Opinnäytetyö (AMK)

Tietoliikenne ja sähköinen kauppa

Sulautetut ohjelmistot

2011

Crista Hiilesrinne

INTERNETPOHJAINEN TYÖKALU LEHTIEN TAITTAMISEEN

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma | Sulautetut ohjelmistot

Elokuu 2011 | 27 s.

Ohjaajat: FM Olli Niemi

TkL Jari-Pekka Paalassalo

Crista Hiilesrinne

INTERNETPOHJAINEN TYÖKALU LEHTIEN TAITTAMISEEN

Tässä opinnäytetyössä suunniteltiin ja toteutettiin eräänlainen internetpohjainen työkalu lehtien taittamiseen. Tavoitteena oli toteuttaa tarkat määrittelyt työkalun ohjelmoimiseen sekä pieni demonstraatio työkalusta.

Työkalun suunnittelussa käytettin tilakaavioita sekä UML-kaavioita hahmottamaan ja rajaamaan työn laajuutta. Ohjelmointityössä käytettiin XML-teknologiaa yhdessä Javan kanssa.

Työkalun avulla voi ladata omalta tietokoneelta XML-tallennusmuodossa olevan artikkelin palvelimelle ja liittää se lehden sivuksi. Alkuun tekstien ja kuvien paikat olivat vakiona jokaisessa lehdessä. Lehden artikkeleita pystyy myös muokkaamaan suoraan verkossa, eikä niitä aina tarvitse ladata uudelleen jokaisen muutoksen jälkeen.

Opinnäytetyön tuloksena saatiin kattavat määrittelyt työkalun ohjelmoimiseen sekä pieni ohjelma demonstroimaan työkalun käyttöä. Ohjelma tarvitsee vielä paljon jatkokehitystä, mutta hyvin tehty suunnittelu on tärkeä, jotta ohjelmasta voisi tulla hyvä.

ASIASANAT:

XML-teknologiat, Java-ohjelmointi, verkko-ohjelmointi, UML

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology | Embedded programming

August 2011 | 27 pages

Instructors: Olli Niemi, M. Sc

Jari-Pekka Paalassalo, Lic.Tech, Principal Lecturer

Crista Hiilesrinne

TOOL FOR CREATING WEB MAGAZINE

A web-based application for creating web magazines was designed and developed in this thesis. The main goal was to create extensive definitions for programming the tool and a small demo to demonstrate the usage.

UML-models and state transition diagrams were used when designing the tool and planning the scope of the project. The tool was programmed with Java and XML-technologies.

With the tool it is possible to create an own web magazine for example to a sport club. A user should be able to load a XML-document to the tool and tool should read and save it to database. Tool creates a magazine from the articles found on the database.

As a result of this thesis, an extensive definitions and a small demo was created. The tool still needs a lot of further development, but it is going to be easier with a good definitions.

KEYWORDS:

XML-technology, Java programming, UML, web programming

SISÄLTÖ

SYMBOLI- JA LYHENNELUETTELO

HTML	Hypertekstin merkintäkieli (Hypertext Markup Language) on World Wide Webin sivunkuvauskieli.
Java	Sun Microsystemsin kehittämä alustariippumaton oliopohjainen ohjelmointikieli.
MySQL	Norminmukainen kyselykieli (My Structured Query Language), jolla voi tehdä hakuja, muutoksia tai lisäyksiä relaatiotietokantaan.
SGML	Norminmukainen yleismerkintäkieli (Standard Generalized Markup Language) on metakieli, jonka avulla voidaan määritellä dokumenttien merkintäkieliä.
XAMPP	Alustariippumaton (cross-platform) sovellus, joka sisältää mm. Apache HTTP-palvelimen ja MySQL:n.
XML	Laajennettavissa oleva merkintäkieli (Extensible Markup Language) on World Wide Web Consortiumin kehittämä merkintäkieli tai standardi, jolla tiedon merkitys on kuvattavissa tiedon sekaan.
XSL	Laajennettavissa oleva tyylikieli (Extensible Style Language) on XML-sovellus, jolla voidaan kuvata, miten XML-dokumentin sisältö tulisi näyttää lukijalle.

1 JOHDANTO

Internetin käyttö yleistyy koko ajan ja yhä useampien sovellusten halutaan toimivan verkon kautta käyttäjien omilla verkkoselaimilla ilman, että tarvitsee ladata ja asentaa tietokoneelle monia erilaisia ohjelmia. Kaikkien turhien paperien käyttöä pyritään välttämään, koska verkossa on lähes rajattomat tallennustilat ja tallennus verkkoon on ekologinen tapa säilyttää tietoa.

Myös lehdet siirtyvät verkkoon ja verkkolehtien yleisyys kasvaa koko ajan. Helsingin Sanomat voidaan kätevästi lukea aamulla suoraan verkosta ja uutiset päivittyvät verkkosivuille heti, kun jotakin uutisoitavaa tapahtuu. Ei tarvitse enää odottaa, että posti tuo seuraavana aamuna lehden.

Kuitenkaan ei löydy hyvää työkalua, jolla käyttäjä itse voisi luoda oman verkkolehden vaikka oman urheiluseuran tai yrityksen uutisille suoraan verkossa. Lähes aina tarvitsee ladata ja asentaa koneelle erillinen ohjelma lehden taittoa varten. Tähän ongelmaan kehitettiin Yoso Oy:n toimeksiannosta internetpohjainen sovellus, jolla voi helposti luoda oman verkkolehden.

Tässä opinnäytetyössä tehtävällä lehtityökalulla tarkoitetaan internetpohjaista työkalua lehtien taittamiseen. Opinnäytetyössä panostetaan lehtityökalun prosessikuvaukseen ja määrittelyihin enemmän kuin itse sovelluksen tekemiseen. Opinnäytetyön tavoitteena on saada tarkat suunnitelmat ja määrittelyt lehtityökalulle sekä pieni ohjelma demonstroimaan työkalun käyttöä.

2 LÄHTÖKOHDAT

2.1 Työn kartoitus

Opinnäytetyö lähti liikkeelle yrityksen toiveesta saada internetpohjainen lehtityökalu lehtien taittamiseen. Aikaisemmin ei samanlaista työkalua ollut saatavilla, joten työ lähti liikkeelle vaatimusten määrittelyllä. Ensimmäisellä tapaamiskerralla opinnäytetyö päätettiin tehdä projektimuotoisesti, jotta työkalua olisi helppo jatkokehittää opinnäytetyön jälkeen ja jotta jo opinnäytetyötä tehdessä saisi kokemusta projektityöstä.

Tarkkaa kuvausta työkalun ulkonäöstä ei annettu, vaan ohjelman suunnitteluun ja toteuttamiseen sai melko vapaat kädet.

Opinnäytetyön suunnitelmasta tehtiin projektisuunnitelma (Liite 1), jonka pohjalta opinnäytetyötä alettiin tehdä.

Opinnäytetyön etenemistä seurattiin viikoittain kirjoitetulla päiväkirjalla sekä aikaisemmin tehdyllä aikataulusuunnittelulla. Opinnäytetyön etenemistä seurattiin myös noin kahden viikon välein käytävillä tilannekatsauksilla, joista laadittiin myös raportti (Liite 2).

2.2 Työn rajaus

Lehtityökalun tekeminen rajattiin melkein heti alkuun koskemaan pelkästään lehden prosessikuvausta sekä lehtityökalun määrittelyjä, koska opinnäytetyötä oli tekemässä vain yksi henkilö ja työkalun tekeminen aloitettiin ihan alusta. Tavoitteena oli kuitenkin saada valmiiksi pieni sovellus, joka esittelisi työkalun käyttöä, jotta jatkokehitys olisi mahdollisimman helppoa. Sovelluksen testausta tulisi hieman ohjelmoinnin yhteydessä, mutta tarkempi testaus päätettiin jättää vähemmälle ajan puutteen vuoksi.

2.3 Toteutustekniikat

2.3.1 Ohjelmointikieli

Ohjelmointikieli on tärkeä valita oikein, jotta ohjelmoimisesta tulisi mahdollisimman helppoa. Eri ohjelmointikieliä vertailtiin ja pohdittiin, mitkä ohjelmointikielet antaisivat parhaat mahdolliset puitteet sovellukselle. Tässä opinnäytetyössä päädyttiin Javaan ja XML-teknologioihin. Java oli jo entuudestaan tuttu ja se sisältää hyviä ohjelmaosia

valmiina. XML-teknologiat eivät olleet tuttuja, mutta toimeksiantajalla oli valmiiksi asiantuntijoita XML-teknologioille, joten ongelmien esiintyessä olisi helppo kysyä neuvoa.

Java

Java on Sun Microsystemsin kehittämä järjestelmäriippumaton oliopohjainen ohjelmointikieli. Järjestelmäriippumattomuus tarkoittaa, että Java-ohjelmia voi kehittää ja testata esimerkiksi Linuxissa, mutta sama koodi toimii myös esimerkiksi Windowsissa. Java on saanut rakenteensa melko pitkälti C- ja C++-ohjelmointikielistä, mutta eroaa niistä yksinkertaisemmalla oliomallilla ja lisäksi sillä on vähemmän matalatasoisia ominaisuuksia. [1]

Javalla tehtävissä internetpohjaisissa sovelluksissa käytetään Java Servlet ja JavaServer Pagesia (JSP). Näiden avulla voidaan helposti luoda dynaamisia internetsivuja. Java Servlet Technology on rajapinta, jonka avulla voidaan luoda internetpohjaisia sovelluksia Javalla. JSP-sivut ovat valmiita sivupohjia, joihin liitetään Java-koodia asiakkaalle lähetettävän sivun dynaamisten osien luontia varten. Yleensä suurin osa JSP-sivujen sisällöstä on HTML-muotoista, mutta sivuilla voidaan käyttää myös muunlaisia JSP-komponentteja, kuten kirjastoja. JSP:n avulla pystytään luomaan internetsivuja yksinkertaisesti ja helposti.

