjelmat toimivat.

Käytämme Express -kehystä kaikissa projekteissamme, joissa käytämme AngularJS -ohjelmistokehystä sekä node.js -ajoympäristöä. Express on kevyt moduuli, joka tekee node.js:n http -moduulista helpommin ymmärrettävän. Se myös laajentaa http -moduulia ja tekee kehittäjälle helpommaksi käsitellä reittejä (server routes), palvelimelta tulevia vastauksia (response), evästeitä ja HTTP -pyyntöjä (request.) (Dayley 2014, 357.) Sen avulla request/response -sykli, jossa lähetämme pyynnön palvelimelle ja vastaanotamme vastauksen, on selkeämpi. Ennen kaikkea voimme luoda Expressin avulla yhteyden palvelimen ja ohjelman välille, sekä saamme tiedon kulkeutumaan ohjelman sisällä helpommin ymmärrettävällä tavalla.



Kuva 5. Main.js -tiedostossa luotu Express -palvelin

Request/response -sykli kulkeutuu ohjelman sisällä seuraavalla tavalla: käyttäjä tekee käyttöliittymässä jotain, joka aktivoi Angular:JS:n scopen kontrollerissa. Aktivoitu scope käynnistää HTTP request -metodin (esimerkiksi GET tai POST) (Dayley 2014, 360), joka ohjautuu edelleen routes -kansiossa olevaan Expressin router -lausekkeeseen. Tässä lausekkeessa lähetämme pyynnön edelleen models -kansiossa olevaan tiedostoon ja

käännämme JSON:in avulla JavaScript -objektit String -muotoon (sekä toisin päin.) Models -kansiossa lähetämme palvelimelle SQL -komennon ja saamme vastauksen (response.) Käyn tasot tarkemmin läpi alla olevassa taulukossa.

Taulukko 3. Tiedon kulkeutuminen MVC -arkkitehtuurimallin mukaan rakennetussa sivustossamme. Vasemmassa sarakkeessa näkyy tiedon kulkeutumisen järjestys ja tason nimi. Oikeassa sarakkeessa tason selitys ja esimerkki kirjoitetusta koodista

1	Käyttöliittymä	Käyttäjä pyytää ruokalistaa AngularJS:n ng-repeat -komennolla, joka palauttaa listan kaikki tuotteet.	<td ng-repeat="article in items”> {{article.Artikelname}} </td>
2	Controller	Controller luo ensin tyhjän listan ja lähettää GET -pyynnön eteenpäin. Kun se vastaanottaa vastauksen, $scope.items täyttyy vastaanotetun listan objekteilla.	$scope.items = []; $http.get('/menu/getAllItems') .then(function(response) { $scope.items = response.data; });

3	Route	Ensin otamme käyttöön Expressin Router -ominaisuuden ja määrittelemme reitin models -kansioissa oleviin tiedostoihin, jotta pyyntömme ohjautuu oikealle SQL -komennolle.	var router =
			require('express').Router();
			var menu =
			require('./../models/menu');
			
			router.get('/getAllItems', function (req, res) {
			menu.loadAllItems(
			function (result) {
			res.json(result);
			});
			});

4	Model	Model -tasolla lähetämme SQL -komennon palvelimelle. Vastauksen lähetämme callback -komennolla takaisin samaa reittiä käyttöliittymään.	var connection = require('./db');
			var async = require('async');
			
			module.exports.loadAllItems = function(callback) {
			connection.query("SELECT * FROM intection.bc_weeklymenu", function(err, rows) {
			callback(rows);
			});
			};

3.5 Seurantaviikko 42

Maanantai 16.10.2017

Aloitin työviikon asettamalla painotalon sivustolle viime viikolla ottamani ja muokkaamani kuvat. WordPressin käyttöliittymän avulla kuvien vaihto ja järjestely on erittäin helppoa, joten minulta ei kulu tähän kovinkaan kauaa aikaa. Jouduin muuttamaan hieman kuvien CSS -tyyliasetuksia ja kuvien kokoa, sillä olin unohtanut muokata ne pienemmäksi. Yleensä ennen kuin laitan sivulle kuvia, muutan ne tiedostokooltaan pienemmiksi. Varsin usein kuvat ovat kooltaan liian isoja tarkoitustaan varten ja jos sivulla näkyy paljon kuvia voi sivun lataamisessa kestää liian kauan. Tällä tavalla saan sivuston siis latautumaan nopeammin.

Tämän jälkeen menimme esimieheni ja vanhemman kehittäjän kanssa lounaalle, jonka aikana juttelimme painotalolle rakentamani WordPress -sivuston ja ruokalistaprojektin nykytilasta. Keskustelun pohjalta esimieheni ehdotti, että viimeistelen WordPress -sivuston rakentamisen ja käytän aikani tällä ja seuraavalla viikolla sen viimeistelemiseen. Tämä sopi minulle hyvin, sillä viime viikkojen aikana on ollut vaikea keskittyä yhteen työtehtävään.

Lounaan jälkeen päätin testata sivuston latausnopeutta. Ensin testaan Firefox -selaimen yksityisessä ikkunassa varmistaakseni, että sivusto avautuisi samalla tavalla kuin ensimmäistä kertaa vierailevalle käyttäjälle. Yksityisen ikkunan kautta avattaessa saa todellisen kuvan latautumisnopeudesta, sillä se ei käytä välimuistiin tallennettuja tiedostoja tai tietoja sivusta.
Huomaan, että sivusto latautuu melko hitaasti. Tämä on ongelmallista, sillä keskivertainen vierailija internetissä ei odota sivuston latautumista kahta sekuntia pidempään. Päätän testata sivuston GTmetrix -testisivuston avulla, joka auttaa analysoimaan miten latausaikaa voisi nopeuttaa. Se antaa kehittäjälle ladattavien tiedostojen koosta, latausajasta sekä järjestyksestä, jossa tiedostot on ladattu.

Testien jälkeen huomaan ongelmaksi muodostuneen kaksi asiaa. Ensinnäkin jotkin aiemmin lisäämäni kuvat ovat edelleen tarpeettoman hyvälaatuisia ja sitä kautta tiedostokooltaan liian suuria. Tämä ei kuitenkaan ole minulle uusi asia, joten sen korjaamiseen ei menisi kovinkaan kauan aikaa. Toinen ongelma tulisi viemään minulta hieman enemmän aikaa, sillä testien perusteella sivusto ei käytä Gzip -kompressointia, joka pakkaisi ladattavat tiedostot pienempään zip -tiedostomuotoon.

Päivän lopuksi ajan WordPress -käyttöliittymässä WP Smush -liitännäisen ohjelman, joka auttaa hieman sivuston kuvien tiedostokoon pienentämisessä. Usein teen tämän itse Gimp -kuvankäsittelyohjelmalla ennen kuin lataan kuvat sivustolle ja liitännäisen avulla saan pienennettyä kuvia vielä hieman.

Tiistai 17.10.2017

Aloitan päiväni siitä mistä eilen lopetin, eli kuvien pienentämisestä. Olen jo useaan kertaan joutunut muokkaamaan kuvia monesta syystä ja tällä kertaa muokkaan niitä jälleen pienemmäksi. Kuvien täytyy olla laadultaan tarpeeksi hyviä ja ammattimaisen näköisiä, mutta kuitenkin tarpeeksi pieniä, jottei käyttäjä joudu lataamaan niitä kovin pitkään. Joskus tämä osoittautuu hieman haasteelliseksi – etenkin, jos sivustolla tulee olemaan useita erillisiä kuvia. Samoin kuin eilen, muokkaan kuvia ensin Gimp -kuvankäsittelyohjelmalla, lataan ne sivustolle ja ajan lopuksi WP Smush -liitännäisen.

Seuraavaksi alan selvittämään toista eilen havaitsemaani ongelmaa, nimittäin Gzip -kompressoinnin puutetta. Olin aiemmin käyttänyt projektissa W3 Total Cache -liitännäistä, joka monien ominaisuuksien lisäksi kompressoi myös tiedostoja. Liitännäinen oli kuitenkin liian monimutkainen ja jouduin poistamaan sen. Esittelin jo silloin vaihtoehtoisen liitännäisen nimeltään WP Super Cache, jota olin ajatellut kokeilla tässä projektissa. Se on toiminnoiltaan hieman yksinkertaisempi ja tuottaa sivustolle Gzip -kompressoinnin. Päätän siis asentaa kyseisen liitännäisen ja kokeilla sitä. Asennan liitännäisen WordPressin käyttöliittymän kautta, joka on helppo prosessi. Sen jälkeen menen liitännäisen asetuksiin ja asetan sen kompressoimaan sivuston tiedostot.

Testaan sivustoa uudestaan samalla tavalla kuin eilenkin. Tarkastelen latausnopeutta ja tiedostojen kokoa, jotka joudun lataamaan, kun vierailen sivulla. Ajan myös aiemmin mainitsemani testausohjelman. Testeistä sain selville, että latausnopeus oli nyt melkein puolet nopeampi. Muokkaamani kuvat ovat huomattavasti pienempiä ja vievät vähemmän aikaa sekä tiedostot saapuivat käyttäjälle pakatussa muodossa.

Perjantai 20.10.2017

Saapuessani aamulla töihin tarkastan aina sähköpostini siltä varalta, jos joku asiakkaitamme on pyytänyt muutoksia tai jos jokin kehittämäni WordPress -sivusto tarvitsee päivittämistä. Huomaan, että olen saanut viikon aikana paljon roskapostia kehityksen alla olevan sivustoni kontaktilomakkeen kautta. Alkureaktioni on huoli siitä, jos sivustolle on yritetty murtautua tai käyttää SQL-injektiota tietokannan muuttamiseen. Tarkasteltuani tar-

kemmin saapuneita roskaviestejä tajuan kuitenkin lähettäjän olleen botti, joka yrittää lähettää linkkejä. Kuitenkin botin lähettämien viestien ansiosta huomaan heikkouden kontaktilomakkeessa, jota en olisi välttämättä aiemmin huomannut. Olisi erittäin ikävää, jos asiakkaamme joutuisi valittamaan sivustolta saapuvasta roskapostista. Päätän siis ottaa tämän päivän tehtäväkseni kontaktilomakkeen turvaamisen.

Bottien estämiseksi on olemassa monta keinoa. Jokaisessa niistä on kuitenkin omat haittansa ja hyötynsä. Eräs hyvä tapa on esimerkiksi pakottaa käyttäjät rekisteröitymään sivustolle ennen kuin he voivat lähettää viestin. Tämä keino ei tässä projektissa ole kuitenkaan kovin käytännöllinen, sillä sivuston tavoite on saada asiakas ottamaan yhteyttä. Viestin lähettämisen tulee täten olla mahdollisimman helppoa ja vaivatonta. Tarvitsen siis tavan estää roskapostin saapumisen häiritsemättä kuitenkaan asiakkaan lomakkeentäyttöprosessia.

Päädyn lopulta kokeilemaan menetelmää nimeltään ”honeypot” (suom. hunajapurkki.) Menetelmän ajatuksena on luoda lomakkeeseen yksi täyttökenttä, joka on ihmiskäyttäjälle näkymätön, mutta jonka botti tunnistaa ja täyttää. Jos kyseinen kohta on täytetty ei sähköpostia lähetetä. Yksi menetelmän hyödyistä on, ettei se vaadi käyttäjältä mitään lisätoimenpiteitä viestin lähettämiseksi. Toisaalta se ei kuitenkaan ole kaikista varmin tapa estää roskaviestien saamista, sillä osa boteista osaa välttää kentän täyttämisen.

”Hunajapurkin” luonti osoittautui kohtalaisen helpoksi tehtäväksi. WordPressin suosion ansiosta lähes joka tarkoitukseen on olemassa liitännäisiä, joten kaikkea ei tarvitse rakentaa itse. Päätän kokeilla liitännäistä nimeltään ”Contact Form 7 honeypot”, joka on luotu erityisesti käyttämälleni ”Contact Form 7” -lomakeliitännäistä varten. Asennan liitännäisen tavalliseen tapaan WordPress -käyttöliittymän avulla ja lisään uuden täyttökentän kontaktilomakkeeseen ohjeiden mukaisesti.

Loppupäivän käytän pienten ulkoasuvirheiden korjaamiseen ja keskustelen vanhemman kehittäjän kanssa, jos kontaktilomake tarvitsee vahvemman suojauksen spam-viestien estämiseksi. Ensi viikolla yritän rakentaa vielä toisen tason varmuuden vuoksi.

