KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Henri Portimo

PYTHON JA DJANGO WEBKEHITYKSESSÄ

Opinnäytetyö
Toukokuu 2014

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2014
Tietojenkäsittelyn koulutusohjelma

Karjalankatu 3
80200 JOENSUU
Puh. 050 311 9160

Tekijä
Henri Portimo

Nimeke
Python ja Django webkehityksessä

Toimeksiantaja
-

Tiivistelmä

Opinnäytetyössä tarkastellaan Pythonia ja Djangoa web-sovelluskehityksessä Linux-käyttöjärjestelmällä. Työn tavoitteena on selvittää Pythonin, Djangon ja Linuxin käyttöä aloittelevan käyttäjän näkökulmasta sekä löytää etuja ja eroja aikaisempiin web-sovelluskehitystekniikoihin verrattuna. Työssä luodaan yksinkertainen äänestyssovellus, jossa sovelluksen hallinnoijat voivat luoda kyselyitä.

Opinnäytetyön teoriaosuudessa esitellään erilaisia web-sovelluskehyksiä sekä sisällönhallintajärjestelmiä. Työn toiminnallisessa osuudessa asennetaan käyttöjärjestelmä sekä luodaan Djangon virallisen dokumentaation ja ohjeiden avulla yksinkertainen äänestyssovellus. Sovellus luodaan Python- ja Django-tekniikoilla. Työssä esitellään Linuxin asentaminen ja web-sovelluskehityksen vaiheet sekä kuvataan, miten luodaan Django-projekti ja -sovellus.

Opinnäytetyön tuloksena syntyi yksinkertainen äänestyssovellus sekä selvisi, miten Python, Django ja Linux eroavat aikaisemmin käytetyistä tekniikoista. Djangon ja Pythonin suurimmiksi eduiksi osoittautuivat koodin ylläpidon helppous, mahdollisuus käyttää luotuja sovelluksia muissa projekteissa sekä Djangon tarjoama hallintapaneeli.

Kieli
suomi

Sivuja 27

Asiasanat
verkko-ohjelmointi, sovelluskehykset, Python, Django

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2014
Degree Programme in Business Information Technology
Karjalankatu 3
FI 80200 JOENSUU
FINLAND
Tel. 050 311 9160

Author
Henri Portimo

Title
Python and Django in Web Development

Commissioned by
-

Abstract

This thesis observes the usage of Python and Django in web application development using Linux as the operating system. The aim is to examine the usage of Python, Django and Linux from the point of view of a beginner and to find advantages and disadvantages when compared to earlier web application development techniques. In this study a simple poll application in which administrators can create polls and users can vote is developed.

In the theoretical part of the thesis different web application frameworks and content management systems are introduced. In the functional part the operating system is installed and a simple poll application is created using the official Django documentation and instructions. The application is created using Python and Django. Installation of Linux and the steps of web application development are described and also how to create a Django project and an application.

As a result of this thesis a simple poll application was created and it was found out how Python, Django and Linux are different from previously techniques. The greatest advantages of Django and Python were found to be the ease of code management, possibility of reusing created applications in other projects and the administration panel offered by Django.

Language	Pages
Finnish	27

Keywords
web development, application frameworks, Python, Django

Sisältö

1 Johdanto

Opinnäytetyön aiheena on tarkastella Pythonia ja Djangoa web-sovelluskehityksessä ensikertalaisen silmin Linux-käyttöjärjestelmän avulla. Web-sovelluskehitystä tarkastellaan Djangon tarjoamien tutoriaalien avulla. Työn tavoitteena oli toteuttaa yksinkertainen internetsivusto, jossa käyttäjät voivat äänestää kyselyissä ja hallinnoijat voivat asettaa ja poistaa kyselyitä.

Aiheen valitsin mielenkiinnostani web-sovelluskehitykseen. Aikaisemmat kokemukseni web-sovelluskehityksestä ovat olleet HTML/PHP-painotteisia, joten uusien tekniikoiden opettelu houkutteli. Uudet tekniikat tulivat esille Visithome-projektin kautta. Visithome-projekti tarjoaa käyttäjille palvelun, mitä kautta etsiä kotimajoitusta Suomesta eri kriteerien avulla.

Opinnäytetyöraportin teoriaosuudessa tarkastellaan työssä käytettyä web-sovelluskehystä, Djangoa, sekä muita kehyksiä ja sisällönhallintajärjestelmiä. Teoriaosuudessa selvitetään myös, miksi sovelluskehykseksi valikoitui juuri Django .

Toiminnallisessa osuudessa käydään läpi kehitysympäristön asentaminen sekä konfigurointi. Kehitysympäristön asentaminen sekä konfiguroiminen sisältää käyttöjärjestelmän sekä käyttämieni ohjelmien asentamisen. Toiminnallinen osuus sisältää myös tutoriaalien läpikäymisen ja sen, miten sivusto luodaan.

Opinnäytetyön tulokset-osio sisältää sivuston kuvauksen. Tuloksissa kerrotaan myös eroja aikaisempaan web-sovelluskehityskokemukseen ja käytettyyn käyttöjärjestelmään. Vertailukohtina tässä osiossa ovat Windows-käyttöjärjestelmä sekä HTML/PHP-internetsivustojen luonti.

Pohdintaosio sisältää mietteitä siitä, millaiseksi koin web-sovelluskehityksen vieraalla käyttöjärjestelmällä, ohjelmointikielellä sekä web-sovelluskehyksellä. Pohdintaosiossa kerron myös millaiseksi koin uusien tekniikoiden opettelun.

2 Web-sovelluskehitys

2.1 Web-sovelluskehykset

Web-sovelluskehykset ovat niin sanottuja pohjia tai kirjastoja, jotka on luotu helpottamaan sekä nopeuttamaan web-sovellusten kehitystä. Web-sovelluskehykset tarjoavat usein web-sovelluksille yleisiä toimintoja kuten istunnonhallintaa, tiedon pysyvyyttä ja sivustomalleja. Tarkoitukseen sopivan web-sovelluskehyksen valinnalla voikin huomattavasti lyhentää kehittämiseen menevää aikaa. (DocForge 2013.)

Web-sovelluskehyksiä on luotu monille eri ohjelmointikielille ja erilaisiin tarkoituksiin. Käytettyjä kieliä ovat muun muassa Java, PHP, C#, Python ja Ruby. Eri kehysten ominaisuudet vaihtelevat sen tarkoituksen ja kielen mukaan, mutta jokaisen perimmäisenä tavoitteena on nopeuttaa ja selkeyttää web-sovelluksien kehitystä. (Tech Magazine for Developers, Designers & SEO Specialists 2013.) Seuraavissa luvuissa esitellään lyhyesti suosittuja web-sovelluskehyksiä.

2.1.1 Vaadin

Vaadin on suomalainen Java-pohjainen web-sovelluskehys. Vaadin sai alkunsa vuonna 2000 web-käyttöliittymäkirjastona nimeltä IT Mill Toolkit, josta se kehittyi vuonna 2009 Vaatimeksi. Vaadin on ilmainen ja avoimen lähdekoodin web-sovelluskehys mikä käyttää Apachen 2.0 -lisenssiä. (Vaadin 2013a.)