XML-teknologiat

XML eli Extensible Markup Language (laajennettavissa oleva merkkauskieli) on W3C:n (World Wide Web Consortiumin) suositus rakenteisten dokumenttien merkintäkieleksi tai standardiksi, jolla tiedon merkitys on kuvattavissa tiedon sekaan. XML-kieli on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin.

XML on tekstimuotoista ja muistuttaa HTML-kieltä, jolla WWW-sivut kirjoitetaan. XML-kieli ei kuitenkaan ole tarkoitettu sivunkuvauskieleksi kuten HTML, vaan sillä kuvataan tiedon rakenne ilman ennalta määrättyjä koodeja. XML-kielellä voi muodostaa uusia koodeja, joiden avulla voidaan luoda dokumentteja hyvinkin erilaisiin ja erityisiin tarkoituksiin. [2]

XML-teknologiat sisältävät myös XSL (Extensible Style Language, laajennettavissa oleva tyylikieli) -muunnokset ja –muotoilukielen, joiden avulla on helppo määritellä säännöt sille, miten XML-dokumentti muunnetaan toiseksi XML-dokumentiksi tai miten

dokumentin sisältö tulisi esittää lukijalle. XSL-muotoiluoliot tarjoavat hienostuneemman visuaalisen asettelumallin kuin HTML ja CSS yhdessä. Siihen sisältyvät mm. ei-länsimaalainen asettelu, alaviitteet, marginaalihuomautukset ja ristiviittausten sivunumerot. XSL-muotoilumalli pohjautuu suorakulmionmuotoisiin ”alueiksi” kutsuttuihin laatikoihin, jotka voivat sisältää tekstiä, tyhjää tilaa tai muita muotoiluolioita. XSL-muotoilin lukee muotoiluoliot päätelläkseen, mitä alueita tulee sijoittaa mihinkin sivun kohtaan. XSL-muotoiluoliot ovat täydellinen XML-sanasto, jolla voidaan järjestää elementit sivulle. [3]

2.3.2 AppFuse

Internetpohjainen sovellus on helppo lähteä ohjelmoimaan valmiiden arkkityyppien avulla. Ne ovat valmiita nk. aloitusprojekteja, joita löytyy internetistä monia. AppFuse on avoimen lähdekoodin projekti ja sovellus, joka käyttää Java-alustaan rakennettuja avoimen lähdekoodin työkaluja auttaakseen nopeasti ja tehokkaasti kehittämään verkkosovelluksia. AppFusen projektirunko on samanlainen kuin minkä tahansa ohjelmointiympäristön luoma runko, kun rakennetaan uusi verkkosovelluksen alusta. AppFuse käyttää Mavenia sekä projektin luomiseen että projektin rakentamiseen, testaukseen ja kehittämiseen [4]. AppFuse suosittelee Maven-sovelluksen käyttöä osana verkkosovelluksen kehittämistä. Maven on helppokäyttöinen sovellus projektinhallintaan, sovelluksien kääntämisen ja rakentamisen automatisoimiseen sekä testaukseen ja dokumentointiin. [5]

AppFusen avulla voi luoda monia erilaisia projekteja. Verkon rungoksi voi valita tarpeidensa mukaan joko JSF:n (Java Server Faces), Spring MVC:n, Struts 2:n, Tapestry 5:n tai pelkän verkkopalvelun. Projektin mukana AppFuse tuo Spring Frameworkin, Hibernate Frameworkin sekä käyttäjän valitseman verkon rungon.

Spring Framework on avoimen lähdekoodin sovelluskehys Java alustalle. Spring tarjoaa täydellisen perusrakenteen tuen Java sovellusten kehittämiseen. Muista sovelluskehyksistä poiketen Spring tarjoaa apuvälineitä myös mm. käyttöliittymän sekä tietokantakerroksen toteuttamiseen. [6]

Sovellukseen luodaan Hibernate Framework, joka on avoimen lähdekoodin olio-relaatiomallin ratkaisu. Java-sovelluksesta tietokantaan kytkeydyttäessä tietokantaratkaisun voi tehdä monella tavalla. Perinteisesti käsin koodattu JDBC (Java Database Connectivity) on hidas ja työläs sekä virheitä saattaa tulla helposti. Moderni

olio-relaatiomallin ratkaisutekniikka, esimerkiksi Hibernate, ratkaisee ristiriidan sovittamalla mallit yhteen, jolloin ei tarvitse koodata käsin SQL-kieltä niin suurta määrää. Hibernaten avulla on helpompaa ja tuottavampaa tehdä tietokantakerroksia kuin käsin koodatessa. Hibernate käyttää HQL-kieltä, joka pohjautuu SQL-kieleen. [7]

Tähän opinnäytetyöhön valittiin verkon rungoksi Struts 2, joka on ilmainen, avoimen lähdekoodin ratkaisu Javalla tehtävien verkkosovellusten luomiseen. Verkkosovellukset eroavat normaaleista verkkosivuista monella tapaa. Verkkosovellukset voivat luoda dynamisia sivuja, kun taas verkkosivut yleensä näyttävät vain staattisia sivuja. Strutsin avulla verkkosovellus voi vuorovaikuttaa tietokantakerroksen sekä muiden sovelluksen kerrosten välillä. [8]

AppFuse mahdollistaa valmiin kansio- ja pakettirakenteen sekä kirjastot, joita tarvitaan Java-pohjaisen internetsovelluksen luomiseen ja kehittämiseen.

AppFusen valmiiksi luoma runko sisältää luokkia ja tiedostoja heti käytettäväksi, mm. käyttäjien hallinta, rekisteröityminen, sisäänkirjautuminen ja tiedostojen lataaminen omalta tietokoneelta. Näiden avulla on helppo luoda kokonaan uusia luokkia tai niitä voi käyttää esimerkinomaisesti, kun luodaan ja kehitetään omaa sovellusta. Tällöin säästyy paljon aikaa kokoonpanon asettelussa. Valmiilla rungolla pystyy myös mm. ottamaan toimivan tietokantayhteyden ja vahvistamaan käyttäjän luotettavaksi sisäänkirjautuessaan sovellukseen.

AppFusella voi luoda yksinkertaisen tai monimoduulisen rungon. Monimoduulinen runko luo erikseen Java-projektit sovelluksen ytimelle (engl. core) sekä verkolle (engl. web). Ydin-projektissa ovat kaikki sovelluksen käyttäjälle näkymättömät ohjelmistokoodit, kuten DAOt (Data Access Object) ja hallintaluokat, joiden päätehtävä on luoda silta DAOn ja käyttäjälle näkyvän verkkon välille. Verkko-projektissa ovat ulkoasuun liittyvät luokat, kuten JSP-sivut sekä Action-luokat, joiden metodien tehtävänä on muokata esimerkiksi käyttäjän tietoja, mikäli käyttäjä niitä muokkaa. [4]

2.3.3 Tietokanta

Lehteen tulevat artikkelit sekä julkaistut lehdet halutaan tallentaa ja tätä varten tarvitaan tietokanta. Tietokannasta sovelluksen on helppo hakea ja muokata artikkeleita.

AppFuse suosittelee tietokannaksi ehkä kaikken suosituinta avoimen lähdekoodin tietokantaa, MySQL:ää. Sitä voi käyttää monella eri alustalla, se on todella luotettava, nopea ja helppo käyttää. [9] MySQL mahdollistaa tietokantojen sisällön eli tiedon käsittelyn ja rakenteen määrittelyn. Tietokantapalvelin hoitaa haun ja toimittaa vastauksen annettuun kyselyyn nopeasti, koska tietokantapalvelin indeksoi tietonsa sisäisesti mahdollistaakseen nopeat haut tietokannassa olevaan tietoon. Tietokantapalvelimen sisäisestä toiminnasta sovelluskehittäjän ei yleensä tarvitse huolehtia, sillä hän käyttää ainoastaan SQL-kyselyjä. [1]

MySQL on helppo asentaa tietokoneelle erillisen XAMPP-sovelluksen avulla. XAMPP on ilmainen avoimen lähdekoodin sovellus, joka sisältää mm. Apache palvelimen ja MySQL-tietokannan. [10]

2.3.4 Ohjelmointiympäristö

Ohjelmointiympäristöksi on monia vaihtoehtoja. Suosituimpia ympäristöjä ovat NetBeans sekä Eclipse. Tässä opinnäytetyössä päädyttiin käyttämään NetBeansia, koska se oli valmiiksi tuttu.

