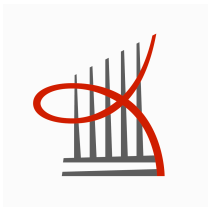


INTERAKTIIVISEN VERKKOSOVELLUKSEN TOTEUTUS PROTOTYYPISTÄ LOPPUTUOTTEEKSI

Niklas Lepistö

Opinnäytetyö
Joulukuu 2014
Tietojenkäsittely
Digimedia

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Digimedia

LEPISTÖ, NIKLAS:
Interaktiivisen verkkosovelluksen toteutus prototyypistä lopputuotteeksi

Opinnäytetyö 28 sivua, joista liitteitä 2 sivua
Joulukuu 2014

Demtrees-nimisen blogisivuston sisällönhallintajärjestelmä oli toteutettu vanhentuneella teknologialla. Sivuston hallinta oli vaikeutunut ja monimutkaistunut. Sivuston ulkoasu ei ollut enää nykyaikainen eikä vastannut demtreesin brändiä. Opinnäytteen tavoitteena oli kehittää nykyaikainen ja interaktiivinen verkkosovellus korvaamaan sisällönhallintajärjestelmä. Kehitettävälle sovellukselle asetettiin laatutavoitteiksi blogikirjoitusten tuottamisen nopeuttaminen ja sivuston ulkoasun uudistaminen nykypäivän trendejä soveltaen.

Sovelluksen suunnittelussa ja toteutuksessa hyödynnettiin iteratiivista kehittämismallia. Tuloksena saatiin sovellus, joka koostuu palvelin- ja asiakasrajapinnoista, jotka on toteutettu moderneilla teknologioilla, kuten Node.js-suoritusympäristöllä ja AngularJS-kehyksellä.

Sovellus on tarkoitus ottaa käyttöön vuoden 2014 lopulla. Jatkokehitys toteutetaan vuoden 2015 aikana.

Asiasanat: verkkosovellus, Node.js, AngularJS

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Digimedia

Niklas Lepistö
Developing a Interactive Web-Application from Prototype to Released Product

Bachelor's thesis 28 pages, appendices 2 pages
December 2014

The purpose of this thesis was to develop a modern and interactive web-application to replace the current content management system on a blog site called demtrees. The application's main goals were to ease and simplify website management, speed up the process of making a blog entry and renew the current visual layout to be more like today's design trends.

Both the design and development processes were done through interactions. The finished application consists of a server backend and client frontend, which have been developed using modern technologies, such as Node.js –runtime environment and AngularJS-framework.

The application is going to be pushed to production environment at the end of the year 2014. Further development of the application is done during 2015.

Key words: web-application, Node.js, AngularJS

SISÄLLYS

LYHENTEET JA TERMIT

AngularJS	Web-sovelluskehys
API	**A**pplication **P**rogramming **I**nterface, sovelluksen rajapinta
Backend	Sovelluksen palvelinpuolen toteutus, ei näy loppukäyttäjälle
CSS3	**C**ascading **S**tyle **S**heets, tyylitiedostokieli, jota käytetään esimerkiksi verkkosivujen ulkoasun määrittämiseen. Lopun numero kuvaa versionumeroa.
Frontend	Sovelluksen loppukäyttäjälle näkyvä asiakasrajapinta, client
Git	Linux-käyttöjärjestelmän luojan Linus Torvaldsin kehittämä versiohallintatyökalu
Gulp.js	Node.js –kehitysympäristön pystyttämiseen käytetty moduuli.
Hapi.js	Node.js:ään pohjautuva palvelinpuolen ohjelmistokehys
HTML5	**H**yper **T**ext **M**arkup **L**anguage, verkkosivujen toteuttamiseen käytetty staattinen kieli. Lopun numero kuvaa versionumeroa.
JavaScript	Koodikieli, joka mahdollistaa dynaamisia tyypityksiä ja toteutuksia
Markdown	Työkalu tekstin muuttamiseksi HTML-syntaksiksi
MongoDB	Avoimen lähdekoodin tietokanta, NoSQL
Node.js	JavaScript-ympäristöön pohjautuva alusta
prototyyppi	Ensimmäinen versio tuotteesta
Rautalankamalli	Esimerkiksi käyttöliittymän sisällön, toimintojen ja ulkoasun luonnosteluun käytettävä menetelmä
stack	Sovelluksessa käytetyt koodikirjastot ja –kehykset, esim. MEAN-stack (**M**ongoDB, **E**xpress, **A**ngularJS, **N**ode.js) ja LAMP-stack (**L**inux, **A**pache, **M**ySQL, **P**HP).
Stylus	Tyylittelykieli ja esikäsittelijä, jota käytetään nopeampaan ja tehokkaampaan CSS-tyylitiedostojen luomiseen
Subversion	Versionhallintatyökalu
UX	**U**ser E**x**perience, käyttäjäkokemus
VPS	Virtuaalipalvelin, **V**irtual **P**rivate **S**erver

1 JOHDANTO

Demtrees on vuonna 2011 perustettu yhden henkilön ylläpitämä blogisivusto, jonka aiheina ovat muun muassa taide ja valokuvaus. Sivuston nykyinen konsepti on ylläpitäjän mukaan vanhentunut ja asialle halutaankin muutosta; demtrees uudistaa brändinsä vuoden 2014 loppuun mennessä.

Opinnäytetyön tavoitteena on toteuttaa demtreesille nykyaikaisia teknologioita mukaileva interaktiivinen eli vuorovaikutteinen verkkosovellus. Opinnäytetyöraportissa kuvataan koko verkkosovelluksen kehittämisprosessi alkaen määrittelyvaiheesta aina valmiiseen lopputuotteeseen asti. Työssä kuvataan käytettyjä teknologioita ja pääpaino on JavaScript-kehysten ja –kirjastojen esittämisessä. Opinnäytetyössä sovelluksen kehittämistyö tehdään iteratiivisen kehittämismallin mukaisesti.

Opinnäytetyön tavoitteena on toteuttaa verkkosovelluksesta mahdollisimman helppokäyttöinen ja nykyaikainen sisällönhallintajärjestelmä. Nykyaikainen sisällönhallintajärjestelmä hyödyntää moderneja teknologioita, joiden avulla voidaan esimerkiksi tehdä toiminnallisuudesta reaaliaikainen ja interaktiivinen. Sovelluksen interaktiivisen toiminnallisuuden toteutuksella tavoitellaan sivuston kävijöiden miellyttämistä. Tärkeimpinä avainsanoina sivuston toteutuksessa ovat interaktiivisuus, responsiivisuus sekä käyttäjäläheisyys ja -kokemus. Laadullisina tavoitteina on kehittää demtreesin brändiä kuvastava helppokäyttöinen ja ulkonäöllisesti mieleenpainuva verkkosovellus. Tavoitteena on kasvattaa sivuston kävijämäärää ja tunnettuutta.

