Riku Seittenranta

Räätälöity sisällönhallintajärjestelmä Louhi Net Oy:lle

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
29.4.2012

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Riku Seittenranta Räätälöity sisällönhallintajärjestelmä Louhi Net Oy:lle 34 sivua 29.4.2012
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Ilpo Kuivanen

Insinöörityössä oli tavoitteena kehittää räätälöity web-pohjainen sisällönhallintajärjestelmä Louhi Net Oy:lle. Järjestelmän tarkoituksena on mahdollistaa Louhi Net Oy:n IT-infrastruktuuriin liittyvien tietojen ylläpito ja hallinta. Toteutettava sisällönhallintajärjestelmä tulee korvaamaan ongelmalliseksi havaitun nykyisen järjestelmän.

Työssä luotiin määritykset uudelle sovellukselle aikaisemman järjestelmän käyttötapauksia ja -kokemuksia hyödyntäen. Näiden määritysten pohjalta suunniteltiin järjestelmä, joka soveltuu paremmin käyttötarkoituksiinsa ja antaa hyvän pohjan jatkokehitykselle.

Järjestelmä toteutettiin käyttäen Python-ohjelmointikieltä sekä Django-websovelluskehystä.

Avainsanat	python, django, CMS

Metropolia

Abstract

Author(s) Title Number of Pages Date	Riku Seittenranta Customized Content Management System for Louhi Net Oy 34 pages 29 April 2012
Degree	Banchelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Ilpo Kuivanen, Senior Lecturer

The purpose of this Banchelor's thesis was to develop a customized web-based content management system for Louhi Net Oy. The purpose of the system is to enable the administration and management of Louhi Net Oy's IT infrastructure information. The newly developed system will serve as a replacement for the current system which had been found to be problematic.

The study was executed by drawing on use cases and experiences with the previous system to identify appropriate software requirements. The new system was then designed according to these requirements so that it will be better suited for the use cases and will provide a solid foundation for further development.

The system was implemented using the Python programming language and the Django web framework.

Keywords	python, django, CMS

Sisällys

Lyhenteitä ja käsitteitä

BSD	BSD-lisenssi. Vapaa ohjelmistolisenssi, joka antaa oikeuden käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmistoa ja sen lähdekoodia. Sallii käytön suljetun lähdekoodin ohjelmistoissa.
CSS	Cascading Style Sheets. Tyylimerkintäkieli, joka mahdollistaa WWW-sivujen ulkoasun määrittelyn.
DRY	Don't Repeat Yourself. Ohjelmistosuunnitteluperiaate, jolla pyritään välttämään uudelleen toistuvaa lähdekoodia.
GPL	GNU General Public License. Vapaa ohjelmistolisenssi, joka antaa oikeuden käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmistoa ja sen lähdekoodia. Lisenssi edellyttää, että samat vapaudet säilyvät myös ohjelmiston muunnelluissa versioissa.
HTML	HyperText Markup Language. Standardoitu merkintäkieli, jolla voidaan määritellä WWW-sivuja.
IDE	Integrated Development Environment. Ohjelmointiympäristö, joka sisältää ohjelmistokehityksessä tarvittavia toimintoja.
MIT	MIT-lisenssi. Vapaa ohjelmistolisenssi, joka antaa oikeuden käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmistoa ja sen lähdekoodia. Sallii käytön suljetun lähdekoodin ohjelmistoissa.
MVC	Model-View-Controller. Ohjelmistoarkkitehtuuri, joka mahdollistaa käyttöliittymän erottamisen sovelluslogiikasta.
ORM	Object-relational mapping. Menetelmä, jolla abstrahoidaan olio-pohjaisen järjestelmän tietomallit yhteensopiviksi relaatiotietokannan kanssa.

Public domain
: Termi, jota käytetään kuvaamaan yleiseen käyttöön tarkoitettuja teoksia. Public domainin alaisen teoksen tekijä luopuu oikeuksistaan teokseen.

Regex
: Regular expression. Lauseke, joka määrittelee säännöllisen merkkijoukon.

Unicode
: Merkistöjärjestelmä, jossa jokaisella merkillä on oma merkkikoodi. Unicoden avulla kaikkien kielten käyttämien merkkien käyttäminen on mahdollista yhdellä merkistöllä.

Sovelluskehys
: Ohjelmisto, joka muodostaa pohjan sen päälle toteutettavalle ohjelmistolle. Sovelluskehys on sovelluskehitystä nopeuttava apuväline.

XML
: Extensible Markup Language. Merkintäkieli, jolla tiedon merkitys on kuvattavissa tiedon sekaan.

1 Johdanto

Insinöörityön aiheena on räätälöity sisällönhallintajärjestelmä Louhi Net Oy:lle. Aloin työskennellä Louhi Net Oy:ssä syksyllä 2010 ensimmäisen harjoitteluni merkeissä. Työskennellessäni Louhi Net Oy:n palveluksessa paljon käytetyn sisällönhallintajärjestelmän rajoitteet ja ongelmat nousivat säännöllisesti esiin. Tästä syntyi idea toteuttaa uusi räätälöity sisällönhallintajärjestelmä, jonka suunnittelussa otettaisiin huomioon nykyisen ratkaisun ongelmakohdat ja Louhi Net Oy:n erityistarpeet.

Louhi Net Oy:llä ei ollut erityistä vaatimusta sovelluksen toteutukseen käytettävistä tekniikoista, minkä takia sain itse valita parhaaksi näkemäni työkalut sovelluksen toteutukseen. Päädyin valitsemaan toteutusalustaksi Django-websovelluskehyksen, johon olin tutustunut jo aikaisempien projektieni yhteydessä. Valintaan vaikutti myös se, että toteutettava sovelluskokonaisuus tulisi olemaan varsin laaja. Django-websovelluskehykseen löytyy lukuisia valmiita sovelluksia, joita yhdistelemällä on mahdollista luoda suuriakin kokonaisuuksia nopeasti.

Tässä insinöörityössä tullaan tarkastelemaan toteutetun sovelluksen hyödyntämiä tekniikoita ja toteutukseen käytettyjä työvälineitä. Työssä käydään läpi myös toteutettuun sovellukseen vaikuttaneita suunnitteluperiaatteita, sekä toteutuksen rakennetta ja käyttöliittymäratkaisuja. Lopuksi pohdin myös insinöörityön onnistumista ja jatkokehitystarpeita.

2 Lähtökohdat ja määrittely

2.1 Louhi Net Oy

Louhi Net Oy on suomalainen hosting-palveluja tarjoava yritys. Louhi Net Oy:n palvelutarjontaan kuuluvat muun muassa webhosting-, sovellus- ja palvelinpalvelut. Webhosting-palveluihin kuuluvat verkkotunnukset eli domainit ja webhotellit, jotka mahdollistavat verkkosivujen laittamisen internetiin sekä sähköpostin omalla verkkotunnuksella. Sovelluspalveluina Louhi tarjoaa muun muassa Exchange ja Sharepoint -palveluita. [37.]

Louhi Net Oy:n palvelinpalvelut sisältävät moneen käyttöön soveltuvia virtuaalipalvelimia sekä fyysisiä palvelimia. Palvelimille on saatavilla erilaisia ylläpitoratkaisuja, jotka mahdollistavat muun muassa palvelimen ympärivuorokautisen seurannan ja vikatilanteisiin reagoinnin. Louhi Net Oy:n palvelinkeskukset sisältävät satoja liiketoimintakriittisiä verkkopalveluita, sekä yli 13 000 webhotellia. [37.]

2.2 Nykyinen järjestelmä

Nykyinen järjestelmä on Louhi Net Oy:n infrastruktuurin keskeisimpiä sovelluksia. Järjestelmää käytetään muun muassa palvelin- ja verkkotietojen, ohjeistuksien sekä prosessikuvausten ylläpitoon. Suuri osa Louhi Net Oy:n henkilökunnasta käyttää järjestelmää päivittäisissä työtehtävissään, minkä takia järjestelmä onkin varsin kriittinen osa Louhi Net Oy:n sisäisen tiedonhallintaa.

Nykyinen järjestelmä perustuu JSPWiki-nimiseen wiki-sovellukseen, jonka päälle on rakennettu sisällönhallintajärjestelmä käyttäen hyväksi ennalta sovittuja konventioita sekä uudelleen käytettäviä sivupohjia. Sovelluksen käyttöönotossa oli päädytty ratkaisuun, jossa sovelluksen ylläpitämät tiedot tallennetaan paljaassa tekstimuodossa tiedostohierarkiaan niin, että yksittäinen tiedosto vastaa yhtä sivua. Sivuja on mahdollista linkittää toisiinsa wiki-

merkintätekstin hyperlinkkitageilla, joilla toiseen sivuun on mahdollista luoda linkki tietämällä halutun kohdesivun nimi.

