Lauri Aaltonen

Verkkosivuston kehittäminen ketterästi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

23.4.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Lauri Aaltonen Verkkosivuston kehittäminen ketterästi 41 sivua + 3 liitettä 23.4.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Yliopettaja Kari Aaltonen

Insinöörityön tarkoitus oli tutkia ketterän kehityksen soveltamista verkkosivustojen kehittämiseen. Asiakkaan kannalta tavoitteena oli rakentaa sivusto, joka toimisi pääasiallisena kanavana yrityksen markkinointiin, ja mahdollistaa sosiaalisen median aktivointi sivuston kautta.

Sivustoprosessien suurimmat haasteet liittyvät ohjelmistotuotannon malleihin, jotka eivät mukaudu mahdollisiin muutoksiin projektin kuluessa. Insinöörityössä tutkittiin, mitä ketterä kehitys tarkoittaa, mitä sovellusmalleja se sisältää ja minkälaisia piirteitä on Scrum-tavassa toteuttaa ketterää kehitystä. Työn tarkoitus oli selvittää, millä tavoin ketterää kehitysmallia voidaan soveltaa käytännön verkkosivusto projektissa.

Insinöörityön lopputuloksena asiakkaalle toteutettiin Drupal-pohjainen verkkosivusto. Drupalin käyttäminen sivuston alustana oli kehityksen kannalta järkevää aikaisemman Drupal-osaamisen perusteella. Sivuston jatkokehitys on helppoa Drupalin laajennettavuuden vuoksi. Scrumin käyttäminen projektinhallintametodina mahdollisti sivuston rakentamisen pienissä osissa, ja asiakas koki saavansa lisäarvoa sen käytöstä.

Projektin läpivienti ketterästi oli tehokasta, ja sivuston jatkokehitys toteutetaan samojen periaatteiden mukaisesti.

Avainsanat	Ketterät kehitysmenetelmät, SCRUM, Drupal

Author Title Number of Pages Date	Lauri Aaltonen Agile website development 41 pages + 3 appendices 23 April 2014
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Kari Aaltonen, Principal Lecturer

The purpose of this bachelor's thesis was to examine the usage of agile development methods in web development process. On customer's behalf the target was to build a website which would function as a marketing channel for the customer and enable activation of social media channels.

The major challenge in website development process are tied to software process models which are not suitable for alterations during the development process. This thesis examines agile development in general, agile development models and characteristics of Scrum. The purpose is to clarify how agile development methods can be applied to website development process.

As a result of this thesis a Drupal based website was built for the customer. Usage of Drupal as a base was practical because of previous experience with Drupal. Further development will be simple as Drupal can be easily extended. Scrum as a project management method enabled development in fragments and the customer felt value-added by Scrum.

The agile project model was effective and further development will be accomplished with the same principles.

Keywords	Agile Project Management, SCRUM, Drupal

Sisällys

Lyhenteet ja käsitteet

Scrum	Ketterän kehityksen malli, jossa korostetaan tuotantoryhmän yhtenäisyyttä ja muutoksiin sopeutumista. Scrum-termi on peräisin rubgystä, ja se tarkoittaa tiivistä, kahdeksan hengen muodostelmaa.
CSS	Cascading Style Sheets eli tyylimäärittely. CSS:n perusteella selain piirtää sivuston ulkoasun.
CMS	Content Management System eli sisällönhallintajärjestelmä. Sivustojen pohjalla käytetty alusta, joka helpottaa sisällön lisäämisen, muokkaamisen ja poistamisen
Kanban	Tapa toteuttaa ketterää kehitystä. Kanbanissa ominaisuuslista priorisoidaan ja työt aloitetaan aina listan tärkeimmästä ominaisuudesta.
XP	eXtreme Programming. Ketterän kehityksen malli, jossa korostetaan tuotantotapojen toimivuutta ja hyväksi havaittuja käytäntöjä.
Crystal	Ketterän kehityksen malli, joka sisältää monta erilaista projektihallinnan viitekehystä, kuten Crystal Clear, Crystal Orange ja Crystal Red. Crystal-metodit keskittyvät ihmisiin, vuorovaikutukseen, yhteisöllisyyteen, taitoihin, lahjakkuuksiin ja kommunikointiin.
DSDM	Dynamic Systems Development Method. Ketterän kehityksen malli, jossa ominaisuuslista priorisoidaan MoSCoW-periaatteen mukaisesti.
MoSCoW	Priorisointitapa, jossa korkein prioriteetti on pakollisilla toiminnoilla ja viimeisenä ovat asiat, jotka eivät tuota juurikaan arvoa projektille. MoSCoW on lyhenne sanoista Must, Should, Could ja Want.
FDD	Ketterän kehityksen malli, jossa korostetaan toiminnollisuutta ja priorisoidaan toteutettavat asiat tärkeimmän toiminnollisuuden mukaan.

1 Johdanto

Insinöörityössä tutkitaan ketterien kehitysmallien soveltumista ja soveltamista käytännön verkkosivustoprojekteihin. Työssä selvitetään erilaisten ohjelmistomallien piirteet ja sivustoprojektien ominaisuudet ja vaatimukset sekä syvennytään Scrum-lähestymistapaan soveltaa ketteriä kehitysmalleja. Työ pyrkii vastaamaan kysymykseen: Tuoko ketterien kehitysmallien soveltaminen lisäarvoa asiakasprojekteihin?

Nykymaailmassa verkkonäkyvyyttä voidaan pitää elinehtona suurimmalle osalle yrityksistä. Verkon kautta tapahtuvaan markkinointiin törmää jatkuvasti esimerkiksi sosiaalisessa mediassa, hakukoneiden hakutuloksissa ja erilaisilla sivustoilla bannereiden muodossa. Verkkosivuston, johon markkinointi yleensä johdetaan, on toimittava tehokkaasti ja yrityksen brändin tukena.

Verkkosivustoja tarjoavia yrityksiä on paljon, mutta vain kourallinen yrityksiä tarjoaa sivustokehitystä ketterästi. Insinöörityön tarkoituksena on luoda verkkosivusto Metropolia Ammattikorkeakoulussa opiskelevan oppilaan Sleeping Panda Films -nimiselle startup -yritykselle noudattaen ketteriä kehitysmalleja. Verkkosivusto koostetaan suunnittelusta julkaisuun asti, ja työssä käydään läpi vaiheittain, miten verkkosivusto rakennetaan. Verkkosivusto toteutetaan Drupal-sisällönhallintajärjestelmällä, ja työssä tutkitaan Drupal-verkkosivustoprojekteihin liittyviä ominaisuuksia.

2 Ohjelmistokehitys ja sivustoprojektit

2.1 Ohjelmistokehityksen pääpiirteet

Sivustoprojektit eroavat ohjelmistokehityksestä jonkin verran, mutta eivät merkittävästi: samoja periaatteita ja kehitysmalleja noudatetaan yleisesti niin ohjelmistokehityksessä kuin sivustoprojekteissa. Sivustoprojektien työnkulusta kerrotaan lisää luvussa 2.3.

Ohjelmistokehityksen eri vaiheet ovat pääpiirteittäin samanlaisia jokaisessa ohjelmistokehitysmallissa. Vaiheiden suorittaminen on riippuvaista käytetystä ohjelmistokehitysmallista, ja esimerkiksi ketterän kehitysmallin mukaiset projektit eroavat laajalti esimer-

kiksi vesiputousmallin mukaisista projekteista. Ohjelmistokehityksen vaiheet ovat valmistelu, määrittely, suunnittelu, rakentaminen, testaaminen ja julkaisu. [1.]

Valmistelu

Valmistelun tarkoituksena on selvittää kaikki projektiin vaikuttavat tekijät ja selvittää määrittelyvaiheeseen kirjattavat asiat. Valmisteluvaiheessa selvitetään myös, onko projektin läpivienti kannattavaa eli mikä on riskien suhde onnistumisen mahdollisuuteen. Asiakkaalle arvoa tuovat laadulliset ja tekniset vaatimukset selvitetään ja määritellään vaatimusmäärittelyssä. Yleensä valmisteluvaiheessa ovat mukana työn toteuttajan lisäksi asiakasyrityksen eri osastot, jotka vaikuttavat projektin kulkuun. [1.]

Määrittely

Projektin valmistelun perusteella laaditaan vaatimusmäärittely. Se sisältää selkeän listauksen projektin eri osa-alueista: mitä projekti sisältää ja mitä se ei sisällä. Listaus toimitetaan asiakkaalle hyväksyttäväksi.

Vaatimusmäärittely voidaan kiteyttää kolmeen kysymykseen: Mikä on ratkaistava ongelma? Miksi se pitää ratkaista? Kenen vastuulla ongelman ratkaisu on? Määrittelyn perimmäinen tarkoitus on tuottaa dokumentti, jossa muotoillaan edellä mainitut kysymykset ymmärrettävään ja loogiseen muotoon. Mikäli vastaukset eivät anna selkeää kuvaa projektin tavoitteista, on projektin toteuttaminen kyseenalaista. [1.]

Suunnittelu

Vaatimusmäärittelyn perusteella luodaan suunnitelma projektin läpiviemiseksi. Suunnitelmasta tulee käydä ilmi muun muassa ohjelmistorakenne eli arkkitehtuuri, kolmansien osapuolien tarjoamat palvelut tai ohjelmistot, käytetyt moduulit ja selkeä kaavio tiedon prosessoinnista. Suunnitelmia voidaan luoda useita ja eri näkökulmista, jolloin asiakasorganisaatiolle jää arvioitavaksi, mikä lähestymistapa sopii ominaisuuksiltaan parhaiten projektiin. Arvioon vaikuttavia ominaisuuksia ovat muun muassa jatkokehitysmahdollisuudet, kompleksisuus, hinta eli budjetti sekä aikataulu. [1, s. 2.]

Rakentaminen

Rakentamisvaiheessa päätetään, mitä teknologioita käytetään vaadittujen ominaisuuksien rakentamiseen. Esimerkiksi jos projektin tarkoitus on parantaa vanhan ohjelmiston käytettävyyttä tai rakentaa ohjelmisto uudelleen, voidaan samoja komponentteja käyttää uudessa projektissa. Sivustoprojekteissa päätettäviä asioita ovat esimerkiksi käytetty kieli, alusta ja palvelin. Yleisesti yritykset erikoistuvat muutamiin teknologioihin ja tarjoavat oletuksena niitä kaikille asiakkaille. [1, s. 2.]

Testaaminen

Testaamisvaiheessa projektin eri osa-alueet testataan ja löydetyt ongelmat korjataan. Rakentamisvaiheessa suoritetaan myös jatkuvaa testausta, mutta testausvaiheen perimmäinen tarkoitus on varmistaa, että ohjelmisto on valmis julkaistavaksi ja täyttää vaaditut määritelmät. Testaus voidaan myös toteuttaa pienellä testiryhmällä, jolloin saadaan tietoa käyttökokemuksesta asiakasrajapinnasta. Edellä mainitussa esimerkissä tuote julkaistaan erikseen testiryhmälle. [1, s. 2.]

Julkaisu

Tuote tai palvelu julkaistaan asiakkaan määrittelemällä tavalla. Sivustoprojektien jälkeen projektin toteuttaja yleensä antaa asiakkaalle oikeudet sivuston ylläpitoon tai jatkaa itse ylläpitäjänä. [1, s. 2.]

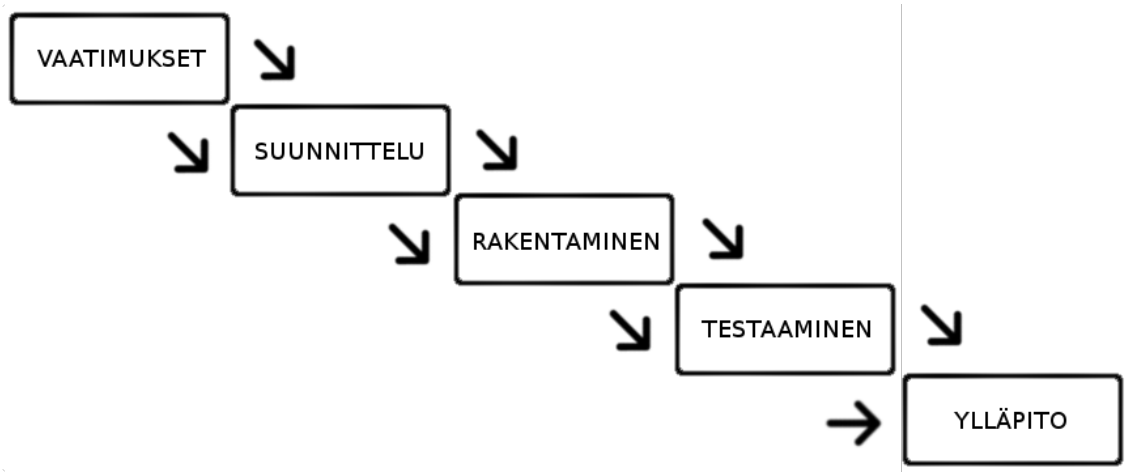
2.2 Erilaiset ohjelmistotuotannon kehitysmallit

Ohjelmistotuotannon vaiheiden suorittamiseen on kehitetty erilaisia malleja. Näitä malleja ovat ketterän kehitysmallin lisäksi esimerkiksi vesiputousmalli, iteratiivinen malli ja spiraalimalli. [1, s. 3.]

Vesiputousmalli

Vesiputousmallia pidetään yhtenä ensimmäisistä ohjelmistokehityksen malleista. Luvussa 2.1 määritellyt vaiheet suoritetaan toistensa jälkeen, jolloin ne ovat lineaarisesti järjestyksessä. Vaiheet suoritetaan loppuun ennen seuraavaan siirtymistä. [1, s. 4–6.]

Kuvasta 1 nähdään, miten vesiputousmallin eri vaiheet muodostavat laskevan kuvion, josta mallin nimi on peräisin.



Kuva 1. Vesiputousmallin eri vaiheet [1, s. 4–6].

Vesiputousmallin hyviä puolia ovat muun muassa käytön helppous ja ymmärrettävyys. Jokainen vaihe sisältää tarkasti määritellyt osuudet, ja ne toteutetaan ennen seuraavaan vaiheeseen siirtymistä. Jokaisen vaiheen jälkeen käydään läpi, mitä on tehty ja mitä seuraavaksi tehdään. Tämä helpottaa vaiheiden aikatauluttamista, jolloin projektin eri vaiheille voidaan määritellä tarkasti päättymisajankohta. Vesiputousmallin mukaisesti läpiviedyistä projekteista jää yleensä kattava dokumentaatio, josta selviävät tehdyt toimenpiteet ja tulokset.

Vesiputousmallin mukaisissa projekteissa on aina riski lopputuotteen toimivuudessa asiakkaan tarpeisiin nähden. Projekti julkaistaan kehityksen loppuvaiheessa, jolloin varsinkin pitkien projektien vaatimukset ovat voineet muuttua, projektissa valmistunut tuote ei ole enää ajankohtainen tai asiakkaan maksukapasiteetti on muuttunut. Vesiputousmallin tehokkuus konkretisoituu lyhyemmissä projekteissa, joissa vaatimukset ovat selkeät ja eri vaiheiden sisältö pystytään määrittelemään tarkasti. [1, s. 4–6.]

Iteratiivinen malli

Iteratiivisen mallin keskeinen teema on lopputuotteen kehitys vaiheittain, josta mallin nimi on peräisin. Iteratiivisessa mallissa kehitysvaiheet seuraavat toisiaan, mutta ne eivät pohjaudu tarkasti määriteltyihin vaatimuksiin. Kehitysvaiheessa rakennetaan osa toiminnollisuuksista, minkä jälkeen tehdyt toimenpiteet ja tulokset käydään läpi ja aloi-

tetaan uusi kehitysvaihe. Iteratiivisessa mallissa lopputuotetta kehitetään versioittain. Jokaisessa kehitysvaiheessa suunnitellaan, rakennetaan, testataan ja julkaistaan tehty toiminnollisuus osaksi tulevaa lopputuotetta. [1, s. 7–9.]

Iteratiivisen mallin selkein etu on kehitysvaiheiden jaksottaminen. Näin toiminnollisuuksia voidaan rakentaa samaan aikaan, jolloin uusia toiminallisuuksia voidaan julkaista nopealla aikataululla. Lopputuotteen alusta julkaistaan aikaisessa vaiheessa, jolloin voidaan todeta sen toimivuus ja mahdolliset ongelmat suunnittelussa tai toteutuksessa heti projektin alkuvaiheessa.

Selkeitä heikkouksia iteratiivisessa mallissa ovat lopputuotteen tarkoituksen muuttuminen ja projektin aikataulujen epävarmuus. Kehitystä tehdään vaiheittain, jolloin vaatimukset voivat muuttua suuntaan, joka ei enää palvele lopputuotetta tarkoituksenmukaisella tavalla. Vaatimusten muuttuessa on myös mahdollista, että osaa rakennetuista toiminnollisuuksista joudutaan muokkaamaan uusien vaatimusten mukaisesti tai rakentamaan kokonaan uudelleen. Tuotteen jatkuva kehittäminen vaikeuttaa myös aikataulujen, kuten lopullisen tuotteen julkaisun ajankohdan, määrittelyä. Iteratiivisen mallin onnistumisen kannalta riskejä tulee analysoida tarkasti. [1, s. 7–9.]