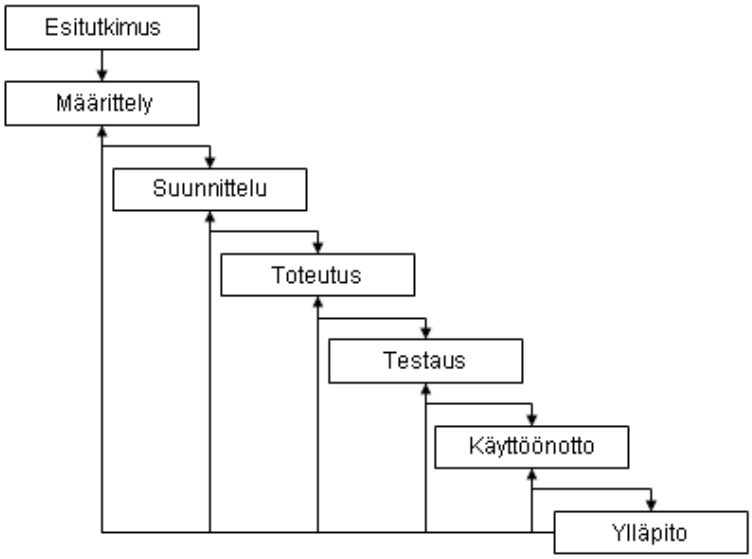
NetBeans on Oracle Corporationin kehittämä ilmainen, avoimen lähdekoodin ohjelmointiympäristö sovelluskehittäjille. Ohjelmointiympäristö sisältää työkalut sovelluksen ohjelmoimiseen, kääntämiseen, ajoon ja testaukseen. NetBeansin avulla voi ohjelmoida Javan lisäksi myös mm. JavaScriptiä, PHP:tä, C-kieltä ja C++-kieltä. [11]

Tässä opinnäytetyössä tehdyt prosessikuvaukset ja tilakaaviot päätettiin tehdä Aris Express -ohjelmalla, joka on ilmainen prosessikuvaukseen tarkoitettu ohjelma. Ohjelman avulla on helppo ja nopea luoda yksinkertaisia prosessikuvauksia, eikä se vaadi prosessinhallinnan syvällistä osaamista. Aris Express tarjoaa käyttäjän käyttöön erilaisia diagrammeja ja monenlaisia objekteja. Ohjelman verkkosivuilla on paljon esimerkkejä ja tutoriaaleja aloittelijoille. [12]

3 TUOTANTOPROSESSI

Opinnäytetyön toteutus päätettiin tehdä ohjelmistotuotannossa tunnetulla prosessilla [13], jota kutsutaan myös vesiputousmalliksi (kuva 1). Tämän mallin mukaan ohjelmistoprosessi lähtee liikkeelle päävaatimusten määrittelyllä ja sitä seuraavat

sovelluksen suunnittelu, toteutus, testaus, käyttöönotto ja ylläpito. Tämän opinnäytetyön aikana ehdittiin tehdä vain kaksi ensimmäistä osiota kunnolla sekä kolmatta osaa vain osittain.



Kuva 1. Kuva vesiputousmallista [13]

3.1 Päävaatimusten määrittely

Lehtityökalun tulisi olla yksinkertainen ja helppokäyttöinen. Työkalun tarkoituksena on saada helposti luotua lehti, jossa on paikat artikkeleille ja kuville sekä siirtää materiaalit tietokannasta lehteen. Alkuun lehden sivut sekä tekstien ja kuvien paikat ovat vakioita ja myöhemmin tulisi olla mahdollisuus laajentaa työkalua siten, että tekstien ja kuvien paikat voisi itse määrittää.

Lehtityökalulle tulisi antaa artikkelit XML-tiedostoformaatissa ja työkalun tulisi osata lukea artikkelin tiedot ja lisätä ne lehteen.

Työkalulla pitäisi saada lehti rakennettua helposti sekä tallennettua valmis lehti tietokantaan. Myös vanhoja lehtiä olisi hyvä pystyä selaamaan. Tässä opinnäytetyössä rajattiin kuitenkin lehtien tallentaminen ja vanhojen lehtien selaaminen pois ajan puutteen vuoksi.

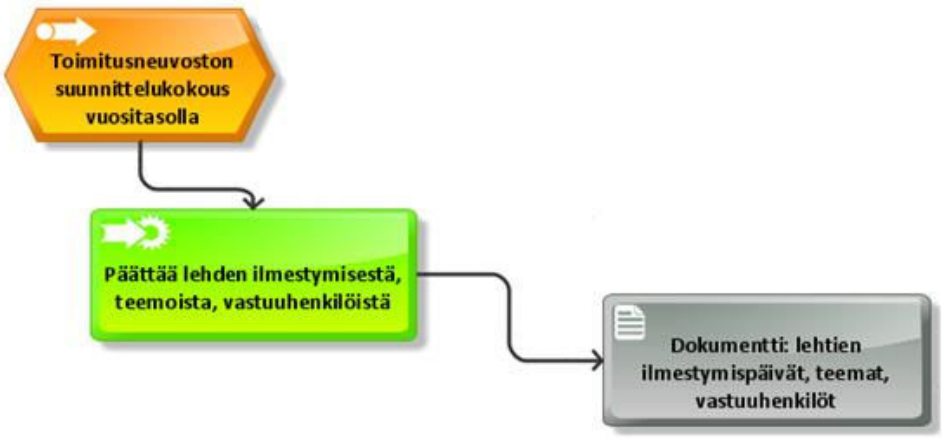
3.2 Suunnittelu

Lehtityökalua lähdettiin suunnittelemaan piirtämällä karkea prosessikuvaus lehden prosessista, jotta olisi hyvä ymmärrys, mitä tarkoitetaan lehtiprosessilla ja mikä osa

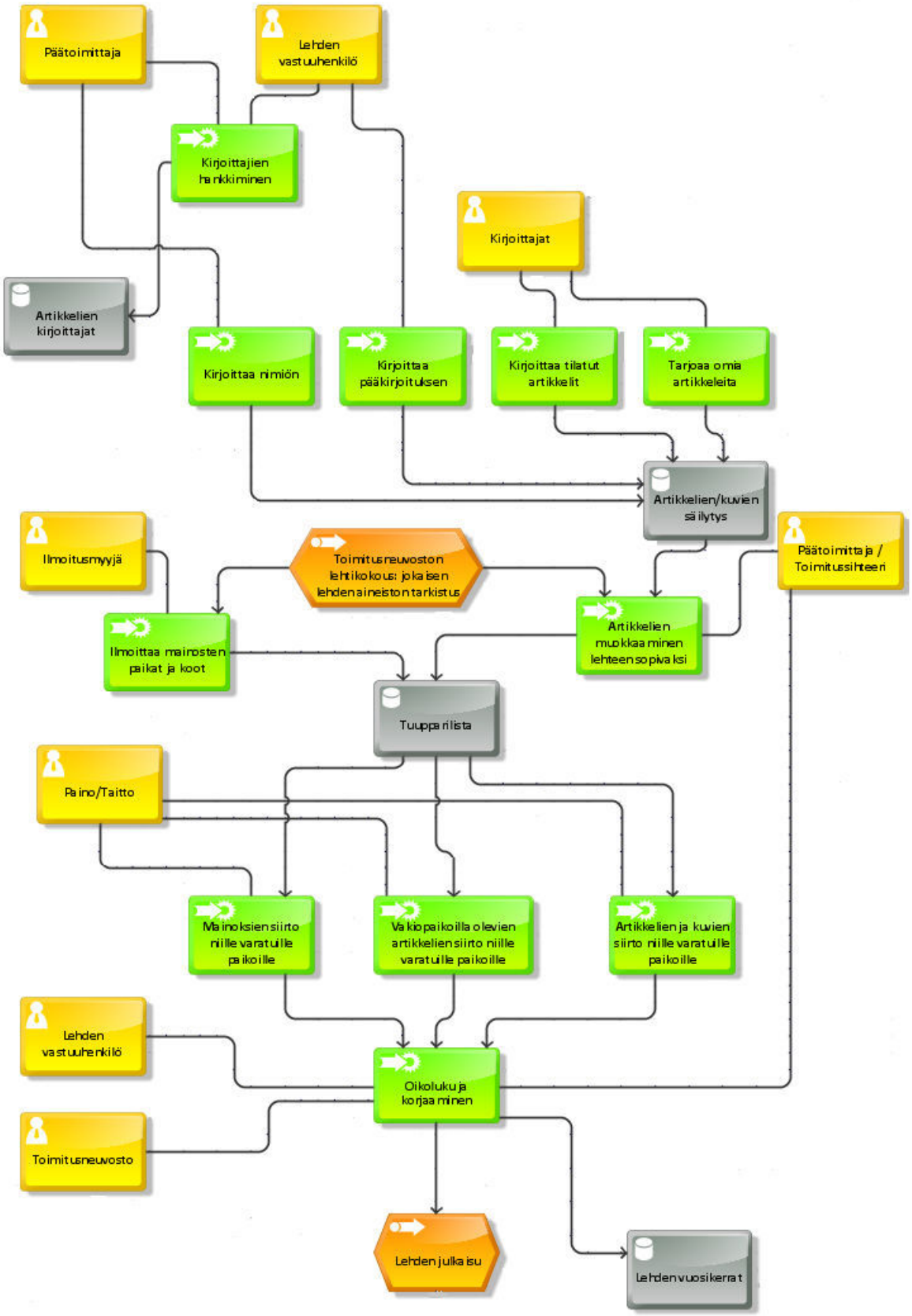
lehtiprosessista tulisi toteuttaa työkaluna. Prosessikuvausta tarkennettiin koskemaan pelkästään taittamisen osaa, eli ohjelmoitavaa sovellusta ja tästä piirrettiin tarkempi UML-kaavio.