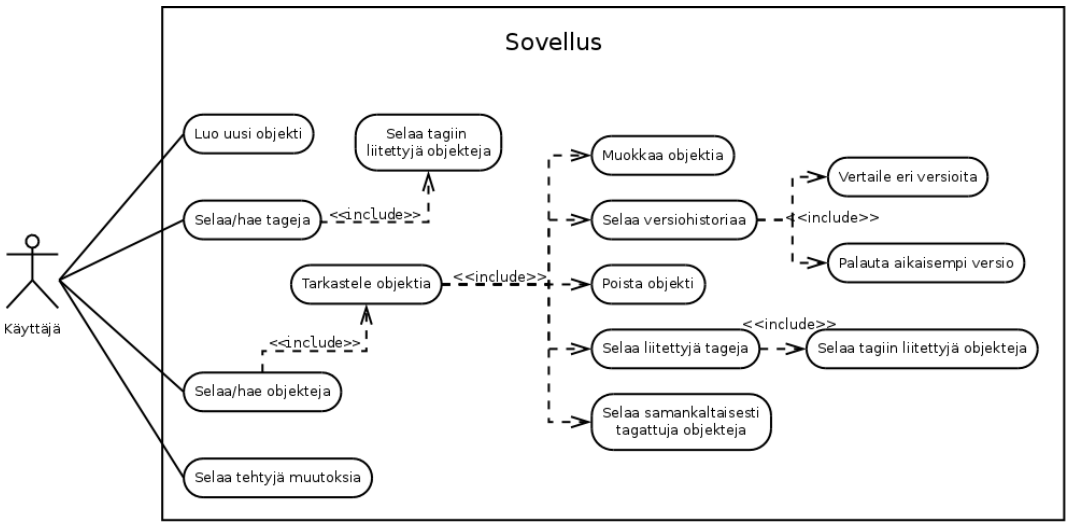
Järjestelmän on todettu kärsivän tiedon yhteneväisyyden ja ajantasaisuuden puutteesta. Tallennetut tiedot ovat paikoin ristiriitaisia, koska muutoksia ei ole aina tehty kaikkiin asianmukaisiin paikkoihin. Toisinaan uusia tietoja ei ole muistettu kirjata lainkaan.

2.3 Sovelluksen kuvaus

Toteutettava sovellus on web-pohjainen järjestelmä, jossa ylläpidetään Louhi Net Oy:n sisäisiä dokumentteja ja ohjeistuksia sekä tietoja infrastruktuurista, kuten verkoista, palvelimista ja päätelaitteista. Sovelluksen vaatimusmäärittely koostettiin Louhi Net Oy:n työntekijöiden kokemuksista nykyisestä järjestelmä. Sovelluksen suunnittelussa tullaan kiinnittämään erityistä huomiota nykyisen järjestelmän ongelmakohtiin, jotta samoilta ongelmilta vältyttäisiin uuden järjestelmän kanssa.

Tässä insinöörityössä objektilla tarkoitetaan mitä tahansa sovelluksen hallitsemaa yksittäistä tietokokonaisuutta. Esimerkki kyseisestä objektista on yksittäinen dokumentti, palvelin tai verkko. Käyttäjän tulee pystyä selaamaan tai hakemaan eri tyyppisiä objekteja. Löytäessään haluamansa käyttäjän on mahdollista tarkastella tai muokata objektin sisältöä. Käyttäjän tulee myös pystyä hakemaan objektiin liittyviä toisia objekteja. Kuvan 1 käyttötapauskaavio havainnollistaa käyttäjän mahdollisia toimia. Sovelluksen tavanomainen käyttäjä on kuka tahansa Louhi Net Oy:n työntekijä, jolla on käyttäjätunnukset sovellukseen.

Objekteihin tulee lisäksi pystyä lisäämään asiasanoja eli tageja niin, että asiasanan perusteella käyttäjän on mahdollista löytää kaikki asiasanaan liitetyt objektit. Objektin luoneesta ja viimeksi muokanneesta käyttäjästä pidetään kirjaa, kuten myös luonnin ja viimeisimmän muokkauksen ajankohdasta.



Kuva 1. Käyttötapauskaavio.

Dokumentit

Sovelluksella voidaan tallentaa geneerisiä tekstipohjaisia dokumentteja. Dokumentteja voidaan jakaa eri kategorioihin, jotta käyttäjä voi selata vain haluamansa tyyppisiä dokumentteja. Dokumentteja on mahdollista hakea nimen, sisällön, kategorian, luonti- tai muokkausajankohdan perusteella.

Palvelimet ja päätelaitteet

Sovelluksella voidaan tallentaa tietoja palvelimista ja päätelaitteista. Näistä täytyy löytyä muun muassa seuraavat tiedot:

- nimi
- kuvaus
- sijainti
- asennetut ohjelmistot ja ohjelmistoversiot
- laitteistotiedot
- verkkosovittimet.

Palvelimia ja päätelaitteita tulee olla mahdollista ryhmitellä käyttäjän valitsemiin ryhmiin.

Verkot

Sovelluksella voidaan tallentaa tietoja Louhi Net Oy:n ylläpitämistä verkoista ja niiden osoitteista. Sovelluksen tulee pystyä kertomaan käyttäjälle, mitä palvelimia tai päätelaitteita mihinkin verkkoon on kytkettynä. Verkoista tulee tallentaa niiden oleellisimmat asiat kuten verkon osoite ja aliverkonpeite.

Sijainnit

Sovelluksella voidaan tallentaa tietoja erilaisista sijainneista, kuten esimerkiksi konesaleista. Sijaintitieto voi sisältää nimen lisäksi myös osoite- ja koordinaattitiedon. Sovelluksen tulee pystyä listaamaan sijaintiin liitetyt palvelimet tai päätelaitteet.

2.4 Sovelluksen vaatimukset

Sovellukseen tallennettavat tiedot ovat alustavasti manuaalisesti päivitettäviä tietoja. Sovellukseen tulee myöhemmin olla mahdollista automatisoida tietojen päivitys hakemalla tietoja automaattisesti Louhi Net Oy:n muista järjestelmistä erilaisten integraatioiden avulla.

Toiminnalliset vaatimukset

- Käyttäjä voi selata/hakea objekteja.
- Käyttäjä voi lisätä, muokata ja poistaa objekteja.
- Käyttäjä voi helposti nähdä/hakea objektiin liittyviä toisia objekteja.
- Käyttäjä voi selata objektien vanhempia versioita ja palauttaa objekti vanhempaan versioon.

Ei-toiminnalliset vaatimukset

- Sovelluksen tulee olla helposti omaksuttavissa uusille käyttäjille.
- Sovelluksen täytyy tukea useita samanaikaisia käyttäjiä.
- Sovelluksen tulee toimia yhtä nopeasti käyttäjämäärästä riippumatta, kun käyttäjä määrä pysyy alle 30.

- Sovellukseen tulee olla mahdollista lisätä myöhemmin erilaisia integraatioita.
- Sovelluksen tulee olla käytettävissä mahdollisimman monella eri laitteella ja internetselaimella.

3 Tekniikat

3.1 HTML

HTML eli HyperText Markup Language on merkintäkieli WWW-sivujen määrittelyyn. HTML-kielestä löytyy useita peruselementtejä, joista WWW-sivuja rakennetaan. Internetselaimet tulkitsevat näitä elementtejä ja muodostavat niiden perusteella käyttäjälle näkymän WWW-sivusta. HTML-kieli tukee tekstin lisäksi muun muassa kuvia ja upotettuja objekteja, kuten esimerkiksi Flash-tiedostoja ja Java-appletteja. HTML-standardi on World Wide Web Consortiumin (W3C) hallinnoima. [23.]

Toteutetussa sovelluksessa käytettiin XHTML-kielen 1.0 Strict -versiota. XHTML on XML-muotovaatimukset täyttävä versio HTML-kielestä. Merkittävin ero XHTML:n ja HTML:ään välillä on XHTML:n tiukemmat muotosäännöt. [38.]

3.2 CSS

CSS eli Cascading Style Sheets on tyylimerkintäkieli. CSS:n avulla voidaan kirjoittaa tyyliohjeita, jotka kuvaavat miltä esimerkiksi HTML-sivun tulisi näyttää. CSS tarjoaa mahdollisuuden muokata WWW-sivujen ulkonäkyä koskematta itse sivun sisältöön tai rakenteeseen. Kuten HTML, myös CSS on W3C:n hallinnoima [30]. [24 s. XXII.]

Toteutetussa sovelluksessa käytettiin CSS-kielen yleisimmin käytettyjä ominaisuuksia, jotka löytyvät versiosta 2.1. CSS 2.1 -määritys kuuluu W3C:n tämän hetkisiin suosituksiin [31]. Projektissa käytettiin lisäksi CSS 3:sta löytyviä uusia ominaisuuksia, kuten liukuväritaustoja sekä pyöristettyjä kulmia. Nämä CSS 3:n tuomat ominaisuudet toimivat progressiivisena kohennuksena niitä tukevilla selaimilla.

3.3 Javascript ja jQuery

Javascript on ohjelmointikieli, jonka yleisin käyttötarkoitus on tuoda WWW-sivuille lisäarvoa muun muassa kehittyneempien käyttöliittymien ja dynaamisen sisällön kautta. Internetselain suorittaa Javascript-lähdekoodin paikallisesti, mikä mahdollistaa tavallisia HTML-sivuja nopeamman reagoinnin käyttäjän toimiin. Javascript-lähdekoodi sijoitetaan HTML-dokumenttiin script-elementin sisälle. Internetselaimen ladatessa sivun se suorittaa script-elementtien sisällä olevan Javascript-lähdekoodin. [25.]

JQuery on Javascript-kielellä toteutettu ohjelmistokirjasto, joka on suunniteltu yksinkertaistamaan selainohjelmointia. JQuery tarjoaa valmiita funktioita usein käytettyjen toiminnallisuuksien, kuten animaatioiden toteuttamiseen sekä käyttäjän toimiin reagointiin. JQuery on vastaavista kirjastoista ylivoimaisesti suosituin ja käytetyin. JQuery on avoimen lähdekoodin vapaa ohjelmistokirjasto, joka on tarjolla MIT ja GPLv2 -lisensseillä. [26.]