Opinnäytetyö hyödyntää lähteinään pääasiassa verkkoartikkeleita ja –sivuja. Työn ensimmäisten lukujen aikana käydään läpi sovelluksen määrittelyä ja suunnittelua, jonka jälkeen keskitytään verkkosovelluksen toteutukseen. Verkkosovelluksen toteuttaminen on pitkä ja monipuolinen prosessi. Tämä opinnäytetyö pyrkii avaamaan lukijalle, kuinka sovellusten toteutus tapahtuu ja mitä asioita on hyvä ottaa huomioon ennen projektin aloittamista ja sen aikanakin. Opinnäytetyön aiheena tehty projekti alkoi vuoden 2014 kesällä ja kehitystyössä tehtiin neljä iteraatiokierrosta. Toteutettu sovellus on tarkoitus ottaa käyttöön vuoden 2014 lopulla ja sitä on tarkoitus jatkokehittää vuoden 2015 aikana.

2 TAUSTA

2.1 Nykyinen järjestelmä

Demtreesin nykyinen sivusto on toteutettu WordPress-sisällönhallintajärjestelmällä. WordPress on avoimen lähdekoodin sisällönhallintajärjestelmä, joka on kehitetty erityisesti blogikirjoituksia varten. Kyseinen sisällönhallintajärjestelmä on ajan saatteessa kasvattanut suosiotaan myös tavallisten verkkosivustojen toteutuksessa järjestelmän helppokäyttöisyyden ansiosta. WordPress on kehitetty PHP-kielellä ja se hyödyntää SQL-tietokantoja.

Nykyinen demtreesin sivusto haluttaisiinkin päivittää uuteen, nykyaikaisempaan järjestelmään. Nykyaikaisemmalla järjestelmällä tarkoitetaan tässä sovellusta, joka on toteutettu moderneilla teknologioilla.

2.2 Iteratiivinen kehittämismalli

Iteratiivinen kehittämismalli on yksi ohjelmistokehityksen prosessimalleista, jossa kehitysprosessin perustoiminnot ovat ajallisesti limittäin. Mallissa sovelluksen ensimmäinen versio kehitetään nopeasti ja se voi olla hyvin pelkistetty. Ensimmäistä versiota parannetaan saadun palautteen perusteella koko kehitystyön ajan, kunnes on päädytty asiakkaan tarpeet ja vaatimukset täyttävään ohjelmistoon, jonka asiakas hyväksyy käyttöönsä. (Wärn 2010).

Iteratiivinen kehittämismalli soveltuu toteuttamiseen hyvin. Toteutettavan verkkosovelluksen tavoitteena on houkutella mahdollisimman monia erilaisia kävijöitä ja iteratiivisessa kehittämismallissa voidaan parhaiten hyödyntää asiakaslähtöisyyttä.

2.3 Uuden sovelluksen määrittely

Kehitettävä sovellus tulee korvaamaan sivuston nykyisen sisällönhallintajärjestelmän WordPressin. Valmiin sovelluksen toiminnallisia vaatimuksina ovat blogikirjoitusten, käyttäjäprofiilien sekä työreferenssien lisääminen, ei-toiminallisina vaatimuksina on

helppokäyttöisyys. Tietojen lisääminen tulee olla mahdollisimman yksinkertaista ja nopeaa. Sovelluksessa voidaan myös hyödyntää reaaliaikaistoimintoja, jos asiakas niin haluaa. Tämä tarkoittaisi esimerkiksi sitä, että jos asiakas lisää sovelluksen hallintapaneelista uuden blogikirjoituksen, jokaisen sivustolla olevan käyttäjän näkymän kirjoituslista päivittyisi automaattisesti. Tietoturvan kehittämisen asiakas haluaa painottaa tiedon luottamuksellisuuteen. Tietoturva-asiat tulee huomioida koko sovellus- ja jatkokehityksen ajan.

Sovelluksen ulkoasun on oltava näyttävä, mutta sen on kuitenkin pysyttävä minimalistisena ja helppokäyttöisenä. Sen tulee myös olla skaalautuva eri päätelaitteille, eli responsiivisesti toteutettu. Asiakas haluaa esimerkiksi pystyä lisäämään blogikirjoituksia omalta puhelimeltaan tarpeen vaatiessa.

2.4 Käytettävät kehitysvälineet

Toteutuksessa käytetään nykyaikaisia ja tämän hetken trendeinä toimivia teknologioita ja ratkaisuja. Nykyaikaisilla ja trendikkäillä teknologioilla tarkoitetaan tässä avoimen lähdekoodin suoritusympäristöä Node.js, frontend-puolen AngularJS-sovelluskehystä ja versionhallintatyökalua Git. Kyseiset teknologiavaihtoehdot ovat modernin sovelluskehityksen kärkeä, mikä on tärkeää asiakkaalle. Asiakas haluaa pystyä hyödyntämään uusien teknologien tuomia mahdollisuuksia ja ominaisuuksia, joiden avulla hän voi kehittää omaa toimintaansa.

Sovelluksen backendistä puhuttaessa viitataan sovelluksen palvelinpuolen tekniseen toteutukseen. Termi frontend tarkoittaa sovelluksen pintakerrosta, käyttöliittymää. Versionhallintatyökalun avulla kirjoitettu koodi saadaan varmuuskopioitua pilveen.

Node.js on JavaScript-kieleen perustuva avoimen lähdekoodin suoritusaikainen ympäristö, jonka toiminnallisuus pohjautuu asynkronisiin tapahtumakutsuihin. Asynkroninen tarkoittaa samanaikaisesti tapahtuvaa ja näihin tapahtumakutsuihin perustuvat kehykset mahdollistavat muun muassa laajenevien ja reaaliaikaisten sovellusten kehittämisen.

AngularJS on Googlen kehittämä ja vuonna 2010 julkaisema sovelluskehys, joka mahdollistaa staattisten HTML-dokumenttien muuttamisen dynaamisiksi verkkosovelluk-

siksi (AngularJS, 2010). AngularJS:n toiminta perustuu kahdensuuntaiseen tiedon sitomiseen, joka toimii käytännössä niin, että taustalla olevan mallin muutokset muuttavat myös käyttäjän näkymään tapahtuneet muutokset. Vastaavasti käyttäjän näkymässä tapahtuvat muutokset muuttavat myös mallin vastaamaan näkymän tietoja. AngularJS on tällä hetkellä yksi kattavimmista ja suosituimmista sovelluskehyksistä ja kehyksen hyvä osaaminen on hyvä etuus työmarkkinoilla. Sen toiminta on helppo sisäistää ja kokematonkin kehittäjä voi alkaa työstämään monimutkaisiakin sovelluksia. Näiden etuuksien ja työkokemuksen pohjalta päätettiin valita tämä kehittävän sovelluksen toteutusteknologiaksi.

Git on Linus Torvaldin kehittämä avoimen lähdekoodin versionhallintajärjestelmä, jonka etuina ovat nopeus sekä datan yhtenäistäminen. Gitin kehittäminen alkoi huhtikuun 3. vuonna 2005 ja sen ensimmäinen julkaisu oli neljä päivää myöhemmin huhtikuun 7. päivänä. Git on tällä hetkellä käyttöönotetuin versionhallintajärjestelmä, jolle on tarjolla useita eri sivustoja, joihin yhteisön jäsenet voivat lisätä kehittämiään lisäosia tai sovelluksia. Suosituin näistä sivustoista on GitHub (Git, git-scm.com).