Vaadin on yhteensopiva Windowsin, Linuxin sekä Mac OS X:n kanssa. Java-pohjaisuutensa vuoksi Vaadin vaatii Java Servlet API 2.4:n tai uudemman version. Tuetut serverit sisältävät muun muassa Apache Tomcatin version 5 sekä Glassfishin version 2 tai uudemmat versiot. Vaadin on yhteensopiva MySQL-, MSSQL-, Oracle-, PostgreSQL- ja HSQLDB-tietokantojen kanssa. (Vaadin 2013b.)

Vaatimen yleisin käyttökohde ja myös sen päätarkoitus on web-sovelluksien luominen. Vaatimen kehittäjät eivät suosittele käyttämään web-sovelluskehystään kokonaisten web-sivustojen luontiin, vaikka mikään ei sitä estäisikään. Tähän tarkoitukseen he suosittelevat käyttämään sisällönhallintajärjestelmää. Vaatimella voi kuitenkin luoda web-sovelluksen ja käyttää sitä sisällönhallintajärjestelmällä luodulla sivustolla. (Vaadin 2013c.)

2.1.2 CakePHP

CakePHP on Michal Tatarynowiczin luoma avoimen lähdekoodin web-sovelluskehys. CakePHP sai alkunsa vuonna 2005 Tatarynowiczin kirjoitettua nopean kehityksen kehyksen PHP-ohjelmointikielellä. CakePHP:n rakenne sai vaikutteita Ruby on Rails -kehyksestä. Kehyksen potentiaalin huomattuaan Tatarynowicz julkaisi kehyksensä MIT:n lisenssin alaisena nimellä Cake. (CakePHP 2013a.) CakePHP tarvitsee toimiakseen web-serverin, kuten Apachen, LightHTTPD:n tai Microsoft IIS:n. CakePHP:n vaatimuksiin kuuluu myös PHP:n versio 5.2.8 tai sitä uudempi versio. Tietokannaksi käy MySQL:n versio 4 tai uudempi versio, PostgreSQL, Microsoft SQL Server tai SQLite. (CakePHP 2013b.)

CakePHP tarjoaa monia syitä, miksi juuri sitä kannattaisi käyttää. MVC-arkkitehtuuri on yksi näistä syistä. Toinen tärkeä ominaisuus on niin sanottu "scaffolding"-ominaisuus, joka liittyy MVC-arkkitehtuuriin. (CakePHP 2013c.)

MVC-arkkitehtuurissa (Model-View-Controller, malli-näkymä-käsittelijä) sovelluksen toiminnot on jaettu mallille, näkymälle ja käsittelijälle. Näkymä on vastuussa tiedon näyttämisestä käyttäjälle. Näkymä voi olla sovelluskohteesta riippuen kuva, tekstiä tai mitä vain. Käsittelijä on ohjelman osa, joka käyttäjän ohjeiden mukaan muokkaa mallia ja näkymää. Malli on ohjelman osa, joka on vastuussa sovelluksen tiedosta, ja se antaa tiedon näkymään tai muuttaa tietoa käsittelijän ohjeiden mukaisesti. (Burbeck 1992.)

Scaffolding-tekniikalla web-sovelluskehittäjä voi nopeasti luoda yksinkertaisen sovelluksen, joka voi luoda, hakea, päivittää ja poistaa tietoa. Tämä tekniikka poistaa tarpeen tehdä kyseiset toiminnot itse, mikä säästää web-sovelluskehittäjän aikaa (CakePHP 2013d).

2.1.3 Django

Django sai alkunsa vuonna 2003, kun Adrian Holovaty ja Simon Willison alkoivat käyttää Pythonia sovellusten rakentamiseen. Motivaationa Djangon kehittämiselle oli tarve saada tehtyä sovelluksia päivien tai jopa vain tuntien aikataululla. Kaksi vuotta web-sovelluskehyksen luomisen aloittamisen jälkeen, vuonna 2005, se julkaistiin avoimena lähdekoodina. Nykyisin Djangolla on useita kymmeniä tuhansia käyttäjiä ja kehitykseen vaikuttavia henkilöitä. (The Django Book 2009a.)

Koska Django on Python-kieleen pohjautuva web-sovelluskehys, sen käyttämiseen tarvitsee Pythonin. Pythonin tuetut versiot ovat 2.6.5 sekä 2.7, mutta Django tarjoaa myös kokeellisen tuen versiosta 3.2.3 versioon 3.3 asti. Pythonin mukana tulee kevyeen testaamiseen soveltuva serveri, mutta oikeaan käyttöön tuleva Django-pohjainen sivusto tarvitsee Apache-serverin sekä mod_wsgi-moduulin. Django tukee virallisesti PostgreSQL-, MySQL-, Oracle- sekä SQLite-tietokantoja. (Django 2013.)

Myös Django soveltaa löyhästi MVC-arkkitehtuuria. Tämä tulee ilmi Djangolla luotujen sovellusten osista: models.py, views.py ja urls.py sekä HTML-mallista. Models.py sisältää kuvauksen tiedosta sekä mahdollistaa toiminnot tiedon luomiselle, hakemiselle, päivittämiselle ja poistamiselle. View.py välittää sivuston sisällön HTML-mallille. Urls.py-tiedoston toimintona on puolestaan määrittää mitä views.py:n toimintoa kutsutaan. Tämän rakenteen ja tiedostojen funktioiden avulla sovelluksen muokkaaminen ei vaadi useiden tiedostojen muokkaamista, mikä puolestaan säästää aikaa. (The Django Book 2009b.)

2.2 Sisällönhallintajärjestelmät

Sisällönhallintajärjestelmät ovat web-sovelluskehyksiä korkeammalla tasolla (sovellusten kirjoittamisessa korkeampi taso tarkoittaa sitä, että se on kauempana konekielestä). Periaatteessa tämä tarkoittaa sitä, että sisällönhallintajärjestelmällä ei voi luoda uutta web-sovelluskehystä, mutta web-sovelluskehystä käyttäen voi luoda uuden sisällönhallintajärjestelmän, eli web-sovelluksen.

Sisällönhallintajärjestelmiä on web-sovelluskehysten tapaan monenlaisia ja erilaisiin tarkoituksiin. Jotkin voivat käydä loistavasti oman blogisivuston luomiseen ja ylläpitämiseen. Sivujen luominen toimii eri sisällönhallintajärjestelmillä hyvinkin erilaisesti. Se voi olla niinkin yksinkertaista kuin valmiin pohjan. Toisella järjestelmällä voi olla mahdollista raahata ja pudottaa eri elementtejä ja sovelluksia sivustopohjan eri tiloihin. Nämä eivät luonnollisestikaan anna niin paljon mahdollisuutta sivun kustomointiin. Paljon monipuolisempia hallintajärjestelmiä ovatkin ne, jotka antavat mahdollisuuden vaikuttaa enemmän kooditasolla.

2.2.1 WordPress

WordPress on yksi maailman suosituimpia avoimen lähdekoodin sisällönhallintajärjestelmiä. Se ei kuitenkaan alkanut tällaisena järjestelmänä, vaan pelkkänä blogisivustoalustana. Vuosien aikana se kehittyi lisäosien ja teemojen kautta sisällönhallintajärjestelmäksi. WordPressin suosiota selittänevät mahdollisesti sen alhaiset ohjelmistovaatimukset, kattavan dokumentoinnin lisäksi. WordPress-sisällönhallintajärjestelmää ei tulisi sekoittaa WordPress.com-palveluun, joka tarjoaa käyttäjilleen valmiin blogisivustoalustan. (WordPress 2013a.)