Tätä insinöörityötä aloittaessa otettiin käyttöön jQuery-kirjaston sen hetken viimeisin versio 1.6.2. Myöhemmin jQuery päivitettiin käyttämään viimeisintä 1.7.1-versiota, koska muutokset näiden versioiden välillä eivät vaikuttaneet jo toteutettuihin toiminnallisuuksiin.

3.4 Python

Python on korkean tason tulkattava ohjelmointikieli. Se on tunnettu muun muassa selkeästä syntaksistaan, korkean tason tietorakenteista sekä kattavasta standardikirjastostaan. Python on yleiskäyttöinen oliopohjainen ohjelmointikieli, joka tukee useita erilaisia ohjelmointiparadigmoja. Tulkattavuutensa takia Pythonia käytetään usein skriptikielenä, mutta kieli soveltuu moniin muihinkin käyttökohteisiin. [3.]

Python on Guido van Rossumin kehittämä ohjelmointikieli, jonka ensimmäisen version Rossum julkaisi vuonna 1991 [20]. Python sai vaikutteita ABC-kielestä, jonka parissa Rossum oli työskennellyt ennen Python-kielen kehittämistä [21].

Python 2.0 julkaistiin vuonna 2000. Versio toi mukanaan Unicode-tuen, automaattisen roskienkeruutuen sekä list comprehension -ominaisuuden. Automaattinen roskienkeruu on muistinhallintatapa, jossa roskienkerääjä pyrkii vapauttamaan muistista tietoja joihin ei enää viitata [42]. List comprehension on lausekerakenne, joka mahdollistaa listojen luomisen käyttämällä hyväksi toisia listoja [43]. Python-kielen 2.x -sarja on yhä laajalti käytössä ja sen viimeisin versio on 2.7. [20.]

Python 3.0 julkaistiin vuonna 2008. Python 3.0 ei ole yhteensopiva aikaisemman 2.x-sarjan kanssa. Python 3.0 kehitettiin korjaamaan aikaisempien versioiden suunnitteluvirheitä ja poistamaan kielen turhia ominaisuuksia. Vanhempien 2.x-sarjan sovellusten päivittämiseksi 3.x yhteensopiviksi on kehitytetty aputyökalu, joka helpottaa versioiden välisessä siirtymävaiheessa. Lisäksi Python 2.6 -versiosta lähtien käyttäjää varoitetaan lähdekoodin mahdollisista yhteensopivuusongelmista 3.x-sarjan kanssa. [20.]

Tässä insinöörityössä luotu projekti vaatii toimiakseen Python 2.5:n tai uudemman 2.x-sarjan version. Toteutetun sovelluksen vaatima Python-versio määräytyy pitkälti käytetystä Django-websovelluskehyksen versiosta. Tällä hetkellä Django ei esimerkiksi vielä virallisesti tue Python 3:a. Lisäksi Djangon viimeisimmästä 1.4 versiosta poistettiin tuki Python 2.4 -versiolle [39].

3.5 Django

Django on Python-ohjelmointikielellä kirjoitettu websovelluskehys. Django mahdollistaa monimutkaisten ja laajojen tietokantapohjaisten websovellusten toteuttamisen nopealla aikataululla. Tähän tavoitteeseen Django pyrkii painottamalla sovelluksien uudelleenkäytettävyyttä ja DRY-periaatteen tärkeyttä. [4.]

Django sai alkunsa vuonna 2003, kun Adrian Holovaty ja Simon Willison työskentelivät Web-kehittäjinä Lawrence Journal-World -uutislehdessä. Heidän vastuulleen kuului lukuisia suuria sivustoja, joiden kehityksessä vaadittiin usein uusia ominaisuuksia nopeilla aikatauluilla. Django kehittyi vastaa-

maan tarpeeseen luoda kokonaan uusia sivustoja tai uusia ominaisuuksia nopeilla aikatauluilla ja pienellä vaivalla. [5, s. 7]

Django julkaistiin vuonna 2005 avoimen lähdekoodin projektina, saaden nimensä jazz-kitaristi Django Reinhardtin mukaan. Vaikka Djangolla on tänä päivänä kehittäjiä ympäri maailmaa, sen alkuperäiset kehittäjät ovat edelleen tiiviisti mukana ohjailemassa Djangon pääkehityssuuntauksia. [5, s. 7]

Tämän insinöörityön kehitys aloitettiin Djangon versiolla 1.3, joka julkaistiin maaliskuussa 2011 [40]. Djangon viimeisin versio 1.4 julkaistiin maaliskuussa 2012. Django 1.4 sisältää paljon erilaisia parannuksia ja korjauksia. Django 1.4 tulee myös samaan turvapäivityksiä ja bugikorjauksia pidempään, minkä takia se tullaan ottamaan käyttöön vielä ennen sovelluksen varsinaista käyttöönottoa. [39.]

Websovelluskehys

Websovelluskehys on WWW-sovellusten kehitykseen suuntautunut sovelluskehys. Websovelluskehys tarjoaa valmiin pohjan, jonka päälle voi rakentaa omia WWW-sovelluksia. Sovelluskehykset sisältävät yleensä usein käytettyjä ominaisuuksia, kuten tietokantayhteyksien ja sessioiden hallinnan. [21.]

Suuri osa websovelluskehyksistä perustuu MVC-malliin (Model-View-Controller). MVC-mallissa sovelluksen tietokantakerros (M), tiedon esityskerros (V) sekä logiikkakerros (C) erotetaan toisistaan. Sovellusten kerroksien eriyttäminen tekee sovelluksen rakenteesta modulaarisemman ja mahdollistaa muun muassa useiden erilaisten esityskerroksien käyttämisen. [21.]

Sovellusarkkitehtuuri

Djangon sovellusarkkitehtuuri mukailee MVC-mallia sen verran, että se voidaan laskea MVC-mallia käyttäväksi websovelluskehykseksi. MVC-malli jakautuu Djangossa seuraavasti:

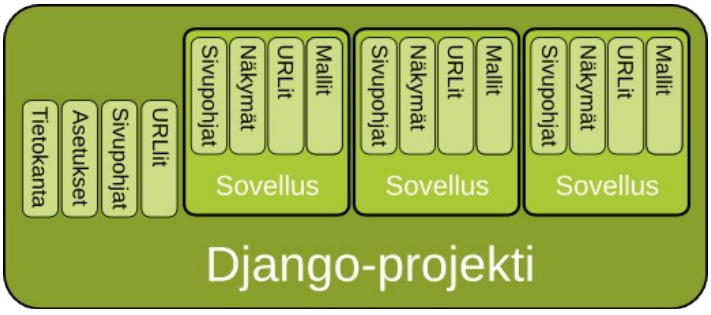
- Tietokantakerros määritellään models.py-tiedostossa, jossa tietokannan taulut määritellään Python-kielen luokkina.

– Esityskerroksena Djangossa toimivat näkymät ja sivupohjat, jotka määrittävät mitä sivulla näytetään ja kuinka se esitetään.
– Logiikkakerroksen osuudesta vastaa Django. Django kutsuu URL-osoitteiden määritysten perusteella sopivaa näkymää. [5, s. 6, 61.]

Djangon sovellusarkkitehtuuria voidaan kutsua myös MTV-mallin (Model-Template-View) mukaiseksi, koska MVC-mallin mukaisen Controller-osan tehtävät suoritetaan Djangossa itsessään. Tällöin esityskerrokseen jäisi MTV-mallin mukaisessa esitystavassa Djangon näkymät, sivupohjien siirtyessä T-osaan (Template). [5, s. 61.]

Django-projektin rakenne

Django-projekti koostuu tavanomaisesti useasta Django-sovelluksesta, jotka yhdessä muodostavat toimivan kokonaisuuden (kuva 2). Yksittäinen Django-sovellus on yleensä luotu mahdollistamaan, jonkin tietyn toiminnallisuuden käyttöönoton projektissa. Django-sovellus sisältää yleensä vähintään tietokantamallien kuvaukset ja näkymiä [5, s.64]. Näiden lisäksi Django-sovelluksella voi olla muun muassa omia URL-määrityksiä sekä sivupohjia. Django-projekti sisältää myös projekti- ja sovelluskohtaisia asetuksia [5, s. 14].



Kuva 2. Havainnollistava kuva Django-projektin rakenteesta.

4 Työkalut

4.1 Eclipse IDE ja PyDev

Sovelluskehittimenä projektin toteutuksessa käytettiin Eclipse IDE:tä, joka on suosittu avoimen lähdekoodin ilmainen sovelluskehitin. Eclipse on laajalti käytetty sovelluskehitin, joka tukee useita eri ohjelmointikieliä kuten Java, C, C++ ja PHP. Eclipsessä otettiin käyttöön lisäksi PyDev-lisäosa, joka tuo sovelluskehittimeen tuen Python-kielelle sekä Django-websovelluskehykselle. Sovelluskehittimen valintaan vaikuttivat pääasiassa aikaisemmat käyttökokemukset Eclipsestä sekä PyDev-lisäosasta. Insinöörityössä käytettiin Eclipse IDE:n versiota 3.7.1 sekä PyDev-lisäosan versiota 2.4.0. [35.]