Tuotantoympäristössä Node.js-ympäristön ja tämän projektin tiedostoja ajetaan PM2-moduulilla. PM2 on tuotantoprosessimanageri Node.js-sovelluksille. Sen avulla voidaan pitää Node.js-sovelluksia ajossa ikuisesti ja ladata niitä uudelleen ilman alasajoaikaa. Tuotantopalvelimelle voidaan siis siirtää ainoastaan muutetut tiedostot ja PM2 osaa automaattisesti käynnistää sovelluksen uudestaan uusien tiedostojen kanssa.

2.5 Visuaalinen toteutus

Työn visuaaliseen toteutukseen kuuluu demtreesin brändin uudistaminen. Prioriteetiltaan tärkeimpänä tavoitteena asiakas piti visuaalisen ilmeen kohentamista ensin sovellukseen ja sitten vasta muuhun graafiseen materiaaliin, kuten demtreesin logoon. Kehitysvaiheen ulkoasun ei tarvitse sisältää uutta logoa eikä yhtenäistä, uudistunutta visuaalista ilmettä, vaan asiakkaalle riittää katselmoinneissa nähtäväksi järjestelmän toiminnallisuuden eteneminen. Tuotantoon menevään version ulkoasun haluttaan mukailevan demtreesin brändiuudistuksen määrityksiä. Asiakas haluaa logosta hieman yksinkertaistetumman version, kuin mitä se nykyisin on. Värimaailman tulee pysyä lähes samana,

mutta väreistä halutaan entistä harmonisemmat. Sovelluksen lopullinen visuaalinen toteutus tapahtuu Adoben tuoteperheen tuotteilla.

Adobe on vuonna 1982 perustettu digitaalimarkkinoinnin ja digitaalisen median ratkaisujen toimittaja (Adobe 2014). Esimerkkejä Adoben ratkaisuista ovat videoiden editointiin tarkoitetut ohjelmistot Adobe Premiere ja Adobe After Effects, kuvankäsittelyohjelmisto Adobe Photoshop sekä vektorigrafiikan tuottamiseen tarkoitettu ohjelmisto Adobe Illustrator. Suunnittelussa työkaluina käytettiin Adobe Photoshopia sovelluksen tarkempaan ulkoasulliseen suunnitteluun ja Adobe Illustratoria muuhun painettavaan materiaaliin, kuten logon ja erilaisten ikonien, suunnitteluun.

Adoben tuotteet valittiin käyttöön yli kymmenen vuoden Adoben ratkaisujen kokemuksen perusteella. Pääasiallinen kokemus on tullut Adobe Photoshopin ja Adobe Illustratorin käytöstä, joita hyödynnettiin myös tämän sovelluksen ja demtreesin brändin suunnittelussa.

2.5.1 Adobe Photoshop

Adobe Photoshop on alun perin Macintosh-tietokoneille kehitetty kuvankäsittelyohjelma rasterikuvien muokkaamiseen (Adope Photoshop 2014). Photoshopin avulla on mahdollista tehdä niin monimutkaisia digitaalisia maalauksia kuin syvällisiä kuvamanipulaatioitakin.

Vuosien aikana on kokeiltu monia eri kuvankäsittelyohjelmia, mutta Photoshop on osoittautunut joka kerta parhaaksi. Photoshopin tehokkuus ja monipuolisuus ovat ainutlaatuisia ja sen käyttöliittymä on hyvinkin intuitiivinen. Se ei ole alkeellisin kuvankäsittelyohjelma, joka markkinoilla on tarjota, mutta pienen opettelun jälkeen se tuntuu lähes ainoalta.

2.5.2 Illustrator CC

Illustrator CC on Adoben kehittämä vektorigrafiikkatyökalu, jota käytetään yleisesti Photoshopin tukena (Adobe Illustrator 2014). Illustratorilla voidaan luoda kaksiulottei-

nen vektorigrafiikka, jota käsitellään jälkeenpäin Photoshopissa luomalla syvyyttä, varjostusta ja muita efektejä kyseiseen vektorituotokseen.

Illustrator on, Photoshopin lailla, erittäin moniulotteinen ohjelmisto. Sen käyttö ei pelkästään keskity painomateriaalin luomiseen vaan sillä voidaan suunnitella esimerkiksi myös ikoneita mobiili- ja verkkosovelluksiin. Kokemusten perusteella Illustratorin sanotaan olevan haastavampi käyttää, kuin Photoshop, mutta sen perinpohjin omaksuttuasi voit suunnitella erittäin näyttäviä vektoriteoksia. Mitään varteenotettavaa haastajaa ei ole Illustratorille löytynyt, sillä se tekee kaiken mitä haluat ja enemmänkin.

3 SUUNNITTELU

3.1 Käytettävyys lähtökohtana

Käytettävyydelle ei voi olla olemassa mitään yksiselitteistä määritelmää, koska siihen liittyy suuri kirjo eri tieteenaloja. On jopa sanottu, että hyvän käytettävyysasiantuntijan tulisi olla samaan aikaan esimerkiksi insinööri, psykologi, sosiologi, kasvatustieteilijä ja taitelija (Kuutti 203, 13).

Sovellusten käytettävyydessä on kyse siitä, miten hyvin käyttäjä voi käyttää ohjelman toimintoja päästäkseen haluamaansa päämäärään. Voidaan olettaa, että jonkin sovelluksen asiakaslähtöisyys on toteutunut, kun sen käytettävyys on hyvä. Kun käytettävyys on toteutettu hyvin, voidaan olettaa asiakastyytyväisyydenkin kasvavan.

Iteratiivinen kehittämismalli ottaa käyttäjät mukaan kehittämisprosessiin heti sen alusta asti. Tällä tavalla käytettävyyden asettamat vaatimukset myös parhaiten toteutuvat. Käytettävyyden suunnittelu tulee olla mukana koko kehittämisprosessin ajan.

3.2 Responsiivinen suunnittelu

Responsiivinen suunnittelu on vielä nuori, joten sen suhteen ei vielä ole olemassa vakiintuneita standardeja. Mobiililaitteiden yleistyminen on kuitenkin tuonut mukanaan yhden tavan responsiiviselle suunnittelulle. Verkkopalvelu voidaan suunnitella lähtien liikkeelle mobiililaitteista, jolloin esimerkiksi sivujen koko sopii kaikille laitteille (Leiniö 2012). Responsiivisen suunnittelun tavoitteena on saavuttaa palvelulle yhtäläinen käytettävyys päätelaitteesta riippumatta.

Responsiivinen sivusto on sivusto, joka mahdollisimman hyvin huomioi erilaiset alustat, joilla sivustoa käytetään. Sivuston tulisi olla sellainen, että kävijän vieraillessa sivustolla eri laitteilla hänen kokemuksensa sivuston käytöstä pysyy suurin piirtein samana. Sivuston päivittämisen kannalta on tärkeää, että jo sovelluksen kehittämisen aikana responsiivisuus on ollut esillä. Staattinen eli pysyvä verkkosivu pysyy samanlaisena huolimatta siitä millä laitteella siellä vieraillaan. Matkapuhelinta käytettäessä pysyvän verkkosivun tunnistaa siitä, että sivua pitää suurentaa, jotta sen sisällön näkisi. Tänä

päivänä oletuksena tulee olla, että sivusto mukautuu sen mukaan mitä laitetta käytetään (Marcotte 2011).