Viikkoanalyysi

Odotin viikon alussa työskenteleväni uudessa projektissa ruokalistan parissa, mutta lopulta aikani kului jälleen painotalon sivuston parissa. Kuvien lisääminen, muokkaaminen ja asettelu on aihe, johon joudun palaamaan aina uudestaan erilaisista syistä. Yleisimpiä syitä on asiakkaan esittämä toive muutoksesta, latausajan optimointi tai itse huomaamani

tarvittu muutos. Kuvia tarvitaan aina sivustolla ja toimivat hyvin epäsuoran viestinnän välineenä sivuston vierailijalle. Jokaisen sivustolla olevan kuvan täytyy kuitenkin palvella lopullista sivuston tavoitetta. Joskus tiettyä kuvaa ei tarvita lainkaan, jolloin se vain lisää latausaikaa. Kehittäjän on kannattavaa kysyä itseltään, tarvitaanko esimerkiksi taustalla olevaa kuvaa tai voisiko kuvan muodossa olevan tekstin toteuttaa jollain muulla keinolla. Myös kuvan tallennusformaatilla on väliä, sillä jotkin formaatit aiheuttavat enemmän latausaikaa kuin toiset. Täytyy myös miettiä, tullaanko kuvassa käyttämään jotain animaatiota tai läpinäkyvää taustaa. (Grigorik 2018)

Tarve kuvien muokkaamiseen syntyy latausnopeuden optimoinnin tarpeesta. Sivuston latausajan optimointi on yksi kehittäjän tärkeimmistä tehtävistä, sillä pitkä latausaika karkottaa vierailijoita. Digitaaliseen markkinointiin erikoistunut yritys Pear Analytics (2009) kokosi yhteen tutkimuksia latausaikojen vaikutuksesta sivuston vierailijan kärsivällisyyteen odottaa. Amazonin tuottama tutkimus vuodelta 2007 osoitti, että jokaisen 100 millisekunnin lisäys vierailijan latausaikaan vähensi Amazon.com:n myyntitulosta yhdellä prosentilla. Toisessa Googlen tuottamassa tutkimuksessa vuodelta 2006 taas selvisi, että hakukoneella tehdyn hakuajan muutos 0.4 sekunnista 0.9 sekuntiin vähensi internetliikennettä ja palvelun mainostuottoa 20%:lla.

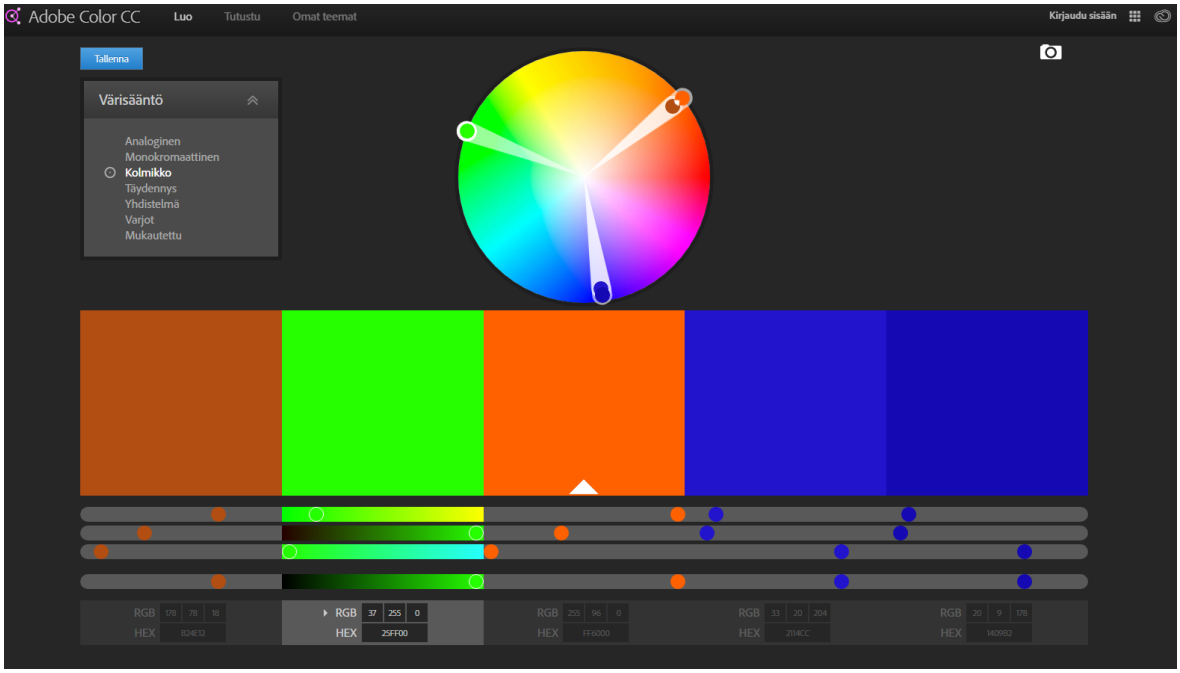
Käytin viikon aikana melko paljon aikaa sivuston latausajan testaamiseen ja optimointiin. Testaamiseen käytin GTmetrix -sivustoa sekä tarkastelemalla Google Chromen ja Firefoxin kehittäjäikkunaa. Testien perusteella ongelmaksi muodostuivat neljä pääasiaa: kuvien koko, tiedostojen yhdistäminen, välimuistin käyttö ja Gzip -kompressointi.
Kuvien muokkaukseen käytin WordPressin Wp Smush -liitännäistä sekä Gimp -kuvankäsittelyohjelmaa. Tiedostojen yhdistämisellä tarkoitan JavaScript-, CSS- ja HTML -tiedostojen yhdistämistä, jolloin palvelimen ei tarvitse lähettää useita tiedostoja sivuston vierailijalle vaan vain yhden tiedoston kaikista tiedostomuodoista. Tämä toimenpide on kuitenkin hieman hankalaa toteuttaa WordPress -kehityksessä, joten jätin sen kehittämisen myöhemmälle vaiheelle. Välimuistin käytön toteuttamiseen asensin WP Super Cache -liitännäisen. Välimuistia käyttämällä voi vähentää palvelimelle lähetettyjen pyyntöjen määrää. (WordPress.org.) Gzip -kompressoinnilla palvelimen vierailijalle lähettämät vastaukset ovat varsinaista tiedostoa huomattavasti pienemmässä pakatussa muodossa ja vierailija purkaa paketin. Se voi parhaimmillaan vähentää lähetettyjen vastauksien kokoa jopa 90%:lla. (Google PageSpeed, 2018.)

Sivuston yleistä ulkoasua rakentaessa tulee muistaa pitää sivu miellyttävän näköisenä sekä sivuston yleinen olemus yksinkertaisena. Sivuston värimaailma tulee siis valita huolella, sillä värit viestivät ja herättävät ihmisissä eri tuntoja. Esimerkiksi sininen väri herättää

ihmisessä luottamusta, kun taas oranssi viestii innovatiivisuudesta. (Williams, Damstra & Stern, 342)

Tavoitteena on, että sivuston vierailija innostuu näkemästään ja löytää helposti sen mitä etsii. Kaikki ylimääräinen visuaalinen materiaali mikä häiritsee vierailijaa, tulee poistaa. Materiaalin tulee edesauttaa haluttua sivuston vierailun lopputulosta. Käyttäjät eivät välttämättä järjestä mielessään sivuston sisältöä ja eri osioiden paikkaa samalla tavalla kuin kehittäjä, joten on tärkeää, että sivustolla pääsee montaa eri reittiä haluttuun kohteeseen (Williams, Damstra & Stern, 343).

Olen yrityksessä oppinut ulkoasun ja värimaailman suunnittelun tärkeyden. Käytän aina sivustossa käytettävien värien valitsemiseen Adobe Color -ohjelmaa, jonka minulle vanhempi ohjelmoija opetti. Sen avulla saa valittua hyvin yhteen sopivia väriteemoja ja päävärin kanssa sopivat sivuvärit (Locke 2017). Teen päivittäin muutoksia sivustoihin ja pyrin rakentamaan sivustosta mahdollisimman houkuttelevan. Pyrin asettamaan itseni sivuston vierailijan asemaan, joka avaa sivun ensimmäistä kertaa. Yritän miettiä, onko ensinäkymässä liikaa värejä, onko tärkein elementti helposti löydettävissä tai onko sivustossa jotain mikä vie vierailijan huomion pois tärkeämmistä elementeistä.



Kuva 6. Adobe Color on ilmainen ohjelma, jonka avulla saa valittua yhteensopivat värit

Viikon aikana sain myös roskapostia sivun kontaktilomakkeen kautta. Roskapostin lähettäjän oli hyvin todennäköisesti netissä pyörivä botti, joka pyrkii etsimään täytettäviä lomakkeita tai kommenttiosioita ja lähettämään viestejä niiden kautta. Tämä ei saa olla

mahdollista, sillä sivuston toimeksiantaja tulee jatkossa vastaanottamaan nämä viestit. Viestejä tulee saapumaan vielä enemmän, kun sivusto on varsinaisessa käytössä. Bottien estämiseksi on olemassa lukuisia eri vaihtoehtoja, joista valitsin honeypot -menetelmän. Tällä menetelmällä luon käyttäjälle piilotetun kentän, jonka kuitenkin netissä liikkuvat botit huomaavat ja täyttävät.

3.6 Seurantaviikko 43

Maanantai 23.10.2017

Viikonloppuna mietin, miten saisin lisättyä suojausta painotalon sivuston kontaktilomakkeeseen roskaviestejä varten. Olin miettinyt erilaisia vaihtoehtoja, kuten CAPTCHA -kuvanvarmennusta tai muita liitännäisiä honeypot:n lisäksi. Lopulta päädyin kokeilemaan turvakysymysten luontia keskusteltuani asiasta vanhemman kehittäjän kanssa. Turvakysymys esitetään käyttäjälle, jotta varmistetaan sen olevan henkilö. Lähettääkseen lomakkeen, käyttäjän tulee vastata helppoon kysymykseen; kuten paljonko saadaan, kun yhteen lisätään kolme tai mikä on Ranskan pääkaupunki. Epäilin ensin sen toteuttamista, sillä lomakkeen tavoite on tehdä asiakkaalle mahdollisimman helpoksi ottaa yhteyttä sen avulla. Turvakysymyksen luonti olisi kuitenkin melko yksinkertaista ja toimii vanhemman kehittäjän mukaan yleensä hyvin roskaviestien välttämiseksi. Päätän siis kokeilla sen rakentamista. Turvakysymysten luontia helpottaa käyttämäni Contact Form 7 -liitännäiseen sisäänrakennettu Quiz -ominaisuus. Lisäämällä kontaktilomakkeen koodiin rivin

[quiz kysely "Mikä on Suomen pääkaupunki?|Helsinki"]

saan luotua kontaktilomakkeeseen kysymyksen, johon pitää antaa määrittämäni vastaus. Turvakysymyksiä voi olla useita, joista sivu arpoo yhden käyttäjän vastattavaksi. Päätän kehittää turvakysymykset lomakkeeseen ja nähdä, jos se on sopiva ratkaisu. Myöhemmin minun tulee keskustella esimieheni ja asiakkaan kanssa toteutuksen toimivuudesta ja on helpompaa näyttää toimivaa ratkaisua. Ratkaistavaksi tehtäväksi muodostuu myös kysymyksen sijoittelu, sillä olen jo ehtinyt suunnitella ja rakentaa lomakkeen ulkoasun. Esteettisesti lomakkeen kenttien tulee olla järkevässä järjestyksessä, eli pakosti täytettävät kentät – kuten etunimi, sukunimi ja sähköposti – ovat ensimmäisenä. Viimeisenä minulla on yrityksen nimi sekä textarea -kentässä olevat kommentit. Täytettäviä kenttiä minulla on kahdessa sarakkeessa, joten tärkeimmät kentät sijaitsevat ylhäällä vasemmalla ja vähemmän tärkeät alhaalla oikealla. Päädyn muokkaamaan lomaketta niin, että lähetyspainike siirtyy oikeaan alanurkkaan kommentit-osion alle siinä missä se ennen oli omalla rivillään lomakkeen alla. Päivän päätteeksi korjasin pieniä bugeja sivulla, lisäsin erilaisia

turvakysymyksiä ja testasin niitä. Lomake näytti mielestäni nyt hyvältä ja toivottavasti myös esimieheni ja asiakkaamme tulevat olemaan kanssani samaa mieltä.

Tiistai 24.10.2017

Saavuttuani aamulla tarkastelin sivua ja mietin, mitä askeleita voisin seuraavaksi ottaa WordPress -projektin kanssa. Muistin aina pohtineeni WordPressin turvallisuutta erityisesti siinä vaiheessa, kun selviää sivuston olevan tehty kyseisellä sisällönhallintajärjestelmällä. Kaikki asiaa tuntevat tietävät, että oletusarvoisesti WordPressin admin-näkymään pääsee alasivulta /admin. Jos joku haluaisi siis hyökätä sivustolle, heidän täytyisi vain yrittää päästä tälle alasivulle. Sen jälkeen he voisivat yrittää erilaisia hyökkäyksiä, kuten esimerkiksi ”brute force” -hyökkäystä, jossa pyritään aggressiivisesti löytämään oikeat kirjautumistunnukset. Tämän innoittamana päätin keskustella asiasta vanhemman kehittäjän kanssa.

Yrityksessä oli tänään melko rauhallista, joten hänellä oli aikaa tutkia yhdessä asiaa kanssani. Sen tiesimme jo, että WordPress tarjoaa melko hyvän suojan yleisimmiltä hyökkäyksiltä. Toisaalta, tahdomme olla varmoja siitä, että sivusto on varmasti suojattu – sen myötä voimme myös soveltaa oppimaamme seuraaviin, kenties suurempiin projekteihin. Ensimmäinen ajatukseni oli, jos /admin -alasivun voisi piilottaa jotenkin, esimerkiksi nimeämällä sen eri tavalla. Vanhempi kehittäjä kuitenkin kertoi sen olevan jo hieman myöhäistä, sillä se toimenpide olisi pitänyt tehdä heti alussa. Jatkossa pyrin tekemään näin, sillä tavalla voin estää ainakin hieman potentiaalisia hyökkäyksiä. Hän kertoi, ettei se kuitenkaan estä kovinkaan tehokkaasti esimerkiksi netissä pyörivien bottien hyökkäyksiltä, sillä ne löytävät helposti piilotetunkin sisäänkirjautumislomakkeen.