4.2 Versionhallinta

Versionhallintatyökalut ovat tärkeä osa ohjelmistotuotantoprojektia. Versionhallintatyökaluilla pidetään kirjaa tiedostoihin kohdistuneista muutoksista, niiden ajankohdista ja tekijöistä. Versionhallintatyökalun avulla koko ohjelmistotuotantoprojektin kehityshistoria pysyy tallessa. Yksittäiset tiedostot tai koko projekti, on mahdollista palauttaa tarvittaessa johonkin aiemmista versioista. Nämä työkalut mahdollistavat myös muun muassa tiedostojen eri versioiden vertailemisen ja muutoksien siirtämisen versioiden välillä. [27.]

Tämän projektin versiohallintatyökaluksi valittiin Git, joka on Linus Torvaldsin kehittämä hajautettu versionhallintatyökalu. Git on avoimen lähdekoodin vapaa projekti, joka on lisensoitu GPLv2:n mukaisesti. [28.]

Git:in valinta projektin versiohallintatyökaluksi perustui lähinnä haluun tutustua uuteen ja paljolti kehuttuun vaihtoehtoon jo ennestään tutun Subversionin (SVN) sijaan. Tämän kaltaisessa yhden tekijän projektissa versiohallintatyökalun valinnalla ei loppujen lopuksi ollut suurta merkitystä, koska eri työkalujen väliset erot nousevat esiin vasta isommissa projekteissa.

4.3 Kehitysympäristö

Projektin kehitysympäristössä käytettiin Djangon mukana tulevaa yksinkertaista kehitysvaiheeseen tarkoitettua WWW-palvelinta. WWW-palvelin soveltuu ainoastaan kehityksessä käytettäväksi, sillä sitä ei ole suunniteltu käsittelemään suuria kävijämääriä eikä sen tietoturvallisuutta ole varmennettu. [29.]

Kehitysympäristössä käytettiin myös virtualenv-sovellusta, joka mahdollistaa eristettyjen Python-ympäristöjen luonnin. Virtualenv-sovelluksen avulla projektin käyttämät Python-kirjastot pystytään eristämään muusta käyttöjärjestelmästä. Tämä mahdollistaa haluttujen Python-kirjastoversioiden käyttämisen käyttöjärjestelmästä riippumatta. Eristetyt Python-ympäristöt mahdollistavat myös useita eri Python-kirjastoversioita käyttävien projektien ajamisen samalla palvelimella. [34.]

Kehitysympäristön tietokantaratkaisuna käytettiin tiedostopohjaista SQLite-relaatiotietokantajärjestelmää. SQLite on vaihtoehto monimutkaisemmille relaatiotietokantajärjestelmille kuten MySQL:lle ja PostgreSQL:lle. Yksinkertaisuutensa ja siirrettävyytensä takia SQLite on erinomainen ratkaisu kehitysympäristön tietokantajärjestelmäksi. SQLiten käyttöönotto on nopeaa eikä se vaadi erillisen tietokantamoottorin asennusta. SQLite on vapaa, Public domainin alainen ohjelmistokirjasto. [32, 33.]

5 Suunnittelu

5.1 Sovellukset

Projektin suunnittelussa lähdettiin liikkeelle nykyisen järjestelmän käyttötarkoituksien kartoittamisella. Näistä käyttötarkoituksista eniten käytetyt valittiin ehdokkaiksi omia Django-sovelluksia varten. Nämä Django-sovellukset tulisivat olemaan projektin keskeisiä osasia. Harkinnan jälkeen ensisijaisiksi Django-sovelluksiksi valittiin Dokumentit- ja Infrastruktuuri-sovellukset. Näihin sovelluksiin päädyttiin, koska niihin saatiin sisällytettyä suuri osa käyttötapauksista.

Dokumentit-sovellus tulee olemaan geneerinen monien eri tyyppisten dokumenttien hallintasovellus, johon on mahdollista tarvittaessa lisätä tuki myös erikoisempiin käyttötarkoituksiin tarkoitetuille dokumenteille kuten muutosilmoituksille tai kokouspöytäkirjoille. Dokumenttien muokkaamisen tulee tapahtua jollakin yksinkertaisella merkintäkielellä mieluiten samanlaisella kuin aikaisemmin.

Infrastruktuuri-sovellus tulee olemaan projektin laajin sovellus. Infrastruktuuri-sovelluksen tulee pystyä hallitsemaan palvelin-, päätelaite-, verkko- ja sijaintitietoja. Näiden lisäksi sovelluksen tulee pystyä hallitsemaan näihin liitoksissa olevia tietoja, kuten käytössä olevia ohjelmisto- ja laitteistotietoja sekä verkkosovitintietoja.

5.2 Periaatteet

”Älä pakota minua ajattelemaan”

Steve Krugin ensimmäinen käytettävyyssääntö on yksinkertainen mutta toimiva. Sivuston tulee olla käyttäjälle itsestään selvä. Käyttäjän pitäisi heti ymmärtää sivuston toimintaperiaate ja rakenne. Mitä enemmän sivusto herättää kysymyksiä käyttäjälle, sitä vaikeampi käyttäjän on se sisäistää. [2, s. 11.]

Hankkiudu eroon turhista sanoista

Steve Krugin kolmas käytettävyyssääntö kehottaa hankkiutumaan eroon turhista sanoista. Turhat sanat vievät tilaa sivulta ja antavat sivusta raskaan vaikutelman. Turhien sanojen poistaminen tekee lisätilaa sivun varsinaiselle sisällölle ja tekee sivuista miellyttävämpiä selata. [2, s. 45.]

Johdonmukainen navigaatio

Web-navigaation tarkoituksena on auttaa käyttäjää löytämään etsimänsä ja kertoa käyttäjälle hänen nykyinen sijaintinsa. Sivulta toiselle säilyvä yhtenäinen navigaatio estää käyttäjää eksymästä sivustolla. Navigaatio kertoo käyttäjälle myös, mitä kyseiseltä sivulta pitäisi löytyä. [2, s. 59-60.]

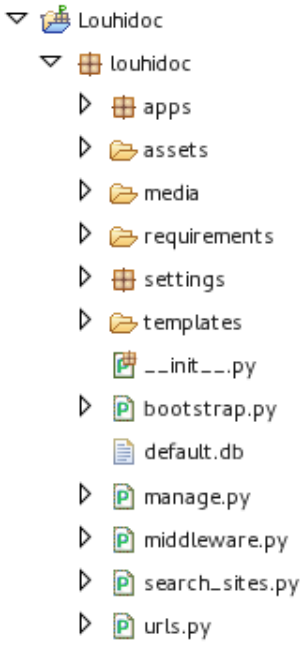
Progressiivinen kohennus (Progressive Enhancement)

Progressiivinen kohennus on web-suunnittelun strategia. Progressiivinen kohennus pyrkii siihen, että käyttäjälle tarjotaan vähintään toimivat perusominaisuudet riippumatta käytettävästä selaimesta tai yhteydestä. Jos kuitenkin käyttäjän yhteys ja selain sen sallivat, niin sivustosta tarjotaan kohennettu versio, joka antaa käyttäjälle paremman käyttökokemuksen. [9.]

6 Toteutus

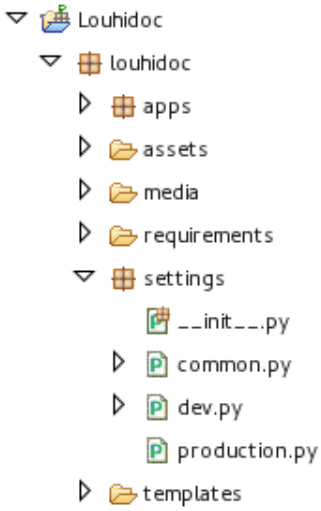
6.1 Hakemistorakenne

Projektin hakemistorakenne noudattaa hyvin pitkälti Djangon oletusprojektirakennetta muutamaa poikkeusta lukuun ottamatta. Projektin asetustiedostojen rakenteeseen tehtiin muutoksia ylläpidollisista syistä. Näihin poikkeusratkaisuihin päädyttiin, jotta projekti säilyisi mahdollisimman helposti ylläpidettävänä ja selkeänä (kuva 3).



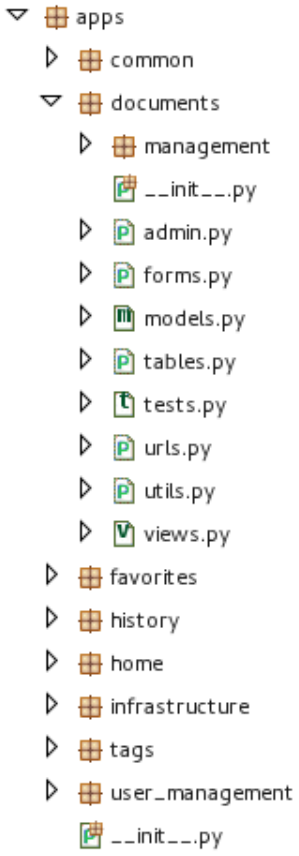
Kuva 3. Hakemistorakenne. Projektin juuri.