3.3 Visuaalinen suunnittelu

Käyttöliittymän visuaalinen suunnittelu kuuluu oleellisena osana verkkopalvelun käytettävyyteen, sillä käyttäjä katselee koko ajan edessään olevaa näkymää vieraillessaan jollakin verkkosivulla. Näköaisti on ihmisen tärkein aisti, joten näköaistiin perustuva käyttöliittymäsuunnittelu on kaikkein yleisin tapa toteuttaa käyttäjälähtöisyyttä. Käyttöliittymän visuaalinen ilme on tärkeä asia, mutta yhtä tärkeä asia on sivuston rakenteen toiminnallisuus. Jos sivuston toiminta on epäloogista ja hankalaa, ei hienokaan visuaalinen toteutus pelasta sitä huonolta käytettävyydeltä (Kuutti 2003, 25, 90).

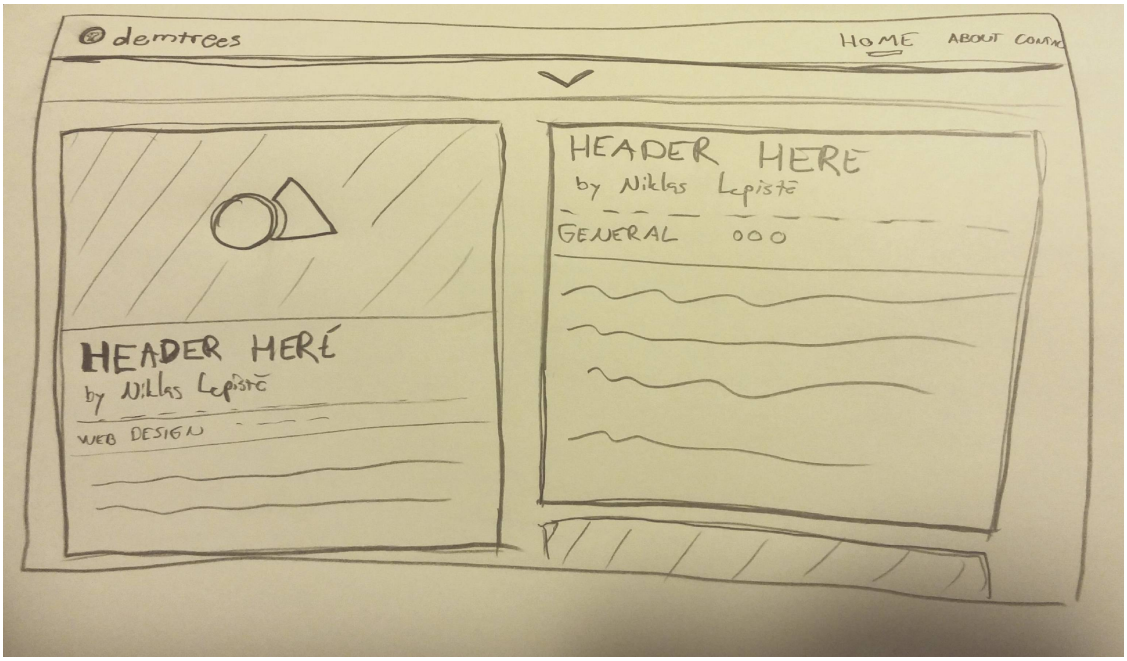
Käyttöliittymän visuaalisella eri osien sommittelulla tarkoitetaan sitä miten näkymän eri osat sijoitetaan näytölle. Visuaalinen suunnittelu on oleellinen osa käyttäjälähtöisen sivuston suunnittelua, jossa tulee ottaa huomioon ihmisen biologiset ominaisuudet kuten se miten hahmotetaan värejä ja asiakokonaisuuksia. Yleisölle suunniteltavan käyttöliittymän tulee sisältää vain yleisesti tunnettuja termejä ja sanontoja, tekniset ilmaisut saattavat aiheuttaa käyttäjissä negatiivisia käyttäjäkokemuksia vaikka ilmaisut sinällään ovat ammattikielessä tunnettuja (Kuutti 2003, 51-52, 100).

3.4 Rautalankamalli

Verkkosivuston suunnitteluvaiheen rautalankamallit olivat hyvin pelkistettyjä ja karkealla tasolla tehtyjä luonnoksia sivuston toiminnasta ja sen rakenteesta. Rautalankamalleja voidaan esimerkiksi piirtää käsin paperille. Rautalankamallin toteutus suunnitteluvaiheessa on yhtä tärkeää kuin lopullisen visuaalisen ulkoasun toteutus. Rautalankavaiheessa määritellään, miten kaikki elementit tulisi sijoittaa ja minkälaista toiminnallisuutta niillä voisi olla. Kyseisessä vaiheessa määritellään myös eri päätelaitteiden näkymät ja suunnitellaan esimerkiksi sovelluksessa käytettäviä ikoneita.

Valmiin rautalangan perusteella voidaan jo alkaa toteuttamaan sovellusta, jolloin itse visuaalinen ilme otetaan käyttöön hieman jälkikäteen. Asiat eivät kuitenkaan aina mene tässä järjestyksessä, vaan toteutus saatetaan aloittaa vasta silloin, kun kokonainen visu-

aalinen toteutus kehitettävälle tuotteelle on toteutunut. Toteutuksessa noudatettiin kuitenkin linjausta, jossa rautalankamallin jälkeen aloitettiin sovelluksen toteuttaminen. Kuvassa 1 on rautalankamalli käyttöliittymän etusivusta.



Kuva 1. Etusivun rautalankamalli

Sovelluksessa haluttiin käyttää visuaalisesti näyttäviä ja katsojaa miellyttäviä elementtejä. Etusivulle tultaessa näytettäisiin ruudun kokoinen valokuva, jonka päällä olisi jokin lausahdus sekä kehoitus vierittäää sivua alaspäin. Kuvan alapuolelta paljastuisivat blogikirjoitukset Pinterest-tyylisinä kortteina. Korttia painamalla kirjoitus avautuisi koko ruudun kokoisena. Kuvassa 2 näytetään rautalankamallia blogikirjoitusten listauksesta.

Kuva 2. Ehdotus blogikirjoitusten listauksesta

4 TOTEUTUS

4.1 Versionhallinta

Koodin varmuuskopiointi on erittäin tärkeä asia ohjelmointityössä. Nykyään varmuuskopiointi ja tehtyjen muutosten ylöskirjaaminen ei ole niin suuri prosessi kuin aikaisemmin, vaan tätä on helpotettu erilaisten versionhallintajärjestelmien, kuten Subversion ja Git, avulla.

Subversion on avoimen lähdekoodin versionhallintajärjestelmä, joka sallii verkon yli tapahtuvan tiedon muokkaamisen ja päivittämisen. Ohjelman kehittäminen alkoi vuonna 1999 ja vuonna 2001 se oli vapaasti käytettävissä. Vuonna 2010 Subversion liitettiin ASF:n tuoteperheeseen ja sen viralliseksi nimeksi tuli "Apache Subversion". (Collins-Sussman 2011). Subversionin pääasiallisena etuna on sen huolellistetumpi käyttöliittymä Gitiin verrattuna (Subversion, subversion.org).