3.2.1 Lehtiprosessi

Lehtiprosessi lähtee käyntiin toimitusneuvoston suunnittelukokouksella (kuva 2), jossa päätetään lehtien ilmestymispäivät, teemat, vastuuhenkilöt ja muut lehden kannalta tarpeelliset asiat. Tämän jälkeen lehden päätoimittaja yhdessä lehden vastuuhenkilön kanssa hankkii artikkelien kirjoittajat, jotka kirjoittavat lehden teeman mukaiset artikkelit. Artikkelit tallennetaan tietokoneelta tietokantaan. Päätoimittaja ja toimitussihteeri käyvät artikkelit läpi ennen jokaista lehden taittamista ja mahdollisesti muokkaavat tai pyytävät kirjoittajia muokkaamaan niitä lehteen sopivammaksi. Tämän jälkeen lehden taittaja lisää artikkelit ja kuvat omille paikoillen lehteen. Lehden vastuuhenkilö sekä päätoimittaja oikolukevat lehden vielä ennen lehden julkaisua ja hyväksytty lehti voidaan julkaista. Julkaistut lehdet tallennetaan myös tietokantaan, jotta niitä voidaan selata myöhemmin. (kuva 3)

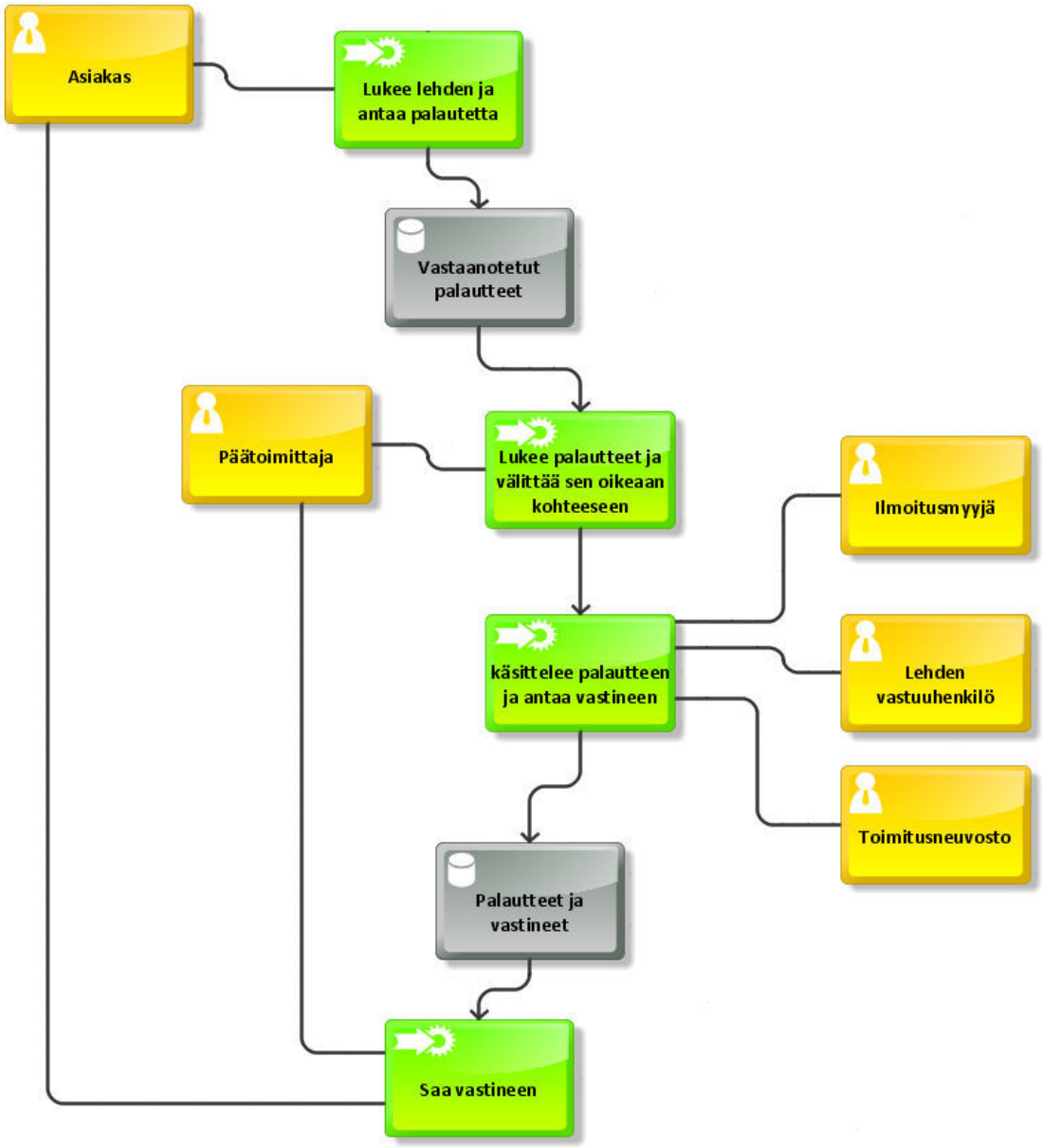


Kuva 2. Suunnittelukokous



Kuva 3. Lehtiprosessi

Asiakas voi lukea lehden omalla web-selaimellaan ja halutessaan antaa siitä palautetta. Päätoimittaja lukee palautteet ja välittää ne oikeille henkilöille, kuten lehden vastuuhenkilölle tai toimitusneuvostolle. Palaute käsitellään ja siitä annetaan vastine, joka lähetetään asiakkaalle. Palautteet ja vastineet tallennetaan myös tietokantaan, jotta niiden avulla voidaan lehteä kehittää edelleen paremmaksi. (kuva 4)

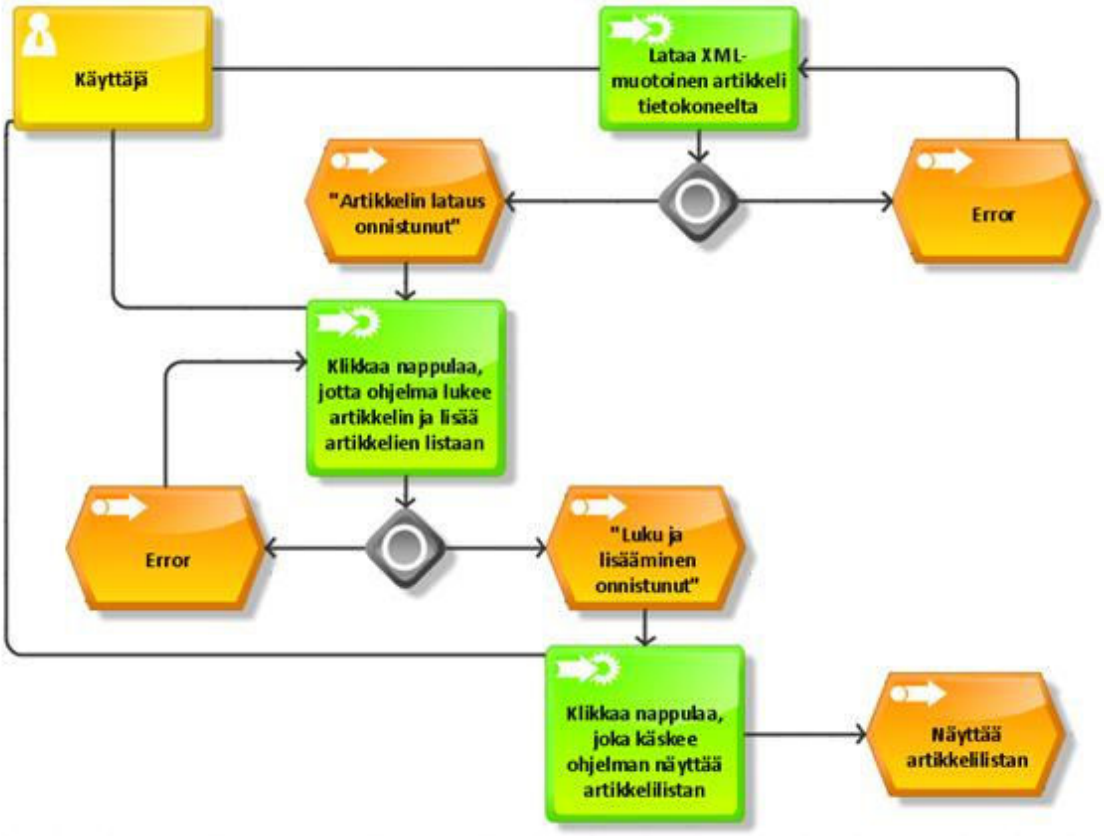


Kuva 4. Lehden palaute

3.2.2 Lehden taittaminen

Lehden taittaminen käsittää siis artikkelien lisäämisen tietokantaan, artikkelien muokkaamisen sekä artikkelien lisäämisen lehden sivuille.

Artikkeleita varten luodaan oma lista, jonka mukaan lehti tehdään. Artikkelin lisääminen listaan vaatii käyttäjää lataamaan XML-tiedostoformaatissa olevan artikkelin ohjelmaan, josta se luetaan ja lisätään tietokantaan sekä artikkelien listaan, joka näytetään käyttäjälle (kuva 5).



Kuva 5. Artikkelin lisääminen artikkelien listaan

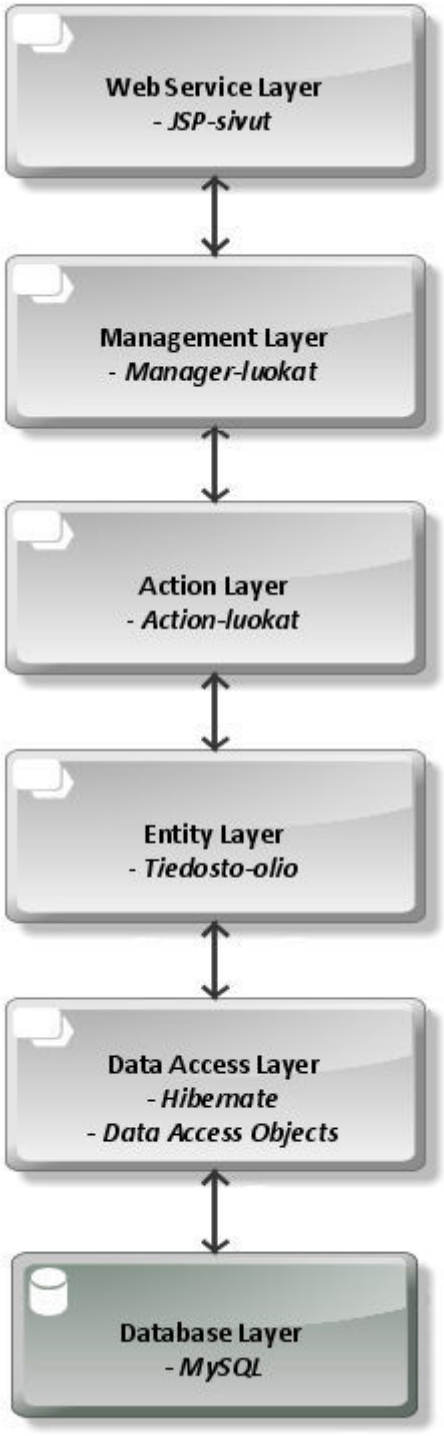
Artikkeleita pitää päästä muokkaamaan sovelluksessa, mikäli niissä esiintyy virheitä tai päätoimittaja haluaa jotain muita muutoksia, jotta ei artikkelia tarvitse uudelleen ladata tietokantaan jokaisen muutoksen jälkeen. Tätä varten pitää luoda sovellukseen erillinen sivu, jossa artikkelia on helppo muokata. Sivulla tulisi olla erilliset tekstikentät kaikkien artikkelin osien muokkaamiseen.