Projektin asetukset hajautettiin erillisiksi tiedostoiksi omaan hakemistoonsa yhden tiedoston sijaan (kuva 4). Asetustiedostojen hajauttaminen mahdollistaa erilaisten asetuksien käyttämisen tuotanto- ja kehitysympäristöissä niin, että kaikki asetustiedostot säilytetään versiohallinnassa. Lisäksi hajauttaminen mahdollistaa yhteisten asetusten määrittelemisen eri ympäristöille pienentäen näin asetustiedostojen kokoa ja parantaen luettavuutta. [6.]



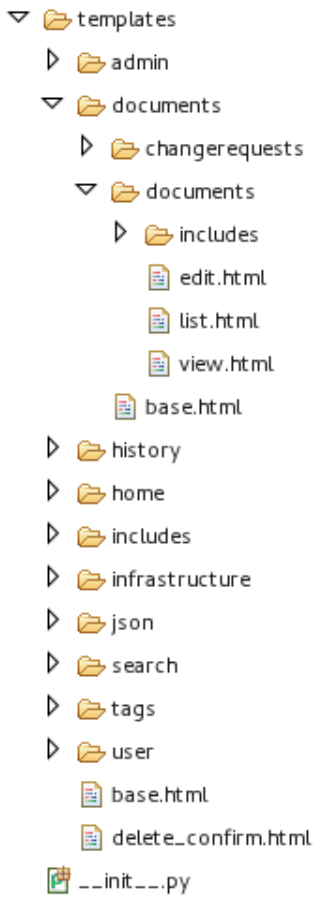
Kuva 4. Hakemistorakenne. Asetustiedostot.

Projektin itsetoteutetut sovellukset sijoitettiin projektin juureen luotuun apps-hakemistoon (kuva 5). Tähän ratkaisuun päädyttiin, jotta sovellukset olisivat selkeästi oma kokonaisuutensa projektin muista hakemistoista.



Kuva 5. Projektin sovellukset.

Projektin sivupohjat sijaitsevat projektin juuressa sijaitsevassa templates-hakemistossa (kuva 6). Sivupohjat on jaoteltu sivuston valikkohierarkiaa mukaillen. Sivupohjahakemiston juuressa on myös includes-hakemisto, joka pitää sisällään uudelleen käytettäviksi suunniteltuja sivupohjia, joita voidaan sisällyttää muihin sivuihin. Includes-hakemistoja löytyy myös syvemmältä hierarkiasta. Nämä hakemistot sisältävät erikoistuneita includes-tiedostoja, joita ei ole syytä säilyttää sivupohjahakemiston juuren includes-hakemistossa.



Kuva 6. Sivupohja-hakemisto.

6.2 Itsetoteutetut Django-sovellukset

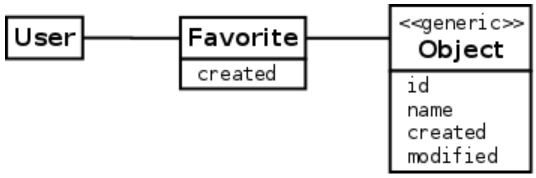
Projektia varten toteutettiin kahdeksan erilaista Django-sovellusta. Sovellusten koko vaihtelee pienistä ja yksinkertaisista isoihin ja monimutkaisiin. Joukossa on myös sovelluksia, joiden ainoana tarkoituksena on huomaamattomasti paketoida kolmannen osapuolen Django-sovellus muuhun projektiin. Pienimmät sovellukset sisältävät muutaman näkymän eivätkä välttämättä ollenkaan tietokantakerroksen kuvausta. Suurikokoisin Infrastruktuurisovellus sen sijaan koostuu yli kymmenestä näkymästä ja tietomallin kuvauksesta.

Etusivu

Etusivu-sovellus on yksinkertainen sovellus, joka vastaa etusivun sisällöstä. Etusivu-sovellus sisältää yhden näkymän, joka näyttää käyttäjälle tietoja viimeaikaisista muutoksista ja uusimmista suosikeista. Näkymä hakee myös käyttäjän viisi viimeiseksi lisäämäänsä suosikkia. Etusivu-sovelluksen on tarkoitus tarjota käyttäjälle nopea katsaus järjestelmän viimeaikaisiin tapahtumiin.

Suosikit

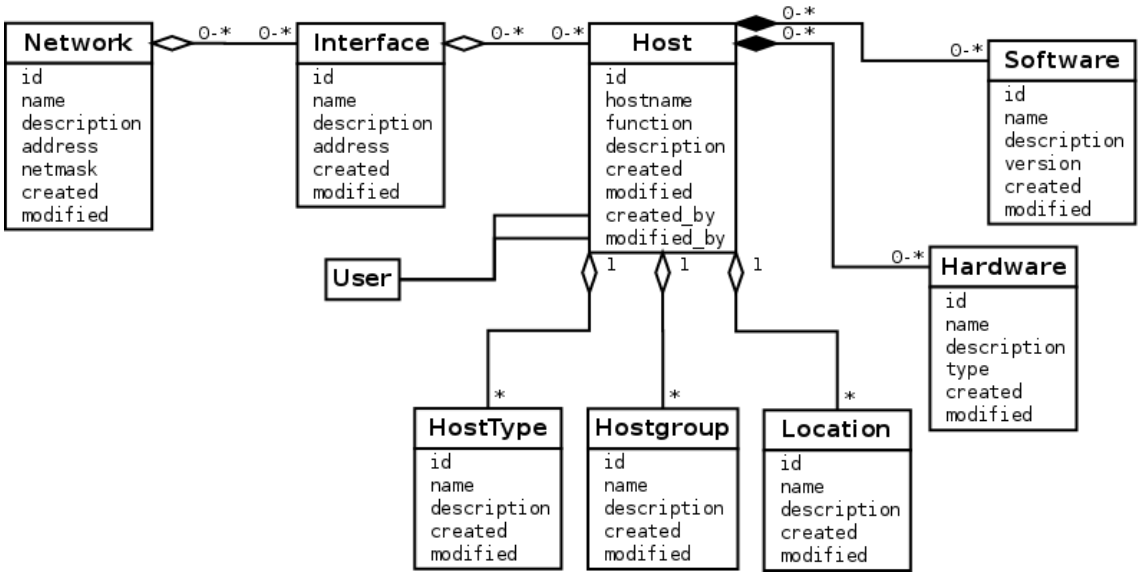
Suosikit-sovellus mahdollistaa käyttäjäkohtaisien suosikkiobjektien valinnan. Suosikit-sovellus sisältää näkymät suosikkien listaamiseksi sekä objektin suosikkitilan vaihtamiseksi ja tarkistamiseksi. Kuvassa 7 on esitetty pelkistetty versio Suosikit-sovelluksen tietomallien luokkakaaviosta.



Kuva 7. Pelkistetty luokkakaavio Suosikit-sovelluksen tietomallien luokista.

Infrastruktuuri

Infrastruktuuri-sovellus on kaikista projektin sovelluksista laajin ja monimutkaisin. Infrastruktuuri-sovellus mahdollistaa muun muassa palvelinten, päätelaitteiden, verkkojen ja sijaintien tietojen hallinnan. Sovelluksen toteutuksessa palvelimet ja päätelaitteet yhdistettiin yhteen tietomalliin niiden yhtenevien sisältövaatimusten johdosta. Kuvassa 8 on esitetty pelkistetty versio Infrastruktuuri-sovelluksen tietomallien luokkakaaviosta.



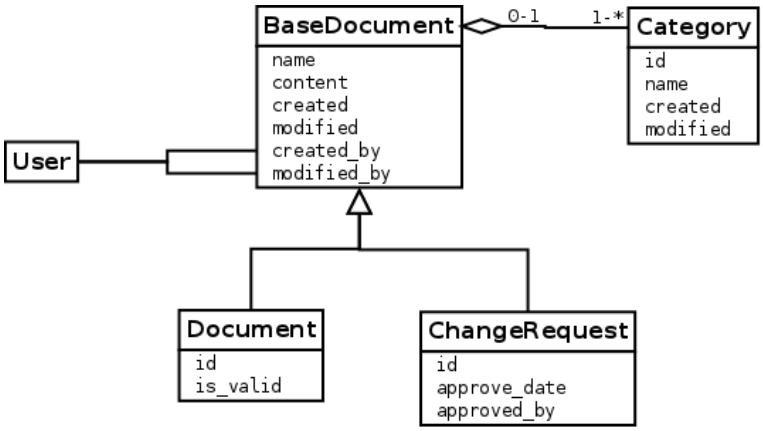
Kuva 8. Pelkistetty luokkakaavio Infrastruktuuri-sovelluksen tietomallien luokista.

Palvelimia ja päätelaitteita tarkastellessa esitetään myös niihin liittyvät ohjelmisto- ja laitteistotiedot sekä verkkosovittimet. Nämä tiedot ovat suoraan muokattavissa kyseisen palvelimen tai päätelaitteen tietoja muokatessa. Palvelimia ja päätelaitteita on myös mahdollista ryhmitellä vapaavalintaisiin ryhmiin. Näin pystytään laittamaan esimerkiksi tietyn asiakkaan palvelimet yhteen ryhmään, jolloin ne kaikki ovat yhdestä paikasta löydettävissä. Verkkoja ja sijainteja tarkastellessa esitellään objektin tietojen lisäksi myös siihen liitoksissa olevat palvelimet ja päätelaitteet.