Näitä kahta versionhallintajärjestelmää verratessa Git osoittautuu usein erittäin paljon paremmaksi vaihtoehdoksi; Git on paljon nopeampi kuin Subversion, Gitin haaraumat pitävät sisällään koko muutoslokihistorian, Gitin tietolähteet ovat pienempiä kuin Subversionin ja niiden tiedostomuodot ovat yksinkertaisia ja täten helposti korjattavissa sekä vapaampia korruptoitumiselle. Itse käytin toteutuksessani Gitiä, sillä se oli minulle työelämän kautta entuudestaan tuttu.

4.2 Työmenetelmät ja standardit

Nykyään web-kehityksen ympäristöjä on helppo pystyttää paikallisesti omalle koneelle. Erittäin suuria parannuksia ympäristöjen pystytykseen toi Node.js -teknologian julkaisu vuonna 2009 (Node.js, Github 2009). Ympäristöjen pystytys onnistuu erilaisten moduulien kautta, joista Grunt- ja Gulp.js ovat tällä hetkellä suosituimpia vaihtoehtoja.

Grunt- ja Gulp.js ovat Node.js –ympäristölle kehitettyjä moduuleita, joiden avulla voit pystyttää kehitysympäristön vaivattomasti. Grunt.js:llä kehitysympäristön pystytys perustuu enemmän käyttäjän konfiguraatioon, kun Gulp.js:llä toiminta nojaa enemmän koodin tuottamiseen. Molemmat moduulit hyödyntävät omia tehtäviään, taskeja. Näiden taskien avulla käyttäjä voi määrittää, mitä tapahtuu milloinkin ja minkä lisäosan kanssa. Voit esimerkiksi määrittää tehtävän, joka kääntää CSS-tyylisi tiiviimpään muotoon, kun ympäristö siirretään tuotantoon. Työssäni käytin kehitysympäristössäni Gulp.js -moduulia ja tälle vielä omia lisäosia, kuten BrowserSync ja Supervisor -lisäosia. Kuvassa 3 näytetään esimerkkejä Gulp-taskeista. Ylemässä taskissa on käytössä Supervisor-lisäosa ja alemmassa taskissa määritellään mitä tiedostoja katsellaan muutosten varalta ja asetetaan BrowserSyncin konfiguraatio.

```
gulp.task('supervision', ['build'], function() {
    supervisor('build/server/server.js', {
      extensions: ['js,jade'],
      ignore: ['client/', 'build/client/js/', 'server/']
    });
})

gulp.task('watch', ['supervision'], function() {

  gulp.watch(paths.styles.watch, ['styles', sync.reload]);
  gulp.watch(paths.templates.watch, ['templates', sync.reload]);
  gulp.watch(paths.partials.watch, ['partials', sync.reload]);
  gulp.watch(paths.media.watch, ['media', sync.reload]);
  gulp.watch(paths.models.watch, ['models']);
  gulp.watch(paths.plugins.watch, ['plugins']);
  gulp.watch(paths.test.watch, ['test']);
  gulp.watch(paths.index.watch, ['index', sync.reload]);
  gulp.watch(paths.server.watch, ['server']);

  var config = {
    files: [paths.scripts.source, paths.styles.source, paths.templates.source, paths.parti
    port: 7533,
    proxy: 'localhost:5533',
    open: false
  };
```

Kuva 3. Esimerkki Gulp-taskeista.

BrowserSync, nimensä mukaisesti, synkronisoi kaikki selaimesi keskenään päivittämään selainnäkymän aina tiedostomuutoksen tapahtuessa. Tämän ominaisuuden avulla voit testata sovellusta jokaisella käyttämälläsi päätelaitteella samanaikaisesti – kaikki samaan porttiin, esimerkiksi localhost:9000, yhdistetyt selaimet päivittyvät yhdellä kertaa. BrowserSync osaa myös synkronisoida käyttäjän toimintoja, eli jos kirjoitat tekstikenttään jotain pöytäkoneesi selaimella, tämä sama teksti näkyy myös puhelimesi selaimessa. (BrowserSync 2014)

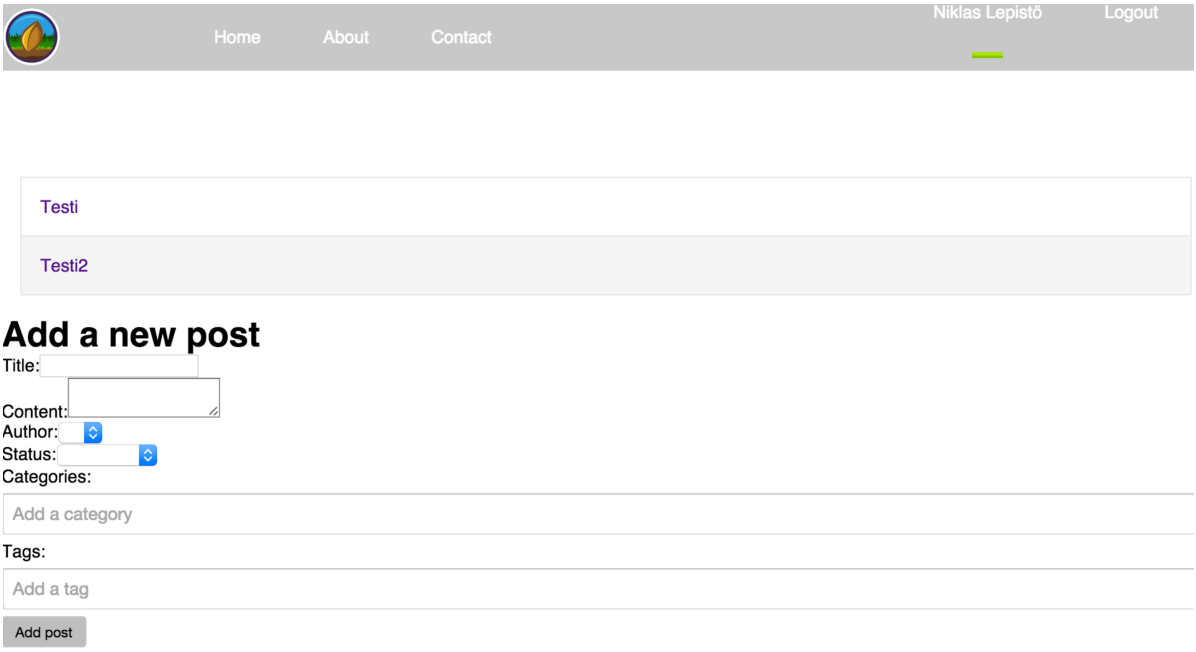
Supervisor-lisäosan tarkoitus on käynnistää kaatunut JavaScript-sovellus uudelleen. Supervisor käynnistää myös sovelluksen uudelleen tiedostomuutoksien tapahtuessa, mikä taas nopeuttaa backend-puolen kehittämistä. (GitHub 2014)

Koodin tuottamiseen käytettiin Sublime Text 2 -tekstieditoria. Tämä tekstieditorin valittiin siksi, koska se on nopea, tehokas, sisältää monia erilaisia toiminnallisuuksia ja lisäosia sekä se on hyvin paljon muokattavissa omiin tarpeisiin. Se on myös suosittu työkalu työelämässä (Sublimetext 2014).