Samaa sivua voisi käyttää myös uuden artikkelin luomiseen suoraan verkossa. Mikäli muokattavalla artikkelilla ei ole tunnistenumeroa eli mikäli artikkeli on tyhjä, luodaan uusi artikkeli, jolle annetaan tiedot tekstikenttiin ja tallennetaan tietokantaan.

Lisäksi kaikki artikkelit pitää saada lisättyä lehden sivuille kätevästi ja lehden ulkoasun tulisi olla selkeä.

3.2.3 Sovelluksen arkkitehtuuri

AppFusen luoman verkkosovelluksen arkkitehtuuri muodostuu kuudesta eri kerroksesta (kuva 6). Alimpana on tietokantakerros (Database layer), joka säilyttää kaiken lehdessä käytettävän tiedon. Tiedonhaku kerrokseen (Data Access layer) kuuluu Hibernate Framework sekä Data Access Objektit eli DAOt, jotka hakevat ja tallentavat entiteettikerroksen (Entity layer) Tiedosto-olion tiedot tietokantaan. Toiminnallisuuskerroksella (Action layer) Action-luokat sisältävät metodeja Tiedosto-olion toiminnallisuuteen, kuten muokkaamiseen ja poistamiseen. Toiminnallisuuskerros kohdistuu osittain myös hallintakerroksen (Management layer) Manager-luokkiin, joissa on Tiedosto-olion hallintaan liittyviä metodeja. Viimeisenä kerroksena on verkkopalvelu kerros (Web Service layer), joka sisältää mm. käyttäjälle näkyvät JSP-sivut.



Kuva 6. Verkkosovelluksen arkkitehtuuri

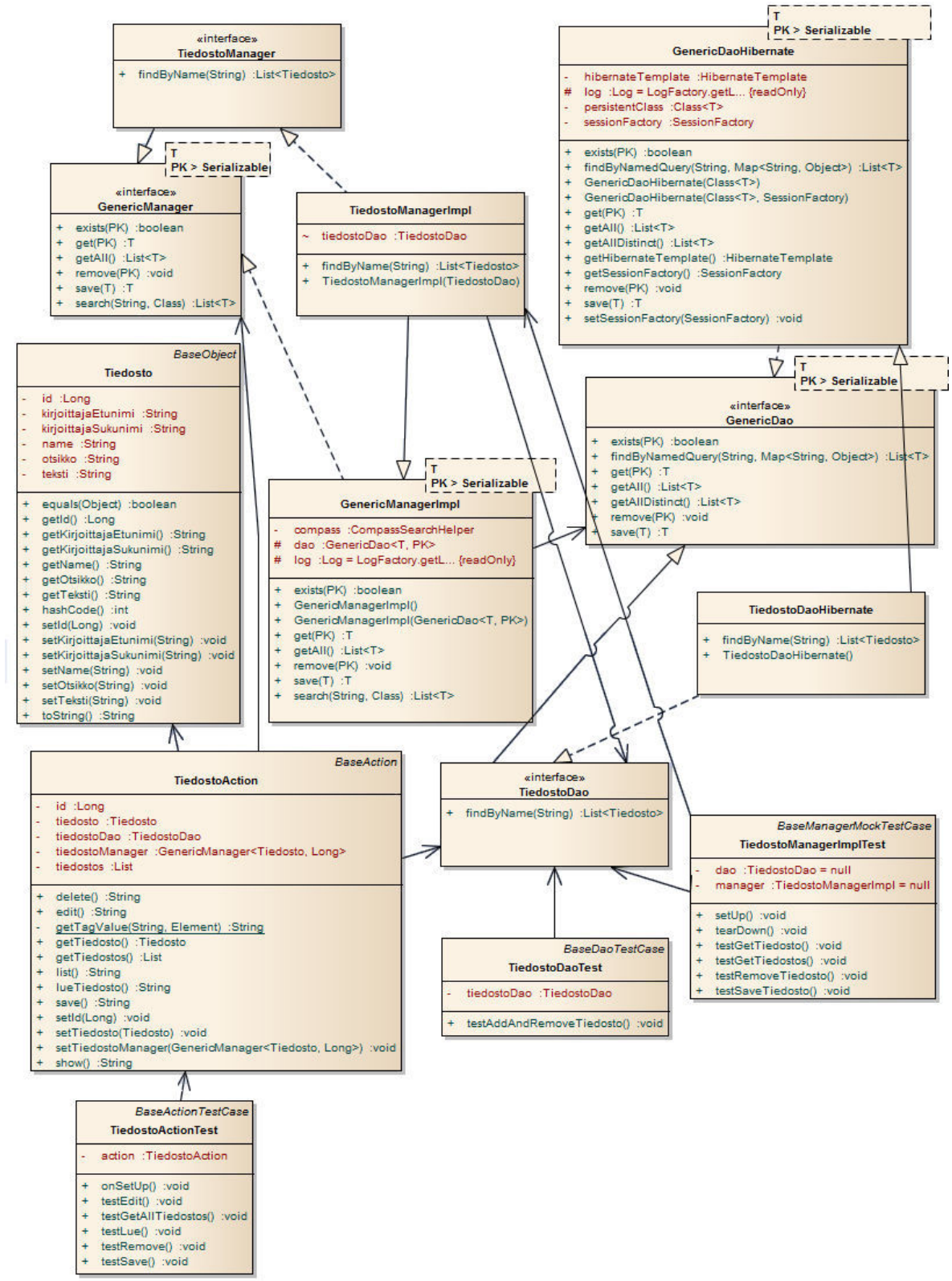
3.2.4 Ohjelmoitava sovellus

Artikkelit pitää erottaa toisistaan, joten jokainen artikkeli saa tunnistenumeron (id), joka tulee automaattisesti, eikä käyttäjän tarvitse siihen vaikuttaa. Artikkeleilla on myös nimi, otsikko, itse artikkeliteksti sekä kirjoittaja. Lisäksi kirjoittajan tiedot jaettiin kahteen osaan, eli etunimelle ja sukunimelle määritettiin omat tietonsa.

Tiedosto-olio eli artikkeli luodaan antamalla tiedostolle tarvittavat tiedot set*-metodien avulla ja tietyn tiedoston tiedot voi hakea get*-metodien avulla. Tähtien kohdalle tulee annettavan tai haettavan tiedon nimi, esimerkiksi setOtsikko(”Tekstin otsikko”), jolloin kyseisen artikkelin otsikoksi tulisi ”Tekstin otsikko”.

Artikkelien muokkaamiseen tarvitaan TiedostoAction-luokka, jossa sijaitsevat metodit mm. artikkelin lisäämiseksi listaan, XML-tiedostoformaatissa olevan artikkelin lukemiseen ja tallentamiseen. Yhteys tietokantaan muodostetaan Data Access Objectin (DAOn) ja Hibernaten avulla.

Tarvittavat luokat ja metodit suunniteltiin luomalla UML-kaavio (kuva 7). UML-kaaviota jouduttiin opinnäytetyön edetessä myös tarkentamaan.



Kuva 7. UML-kaavio taitto-ohjelman toiminnasta

3.3 Toteutus

3.3.1 Sovelluksen runko

Toteutuksessa käytettiin osittain nk. testivetoista kehitystä (Test-Driven Development), joka tarkoittaa, että testausluokkia kirjoitettiin ennen ohjelmaluokkia. Testiluokat on helpompi kirjoittaa ensin, koska tiedetään, mitä ohjelman tulisi tehdä ja miten sitä tulisi testata. Tällöin luodaan ensin tavoitteet, joiden ympärille pyritään luomaan ohjelma. Kun testi menee läpi, tiedetään, että ohjelma tekee sen, mitä halutaan. Testivetoinen kehitys ei kuitenkaan ole testausmenetelmä, vaan ennemminkin suunnittelumenetelmä, vaikka tuloksena syntyy käyttökelpoisia testejäkin.

Toteutus aloitettiin rakentamalla monimoduularinen runko AppFusen avulla. Ydin-projektiin luotiin ensin Tiedosto-olio, joka määrittelee, mitä kaikkia tietoja artikkeli tarvitsee.

Artikkelin saama id generoidaan automaattisesti. Tämä saadaan aikaan seuraavalla koodilla:

Koodi 1. Automaattinen generointi id-arvolle

```
@Id @GeneratedValue(strategy = GenerationType.AUTO)
public Long getId() {
    return this.id;
}
```

3.3.2 Tietokantayhteys

Tämän jälkeen alettiin luomaan tietokantayhteyttä, jotta artikkelit on helppo tallentaa ja hakea. Tiedosto-oliolle luotiin DAO eli objekti, joka on yhteydessä tietokantaan. Ennen TiedostoDaon luomista luotiin testiDao, joka testaa TiedostoDaon avulla yhteyden tietokantaan löytääkseen nimen avulla artikkelin (koodi 2).

Koodi 2. TiedostoDAOn testauksen käytettävä metodi

```
@Test
public void testFindByName() throws Exception {
    List<Tiedosto> tiedostot = tiedostoDao.findByName("TekstinNimi");
    assertTrue(tiedostot.size() > 0);
}
```

Koodissa 2 oleva @Test-annotaatio ilmaisee Hibernatelle kyseessä olevan testimetodi.