Spiraalimalli

Spiraalimalli on yhdistelmä iteratiivisesta ja vesiputousmallista. Vesiputousmallin lineaarisuus yhdistettynä iteratiivisen mallin kehitysvaiheisiin edesauttaa vaatimuksiin sopeutumista. Spiraalimallin keskeinen etu on riskien analyysi ja hallinta, jolloin lopputuotteen arvo asiakkaalle pysyy hyvänä ja projektin riskejä voidaan hallita.

Vaatimuksiin ja muutoksiin sopeutuminen on yksi spiraalimallin vahvuuksista. Projektiin voidaan lisätä toiminnollisuuksia, ja projektin vaatimuksien muutokset eivät aiheuta ristiriitaa aiempien vaatimusten tai suunnittelun kanssa. Spiraalimalli on edellä mainituista malleista lähimpänä ketterää kehitystä. [1, s. 10–12.]

2.3 Sivustoprojektien pääpiirteet

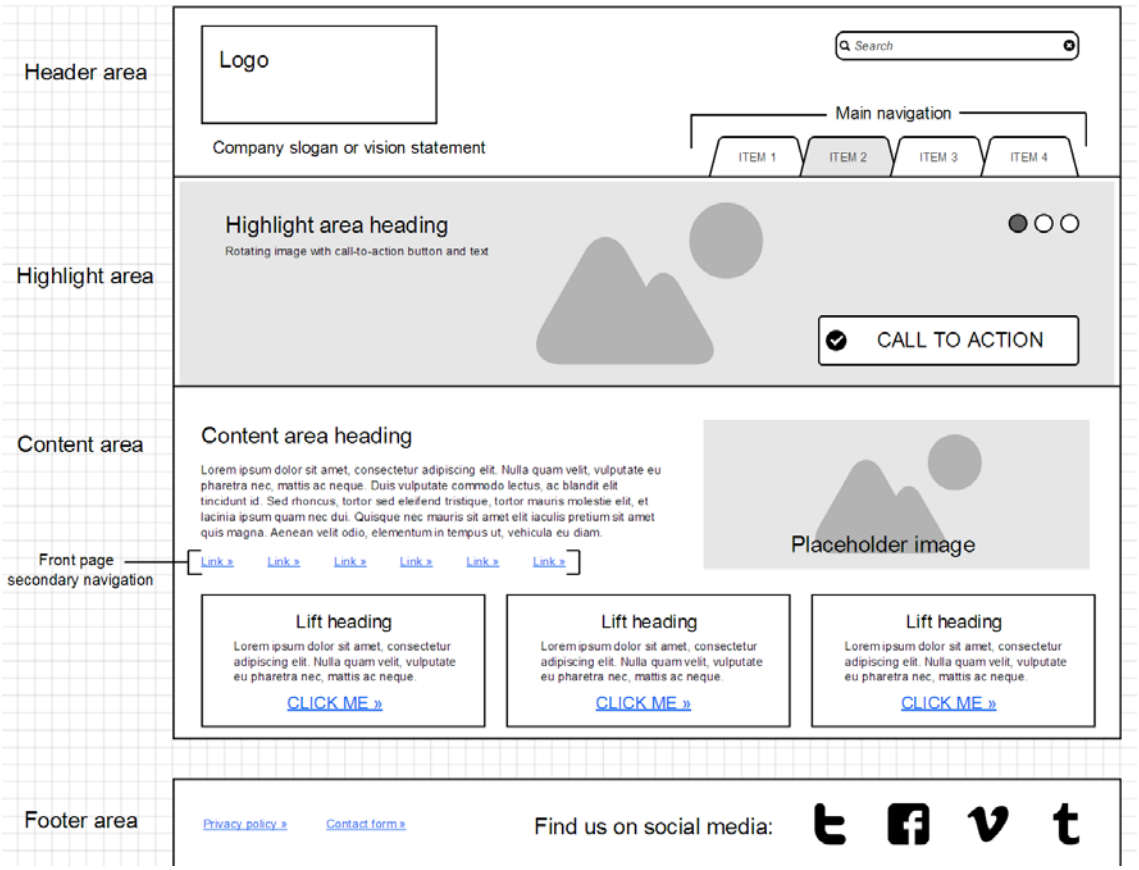
Sivustoprojektien kokonaisuuden ymmärtäminen on olennainen osa ketterää sivustokehitystä [4]. Kuten ohjelmistokehitys, sivustokehitys noudattelee vakiintuneita malleja, työmenetelmiä ja -vaiheita. Nämä vaiheet perustuvat ohjelmistokehityksen vaiheisiin, mutta tehtävät toimenpiteet ovat sivustoprojekteille ominaisia. Ketterä kehitysmalli muuttaa olennaisesti vaiheiden läpivientiä, jolloin on tärkeää tietää perinteisen sivustokehityksen työvaiheet. Perinteisen sivustokehityksen työvaiheet ovat suunnittelu, rakentaminen ja julkaiseminen. [3.]

Suunnittelu

Suunnitteluvaiheen perimmäinen tarkoitus on selvittää asiakkaan liiketoimintaa parantavat elementit ja rakentaa niistä toimiva konsepti, jonka konkretisoituma on verkkosivuston ulkoasu. Alakohtaiset normit voivat vaikuttaa muun muassa graafiseen suunnitteluun, jolloin sivusto peilaa asiakkaan roolia oman alansa asiantuntijana. Yrityksen asiakassegmenttiä profiloimalla voidaan sivuston ulkoasu suunnitella käyttäjälähtöisesti, jolloin sivuston markkinoinnillinen tavoite on helpompi saavuttaa. Yleensä projektin toteuttaja tutkii alan graafista suuntautumista ja käyttää tutkimuksen tietoja suunnitteluvaiheen läpiviennissä.

Vaatimusmäärittely toimii suunnitteluvaiheen tukena, ja siitä ilmenevät haasteet, ongelmat ja ehdotettu ratkaisu. Yritykselle verkkosivusto voi toimia esimerkiksi käyntikorttina, tiedonjakelualustana tai markkinointikanavana riippuen yrityksen tarpeesta.

Käyttöliittymäsuunnittelu tukee sivuston graafisen ulkoasun suunnittelua. Rautalankamallin rakentaminen on sivuston käyttöliittymän kannalta olennainen osa, jossa otetaan huomioon esimerkiksi erilaisten navigaatioelementtien, call-to-action-elementtien ja sisällön sijoittuminen sivuille, kuten kuvasta 2 voidaan nähdä. Projektista riippuen rautalankamallin rakentaminen ei ole välttämätöntä, mutta se helpottaa graafista suunnittelua ja antaa asiakkaalle käsityksen, millä tavoin sivuston keskeiset elementit sijoittuvat sivulle. [4.]



Kuva 2. Esimerkki rautalankamallista, jossa on esitelty etusivun keskeiset alueet ja niiden elementit.

Asiakas tekee verkkosivuston sisällön tai se ostetaan sisällöntuottajalta. Sisältö on tärkein elementti muun muassa hakukonenäkyvyyden kannalta, joten sen (sisällön) tuottamiseen kannattaa varata aikaa. Projektista riippuen voidaan osa sisällöstä syöttää järjestelmään myös jälkikäteen, jolloin sivusto voidaan julkaista aikaisemmin esimerkiksi tilastotiedon keräystä varten. Suunnitteluvaiheessa käytetään yleisesti niin sanottua ladontatekstiä eli placeholder-tekstiä, jonka tarkoituksena on määrittää, millä tavoin teksti latoutuu sivulle ja sivuston eri kohtiin.

Suunnitteluvaihe voidaan katsoa päättyneeksi, kun asiakas on tyytyväinen sivuston rakenteeseen ja käyttöliittymään eli ulkoasuun. [4.]

Rakentaminen

Rakentaminen aloitetaan valitsemalla alusta, jonka päälle sivusto rakennetaan. Yleinen käytäntö on käyttää sisällönhallintajärjestelmää, joka auttaa asiakasta sisällön hallitsemisessa. Asiakkaalla voi ennestään olla kokemusta erilaisista sisällönhallintajärjestelmistä, jolloin häntä (asiakasta) ei välttämättä tarvitse erikseen kouluttaa käyttämään sivuston peruselementtejä. Joissain projekteissa asiakas syöttää itse sisällön sivuille.

Rakentaminen itsessään on hyvin suoraviivainen prosessi: Tuotetaan sivusto, joka vastaa asiakkaalle toimitettua ulkoasua. Kun sivusto on pääpiirteittäin valmis, tuotetaan ohjeisto, jonka perusteella asiakas kykenee käyttämään sivustoa. Kuten edellisessä kappaleessa todettiin, joidenkin asiakkaiden kohdalla ohjeiston tarvitsee kattaa vain ne toiminnollisuudet, joita asiakas ei osaa käyttää. Ennen varsinaista julkaisua sivusto testataan mahdollisten yhteensopivuus- ja käytettävyysongelmien minimoimiseksi. [4.]

Julkaiseminen

Sivusto siirretään asiakkaan palvelimelle. Mahdollisimman lyhyen toimimattomuusjakson turvaamiseksi uusi sivusto aktivoidaan ajankohtana, jolloin sivustolla on mahdollisimman vähän käyttäjiä. [3.] Tämän jälkeen sivustoa tarkkaillaan mahdollisten virheiden ja toimimattomuuden kannalta [4]. Projektista riippuen sivuston toteuttajan ja asiakkaan yhteistyö jatkunee sivuston julkaisun jälkeen esimerkiksi ylläpitotoimenpiteiden merkeissä [3].

3 Ketterät kehitysmallit

3.1 Mallit

Ketterien kehitysmallien sovelluskehyksiä on monia. Tämä työ keskittyy Scrum-lähestymistapaan toteuttaa ketterää kehitystä. Siinä korostetaan projektihallinnan periaatteita ja tuotantoryhmän vapautta valita omat tuotantometodinsa. [6.]

Muiden sovelluskehyksien tunteminen on hyödyllistä ajatellen erilaisten projektien luonteita, koska kasvavilla markkinoilla ainoa pysyvä asia on muutos ja erilaiset kehitysmallit ovat ratkaisu muutoksen hallintaan. [12.]

Kanban

Kanban-sovelluskehys pyrkii rajoittamaan toiminnollisuuksien rakentamisen yksittäisiin toiminnollisuuksiin. Scrum-lähestymistavassa jokaiseen kehitysjaksoon on varattu tietty aika, jossa valitut toiminnollisuudet rakennetaan. Kanbanissa pyritään ottamaan uusia toiminnollisuuksia työlistaan aina tilaisuuden salliessa. Kehitysjaksot pysyvät samanpituisina kuin Scrumissa, mutta jokaisen jakson jälkeen toimitetaan ne toiminnollisuudet, jotka ovat valmiina. [7.]

Kanban perustuu kolmeen periaatteeseen:

- työmäärän visualisointi kokonaisuuden ja päivän tehtävien osalta
- prosessien rajoittaminen työmäärän osalta ja julkaistavien elementtien tasapainottaminen projektin kulkuun nähden
- työtehtävien tekeminen priorisoidun listan mukaan [7].

Kanban ei sinällään ole puhdasverinen sovelluskehys, mutta se auttaa varsinkin aloittelevia yrityksiä tuomaan ketterän kehitysmallin hyödyt konkreettisemmalle tasolle [8].

Extreme Programming

Extreme programming (XP) keskittyy tuotannollisempaan lähestymistapaan kuin Scrum. Työvaiheita XP:ssa on neljä: suunnittelu, kuuntelu, testaaminen ja rakentaminen eli koodaus. Jokainen kehitysjakso sisältää edellä mainitut vaiheet, ja kuten Scrumissa jokaisen vaiheen jälkeen julkaistaan toimiva lisä ohjelmaan. Erona Scrum-lähestymistapaan on muun muassa se, että kehitysjakson aikana voidaan rakennettavia toiminnollisuuksia vaihtaa, mikäli sen (toiminnollisuuden) rakentamista ei ole vielä aloitettu. XP:ssa työlistaa käydään läpi järjestelmällisesti prioriteetin mukaan, kun Scrum taas painottuu niihin toiminnollisuuksiin, jotka tuotantoryhmä on valinnut kehitysjaksoon.

Scrum ei määrittele parhaita käytäntöjä toimintojen suorittamiseen, kun XP taas pyrkii määriteltyihin, parhaiksi havaittuihin käytäntöihin. Yksi XP:n yksi periaatteista on noudatella erilaisia tuotantotapoja, kuten test-driven-development, jonka tarkoituksena on kehittää toiminnollisuuksia rakentamalla ensin testit, jotka toiminnollisuuden tulee läpäistä ollakseen julkaistavissa. [9.]

Crystal

Crystal-kehitysmalli kattaa muun muassa Crystal clear-, Crystal yellow-, Crystal orange- ja monia muita metodeita vastaamaan erilaisten projektien vaatimuksiin [7]. Projektien vaatimukset ja projektiryhmän koko määrittelevät käytetyn Crystal-metodin. Huomioitavia asioita ovat esimerkiksi kommunikointitiheys, järjestelmän kriittisyys ja priorisointi. Jokainen metodi sisältää erilaisen määritelmän käytetyistä säännöistä, tuotantomenetelmistä ja -toimenpiteistä sekä -tavoista. [10.]

Crystal clear -metodi perustuu ihmisläheiseen lähestymistapaan jossa projektiryhmä voi soveltaa muita metodeja tarvittaessa. Crystal Clear -metodia käytetään yleisesti pienissä projekteissa, joiden lopputuote ei ole kriittinen esimerkiksi yrityksen toiminnan kannalta. Muut Crystal-metodit ottavat mukaan aspekteja muun muassa XP:sta, ja mitä kriittisemmästä projektista on kyse, sitä enemmän sovelletaan esimerkiksi test-driven-developmentia, joka ohjaa sovelluskehitystä lähes virheettömään lopputulokseen. [11.]

Dynamic Systems Development Method (DSDM)

DSDM on yleisluontoisempi lähestymistapa projekteihin kuin Scrum. Siinä missä Scrum keskittyy tuotantoryhmän ominaisuuksiin ja vapauksiin sekä kehitysjaksojen läpivientiin, DSDM sisältää perinteisempiä projektinhallinnan toimintoja, kuten tarkan dokumentoinnin tehdyistä toimenpiteistä. [12.] Käytännössä DSDM ei ole kovin kaukana Scrum-lähestymistavasta, mutta perinteisemmille yrityksille DSDM:n soveltaminen saattaa olla helpompaa kuin Scrumin.

DSDM perustuu MoSCoW-sääntöihin. Nämä säännöt jaottelevat rakennettavat toiminnaillisuudet seuraaviin ryhmiin:

- pakolliset toiminnollisuudet, joiden toteutuminen on kriittistä järjestelmän toimivuuden kannalta
- tärkeät toiminnollisuudet, jotka ovat olennaisia järjestelmän kannalta, mutta eivät vitaaleja järjestelmän toimivuudelle
- mahdolliset toiminnollisuudet, jotka parantavat järjestelmän käytettävyyttä mutta joita voi helposti siirtää myöhempiin kehitysjaksoihin
- halutut toiminnollisuudet, jotka vaikuttavat määriteltyyn käyttäjäryhmään ja joilla ei ole kokonaisuuden kannalta oleellista arvoa

DSDM-kehityksessä tärkeintä on rakentaa kriittisimmät toiminnollisuudet ensin, jolloin asiakkaalle voidaan toimittaan prototyyppi, joka määrittää järjestelmään lisättävän toiminnallisuuden. [13.]

Feature Driven Development (FDD)

Kuten mallin nimi antaa olettaa, FFD:ssä kehitys tehdään toiminnollisuuksien mukaan. FFD-mallissa on viisi erilaista kohtaa, jotka seuraavat toisiaan koko prosessin ja projektin ajan. Kehitysjaksot sisältävät kolme viimeistä kohtaa. Nämä kohdat ovat seuraavat:

- Rakennetaan kokonaiskuva projektista, vaatimuksista ja edellytyksistä, ja apuna käytetään luokkadiagrammeja, joista selviää, miten eri toiminnollisuudet linkittyvät keskenään.
- Rakennetaan toiminnollisuuslista, josta selviävät kaikki järjestelmän toiminnollisuudet; jokainen toiminnollisuus vastaa johonkin liiketoiminnan ongelmaan.
- Luodaan alustava suunnitelma toiminnon rakentamiseen ja määritellään sen toteuttaja tai toteuttajat toiminnollisuuden mukaan.
- Suunnitellaan toiminnot ja keinot toteuttaa ne.
- Rakennetaan toiminnollisuus.

Kun kehitysjakso on päättynyt, arvioidaan asiakkaalle tuotu arvo ja siirrytään seuraavaan kehitysjaksoon. Scrumiin verrattuna FFD keskittyy yhteen toiminnollisuuteen kerralla, kun taas Scrumissa kehitysjakso sisältää ennalta määrätyt toiminnollisuudet, jotka aiotaan kehitysjaksossa rakentaa. [14.]