Selkeästi suurimmaksi turvariskiksi osoittautuivat liitännäiset. WordPress itsessään on rakennettu turvalliseksi, mutta liitännäisiä on tuhansia ja sen myötä myös useita eri taustan omaavia kehittäjiä. Liitännäisen kehittäjä saattaa olla aihepiiriin erikoistuva yritys tai vaikkapa itseni kaltainen opiskelija, joten kaikissa liitännäisissä ei välttämättä olla huomioitu mahdollisia tietoturvariskejä lainkaan. Pyrin olemaan aina varovainen liitännäistä asentaessa, sillä liian monen liitännäisen käyttö hidastaa myös sivuston toimintaa sekä tekee siitä sekavaa. Tästä syystä käytin loppupäivän käyttämieni liitännäisten tarkastelemiseen. Tämä on yleisesti ottaen muutenkin kannattavaa, sillä kehityksen eri vaiheissa tulee yrittäneeksi erilaisia ratkaisuja, joita ei välttämättä muista aina poistaa. Siispä ohjelmaan aina kertyy yleensä jotain ylimääräistä, jota ei tarvita lopullisessa tuotteessa.

Perjantai 27.10.2017

Saapuessani aamulla töihin keskustelin esimieheni kanssa, oliko hän ehtinyt tarkastelemaan painotalolle rakentamaani WordPress -sivua, mutta hän ei ollut viikon aikana ehtinyt tekemään niin. Sivusto on pääpiirteiltään nyt valmis, joten päätin keskittyä tänään rakentamaan jälleen ruokalistaa.

Sivustolla on jo käytössä tietokanta, joten vanhemman kehittäjän avulla kirjoitan ruokalistan käyttämään kyseistä tietokantaa. Kyseisessä tietokannassa on jo valmiiksi aterioita, joita asiakas voi luoda – sen jälkeen ruokalistalle on tarkoitus lisätä nämä aikaisemmin luodut ateriat. Käytännössä tulen myöhemmin luomaan tietokantaan weeklymenu -taulun, jossa käytän viiteavaimena luodun aterian nimeä toisesta taulusta. Muita tauluun syötettäviä tietoja tulee olemaan pääavaimena oleva id, jonka tietokanta generoi itse auto_incerement -asetuksella; päivämäärä, jolloin ruoka-annos halutaan tarjoilla; kommentti annoksesta; viiteavain annosryhmä, johon ruoka-annos tietokannassa kuuluu sekä viiteavain aterian hinta. Lopputuloksena ruokalistan tulisi näyttää esimerkiksi tältä:

Taulukko 4. Esimerkki ruokalistan näkymästä

Montag	**Dienstag**	**Mittwoch**	**Donnerstag**	**Freitag**
23.10.17	24.10.17	25.10.17	26.10.17	27.10.17
Aterian nimi Annostyyppi Hinta € Kommentti		**Maultaschen** Menu 1 5,50 € Traditionelles bayerisches Essen		**Currywurst** Menu 2 3,70 €

Ruokalista-projektissa olen saanut jo tiedon kulkeutumaan tietokannasta niin, että se tulostuu selaimen konsoliin. Muokkaan tätä koodia niin, että teen request -pyynnön lokaalin tietokannan sijasta projektissa käytettävästä kannasta. Muokkaukseen ei mene kauan aikaa, sillä muutan vain taulujen ja haettavien tietojen nimiä. Tähän tehtävään menee lopulta koko päivä, joten seuraava haasteeni projektin parissa tulee olemaan tiedon näyttäminen sivulla. Sen jälkeen tavoitteena on saada syötettyä jotain tänään luomaani weeklymenu -tauluun sekä saada luotua ohjelmaan keino käyttää date -formaattia.

Viikkoanalyysi

Viikon aikana en saanut yhteydenottoja asiakkailta tai toimeksiantoja esimieheltäni, joten päädyin työskentelemään toissijaisten tehtävien parissa. Tekemäni tehtävät olivat toki tärkeitä, mutta olin aina lykännyt niitä myöhemmäksi.

Painotalolle rakentamani WordPress -sivuston parissa työskentelin tietoturvaan liittyvien asioiden parissa. Kehitin edelleen lomakkeen kautta lähetettyjen roskaviestien estämistä rakentamalla turvakysymyksiä, joita kysytään käyttäjältä ennen kuin lomake lähetetään. Kysymykset ovat tarkoituksella helppoja, jottei käyttäjä ärsyynny niistä. Keith (2015) listaa kuusi erilaista tapaa estää Contact Form 7:n kautta lähetetyt viestit, joista kätevimmäksi hänen mielestään osoittautuu turvakysymykset. Tämä sen takia, koska liitännäisessä on sisäänrakennettu keino luoda kysymykset. Mikään ei tietenkään estä käyttämästä useaa eri menetelmää ja tätä hän myös suosittelee, sillä hän sai omalla sivustollaan estettyä roskaviestit kokonaan käytettyään useaa eri menetelmää samassa lomakkeessa.

Keithin (2015) esittämät kuusi menetelmää ovat:

1. Turvakysymys
2. Vähimmäiskirjainten määrittäminen lomakkeisiin
3. CAPTCHA
4. Contact Form 7 Honeypot -liitännäinen
5. Akismet -anti-spam liitännäinen
6. Bad Behaviour -liitännäinen

Toinen tietoturvaan liittyvä tehtävä oli tutkia erilaisia hyväksi koettuja toimintatapoja tietoturvan parantamiseksi sekä yleisimpiä uhkia, joita WordPress -sivustot kohtaavat. Wright (2017) selvitti, että tietoturvauhkista 52% liittyy liitännäisiin. Hän listaa kirjoituksessaan myös viisi yleisintä tietoturvauhkaa, joita WordPress -kehittäjän tulee huomioida. Yhden uhkista – ”brute force” -hyökkäyksen - mainitsin viikon päiväraportissani. Kyseisessä hyökkäyksessä pyritään kirjautumaan sisään kokeilemalla eri kirjautumistunnuksia lukuisia kertoja, kunnes oikea yhdistelmä kirjautumisnimestä ja salasanasta löytyy. Tämän kaltaisen hyökkäyksen toteuttaa usein botti, joka saattaa yrittää useamman päivän sisäänkirjautumista. Williams, Damstra ja Stern (2015, 371) neuvovat rajoittamaan sisäänkirjautumisyritysten määrää. He ovat kirjoittaneet kappaleen kirjassaan pelkästään tietoturvasta ja listaavat useita tapoja, joilla voi parantaa sivuston turvallisuutta.

Taulukko 5. Williamsin, Damstran ja Sternin (2015, 370-375) neuvot tietoturvan partamiseen.

1	Käytä uusinta WordPress -versiota	Sisällönhallintajärjestelmän kehittäjät tekevät jatkuvasti parannuksia tietoturvaan, joten uusimman

		version avulla saa parhaimman tietoturvan.
2	Piilota versionumero	Mahdolliset hyökkääjät saattavat käyttää hyödykseen vanhempaa järjestelmäversiota.
3	Älä käytä oletuskäyttönimeä	Admin -käyttäjänimi on ennalta-arvattava.
4	Rajoita sisäänkirjautumisyritysten määrää	Yritysten määrää rajoittamalla voidaan ehkäistä ”brute force” -hyökkäyksiä.
5	Käytä hyviä salasanoja	Monimutkaisen ja turvallisen salasanan arvaaminen on vaikeampaa.
6	Vaihda tietokantataulujen etuliite	Jos hyökkääjä pääsee lukemaan tietokantatauluja, niiden luku on vaikeampaa kuin vakioetuliitteiden kanssa.
7	Siirrä wp-config -tiedosto pois sivuston päähakemistosta	Estetään wp-config -tekstitiedoston näyttäminen sivulla mahdollisten virheiden tapahtuessa.
8	Siirrä wp-content -kansio muualle	Tekee hyökkääjien käyttämien automatisoitujen työkalujen toiminnan hankalammaksi
9	Käytä WordPressin turvaavainominaisuutta	Tekee hyökkääjien työstä vaikeampaa, sillä heidän täytyy selvittää useampi avain ennen kirjautumista.
10	Pakota SSL sisäänkirjautumiseen	Suositellaan vain erikoistapauksissa. Sisääkirjautuessa käytetään SSL -suojattua sivua.
11	Tarkasta tiedostojen oikeudet	Tiedostojen käyttöoikeudet on hyvä käydä läpi, jottei mahdollisen hyökkäyksen sattuessa kaikilla käyttäjillä ole tarpeettoman paljon oikeuksia.
12	Tarkasta käyttäjien MySQL -oikeudet	Tietokannan muokkaukseen ei tulisi antaa liian monelle käyttäjälle oikeuksia.
13	Tarkasta käyttäjien oikeudet WordPressissä	WordPressiin on rakennettu useita eri käyttäjärooleja, joilla on kaikilla omat oikeudet sivuston sisällön tarkasteluun ja muokkaamiseen. On siis hyödyllistä käydä ne läpi ja antaa sopivat oikeudet käyttäjille.

Ensi viikolla tulen jatkamaan enemmän ruokalistaprojektin kanssa, mikäli painotalo ei ota minuun yhteyttä sivuston suhteen. Tämän viikon tehtävät olivat tärkeydeltään toissijaisia, mutta ne oli hyvä tehdä nyt, kun minulla oli aikaa.

3.7 Seurantaviikko 44

Maanantai 30.10.2017

Painotalo oli ottanut viikon aikana yhteyttä ja toivonut muutoksia kontaktilomakkeeseen. He olivat esittäneet toiveen, että heidän asiakkaansa voisivat yhteydenoton lomassa ilmoittaa myös tarkemmin painotoiminnan aiheen sekä painettavien artikkeleiden määrän. Kävimme esimieheni kanssa toivottuja muutoksia läpi ja minulle kävi selväksi melko nopeasti, että lomakkeen ulkoasu tulisi suunnitella uudestaan. Asiakas oli toivonut tarpeeksi monta lisäystä, joten katsoin järkeväksi varata kontaktilomakkeelle oma rivinsä sivulta. Tähän asti se oli ollut samalla rivillä yrityksen yhteystietojen kanssa, eli vasemmalla puolella oli ollut yhteystiedot ja oikealla lomake. Puhuin esimieheni kanssa suunnitelmastani ja päätimme, että pidämme yhteystietojen layoutin ja laitamme sen kanssa samalle riville oikealle puolelle Google Maps -osion, jossa näkyy kartalla yrityksen sijainti.

Aloin suunnittelemaan uuden kontaktilomakkeen asettelua. Esimieheni oli lähettänyt tarkemmat tiedot uusista lomakkeen kentistä ja päätin, että painotyypin ja artikkeleiden määrän valinta olisi luonnollisinta dropdown -vetovalikon kautta. Lähetin ehdotukseni lomakkeen layoutista hänelle, jonka hän hyväksyi. Aloin rakentamaan lomaketta ja päätin, että se saattaisi olla kaikista helpointa Bootstrap -kehyksen kanssa. Teemassani ei ollut kuitenkaan tätä käytössä, joten asensin Bootstrap for Contact Form 7 -liitännäisen. Asennuksen jälkeen se kuitenkin täysin rikkoi koko sivuston tyylin, joten poistin sen. Päätin tämän jälkeen yrittää rakentaa lomakkeen itse CSS -tyylitiedostoa muokkaamalla. Contact Form 7 antaa kehittäjälle tarpeeksi vapauksia, jotta pystyin kustomoimaan lomaketta itse. Tavoitteeni oli yksinkertaisesti jakaa osa lomakkeen täyttökentistä vasemmalle ja oikealle puolelle.

Kuva 7. Uusi, omalla rivillään oleva kontaktilomake kehitysvaiheessa

Onnistuin tekemään toimivan ratkaisun, jossa määritin täytekentille leveydeksi 50%. Sen jälkeen käytin ”float” -sääntöä asettamaan ne vasemmalle ja oikealle puolelle. Otin huomioon myös responsiivisuuden: kun käyttäjän näyttö on 768 pikseliä tai vähemmän, täytekenttien leveydelle annetaan arvo 100%. Tällä tavalla täytekentät eivät ole liian pieniä esimerkiksi kännykän näytöllä. Contact Form 7:n avulla oli myös erittäin helppoa tehdä vetovalikko eikä siihen mennyt minulta kovinkaan kauan aikaa.

Päivän lopuksi esittelin uuden lomakkeen esimiehelleni, joka oli tyytyväinen lopputulokseen. Päätimme, että huomenna viimeistelen muutokset asettamalla yhteystietojen viereen Google Maps -näkymän yrityksen sijainnista ja testaamalla lomakkeen toimivuutta.

Tiistai 31.10.2017

Aloitin päiväni jatkamalla siitä mihin oli eilen jäänyt, eli asiakkaan toivomien muutoksien tekemistä. Päätin ensiksi testata uutta lomaketta, joten lähetin sen kautta yhteydenottopyynnön, joka on ohjattu vielä kehitysvaiheessa itselleni. Lomake saapui, kuten olin toivonutkin, mutta ilman uusien täyttökenttien tietoja. Huomasin myös osan sähköpostin tekstistä olevan vielä englanniksi, joten korjasin tämän. Korjattuani sähköpostiviestin kielen, päätin testata lomaketta tarkemmin virheilmoitusten osalta. Sen kautta ei tulisi saada lähetettyä viestiä ilman tiettyjen kenttien täyttämistä. Virheilmoitukset toimivat hyvin, mutta huomasin niiden olevan vielä englanniksi. En siis ollut vielä huomannut kääntää virheilmoituksia saksan kielelle, joten päätin tehdä tämän ennen lounaalle menoa.