Seuraavaksi luotiin TiedostoDao (koodi 3), joka tarvitsee toteutuakseen myös Hibernate Frameworkin, joka on yksi valmiista osista AppFusen rungossa ja auttaa tietokantayhteyden luomisessa merkittävästi.

Koodi 3. TiedostoDao-luokka – TiedostoDao.java

```
import org.crista.dev.dao.GenericDao;
import org.crista.dev.cr.filehandler.model.Tiedosto;
import org.crista.dev.cr.filehandler.dao.hibernate.TiedostoDaoHibernate

import java.util.List;

public interface TiedostoDao extends GenericDao<Tiedosto, Long> {
    public List<Tiedosto> findByName(String name);
}
```

TiedostoDao käyttää Hibernate Frameworkia hakeakseen tietoja tietokantataulusta. Hibernatelle luotiin oma luokka, joka sisältää tarvittavan metodin (koodi 4).

Koodi 4. TiedostoDaoHibernate.javan metodi, jolla haetaan tietokannasta

```
public List<Tiedosto> findByName(String name) {
   return getHibernateTemplate().find("from Tiedosto where name=?", name);
}
```

Näin saatiin yhteys tietokantaan ja tietokannasta haettua tiedosto nimen avulla. TiedostoDao käyttää myös GenericDao-luokkaa, joka yhdessä GenericDaoHibernate-luokan kanssa sisältää metodit mm. tiedostojen tallentamiseen ja poistamiseen tietokannasta.

3.3.3 Sovelluksen toiminnallisuudet

Käyttäjä lataa sovellukseen XML-tiedostoformaatissa olevan tiedoston. Tiedostossa tulee olla juurielementtinä lehti. Lehti-elementti voi sisältää useampiakin sivu-elementtejä, mutta sivu-elementti voi sisältää vain yhden nimen, otsikon, tekstin sekä kirjoittajan etu- ja sukunimen. Nämä ovat kuitenkin pakollisia, eli niitä ei voi jättää pois. Nimi, otsikko, teksti, kirjoittajan etunimi ja sukunimi voivat sisältää vain jäsennettyä merkkimuotoista tietoa eli tekstiä, joka ei ole merkintä. Koodi 5 esittelee esimerkin oikeellisesta XML-tiedostosta, joka tulisi ladata sovellukselle.

Koodi 5. Esimerkki sovellukselle ladattavasta XML-tiedostosta

```
<?xml version="1.0"?>
<!DOCTYPE lehti [
		<!ELEMENT lehti (sivu+) >
		<!ELEMENT sivu (nimi, otsikko, teksti, kirjoittajaEtunimi, kirjoittajaSukunimi) >
		<!ELEMENT nimi (#PCDATA)>
		<!ELEMENT otsikko (#PCDATA)>
		<!ELEMENT teksti (#PCDATA)>
		<!ELEMENT kirjoittajaEtunimi (#PCDATA)>
		<!ELEMENT kirjoittajaSukunimi (#PCDATA)>
]>
<lehti>
		<sivu>
			<nimi>TeksinNimi</nimi>
			<otsikko>Tekstin Otsikko</otsikko>
			<teksti>Artikkelin teksti </teksti>
			<kirjoittajaEtunimi>Crista</kirjoittajaEtunimi>
			<kirjoittajaSukunimi>Hiilesrinne</kirjoittajaSukunimi>
		</sivu>
</lehti>
```

AppFusella on tiedoston lataamiseen tietokoneelta valmis UploadFile-luokka, jonka avulla on helppo tuoda tiedostoja tietokoneelta. Tiedostoja ei kuitenkaan lisätä automaattisesti tietokantaan, joten tälle tapahtumalle piti luoda oma metodinsa (Koodi 6). Sovelluksen pitää lukea XML-tiedosto ja lisätä tiedoston tiedot Tiedosto-olion avulla tietokantaan.

Koodi 6. XML-tiedoston lukuun käytettävä metodi TiedostoAction-luokassa

```
public String lueTiedosto() {
    String nimi;
    String otsikko;
    String teksti;
    String kirjoittajaEtu;
    String kirjoittajaSuku;

    try {
```

```
//luetaan file.xml
    File fXmlFile = new
File("C:/Source/cristanoppari/web/src/main/webapp/resources/file.xml");
    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    Document doc = dBuilder.parse(fXmlFile);
    doc.getDocumentElement().normalize();

//haetaan listaan kaikki elementit, joka alkaa sivu-elementistä
    NodeList nList = doc.getElementsByTagName("sivu");

//käydään lista läpi ja tallennetaan
    for (int i = 0; i < nList.getLength(); i++) {

       Node nNode = nList.item(i);
       if (nNode.getNodeType() == Node.ELEMENT_NODE) {
          Element eElement = (Element) nNode;

//tallennetaan file.xml:n elementtien tiedot
          nimi = getTagValue("nimi",eElement);
          otsikko = getTagValue("otsikko",eElement);
          teksti = getTagValue("teksti",eElement);
          kirjoittajaEtu = getTagValue("kirjoittajaEtunimi",eElement);
          kirjoittajaSuku = getTagValue("kirjoittajaSukunimi",eElement);

          addTiedosto(nimi,otsikko,teksti,kirjoittajaEtu,kirjoittajaSuku);
       }
    }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  return SUCCESS;
}
```

lueTiedosto()-metodi tarvitsee erillisen metodin saadakseen XML-tiedoston tiedot. Koodi 7 esittelee tämän metodin.

Koodi 7. Metodi XML-tiedoston tagien arvojen saamiseksi

```
 //Metodi elementtien lukemiseen xml-tiedostosta
private static String getTagValue(String sTag, Element eElement){
  NodeList nlList=eElement.getElementsByTagName(sTag).item(0).getChildNodes();
```

```
  Node nValue = (Node) nlList.item(0);
  return nValue.getNodeValue();
}
```

Kun artikkelin tiedot on tallennettu, ne pitää vielä lisätä Tiedosto-olion avulla tietokantaan koodin 8 mukaisesti.

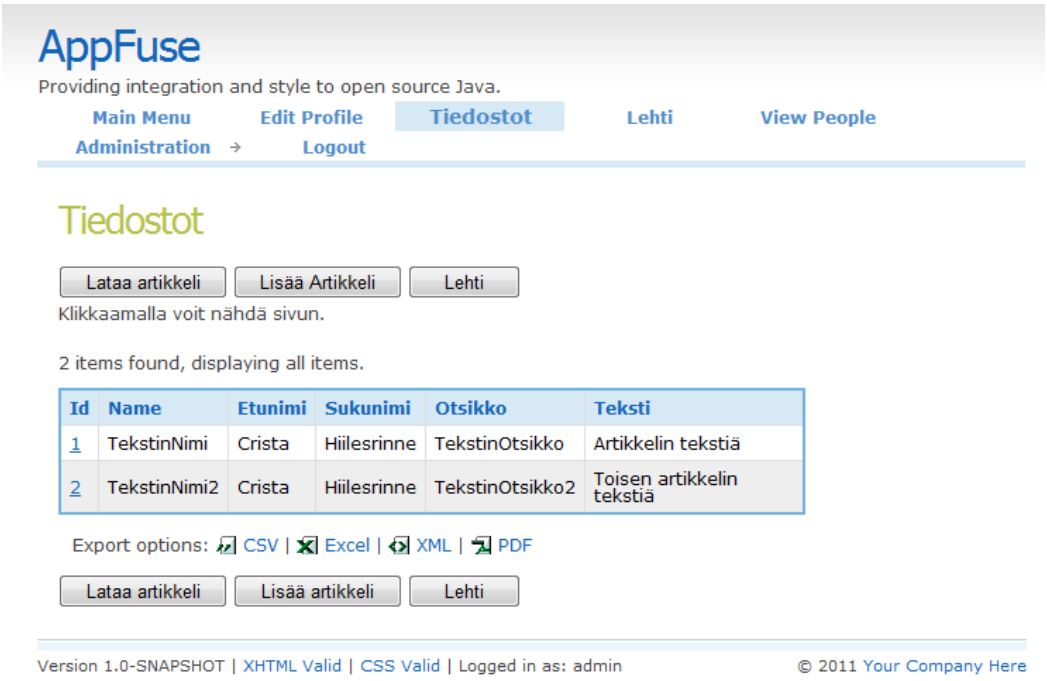
Koodi 8. Metodi artikkelin lisäämiseksi tietokantaan

```
//Metodi tietojen lisäämiseksi Tiedosto-olioon
public void addTiedosto(String nimi, String otsikko, String teksti, String
kirjoittajaEtu, String kirjoittajaSuku) throws Exception {
        Tiedosto artikkeli = new Tiedosto();
        artikkeli.setName(nimi);
        artikkeli.setOtsikko(otsikko);
        artikkeli.setTeksti(teksti);
        artikkeli.setKirjoittajaEtunimi(kirjoittajaEtu);
        artikkeli.setKirjoittajaSukunimi(kirjoittajaSuku);
//Tallennetaan artikkeli tietokantaan
        artikkeli = tiedostoManager.save(artikkeli);
//Tulostetaan käyttäjälle tieto, että tiedosto on lisätty onnistuneesti.
        String key = "tiedosto.added";
        saveMessage(getText(key));
    }
```

Sovellus ilmoittaa, kun tiedosto on lisätty onnistuneesti (kuva 8).

Kuva 8. Ilmoitus, kun tiedosto on lisätty onnistuneesti.

Sovellus näyttää tietokantataulun artikkelien listasta kuvan 9 mukaisesti.