3.2 Periaatteet

Ketterän kehityksen periaatteet ohjaavat interaktiiviseen työmalliin asiakkaan kanssa. Periaatteita ovat muun muassa yksilön ja vuorovaikutuksen arvostaminen, tuotteen tai palvelun toimivuuden takaaminen, vuorovaikutteisuus ja muutoksiin sopeutuminen. Näitä periaatteita korostetaan ketterässä kehityksessä. Muita periaatteita ovat muun muassa oikeiden työkalujen ja menetelmien soveltaminen ja käyttäminen, oikeaoppinen dokumentointi ja yhteisymmärrys projektin yksityiskohdista, joiden perusteella luodaan sopimus toimeksiantajan ja toimittajan välille.

Periaatteet muodostavat ketterän kehityksen perustan. Liiketoiminnallisesti tärkein – mitattavissa oleva – aspekti on asiakkaan tyytyväisyys projektiin ja lopputuotteeseen. Kehityksen aikana asiakkaan aktiivinen osallistuminen projektiin on olennaista ketterän kehityksen onnistumisen kannalta. Ketterässä projektissa kehitystyö tehdään tiiviissä yhteistyössä muun organisaation kanssa, mikä edesauttaa muutoksiin sopeutumista ja reagoimista. [15.]

Toimiva lopputuote on asiakkaalle tärkein mittari, jolla arvioidaan projektin onnistuminen. Ketterässä kehitysmallissa lopputuotetta kehitetään jatkuvasti, mutta esimerkiksi sivustoprojektien luonteesta riippuen se (sivusto) voidaan julkaista ennen kuin kaikki vaaditut toiminnollisuudet on rakennettu. Ketterän kehitysmallin periaatteen mukaisesti lopputuotteesta julkaistaan aika ajoin versio, jotta toimeksiantaja voi tarkistaa, että projekti on menossa sopimuksen mukaiseen suuntaan.

Yksi ketterän kehitysmallin periaatteista on luoda projektiryhmä, joka koostuu motivoituneista ja osaavista tekijöistä [15]. Projektiryhmän ja projektin muiden osapuolten kesken voidaan soveltaa syväjohtamisen kulmakiviä. Niitä ovat yksilöllinen kohtaaminen, inspiroiva tapa motivoida, luottamuksen rakentaminen ja älyllinen stimulointi. Syväjohtamisen tarkoitus on tuoda mainitut käyttäytymismallit esiin oman esimerkin avulla, joka on keskeinen teema syväjohtamisessa, kuten kuvasta 3 voidaan nähdä. Syväjohtamisen menetelmillä projektiryhmän jäsenet innostavat ja haastavat mutta samalla luottavat, arvostavat ja motivoivat toisiaan. [16.]

Kuva 3. Syväjohtamisen kulmakivet [16].

3.3 Roolit

Jokaisessa ketterässä projektissa määritellään henkilöt tiettyihin rooleihin. Esitellyt roolit perustuvat Scrum-lähestymistapaan toteuttaa ketterää kehitystä, kuten johdannossa mainittiin. Näitä rooleja ovat muun muassa asiakasjohtaja, scrum master ja agile mentor. [15.]

Rooleille ei ole virallisia suomennoksia, koska roolien englanninkieliset nimet ovat vakiintuneet myös suomalaiseen yritysmaailmaan. Insinöörityössä käytetään vapaita suomennoksia roolien nimistä.

Tuotantoryhmä eli production team

Tuotantoryhmä vastaa tuotteen tai palvelun kehityksestä. Ketterän kehityksen periaatteiden mukaisesti tuotantoryhmä pyrkii tuottamaan julkaisukelpoisen lisäyksen tai version tuotteeseen jokaisen kehitysjakson eli sprintin jälkeen.

Tuotantoryhmän koko on yleensä viidestä yhdeksään henkeä. Tyypillisiä ohjelmistokehityksen tai sivustokehityksen rooleja ei ole, jolloin pääpaino siirtyy yksittäisten prosessien sijaan yhteiseen tavoitteeseen, joka määritellään jokaiselle kehitysjaksolle erikseen. [15.]

Asiakasjohtaja eli product owner

Asiakas määrittelee omalta puoleltaan asiakasjohtajan, joka toimii välikätenä asiakasomistajien kanssa, asiakasyrityksen sisäisten ja sidosryhmien kanssa ja ensi kätenä varmistaa, että kehityksen suunta on asiakkaan näkökulmasta oikea. Asiakasjohtaja on yleensä oman alansa asiantuntija, jolla on tietämystä alansa markkinoinnista, käytännöistä, kilpailusta ja tulevan tuotteen oletetuista käyttäjistä. Asiakasjohtaja on tyypillisesti avainasiakasomistaja. [15.]

Asiakasjohtajan tehtävä on luoda ominaisuuslista ja priorisoida se liiketoiminnan kannalta arvokkaimpien ominaisuuksien mukaan [17] .

Scrum master

Ketterän projektin projektipäällikkö eli scrum master vastaa projektiryhmästä ja sen toiminnan edesauttamisesta. Projektipäällikkö sanana on harhaanjohtava, koska sananmukaista roolia ei Scrumissa ole vaan perinteisen projektipäällikön tehtävät jakautuvat yleensä tuotantoryhmän kesken.

Scrum masterin rooli on varmistaa, että projektiryhmä työskentelee ketterien kehitysmallien periaatteiden mukaisesti. Scrum master ohjaa ja valmentaa projektiryhmää ja mahdollistaa töiden optimaalisen suorittamisen poistamalla erilaisia häiriötekijöitä prosesseista.

Scrum masterin arvovalta kohdistuu prosesseihin. Hänen tehtävänsä ei ole muokata tai vaikuttaa yksittäisten projektiryhmän jäsenten työtapoihin tai tehtäviin, vaan muokata kokonaisuutta prosessina. Hän voi päättää esimerkiksi lyhentää tulevien kehitysjaksojen pituutta projektin vaatimusten kasvaessa. [15.]

Asiakasomistajat eli stakeholders

Asiakasomistajia ovat ne tahot, jotka eivät suoranaisesti ole vastuussa projektista, mutta antavat arvokasta palautetta ja projektin lopputulos vaikuttaa heidän toimintaansa. Asiakasomistajia voi olla yrityksen eri osastoista tai jopa eri yrityksistä. [15.]

Agile mentor

Agile mentorin rooli projekteissa on tarjota asiantuntevaa ja kokemukseen pohjautuvaa tietoa ketterästä kehityksestä. Agile mentor on käytännössä konsultti, joka tuo osaamisellaan lisäarvoa tuotantoryhmälle. [15.]

3.4 Välineet

Tuotevisio eli product vision statement

Tuotevisio kuvaa tavoitetta, joka projektin on tarkoitus saavuttaa. Se sisältää myös määritelmän liiketoiminnallisesta vaikutuksesta: miten lopputuote vaikuttaa yrityksen strategiaan ja kenen käyttöön lopputuote tulee. Tuotevision määrittelee asiakasjohtaja, joka on myös vastuussa sen toteutumisesta. Pitkissä projekteissa tuotevisiota voidaan päivittää tarpeen tullen, kuitenkin vähintään kerran vuodessa.

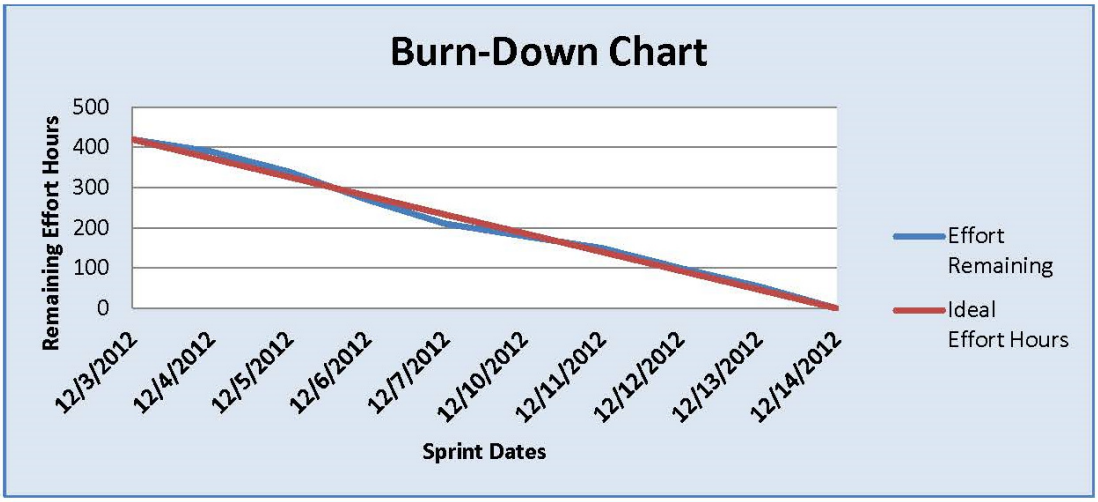
Ominaisuuslista eli product backlog

Ominaisuuslista sisältää ne asiat, joiden perusteella määritellään projektin laajuus. Lista jaetaan osiin prioriteetin tai aika- ja kustannusarvion mukaan, ja jokainen ominaisuus sisältää lyhyen kuvauksen, millä tavoin ominaisuus vaikuttaa projektiin. Näin lista saadaan jaettua kehitysjaksoihin eli sprintteihin. Ominaisuuslista on keskeinen tekijä ketterän kehityksen onnistumisessa. [15.]

Ominaisuuslistan koostetaan yleensä käyttötapauksien muodossa. Käyttötapaukset kuvaavat yksittäisen ominaisuuden käyttöä käyttäjän näkökulmasta. Listassa voi olla myös työtehtäviin liittyviä ominaisuuksia, kuten projektissa käytettävien komponenttien etsiminen. Lista elää projektin kuluessa, jolloin kehitysjaksoissa tehdyt toimenpiteet saattavat synnyttää muita tehtäviä, niitä voidaan jakaa vieläkin pienempiin osiin tai asiakas voi projektin edetessä määritellä lisäominaisuuksia. [18.]

Kehitysjakson jäljellä olevan työn kuvaaja eli sprint burn down chart

Kehitysjakson jäljellä olevasta työstä koostetaan graafi, josta nähdään, onko kehitysjakso edennyt halutulla tavalla. Kuvassa 4 nähdään esimerkki kuvaajasta, jossa x-akselilla on kulunut aika ja y-akselilla jäljellä oleva kehitysaika laskettuna tunteina. Graafista näkee, mikäli kehitysjaksoon voidaan ottaa ylimääräisen toiminnollisuuden kehittäminen tai vaihtoehtoisesti toiminnollisuuden vähentäminen jaksosta. [19.]



Kuva 4. Kehitysjakson jäljellä olevan ajan kuvaaja [19].

Tuotteen kehityskaari eli product roadmap

Tuotteen kehityskaari ilmaisee karkeasti projektin aikataulun keskeiset tavoitteet aikajanalla. Se on arvio, joka elää projektin edetessä, mutta antaa yleiskuvan projektin etapeista. [20.] Projektiryhmälle kehityskaari ilmaisee tulevia toimenpiteitä, joiden perusteella voidaan esimerkiksi valita käytetyt työkalut niin, että niitä voidaan hyödyntää tulevissa kehitysjaksoissa [15].

Julkaisusuunnitelma eli release plan

Julkaisusuunnitelma muodostetaan suunnitellun kehityskaaren, ominaisuuslistan ja kehitysjaksojen aikataulujen perusteella. Toisin kuin kehityskaari, julkaisusuunnitelma sisältää yksityiskohtaisemman ja aikatauluihin sidotun dokumentin projektin eri etapeista. Ketterän kehityksen mukaisesti julkaisusuunnitelmaan tehdään muutoksia projektin

edetessä, mutta se (julkaisusuunnitelma) antaa asiakkaalle selkeän ja tavoitteellisen arvion projektin kulusta.

Kehitysjakson suunnitelma eli sprint backlog

Kehitysjakson suunnitelma sisältää ne tehtävät ja toiminnot, jotka tuotantoryhmän on tarkoitus suorittaa kehitysjakson aikana. Suunnitelmaan kirjataan tehtävän prioriteetti ja arvio tehtävän suorittamiseen käytettävästä ajasta. Suunnitelmaa päivitetään aina, kun uutta tietoa tehtävästä on saatavilla, kuitenkin vähintään kerran päivässä. [15.]

Kuvassa 5 on esitelty esimerkki kehitysjakson suunnitelmasta. Suunnitelmat laaditaan aina projektikohtaisesti, ja tuotantoryhmällä on suuri vaikutus siihen, millä tavoin tehtävät kirjataan suunnitelmaan. [21.]

User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information.	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests ...	12	12	10	6		
	Code the ...	8	8	8	4		

Kuva 5. Kehitysjakson suunnitelman esimerkki [21].

Kasvu eli Increment

Jokaisen kehitysjakson alussa tuotantoryhmä arvioi, mitkä toiminnollisuudet voidaan kehitysjakson aikana toteuttaa. Kehitysjakson tavoite on lisätä toiminnallisuuksia lopputuotteen. [15.]

3.5 Tapahtumat

Projektisuunnittelu eli project planning

Projektin alussa käydään läpi projektin eri osa-alueet pääpiirteittäin. Suunnittelun tarkoitus on määritellä tuotevisio ja tuotteen kehityskaari, ja suunnitteluun voidaan varata esimerkiksi yksi päivä. [15.]

Julkaisusuunnitelma eli release planning

Julkaisusuunnittelussa otetaan huomioon ominaisuuslistassa määritellyt tehtävät ja tuotantoryhmän tehokkuus esimerkiksi edellisten projektien perusteella. Julkaisusuunnittelun tarkoitus on tuottaa pääpiirteinen julkaisusuunnitelma, johon on määritelty tai arvioitu projektinpäättymispäivä ja sen jakautuminen kehitysjaksoihin.

Kehitysjakso eli sprint

Kehitysjakso on yleensä 1–4 viikon pituinen jakso, jonka aikana tuotantoryhmä pyrkii tuottamaan toimivan lisäyksen lopputuotteeseen. Projektista riippuen kehitysjakso voi olla myös huomattavasti lyhyempi, mutta ei kovinkaan paljon pidempi.

Kehitysjakson suunnittelu eli sprint planning

Ennen jokaista kehitysjaksoa tuotantoryhmä, scrum master ja asiakasjohtaja suunnittelevat tulevan kehitysjakson sisällön. Suunnittelun aluksi asiakasjohtaja tekee listauksen ominaisuuslistan perusteella ja esittelee kehitysjakson aikana rakennettavat ominaisuudet. Tuotantoryhmä pyrkii tämän jälkeen selvittämään tarkasti kaikkien tehtävien ominaisuuksien tiedot. Kuten kohdassa ”ominaisuuslista” todettiin, tiedot ovat yleensä käyttäjätarinoiden muodossa. Kehitysjakson suunnittelun tarkoitus on tuottaa tarkasti dokumentoitu ja kuvattu listaus tehtävistä toiminnallisuuksista. Kehitysjakson suunnittelu tuottaa nimensä mukaisen dokumentin eli kehitysjakson suunnitelman. [22.]

Päivittäinen tapaaminen eli daily scrum

Kehitysjaksojen aikana tuotantoryhmä ja scrum master tapaavat päivittäin. Päivittäisen tapaamisen tarkoituksena on selvittää tehdyt toimenpiteet, päivän tavoitteet ja mahdolliset työnteon esteet. Tapaamisen tarkoitus ei ole selvittää ongelmia vaan antaa kuva projektin kulusta. Päivittäisten tapaamisten perusteella scrum master päivittää kehitysjakson jäljellä olevan työn kuvaajan. [23.]

Kehitysjakson läpikäynti eli sprint review

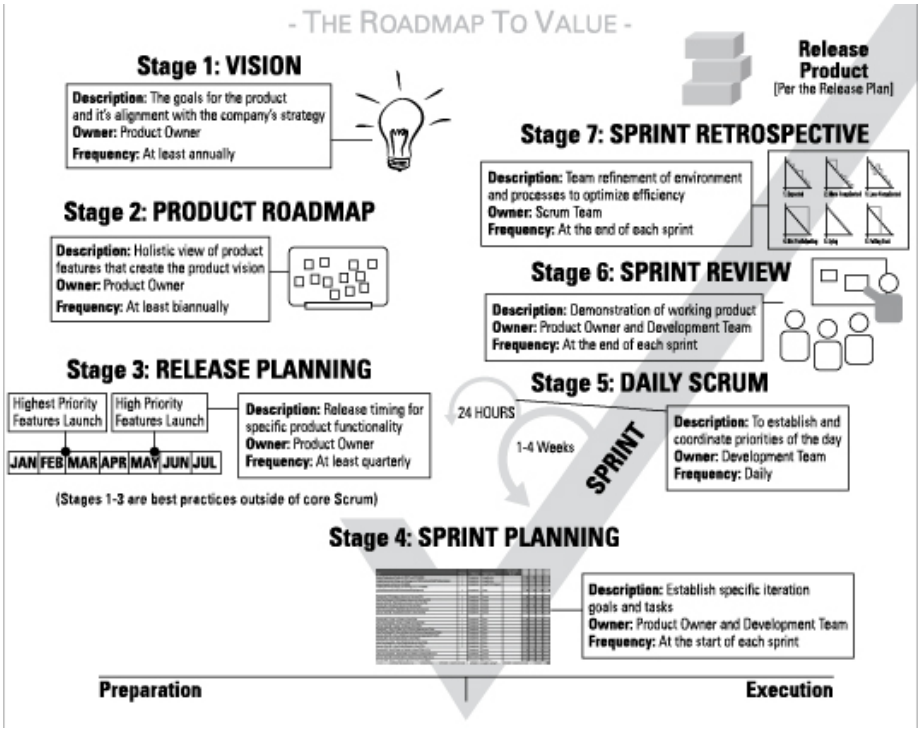
Kehitysjakson jälkeen tuotantoryhmä esittelee tehdyn lisäyksen lopputuotteeseen. Läpikäynnin tulisi olla luonnollinen tapa päättää kehitysjakso, ja yleisesti läpikäynti suoritetaan epävirallisessa kokouksessa. [15.]