4.3 Ensimmäinen prototyyppi

Prototyyppien kehitys on erittäin tärkeä osa asiakaskokemusta ja sovelluskehityksen kaarta. Kehityksen tapahtuessa pienissä iteraatioissa, sovelluskehityksen riskit pienenevät alusta alkaen ja asiakastyytyväisyyttä on mahdollista parantaa jokaisessa iteraatiossa, sillä asiakas pääsee näkemään, miten toteutus on mennyt eteenpäin ja hän voi antaa kommentteja, mihin suuntaan hän haluaisi sovelluksen vielä kehittyvän. Perinteisessä vesiputousmallissa sovelluskehitys on erittäin riskialtista, sillä siinä asiakas ei pääse näkemään, miten kehitys etenee, ennen kuin valmis tuote annetaan nähtäväksi. Tämän kaavan mukainen tuote ei saata vastata asiakkaan pyyntöjä ja odotuksia.

Ensimmäinen prototyyppi kehittämästäni sovelluksesta sisälsi nopean ulkoasutoteutuksen ja pientä toiminnallisuutta. Prototyypissä pystyi luomaan pääkäyttäjän tunnukset, kirjautumaan sisään sovellukseen ja lisäämään blogikirjoituksia. Otsikkokuvaa ei vielä tässä vaiheessa pystynyt määrittämään, ja kirjoituksien lisääminen tapahtui hallintapaneelista samalta sivulta, johon listattiin kaikki olemassa olevat kirjoitukset. Seuraavassa kuvassa on prototyypin hallintapaneelinäkymä.



Kuva 4. Prototyypin blogikirjoitusnäkymä hallintpaneelissa

4.4 Backend-toteutus

Sovelluksen backend-puoli on toteutettu Node.js -suoritusympäristön Hapi.js -kehyksellä. Blogikirjoitusten ja käyttäjätunnusten tallennus MongoDB-tietokannoilla.

Hapi.js on Walmart Labsin kehittämä Node.js –ympäristöä hyödyntävä palvelinpuolen kehys, joka tukee uudelleenkäytettävän sovelluslogiikan kirjoittamista (Hapi.js, 2011). Hapi.js –kehyksessä painotetaan myös tietoturva-asioita. Kehyksen valintaan vaikutti myös, että kehyksen on todettu olevan tehokas. Muun muassa kauppaketju Walmartin omat sivut, jotka hyödyntävät Hapi.js –kehystä, eivät ole kertaakaan kaatuneet Yhdysvaltojen suosittujen Black Friday –päivän alennusmyyntien aikana (Evan Hammer, 2014). Kehyksen valintaan vaikutti nimenomaan Hapi.js:n tehokkuus. Kokemuksen perusteella Hapi.js on syntaksiltaan paljon loogisempi ja helpommin omaksuttava kuin vastaavat palvelinpuolen kehykset, kuten esimerkiksi Express.js, ja tämän dokumentaatio on erittäin kattava. Seuraavassa kuvassa on yksinkertaisen sovelluksen toteutuksen vertailua Express- ja Hapi.js:n välillä.

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

app.get('/', function(req, res) {
    res.send('Hello world');
});

var server = app.listen(3000, function() {
    console.log('Express is listening to http://localhost:3000');
});
```

```
var Hapi = require('hapi');
var server = new Hapi.Server(3000);

server.route({
    method: 'GET',
    path: '/',
    handler: function(request, reply) {
        reply('Hello world');
    }
});