Toimiakseen WordPress vaatii PHP:n version 5.2.4 tai uudemman sekä MySQL:n version 5.0 tai uudemman. Serveriksi suositellaan Apachea tai Nginxiä niiden monipuolisuuden vuoksi. Myös suPHP-moduulia suositellaan turvallisuuden lisäämiseksi. (WordPress 2013b.)

WordPress-sisällönhallintajärjestelmä soveltuu blogisivustohistoriansa ansiosta erittäin hyvin blogisivustojen ylläpitämiseen. Kymmenien tuhansien lisäosien avulla WordPressia voi käyttää kuitenkin erittäin monipuolisesti (WordPress 2013c.). Jo mainitun blogisivuston lisäksi WordPress soveltuu muun muassa kuvagallerioiden ylläpitoon, uutissivustoiden luomiseen sekä moniin muihin tarkoituksiin.

WordPressin erittäin kattava lisäosatarjonta voi tosin tuoda myös suuria ongelmia. Sivuston luoja on voinut käyttää useita eri lisäosia tehdäkseen sivustostaan mieleisen. WordPressin päivittyessä voi kuitenkin käydä niin, että käytetyt lisäosat eivät enää ole yhteensopivia uuden WordPress-version kanssa. (Terrar 2011.) Ongelman voi välttää jättämällä päivityksen väliin, mutta kriittisempien päivitysten, kuten tietoturvapäivitysten, kohdalla se ei olisi suotavaa.

2.2.2 Concrete

Concrete on avoimen lähdekoodin ilmainen sisällönhallintajärjestelmä, jonka ensimmäinen versio julkaistiin vuonna 2003. Vuodesta 2003 lähtien Concreten kehittämisessä on ollut kolme ohjenuoraa: sen pitäminen yksinkertaisena, joustavana ja vakaana. Concretesta ilmestyi joka vuosi uusi versio vuoteen 2008 asti, jolloin järjestelmän nimeksi vakiintui Concrete5. (Concrete5 2013a.)

Concrete5:n yksi erikoisimmista piirteistä on sen mahdollisuus muokata sivustoja suoraan selaimen kautta. Tämän ansiosta tehdyt muutokset voi nähdä paljon nopeammin verrattuna muokkausten tekemiseen erillisiin tiedostoihin. Toinen erikoinen piirre on järjestelmän "block"-osiot (laatikoita, joihin voi lisätä erilaisia toiminnallisuuksia). Käyttäjä voi lisätä sivuston ulkoasuun erilaisia "blockeja", jotka voivat sisältää esimerkiksi sisäänkirjautumisen, kuvagallerian tai lomakkeen. Sivuston muokkaamisen suoraan selaimesta voi jäljittää yksinkertaisena pitämisen ohjenuoraan. Tämän ansiosta ei tarvitse etsiä oikeaa muokattavaa tiedostoa tai kohtaa koodista. Oikea kohta löytyy klikkaamalla selaimen kautta blockia, jota haluaa muokata.

Concrete5:n laitteistovaatimukset ovat WordPressia suuremmat. PHP:sta vaaditaan versio 5.2.4 tai uudempi, mutta 5.3 on suositeltu. PHP:n lisäksi tarvitaan useita PHP-moduuleja. MySQL:stä vaaditaan versio 5 tai uudempi. Suositeltu serveri on Apache, vaikkakin IIS toimii. (Concrete5 2013b.)

Conrete5:tä kutsutaan usein mieluummin suunnittelijan sisällönhallintajärjestelmäksi kuin web-kehittäjän järjestelmäksi. Concrete5:n oppimiskäyrä on erittäin loiva muihin verrattuna, lähinnä järjestelmän valmiiden blockien ansiosta. Useimmissa tapauksissa joku on jo tehnyt valmiiksi tarvittavan ominaisuuden sisältävän moduulin, jonka voi ladata Concrete5:n markkinapaikalta (Sufyan bin Uzayr 2012.). Myös sivuston suora muokkaus selaimen kautta helpottaa järjestelmän käyttämistä.

Concrete5:tä ei kuitenkaan suositella käytettäväksi niin sanotuissa pilvipalveluissa. Pilvipalveluissa prosessit ovat hajautettuina useisiin virtuaaliympäristöihin. Oman serverin pitämisellä on se etu, että sen asetuksia pystyy säätämään täysin vapaasti. Pilvipalveluihin turvautuessa annetaan pois mahdollisuus hallita täysin omaa serveriä. Tämä johtaa useimmissa tapauksissa sivuston hitaaseen toimintaan. (Maruna 2010.)

2.2.3 Drupal

Drupal on myös yksi erittäin tunnettu ja suosittu avoimen lähdekoodin sisällönhallintajärjestelmä. Drupalin kehittäminen alkoi vuonna 2000 Antwerpenin yliopistossa internetiä keskenään jakaneiden opiskelijoiden keskuudessa. Ennen kehittymistään sisällönhallintajärjestelmäksi opiskelijoiden kehittämää systeemiä käytettiin ryhmänsisäiseen tiedonjakoon. Vuonna 2001 Dries Buytaert, Drupalin kehittäjä, julkaisi järjestelmänsä, jota hän oli käyttänyt opiskelukavereidensa kanssa yhteydenpitoon. (Drupal 2013a.)

Drupalin laitteistovaatimukset eivät poikkea suuresti muista sisällönhallintajärjestelmistä. PHP:n versioksi suositellaan Drupalin versiosta riippuen 5.2, 5.3 tai 5.3.10. Tietokannaksi voi Drupalin versiolle 6 valita MySQL 4.1:n tai Post-

greSQL 7.1:n. Versio 7 vaatii MySQL 5.0.15:n PostgreSQL 8.3:n tai SQLite 3.3.7:n. Kolmella edellä mainitusta käyvät myös uudemmat versiot. Drupalin asennus vaatii 15 megatavua levytilaa, 60 megatavua tarvitaan, jos sivusto käyttää useita moduuleja ja teemoja. Lisää tilaa tarvitaan silloin, jos sivusto sallii käyttäjien ladata sivustolle omaa mediasisältöään, kuten kuvia ja videoita. (Drupal 2013b.)

Drupal on erittäin monipuolinen sisällönhallintajärjestelmä. Drupalin sivuilla kerrotaan, että Drupal käy joustavuutensa ansiosta moniin erilaisiin sivustoihin, jotka voivat kehittyä useaan suuntaan. Esimerkiksi blogisivusto voi kehittyä verkkokaupaksi tai foorumiksi. Käyttämällä Drupalia sivustosta saa myös helposti muiden sivustojen kanssa vuorovaikutuksessa olevan. Tärkeämpää kuitenkin on Drupal-sivuston listauksessa se, mihin tarkoituksiin Drupalia ei ole optimoitu. Näitä kohtia on listattu kolme: pelkän blogisivuston pitäminen, wikisivuston tekeminen ja keskustelufoorumin luominen. Sivustolla suositellaankin vaihtoehtoisia järjestelmiä näitä varten. Blogeille suositellaan WordPressia tai Bloggeria, wiki-sivustolle MediaWikia ja foorumille SimpleMachines- tai phpBB-järjestelmää. Drupalilla on kuitenkin tarjota myös foorumimoduuli omaan järjestelmäänsä. (Drupal 2012.)