Kehityspalaveri eli sprint retrospective

Kehityspalaveri pidetään yleensä kehitysjakson lopussa. Sen tarkoitus on selvittää, mitä asioita tuotantoryhmä voi kehittää omassa toiminnassaan. On monia erilaisia tapoja toteuttaa läpikäynti. Scrum master voi esimerkiksi kysellä yksittäisiltä tuotantoryhmäläisiltä projektiin vaikuttavista asioista, tai vaihtoehtoisesti jokainen voi listata asioita, mihin tulisi kiinnittää huomiota. Jokaisen kehitysjakson alussa edellä mainitusta listasta poimitaan asioita, joihin kiinnitetään huomiota tulevassa kehitysjaksossa. [15.]

3.6 Vaiheet

Ketterän kehityksen projektit noudattelevat pitkälti kuvan 6 mukaista kaavaa. On tärkeää tietää, termien ja tapahtumien lisäksi, missä järjestyksessä vaiheet suoritetaan.



Kuva 6. Ketterän kehityksen vaiheet [15].

Ketterässä kehityksessä kaikki vaiheet pyritään suorittamaan tehokkaasti. Mitä nopeammin projekti etenee kehitysjaksojen aloittamiseen, sitä nopeammin tuloksia alkaa kertyä. Ketterän kehityksen vaiheet ovat seuraavat:

- Asiakasjohtaja määrittelee tuotevision.
- Asiakasjohtaja luo tuotteen kehityskaaren.
- Asiakasjohtaja luo julkaisusuunnitelman.
- Asiakasjohtaja, tuotantoryhmä ja scrum master suunnittelevat kehitysjaksojen läpiviennin eli tulevat iteraatiot. Jokaisen kehitysjakson alussa tehdään uusi suunnitelma, minkä jälkeen itse kehitysjakso alkaa.
- Kehitysjakson jokaisena työpäivänä tuotantoryhmä kokoontuu päivittäin.
- Kehitysjakson päätteeksi käydään kehitysjakson saavutukset läpi asiakasjohtajan, tuotantoryhmän ja muiden projektiin osallistuvien tahojen kesken.

- Tuotantotiimi pitää kehityspalaverin, jonka perusteella muokataan toimintaa seuraavalle kehitysjaksolle.
- Kehitysjaksojen jälkeen projekti on päättynyt. [15.]

3.7 Tavoitteet

Ketterän kehitysmallin tuomat edut perustuvat turhien prosessien vähentämiseen: "Jos jokin ei tuota asiakkalle suoraa lisäarvoa, se on turhaa" [24, s. 18]. Tämä periaate on vahvasti edustettuna projektin elinkaaren jokaisessa vaiheessa. Ideaalitilanteessa haluttu toiminnallisuus kehitetään käytännössä katsoen välittömästi. Ideaalitilanne on tavoite, johon ketterässä kehitysmallissa pyritään.

Turhien prosessien karsiminen on välttämätöntä tavoitteisiin pääsemiseksi ja tärkeä osa ketterää kehitystä. Karsinta on aloitettava määrittelemällä, mitkä prosessit aiheuttavat turhaa työtä ja miten se ilmenee. Taulukosta 1 nähdään, mitkä ovat perinteisen tuotannon ja vastaavasti ohjelmistotuotannon seitsemän pääaluetta, joihin kiinnitetään huomiota turhien prosessien ja töiden karsimisessa.

Taulukko 1. Perinteisen tuotannon ja ohjelmistotuotannon seitsemän optimoitavaa prosessia [24, s. 19].

Perinteinen tuotanto	**Ohjelmistotuotanto**
varastointi	keskeneräinen tai osittain toteutettu toiminnallisuus
liika jalostus	turhat prosessit
liika tuotanto	ylimääräiset toiminnallisuudet
kuljetus	osallistuminen yhtäaikaisesti moneen projektiin
odottaminen	odottaminen
vuorovaikutus	vuorovaikutus

Turhien prosessien poistaminen aloitetaan määrittelemällä, mitkä prosessit aiheuttavat ylimääräistä tai turhaa työtä. [24.]

Keskeneräinen tai osittain toteutettu toiminnallisuus

Kaiken toiminnollisuuden tulee olla viimeisteltyä ja mahdollisuuksien mukaan integroituna päätuotteeseen. Osittain toteutetut toiminnollisuudet jäävät herkästi koko projektin ulkopuolelle, tai integrointivaiheessa huomataan, ettei toiminnollisuus enää vastaa toivottua lopputulosta. [24, s. 19.]

Turhat prosessit

Dokumentointi ja dokumentit, joihin ei ole aikaa tai kiinnostusta perehtyä, voidaan määritellä turhaksi. Asiakkaalle arvokkuudeltaan vähäiset dokumentit pidetään lyhyinä ja suurpiirteisinä, ja niiden tekeminen priorisoidaan tuotannollisen työn alapuolelle. Dokumentit kirjoitetaan niin, että niiden sisältämät vaatimukset voidaan ymmärtää ja vahvistaa niin asiakkaan kuin projektiryhmän puolesta.

Dokumentoinnin arvokkuus voidaan selvittää kysymällä: Odottaako joku kyseistä dokumenttia? Tarvitaanko sitä jatkoa varten esimerkiksi testaamisen ja ohjeiden kirjoittamiseen? Onko tuotanto riippuvainen dokumentista? Vastauksen perusteella määritellään, onko dokumentointi turhaa vai tuottaako se arvoa asiakkaalle. [24, s. 19.]

Ylimääräiset toiminnollisuudet

Jokainen toiminnollisuus rakennetaan, testataan, integroidaan ja dokumentoidaan sekä ylläpidetään koko järjestelmän elinkaaren ajan. Samalla jokainen uusi toiminnollisuus ja koodi toiminnollisuuden taustalla lisää järjestelmän monimutkaisuutta ja avaa riskin järjestelmän epävakaudelle. Mikäli asiakas ei ole määritellyt toiminnollisuutta, sen rakentamiseen ei ole perusteita. Projektin laajuuden kurissa pitäminen edesauttaa projektin tavoitteiden täyttymistä. [24, s. 19–20.]

Osallistuminen yhtäaikaisesti moneen projektiin

Keskittymisen siirtäminen projektista toiseen vie turhaa aikaa. Yksi ketterän kehitysmallin periaatteista on pitää projektiryhmä keskittyneenä omaan projektiinsa, jolloin hyödyt moninkertaistuvat. Monen projektin pitäminen tuotannossa on houkuttelevaa, mutta projektiryhmän kierrättäminen projektien välillä aiheuttaa turhaa työtä ja pidentää projektiin käytettyä aikaa. [24, s. 19–20.]

Odottaminen

Yksi ketterän kehityksen tarkoituksista on vähentää kehitykseen menevää aikaa ja tehdä siitä tehokasta eli ketterää. Mikäli kehitysjaksoja ei voida viedä läpi esimerkiksi asiakkaan puutteellisen tarkastuksen ja hyväksynnän takia, tämä aiheuttaa turhaa odottamista. Viiveet voivat johtua myös tuotannollisista, henkilöstöpoliittisista tai taloudellisista syistä. Vaikka jokaisen kehitysjakson alussa käydään läpi tavoitteet, voi tuotantoryhmällä herätä kysymyksiä toiminnollisuuksista, joihin vaaditaan asiakasjohtajan selvitystä.

Projektiviestintä hoidetaan yleisesti sähköpostitse. Tämä on yleensä suuri viiveiden aiheuttaja, koska viestiä joutuu odottamaan eikä se välttämättä sisällä tarvittavaa tietoa. Sähköposti on työkalu, joka on kiteyttänyt paikkansa yritysmaailmassa, niin hyvässä kuin pahassa. Ketterä kehitys kannustaa vuorovaikutteisuuteen, joka ei perustu odottamiseen. [24, s. 21.]

Vuorovaikutus

Ketterän kehityksen malli kannustaa välittömään vuorovaikutukseen. Siihen on mahdollista päästä, mikäli projektiryhmä työskentelee samassa tilassa tai välittömässä läheisyydessä. Tarvittaessa ryhmä voi työskennellä asiakkaan tiloissa, jolloin asiakkaan palautteen vastaanottamisen ja reagoimisen viive on minimaalinen. [24, s. 21.]

Viat

Aikaisessa vaiheessa korjatut viat vähentävät turhan työn määrää. On erityisen tärkeää, että tuotantoprosessi noudattelee kaavaa, jossa testaaminen, tuotteen integroiminen päätuotteeseen ja julkaisu tehdään mahdollisimman usein.

Projektin hallinnointi

Projektin hallinnointi ei suoraan vaikuta lopputuotteen arvoon tai lisää sitä. On kuitenkin hyvä tiedostaa, että hallinnointi saattaa aiheuttaa suuren määrän turhaa työtä. Esimerkiksi tuotantoryhmän seuraaminen erilaisten työkalujen avulla voi aiheuttaa enemmän haittaa kuin hyötyä. Myös prosessit projektin vaatimusten muuttamiseen aiheuttavat

yleensä turhaa odottamista, jolloin tuotantoryhmä ei pysty esimerkiksi suorittamaan omaa osuuttaan kehitysjaksosta.

Turhan työn havainnointi ja poistaminen on jatkuva prosessi. Ketterän kehityksen edut turhan työn minimoimiseen perustuvat jatkuvaan kehitykseen, joka koskee sekä tuotantoryhmää että tuotetta. Kuten luvussa 3.5 on kerrottu, jokaisen kehitysjakson jälkeen tuotantoryhmä pitää kehityspalaverin, jossa käydään läpi asioita, jotka hidastavat tai vaikeuttavat tuotantoryhmäläisten työtä. Näin ryhmä itse pystyy määrittelemään, mikä työ on turhaa ja miten se voidaan minimoida. [24, s. 22.]

4 Ketterä kehitysmenetelmä käytännön asiakasprojektissa

4.1 Vaatimusmäärittely

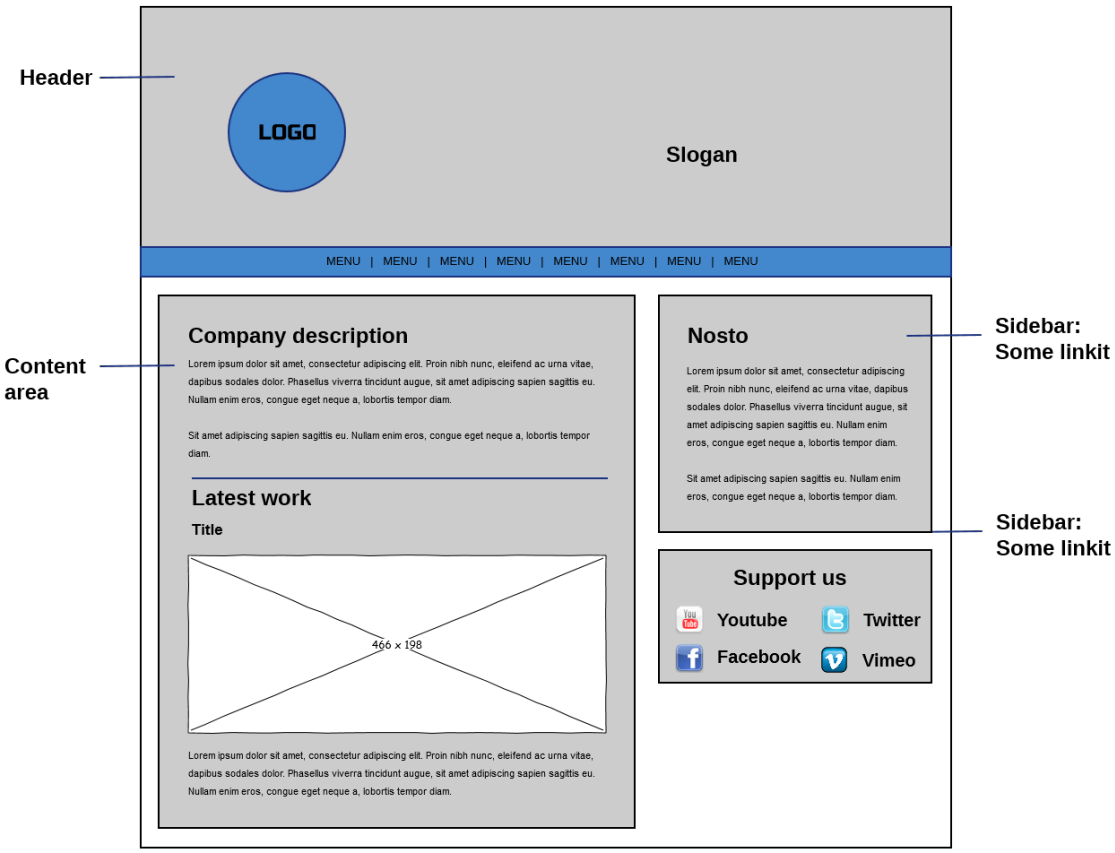
Insinöörityö aloitettiin palaverilla, jossa määriteltiin tulevan sivustoprojektin vaatimukset. Palaverissa käytiin läpi vastaavanlaisten yritysten sivustoja ja määriteltiin, mitä elementtejä uusille sivuille olisi hyvä saada. Tärkeäksi ominaisuudeksi nousi asiakkaan mahdollisuus lisätä, muokata ja poistaa sisältöä. Sivusto päätettiin rakentaa Drupal-sisällönhallintajärjestelmän päälle. Drupal on yksi suosituimmista sisällönhallintajärjestelmistä, ja sen laajennettavuus on omaa luokkaansa. [25.]

Ketterän kehityksen mukaisesti palaverin jälkeen määriteltiin prioriteetteihin järjestetty ominaisuuslistaus. Samalla määriteltiin tuotevisio ja luotiin alustava kehityskaari sekä julkaisusuunnitelma. Aikataulujen kiireellisyyden vuoksi päätettiin, että projektin alun kehitys tehdään nopealla aikataululla. Päätettiin myös, että sivuston kehitystä jatketaan julkaisun jälkeen, joten kaikkien toiminnollisuuksien rakentaminen insinöörityön puitteissa ei ollut välttämätöntä.

4.2 Suunnittelu

Varsinainen suunnittelu aloitettiin tekemällä rautalankamalli asiakkaan visioiman rakenteen perusteella. Asiakas oli toimittanut oman näkemyksensä sivuston pääelementeistä, joten rautalankamallin tekeminen oli suoraviivaista. Liite 2 sisältää asiakkaan suun-

nitteleman rautalankamallin, ja kuvasta 7 nähdään, miten yrityksen logo, navigaatio-elementit ja sisältö sijoittuvat refakturoituun rautalankamalliin.

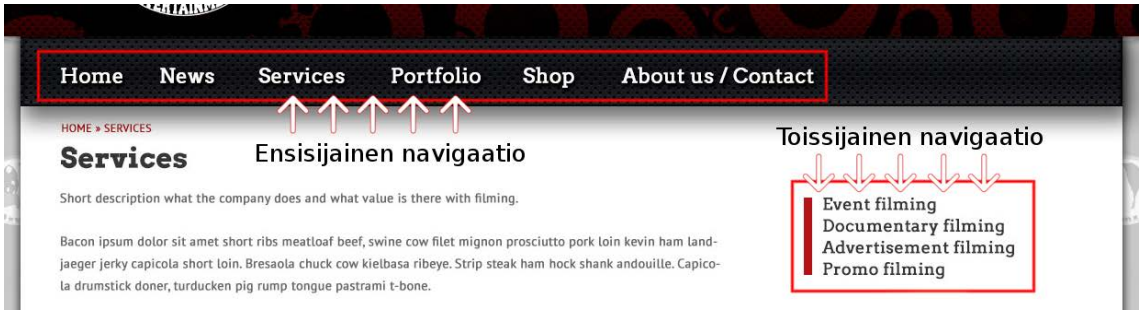


Kuva 7. Elementtien sijoittelu rautalankamallissa.

Rautalankamalli toteutettiin vain etusivusta, koska sivujen sisältö oli määrittelyssä todettu joko kevyeksi tai tekstipainotteiseksi. Ketterän kehityksen periaatteiden mukaisesti osa sivuston elementeistä jätettiin määrittelemättä, koska niiden tarkempi tarkoitus selviäisi myöhemmässä kehitysvaiheessa. Asiakkaan toiveiden mukaisesti sosiaalisen median aktivointi toteutettaisiin pääsivun call-to-action-linkeillä.