Kuva 9. Lista tietokannassa olevista artikkeleista (http://localhost:8080/tiedostos)

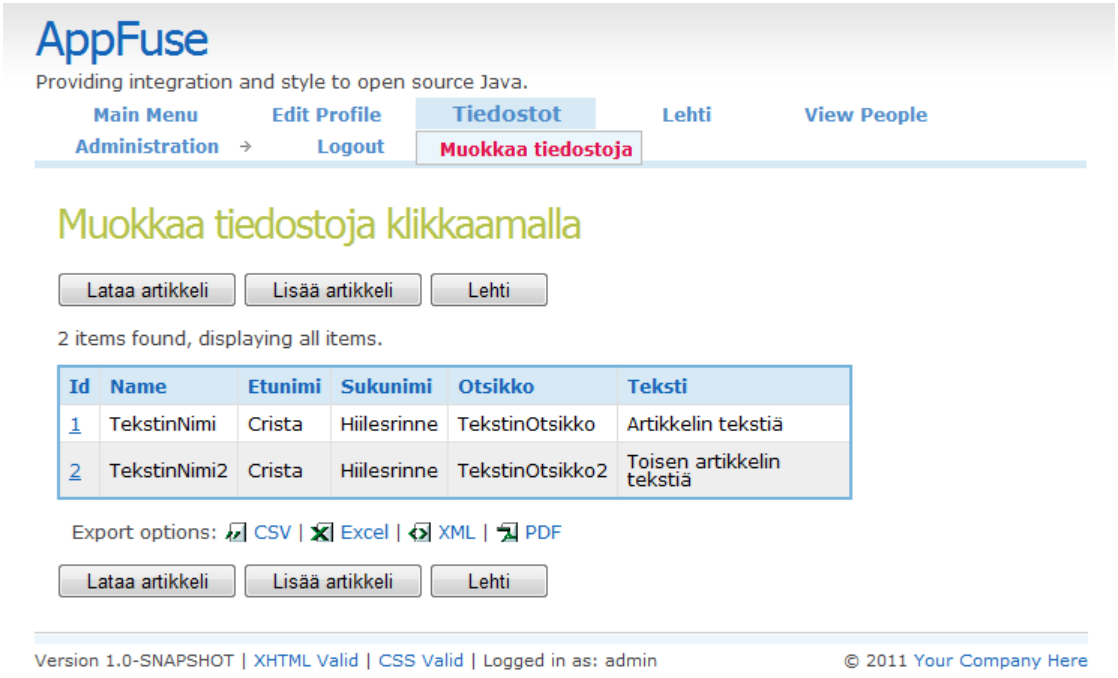
Artikkelin voi ladata tietokoneelta napauttamalla ”Lataa artikkeli”, jolloin sovellus näyttää AppFusen valmiin uploadFile-sivun. Artikkelin voi myös lisätä suoraan verkossa napauttamalla ”Lisää artikkeli”, jolloin tulee esille sivu, johon voi itse kirjoittaa artikkelin nimen, otsikon, tekstin ja kirjoittajan nimet ja tallentaa tiedot. ”Lehti”-painiketta napauttamalla sovellus näyttää kaikki artikkelit allekkain (kuva 10). Tässä opinnäytetyössä ei kovin paljoa kiinnitetty huomiota lehden ulkoasuun. Ulkoasuun voisi vaikuttaa antamalla sovellukselle erillisen XSL-dokumentin, joka määrittelisi artikkelin osien paikat lehdessä.

Kuva 10. Lehden tekstit allekkain lehden sivulla (http://localhost:8080/lehti).

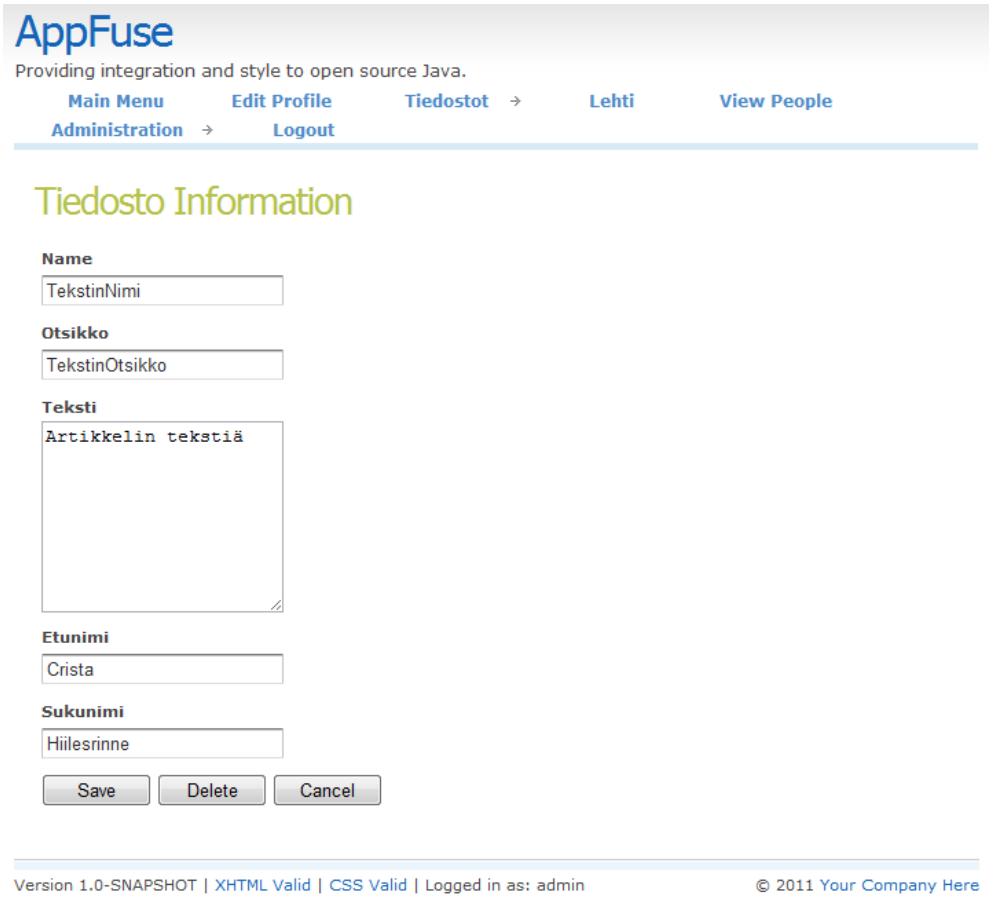
Lehden artikkelit saa myös yksitellen näkyviin (kuva 11) napauttamalla listasta kyseisen artikkelin nimeä.

Kuva 11. Artikkelit saa näkyviin myös yksitellen (http://localhost:8080/sivut?id=1)

Tiedostojen muokkaaminen tapahtuu Tiedostot-välilehden alasivulta ”Muokkaa tiedostoja” (kuva 12). Sivu on muuten samanlainen kuin aikaisemmin, mutta nyt tiedoston nimeä napauttamalla pääsee muokkaamaan tiedoston tekstejä (kuva 13).



Kuva 12. Tiedostojen muokkaaminen (http://localhost:8080/muokkaa)



Kuva 13. Tiedoston muokkaamiseen tarkoitettu sivu (http://localhost:8080/editTiedosto?id=1)

Kun sovellus julkaistaan, tulisi asetuksia muokata siten, että lehden toimittajan kirjautuessa sisälle näkyisi tiedostot ja tiedostojen muokkaaminen. Mikäli ei kirjaudu sisälle, tulisi näkyä vain lehti, eikä sitä pystyisi muokkaamaan. Tällöin eivät ulkopuoliset käyttäjät pääse sekoittamaan lehden sisältöä.

4 YHTEENVETO

Tässä opinnäytetyössä suunniteltiin ja toteutettiin internetpohjainen sovellus verkkolehtien taittamiseen. Tulokseksi saatiin pieni sovellus, joka havainnollistaa työkalun käytön.

Lopputuloksena saatiin kattavat määrittelyt ja suunnitelma lehtityökalun toiminnasta. Sovelluksen avulla asiakkaan on helppo luoda ja hallita omia verkkolehtiä suoraan verkossa. Artikkelien muokkaaminen on yksinkertaista ja sen voi tehdä suoraan verkossa, eikä jokaista artikkelia tarvitse ladata tietokantaan uudelleen jokaisen korjauksen jälkeen.

Lehtityökalu tällaisenaan on vielä hyvin vajaavainen ja vaatii paljon jatkokehittämistä, erityisesti lehden ulkonäön kannalta. Lehteen olisi hyvä luoda mahdollisuus omien XSL-dokumenttien mukaiseen ulkonäköön. Tehtyjen määrittelyjen perusteella sovelluksen kehittäminen on kuitenkin helppoa.

Valmista sovellusta voisi mahdollisesti käyttää myös muiden tekstien julkaisemiseen, esimerkiksi aloitteleva kirjailija voi julkaista vaikka novellejaan tai kokki voi julkaista reseptejään. Lehtityökalun avulla voisi julkaista melkeinpä mitä tahansa tekstiä sisältäviä tiedostoja.