server.start(function() {
    console.log('Hapi is listening to http://localhost:3000');
});
```

Kuva 5 Yksinkertainen backend-toteutus Express- ja Hapi.js:llä

MongoDB on avoimen lähdekoodin dokumenttipohjainen tietokanta ja johtava NoSQL-tietokantaratkaisu. NoSQL-tietokannat eivät käytä perinteisiä SQL-kyselyitä, eivätkä ne tarvitse valmista, määriteltyä pohjaa datan säilyttämiseen. Dokumenttipohjaiset ja NoSQL-tietokannat tarjoavat dynaamisen, tehokkaan ja yksinkertaisen tavan käsitellä tietoa. Nämä tietokannat ovat myös erittäin skaalautuvia sekä korkeasti käytettävissä. Korkea käytettävyys viittaa siihen, kuinka hyvin tietokannat ovat saatavilla ongelmatilanteiden tapahtuessa. Näiden asioiden perusteella MongoDB-tietokannat valittiin sovelluksen käyttöön. Sovelluksen kautta luodut käyttäjätunnukset ja blogikirjoitukset tallennetaan MongoDB-dokumentteihin.

Tätä projektia ennen backend-kehittäminen oli vähäistä, joten helposti omaksuttava syntaksi auttoi erittäin paljon sovelluksen kehittämisessä. Pyörää ei kuitenkaan tarvinnut

keksiäuudelleen, vaan heti Hapi.js:n ja MongoDB:n yhteisen alkutustumisen jälkeen alettiin etsiä jo valmiita backend-puolen moduuleita, joilla työskentelyä voitaisiin nopeuttaa. Järjestelmä haluttiin kuitenkin toteuttaa itse, joten kaikki löydetyt valmiit paketit syrjäytettiin. GitHub-versionhallintapalvelusta hyödynnettiin käyttäjän jedireza, oikealta nimeltä Reza Akhavan, Hapi.js –runkoa, Framea.

Frame on käyttäjätunnuksien hallitsemiseen tarkoitettu API, johon kehittäjä itse luo oman frontend-näkymän. Frame oli myös saanut muiltakin GitHubin käyttäjiltä huomiota. Kyseisen paketin käyttöönotto mahdollisti keskittymisen enemmän itse blogitoimintojen kehittämiseen, eikä aikaa tuhlaantunut sovelluksen sisäänkirjautumisen toteutuksen miettimiseen. Mitä luultavammin joku toinen on jo miettinyt ja toteuttanut ne paremmin, kuin itse voisit. Framen kohdalla näin olikin. Käyttäjätunnusten hallinta oli yksinkertaista ja helppoa, eikä tähän tarvinnut kuin tehdä oma frontend-toteutus, niin Frame oli täysin hyödynnettävissä.

4.5 Frontend-toteutus

Frontendin toteutuksen työkaluiksi valittiin AngularJS-sovelluskehys, HTML5-toteutukset ja CSS3-tyylimääritelmät. Näiden kolmen yhdistelmä mahdollistaa modernin verkkosovellusten toteutuksen niin ulkonäöllisesti kuin toiminnallisestikin.

HTML5 on W3C:n kehittämän verkkosivujen standardikielen viides versio. Kyseessä olevassa versiossa on tullut HTML4-kieleen uusia elementtejä, kuten tuen video- ja äänitiedostoille, sekä integraation skaalautuville vektorigrafiikoille (HTML5, 2014). HTML5:ssä standardista poistetaan myös useita tarpeettomia elementtejä. HTML5-kielen kautta verkkosivujen tuottaminen on paljon käyttäjäystävällisempää, sillä W3C tarjoaa myös semanttisia sivuelementtejä kuvastamaan paremmin, mikä osio sivusta vastaa mitäkin. Esimerkiksi, jos HTML4-standardissa halusit määrittää yläpalkin tai osion tietylle rakenteelle, kuten vaikka blogikirjoitukselle, täytyi sinun käyttää yleistä div-elementtiä. HTML5-standardissa voit määrittää kyseisen yläpalkin käyttämällä header-elementtiä, joka kuvastaa hyvin paljon paremmin, että kyseessä on alueen yläosio.

HTML-toteutusten nopeuttamaksi valittiin Jade. Jade on Node.js-ympäristöä hyödyntävä kieli HTML-pohjien kirjoittamiseen (Jade, 2009). Jaden käyttö nopeuttaa perinteisen

HTML:n kirjoittamista huomattavasti, sillä kyseisestä syntaksista poistetaan kokonaan HTML-tagien lopetus ja sisäkkäisten elementtien luominen tapahtuu sisennyksillä. Liitteessä 1 on nähtävillä esimerkki Jade-pohjasta. Jade-pohjat voi halutessaan kääntää HTML-pohjiksi, mutta päätin itse käyttää sovelluksessani pelkkiä Jade-pohjia, sillä backend-puolen Hapi.js tarjosi niille hyvän tuen.

Myös CSS3 on W3C:n kehittämä kieli, jota käytetään verkkosivujen ulkoasun määrittelyyn ja tyylittelyyn. CSS3 on tyylittelykielen kolmas versio ja se pitää sisällään useita eri uudistuksia aikaisempaan versioon verrattuna. Yksi suurimpia, ja luultavasti käytetyimpiä, uudistuksia ovat CSS-animaatiot. Aikaisemmin verkkosivujen animointiin jouduttiin käyttämään erillisiä animoituja kuvia tai Flash-animaatioita. Nykyään näiden tekeminen onnistuu suoraan tyylitiedostojen kautta, mikä helpottaa ja nopeuttaa sivustokehitystä paljon.

Sovelluksen kehitysvaiheessa hyödynsin myös Stylusta, CSS-esikäsittelijää. Styluksen käyttö nopeuttaa verkkosovellusten tyylittelyä, koska sitä käyttäessä voit jättää aaltosulut, kaksoispisteet sekä puolipisteet pois, joita perinteisessä CSS-tyylittelyssä vaaditaan. Liitteessä 2 näytetään Styluksen syntaksia. Styluksessa on myös mahdollista asettaa muuttujia, joihin voi tallentaa esimerkiksi värikoodeja, jotka ovat myöhemmin tyyleissä helposti haettavissa. Selaimet eivät suoraan pysty lukemaan Stylus-tiedostoja, joten Stylus-tyylit käännetään normaaleiksi CSS-tyyleiksi kehitysympäristön käynnistämisen ja tiedostomuutosten yhteydessä.

Ensimmäinen Framea hyödyntävä frontend-paketti, ngFrame, oli myös saatavilla. GitHub käyttäjän Silom, Valentin Wetenkamp, kehittämä ngFrame hyödyntää AngularJS -kirjastoa. Wetenkampin ngFramen kanssa törmättiin useisiin eri ongelmiin ja jouduttiin refaktoroimaan, eli muokkaamaan, koodia paljon järkevämmäksi, hyödyntäen niitä oppeja, joita työelämässä on saanut. Omia käytäntöjä mukaillen koodista tuli paljon selkeämpää ja lyhyempää. Kaikkia osa-alueita ei prototyyppiversiossa saatu refaktoroitua, mutta tarkoituksena on korvata suuri osa Wetenkampin tekemästä koodista omalla koodilla.

4.6 Tuotantokelpoinen versio

Toteutukseni ensimmäinen tuotantokelpoinen versio sisälsi kategorioiden ja tagien listaamisen hallintapaneelissa ja näiden lisäämisen kirjoituksiin. Myös vielä puuttuva otsikkokuvien lisääminen otettiin käyttöön.

Blogikirjoituksien sisältötekstien tuottamiseen tehtiin myös pieni muutos. Ensimmäisessä prototyypissä käyttäjä pystyi lisäämään vain staattista tekstiä, mutta tuotantoversiossa kirjoitukset laitettiin hyödyntämään Markdown-syntaksia. Markdown on web-kirjoittajille suunnattu teksti-HTML -muunnin, joka mahdollistaa helppolukuisen ja helposti kirjoitettavan tekstiformaatin muuntamisen validiksi XHTML-syntaksiksi (John Gruber, Markdown). Markdown eroaa staattisesta tekstistä siten, että sen avulla voidaan tyylitellä tekstiä erikoismerkeillä. Esimerkiksi risuaidalla voidaan määrittää, että kyseessä on otsikko ja selain muutaa sen h1-elementitiksi, joka on HTML-syntaksissa otsikko.

Aivan ensimmäiseen tuotantokelpoiseen versioon ei ollut hyödynnetty demtreesin uutta visuaalista ilmettä, josta asiakas antoikin pientä kritiikkiä. Asiakas oli kuitenkin ymmärtäväinen asian suhteen, sillä kaikki minimivaatimusten mukaiset toiminnallisuudet olivat jo toteutettu. Järjestelmän tuotantovaihetta voitiin siis hyvin siirtää myöhemmäksi ajankohdaksi, jotta ulkoasun tyylittelyt saataisiin mietittyä vielä kertaalleen oikein. Asiakas kiitti toiminnoista, joissa kategorioita ja avainsanoja voitiin lisätä. Asiakas piti järjestelmän ominaisuudesta, jossa käyttäjä voi valita ikonilistasta kategoriaa tai avainsanaa kuvastavan ikonin. Liitteessä 3 näytetään sovelluksen tavallisen käyttäjän etusivunäkymä.

5 POHDINTA

Onnistuin työssäni hyvin, sillä asiakas oli tyytyväinen kehittämääni sovellukseen, niin sen nykyaikaiseen ja trendikkääseen ulkoasuun, kuin sen nopeisiin ja helppokäyttöisiin toiminnallisuuksiin. Asiakas oli myös tyytyväinen toteutettuun brändiuudistukseen ja kehitetty sovellus vastasi hyvin hänen omia näkemyksiään aiheesta. Oma ajankäyttöni projektia työstäessä olisi kuitenkin voinut olla paremmin järjesteltävissä. Päivätyön ja muiden kouluprojektien ohella työn tekeminen oli haastavaa.

Valitsemani työkalut olivat juuri oikeat tämänlaisen työn tekemiseen. Ohjelmistosuunnittelijan työssä ei voi olla ikinä liikaa tietoa erilaisista teknologioista ja toimintatavoista, joten pystyin toteuttamaan sovelluksen suurella mielenkiinnolla. Opin erittäin paljon uutta asiaa varsinkin backend-puolen JavaScript-toteutuksista ja aion jatkossakin tutustua syvemmin, kuinka asioita voitaisiin toteuttaa vielä paremmin. Opin myös sen, että kaikkia järjestelmän osia ei välttämättä tarvitse toteuttaa alusta asti itse, vaan näihin tarpeisiin löytyy jo valmiita, muiden ihmisten toteuttamia moduuleita tai lisäosia. Mutta vaikka joitain tiettyjä toiminnallisuuksia tai lisäosia olisi jo toteutettu muiden puolesta, ei se tarkoita, että ne ovat paremmin tehtyjä, kuin mitä itse voisit tehdä.

Sovelluksen kehittäminen auttoi minua myös kasvattamaan omaa käyttäjäläheistä näkemystä ja asennetta suunnittelutyöhön. Vaikka hallitsenkin Photoshopin käytön ja osaan tehdä sillä hyvännäköisiä ulkoasusuunnitelmia, ei se tarkoita, että suunnittelemani ulkoasut olisivat käyttäjäkokemuksen kannalta parhaita mahdollisia. Tässä työssä pystyin palaamaan suunnittelemiini ja toteuttamiini ulkoasuihin ja miettimään, mitkä asiat voisin toteuttaa vielä paremmin, jotta jokin toiminnallisuus olisi vielä helppokäyttöisempää ja intuitiivisempaa.

Erittäin suuret kiitokset haluaisin myös osoittaa työpaikkani työtovereille, etenkin Jaakko Saloselle, joka oli suuressa osassa opinnäytetyöni etenemistä ajatellen. REST-arkkitehtuurimalliin pohjautuvien järjestelmien kehittäminen oli itselle hyvin tuntematon alue, mutta Salonen osasi avata minulle kyseisen mallin toimintatavan muutamalla lauseella ja kuvalla.

6 Jatkokehitys

Jatkokehityksen kannalta sovimme asiakkaan kanssa, että järjestelmä olisi oma elävä organisminsa, jota kehitettäisiin ajan ja tarpeiden mukaan. Päätimme myös laatia listan toiminnallisuuksista, jotka olisivat hyvä lisätä hieman julkaisun jälkeen. Asiakas voisi kuitenkin vaikuttaa tähän listaan tarpeen vaatiessa. Uusien toiminnallisuuksien ja ominaisuuksien lisääminen ja toteuttaminen tapahtuu ensin paikallisessa kehitysympäristössä. Näiden valmistuttua ne siirrettäisiin lennosta tuotantoympäristössä olevaan järjestelmään ilman käyttökatkoksia.

Tuotantoympäristöön järjestelmän siirtäminen tehdään Flightplan.js (Patrick Stadler, Flightplan 2014, Github.com) -moduulin avulla. Flighplan.js-moduulilla voidaan ajaa lokaalisti komentorivikomentoja tarvittavien tiedostojen etsimiseen ja muistiin tallentamiseen, yhdistää ulkopuoliseen palvelimeen ja ajaa komentoja tuotantopalvelimella purkaaksesi tallennetut tiedostot oikeaan palvelinkansioon. Olen aiemmin käyttänyt tätä moduulia toisissa, vastaavan stackin projekteissa ja olen ollut erittäin tyytyväinen sen toimintaan. En myöskään ole löytänyt yhtään vastaavaa moduulia tai pakettia, jolla voisi siirtää JavaScript-sovelluksia suoraan tuotantopalvelimelle. Flightplan hyödyntää vastaavanlaista toimintoa kuin Python kielen Fabric-lisäosa.

Seuraava askel demtreesin uuden sovelluksen kehittämisessä olisi luultavasti kommentointimahdollisuuden sekä reaaliaikaisuuden lisääminen. Kumpikaan ei tosin ole välttämättömiä ominaisuuksia asiakkaalle, mutta hän on esittänyt kiinnostuksensa näiden kahden ominaisuuden lisäämiseksi. Myös backend-puolen koodin uudelleenkirjoittaminen ja korjaaminen tietyiltä osin olisi järkevää. Jatkokehitys elää kuitenkin asiakkaankin initiaatiosta, joten en voi aivan suoraan lisätä uusia toiminnallisuuksia, jotka saattavat itsestä tuntua tarpeellisilta.

7 LÄHTEET

Adobe. 2014. Luettu 4.12.2014
http://www.adobe.com/fi/

Adobe Illustrator. 2014. Luettu 3.12.2014.
http://www.adobe.com/fi/products/illustrator/features.html

Adobe Photoshop. 2014. Luettu 3.2.2014.
http://www.photoshop.com/products/photoshop

BrowserSync. 2014. Luettu 28.11.2014
http://www.BrowserSync.io/

Collins-Sussman, B., Fitzpatrick, B. ja Pilato, C. 2011. Version Control with Subversion. For Subversion 1.7. Julkaistu 2011. Luettu 2.12.2014 http://svnbook.red-bean.com/en/1.7/svn-book.pdf

GitHub. 2014. Luettu 29.11.2014. https://github.com/isaacs/node-supervisor

Gruber, J. 2014. Markdown. Luettu 23.11.2014
http://daringfireball.net/projects/markdown/

Hapi.js 2011. Luettu 3.12.2014. http://hapijs.com/

Kuutti, V. 2003. Käytettävyys, suunnittelu ja arviointi. Saarijärvi: Gummerus Kirjapaino Oy.

Leiniö, T. 2012. Mitä on responsiivinen design? Julkaistu 19.7.2012. Luettu 15.06.2013.
http://www.sofokus.com/blogi/mita-on-responsiivinen-design/

Marcotte, E. 2011. Responsive Web Design. Julkaistu 2011. Luettu 2.12.2014
http://www.reprosol.be/sites/reprosol.beta.the-aim.be/files/responsive-webdesign%28ethan-marcotte%29.pdf

Sublimetext. 2014. Luettu 3.12.2014. http://www.sublimetext.com/

Wärn, F. 2010. Käyttäjäkeskeisen suunnittelun ja ketterien menetelmien yhdistäminen prosessimallien näkökulmasta. Julkaistu 2.12.2010. Luettu 2.12.2014. http://www.soberit.hut.fi/T-121/shared/thesis/kandityot/kandi_Joonas_Warn.pdf

Node.js, Github 2011. Luettu 4.12.2014
https://github.com/joyent/node/releases/tag/v0.0.1

HTML5, W3C 2014. Luettu 6.12.2014
http://www.w3schools.com/html/html5_intro.asp

Evan Hammer, Performance at Rest 2014. Luettu 12.12.2014
http://hueniverse.com/2014/08/20/performance-at-rest/

8 LIITTEET

Liite 1. Esimerkki Jade-syntaksista