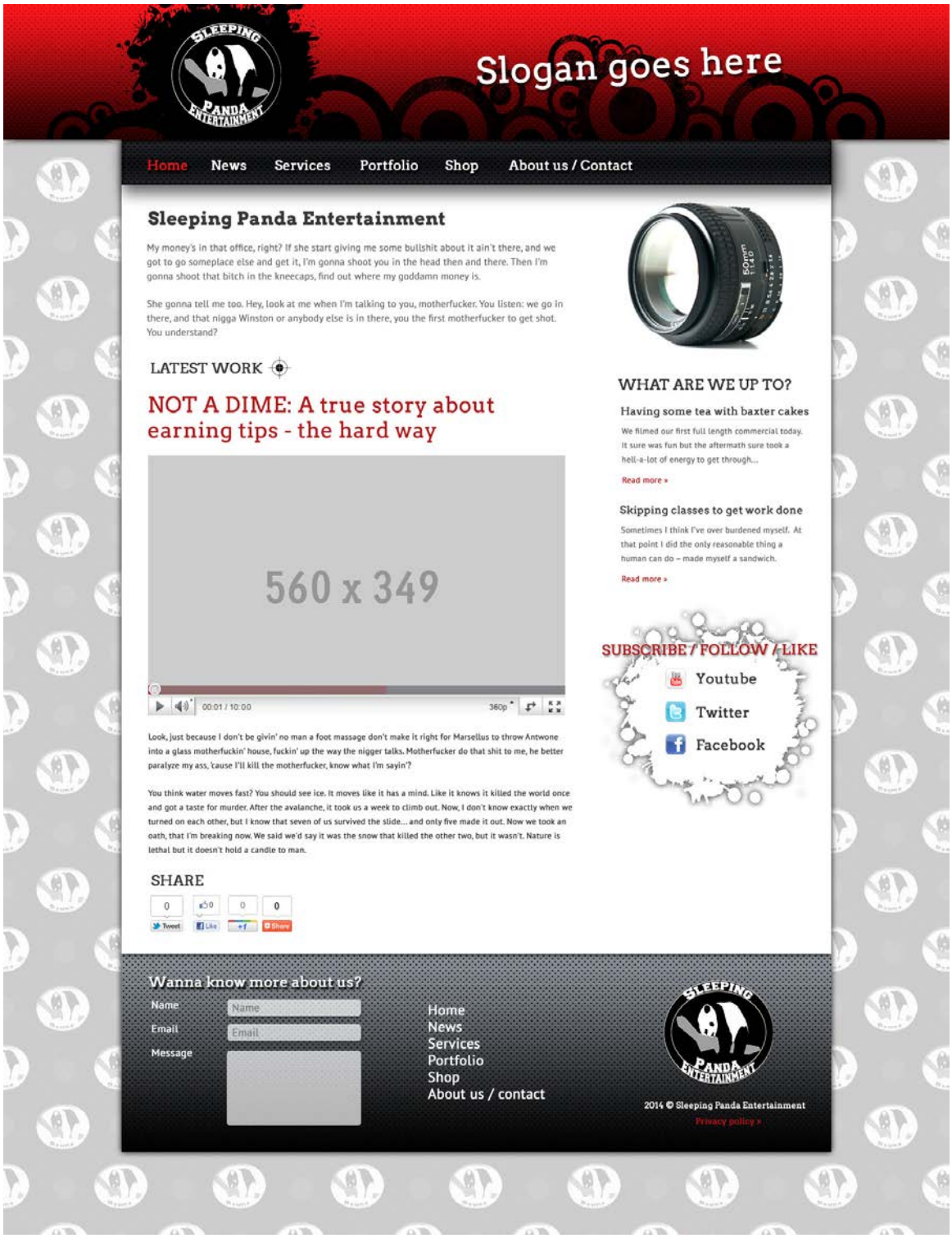
Ulkoasun visualisoinnin ja brändäyksen teki insinöörityön toteuttaja. Koska yritys oli uusi, siltä puuttui graafinen ohjeistus, jonka perusteella olisi voinut tehdä värimaailman, fonttien valinnan ja yleisen ulkoasun muotoilut. Osa kuvituskuvista ladattiin sxc.hu-palvelusta, joka tarjoaa kuvituskuvia ilmais- ja ostolisensseillä. Osa kuvista luotiin valmiilla vektoripohjilla, joita sai käyttää kaupalliseen työhön. Nämä vektorit ladattiin Deviant-art-palvelusta.

Ulkoasun visualisoinnissa pyrittiin ottamaan huomioon sivuston käytettävyys: Käyttäjälle sivuston tulisi olla selkeä, ja sivuilta tulisi selvitä kaikki oleellinen tieto kahdella klikkauksella. Käytettävyyden kannalta sivupolun eli breadcrumbin käyttäminen oli olennaista, koska sivustohierarkia on syvimmillään kolmitasoinen. Toissijaisen navigaation, joka nähdään kuvassa 8, merkitys on käyttäjälle vähäinen: suurin osa navigoinnista tapahtuu päätasoilta, jolloin käyttäjän ei tarvitse etsiä linkkejä alasivuille.



Kuva 8. Sivuston ensisijainen ja toissijainen navigaatio.

Ulkoasun visualisoinnin lopputuloksena saatiin aikaan layout, jonka asiakas hyväksyi suoraan ilman vedosversioiden tekemistä. Kuvassa 9 nähdään etusivun ulkoasu, josta selviävät sivuston graafiset elementit.



Kuva 9. Sleeping Panda Films -sivuston etusivun graafinen ulkoasu.

4.3 Ympäristön pystyttäminen

Sivuston rakentaminen aloitettiin asentamalla kehitysympäristö virtuaalipalvelimelle. Virtuaalipalvelin vuokrattiin DigitalOcean-nimiseltä yritykseltä, joka tarjoaa kehittäjälähtöisiä palvelimia todella edullisesti. DigitalOcean on kätevä muun muassa projekteihin, joissa tarvitaan nopeasti kustannustehokas palvelin: Kehitysympäristö on toimintakunnossa – ilman lisäosia tosin – noin 55 sekunnissa. Virtuaalipalvelin mahdollistaa palvelimen skaalautumisen portaattomasti, mikäli palvelimelle tarvitaan esimerkiksi lisää muistia tai tallennustilaa. [26.]

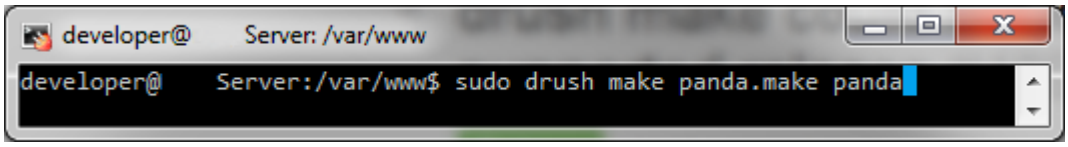
Työssä käytetty kehitysympäristö perustuu LAMP-kokonaisuuteen. LAMP muodostuu sanoista Linux, Apache, MySQL ja PHP. LAMPin käyttöönotto on helppoa sen suosion takia, ja esimerkiksi DigitalOceanin foorumeilta löytyy kattava dokumentaatio komponenttien asentamisesta. [26; 27.]

Tietokannan ja tietokantakäyttäjien lisäämiseen asennettiin phpMyAdmin, joka mahdollistaa MySQL-tietokantojen käyttämisen verkkokäyttöliittymän kautta. Drupal tarvitsee jokaiselle sivustolle oman tietokantansa, joten oli luonnollista tehdä sitä varten oma tietokanta ja tietokantakäyttäjä. Tiedot tietokannasta ja käyttäjästä asennetaan Drupalin hallintatiedostoon. [28.]

Palvelimelle asennettiin SASS-preprosessori. Se mahdollistaa ohjelmointimaisen lähestymistavan CSS-tiedostojen rakentamiseen, jolloin esimerkiksi responsiivisuuden rakentaminen helpottuu olennaisesti. Palvelimelle asennettiin myös Suzy-, Singularity- ja Breakpoints-lisäosat, jotka helpottavat sivuston teemausta ja responsiivisuuden rakentamista olennaisesti. [29; 30; 31; 32.]

Samalla asennettiin Drupalin komentokehotteessa toimiva Drush, josta on muodostunut lähes välttämätön komponentti Drupal-kehittäjien keskuudessa. Drush helpottaa sivustojen asentamista, rakentamista ja ylläpitoa tarjoamalla helppokäyttöisen ja selkeisiin komentoihin perustuvan työkalun. Drushin avulla sivusto voidaan esimerkiksi asentaa paketinhallinnan avulla, jolloin yhdellä komennolla saadaan kaikki halutut Drupal-lisäosat, teemat ja tiedostot. [25.] Liite 2 sisältää esimerkkimallin paketinhallintatiedostosta, jossa ovat sivuston olennaiset komponentit ja teema.

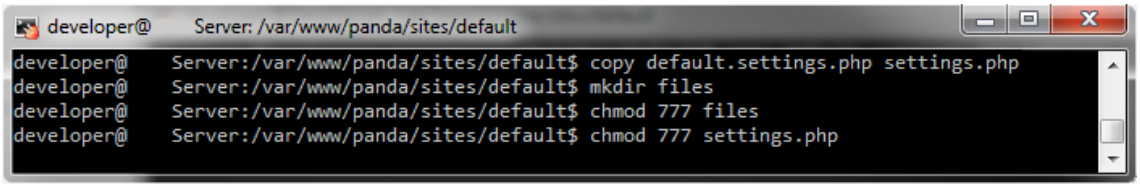
Kuvassa 10 nähdään, miten yksinkertaista on Drupal-sivuston tarvittavien komponenttien lataaminen paketinhallintatiedostoa käyttäen. Työssä käytetään Drupalin uusinta versiota, joka kirjoitushetkellä on 7.26. Paketinhallintatiedostosta näkee kaikki ne moduulit, joita käytännön työssä käytetään.



Kuva 10. Paketinhallintatiedoston käyttäminen Drupal-komponenttien lataamisessa.

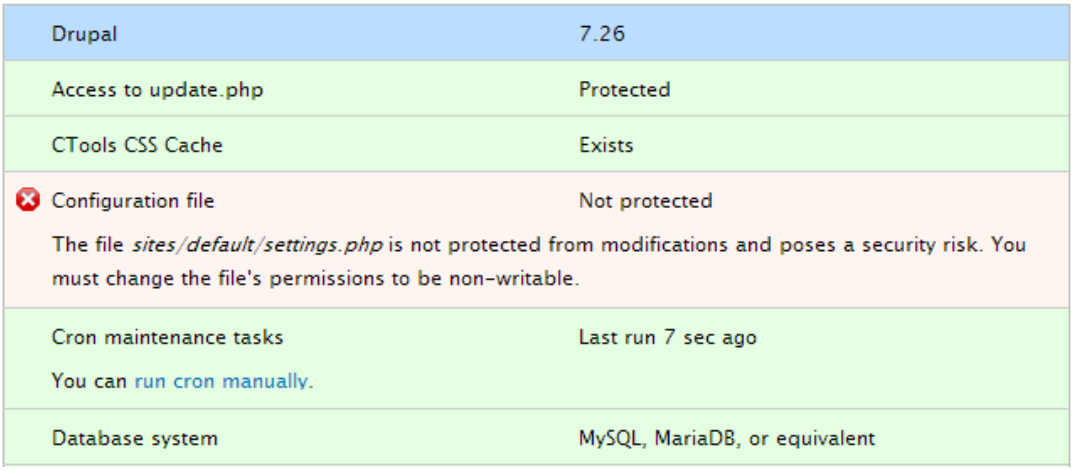
Drupalin asentaminen viedään loppuun graafisen käyttöliittymän kautta. Graafinen käyttöliittymä vie asennusprosessin läpi kohta kohdalta, mutta kehittäjälle olennaisin kohta on antaa käyttöoikeudet järjestelmätiedostolle sekä määrittää tietokanta ja yhteys siihen. Graafinen käyttöliittymä sisältää kaiken tarvittavan tiedon, jonka avulla Drupalin saa asennettua käyttökuntoon.

Jotta Drupal voi tehdä muutoksia järjestelmään, on tiedostojen kirjoittamisoikeudet vapautettava asennushetkeksi. Linux-ympäristössä tämän voi tehdä kuvan 11 mukaisella komennolla, jossa kopioidaan oletustiedosto järjestelmätiedostoksi, luodaan Drupalin tarvitsema tiedostokansio, annetaan tiedostokansioon kirjoitusoikeudet ja mahdollistetaan järjestelmän kirjoittaminen järjestelmätiedostoon. Drupal tarvitsee kansion luoduille tiedostoille ja muodostetuille kuville, ja se toimii järjestelmän välimuistien säilytyspaikkana.



Kuva 11. Tiedosto-oikeuksien asettaminen rakennettavalle Drupal-sivustolle.

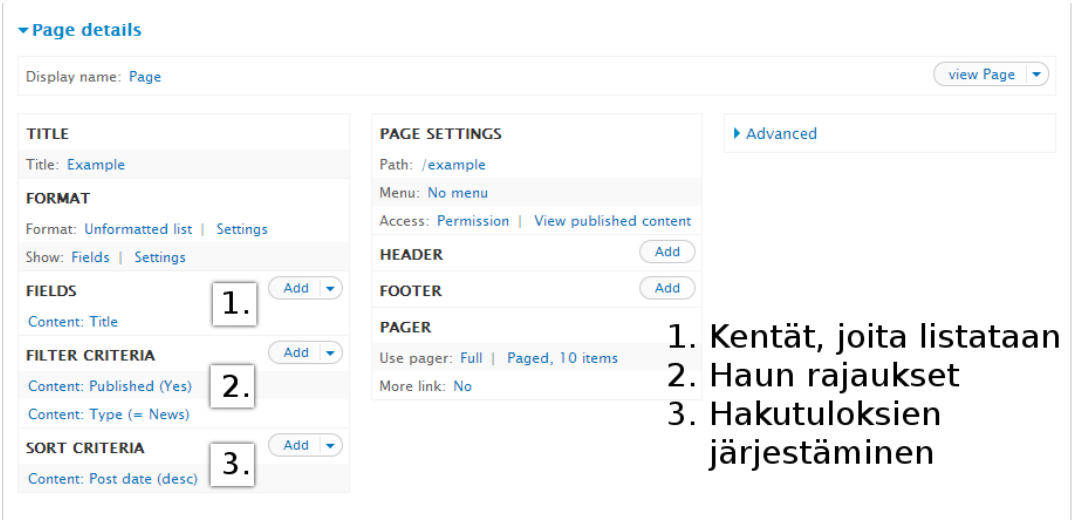
Kun tietokantayhteys on muodostettu, tulee kirjoitusoikeudet palauttaa takaisin oletusarvoihinsa. Kehitysympäristön kanssa voidaan käyttää Drupalin sisäistä raportointia, jolloin ongelmatilanteiden huomaaminen ja selvittäminen helpottuu olennaisesti. Kuvassa 12 nähdään raportoinnin pääsivun elementtejä, ja esimerkkinä kirjoitusoikeudet järjestelmätiedostoon on jätetty päälle.



Kuva 12. Drupalin raportointijärjestelmä ja kirjoitusoikeudet järjestelmätiedostoon.

Sivuston rakennusteemaksi valittiin Omega 4. Sen vahvuudet pohjautuvat pohjamalleihin, jolloin erilaisilla sivuilla voi olla erilainen sivustorakenne. Drupalin Context-moduulin avulla erilaiset komponentit on helppo asentaa niin, että ne näkyvät joka sivulla. Ketterän kehityksen mukaisesti Omega 4 -teema on versatiili ja sen avulla muutosten tekeminen on helppoa ja vaivatonta. [25.]

Drupalin ja tämän projektin tärkein erillismoduuli on Views, joka mahdollistaa erilaisten listausten rakentamisen helppokäyttöisen käyttöliittymän kautta. Viewsin avulla voidaan tehdä tietokantahakuja ja yhdistellä erilaisia sisältötyyppejä relaatioiden avulla. Listauksia voidaan rajata esimerkiksi sisältötyypin mukaan, ja niitä voidaan esittää julkaisupäivämäärän ja nimen perusteella tai vaikkapa sisällönluojan mukaan. Kuvassa 13 nähdään Viewsin perusnäkymä, johon on korostettu sen päätyökalut. Tässä projektissa Viewsiä käytetään listaamaan uutisia ja palveluita.



Kuva 13. Views-moduulin perusnäkymä.

Viewsin on todettu olevan niin tärkeä komponentti Drupal-kehityksessä, että se on päätetty lisätä Drupalin ytimeen versiosta 8 lähtien.

Toinen olennaisesti tärkeä moduuli kehitykselle on Context. Contextin avulla voidaan määrittää erilaisten elementtien näkyminen esimerkiksi sivupolun, tietotyypin, käyttäjän tai kielen mukaan. Tässä työssä Contextia käytetään muun muassa listauksien näyttämiseen polun mukaan ja pääelementtien sisällyttämiseen kaikille sivuille.