Käytän käännöstyöhön DeepL -sivuston kääntäjää, jota toinen kehittäjä yrityksessä oli suositellut. Se käyttää koneoppimista käännösten parantamiseen, joka tekee käännöksistä ajan myötä edelleen tarkempia. Kääntäjän avulla kirjoitan kaikki mahdolliset virheviestit uudestaan saksaksi. Käännöstyön jälkeen lähden lounaalle vanhemman kehittäjän kanssa. Keskustelemme Google Maps -osion rakentamisesta sivulle ja hän varoittaa, että se saattaa lisätä etusivun latausaikaa huomattavasti.

Päätän silti yrittää kartan implementoimista sivustolle. Toinen vaihtoehto voisi olla jonkin muun kartta-applikaation käyttöä tai vain kuvan lisäämistä. WordPressistä löytyy liitännäisiä myös Google Mapsin implementointiin. Päätän kokeilla WP Google Maps -liitännäistä, joka on suosituin. Kartan asentaminen ja sijainnin laittaminen on helppoa, jonka jälkeen asetan kartan haluttuun paikkaan sivun muokkausikkunassa. Muokattuani loppupäivän karttaa ja yhteystietojen sijaintia saan tehtyä rivistä haluamani näköisen.

Perjantai 3.11.2017

Aamulla töihin tultuani keskustelin esimieheni kanssa viikon aika rakentamastani uudesta kontaktilomakkeesta. Hän kertoi myös asiakkaamme katsoneen ratkaisua ja pitäneen sitä hyvänä. Samoin kuin viime viikolla, päätän jatkaa ruokalan menulistan kehittämistä. Minulle näyttää muodostuvan työtapa, jossa maanantaisin ja tiistaisin keskityn painotalon sivustoon ja perjantaisin menun rakentamiseen. Tämä on mielestäni vain hyvä asia, sillä on

vaikeaa vaihtaa erilaiseen työtehtävään keskellä työpäivää. Nyt pystyn keskittymään täysin yhteen työtehtävään päivän aikana.

Viimeksi projektin parissa työskennellessäni sain toisen kehittäjän avulla siirrettyä työni lokaalista kehitysympäristöstä yhteiseen, Git -versionhallintatyökalun kautta jaettuun ympäristöön. Työpäiväni alussa kirjaudun sisään ympäristöön ja haen muutokset Gitistä git pull -komennolla. Jaan työympäristön kahden muun kehittäjän kanssa ja jokainen meistä työskentelee eri tehtävän parissa, joten versiokonflikteja ei tapahdu.

Keskustelen seuraavista työvaiheista projektia vetävän vanhemman kehittäjän kanssa. Hän esittää näkemyksen, että menuikkunoita tulisi olla kaksi: yksi tavalliselle käyttäjälle ja yksi sisäänkirjautuneelle henkilökunnan jäsenelle, joka voi muokata listaa. Alussa voimme rakentaa ruokalistaa vain yhdellä sivulla niin, että sitä voidaan muokata. Vasta myöhemmin rakennan toisen näkymän asiakkaalle, joka on käytännössä sama ikkuna ilman muokkausmahdollisuuksia.
Päätän aloittaa lisäysominaisuuksien kehityksen, sillä haluan saada syötettyä jotain tietokantaan. Olin jo aiemmin tehnyt sivulle modal -ponnahdusikkunan, jonka kautta ruokaannoksia voidaan lisätä tietokantaan. Huomaan tehneeni jo syöttökentät lomakkeeseen, joten sen päälle on hyvä kehittää toiminnallisuutta.

Angularissa voin määrittää syöttökentän arvon antamalla sille ng-model -arvon:

<input type="text" class="input-group date btn-block" maxlength="40" placeholder="Alternative beschreibung" ng-model="weeklymenu.description"></input>.

Tämä ilmoittaa lomaketta syötettäessä, että tällä syöttökentällä annettava arvo on weeklymenu -objektissa description -muuttuja. Tämä tieto siirtyy kontrolleriin, jossa tieto käsitellään ja lähetetään routeriin. Router kerää kaikki tiedot, mitä haluamme lähettää tietokantaan ja lähettää ne yhtenä objektina model -tasolle, jossa ne syötetään SQL -lauseeseen ja lähetetään tietokantaan. Käytännössä siis samaa reittiä pitkin, kuin tiedon saapuessa käyttöliittymään.

Olen jo aikaisemmissa projekteissa rakentanut MVC -mallin mukaisen tiedonkulkurakenteen. Saan päivän aikana tehtyä lomakkeen, jonka kautta saan syötettyä tietokantaan weeklymenu -tauluun tietoa. Myös tietokanta näyttää toimivan tässä vaiheessa tarpeeksi hyvin, joten olen iloinen päivän ja viikon saavutuksiin.

Viikkoanalyysi

Viikkooni kuului sekä ruokalistaprojektin että painotalolle rakentamani WordPress -sivuston parissa työskentelyä. Olen mielestäni pystynyt kuitenkin paremmin järjestämään aikatauluni niin, että järjestän jokaiselle tehtävälle yhden päivän. Tällä tavalla kykenen paremmin keskittymään yhteen projektiin ja tiettyihin tehtäviin päivän aikana. Seuranta-ajan alussa työpäiväni olivat haastavimpia, sillä saatoin käyttää päivän työaikaa täysin erilaisiin tehtäviin. Tällä viikolla käytin kaksi ensimmäistä työpäivääni painotalon sivuston muokkaamiseen ja viimeisen päivän ruokalistan kehittämiseen.

Painotalon sivustossa tein muutoksia kontaktilomakkeeseen asiakkaan pyynnöstä. Käytän kontaktilomakkeen luontiin suosittua Contact Form 7 -liitännäistä. Olen havainnut sen olevan tähän mennessä käytännöllisin tapa luoda lomakkeita WordPressissä. Kenttien luominen on tehty yksinkertaiseksi ja viestien lähetysasetusten muokkaaminen on tehty mahdollisimman helpoksi. Liitännäinen mainostaa itseään lauseella ”yksinkertainen, mutta joustava” (Weller 2015). Contact Form 7 antaa mahdollisuuden muokata lomakkeen tyylisääntöjä editorilla, johon voi kirjoittaa CSS -koodia. Toisin kuin kilpailevat lomakeliitännäiset, se ei kuitenkaan tarjoa helppoa tapaa muokata lomakkeen ulkoasua. (Weller 2015)

Tavoitteeni oli luoda lomakkeeseen syötekenttiä niin, että yhdellä rivillä on kaksi vierekkäin. Sain tehtyä tämän niin, että asetin lomakkeille säännöksi:

```
width: 50%;
/* jos kenttä halutaan oikealle */
float: right;
/* tai  jos se halutaan vasemmalle */
float: left;.
```

Tällä tavalla sain käytettyä div -elementin koko leveyttä, jonka sisällä lomakkeen kentät sijaitsivat. (Computer Hope 2017.)
Käyttäjän näyttöruudun ollessa 768 pikseliä tai vähemmän, syöttökentät vievät kukin oman rivinsä leveyden. Yllä olevaa CSS -koodia muokataan siis niin, että float -ominaisuudet poistetaan ja width -ominaisuuden arvo vaihdetaan arvosta 50% arvoon 100%. Tällä tavalla käyttäjän näkymä on pienemmällä näytöllä selkeämpi ja käyttäjäkokemus on positiivisempi.

Lomakkeen muutostarpeen aiheuttamana jouduin suunnittelemaan yhteydenottotietojen sijaintia uudelleen. Lomake ja yrityksen tiedot olivat aiemmin olleet samalla rivillä, mutta

nyt lomakkeen koko vei oman rivinsä tilaa sivusta. Päätin siis laittaa yrityksen yhteystietojen viereen interaktiivisen kartan, jossa näkyy painotalon sijainti Stuttgartissa.
Kartan päätin toteuttaa Googlen kehittämän Google Maps -karttapalvelun avulla. Karttapalvelun voi tehdä sivustolle monella erilaisella tavalla. Yksinkertaisin tavoista on syöttää sivuston koodiin iframe -elementti, joka saadaan Google Maps -sivustolta. Sivustossa tulee etsiä haluttu paikka kartalta ja kopioida sivusto luoma elementti. Tämän jälkeen se voidaan syöttää omalle sivulle ja kartta tulee näkyviin. (Wpbeginner 2018.)

En halunnut kuitenkaan rakentaa karttaa iframe -elementin avulla. Djuraskovic (2017) kertoi blogissaan, että iframe -elementin avulla saa tehtyä kartan helposti, mutta se ei ole sivuston lautausajan kannalta nopein. Hän kertoo syyksi sen, että iframe alkaa latautumaan välittömästi käyttäjän saapuessa sivulle, vaikkei käyttäjä edes tarkastelisi karttaa. Tämä aiheuttaa sivun latausajan kasvun, joten ratkaisu ei tältä osin ollut projektissani toimiva.
Tämän takia päätin tehdä karta WP Google Maps -liitännäisen avulla, joka ei käytä iframe -elementtejä (WordPress). Liitännäinen toimi hyvin, oli helppo asentaa eikä vaikuttanut sivuston latausaikaan merkittävällä tavalla.

Viikon aikana edistyin myös ruokalistaprojektissa. Saimme siirrettyä tähän asti lokaalisti kehittämäni projektin emoprojektiimme, johon on koottu kaikki toiminnallisuudet, joita olemme tehneet. Käytämme versionhallintaan Gittiä, joka on yksi suosituimmista versionhallintajärjestelmistä. Gitin etuja ovat hyvä suorituskyky ja joustavuus (Quickscrum 2017). Yrityksessä on tapana ladata päivän alussa uusin versio Gitistä ja ladata oma, toimiva versio, kun on saanut kehitettyä ja testattua toimivan ominaisuuden. Tällä tavalla emme kohtaa versiokonflikteja. Konfliktien sattuessa pystymme kuitenkin aina lataamaan jostain viimeisimmän toimivan version. Käytämme myös kommentteja aina omaa versiota ladatessa, jotta tiedämme mitä muutoksia on tehty siihen versioon.

3.8 Seurantaviikko 45

Maanantai 6.11.2017

Viikonlopun aikana en ole saanut uutta toimeksiantoa tai uusia tehtäviä, joten päätin jatkaa viikkomenun kehitystä. Viime viikolla sain syötettyä tietokantaan testilomakkeelta tietoa, joten projektin perusasiat ovat tällä hetkellä toiminnallisia. Seuraava tehtäväni on saada luotua sivustolle tapa siirtyä toiselle viikolle sivun näkymässä, jotta käyttäjät voivat nähdä myös seuraavien viikkojen ruokalistan.

Alan etsimään tapaa kehittää aikaan perustuvaa listaa ja haluan löytää keinon, jolla saan nykyisen päivän näkyviin. Sen jälkeen täytyy löytää sen viikon ensimmäinen päivä eli maanantai, lisätä se listaan ja lisätä listaan vielä seuraavat neljä päivää, eli perjantaihin asti. Jos saan oikeat päivämäärät näkyviin, voin tehdä yksinkertaisen laskutoimenpiteen joka tapahtuu napinpainalluksen yhteydessä ja se siirtää käyttäjän seuraavan tai edellisen viikon näkymään. Päätin etsiä, jos olisi olemassa valmis moduuli, joka käsittelisi päivämääriä jo valmiiksi. Löysin lopulta yhden node.js -moduulin nimeltään moment,js, joka vaikuttaa olevan myös melko suosittu. Sen avulla saan erilaisia työkaluja ajan muokkaamiseen, eikä minun tarvitse rakentaa kaikkea alusta asti itse.

Momentin avulla saan esimerkiksi nykyisen päivämäärän komennolla

var currentDate = moment().locale('de');.

Momentin avulla voin myös formatoida päivämäärän muotoa sekä toivottavasti myös saada viikonpäivät näkymään. Tämän jälkeen pitäisi olla helpompaa myös selvittää, miten saan ladattua seuraavan ja edellisen viikon päivämäärät.

Haluan tässä vaiheessa vain saada listan viikonpäivistä näkymään sivulla ja myöhemmin yritän laittaa ne niille kuuluvien viikonpäivien alle. Päivän aika löydänkin momentista keinon muuttaa aikamäärettä viikon alkuun, komennolla

moment().startOf('week');.

Löysin momentin dokumentaatiosta keinon lisätä listaan viikon kaikki päivät ja yritän implementoida sen tämän viikon aikana.

Tiistai 7.11.2017

Aloitan työpäiväni perehtymällä viikonpäivien listaamiseen. Tämä vaikuttaa tehtävältä, joka pitää saada valmiiksi ennen kuin voin jatkaa projektin muiden osa-alueiden parissa. Tästä syystä asetan päivän prioriteetiksi selvittää, miten saisin tämän toteutettua. Olen tarkastellut momentin dokumentaatiota, jos sieltä löytyisi jokin keino tuoda kaikki työviikonpäivät listamuodossa.