2.3 Web-sovelluskehyksen valinta

Aikaisemmin olin käyttänyt Concrete5:tä sekä WordPressia, mutta halusin tutustua työssä johonkin uuteen käyttöympäristöön. Opinnäytetyön teoriaosuutta kirjoittaessani tutustuin pintapuolisesti useaan mahdolliseen web-sovelluskehykseen sekä sisällönhallintajärjestelmään. Tässä vaiheessa tutustuin myös Django-web-sovelluskehykseen. Django kuitenkin vaati myös Python-ohjelmointikielen osaamisen, joten kävin läpi myös Googlen tarjoamia Python-tutoriaaleja.

Pythonin ja Djangon valinta kuitenkin tarkoitti kuitenkin kokonaan uuden ohjelmointikielen sekä web-sovelluskehyksen opettelua. Tästä huolimatta tutustuttuani Pythoniin sekä Djangoon päätin, että valitsisin juuri tämän vaihtoehdon.

Painavin syy näiden valintaa oli se, että olin jo käyttänyt sisällönhallintajärjestelmää aikaisemmassa projektissa. Googlen tutoriaalien ansiosta myös huomasin, että Python ei eroa PHP:n syntaksista kovinkaan suuresti. Päätökseeni vaikutti myös se, että Python-kielen osaajista on pulaa PHP-kielen jäädessä pikkuhiljaa syrjään (Storås 2012). Kielen oppimisesta voisi siis olla hyötyä tulevaisuudessa opinnäytetyön ja koulusta valmistumisen jälkeen.

3 Django-projektin luominen

3.1 Kehitysympäristön asentaminen

3.1.1 Käyttöjärjestelmä

Portaalin kehittämistä varten tarvitsin Linux-käyttöjärjestelmän. Kehittämiseen olisi soveltunut myös Windows-käyttöjärjestelmä, mutta useammankaan yrityksen jälkeen en ollut saanut kaikkea toimimaan niin kuin pitäisi. Linux-käyttöjärjestelmän version valinta oli helppoa, sillä olin aikaisemmin jo hiukan tutustunut ilmaiseen Linux Ubuntu -käyttöjärjestelmään. Ubuntun asentamisessa minulla oli kaksi vaihtoehtoa: joko asentaminen suoraan työkoneeni kovalevylle tai Oracle VM Virtualbox -virtuaalikoneelle. Päädyin lopulta asentamaan sen virtuaalikoneelle, sillä olin tehnyt niin aiemmin Ubuntua käyttäessäni.

Käyttöjärjestelmän asentaminen ei tuottanut minkäänlaisia ongelmia. Oracle VM Virtualboxin avulla Ubuntun asentaminen oli erittäin ohjattua ja selkeää. Ennen asennusta Virtualboxista pystyi muuttamaan asennettavalle käyttöjärjestelmälle annettavia resursseja, muun muassa muistia, kovalevytilaa ja prosessoritehoa. Ubuntun asentaminen ja sen päivittäminen tapahtui niin ikään erittäin ohjatusti. Aikaa Ubuntun saattamiseksi käyttökuntoon meni noin 10–15 minuuttia.

3.1.2 Versionhallinta

Missä tahansa projektissa versionhallinta on erittäin tärkeää. Versionhallinnan avulla näkee helposti tehdyt muutokset ja sen, kuka ne on tehnyt. Versionhallinnan avulla varmistetaan myös se, että kaikilla projektiin osallistujilla on sama versio tiedostoista. Versionhallinnan avulla yhdestä projektista voi tehdä myös eri haaroja, jolloin projektiin kohdistuvat muokkaukset eivät vaikuta toisiinsa. Myös mahdollisuus palata aikaisempiin projektin versioihin on yksi tärkeä versionhallinnan ominaisuus.

Projektissa käytin Git-versionhallintaohjelmaa. Git asennetaan komennolla "`sudo apt-get install git`". Komennolla "`which git`" voi tarkistaa ohjelman asennuskansion ja varmistaa, että se asentui (Atlassian Documentation 2013). Asentamisen jälkeen ohjelmaan syötetään omat tiedot. Oma nimi syötetään komennolla "`git config --global user.name "omanimi"`". Sähköpostiosoite annetaan komennolla "`git config --global user.email "osoite"`". Tietojen antamisen jälkeen projektitiedostot ladataan Bitbucket-sivustolta omalle tietokoneelle. Tiedostojen kopioiminen tapahtui käyttämällä "`git clone`" -komentoa sekä projektikansion uniikkia osoitetta. Kuka tahansa ei kuitenkaan voi kopioida projektitiedostoja itselleen, sillä sen luoneen henkilön täytyy antaa oikeudet Bitbucket-sivustolle rekisteröityneelle henkilölle sen kopioimiseen. Aikaisemmassa vaiheessa annettu sähköpostiosoite täytyy vastata sitä osoitetta, jolle oikeudet on annettu.

3.1.3 Pythonin ja Djangon asennus

Ennen Djangon käyttämistä täytyy asentaa Python. Pythonin voi ladata sen kotisivulta www.python.org:sta, mutta useimmissa Linux-asennuksissa se on valmiina mukana. Pythonin version käyttämällään Linuxilla voi tarkistaa kirjoittamalla komennon "`python`" komentokehotteeseen. Jos Python on asennettu kyseiselle Linuxille, komentokehote näyttää nykyisen Pythonin versionumeron.

Djangon tämänhetkisen virallisen version voi asentaa Linuxille komennolla "`sudo pip install Django`". Jos pip-ohjelmaa ei ole asennettuna, komentokehote huomauttaa siitä ja ohjaa käyttäjää asentamaan sen komennolla "`sudo apt-get install python-pip`". Djangon asentamisen jälkeen asentumisen onnistuminen ja Djangon versionumero voidaan todeta kirjoittamalla ensin komento "`python`", jonka jälkeen kirjoitetaan komennot "`import Django`" ja "`print(django.get_version())`". Kun Djangon ja Pythonin asennus on valmis, voidaan siirtyä itse Django-sivuston luontiin. Sivuston luomisessa käytin Djangon versiota 1.6.5.

3.2 Django-sivuston luonti

Tässä luvussa selostetaan, miten yksikertainen Django-sivusto luodaan. Sivusto koostuu yleisestä näkymästä, joka sisältää listauksen luoduista kyselyistä, äänestyslomakkeen, jonka avulla äänestetään kyselyissä sekä äänestämisen jälkeen näytettävistä äänien määrästä. Sivustolla on myös hallinnoijakäyttäjille admin-näkymä, hallintapaneeli. Hallintapaneelin avulla hallinnoijakäyttäjät voivat luoda, poistaa, muokata sekä etsiä kyselyitä.