Dokumentit

Dokumentit-sovellus mahdollistaa geneeristen tekstipohjaisten sivujen luonnin. Sovelluksen mahdollisiin käyttökohteisiin kuuluu muun muassa ohjeistuksien ja raporttien säilöminen. Dokumentit-sovellukseen on mahdollista lisätä pienellä vaivalla myös erikoistuneempia tietomalleja periyttämällä uusia tietomalleja BaseDocument-luokasta. Dokumentit-sovellus mahdollistaa dokumenttien selaamisen ja hakemisen hakusanoin ja kategorioin.

Dokumentit-sovelluksen tietomallit voidaan periyttää abstraktista BaseDocument-luokasta. BaseDocument-luokka sisältää vain tarpeellisimmat tiedot, kuten dokumentin nimen ja sisällön, kategorian sekä luonti- ja muokkausajat. BaseDocument-luokasta löytyy myös tieto dokumentin luoneesta ja viimeksi muokanneesta käyttäjästä. Kuvassa 9 on esitetty pelkistetty versio Dokumentit-sovelluksen tietomallien luokkakaaviosta.



Kuva 9. Pelkistetty luokkakaavio Dokumentit-sovelluksen tietomallien luokista.

Sivujen varsinainen sisältö tallennetaan tekstipohjaisessa Markdown-formaatissa. Markdown-formaatti on helppo lukuinen merkintäformaatti, joka renderöidään HTML-muotoon sivua katsellessa. Toteutuksessa käytetty Markdown-formatointi käyttää hyväkseen markdown-python -kirjastoa, joka on Python-kielinen toteutus Markdown-kielestä. Markdown-kielen Python-toteutus tukee myös taulukkoja, joita alkuperäinen Markdown-määrittely ei tue. Sivun sisältöä muokatessa Markdown-formaatista generoitua sivua on mahdollista tarkastella esikatselutoiminnolla ennen tallentamista. Markdown-formaatti valittiin, koska JSPWikin käyttämälle merkintäkielelle ei

suoraan löytynyt yhteensopivaa Javascript-pohjaista editoria, joka olisi integroitunut Djangoon ilman suurta lisätyötä. Merkintäkielen vaihtumisen takia kaikki aikaisemmasta järjestelmästä Dokumentit-sovellukseen tuotavat tiedot on ajettava erikseen toteutettavan muuntimen läpi. Tämä saattaa aiheuttaa jonkin verran vaivaa aikaisempaan merkintäkieleen tottuneille käyttäjille.

Dokumenttia tarkastellessa sen oikealle puolelle ilmestyy sivupalkki, josta on nähtävissä dokumentin tietoja, liitettyjä asiasanoja sekä liittyviä dokumentteja. Sivupalkissa on myös linkki dokumentin historia-sivulle, josta dokumentin muokkaushistoria on tarkasteltavissa.

Historia-sovellus

Historia-sovellus mahdollistaa objektien muutoshistorian tarkastelun, vertailun eri versioiden välillä, sekä objektin palauttamisen aiempaan versioon. Historia-sovellus käyttää hyväkseen kolmannen osapuolen django-reversion -sovellusta

Asiasanat eli tagit

Tag-sovellus mahdollistaa asiasanojen selaamisen ja muokkauksen. Asiasanoja on mahdollista etsiä selaamalla, hakemalla hakusanalla tai asiasanapilven kautta. Asiasanaa tarkastellessa käyttäjälle esitetään listaus kaikista objekteista, joihin on liitettynä kyseinen asiasana. Tag-sovellus hyödyntää toteutuksessaan kolmannen osapuolen django-taggit-sovellusta.

Käyttäjähallinta

Käyttäjähallinta-sovellus sisältää toiminnallisuuksia käyttäjien sisään- ja uloskirjaamiseksi sekä autentikoimiseksi. Sovellus on myös helposti laajennettavissa kattamaan esimerkiksi käyttäjäkohtaiset profiilit.

Yleiset

Yleiset-sovellus sisältää yleiskäyttöisiä funktioita, näkymiä ja sivupohjaelementtejä, joita muut sovellukset pystyvät hyödyntämään. Yleiset-sovelluksen tarkoitus on DRY-periaatteen mukaisesti kerätä yhteen geneerisiä lähdekoodipalasia turhan lähdekoodin vähentämiseksi.

6.3 Kolmannen osapuolen Django-sovellukset

Näin laajan sovelluksen toteutus näin lyhyessä ajassa ei olisi ollut mahdollista ilman lukuisia jo toteutettuja ja vapaita Django-sovelluksia. Hyödyntäen tehokkaasti kolmannen osapuolen toteuttamia Django-sovelluksia on mahdollista säästää hyvin paljon aikaa ja vaivaa. Seuraavaksi käymme lyhyesti läpi toteutuksessa käytetyt kolmansien osapuolien toteuttamat Django-sovellukset.

Django-tables2

Django-tables2 on Bradley Ayersin kehittämä sovellus, joka mahdollistaa Djangon ORM:llä haetun datan suoran esittämisen HTML-taulukkomuodossa. Django-tables2 tukee tiedon automaattista sivutusta, sekä sarakkeittain järjestämistä. [10.]

Django-taggit ja liittyvät sovellukset

Django-taggit on Alex Gaynorin kehittämä sovellus, joka mahdollistaa asiasana eli tagi -ominaisuuksien lisäämisen jo olemassa oleviin projekteihin [11]. Django-taggit -sovelluksen lisäksi projektissa otettiin käyttöön Django-taggit-templatetags -sovellus, joka lisää sivupohjaelementtejä helpottamaan asiasanalistauksien esittämistä [12].

Django-markitup

Django-markitup on sovellus, joka mahdollistaa MarkItUp-editorin helpon integroinnin Django-projektiin. MarkItUp-editori on yleiskäyttöinen jQueryllä

toteutettu merkkauskielieditori, joka tukee muun muassa Markdown-, Textile- ja BBCode-formaattia. [14.]

Django-reversion

Django-reversion on sovellus, joka mahdollistaa versionhallintaominaisuuksien liittämisen Django-projekteihin. Sovellus pitää kirjaa tehdyistä muutoksista, ja mahdollistaa objektien palauttamisen aiempiin versioihin. Sovellus mahdollistaa myös poistettujen objektien palauttamisen. [15.]

Django-locking

Django-locking on sovellus, joka estää objektin samanaikaisen muokkaamisen lukitsemalla objektin sitä muokkaavalle käyttäjälle muokkauksen ajaksi. [16.]

South

South on sovellus, joka mahdollistaa Django-sovellusten tietokantojen skeemamigroinnin [17]. Ilman Southin kaltaista työkalua Djangon luomiin tietokantatauluihin ei pysty tekemään muutoksia muokkaamalla Djangon Model-luokkia. Jotta Model-luokkien muutokset pystytään luotettavasti siirtämään myös itse tietokantaan tarvitaan Southin kaltaista työkalua.

Haystack

Django-haystack mahdollistaa modulaarisen haun integroinnin Django-sovelluksiin. Se tukee useita erilaisia hakumoottoreita, joiden vaihtaminen keskenään ei vaadi muutoksia sovelluksiin. [18.]

6.4 Muut kirjastot ja sovellukset

Markdown

Markdown on merkintäkieli, jonka avulla pystyy kirjoittamaan helppolukuisia tekstitiedostoja. Markdownin suunnittelussa on tavoiteltu sitä, että sillä muotoillut tekstitiedostot olisivat julkistuskelpoisia myös sellaisenaan. Markdown on myös Perl-kielellä toteutettu text-to-HTML-käännöstyökalu, joka pystyy muuntamaan Markdown-kielellä muotoillun tekstin HTML-muotoon (kuva 10). Markdown-kielen käännöstyökaluja löytyy myös monille muille ohjelmointikielille. Tässä insinöörityössä käytettiin Python-kielellä tehtyä python-markdown -toteutusta. Markdown on saanut vaikutteita useista muista text-to-HTML-merkintäkielistä, mutta sen suurimpana vaikuttajana on ollut tavanomaisen sähköpostiviestin ulkoasu. [36.]

Kuva 10. Havainnollistava kuva Markdown-formatoidun tekstin muunnoksesta HTML-muotoon.

Django-dynamic-formset

Django-dynamic-formset on jQuery-liitännäinen. Django-dynamic-formset mahdollistaa Djangon tukemien lomakkeiden sisäisten lomakkeiden rivien dynaamisen lisäämisen ja poistamisen ilman sivulatauksia. Tässä insinöörityössä django-dynamic-formset-liitännäistä käytetään muun muassa palvelimiin ja päätelaitteisiin liitettyjen ohjelmisto- ja laitteistotietojen muokkaamisen käyttömukavuuden lisäämiseen. Django-dynamic-formset on vapaa avoimen lähdekoodin sovellus, joka on uuden BSD-lisenssin alainen. [7.]