Löydän kaksi keinoa listata päivät, joista yksi on käyttää momentin weekdays -metodia. Hetken ajan yritettyäni en saa sitä kuitenkaan toimimaan, joten päätän perehtyä toiseen keinoon. Siinä tehdään funktio, johon syötetään nykyinen päivämäärä, selvitetään mikä on

sen päivän viikon ensimmäinen päivä (komennolla, jonka esitin viime työpäiväni raportissa) ja ajetaan for -silmukka lisäten aina yksi päivä lisää. Päätän rakentaa sen.
Saan lopulta tehtyä silmukan onnistuneesti ja näytettyä sen sivulla Angularin ng-repeat -ehdolla. Itse silmukka ja siihen liittyvät muut variaabelit näyttävät lopulta tältä:

```
$scope.weeklyitems = [];
$scope.startMenu = moment().format("YYYYMMDD");
var currentDate = moment().locale('de');
$scope.workweekdays = ['Montag','Dienstag','Mittwoch','Donnerstag','Freitag'];
var fnWeekDays = function(dt) {
  var weekStart = dt.clone().startOf('week');
  var weekEnd = dt.clone().endOf('week');
  var days = [];
  for (i = 0; i <= 6; i++) {
    days.push(moment(weekStart).add(i, 'days').format("YYYYMMDD"));
  };
  return days;
}
$scope.weekDays = fnWeekDays(currentDate);
```

Käytän siis momenttia selvittämään nykyisen päivämäärän sekä muokkaamaan näytettävän ajan formaattia. Silmukassa pusken days -listaan päivämäärät ja lähetän sen edelleen. Olen myös tehnyt listan päivien nimistä itse, sillä ne eivät tule muuttumaan, mutta senkin olisi voinut tehdä momentin avulla.

Lounastauon jälkeen laitan päivät näkymään käyttöliittymän tasolla. Käytin jo yksinkertaista ng-repeat -ehtoa näyttämään viikon kaikki päivät ja nyt kehitän sitä edelleen. En halua näyttää viikonlopun päivämääriä ja haluan näyttää yhden päivämäärän yhden viikonpäivän kohdalla. Saan ajettua päivät

```
<div ng-repeat="day in workweekdays" class="col-sm-2 nopadding">
 <h4 class="text-center">{{day}}</h4>
 <div class="text-center">{{weekDays[$index] | amParse:'YYYYMMDD' | amDateFormat:
'DD.MM.YY'}}</div></div>
```

komennolla. Ng-repeat käy läpi viikonpäivät, joita olen määitellyt olevan viisi. Sen jälkeen se lisää aina workweekdays -listasta seuraavan muuttujan ja lisää myös juuri tekemästäni for -silmukasta päivämäärän. Silmukassa weekDays -listaan on aina lisätty viikon ensim-

mäinen päivä ensiksi, joten se lisätään sivulle ensimmäisenä. Muutan myös päivämäärän formaatin luettavampaan muotoon. Jokaisessa div -elementissä olen myös antanut Bootstrapin col-sm-2 luokan, joka järjestää listan haluamaani muotoon.

Viikkoanalyysi

Viikon tehtäväksi muodostui ruokalan ruokalistan kehittämisen jatkaminen. Päätin rakentaa keinon käyttää aikaa ohjelmassa, sillä se on yksi keskeisimmistä ohjelman toiminnoista. Ohjelmassa tulee pystyä muun muassa liikkumaan viikkojen välillä ja määrittämään aika annokselle. Ratkaisun löysin node.js:n tarjoamasta moment -moduulista, jonka avulla voi jäsentää, valitoida, manipuloida ja näyttää päivämääriä ja aikoja Javascriptinä (Moment.js). Node.js on hyödyllinen työkalu ja momentin asennus oli helppoa npm-paketinhallintajärjestelmän avulla. Asentaminen tapahtuu npm-ikkunassa komennolla *npm install moment*, jos haluaa asentaa momentin. (Dayley 2014, 43.) Moduuli osoittautui hyödylliseksi työkaluksi ajan käyttämiseen ohjelmassa, sillä sen avulla pystyin tekemään haluamani tehtävät. Ensimmäinen tehtävä, jonka halusin moduulin toteuttavan, oli nykyisen päivän näyttäminen. Tämä toteutui peruskomennolla *moment()*. Ajan pystyi myös formatoimaan Saksan aikaformaattiin komennolla *locale('de')*.

Momentin käyttäminen oli helppoa myös siksi, että moduulin tekijät olivat tuottaneet kattavan dokumentaation erilaisista käyttötavoista. Dokumetaatiossa esiteltiin useita esimerkkejä, joista pystyin löytämään mainitsemani komentoni. Löysin myös weekdays -komennon, joka ei kuitenkaan toiminut. Lopulta sain rakennettua esittelemäni for-silmukan, joka käy läpi viikonpäivät ja lisää ne listaan. Käytin momentin komentoa *startOf('week')*, jonka avulla sain viikon ensimmäisen päivän ja sain ajettua silmukan tästä päivästä aloittaen.

Päivät sain tulostettua käyttönäkymään Angularin komennolla ngRepeat. NgRepeat -direktiivi käy läpi listan ja toteuttaa HTML-elementin sisällä sille annetun komentorivin jokaisen listalla olevan esineen kohdalla (AngularJS). NgRepeat on yksi niistä useista direktiiveista, jotka tekevät Angularista suositun työkalun webohjelmoijien keskuudessa. Sen avulla pystyy käyttämään Angularin HTML-elementteja, jotka tuottavat valmiiksi halutun toiminnallisuuden. Angularin käyttäminen on helppoa ja olen aina yllättynyt, kuinka helppoa kehitystyö sen kanssa on.

Tämän viikon analyysi jää hieman lyhyemmäksi, sillä minulla oli viikon aikana vain kaksi työpäivää. Toivottavasti ensi viikolla pystyn silti jatkamaan tämän viikon saavutusteni pohjalta, sillä vaikka tämän viikon raportti jäikin hieman tavallista lyhyemmäksi, sain mielestä-

ni aikaan melko hyvää kehitystä. Olin projektin alkaessa hieman huolestunut, jos löytäisin hyvän tavan ajan muokkaamiseen, sillä se täytyy viedä ja hakea tietokannasta oikeassa muodossa. Tietokannan käyttämä aikamuoto olisi käyttäjän näkökannalta oudon näköinen, joten minun olisi täytynyt rakentaa keino formatoida ajan näkymä aina tietokannan ja käyttäjän välillä. Momentin avulla tämä työvaihe muuttui helpommaksi, joten ennakkoodotuksista huolimatta se ei vienyt lopulta kovinkaan kauan kehitysaikaa. Ruokalistaprojektia on helpompaa jatkaa tästä.

3.9 Seurantaviikko 46

Tiistai 14.11.2017

Olin viime työviikon lopussa sekä eilen maanantaina sairaslomalla, joten kyseisiltä päiviltä ei ole tämän takia raporttia. Töihin saapuessani menen keskustelemaan esimieheni kanssa, jos minulle olisi uusia toimeksiantoja tai joku asiakkaistamme olisi ottanut yhteyttä. Viikon aikana poissa ollessani ei ollut kuitenkaan tapahtunut mitään itseäni koskevaa, joten päätin jatkaa viikkomenun parissa.

Olen viime viikolla onnistunut edistymään huomattavasti. Sain momentin avulla luotua keinon käyttää päivämääriä, joka on yksi tärkeimmistä toiminnoista projektin etenemisen kannalta. Käyttäjän tulee pystyä ruoka-annoksia menuun lisätessään määrittelemään päivämäärän. Viikkojen välillä tulee pystyä myös liikkumaan, jotta käyttäjät näkevät myös seuraavien viikkojen listan. Käytännössä tämä tulee tarkoittamaan sitä, että minun pitää kehittää kaksi nappia, jotka päivittävät listaa viikonpäivien osalta niitä painettaessa. Ruoka-annoksen lisäyksessä annettavan päivämäärän tulee olla luonnollisen tuntuinen käyttäjälle, joten ennen sen tekemistä tulen kysymään vielä vanhemmalta kehittäjältä eri tavoista kehittää se.

Päätän alkaa kehittämään tapaa siirtyä käyttöliittymässä viikkojen välillä. Voin kenties käyttää ratkaisun koodia myös muissa osioissa. Nappia painaessa tulee tapahtua kaksi asiaa: haetaan halutun viikon päivämäärät sekä päivitetään näkymä. Entisistä Angularin kanssa tekemistäni tunsin jo $watch -metodin, jota voisin käyttää näkymän päivittämiseen. Tätä metodia tulisin tarvitsemaan ehkä myöhemminkin, sillä ruoka-annoksen lisäämisen yhteydessä listan tulee päivittyä myös.

Muotoilen sivuston ulkoasua niin, että napit mahtuvat omalle paikalleen. Tämän jälkeen alan tutkimaan momentin dokumentaatiosta erilaisia keinoja manipuloida päivämääriä. Löydän add- ja subtract-metodit, joiden avulla voi lisätä ja vähentää aikaa. Yritän aluksi

dokumentaatiossa annettujen esimerkkien avulla kehittää toimivaa ratkaisua, mutta se osoittautuu hankalaksi. Päätän etsiä netistä samankaltaisia ratkaisuja, jotka käyttävät moment -moduulia. Löydän yhden, joka voisi auttaa minua kehittämään oma ratkaisuni.

Voin käyttää aiemmin luomaani funktiota, johon syötän napinpainalluksen yhteydessä add- tai subtract-metodilla uuden päivämäärän. Tekemäni $watch sen jälkeen huomaa muutoksen ja ajaa oman ohjelman, jossa päivitän käyttönäkymän. Lopputuloksena saan muutettua listassa olevia päivämääriä sekä päivitettyä näkymän onnistuneesti.

Perjantai 17.11.2017

Viikon aikana Wordpressiin on tullut uusi laaja päivitys, joka päivittää Wordpressin versioon 4.9. Wordpressiin tulee säännöllisesti päivityksiä, joita tarkkailen töissä ollessani. Yleensä päivitykset ovat pieniä, esimerkiksi versiosta 4.83 versioon 4.85. Nämä päivitykset eivät aiheuta kovinkaan suurta vaivaa ja ne voi jopa määritellä latautumaan automaattisesti. Tällä kertaa kyseessä on suurempi päivitys, joka vaatii enemmän huomiota.

Wordpressin päivittäminen on tärkeä tehtävä, sillä sen mukana tulee muun muassa tietoturvapäivityksiä, joissa saattaa olla tärkeitä uudistuksia uusia uhkia kohtaan. Päivittämättä jättäminen on siis suuri turvallisuusuhka. Olen tähän asti luottanut Wordpressin turvallisuuteen ja oma osuuteni tietoturvaan on ollut enimmäkseen varoa asentamasta liian montaa liitännäistä.

Tästä syystä päätän asentaa päivityksen kaikkiin Wordpress -sivuihini mahdollisimman nopeasti. Asennan sen ensiksi kehityksen alla olevaan painotalon sivustoon. Päivittäminen osoittautuu helpoksi, kuten se yleensä on WordPressissä. Tämän jälkeen alan tarkastelemaan sivustoa ja onko mitään näkyviä muutoksia tapahtunut käyttöliittymässä. Yleensä suuremman päivityksen jälkeen testaan useimpien liitännäisten ja muiden osioiden toimivuuden ja päivitän myös ne ajan tasalle.

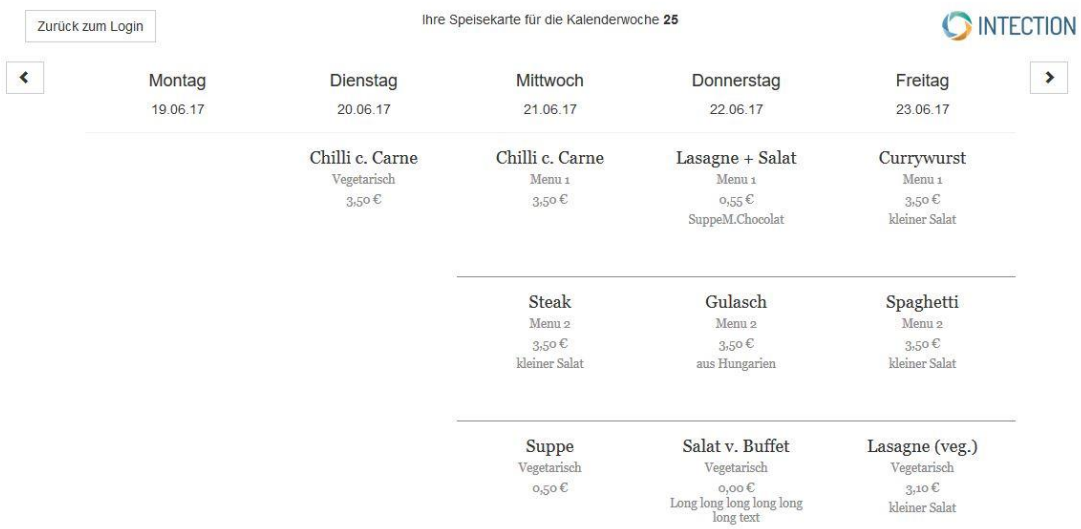
Jossain vaiheessa huomaan, että sivua muokatessa tietojen tallentaminen ei onnistu. Selviteltyäni asiaa epäilen sen johtuvan Visual Composer -liitännäisestä, joka on yksi keskeisimmistä käytössä olevista liitännäisistäni. Mikäli se ei toimi, sivustoa on erittäin vaikeaa muokata. Sen korjaaminen on siis tärkein prioriteettini. Aloin selvittämään, johtuiko virhe liitännäisestä, päivityksestä vai jostain muusta. Pian selvisi, että vanha versio Visual Composerista ei toiminut optimaalisesti uuden Wordpressin version kanssa.