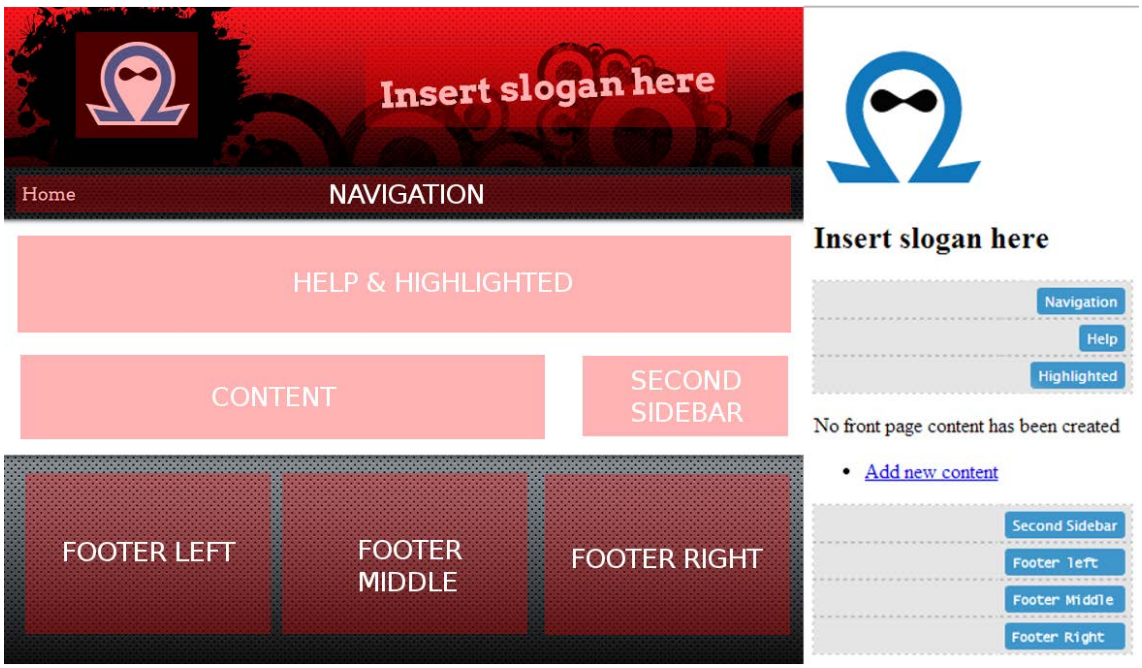
Kolmanneksi tärkein moduuli ja toiminnollisuus kätkeytyy Webforms-moduuliin. Sen avulla voidaan tehdä lomakkeita, esimerkiksi yhteydenottolomake, joka mahdollistaa vuorovaikutuksen käyttäjien ja asiakkaan kanssa. Tässä työssä Webformia käytetään yhteydenottolomakkeen rakentamiseen ja edellä mainittua Context-moduulia sen näyttämiseen. [25.]

4.4 Rakentaminen ja kehitysjaksojen aloittaminen

Kehitysjakso 1

Ensimmäisen kehitysjakson tarkoitus oli rakentaa teema eli luoda sivuston pääkomponentit. Suunnitelmien mukaan sivusto sisältäisi kaksi erilaista sivupohjaa: etusivun ja alasivun. Niistä ensimmäisen kehitysjakson tavoitteena oli luoda etusivun teema ja

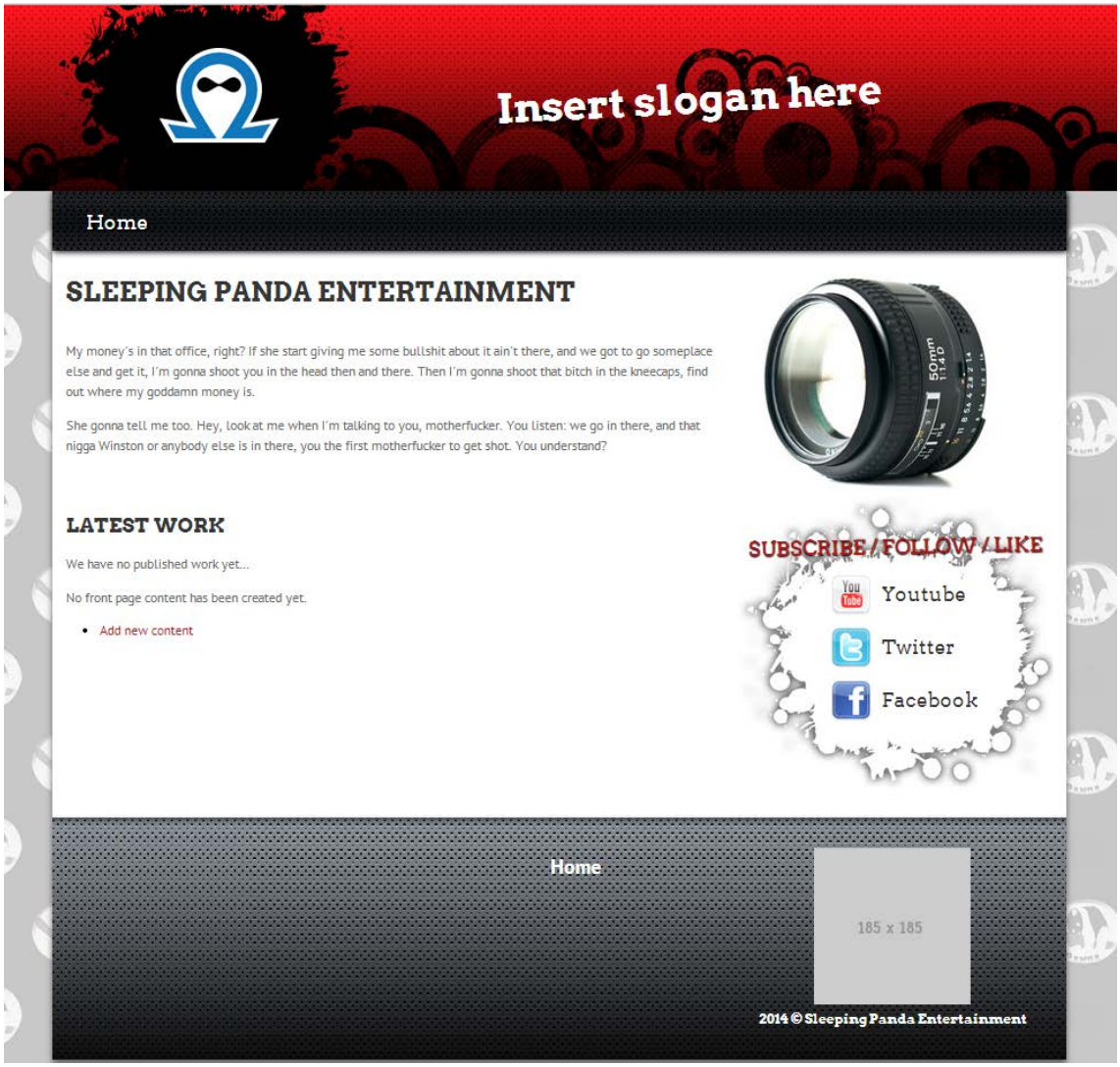
samalla sivuston pysyvien komponenttien teema. Kuvassa 14 nähdään, miten Omega 4 -teeman oletusulkoasu korreloi rakennetun ulkoasun kanssa.



Kuva 14. Etusivu teemoitettuna ja sen korrelaatio Omega 4 -perusteeman kanssa.

Sivuston responsiivisuuden kannalta oli tärkeää rakentaa sivusto gridien eli palstojen varaan. Kuten luvussa 4.3 kerrottiin, sivustolle asennettiin lisäosia helpottamaan responsiivisuuden rakentamista. Responsiivisuus ei ollut tämän kehitysvaiheen työlistalla, mutta jatkoa ajatellen oli tärkeää, että sivusto teemoitetaan tukemaan responsiivisuutta.

Jakson lopuksi käytiin asiakkaan kanssa läpi tehdyt toimenpiteet. Tässä vaiheessa kehitystä päätettiin siirtää responsiivisuuden rakentaminen jatkokehitykseen aikataulujen kiireellisyyden vuoksi. Sivustolla ei ollut vielä toiminnallisuutta, jota asiakas olisi voinut testata, joten käyttötilin luonti ja käyttöönoton opastus siirrettiin seuraavaan kehitysjaksoon. Kuvasta 15 nähdään etusivu ja sosiaalisen median linkkinosto.



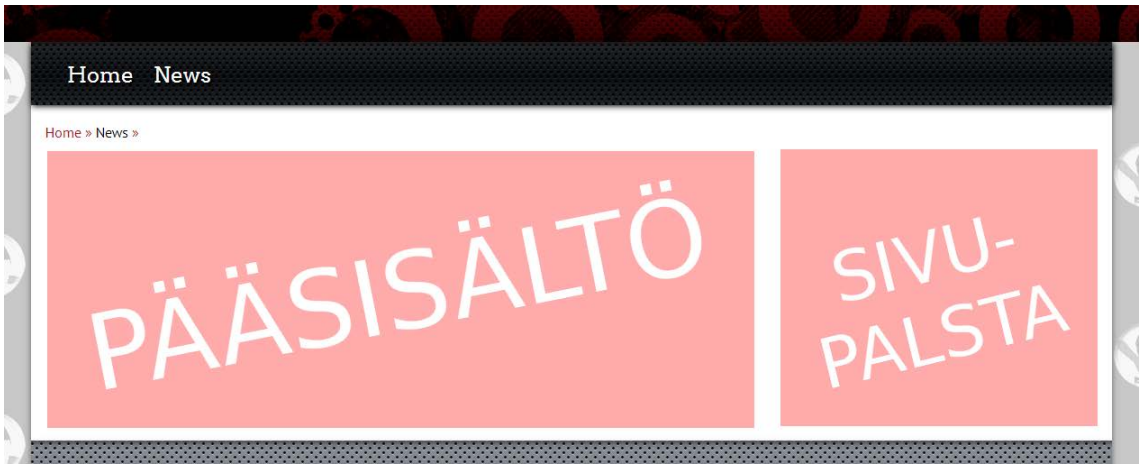
Kuva 15. Etusivu ensimmäisen kehitysjakson jälkeen. Sosiaalisen median linkit oikeassa palstassa.

Kehitysjakso 2

Toisen kehitysjakson tavoitteena oli luoda uutislistaus, joka näkyisi omalla sivullaan sekä etusivulla lyhyenä listauksena. Itse uutiset näkyisivät omalla alasivullaan. Asiakkaan kannalta oli olennaista, että uutislistaukset ovat dynaamisia, jolloin asiakkaan lisätessä uutisen listaukset päivittyvät automaattisesti. Näitä listauksia tehdään Drupalissa Views- moduulin avulla, joka esiteltiin luvussa 4.3.

Ennen uutistoiminnallisuuden rakentamista rakennettiin alasivun teema. Etusivusta poiketen alasivut noudattelivat kaksijakoista teemaa ilman niin sanottua nostoaluetta,

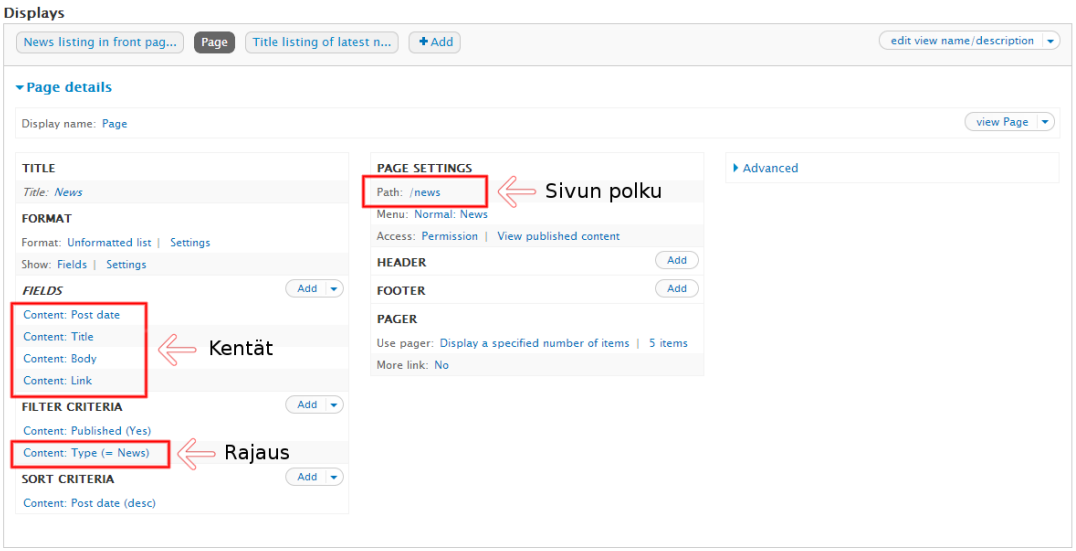
jolloin sisältö on kuvan 16 mukaisesti leveällä palstalla ja navigaatio sekä nostot oikealla puolella, kapeassa palstassa. Muutoin teema oli sama kuin etusivulla.



Kuva 16. Sleeping panda films -sivuston alasivujen ulkoasu.

Jotta uutisia pystyisi käsittelemään erillisinä komponentteina, rakennettiin sisältötyyppi, joka korreloi uutisen kanssa. Tämä sisältötyyppi ei eroa olennaisesti Drupalin oletussisältöyypistä, mutta näin voidaan tarvittaessa liittää uutissisältötyyppiin uusia kenttiä. Drupal luo oletuksena kaikille sisältötyypeille esimerkiksi kentän, johon tallennetaan sisältötyypin luomisajankohta, muokkaamisajankohta ja relaatio sisältötyypin instanssin luojan Drupal-tiliin.

Uutislistausten tekeminen aloitettiin uutissivun rakentamisella. Käytännössä sivulla listataan uutisen julkaisupäivämäärä, otsikko, lyhyt kuvaus uutisesta ja linkki varsinaiseen uutiseen. Kaikki mainitut elementit korreloivat Views-listauksen elementtien kanssa, mikä nähdään kuvassa 17. Koska kyseessä on kokonainen sivu, listaukseen määritellään sivun polku ja näkyvyys halutussa valikossa.

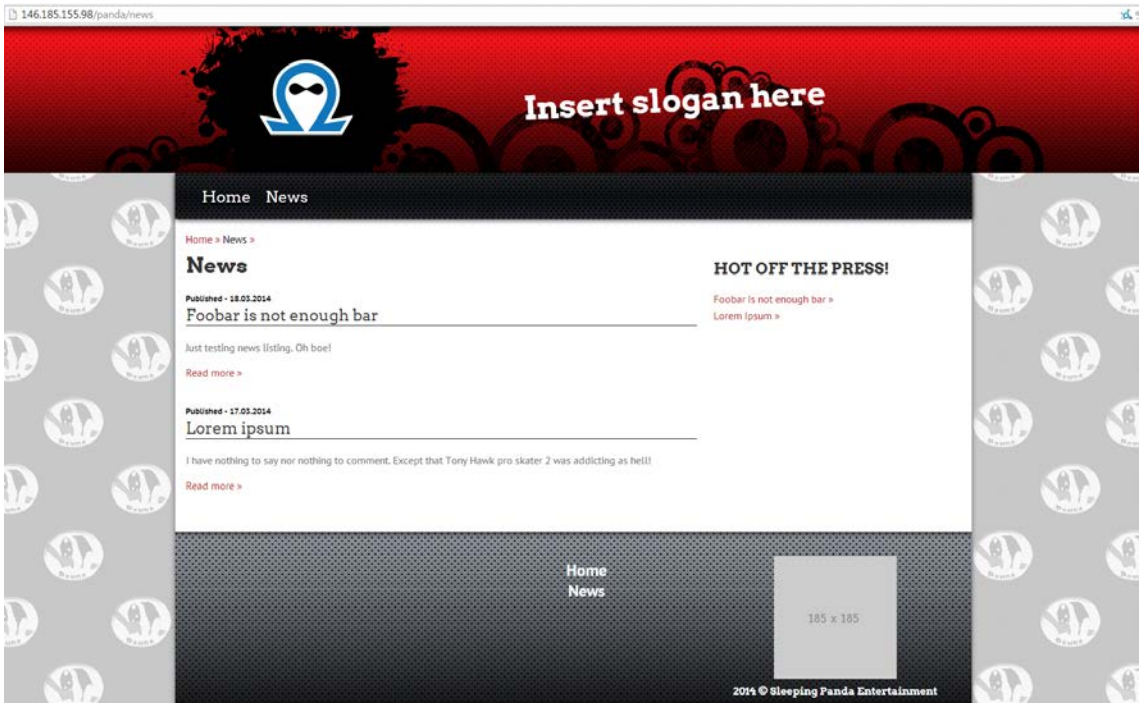


Kuva 17. Uutissivun listauksen komponentit.