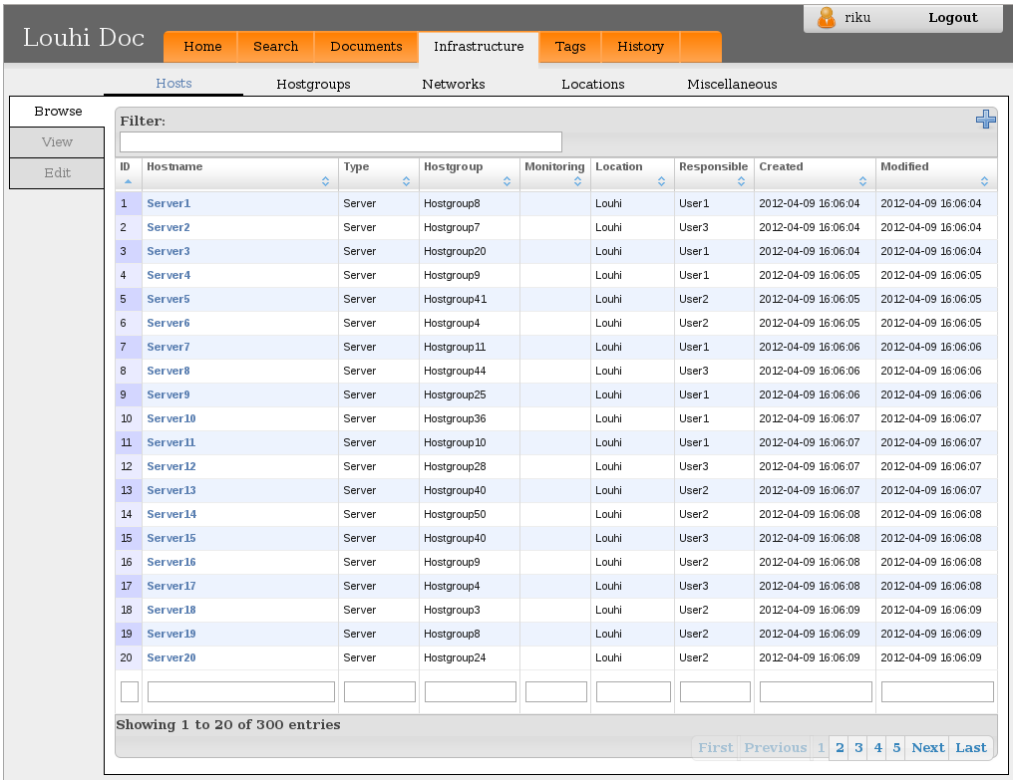
Datatables

Datatables on jQuery-liitännäinen, jonka avulla on mahdollista lisätä kehittyneitä haku- ja selausominaisuuksia mihin tahansa HTML-muotoiseen taulukkoon. Datatables mahdollistaa myös datan hakemisen erilaisista ulkoisista lähteistä. [8.]

Datatables lisättiin projektiin vasta loppuvaiheessa, koska sen tuoma toiminnallisuus kuuluu progressiivisen kohennuksen piirin. Progressiivisen kohentamisen mukaisesti Datatables-liitännäistä ei voitu käyttää ensisijaisena tiedon hakuvälineenä, koska liitännäinen vaatii aktiivisen Javascript-tuen selaimelta. Näin ollen Datatables-liitännäinen otetaan automaattisesti käyttöön vain, jos käyttäjän selain tukee Javascriptia.

6.5 Käyttöliittymä

Sovelluksen käyttöliittymä koostuu yläotsakkeesta sekä varsinaisesta sivunäkymästä. Käyttöliittymä on suunniteltu siten, että selainikkunan kasvaessa sovelluksen varsinainen sivunäkymä laajenee selainikkunan mukana. Tällöin sovellus täyttää aina koko selainikkunan. Kuvassa 11 on kokonaiskuvan antava kuvakaappaus sovelluksen käyttöliittymästä.



Kuva 11. Yleisnäkymä. Infrastructure-välilehti ja hosts-näkymä valittuna.

Yläotsake

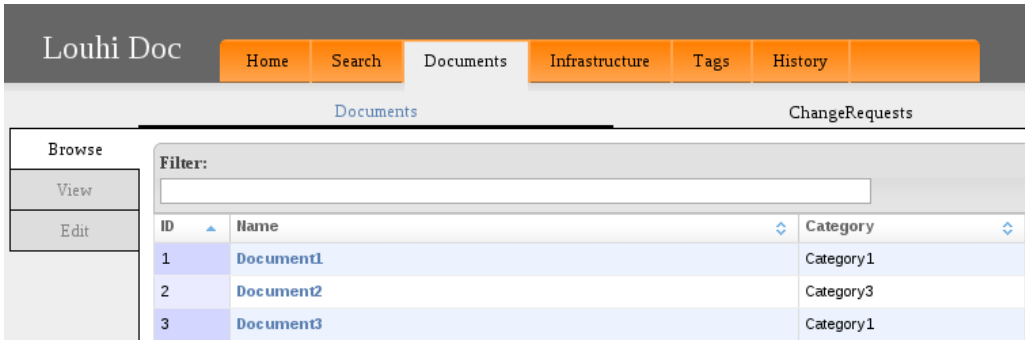
Yläotsake säilyy sisällöltään muuttumattomana käyttäjän siirtyessä sivulta toiselle (kuva 12). Yläotsakkeen vasemmassa reunassa on näkyvissä sovelluksen nimi, jonka oikealla puolella sijaitsee sovelluksen päänavigaatio. Päänavigaatiossa päädyttiin käyttämään tabeja, joita Steve Krug suosittelee kirjassaan [2]. Yläotsakkeen oikeaan reunaan on sijoitettu uloskirjautumispainike sekä tieto sisäänkirjautuneen käyttäjän identiteetistä.

Kuva 12. Yläotsake. Home-välilehti valittuna.

Alemman tason navigaatio ja toimintovalikko

Sovelluksen sivunäkymä koostuu tavanomaisesti päänavigaation alinavigaatiosta, toimintovalikosta sekä varsinaisesta sivun sisällöstä (kuvat 11 ja 13). Alinavigaatiolla pystytään lisäämään yksi syvyyskerros lisää sovelluksen hierarkkiseen rakenteeseen. Toimintovalikon sisältö vaihtelee tietotyyppikoh-

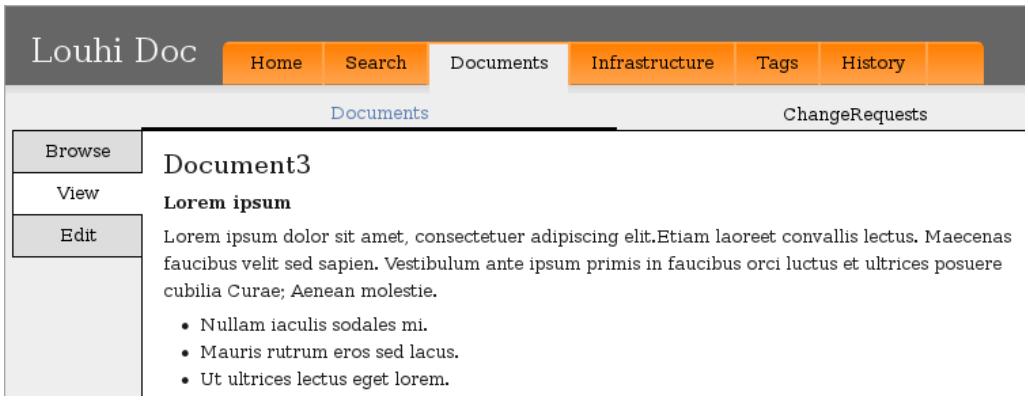
taisesti riippuen siitä, mitkä toiminnot ovat käytännöllisiä millekin tietotyypille. Toimintovalikko toimii yhtä aikaa sekä ilmaisimena, että valitsimena sille, mitä käyttäjän sen hetkisen sivun funktio on. Toimintovalikon yleisimmin käytetyt elementit ovat selaus (Browse), tarkastelu (View) sekä muokkaus (Edit). Toimintovalikko on sijoitettuna sivun varsinaisen sisällön vasemman puolen ylänurkan tasolle.



Kuva 13. Alempi navigaatio sekä toimintovalikko. Dokumenttien selaussivu.

Sivusisältö

Varsinainen sivusisältö vaihtelee tieto- ja toimintotyypeittäin. Selaustoiminnolla käyttäjälle esitellään valittujen tietojen selaus- ja etsintäsivu, jolta käyttäjä pystyy etenemään valitsemalleen tarkastelusivulle. Tarkastelusivulta (kuva 14) käyttäjän on muun muassa mahdollista siirtyä muokkaussivulle tai jollekin monista sisältösivupalkin tarjoamista linkeistä.



Kuva 14. Sivusisältö. Dokumentin tarkastelusivu.

Sisältösivupalkki

Sisältösivupalkki (kuva 15) on käyttöliittymän elementti, joka tulee näkyviin sisällön oikealle puolelle kun käyttäjä siirtyy tarkastelu- tai muokkaussivulle. Sisältösivupalkki näyttää tietoja kyseisestä objektista, siihen liitetyistä asiasanoista ja samantyyppisin asiasanoin merkityistä objekteista. Sivupalkki myös ilmaisee käyttäjälle ajan, jonka objekti on lukittuna sen lukinneelle käyttäjälle. Sisältösivupalkki sisältää myös ilmaisimen objektin Suosikki-tilalle. Ilmaisinta klikkaamalla Suosikki-tilaa on myös mahdollista vaihtaa. Sisältösivupalkki voidaan myös piilottaa tekstialueen koon maksimoimiseksi.

Kuva 15. Sisältösivupalkki.

6.6 Jatkokehitys

Projektin jatkokehityksessä tullaan seuraavaksi todennäköisesti keskittymään käyttäjäpalautteesta saataviin kehitysehdotuksiin, joiden jälkeen voidaan keskittyä Louhi Net Oy:n muiden järjestelmien integrointeihin. Integroinnit tulevat toivottavasti lähitulevaisuudessa mahdollistamaan muun muassa automaattisen ohjelmisto- ja laitteistotietojen sekä sopimustietojen keruun. Myös eräs huomiota kaipaava osa-alue tulee olemaan sovelluksen testauksen ja testien kattavuuden lisääminen. Jatkokehitettävää projektissa tulee varmasti riittämään vielä lähivuosina.