Ongelma oli sinänsä hieman erikoinen, että Visual Composerista oli olemassa uusi, uuden Wordpress -version kanssa yhteensopiva päivitys. En pystynyt kuitenkaan lataamaan sitä, sillä teeman luoja – jonka kautta olin ladannut myös Visual Composerin – ei ollut tehnyt päivitystä teemaansa, jonka kautta saisi liitännäisen uusimman version. Tämä tarkoitti sitä, että ongelman korjaamiseen oli vain kaksi vaihtoehtoa: odottaa teeman päivittymistä tai siirtyä takaisin aiempaan Wordpressin versioon, jossa liitännäinen vielä toimi.
Koska päätös oli kohtalaisen tärkeä projektin kannalta, keskustelin esimieheni kanssa ongelmasta. Hän otti esille sen näkökulman, että asiakkaan tulee pystyä tekemään muutoksia sivulla. Emme voi odottaa, että kolmas osapuoli jossain vaiheessa julkaisee päivityksen – varsinkin, kun emme yhtään tiedä milloin se tapahtuu.

Päätämme, että päivitän Wordpressin takaisin aiempaan versioon. Tämä toimenpide onkin hieman hankalampi, kuin uudempaan päivittäminen. Se pitää sisällään tiedostojen poistamista ja vanhemman version ydintiedostojen siirtämistä niiden tilalle. Käytän FileZilla -tiedonsiirto-ohjelmaa tiedostojen siirtämiseen. Tämän jälkeen kirjaudun takaisin Wordpress sivulle ja päivitän tietokannan. Helpotukseksekseni vanhan Wordpress -version käyttöönotto onnistuu ilman virheitä, eikä minun tarvitse jäädä ylitöihin.

Viikkoanalyysi

Myös tämän viikon raportti on hieman lyhyempi vain kahden työpäivän takia. Viikon työtehtävät liittyivät sekä painotalon sivustoon sekä ruokalistaprojektiin. Viikon alussa olin suunnitellut kehittäväni ruokalistaa edelleen viime viikolla lataamani moment -moduulin avulla. Tavoitteeni viikolle oli saada lisättyä uusia ominaisuuksia, kuten näppäimet, joita painamalla käyttäjä voi siirtyä eri viikkojen näkymään. Alla olevassa kuvassa näkyy sivustolle rakentamani ulkoasu, jossa näkyvät ateriat olen kirjoittanut malliksi sekä mainitsemani napit kuvan reunoilla. Kuvasta näkyy myös, kuinka olen suunnitellut aterioiden järjestäytymisen listalla.



Kuva 8. Rautalankamalli lopullisesta asiakkaan käyttönäkymästä ruokalistaprojektissa. Viikkojen vaihtamiseen käytettävät napit oikealla ja vasemmalla puolella. Sisäänkirjautuneella käyttäjällä tulee näkymään myös annosten vieressä poisto- ja muokkausnapit sekä ylhäällä otsikon vieressä annoksen lisäysnappi, joita painamalla aukeaa modaali-ikkuna

Hyödynnän rautakisussani $watch -metodia, joka on yksi $rootScope -parametrissa olevista metodeista (AngularJS). Sen avulla pyrin päivittämään sivuston näkymän aina muutosten tapahtuessa, eli esimerkiksi viikkojen välillä siirtymisessä. Toinen ruokalistassa käyttämäni toiminnallisuus on add- ja subtract-metodit, joiden avulla saan muutettua päivämäärää, josta luon listan viikonpäivistä (Moment.js).

Suunnitelmani muuttuivat perjantaina, sillä WordPress oli julkaissut uuden päivityksen. Päivitys oli tällä kerralla hieman suurempi, sillä siinä siirryttiin versioon 4.9. Päivityksen jälkeen huomasin sivuston toimivan virheellisellä tavalla. Selvitettyäni asiaa huomasin WordPressin uuden version rikkovan Visual Composer -liitännäisen toimintaa, joka vaikutti olevan melko yleinen ongelma (Qode Themes -keskustelupalsta). Kyseisestä liitännäisestä oli uudempi versio, joka oli yhteensopiva WordPressin uuden version kanssa, mutta käytössäni olevan teeman luoja ei ollut päivittänyt teemaa. Ainut keino saada liitännäisen uusin versio on ladata se teeman päivityksen yhteydessä. Teeman luoja ei ollut ilmoittanut uuden version julkaisusta, joten tilanne oli osaltani hankala. Sivuston toiminnasta ei voida tinkiä, joten ainut vaihtoehto oli siirtää WordPress käyttämään vanhaa versiota, jossa Visual Composer vielä toimi. Sain tehtyä päivityksen vanhempaan versioon Dominiquen (2015) blogissa olevien neuvojen avulla. Päivittäminen onnistui mielestäni hyvin, mutta tuntui melko turhalta työltä selvittää tämänkaltaisia virheitä. WordPress päivittyy yleensä hyvästä syystä ja on tietoturvan kannalta hyvä käyttää aina uusinta versiota. Nyt joudum-

me odottamaan toisten kehittämiä muutoksia, jotta sivuston tietoturva olisi paras mahdollinen.

Ensi viikon tulevista tehtävistä on vaikeaa päätellä, mitä ne ovat. Ajan salliessa aion jatkaa ruokalistaprojektia, mutta esimiehen kanssa keskustellessani tämän viikon viimeisenä työpäivänä vaikutti siltä, että ensi viikolla pääsen työskentelemään erilaisissa tehtävissä. Hän vaikutti myös melko tuohtuneelta WordPressin tuottamasta ongelmasta, sillä työaikaa kului näin erikoisen virheen korjaamiseen.

3.10 Seurantaviikko 47

Maanantai 20.11.2017

Esimieheni oli keskustellut asiakkaamme kanssa toiseen palvelimeen siirtymisestä. Kyseessä on sivusto, jonka olin tehnyt ennen päiväkirjaraportoinnin aloittamista. Hän antoi minulle viikon tärkeimmäksi tehtäväksi sivuston siirtämisen uudelle palvelimelle. Sivusto on tehty myös Wordpressin avulla, joten tehtävä tulisi olemaan hieman monimutkaisempi kuin esimerkiksi Angularilla tehdyn sivuston siirtäminen.

Siirron tulisi tapahtua mahdollisimman nopeasti ja vaivattomasti, jottei yrityksen sivusto ole alhaalla kovin kauaa. Siirron aikana ei saisi myöskään kadota tietoja, joten ennen siirron aloittamista teen backup -version sivusta. Olen useasti aiemmin siirtänyt Wordpress -projektin lokaalista kehitysympäristöstä nettiin, mutten ole aiemmin siirtänyt palvelimesta toiseen. Mahdollisimman helpon siirron mahdollistamiseksi etsin ensin neuvoja, miten siirto käytännössä kannattaisi toteuttaa. Haluan välttää mahdolliset virheet ja tehdä siirron kerralla oikein. Vietän tähän toimenpiteeseen aikaa lounastaukoon asti.

Tauon jälkeen päätän aloittaa siirtoprosessin yhden löytämäni ohjelistaa noudattaen. Ensimmäinen askel siirtämisessä on asentaa Duplicator -liitännäinen, joka on luotu juuri tähän tarkoitukseen. Liitännäinen on saanut hyviä arvosanoja ja näyttää olevan luotettava, joten päätän käyttää sitä. Se luo paketin sivustosta, joka helpottaa asennusta toisella palvelimella.
Tämän jälkeen minun tulee ladata liitännäisen luomat paketit uudelle palvelimelle FTP -tiedonsiirto-ohjelman avulla. Tällä kerralla päätän käyttää Cyberduck -ohjelmaa, jota toinen kehittäjä oli minulle suositellut. Palvelimella minun tulee laittaa tiedostot root -hakemistoon. Tämän jälkeen täytyy luoda tietokanta uudella palvelimella, jota Wordpress voi käyttää. Tietokantaan täytyy myös sallia käyttöoikeudet sivustolle.

Näiden askeleiden jälkeen pystyn aloittamaan asennusprosessin. Duplicator -liitännäinen loi aikaisemmin kaksi tiedostoa, jotka siirsin uudelle palvelimelle: zip -tiedoston, jossa on sivuston kaikki tiedostot sekä installer.php -tiedoston, jota käytän asentamisessa. Aloitan asennusohjelman ja noudatan sen antamia ohjeita. Asennusohjelmaan tulee syöttää uuden palvelimen käyttämän tietokannan käyttäjätunnukset, jonka jälkeen asennus on valmis. Kokeilen uudessa palvelimessa olevaa sivustoa ja se toimii juuri niin kuin pitääkin. Ilmoita päivän lopuksi vanhemmalle kehittäjälle sivuston tilan, sillä hän oli lupautunut tekemään viimeiset vaiheet, kuten domain -nimen vaihdon sekä muut toimenpiteet, jotka vaikuttavat esimerkiksi sivuston näkymiseen Google -hauissa. Yritys käyttää myös mainontaa Google -hauissa, joten vanhempi kehittäjä halusi varmistaa, että nämä toiminnot eivät kärsi migraation takia.

Tiistai 21.11.2017

Testaan päivän aluksi vielä uudestaan eilen siirtämäni sivuston toimintaa. Haluan varmistaa, ettei asiakkaamme kohtaa sivustoa muokatessaan mitään virheitä. Testaan blogin muokkaamista, sivujen elementtien lisäämistä ja tallentamista sekä yhteydenottolomakkeen toimintaa. Testieni perusteella sivuston kaikki toiminnot ovat edelleen kunnossa, joten voin jatkaa muissa tehtävissä.

Otan tehtäväkseni jatkaa viikkomenun kehittämistä. Viime kerralla sain siis luotua viikkojen vaihtamisen näkymässä. Seuraava tehtäväni on viimeistellä aterian lisäyslomake. Tällä hetkellä lomake tulee esille modal -ponnahdusikkunan kautta nappia painettaessa. Tavoitteenani on, että saisin päivän aikana aterian lisäyksen toimimaan sekä kehittää käytännöllinen keino päivämäärän syöttämiseen. Päivämäärän lisäyksessä tulee ottaa huomioon, että mahdollinen käyttäjä saattaa haluta syöttää aterian tietokantaan pidemmän ajan päähän tai vaikkapa seuraavalle päivälle. Päätin keskustella toisen kehittäjän kanssa asiasta ja hän ehdotti, että voisi olla kaksi tapaa lisätä aterialle päivämäärä: kalenterin muodossa sekä klikkaamalla viikonpäivää. Lisäyksessä voisi olla pienemmässä mittakaavassa samanlainen aterioiden näyttökenttä, kuin päänäkymässä. Viikkojen välillä voisi liikkua samalla tavalla ja tiettyä päivää klikkaamalla saisi sen päivän valittua päivämäärän syöttökenttään.

Olen aikaisemmin luonut toimivan tiedonkulun käyttöliittymästä aina MVC -mallin model -tasolle saakka. Nyt tehtäväni on vain muokata sitä niin, että saan syötettyä halutut tiedot. Kuten aikaisemmassa viikkoraportissa olen maininnut, weeklymenu -taulussa käytetään useaa viiteavainta. Aterian lisäyksessä tarvitaan kuitenkin vain kolme täyttökenttää: ruokaannos, joka saadaan toisesta taulusta; vaihtoehtoinen kommentti ja aterian päivämäärä.

Aterian valitsemiseen tarvitsen vetovalikon, jossa näkyy kaikki valittavat ateriat. Kommenttiin riittää tavallinen tekstisyöte. Päätän kehittää vetovalikon ensiksi. Minun tulee hakea valikkoon toisen taulun artikkelit. Olen aiemmassa projektissa tehnyt samanlaisen ratkaisun, joten tiedän siihen tarvitsevani Angularin ngOptions -attribuuttia. Se luo select -elementtiin ruoka-annoksen sisällään pitävät option -elementit. Voisin yrittää luoda sen myös ngRepeat -attribuutilla, mutta select -listaan ngOptions on melestäni parempi valinta. Tulen tarvitsemaan get -lausekkeen aterioiden hakemiseen, joten kirjoitan

```
$scope.items = [];
$http.get('/menu/getAllItems').then(function(response){
  $scope.items = response.data;
});
```

kontrolleriin. Tämän komennon avulla saan haettua menu -taulusta ateriat. Sen jälkeen kirjoitan select -elementin, joka käy läpi hakemani taulun ja asettaa ne vetovalikkoon käyttäen aterian nimeä.

```
<select ng-options="item.id as item.Artikelname for item in items" ng-model="weeklymenu.menu"></select>
```

Nyt lomakkeessa on vetovalikko, josta voi valita olemassa olevan aterian. Ateria tallentuu weeklymenu -tauluun menu -kenttään. Tämän jälkeen luon tekstikentän kommentille, jossa käytän input -elementin text -muotoa. Käytän tässäkin kentässä ngModel -attribuuttia, joka ilmaisee ohjelmassa mikä tieto kentässä sijaitsee.

Päivän lopuksi alan tarkastelemaan erilasia mahdollisuuksia ajan valitsemiseen kalenterista. En halua alkaa rakentamaan kalenteria täysin itse, joten alan etsimään tähän tarkoitukseen luotua moduulia. Ajankäytön hallintaan käyttämäni moment ei tarjoa kalenteria, mutta sen sijaan löydän datepicker -moduulin. Merkitsen sen muistiin ja päätän aloittaa kalenterin rakentamisen ensi kerralla tämän moduulin kanssa.