```
html(lang='en' ng-app='roots')
  head
    base(href='#/')
    title demtrees
    meta(name='viewport' content='width=device-width, initial-scale=1.0')
    link(rel='stylesheet' href='public/css/styles.css')
    link(rel='stylesheet' href='public/css/font-awesome.min.css')


  body(ng-cloak ng-controller='AppInformations')
    .navbar(ng-controller='NavCtrl')
      .navbar-header.col-15
        a.navbar-brand(href='#/')
          img.navbar-logo(src='public/media/seed.svg')
      .navbar-nav.col-85
        ul.nav.col-70
          li(access-level='accessLevels.public' )
            a(ui-sref='public.home' ui-sref-active="active")
              span Home
          li(access-level='accessLevels.public' )
            a(ui-sref='public.about' ui-sref-active="active")
              span About
          li(access-level='accessLevels.public')
            a(ui-sref='public.contact' ui-sref-active="active")
```

Liite 2. Esimerkki Stylus-syntaksista.

```
.post-meta
  padding 25px

.list
  background #fff
  padding 15px
  .list-item
    padding 15px
    border-right 1px solid #e7e7e7
    border-left 1px solid #e7e7e7
    &.odd
    &:last-child
      border-bottom 1px solid #e7e7e7
    &:first-child
      border-top 1px solid #e7e7e7
    &.even
      background #f7f7f7
```

Liite 3. Valmiin sovelluksen etusivunäkymä