6.7 Käyttöönotto

Toteutukseen käytettyjen tekniikoiden ansiosta sovellus on suurilta osin alustariippumaton. Pythonista löytyy toteutuksia niin Windows ja Linux kuin Mac OS X -alustoille [3]. Näin ollen palvelinympäristön ainoana rajoitteena onkin johonkin yksittäiseen alustaan sidotut ohjelmistokirjastot tai ohjelmointiratkaisut. Sovelluksen toteutuksessa ei erityisesti painotettu alustariippumattomuutta, koska siihen ei nähty tarvetta. Sovelluksen kehitys tapahtui Linux-alustalla, joten se tulee olemaan luonnollinen valinta myös tuotantoympäristölle. Toteutuksessa ei käytetty ohjelmistokirjastoja tai ohjelmointiratkaisuja, jotka suoraan estäisivät sovelluksen ajamisen muissa kuin Linux-ympäristössä. Käytännössä pienillä muutoksilla sovellus olisi mahdollista siirtää esimerkiksi Windows-alustalle.

Aikatauluongelmista johtuen järjestelmän käyttöönottoa ei ehditty sisällyttää tähän insinöörityöhön. Sovelluksen palvelinympäristöksi on kuitenkin suunniteltu käytettävän Red Hat Enterprise Linuxin (RHEL) viimeisintä 6.2 versiota. Red Hat on yksi suurimmista yritystason Linux-jakeluja tarjoavista yrityksistä, minkä takia se on hyvä valinta sovelluksen ympäristöksi [41]. Sovelluksen käyttämää tietokantaa ei ole vielä päätetty, mutta valinta tullaan mitä luultavammin käymään MySQL:n ja PostgreSQL:n välillä.

Datan migrointi

Nykyisestä järjestelmästä kerättyjen tietojen avulla toteutettiin järjestelmän datalle myös migrointifunktiot. Nämä funktiot käyvät läpi nykyisen järjestelmän tiedostohierarkiaa luoden tiedostojen sisällöistä objekteja uuteen järjestelmään. Migrointifunktiot tunnistavat erilaiset tietotyypit tiedostonimien yhtenevistä etuliitteistä, joista on mahdollista eritellä nykyisen järjestelmän erilaisia sivutyyppejä. Migrointifunktiot käyvät tiedostoja läpi rivi riviltä hakien määrättyihin regex-kuvauksiin sopivia tietoja. Funktio kerää halutut tiedot talteen tiedostokohtaisesti ja luo näistä tiedoista uuteen järjestelmään sisältöä vastaavat objektit.

7 Yhteenveto

Insinöörityössä toteutettiin Louhi Net Oy:lle räätälöity sisällönhallintajärjestelmä yrityksen sisäisen tiedon hallintaa varten. Toteutetun sovelluksen toivotaan sen lopullisen käyttöönoton jälkeen tehostavan työskentelyä ja vähentävän tiedon ylläpitoon kuluvaa aikaa. Toteutettu sovellus tulee myös tulevaisuudessa mahdollistamaan lähes reaaliaikaisen informaation esittämisen erilaisten integraatioiden avulla. Integraatioiden avulla sovelluksen tarjoamaa hyötyä ja tietomäärää saadaan kasvatettua, tuottaen näin ollen lisäarvoa käyttäjilleen.

Sovelluksen toteutuksessa pyrittiin mahdollisimman modulaariseen ja selkeään rakenteeseen, jotta jatkokehitysvaiheessa sovelluksen rakenne olisi helposti hahmotettavissa myös toisille sovelluskehittäjille. Sovelluksen jatkokehitystä tullaan jatkamaan Louhi Net Oy:n palveluksessa käytössä olevien resurssien mukaisesti. Jatkokehitystoimien ohella tullaan myös perehdyttämään muita Louhi Net Oy:n ohjelmistokehitysyksikön jäseniä sovelluksen tarkempaan toteutukseen.

Insinöörityön aihe oli mielenkiintoinen ja haasteellinen. Koko insinöörityöprosessi oli hyvin opettavainen ja hyödyllinen oman osaamiseni kehittämisessä. Oli hienoa päästä toteuttamaan insinöörityö, joka tulee helpottamaan ja tehostamaan Louhi Net Oy:n henkilöstön työntekoa tulevaisuudessa.

Lähteet

1. Lutz, Mark. 2009. Learning Python, Fourth Edition. O'Reilly.
2. Krug, Steve. 2006. Don't Make Me Think – A Common Sense Approach To Web Usability (Second Edition). New Riders Publishing.
3. Python (programming language). Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Python_(programming_language)>. Luettu 15.02.2012.
4. Django (web framework). Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Django_(web_framework)>. Luettu 15-.02.2012.
5. Adrian Holovaty, Jacob Kaplan-Moss. 2008. The Definitive Guide to Django: Web Development Done Right. Apress.
6. Randall Degges. The Perfect Django Settings File. Verkkodokumentti. <http://rdegges.com/the-perfect-django-settings-file>. Luettu 18.02.2012.
7. Django-dynamic-formset. Verkkodokumentti. Google Code. <http://code.google.com/p/django-dynamic-formset/>. Luettu 24.02.2012.
8. Datatables. Verkkodokumentti. <http://datatables.net/>. Luettu 24.02.2012.
9. Progressive_enhancement. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Progressive_enhancement>. Luettu 28.02.2012.
10. Django-tables2. Verkkodokumentti. Github. <https://github.com/bradleyayers/django-tables2>. Luettu 28.02.2012.
11. Django-taggit. Verkkodokumentti. Github. <https://github.com/alex/django-taggit>. Luettu 28.02.2012.
12. Django-taggit-templatetags. Verkkodokumentti. Github. <https://github.com/feuervogel/django-taggit-templatetags>. Luettu 28.02.2012.

13. Django-taggit-autosuggest. Verkkodokumentti. Bitbucket. <https://bitbucket.org/fabian/django-taggit-autosuggest>. Luettu 28.02.2012.
14. Django-markitup. Verkkodokumentti. Bitbucket. <https://bitbucket.org/carljm/django-markitup>. Luettu 28.02.2012.
15. Django-reversion. Verkkodokumentti. Github. <https://github.com/etianen/django-reversion>. Luettu 28.02.2012.
16. Django-locking. Verkkodokumentti. Github. <https://github.com/RobCombs/django-locking>. Luettu 28.02.2012.
17. South. Verkkodokumentti. <http://south.aeracode.org/docs/about.html>. Luettu 28.02.2012.
18. Haystack. Verkkodokumentti. <http://haystacksearch.org/>. Luettu 28.02.2012.
19. Django-auth-ldap. Verkkodokumentti. Bitbucket. <https://bitbucket.org/psagers/django-auth-ldap/wiki/Home>. Luettu 28.02.2012.
20. History of Python. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/History_of_Python>. Luettu 28.02.2012.
21. ABC programming language. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/ABC_programming_language>. Luettu 28.02.2012.
22. Web application framework. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Web_application_framework>. Luettu 29.02.2012.
23. HTML. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/HTML>. Luettu 29.02.2012.
24. Jukka K. Korpela. 2008. CSS verkkosivujen muotoilussa. Docendo.
25. Javascript. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/JavaScript>. Luettu 29.02.2012.
26. JQuery. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/JQuery>. Luettu 29.02.2012.
27. Revision control. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Revision_control>. Luettu 29.02.2012.
28. Git. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Git_(software)>. Luettu 29.02.2012.

29. Django-admin. Verkkodokumentti. Django-projektin dokumentaatio. <https://docs.djangoproject.com/en/dev/ref/django-admin/>. Luettu 29.02.2012.
30. Cascading Style Sheets. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Cascading_Style_Sheets>. Luettu 01.03.2012.
31. W3C Standards. Verkkodokumentti. W3C. <http://www.w3.org/TR/#tr_CSS>. Luettu 04.03.2012.
32. SQLite. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/SQLite>. Luettu 13.03.2012.
33. SQLite. Verkkodokumentti. <http://www.sqlite.org>. Luettu 13.03.2012.
34. Virtualenv. Verkkodokumentti. PyPI - the Python Package Index <http://pypi.python.org/pypi/virtualenv>. Luettu 13.03.2012.
35. Eclipse. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Eclipse_(software)>. Luettu 13.03.2012.
36. Markdown. Verkkodokumentti. <http://daringfireball.net/projects/markdown/>. Luettu 14.03.2012.
37. Louhi Net Oy. Verkkodokumentti. <http://www.louhi.fi />. Luettu 25.03.2012.
38. XHTML. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/XHTML>. Luettu 09.04.2012.
39. Django 1.4 Release Notes. Verkkodokumentti. <https://docs.djangoproject.com/en/dev/releases/1.4/>. Luettu 10.04.2012.
40. Django 1.3 Release Notes. Verkkodokumentti. <https://docs.djangoproject.com/en/dev/releases/1.3/>. Luettu 10.04.2012.
41. Red Hat. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Red_Hat>. Luettu 12.04.2012.
42. Garbage collection. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29>. Luettu 18.04.2012.
43. List comprehension. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/List_comprehension>. Luettu 18.04.2012.