Perjantai 24.11.2017

Aamulla sain tietää esimieheni ja vanhemman kehittäjän keskustelleen Wordpressin käytön lopettamista. Viime viikolla esiintynyt ongelma Visual Composer -liitännäisen ja Wordpressin uuden version yhteensopivuudesta nosti esiin sen ongelman, joka Wordpress -sivustoissa voi olla. Nimittäin tämänkaltaisia ongelmia on vaikeaa korjata, sillä vastuu tiettyjen osa-alueiden kehityksestä siirtyy kolmansille osapuolille. Tässä tapauk-

sessa siis WordPressille, käytössämme olevan teeman kehittäjälle sekä Visual Composer -liitännäisen omistavalle taholle. Tahdomme ensisijaisesti pystyä aina vaikuttamaan itse nettisivujen kehitykseen ja ylläpitoon, mutta samalla hyödyntää sisällönhallintaohjelman tarjoamia etuja.

Päätimme, että käytän alkupäivän tutkimalla eri vaihtoehtoja WordPressille. Lounastauon jälkeen käymme yhdessä läpi eri vaihtoehtoja ja esitän oman mielipiteeni korvaavasta CMS-järjestelmästä. Myös vanhempi kehittäjä tulee esittämään oman näkemyksensä asiasta. Omalta kohdaltani koen WordPressin ongelmaksi sen, että suurimman osan ajasta käytän muiden koodia. Uuden ominaisuuden tekemiseen riittää yleensä jonkin liitännäisen lataaminen ja itse koodin muokkaus osoittautuu usein hankalaksi tehtäväksi. Tässä kiteytyykin Wordpressin olemus: se on varsin helppo tapa rakentaa nettisivusto, mutta ongelmatilanteissa joudut luottamaan muiden tekemiin ratkaisuihin. Toinen itseäni häiritsevä elementti on Wordpressin suuri koko. Latausaika on aina yksi ongelmista, joita Wordpress -projekteissa joudun selvittämään. Järjestelmä tuntuu välillä turhan raskaalta pienenkin sivuston rakentamiseen. Tämän takia koen, että Wordpressin korvaavan CMS -järjestelmän tulee olla kevyempi ja antaa enemmän vapauksia kehittäjälle muokata sivuston koodia.

Tästä syystä en huomioi suosittua Joomla -sisällönhallintajärjestelmää, sillä se on monella tavalla samankaltainen kuin WordPress. Lukemani perusteella Drupal vaikuttaa lupaavammalta vaihtoehdolta. Se on myös varsin suosittu ja sen hyötyihin luetaan laajempi muokattavuus. Toisaalta myös sen sanotaan olevan kehitysalustana hieman haastavampi. En koe haastavuutta silti haitaksi, sillä Wordpressin helppous ei ole ollut yrityksessämme suuri etu. Näistä syistä päätän esittää siirtymistä Drupal -sisällönhallintajärjestelmään.

Lounastauon jälkeen aloitamme kokouksen. Esittelemäni Drupal on ennestään tuttu sekä esimiehelleni että vanhemmalle kehittäjälle, tosin he eivät ole yrityksessä vielä sitä kokeilleet. Syyni sen valitsemiseen ovat heidän mukaansa hyvät ja oikeat, joten koen heidän olleen positiivisia Drupalin tarjoamista hyödyistä. Vanhempi kehittäjä oli omassa ehdotuksessaan valinnut samoista syistä toisen sisällönhallintajärjestelmän nimeltään Grav. Grav on rakennettu olemaan mahdollisimman kevyt ja muokattava. Gravissa on mahdollista muokata myös itse järjestelmän käyttöliittymää. Samoin kuin monissa muissakin CMS -järjestelmissä, myös Gravissa voi ladata liitännäisiä. Nämä liitännäiset ovat kuitenkin myös erittäin muokattavia ja vaativat kehittäjältä niiden muokkaamista omaan sivustoon sopivaksi. Gravin käyttöjärjestelmä on luotu antamaan kehittäjälle enemmän vapauksia

muokata pohjatiedostoja sekä rakentaa tekniikasta tietämättömälle asiakkaalle alusta, johon luoda helposti sisältöä sivulle.

Tutustuttuani Graviin paremmin yhdyn vanhemman kehittäjän näkemykseen siitä. Se ei ole välttämättä yhtä suosittu kuin ehdottamani Drupal, mutta antaa meille enemmän vapauksia sivuston suhteen. Sen avulla emme joudu luottamaan muiden luomiin liitännäisiin tai eri versioiden yhteensopivuuteen. Saan esimieheltäni toimeksiannon luoda pohja Graville, johon voimme alkaa rakentamaan seuraavaa sivustoa. Tehtäväni on opetella sen toimintaperiaatteet, jotta seuraavassa projektissa voimme jo alkaa käyttämään sitä WordPressin sijasta. Jatkossa tehtäväni tulee olemaan saada valmiiksi ruokalistaprojekti sekä tutustua uuteen valitsemaamme sisällönhallintajärjestelmään.

Viikkoanalyysi

Viikon aikana pääsin jälleen työskentelemään erilaisissa tehtävissä, joka on yleensä mukavaa. Tällä kertaa pystyn analysoimaan muitakin aiheita kuin vain painotalon sivustoa sekä viikkomenua. Viime viikon lopulla esimieheni oli keskustellut entisen asiakkaamme kanssa palvelimen vaihtamisesta, josta oli sovittu tarkemmin täksi päiväksi. Sain sähköpostin kautta uuden palvelimen yhteystiedot ja tarkoitus oli siirtää sivusto päivän aikana uudelle palvelimelle. Vanhempi kehittäjä tekisi hakukoneenoptimoinnin sekä muut mainontaan ja hakukoneisiin liittyvät toimenpiteet siirtämiseni jälkeen. Tavoitteemme oli, ettei sivusto olisi näkymätön pitkään.

En ollut ennen siirtänyt sivustoa palvelimelta toiselle ja tehtävä tuli saada päivän aikana tehtyä, joten päätin etsiä internetistä neuvoja. WordPressin ollessa suosittu järjestelmä ajattelin jonkun varmasti tehneen siirron aikaisemmin. Löysin wpbeginner.com -sivustolta (2018) ohjeet WordPress -sivuston siirtämiselle, joten päätin seurata sivuston antamia neuvoja. Wpbeginner.com on sivusto, johon olen myös opinnäytetyön aikana luottanut ja usein sivustolta löytyy hyviä neuvoja WordPress -kehittäjälle. Tällä kertaa ohjeissa käytetään Duplicator -liitännäistä, joka on luotu juuri sivuston siirtämiseen. Seuraan ohjeita liitännäisen käyttöön ja tiedostojen siirtoon käytän tällä kertaa Cyberduck -tiedonsiirto-ohjelmaa, jota toinen kehittäjä minulle oli suositellut FileZillaa nopeamman siirtonopeuden takia. Yritin tätä viikkoanalyysia varten etsiä hänen väitettään tukevan artikkelin, mutta en tällaista löytänyt. Cyberduckin nopeampaan latausnopeuteen voidaan katsoa syyksi sen, että sen voi asetuksista säätää käyttämään useaa samanaikaista yhteyttä tiedostojen lataukseen (Cyberduck).

WordPressiin liittyvä toinen teema oli itse asiassa sen käytön lopettaminen. Viime viikon WordPressin uuden päivityksen ja Visual Composer -liitännäisen aiheuttama virhe johti tilanteeseen, jossa päätimme siirtyä toisen sisällönhallintaohjelman käyttöön. Päätös oli siltä osin tärkeä, että minusta tulisi uuden käytettävän järjestelmän pääasiallinen kehittäjä yrityksessä. Tämän takia käytimme kokonaisen päivän päätöksen tekemiseen. Lopulta päädyimme Grav -sisällönhallintajärjestelmään, joka korvaisi WordPressin. Suurin ero kolmen suurimman, perinteisen CMS-järjestelmän – WordPressin, Drupalin ja Joomlan – ja Gravin välillä on se, että siinä missä perinteiset järjestelmät ovat tehty yksinkertaisten sivustojen kehittämisen helpottamiseksi, monimutkaisempien ratkaisujen kehittäminen on huomattavasti vaikeampaa. Grav ei käytä myöskään perinteisten CMS-järjestelmien tavoin tiukkaa tietokantarakennetta, joten järjestelmää on muokattavampi. (Miller 2015.)

Viikon aikana sain mahdollisuuden myös työskennellä viikkomenun parissa. Tällä kertaa viikon tavoitteena oli kehittää aterian lisäyslomaketta ja erityisesti päivämäärän lisäystä sekä toisesta taulukosta saatavien ruoka-annosten vetovalikkoa. Angular tarjoaa kehittäjälle erilaisia direktiivejä lomakkeen kenttien hyödyntämiseen ja projektissa käytän ngOptions -direktiiviä. Sen avulla voin käydä läpi taulukon muuttujat ja syöttää ne select – elementtiin (Dayley, 451). Vetovalikon teko oli melko helppoa tehdä, sillä tarvittavan get-lauseen luonti onnistui ottamalla mallia aiemmista luomistani lauseista. Ainoa muuttuva tekijä oli tietokantataulu, josta sen hain. Tällä viikolla en ehtinyt kuin pintapuolisesti tutustumaan ajankäytön valitsemiseen käytettävistä työtavoista ja mahdollisesta moduulista. Seuraavat vaiheet ruokalistaprojektissa tulevat olemaan ajan määrittäminen aterialle, aterioiden näyttäminen päivämäärän perusteella sekä erilaiset pienet käyttäjäkokemusta parantavat ominaisuudet.

4 Pohdinta ja päätelmät

Olin ennen raportointini aloittamista työskennellyt yrityksessä jo vuoden verran, joten siltä osin työni luonne ei muuttunut suuresti. Pääsin aloittamaan viikkoraportointini käynnissä olevan painotaloprojektin tiimoilta, joten opinnäytetyössä olevat kymmenen viikkoa kuvastivat mielestäni hyvin projektin etenemistä sen keskivaiheessa. Pääsin kirjoittamaan myös projektin synnystä sekä ensimmäisistä työvaiheista Angularilla ja node.js -run-time-ympäristöllä rakentamani viikkomenun kautta. Viikkoraporttien alkuvaiheessa sain myös toimeksiannon rakentaa konsultille pienimuotoisen nettisivun, jonka sain tehtyä viikkojen aikana alusta loppuun. Tarkastellessani viikkoja on täten mukavaa nähdä, että pystyin raportoimaan työprojektieni eri vaiheista.

Liittyessäni yritykseen vuosi ennen raportointiviikkojen alkamista, sain mahdollisuuden työskennellä ensi kertaa Wordpress -sisällönhallintaohjelman kanssa. Koska olin siis ehtinyt jo oppimaan melko paljon kyseisestä sisällönhallintajärjestelmästä, minusta tuntui, etten tulisi oppimaan kovinkaan paljon uutta Wordpressistä raportointiviikkojen aikana. Myös Angulariin tutustuin vasta työpaikassani ja sen parissa olin ehtinyt työskentelemään jo kahdessa aiemmassa projektissa. Odotukseni olivat raportointivaiheen alussa siis melko vähäiset tietotasoni kehittymisen suhteen. Tunsin ennalta yrityksen sidosryhmät ja ryhmädynamiikan sekä yrityksen toimintatavat, joten raportoinnin haasteena tulisi olemaan tehtävien ja yrityksen esittely tavalla, josta selviää olennainen tieto – nähdä ja kertoa niistä tavallaan ensikertalaisen silmin.

Ilokseni sain todeta, että toiminnastani löytyi loppujen lopuksi yllättävän paljon kirjoitettavaa. Otin työssäni toimintaperiaatteeksi esitellä ja käydä läpi keskeiset käsitteet ja perusteet, jonka kautta huomasin lopulta oppivani myös itse paljon. Kuten mainitsin, tutustuin useisiin kehitystapoihin vasta yritykseen siirryttyäni ja jouduin melko nopeasti tuottamaan aineistoa. Tämän seurauksena minulta oli jäänyt paljon esimerkiksi Javascriptin perusasioita oppimatta. Keskityin opettelemaan usein vain ongelman ratkaisemiseen tarvittavan tiedon, joten opinnäytetyön avulla pääsin perehtymään tarkemmin muun muassa Angularissa käytettyihin ng-direktiiveihin ja niiden tarjoamiin mahdollisuuksiin. Opinnäytetyön tekoa varten hankkimani kirjat osoittautuivat myös hyviksi valinnoiksi kokonaisvaltaiseen oppimiseen. Erityisesti tahdon mainita Wordpressistä kertovan kirjan Professional Wordpress: Design and Development (Wrox 2015). Kuvittelin vuoden kokemukseni jälkeen tuntevani Wordpressin tarpeeksi hyvin, mutta kirjan avulla huomasin puutteet tietotasossani. Sen avulla opin paljon hyviä käytänteitä varsinkin tietoturvan ja sivuston vierailijoiden käyttäjäkokemuksen parantamiseksi.