3.2.1 Projektin ja sovelluksen luominen

Django-projekti aloitetaan komennolla "`django-admin.py startproject sivusto`", joka luo käyttäjän haluamaan tiedostopolkuun sivuston tarvitsemat tiedostot. Komento ei luo kaikkia projektissa tarvittavia tiedostoja, mutta tiedostorakenne projektin aloittamisen jälkeen näyttää tältä:

```
sivusto/
        manage.py
        sivusto/
                __init__.py
                settings.py
                urls.py
                wsgi.py
```

Projektin luomisen jälkeen on mahdollista valita, mitä tietokantaa haluaa projektillaan käyttää. Käytetty tietokanta vaihdetaan settings.py-tiedostosta. Tiedosto sisältää myös muita asetuksia, kuten aikavyöhykkeen. Settings.py sisältää myös tiedon kaikista projektin käyttämistä sovelluksista. Projektin alussa siellä on kuitenkin vain Djangon itsensä sisältämiä sovelluksia. Käyttäjän omat sovellukset lisätään kyseiseen listaan. Sovelluksien käyttöönottoa varten täytyy ajaa komento "`python manage.py syncdb`", joka luo tarvittavat tietokantataulut. On hyvä huomata, että "`django-admin.py`"-skriptiä ei tarvitse ajaa Python-tulkin kanssa toisin kuin "`manage.py`"-skriptiä. Tämä johtuu siitä, että Djangon asennusvaiheessa "`django-admin.py`" lisätään automaattisesti käyttöjärjestelmän PATH-ympäristömuuttujaan.

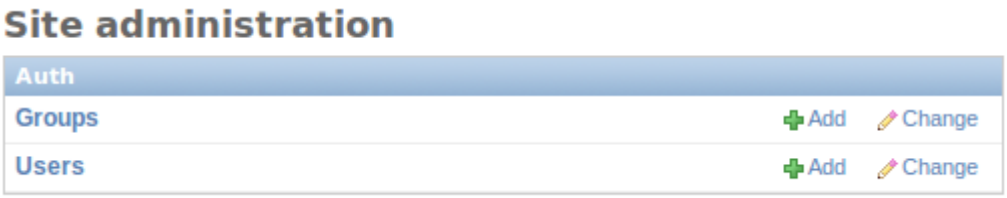
Projektin alkukonfiguraatioiden jälkeen voidaan alkaa tehdä omaa sovellusta Django-sivustolle. Sovelluksen tekeminen aloitetaan komennolla "`python manage.py startapp kyselyt`". Komento luo sovellukselle tiedostorakenteen, jonka avulla sovellus luodaan. Sovelluksen tiedostorakenne komennon jälkeen on seuraavanlainen:

```
kyselyt/
        __init__.py
        admin.py
        models.py
        tests.py
        views.py
```

Sovelluksen käyttämät mallit eli tietokantarakenteet määritellään models.py-tiedostoon. Esimerkiksi kyselyt-sovellus käyttää malleja Kysely ja Valinta. Kysely-mallissa on tietokantakentät kysymykselle ja julkaisupäivämäärälle ja Valinta-mallissa on vastausvaihtoehdot ja äänimäärät. Koska kyselyt-sovellus on uusi Djangolle, se täytyy lisätä settings.py-tiedostoon jotta Django osaa sisältää sen osaksi projektia. Sovellus lisätään lopullisesti osaksi projektia komennolla "`python manage.py syncdb`". Komento luo models.py-tiedostossa kuvattujen mallien mukaiset tietokantataulut.

3.2.2 Admin-näkymä

Djangon admin-näkymä eli hallintapaneeli on heti käyttövalmiina projektin aloittamisen jälkeen. Admin-näkymään pääsee käynnistämällä kehitysserverin komennolla "`python manage.py runserver`" ja menemällä paikalliseen verkkoalueeseen, local domainiin. Admin-näkymä ei vielä tässä vaiheessa sisällä vasta luodun sovelluksen tietoja vaan pelkästään admin-näkymän itse luomat Ryhmät (Groups) ja Käyttäjät (Users) (kuva 1).



Kuva 1 Juuri luotu admin-näkymä.

Sovelluksen tiedot saa näkyviin hallintapaneeliin muokkaamalla admin.py-tiedostoa ja lisäämällä sinne seuraavat koodirivit:

```
from kyselyt.models import Kysely
admin.site.register(Kysely)
```

Tämän jälkeen admin-näkymä osaa näyttää luodun sovelluksen kyselyt. Admin-näkymä ei kuitenkaan vielä näytä kyselyiden vastausvaihtoehtoja. Koska vastausvaihtoehdot ovat sopivia vain yhdelle kyselylle, on viisasta sitoa vastausvaihtoehtojen muokkaaminen kyselyiden muokkaamiseen. Myös muiden hallinta- sekä etsimistyökalujen lisääminen on tässä vaiheessa kannattavaa. Muokkausten jälkeen admin.py-tiedosto näyttää tältä:

```
from django.contrib import admin
from kyselyt.models import Kysely, Valinta

class ValintaRivit(admin.TabularInline):
          model = Valinta
          extra = 3

class KyselyAdmin(admin.ModelAdmin):
          fieldsets = [
                    (None,     {'fields': ['kysymys']}),
```

```
                    ('Paivamaara', {'fields': ['pvmr'], 'classes':
['collapse']}),
            ]
            list_display = ('kysymys', 'pvmr', 'julkaistiin')
            inlines = [ValintaRivit]
            list_filter = ['pvmr']
            search_fields = ['kysymys']

admin.site.register(Kysely, KyselyAdmin)
```

Luvussa 4.1 käyn tarkemmin läpi, mitä kaikkia muutoksia admin-näkymään yllä olevassa vaiheessa on tehty.

3.2.3 Yleinen näkymä

Yleisellä näkymällä tarkoitetaan sitä, mitä tietoa käyttäjälle näytetään ja miten se näytetään. Yleinen näkymä tehdään sovelluksen views.py-tiedostoon:

```
def index(request):
            viimlista = Kysely.objects.all().order_by('-pvmr')[:5]
            sisalto = {'viimlista': viimlista}
            return render(request, 'kyselyt/index.html', sisalto)
```

Yllä olevassa esimerkissä sovelluksen etusivulle tulostetaan viimeisimmät 5 kyselyä ja ne järjestetään päivämäärän mukaan. Muuttuja "sisalto" saa itselleen viimeisimmän kysymyslistan, joka tulostetaan index.html-ulkoasun ohjeiden mukaisesti. Views-tiedosto täytyy sitoa osoitteeseen kyselyt-kansion urls.py-tiedostoon:

```
urlpatterns = patterns('',
            url(r'^$', views.index, name='index'),)
```

Tämän lisäksi uusi URL täytyy sisällyttää koko projektin urls.py-tiedostoon, minkä jälkeen sivu on katsottavissa paikallisessa verkkoalueessa:

```
urlpatterns = patterns('',
            url(r'^kyselyt/', include('kyselyt.urls')),
            url(r'^admin/', include(admin.site.urls)),
)
```

Aikaisemmin mainitsemani index.html-ulkoasu käyttää Djangon omaa merkintäkieltä. Merkintäkielen avulla voidaan käyttää toistorakenteita, mikä helpottaa koodin kirjoittamista ja vähentää työtä. Ulkoasutiedosto näyttää tältä:

```
<h3>Kyselyt</h3>
{% if viimlista %}
    <ul>
    {% for kysely in viimlista %}
        <li><a href="{% url 'kyselyt:tiedot' kysely.id %}">{{ kyse-
ly.kysymys }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>Ei kyselyja.</p>
{% endif %}
```

Tiedoston ensimmäinen rivi tarkistaa, että on olemassa viimlista-niminen muuttuja, joka sisältää 5 viimeisintä luotua kyselyä. Jos muuttuja sisältää tietoa, se tulostaa sitä niin pitkään listamuodossa kuin siellä on tietueita. Jokainen listattu kysymys on linkitetty myös kysymyksen äänestyslomakkeeseen.