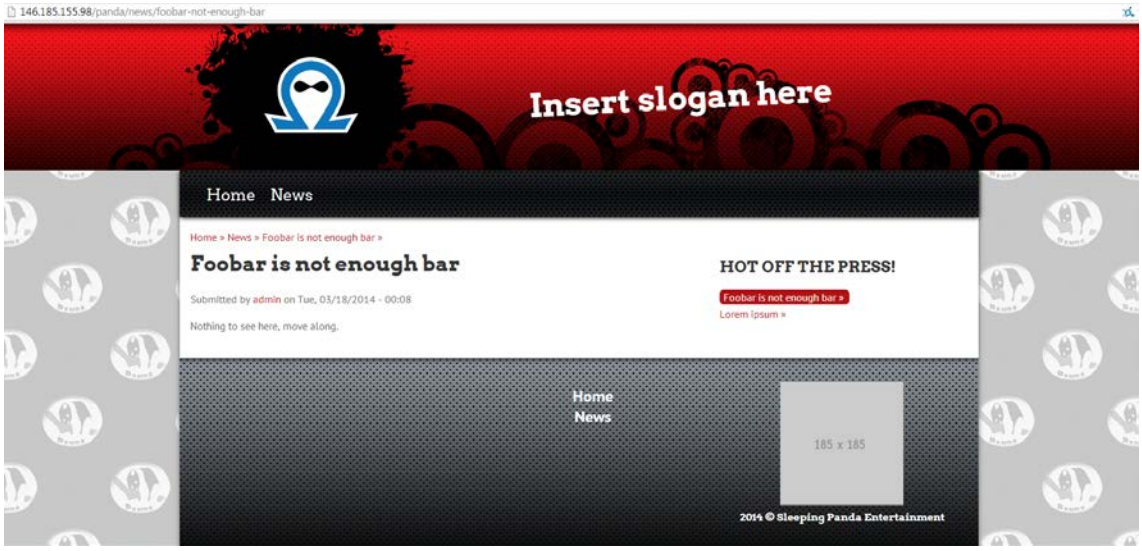
Uutissivun oikeaan reunaan rakennettiin navigaatio, joka perustuu samankaltaiseen listaukseen. Tähän listaukseen tulostetaan kaikki uutiset otsikoittain, jotka toimivat linkkinä sisältöön. Uutisen omalla sivulla listauksessa näkyy, millä sivulla ollaan, jolloin se (listaus) toimii myös navigaationa. Kaikki uutislistaukset järjestetään uutisten julkaisuajankohdan mukaan, jolloin uusin uutinen listataan ensimmäisenä.

Sivuston osoitteiston järkevöittämiseksi jokaiselle tietotyypille määritellään oletussivustopolkunsa, jolloin uusien uutisten sivustopolku luodaan aina samalla alkuliitteellä. Esimerkiksi uutisten kohdalla sivustopolun alkuliite on ”news” ja palveluiden ”services”. Tämä on olennaista jatkokehityksen ja hakukonenäkyvyyden kannalta.

Kehitysjakson jälkeen asiakkaalle toimitettiin tunnukset järjestelmään ja ohjeistettiin, miten uutisia lisätään. Asiakas kävi lisäämässä testiuutisen ja hyväksyi tehdyt toimenpiteet. Kuvista 18 ja 19 nähdään, miten uutiset listautuvat pääsivulle ja itse uutisen sivulle.



Kuva 18. Uutislistaus, jossa uutiset lyhennelminä ja sivun reunassa sama listaus otsikoittain.



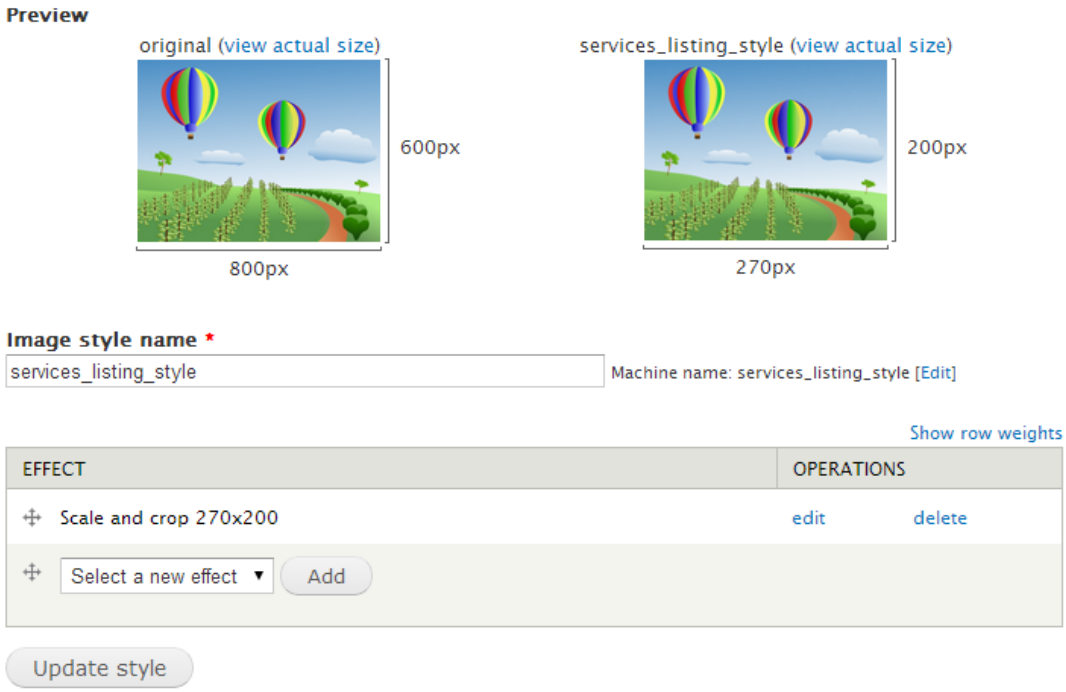
Kuva 19. Uutisen alasivu, jossa oikealla puolella aktiivinen linkki punaisella taustalla.

Kehitysjakso 3

Kehitysvaiheessa 3 oli tarkoitus rakentaa sivut palveluille. Palveluiden rakentaminen käytännössä noudattelee lähes täysin samanlaista kaavaa kuin uutisten rakentaminen. Erona uutissisältötyyppiin palveluiden sisältötyypissä on lisäksi listauskuva, joka näytetään palveluiden listaussivulla. Samalla luotiin valikko eri palveluille. Siihen käytettiin

Drupalin omaa valikkojärjestelmää, toisin kuin uutislistauksessa, jossa valikko toteutettiin Views-moduulilla.

Palveluiden sisältötyyppi sisältää kuvan, joka näytetään listauksessa. Drupalissa itsessään on mahdollista määrittää kuville tehtäviä prosessointeja, kuten skaalausta, rajausta ja erilaisia suodattimia, jolloin kuvia ei tarvitse erikseen muokata sopivaan muotoon. Kuvassa 20 nähdään kuvatyyli, jota käytetään palveluiden listaamisessa. Kuvaa voi jatkossa käyttää muualla, koska alkuperäinen kuva tallennetaan palvelimelle muokkaamattomana.

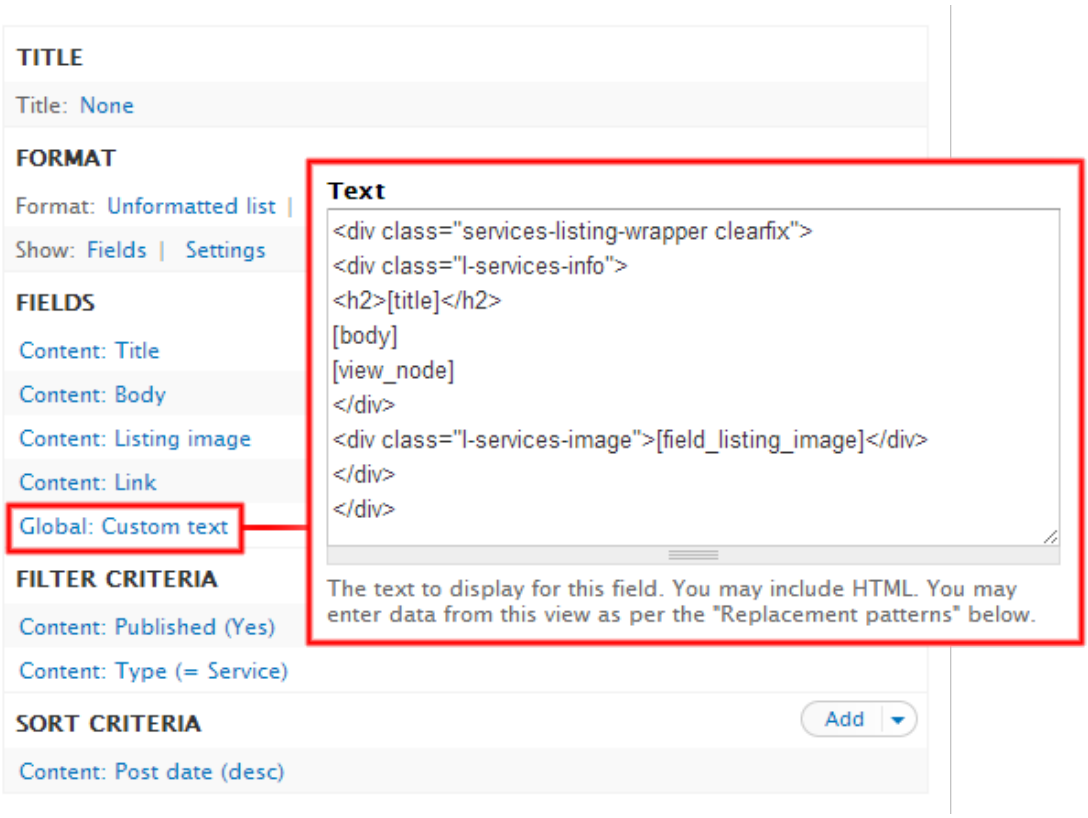


Kuva 20. Palvelukuvien listaustyyli.

Palveluiden listaussivu on rakenteeltaan erilainen kuin uutissivun listaus. Palveluiden listaussivulle on jätetty asiakkaalle alue, johon hän voi kirjoittaa yleiskuvauksen palveluista. Palvelut-sivu on perussivu, jonka loppuun lisätään Context-moduulilla palvelulistaus. Vasempaan palstaan lisätään valikko kaikista järjestelmään lisätyistä palveluista.

Palvelut sivujen listauksessa käytetään Views-moduulin toiminnollisuutta, joka mahdollistaa kenttien kirjoittamisen uuteen pohjaan. Käytännössä tämä helpottaa teemoittamista, koska kentille voi antaa erilaisia määritteitä, ne voidaan ympäröidä eri elemen-

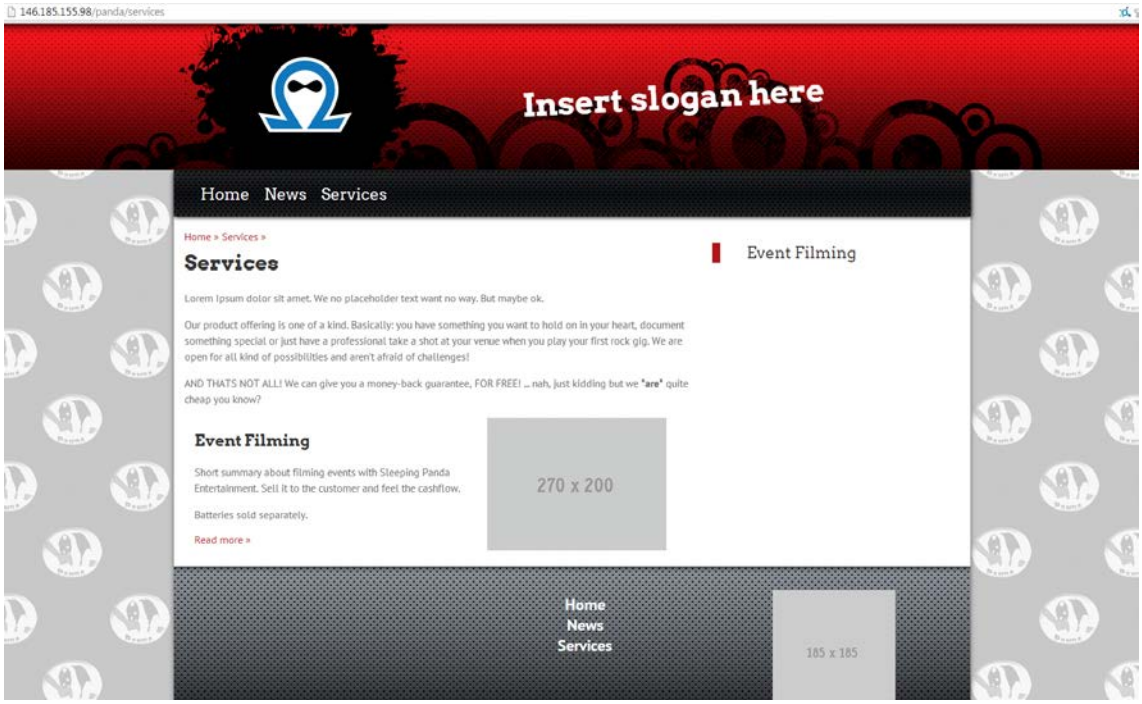
teillä tai niiden järjestystä voidaan muuttaa. Tämä on toteutettu ylimääräisen, generoidun, kentän avulla, joka nähdään kuvassa 21.



Kuva 21. Palvelulistauksen muodostaminen uuteen pohjaan.

Tätä tekniikkaa käyttäessä on muistettava piilottaa kentät tavallisesta listauksesta, jolloin listauskomponentit eivät tulostu kahteen kertaan. Tämä tehdään asettamalla jokaisen uudelleen kirjoitettavan kentän kohdalle asetus ”Exlude from display”, jolloin Drupal ei tulosta kenttää ruudulle.

Kehitysjakson jälkeen toiminnollisuus käytiin läpi asiakkaan kanssa. Asiakas lisäsi testipalvelun ja totesi toiminnollisuuden olevan sovitun mukainen. Kuvasta 22 nähdään palvelulistauksen sivu, jossa oikealla puolella näkyy toissijainen navigaatio ja pääsisällössä nosto palveluun.



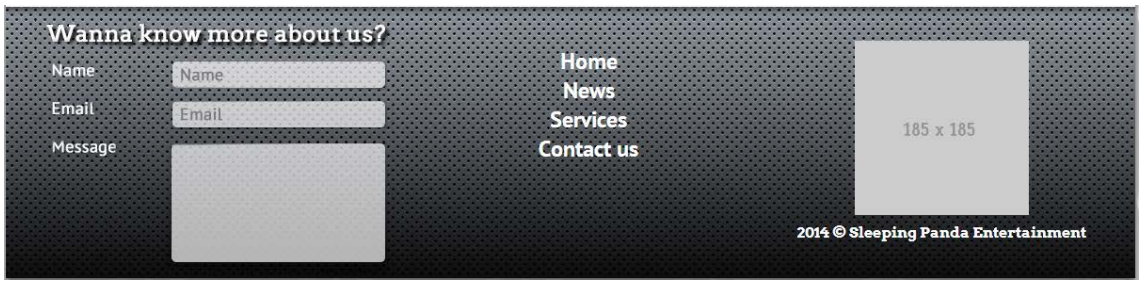
Kuva 22. Palvelulistaus, jossa toissijainen navigaatio oikealla ja palvelunostot keskellä.

Kehitysjakso 4

Viimeisessä, neljännessä kehitysjaksossa oli tarkoitus rakentaa sivustolle yhteydenottolomake. Yhteydenottolomake näkyy jokaisen sivun vasemmassa alareunassa, ja itse lomakesivulla on tietoja yrityksestä. Tämä toiminnollisuus koostetaan Webform-moduulin avulla. Jokaiselle kentälle määritellään tyyppi, pakollisuus ja muita ominaisuuksia. Jotta lomakkeen voi liittää sivustoon erillisenä komponenttia, tulee asetuksista laittaa kohta ”Available as block” päälle, jolloin lomakkeen voi asettaa Contextin avulla haluamaansa kohtaan ja/tai sivulle.

Kun lomake on syötetty sivuille, määritellään, mitä lomakkeiden lähettäneiden tiedoille tehdään. Oletusarvona tiedot lähetetään asiakkaan hallitsemaan sähköpostiin, mutta Drupal tallentaa lähetetyt lomakkeet myös omaan tietokantaansa. Näin tiedot tallentuvat, vaikka sähköpostin lähettämisessä tapahtuisi virhe.

Kehitysvaiheen jälkeen toiminnollisuus esitettiin asiakkaalle ja hän lähetti järjestelmän kautta testilomakkeen. Kuvassa 23 nähdään yhteydenottolomake. Lomake tulostuu jokaiselle sivulle.



Kuva 23. Yhteydenottolomake jokaisen sivun alareunassa.

4.5 Julkaiseminen

Varsinaista julkaisemista ei tehty. Asiakkaan kanssa sovittiin, että sivusto on kehityspalvelimella niin kauan, että asiakas saa hankittua itselleen oman palvelimen. Sivusto on julkaistuna kirjoitushetkellä, mutta palvelimelle ei ohjaa mikään järkevä osoite. Viralisen julkaisun voi tehdä vasta, kun asiakas hankkii itselleen verkko-osoitteen. Projektin päätteeksi asiakkaalle toimitettiin sivuston tiedostot, tietokanta ja ohjeet, kuinka sivustoa käytetään.

5 Yhteenveto

Insinöörityön tarkoituksena oli selvittää ketterän kehityksen ja Scrum-lähestymistavan perusteet ja soveltaa niitä käytännön asiakasprojektiin. Tavoitteena oli saada julkaisukelpoinen sivusto, johon asiakas pääsee itse lisäämään sisältöä ja joka toimii asiakkaan liiketoiminnan tukena.

Ketterä kehitys ja Scrum mahdollistivat muutoksiin vastaamisen ja sopeutumisen projektin aikana. Scrumin käyttäminen projektihallintametodina ilmeni asiakkaalle muun muassa projektin verkkaisena etenemisenä. Asiakas koki saavansa lisäarvoa Scrumin käyttämisestä.