Eräs keskeisistä teemoista, joka esiintyy opinnäytetyössäni, on ajankäytön hallinta. Kuten tämän kappaleen alussa mainitsin, työskentelin opinnäytetyöhön käytetyn kymmenen raportointiviikon aikana kolmessa eri projektissa ja sain yhteydenottoja vanhempia projekteja koskien. Tämän lisäksi myös työpäivien määrä viikkojen aikana vaihteli, mutta keskimäärin minulla oli kolme työpäivää viikossa. Työpäivien vähäisemmän määrän takia minun tuli olla mahdollisimman tehokas, jotta edistyisin tehtävissäni. Ensimmäistä kertaa yrityksessä työskennellessäni kohtasin ongelman, jossa minun täytyi keskittyä useaan eri projektiin. Joskus kävikin niin, että päivän aikana jouduin siirtämään huomiotani usean eri projektin välillä, joka aiheutti kohdallani tarpeetonta stressiä. Tämän lisäksi tehtävään perehtymiseen kului aina hieman aikaa ja yhteen laskettuna tehtävien välillä siirtymiseen kului yllättävän paljon aikaa, joka mielestäni oli turhaa.
Opinnäytetyön edetessä näkyy omasta mielestäni kehittymiseni tässäkin asiassa. Vertaillessani ensimmäisiä ja viimeisiä päivätekstejä, on selkeästi havainoitavissa, kuinka olen pystynyt jaottelemaan tehtävät eri päiville. Viikkoraporttien loppua kohden on nähtävissä, kuinka olen jakanut painotalolle rakentamani sivuston kehityksen viikon alulle ja ruokalistan teon taas viikon viimeisille työpäiville. Sen lisäksi että aikaa säästyy, se oli myös itseni kannalta henkisellä tasolla parempi ratkaisu. Työpäivän aloittaminen oli huomattavasti helpompaa, kun tiesi mikä sen päivän suunnitelma oli.

Yritys jossa työskentelin, oli erittäin pieni. Sen toiminta perustui ensisijaisesti asiakkailta saatuihin IT-projekteihin sekä RFID-tuotteita myyvään verkkokauppaan. Verkkokaupan tuottama tasainen tulonlähde toi vakautta yrityksen toimintaan, mutta projektien kautta yritys pystyi saamaan paremman tuloksen. Projekteihin perustuvan pienen yrityksen toiminta on kuitenkin välillä hankalaa, sillä tulevaan on vaikeaa valmistautua. On ajanjaksoja, jolloin hankkeita olisi tarjolla paljon – sekä aikoja, jolloin kukaan ei ota edes yhteyttä. Yrityksen täytyy ennen hankkeeseen ryhtymistä harkita asiaa myös käytettävissä olevien henkilöstöresurssien osalta, sillä yrityksen pienen koon takia kaikkia projekteja ei yksinkertaisesti pystytä toteuttamaan.

Opinnäytetyöni aikana oli käynyt niin, että yritys oli ottanut tehtäväkseen liian monta hanketta. Omalla kohdallani se aiheutti edellä mainitsemani ongelman ajankäytön suhteen, mutta myös itsenäisen työskentelyn lisääntymisen. Vanhempien työntekijöiden aikataulu muuttui varsin kiireiseksi, joten heillä ei ollut kovinkaan paljon aikaa avustaa minua ongelmatilanteissa. Aiemmin minulle oltiin annettu ohje pyytää aina apua vanhemmilta kehittäjiltä, jottei minulla kulu aikaa virheiden selvittämiseen. Yrityksen tavoitteena on lopulta kuitenkin tuottaa tulosta, joten ongelmia yhdessä selvittämällä pystyimme edistymään hankkeissamme nopeammin ja sain tehtyä työni paremmin. Tämä osoittautui monella tavoin positiiviseksi ongelmaksi, sillä sen kautta sain kehitettyä ongelmanratkaisutaitoani

edelleen. Myös projekteihin liittyviin päätöksiin pääsin vaikuttamaan enemmän - ruokalistan kehityksessä tein kaikki työvaiheet aina suunnittelusta ohjelmointityöhön saakka itse.

Opinnäytetyötä tehdessäni huomasin nauttivani front-end -painotteisesta työstä. Erityisesti pidin kehitystyöstä Angularin ja node.js:n kanssa. Koen, että olen oppinut yrityksessä valtavasti erityisesti webohjelmoinnista sekä laaja-alaisesti ohjelman eri kehitysvaiheista. Pienessä yrityksessä täytyy osata suunnitella ohjelman rakenne ja kehitykseen käytettävät työkalut, mutta myös sen testaaminen on ohjelmoijan vastuulla. Tämän lisäksi häneltä vaaditaan myös kykyä nähdä ohjelma asiakkaan ja liike-elämän näkökulmasta. Wordpress -projektissa ongelmat painottuivat erityisesti juuri käyttäjäkokemuksen parantamiseen sekä sivuston pääasiallisen tehtävän korostamiseen, joka oli sivuston vierailijan yhteydenottamiseen kontaktilomakkeen avulla. Ohjelmointitaitojeni ylläpitämiseksi ja uuden oppimiseksi oli hyvä, että sain ruokalistaprojektin. Päiväkirjaraportoinnissa pystyin hyödyntämään näitä kahta melko erilaista projektia ja kykenin kertomaan tehtävistäni sekä teknisellä, ohjelmien käytännön toimivuuden tasolla, että mainontaan ja asiakaskokemukseen liittyvällä liike-elämän tasolla. Henkilökohtainen mielipiteeni on, että opiskelijan on hyödyllisempää työskennellä pienessä yrityksessä, sillä silloin pääsee näkemään alan toimintaa laajemmin – vaikkakin pienemmässä mittakaavassa.

Viimeisten raportointiviikkojen aikana keskustelimme useastikin Wordpressiin liittyvistä ongelmista, jotka liittyivät riippuvaisuuten kolmansista osapuolista sivuston toimivuuden kannalta. Kyseisessä sisällönhallintajärjestelmässä hyödynnetään tuhansien kehittäjien rakentamia teemoja ja liitännäisiä, jotka eivät aina kuitenkaan ole keskenään yhteensopivia. Viikkojen aikana kohtasimme yhden melko suuren ongelman juuri tähän liittyen, joka oli anteeksiantamatonta. Asiakkaidemme tulee pystyä luottamaan, että tekemämme ohjelmat ja sivustot toimivat. Koska tämä ei ollut ensimmäinen kerta, kun Wordpress -sivustomme lakkasi toimimasta, päätimme jatkoa ajatellen siirtyä johonkin muuhun ratkaisuun. Tahdomme tarjota asiakkaille mahdollisuuden muokata heidän sivustonsa aineistoa ja tästä syystä päätimme etsiä jonkun toisen web-sisällönhallintajärjestelmän. Kuten viimeisen seurantaviikon osiosta selviää, pohdimme vakavasti Grav -sisällönhallintajärjestelmään siirtymistä.

Alkuasetelmiin nähden olen tyytyväinen opinnäytetyössä esittelemiini tuloksiin. Pystyin raporttiviikoilla kertomaan mielestäni hyvin niistä haasteista ja onnistumisista, joita viikkojen aikana koin. Pyrkimykseni oli myös alusta asti kuvata työni ja siihen liittyvien teknisten aspektien lisäksi myös yleisesti työelämää Saksassa sekä pienessä ohjelmointiyrityksessä. Koen ylpeyttä siitä työstä, mitä olemme vähäisellä henkilöstöllä pystyneet saavuttamaan sekä omista saavutuksistani. Pienessä yrityksessä toiminta tulee optimoida tarkasti

ja työnteko on itsenäisempää. Opinnäytetyössä pääsin käymään läpi omat heikkouteni ja parantamaan niitä. Tarkastellessani viikkoraportteja, on motivoivaa nähdä se kehitys, mitä koin näiden kymmenen viikon aikana. Kirjoittamieni viikkoanalyysien avulla sain mahdollisuuden syventää tietotasoani töissä käyttämistäni työtavoista ja ohjelmista. Viikkoanalyyseja kirjoittaessani näin myös ne virheet ja parantamisen kohteet viikon tehtävissä. Aion jatkossakin pyrkiä käymään läpi viikon aikana tehdyt työtehtävät ja miettiä eri lähestymistapoja ongelmiin. Oppimani perusteella on hyvä jatkaa eteenpäin ohjelmistokehittäjän ammatissa.

Lähteet

Agrawal, H. 2017. How To Uninstall W3 Total Cache WordPress Plugin. Luettavissa: https://www.shoutmeloud.com/how-to-delete-uninstall-w3-total-cache-wordpress-plugin.html. Luettu 23.9.2017.

Agrawal, H. 2017. WP-Rocket – Is It Really The Best Cache Plugin For WordPress? Luettavissa: https://www.shoutmeloud.com/wp-rocket-review-cache-plugin.html. Luettu 23.9.2017.

AngularJS. AngularJS API Docs. Luettavissa: https://docs.angularjs.org/api. Luettu 16.5.2018.

Bowby, S. 2014. 6 Phases of the Web Site Design and Development Process. Luettavissa: https://www.idesignstudios.com/blog/web-design/phases-web-design-development-process/#.Wgh3OIhw2Uk. Luettu 7.10.2017.

Computer Hope. 2017. How to prevent a div from breaking to the next line. Luettavissa: https://www.computerhope.com/issues/ch001709.htm. Luettu 13.5.2018.

Cyberduck. File Transfers. Luettavissa: https://trac.cyberduck.io/wiki/help/en/howto/transfer. Luettu 17.5.2018.

Dayley, B. 2014. Node.js, MongoDB and AngularJS Web Development. Addison-Wesley.

Djuraskovic, O. 2017. Speed up site by properly added Google Maps – wipe 70 req & 2MB. Luettavissa: https://firstsiteguide.com/speed-site-properly-including-google-maps/. Luettu 15.5.2018.

Dominique, J. 2015. How To Downgrade WordPress To A Previous Version. Luettavissa: http://dominiquej.com/how-to-downgrade-wordpress-to-previous-version/. Luettu 17.5.2018.

Grigorik, I. 2018. Image Optimization. Luettavissa: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization. Luettu 6.3.2018.

Keith, K. 2015. 6 ways to stop Contact Form 7 spam on WordPress websites. Luettavissa: https://barn2.co.uk/stop-contact-form-7-spam/. Luettu 5.5.2018.

Kelly, R. How Webpage Load Time Is Related to Visitor Loss. Luettavissa: https://pearanalytics.com/blog/2009/how-webpage-load-time-related-to-visitor-loss/. Luettu 6.3.2018.

Locke, J. 2017. Using Adobe Color (Kuler) for Web Design. Luettavissa: https://www.lockedowndesign.com/adobe-color/. Luettu 23.11.2017.

Miller, A. 2015. Traditional CMS Platforms and Grav. Luettavissa: https://getgrav.org/blog/traditional-cms-platforms-and-grav. Luettu 17.5.2018.

Morey, R. 2017. How to Install XAMPP and WordPress Locally on PC/Windows. Luettavissa: https://premium.wpmudev.org/blog/setting-up-xampp/?utm_expid=3606929-109.P6e7JvhjTrWFxwrJZjRkog.0&utm_referrer=https%3A%2F%2Fpremium.wpmudev.org%2Fblog%2Fsetting-up-xampp%2F. Luettu 23.9.2017.

Moment.js. Parse, validate, manipulate, and display dates and times in JavaScript. Luettavissa: https://momentjs.com/. Luettu 16.5.2018.

PageSpeed Tools. 2018. Enable Compression. Luettavissa: https://developers.google.com/speed/docs/insights/EnableCompression. Luettu 9.3.2018.

Peshev, M. 2016. Don’t Call Yourself a Developer If You Don’t Code. Luettavissa: https://devwp.eu/dont-call-yourself-a-developer-if-you-dont-code/. Luettu 24.9.2017.

Qode Themes -keskustelupalsta. Visual Composer Not Working Since WP 4.9 - keskustelu. Luettavissa: https://qode.ticksy.com/ticket/1373301/. Luettu 17.5.2018.

Quickscrum. 2017. What is Git? What benefits does Git offer? Luettavissa: https://www.quickscrum.com/Article/ArticleDetails/5181/1/What-is-Git-What-benefits-does-Git-offer. Luettu 15.5.2018.

Varying Vagrant Vagrants. Luettavissa: https://varyingvagrantvagrants.org/. Luettu 31.9.2017.

Weller, N.B. 2015. How To Customize The Style Of Contact Form 7 To Match Your Website. Luettavissa: https://www.elegantthemes.com/blog/tips-tricks/how-to-customize-the-style-of-contact-form-7-to-match-your-website. Luettu 13.5.2018.

White, M. 2014. How WordPress Could Sink Your Business. Luettavissa: https://doc4design.com/news/wordpress-could-sink-your-business. Luettu 30.9.2017.

Williams, B., Damstra, D. & Stern, H. 2015. Professional WordPress: Design and Development. 3. painos. Wrox.

WordPress.org. WordPress Optimization/Caching. Luettavissa: https://codex.wordpress.org/WordPress_Optimization/Caching. Luettu 7.3.2018.

WordPress. WP Google Maps. Luettavissa: https://fi.wordpress.org/plugins/wp-google-maps/#description. Luettu 15.5.2018.

Wpbeginner. 2018. How to Add Google Maps in WordPress. Luettavissa: http://www.wpbeginner.com/wp-tutorials/how-to-add-google-maps-in-wordpress/. Luettu 13.5.2018.

Wpbeginner. 2017. How to Move WordPress From Local Server to Live Site. Luettavissa: http://www.wpbeginner.com/wp-tutorials/how-to-move-wordpress-from-local-server-to-live-site/. Luettu 27.9.2017.

Wpbeginner. 2018. How to Move WordPress to a New Host or Server With No Downtime. Luettavissa: http://www.wpbeginner.com/wp-tutorials/how-to-move-wordpress-to-a-new-host-or-server-with-no-downtime/. Luettu 18.5.2018.

Wright, K. 2017. 5 Common WordPress Security Issues. Luettavissa: https://ithemes.com/2017/01/16/wordpress-security-issues/. Luettu 5.5.2018.