Äänestys-sivusto tarvitsee myös toiminnallisuuden äänestää. Äänestämisellä on oma funktionsa views.py-tiedostossa. Äänestysfunktio tarkistaa, että valinta on tehty. Jos valinta on tehty, äänestysfunktio lisää yhden äänen vaihtoehdolle ja siirtää käyttäjän tulossivulle. Jos ääntä ei anneta, funktio ohjaa takaisin kysymykseen. Kysymyslistan tapaan äänestysfunktio ja äänestyksen tulokset tulostetaan käyttäjälle html-tiedoston kuvauksen mukaisesti (kuva 2).

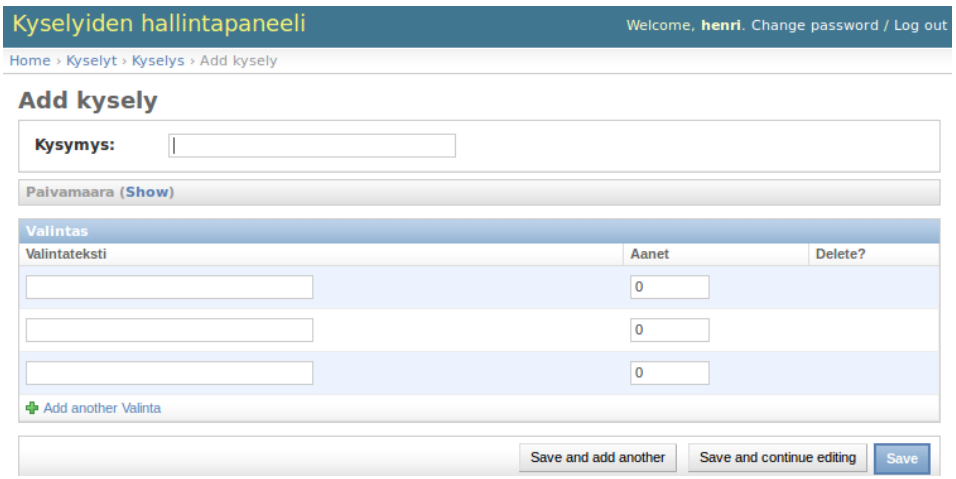
Kuva 2 Yleinen näkymä.

4 Tulokset

4.1 Sivusto (Django)

Luodulla Django-sivustolla normaali käyttäjä voi äänestää sivuston hallinnoijan luomissa äänestyksissä. Normaalit käyttäjät voivat myös tarkastella äänestettyään tuloksia. Normaalien käyttäjien oikeudet eivät tässä vaiheessa sivuston kehitystä anna mahdollisuutta lisätä uusia kyselyitä tai poistaa vanhoja.

Sivuston hallinnoijat voivat luoda kyselyiden hallintapaneelin avulla kyselyitä, joihin normaalit käyttäjät vastaavat. Kyselyitä voi luoda vain sivuston hallintapaneelin kautta (kuva 3).

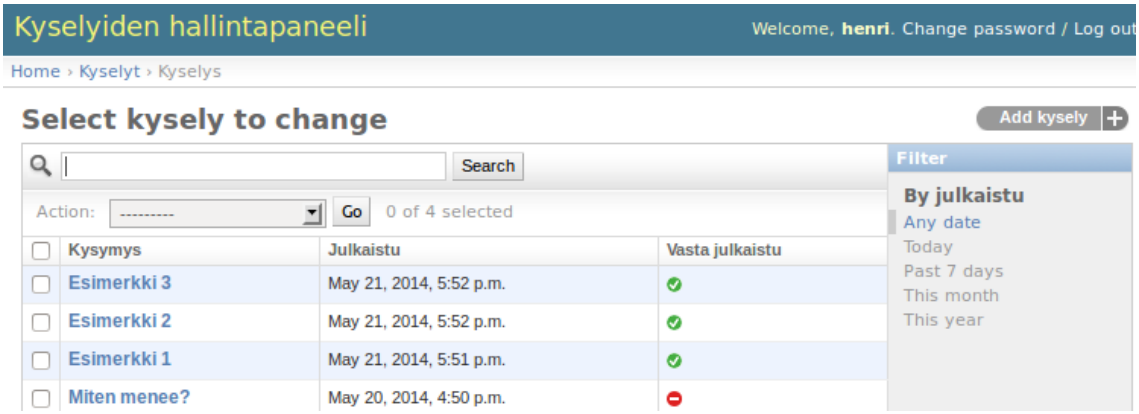


Kuva 3 Kyselyiden hallintapaneelin lisäysvalikko

Kuvassa 3 on lisäysvalikko. Jokaiselle kyselylle täytyy antaa nimi sekä asettaa päivämäärä "Paivamaara (Show)"-valikosta. Kyselyihin voi laittaa useita eri vastausvaihtoehtoja sekä poistaa jo asetettuja vastausvaihtoehtoja. Hallinnoija voi myös muokata vastausvaihtoehtojen annettuja ääniä.

Sivuston hallinnoijalla on myös oikeus selata, poistaa ja muokata olemassa olevia kyselyitä. Hallinnoija voi hakea kyselyitä hakusanojen perusteella tai järjestää ne joko laskevaan tai nousevaan järjestykseen kysymyksen julkaisupäivä-

määrän tai julkaisun läheisyyden mukaan. Kyselyitä on myös mahdollista suodattaa pois käyttämällä aikaan perustuvia suodattimia (kuva 4).



Kuva 4 Kyselyiden hallintapaneelin selausvalikko

4.2 Linuxin ja Windowsin erot

Windowsin ja Linuxin eroavaisuuden huomasi jo heti alussa Apachen, MySQL:n ja PHP:n asennustavasta. Windowsilla asennus hoitui erittäin helposti lataamalla XAMPP-ohjelma ja ajamalla asennusohjelma läpi. Tämän jälkeen palvelut pystyi käynnistämään suoraan asennetun ohjelman kautta. Linuxilla asia oli hieman monivaiheisempi. Asentaminen eroaa jo heti siten, että prosessi tehdään komentokehotteen kautta. Ennen kyseisten ohjelmien asennusta täytyy asentaa aptitude-ohjelma komentokehotteen komennolla "`sudo apt-get install aptitude`". Vasta tämän asennuksen jälkeen voidaan asentaa Apache, MySQL ja PHP. Näiden asennuskomento komentokehotteessa on "`sudo aptitude install apache2 php5 apache2.2-common libapache2-mod-auth-mysql php5-mysql mysql-server`". Tällä komennolla Linux lataa tarvittavat paketit ohjelmien asentamiseen.