Projektin toteutukseen oli varattu kaksi kuukautta aikaa, josta kehityksen osuus kesti noin kuukauden verran. Aloituspalaverin jälkeen suunnitteluun varattiin neljä viikkoa, jonka aikana tehtiin vaatimusmäärittely, rautalankamalli ja sivuston alustava ulkoasu. Osa-alueet kestivät viikon, paitsi sivuston ulkoasun suunnittelu, johon varasin kaksi viikkoa, koska graafinen suunnittelu ei ole vahvinta osaamisaluettani.

Kehitysympäristön asentaminen DigitalOcean-virtuaalipalvelimelle ja Drupalin käyttäminen sivuston alustana osoittautuivat hyviksi valinnoiksi. Oman kehityspalvelimen käyttäminen helpotti sivuston asetusten tekemistä ja aikaisempi kokemus Drupalista mahdollisti toiminnollisuuksien rakentamisen ketterästi. Jatkokehityksen kannalta Drupalin käyttäminen oli järkevä ratkaisu sen laajennettavuuden takia.

Kehitysvaiheiden läpivienti saatiin pidettyä tiiviinä, jolloin ketterän kehityksen mukainen ominaisuuslista eli koko projektin aikana. Asiakas omaksui oman roolinsa ketterän kehityksen periaatteiden mukaisesti ja antoi palautetta ja kehitysideoita ja toimi laadunvalvojana ja osittain myös testaajana. Sivusto käytiin toiminnollisuuksien valmistuessa läpi, ja asiakkaalle toimitettu ohje projektin päättyessä oli lähinnä muistilista sivuston eri toiminnollisuuksista.

Asiakkaan mielestä projekti onnistui yli odotusten ja sen (projektin) läpivienti ketterästi toi lisäarvoa lopputulokseen. Sivuston rakentaminen voitiin aloittaa käytännössä heti ensimmäisen ulkoasun vedosversion hyväksynnän jälkeen, ja asiakas oli ulkoasuun tyytyväinen. Sivuston kehitystä jatketaan asiakkaan kanssa samoja periaatteita noudatten.

Lähteet

1 Software Development Life Cycle (SDLC). Verkkodokumentti. Simply Easy Learning by tutorialspoint.com. <http://www.tutorialspoint.com/sdlc/sdlc_tutorial.pdf> Luettu 14.2.2014.

2 Goto, Kelly & Cotler, Emily. 2005. Web ReDesign 2.0 | Workflow that works. New Riders Publishing.

3 Sinkula, Mike. 2011. A Website Project Workflow that Works. Verkkodokumenti.<http://www.premiumdw.com/case-studies/a-website-project-workflow-that-works/> Päivitetty 4.5.2011. Luettu 14.2.2014.

4 Hietala, Henri. 2014. Senior Director of Web Development, Innofactor. Haastattelu 1.4.2014.

5 Search Engine Optimization. 2010. Verkkodokumentti. Google Inc. <http://static.googleusercontent.com/media/www.google.fi/en/fi/webmasters/docs/search-engine-optimization-starter-guide.pdf> Luettu 21.3.2014.

6 What is scrum? Verkkodokumentti. Scrum.org. <https://www.scrum.org/Resources/What-is-Scrum> Luettu 23.2.2014.

7 Agile Methodologies for Software Development. Verkkodokumentti. VersionOne, Inc. <http://www.versionone.com/Agile101/Agile-Development-Methodologies-Scrum-Kanban-Lean-XP/> Luettu 24.2.2014.

8 Kanban and Agile. Verkkosivusto. LeanKit Inc. <http://leankit.com/kanban/kanban-agile/> Luettu 24.2.2014.

9 Cohn, Mike. 2007. Differences Between Scrum and Extreme Programming. Verkkodokumentti. <http://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming> Päivitetty 6.4.2007. Luettu 24.2.2014.

10 Freedman, Rick. 2010. Four variants of agile development methods. Verkkoblogi. <http://www.techrepublic.com/blog/tech-decision-maker/four-variants-of-agile-development-methods/3664/> Päivitetty 26.5.2010. Luettu 24.2.2014.

11 Mauch, MB. & Bassuday. K. & Van zyl, J. & Le roux, K. Crystal Methodology COS 730. 2012. Verkkodokumentti. <http://www.slideshare.net/bassuday/crystal-methodology> Päivitetty 6.5.2012. Luettu 26.2.2014.

12 Mills, Amy. 2013. Agile, DSDM, SCRUM. Confused, Let us Help. Verkkosivusto. <http://www.quanta.co.uk/news/2013/11/agiledsdmscrum-confusedlet-us-help> Päivitetty 11.12.2013. Luettu 28.2.2014.

13 AGILE Methods of Software Development. 2011. Verkkosivusto. <http://dsdmofagilemethodology.wikidot.com/> Päivitetty 20.8.2011. Luettu 26.2.2014.

14 Feature Driven development. 2002. Verkkodokumentti. Nebulon Pty, Ltd. <http://www.featuredrivendevelopment.com/files/fddprocessesA4.pdf> Luettu 1.3.2014.

15 Layton, Mark. 2012. Agile Project Management For Dummies. John Wiley & Sons, Inc.

16 Deep lead -esittely. 2013. Verkkodokumentti Deep Lead Oy. <http://www.slideshare.net/DeepLead/deep-lead-esittely> Luettu 1.3.2014.

17 Scrum Product Owner. 2009. Verkkodokumentti. Scrum methodology. <http://scrummethodology.com/scrum-product-owner/> Päivitetty 9.9.2009. Luettu 1.3.2014.

18 Scrum Product Backlog. Verkkodokumentti. Mountain Goat Software. <http://www.mountaingoatsoftware.com/agile/scrum/product-backlog/> Luettu. 21.2.2014.

19 The Burn-Down Chart: An Effective Planning and Tracking Tool. 2013. Verkkodokumentti. Scrum Alliance. <http://www.scrumalliance.org/community/articles/2013/august/burn-down-chart-%E2%80%93-an-effective-planning-and-tracki> Luettu 1.3.2014.

20 Agile/Scrum and Product Roadmaps. 2008. Verkkodokumentti. On Product Management. <http://onproductmanagement.net/2008/10/28/agilescrum-and-product-roadmaps/> Luettu 22.3.2014.

21 Sprint Backlog. Verkkodokumentti. Mountain Goat Software. <http://www.mountaingoatsoftware.com/agile/scrum/sprint-backlog/> Luettu 23.3.2014.

22 Sprint Planning Meeting. Verkkodokumentti. Mountain Goat Software. <http://www.mountaingoatsoftware.com/agile/scrum/sprint-planning-meeting/> Luettu 2.3.2014.

23 Daily Scrum. Verkkodokumentti. Mountain Goat Software.<http://www.mountaingoatsoftware.com/agile/scrum/daily-scrum/> Luettu 2.3.2014.

24 Poppendieck, Mary & Poppendieck, Tom. 2003. Lean Software Development: An Agile Toolkit. Addison Wesley.

25 Drupal. Verkkosivusto. Drupal.org <https://drupal.org/> Luettu 18.2.2014.

26 Digital Ocean. Verkkosivusto. DigitalOcean Inc. <https://www.digitalocean.com/> Luettu 18.2.2014.

27 LAMP Stack - Web Stack (MySQL). Verkkosivusto. Turnkey Linux. <http://www.turnkeylinux.org/lampstack> Luettu 20.2.2014

28 phpMyAdmin. Verkkosivusto. The phpMyAdmin Project <http://www.phpmyadmin.net/> Luettu 22.2.2014.

29 SASS: CSS with superpowers. Verkkosivusto. Sass. <http://sass-lang.com/> Luettu 22.2.2014.

30 Suzanne, Eric. 2014. Suzy: Power tools for the web. Verkkosivusto.. <http://susy.oddbird.net/> Päivitetty 7.2.2014. Luettu 22.2.2014.

31 Singularity: Grids without limits. Verkkosivusto. Team-sass. <http://singularity.gs/> Luettu 22.2.2014.

32 Breakpoint: Really Simple, Organized, Media Queries with Sass. Verkkosivusto. Breakpoint-sass. <http://breakpoint-sass.com/> Luettu 22.2.2014.

Suppea ketterän kehityksen mukainen vaatimusmäärittely

Ominaisuuslista

Kehitysjakso	Tehtävä	Rooli	Käyttäjätarina
2	1	Ylläpitäjä	… voi kirjautua sivustolle ja selata luotua sisältöä
2	2	Ylläpitäjä	… voi lisätä, muokata, julkaista ja poistaa uutisia
3	3	Ylläpitäjä	… voi lisätä, muokata, julkaista ja poistaa palveluita
4	4	Ylläpitäjä	… voi tarkistaa lähetetyt yhteydenottopyynnöt sivustolta
2	5	Ylläpitäjä	… voi tarkistaa järjestelmän raportointia
1-4	6	Käyttäjä	… voi navigoida sivuilla helposti ja vaivatta
2	7	Käyttäjä	… näkee etusivulla koosteen uusimmista uutisista
3	8	Käyttäjä	… näkee palvelut -sivulla listauksen yrityksen palveluista
3	9	Käyttäjä	… saa lisätietoa palvelusta navigoimalla palvelusta kertovalle alasivulle
4	10	Käyttäjä	… voi lähettää yhteydenottopyynnön.
1	11	Käyttäjä	… saa tietoa yrityksestä etusivulta
-	12	Käyttäjä	… voi selata helposti sivustoa laiteriippumattomasti

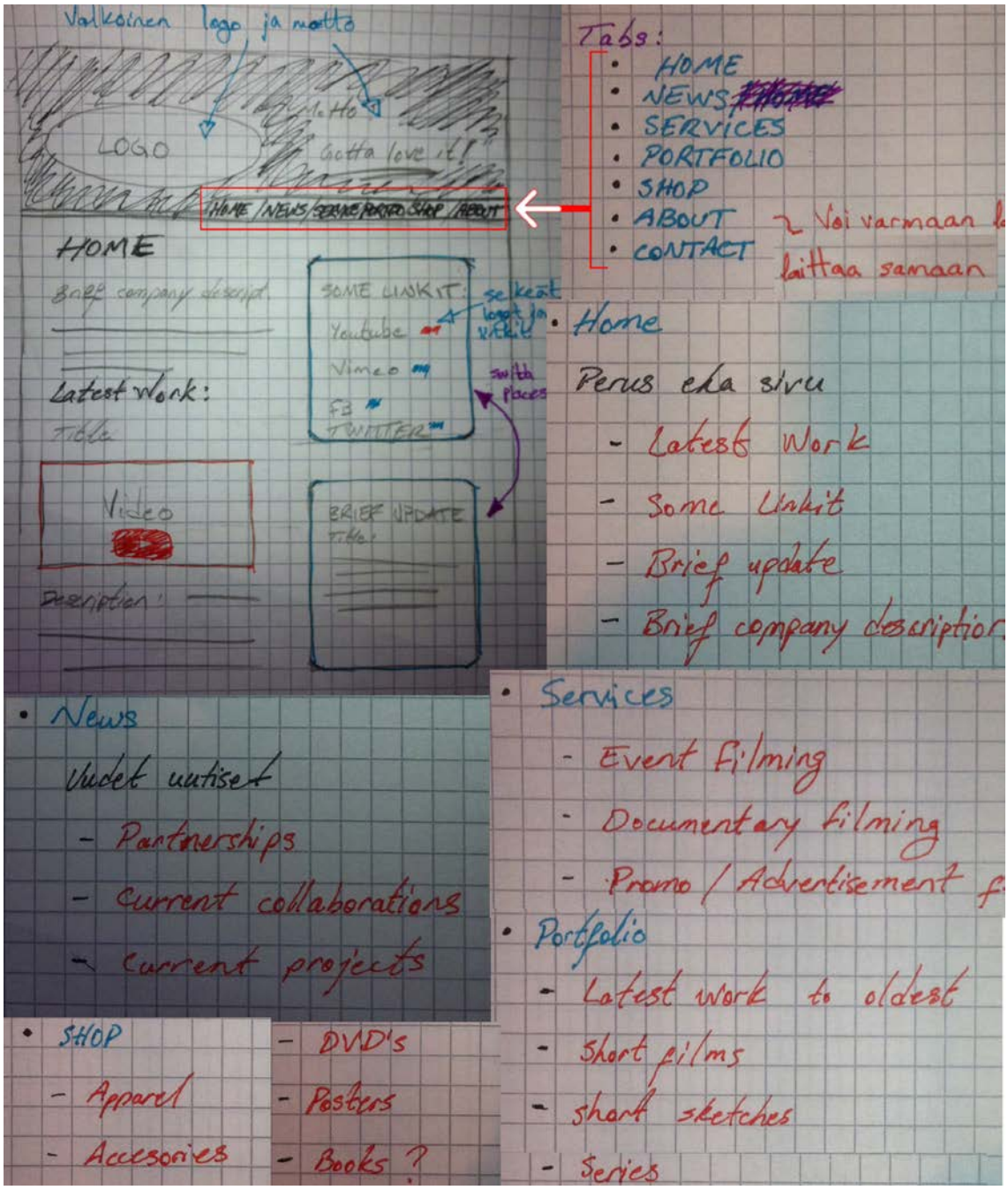
Kehityskaari

- Suunnitellaan sivuston käyttöliittymä ja rakennetaan rautalankamalli
- Rakennetaan rautalankamallin perusteella sivuston ulkoasu ja teema
- Asennetaan kehitysympäristö ja tarvittavat komponentit
- Rakennetaan sivuston teema ja pohjamalli
- Rakennetaan uutisten listaus etusivulle, navigointimainen listaus kaikille sivuille, joiden tyyppi on ”uutinen”. Luodaan mahdollisuus lisätä, muokata ja poistaa uutisia.
- Rakennetaan palvelulistaus ja mahdollisuus lisätä, muokata ja poistaa palveluita.
- Rakennetaan mahdollisuus vuorovaikutukseen eli yhteydenottolomake

Julkaisusuunnitelma

- 14.2. - Julkaistaan sivuston etusivu
- 21.2. - Julkaistaan uutisosio
- 28.2. - Julkaistaan palvelut –osio
- 7.3. - Julkaistaan mahdollisuus lähettää yhteydenottopyyntö

Asiakkaan toimittama ohjeistus sivustokehitykseen



Drush paketinhallintatiedosto

```
; Generated makefile from http://drushmake.me
; ------------
; Each makefile should begin by declaring the core version of Drupal that all
; projects should be compatible with.

core = 7.x

; Every makefile needs to declare its Drush Make API version. This version of
; drush make uses API version `2`.

api = 2

; In order for your makefile to generate a full Drupal site, you must include
; a core project. This is usually Drupal core, but you can also specify
; alternative core projects like Pressflow. Note that makefiles included with
; install profiles *should not* include a core project.

; Drupal 7.x. Requires the `core` property to be set to 7.x.
projects[drupal][version] = 7

; Modules
; --------
projects[admin_menu][version] = 3.0-rc4
projects[admin_menu][type] = "module"
projects[admin_menu][subdir] = "contrib"
projects[module_filter][version] = 2.0-alpha2
projects[module_filter][type] = "module"
projects[module_filter][subdir] = "contrib"
projects[ctools][version] = 1.4
projects[ctools][type] = "module"
projects[ctools][subdir] = "contrib"
projects[context][version] = 3.2
projects[context][type] = "module"
projects[context][subdir] = "contrib"
projects[devel][version] = 1.4
projects[devel][type] = "module"
projects[devel][subdir] = "contrib"
projects[email][version] = 1.3
projects[email][type] = "module"
projects[email][subdir] = "contrib"
projects[imce][version] = 1.8
projects[imce][type] = "module"
projects[imce][subdir] = "contrib"
projects[backup_migrate][version] = 2.8
projects[backup_migrate][type] = "module"
projects[backup_migrate][subdir] = "contrib"
```

```
projects[custom_breadcrumbs][version] = 2.0-alpha3
projects[custom_breadcrumbs][type] = "module"
projects[custom_breadcrumbs][subdir] = "contrib"
projects[front][version] = 2.4
projects[front][type] = "module"
projects[front][subdir] = "contrib"
projects[globalredirect][version] = 1.5
projects[globalredirect][type] = "module"
projects[globalredirect][subdir] = "contrib"
projects[google_analytics][version] = 1.4
projects[google_analytics][type] = "module"
projects[google_analytics][subdir] = "contrib"
projects[libraries][version] = 2.2
projects[libraries][type] = "module"
projects[libraries][subdir] = "contrib"
projects[menu_block][version] = 2.3
projects[menu_block][type] = "module"
projects[menu_block][subdir] = "contrib"
projects[ckeditor][version] = 1.13
projects[ckeditor][type] = "module"
projects[ckeditor][subdir] = "contrib"
projects[imce_wysiwyg][version] = 1.0
projects[imce_wysiwyg][type] = "module"
projects[imce_wysiwyg][subdir] = "contrib"
projects[views][version] = 3.7
projects[views][type] = "module"
projects[views][subdir] = "contrib"
projects[webform][version] = 3.20
projects[webform][type] = "module"
projects[webform][subdir] = "contrib"
projects[xmlsitemap][version] = 2.0
projects[xmlsitemap][type] = "module"
projects[xmlsitemap][subdir] = "contrib"
projects[context_omega][subdir] = contrib
projects[metatag][subdir] = contrib

; Themes
projects[] = omega

; Libraries
libraries[jquery][download][type] = "file"
libraries[jquery][download][url] =
"https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"
libraries[jqueryui][download][type] = "file"
libraries[jqueryui][download][url] =
"https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/jquery-ui.min.js"
```