Toinen suuri ero Linuxin ja Windowsin välillä on luku- ja kirjoitusoikeudet. Windowsilla työskennellessä harvoin tulee vastaan tilannetta, että oikeudet eivät riittäisi tiedostojen kirjoittamiseen. Niissä harvoissa tapauksissa jolloin niin käy, kirjoitusoikeuden saa yleensä avaamalla tiedostot järjestelmänvalvojana. Linu-

xilla asia ei taaskaan ole aivan niin yksinkertainen. Osa Linuxin kansioista on suojattu kirjoittamiselta, eli käyttäjä ei pysty tallentamaan mitään kyseisiin kansioihin. Oikeuksia ei voi myös lisätä kuten Windowsissa. Linuxissa oikeuksien lisääminen tapahtuu, ohjelmien asentamisen tapaan, komentokehotteen kautta. Kirjoitusoikeudet sain haluamalleni kansiolle komennolla "`sudo chmod 7777 /var/www`", jossa "/var/www" on kansion polku.

4.2 Pythonin ja PHP:n erot

Opinnäytetyön aikana huomasin eroja Pythonin ja PHP:n käytössä, mutta niillä oli myös keskenään samankaltaisia piirteitä.. Taulukossa 1 olevat koodiesimerkit tuovat esille näitä eroavaisuuksia sekä samankaltaisuuksia.

Taulukko 1. Yksinkertainen laskuri tehtynä PHP/HTML-koodilla ja Python/Django-koodilla.

PHP/HTML	Python/Django

1. <?php
2. $a=$_POST["luku1"];
3. $b=$_POST["luku2"];
4. $c=$_POST["valinta"];
 	
5. if($valinta==1) {
6. $tulos = $a+$b;}
7.
8. elseif($valinta==2){
9. $tulos = $a+$b;}
10.
11. else {
12. tulos = "Operaatiota ei valittu.";}
13.
14. echo "Tulos: $tulos."}
15. ?>
	1. def laskuri(request):
	2. a=request.POST['luku1']
	3. b=request.POST['luku2']
	4. valinta=request.POST['valinta']
	
	5. if valinta == 1:
	6. c = a+b
	7.
	8. elif valinta == 2:
	9. c = a-b
	10.
	11. else
	12. c = "Operaatiota ei valittu."
	
	13. tulos = "<html><body>Tulos %s:</body></html> % c
	14. return HttpResponse(tulos)

Taulukossa 1 on kuvattu yksinkertainen summa- ja vähennyslaskuri. Molemmissa esimerkeissä oletetaan, että käyttäjä on syöttänyt kaksi lukua ja valitsee, haluaako hän yhteen- vai vähennyslaskun. Sekä PHP:ssa että Pythonissa käyttäjän antamat luvut talletetaan muuttujiin taulukon riveillä 2–4. Python eroaa Djangon tavasta tallettaa muuttuja vain vähän. Ensimmäisenä erona PHP:ssa näkyy se, että muuttujat täytyy määrittää käyttämällä $-merkkiä, kun taas Pyt-

honissa ei ole tähän tarvetta. PHP:ssa rivit myös täytyy lopettaa puolipisteellä, mutta Pythonissa rivi lopetetaan rivinvaihdolla.

Seuraavaksi taulukon 1 esimerkeissä riviltä 5 lähtien käydään läpi ehtolauseke, jonka mukaan suoritetaan käyttäjän haluama operaatio. PHP:ssä ehtolausekkeen suorittaminen vaatii huomattavasti enemmän työtä koodaajalta kuin Pythonissa. PHP vaatii lausekkeen ehdon olevan sulkujen sisässä ja ehtolausekkeen läpikäynti tapahtuu kaarisulkujen sisällä. Pythonissa sulkuja ei tarvitse. Ehtolausekkeen läpikäyminen merkitään kaksoispisteen avulla (rivi 5) ja sisentämällä tätä seuraava rivi (rivi 6). Pythonissa ehtolausekkeen loppumista merkitään palaamalla aiemmalle sisennysriville, PHP:ssa se merkitään käyttämällä kaarisulkua.

Myös tuloksen palauttaminen eroaa PHP/HTML:n ja Python/Djangon välillä (taulukon 1 rivit 14). PHP:ssa tuloksen voi palauttaa echo-komennolla. Pythonissa tulos täytyy palauttaa HttpResponse-funktiolla. Molemmissa tapauksissa lopputuloksena on erittäin yksinkertainen sivu, joka sisältää vain muuttujan arvon.

5 Pohdinta

Linux käyttöjärjestelmänä opinnäytetyöprojektissa oli lähinnä olosuhteiden pakottama. Vaikka projektia olisi mahdollista ollut tehdä Windows-käyttöjärjestelmällä, olisi sen käyttöönotto ollut tarvittavine ohjelmineen huomattavasti työläämpää ja enemmän aikaa vievää. Linux myös hieman hidasti työskentelyä; Windowsilla opitut asiat eivät juurikaan auttaneet Linuxin käytössä. Suurimpana erona tuttuun Windowsiin oli eri ohjelmien ja kirjastojen asentaminen sekä käyttäjän luku- ja kirjoitusoikeudet. Sen lisäksi, että en esimerkiksi tiennyt nimeltä, mitkä olisivat tarkoitukseeni sopivia ohjelmia (Apache, MySQL, PHP), oli myös asennusprosessi erilainen Windowsiin verrattuna.

Opinnäytetyön alussa Linuxin käyttäminen oli huomattavasti hitaampaa Windowsiin verrattuna, mutta Pythonin ja Djangon käyttäminen projektissa teki siitä oikean valinnan. Vaikka Linux-käyttöjärjestelmän käyttäminen oli edelleenkin hitaampaa kuin Windowsin, alun vaikeuksista yli päästessä alkoi myös työskentely sujua paljon paremmin. Linux-käyttöjärjestelmän käytön aloittaminen olikin haastavin vaihe siihen liittyen; mikään ei ollut tuttua eikä Windowsin käytöstä saaduista tiedoista ollut juurikaan hyötyä. Apua kuitenkin löytyi runsaasti useasta eri lähteestä, kuten virallisesta Ubuntu-dokumentaatiosta, Ubuntun foorumeilta sekä superuser.com-sivustolta. Esimerkiksi kirjoitusoikeuksien antaminen käyttäjälle ilman apua olisi osoittautunut erittäin hankalaksi, mutta superuser.com-sivuston käyttäjällä oli sama ongelma, jonka ratkaisu sopi myös minulle. Kaiken kaikkiaan Linuxista jäänyt mielikuva on enemmän positiivinen kuin negatiivinen, mutta silti tulen jatkossa suosimaan Windowsia niin web-sovelluskehityksessä kuin muissakin toimissa.

Uuden ohjelmointikielen Pythonin opettelu ja käyttäminen opinnäytetyössä vaikutti alussa erittäin suurelta työltä. Googlen tutoriaalien avulla kuitenkin huomasin, että Pythonin opettelu onnistui huomattavasti helpommin kuin odotin. Erittäin paljon apua oli myös aikaisemmasta PHP-kokemuksesta, joka teki kielen käytöstä helpompaa. Alussa huomattavin, ja samalla eniten virheitä aiheuttanut, ero oli se, miten ohjelmakoodi sisennettiin. PHP:ssä sisennys tehdään {}-merkeillä, kun taas Pythonissa sisennetään käyttämällä välilyöntejä. Väärissä paikoissa olevat välilyönnit voivatkin aiheuttaa virheitä koodissa, jotka estävät sen toimimisen ja ovat joskus erittäin vaikeita löytää.

Python tarjoaa myös erittäin kattavan dokumentoinnin Pythonin syntaksiin, elementteihin, funktioihin sekä muuhun kielen käyttöön. Oppaita on täysin kokemattomille sekä kokeneille ohjelmoijille. Tämän vuoksi Pythonia voisi suositella jopa ensimmäiseksi ohjelmointikieleksi kokemattomille, mutta ohjelmoinnista kiinnostuneille.

Opinnäytetyössä käytettiin Django-web-sovelluskehystä. Se oli ensimmäinen sovelluskehys, johon olen tutustunut tarkemmin. Pythonin tapaan myös Django tarjoaa käyttäjilleen laajan dokumentoinnin sekä tutoriaalin yksinkertaisen Djan-

go-sovelluksen kirjoittamiseen. Tutoriaalin avulla sain pienen käsityksen Djangon toiminnasta, ja sitä seuraavat tutoriaalit tarjosivat syventävää tietoa ja ohjeita. Perustutoriaalien jälkeiset ohjeet kertovat muun muassa sähköpostin lähettämisestä Djangon avulla, tiedostojen hallinnasta sekä testaamisesta. Kaiken kaikkiaan Djangon tutoriaalien avulla voi päästä hyvinkin pitkälle.

Oman kokemukseni perusteella voin hyvinkin nähdä, miten Djangolla web-sovelluksien kehitys on nopeampaa kuin HTML/PHP-kehitys. Alussa kehitys on tietysti hitaampaa, mutta mitä tutummaksi Django tulee, sitä enemmän siinä näkee etuja. Yksi suurimmista eduista omasta mielestäni on mahdollisuus paketoida jokin yksittäinen osa web-sovelluksesta, esimerkiksi sivustolle rekisteröityminen. Sovelluksen osan pakkaamisen jälkeen sitä voi käyttää missä tahansa muissa Django-projekteissa, jotka voivat vaatia rekisteröitymisen toiminnallisuudessaan. Internetsivujen kehittäminen osissa myös helpottaa sen ylläpitoa, sillä muokkausten tekeminen on helppoa, kun tietää mikä osaa tarvitsee työtä. Myös Djangon tarjoama admin-näkymä helpottaa suuresti sivuston sekä tietokannan ylläpitoa. Admin-näkymän kautta voi helposti muokata sivuston käyttämiä tietokantatietueita selkeän käyttöliittymän kautta. Muokattaviin tietueisiin kuuluvat muun muassa käyttäjät sekä ryhmät, joihin käyttäjät voivat kuulua. Ryhmien avulla voi esimerkiksi erotella normaalit käyttäjät moderaattoreista ja moderaattorit sivuston ylläpitäjistä. Myös sivustolle annettua tietoa voi muokata helposti admin-näkymän kautta, esimerkiksi virheellisesti syötetyn tiedon voi muokata tätä kautta ilman, että tarvitsisi erillistä ohjelmaa päästäkseen tietokantaan ja muokkaamaan sitä tätä kautta.

Oman kokemukseni perusteella Django on erittäin suositeltava web-sovelluskehys. Laajat tutoriaalit ja dokumentoinnit docs.djangoproject.com-sivustolla auttavat huomattavasti vasta-alkajaa, mikä tekee Djangon käytön aloittamisesta suhteellisen helppoa. Djangon tehokkaaseen käyttöön kuitenkin tarvitsee oman aikansa, mikä on usein normaalia uusien tekniikoiden omaksumisessa.

Lähteet

Atlassian Documentation. 2013. Bitbucket Documentation. https://confluence.atlassian.com/pages/viewpage.action?pageId=269982882 5.8.2013.

Burbeck, S. 1992. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html 23.5.2013.

CakePHP. 2013a. History of CakePHP. 23.5.2013. http://book.cakephp.org/1.1/en/introduction-to-cakephp.html

CakePHP. 2013b. Requirements. 23.5.2013. http://book.cakephp.org/2.0/en/installation.html

CakePHP. 2013c. What is CakePHP? Why Use it? http://book.cakephp.org/2.0/en/cakephp-overview/what-is-cakephp-why-use-it.html 23.5.2013.

CakePHP. 2013d. Scaffolding. http://book.cakephp.org/2.0/en/controllers/scaffolding.html 23.5.2013.

Concrete5. 2013a. History. http://www.concrete5.org/about/our_philosophy/history/ 15.5.2013.

Concrete5. 2013b. System Requirements. http://www.concrete5.org/documentation/background/system_requirements/ 15.5.2013.

Django. 2013. How to install Django. https://docs.djangoproject.com/en/1.5/topics/install/ 23.5.2013.

DocForge. 2013. Web application framework. http://docforge.com/wiki/Web_application_framework 8.4.2013.

Drupal. 2012. Is Drupal the right tool for the job? http://drupal.org/node/346217 16.5.2013

Drupal. 2013a. History. http://drupal.org/about/history 16.5.2013.

Drupal. 2013b. System Requirements. http://drupal.org/requirements 16.5.2013.

Maruna, F. 2010. "The cloud" is bad for web hosting. Concrete5. http://www.concrete5.org/about/blog/client-work/the-cloud-is-bad-for-web-hosting/ 16.5.2013.

Storås, N. 2012. Töitä? Python-osaajista on pula. Tietoviikko. http://www.tietoviikko.fi/kehittaja/toita+pythonosaajista+on+pula/a849527 23.5.2013.

Sufyan bin Uzayr. 2012. Is Concrete5 The Right CMS For Your Website? http://www.noupe.com/tools/is-concrete5-the-right-cms-for-your-website-73213.html 29.5.2014.

Tech Magazine for Developers, Designers & SEO Specialists. 2013. Most popular web application frameworks. http://www.hurricanesoftwares.com/most-popular-web-application-frameworks/ 8.4.2013.

Terrar, D. 2011. Why WordPress ISN'T A Good Choice For Your Website (Really?). http://www.cloudave.com/14612/why-wordpress-isnt-a-good-choice-for-your-website-really/ 23.5.2012.

The Django Book. 2009a. Django's History.
http://www.djangobook.com/en/2.0/chapter01.html 15.4.2013.
The Django Book. 2009b. The MVC Design Pattern.
http://www.djangobook.com/en/2.0/chapter01.html
Vaadin. 2013a. How mature is the framework and what is the history of Vaadin?.
https://vaadin.com/faq 8.4.2013
Vaadin. 2013b. Supported Technologies.
http://vaadin.com/download/release/7.0/7.0.6/release-notes.html
23.5.2013
Vaadin. 2013c. When should I not use Vaadin?
https://vaadin.com/faq 23.5.2013
WordPress. 2013a. About WordPress.
http://wordpress.org/about/ 15.5.2013.
WordPress. 2013b. Requirements.
http://wordpress.org/about/requirements/ 15.5.2013.
WordPress. 2013c. Plugin Directory. 29.5.2013.
Zend Framework 2. 2013. About.
http://framework.zend.com/about/ 8.4.2013.