LÄHTEET

[1] Peltomäki, Juha. Silander, Simo. Kosonen, Pekka. ”Java 2 – ohjelmoinnin peruskirja”. WSOY. 2005

[2] “Extensible Markup Language (XML) 1.0 (Fifth Edition)”. [www-dokumentti]. Saatavilla: http://www.w3.org/TR/REC-xml/ (luettu 4.3.2011)

[3] Harold, Elliote. “Tehokäyttäjän opas – XML”. Suomentanut Saxberg, Pekka. Gummerus. 2000

[4] “AppFuseDefinition”. [www-dokumentti]. Saatavilla: http://raibledesigns.com/wiki/Wiki.jsp?page=AppFuseDefinition (luettu: 21.7.2011)

[5] “Maven”. [www-dokumentti]. Saatavilla: maven.apache.org (luettu: 10.5.2011)

[6] ”Spring Framework – Reference Dokumentation”. [www-dokumentti]. Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/ (luettu: 29.7.2011)

[7] ”Hibernate”. [www-dokumentti]. Saatavilla: http://www.appfuse.org/display/APF/Hibernate (luettu: 21.7.2011)

[8] ”Apache Struts”. [www-dokumentti]. Saatavilla: http://struts.apache.org/ (luettu: 31.7.2011)

[9] ”MySQL”. [www-dokumentti]. Saatavilla: www.mysql.com/why-mysql (luettu: 10.5.2011)

[10] ”Apache friends - xampp ”. [www-dokumentti]. Saatavilla: http://www.apachefriends.org/en/xampp.html (luettu: 10.5.2011)

[11] ”NetBeans IDE features”. [www-dokumentti]. Saatavilla: http://netbeans.org/features/index.html (luettu: 10.5.2011)

[12] ”Aris Express”. [www-dokumentti]. Saatavilla: http://www.ariscommunity.com/aris-express (luettu: 21.7.2011)

[13] ”Ohjelmistotuotantoprosessi”. [www-dokumentti]. Saatavilla: http://www.oamk.fi/sbc/ohjelmistotuote/ohjelmistotuotanto/ohjelmistotuotantoprosessi.htm (luettu 4.3.2011)

LIITTEET

Liite 1 Projektisuunnitelma

Liite 2 Tilannekatsaukset

LEHTITYÖKALU

PROJEKTISUUNNITELMA

CRISTA HIILESRINNE

17.2.2011

SISÄLLYSLUETTELO

1. Yleistä

Lehtityökalu on selainpohjainen työkalu lehden taittamiseen. Työkalu tehdään projektimuotoisesti ja sen tekee Turun Ammattikorkeakoulun tietotekniikan opiskelija opinnäytetyökseen.

Tämän projektisuunnitelman on tarkoitus olla ohjenuorana, mitä projektin jälkeen tulisi olla valmiina. Koko lehtityökalua ei tehdä tämän projektin aikana, vaan työ koostuu lähinnä lehtityökalun suunnittelusta ja määrittelemisestä sekä pienestä demosta.

2. Projektin laajuus ja tulokset

Projektityö on jaettu karkeasti kolmeen alueeseen. Prosessikuvaukseen, määrittelyyn sekä testaukseen ja toteutukseen.

2.1. Prosessikuvaus

Projekti aloitetaan tekemällä prosessikuvaus yleisesti lehtiprosessista, josta rajataan erityisesti lehden taittamiseen kuuluva osa ja tätä osaa tarkennetaan tilakaavioilla. Opinnäytetyön osaa analysoidaan ja mahdollisesti rajataan lisää, mikäli tarpeellista. Myös toiminnallisuutta ja ei-toiminnallisuutta pohditaan.

Tuloksena prosessikuvauksesta tulisi saada lehtiprosessista prosessikuvaus, tarkempi tilakaavio opinnäytetyön osasta sekä projektisuunnitelma.

2.2. Määrittelyt

Määrittelyosa on ehkä projektin tärkein ja aikaavievin osio. Lehteen tulevien artikkelien tiedot ja toiminnot määritellään siten, että niistä on helppo lähteä toteuttamaan työkalua.

Tuloksena tästä osiosta saadaan UML-mallit sekä sovelluksen arkkitehtuuria osoittava kaavio.

2.3. Toteutus ja testaus

Ennen toteutusvaihetta on opinnäytetyön tekijän opiskeltava XML-teknologioita.

Toteutus tehdään testivetoisena, eli testiluokkia tehdään ennen ohjelman pääluokkia. Tällöin luodaan ensin tavoitteet ja niiden avulla luodaan haluttu ohjelma.

Lehtityökalusta toteutetaan pieni demo, mutta testaukseen ei käytetä tämän opinnäytetyön puitteissa juurikaan aikaa. Toteutukseen käytetään Javaa sekä XML-teknologiaa. Tuloksena saatavat koodit kommentoidaan ja dokumentoidaan.

2.4. Projektin lopputulos

Projektin lopputuloksena on opinnäytetyö, jonka liitteenä ovat projektisuunnitelma, projektin päiväkirja sekä tilannekatsaukset. Päiväkirja on excel-dokumentti, johon on viikottain kirjattu tehtävät, joita on tehty sekä paljonko tehtävä on vielä kesken. Tilannekatsauksia on noin kahden viikon välein ja näistä tehdään muistio, joka sisältää tiedot aikaansaannoksista sekä aikataulussa olemisesta.

Kirjallisen opinnäytetyön lisäksi projektin lopputuloksena on pieni demo lehtityökalusta.

3. Projektissa käytettävät ohjelmistot ja laitteistot

Prosessikuvaukseen sekä tilakaavioihin käytetään ARIS Express –ohjelmaa, joka on ilmainen ja helppokäyttöinen mallinnusohjelma prosessinhallintaan.

StarUML-ohjelmaa käytetään UML-mallinnukseen.

Dokumentit kirjoitetaan joko Microsoft Wordillä tai Excelillä, tai Googlen omissa dokumenttiohjelmissa.

Kaikki ohjelmat on asennettu opiskelijan henkilökohtaiseen kannettavaan tietokoneeseen ja tätä kannettavaa käytetään koko projektin ajan pääasiallisena tietokoneena.

4. Projektin osapuolet

4.1. Opinnäytetyön tekijä

Crista Hiilesrinne
crizuh@gmail.com
puh. 045-6508575

4.2. Opinnäytetyön ohjaajat

Olli Niemi, Yoso Oy
olli.niemi@yoso.fi
puh. 040-7768972

Jari-Pekka Paalassalo, Turku AMK
jari-pekka.paalassalo@turkuamk.fi
puh. 050-5985847

5. Projektin tiedonvälitys

Projektin tiedonvälitys ohjaajien kanssa tapahtuu lähinnä Gmailin chatin kautta sekä sähköpostitse. Myös muutamia henkilökohtaisia tapaamisia järjestetään projektin aikana. Projektin aikana pidetään tilannekatsaus kahden viikon välein ja tarvittaessa useammin.

Projektin aikana täytetään Google-dokumenteissa sijaitsevaa aikataulu-exceliä viikoittain. Exceliin lisätään tehtävät, joita on viikon aikana tehty sekä paljonko tehtävä on vielä kesken ja mahdolliset ongelmat.

Dokumentointi tapahtuu Microsoft Wordillä ja dokumentit tallennetaan Google dokumentteihin, jossa ne on jaossa projektin osapuolien kanssa. Dokumentit tehdään suomen kielellä.

Dokumentteja, joita projektin aikana tehdään, ovat projektisuunnitelma, tilannekatsausten muistio sekä itse kirjallinen opinnäytetyö ja lisäksi erilaisia kaavioita prosessista.

6. Projektin aikataulu

Projekti alkoi tammi-helmikuun vaihteessa 2011 ja jatkuu heinäkuun lopulle.

Projektin tekemiselle ei ole asetettu tarkkoja tunteja, jolloin projektia tulisi tehdä, mutta kaikenkaikkiaan projektia tehdään viikossa noin 4 henkilötyöpäivää. Projektin kesto on opinnäytetyön kestoinen, joka tarkoittaa noin 60 henkilötyöpäivää.

Tilannekatsaukset

Tilannekatsauksien osapuolina ovat opinnäytetyöntekijä Crista Hiilesrinne sekä toimeksiantaja Olli Niemi, ellei toisin mainita.

11.02.2011

- Prosessikaaviota ja tilakaaviota artikkelin lisäämiseksi tehty sekä tuupparilistan tilakaaviota aloitettu
- Seuraavaksi pitäisi saada tuupparilistan tilakaavio valmiiksi (mahdolliset muutokset myös muihin kaavioihin)
- Projektisuunnitelmaa pitäisi täydentää mm.
    - Lopputulokset (dokut)
    - Mitä totetutetaan
    - Tehtäviä, jotta tulokset saavutetaan
- Sekä aikataulun päivitys (mitä tehty ja paljon käytetty aikaa)

25.02.2011

- Projektisuunnitelma on nyt tehtynä ja aikataulut päivitetty
- Seuraavaksi pitäisi aloittaa XML ja XSD opiskelua ja pikkuhiljaa pitää olla selvillä, mitä tehdään

11.03.2011

- oppari ei edennyt (kipeänä sekä muiden kurssien viimeisiä töiden palautuksia)
- jatkona edelleen XML opiskelua sekä tutustuminen AppFuseen.

25.03.2011 (henk.koht. miitti Turussa)

- AppFuse on hieman tullut tutuksi, mutta ei pintaa syvemmältä (ei vielä koodausta)
- Miitissä käytiin hieman läpi AppFusen toimintaa
- Keskusteltiin opparissa tehtävän työkalun ominaisuuksista
    - ihan yksinkertainen demo, 2-4 sivua
    - teksit ja mainokset staattisia

- Seuraavaksi kunnolla AppFusen kimppuun

08.04.2011

- AppFusen Person-luokka tullut tutuksi
- Ongelmia jonkun verran ajoissa, ei mene läpi, mutta selvittelemällä virheitä oppii paljon siitä, mitä erilaiset konffaustiedostot pitää sisällään.
- Oppari jatkuu AppFusen opettelulla

15.04.2011

- AppFusen ja XML:n opetteluun on mennyt paljon enemän aikaa kuin suunniteltu, joten keskusteltiin valmistumisen siirtämistä elokuuhun, jotta AppFusen ehtii sisäistämään ja tekemään kunnollisen demon, eikä tarvitse laittaa keskeneräistä demoa oppariin.
- Jatkuu edelleen AppFusen opettelulla, lisäksi projektisuunnitelman päivittäminen

29.04.2011

- AppFusen kanssa ollut edelleen ongelmia, ajot eivät mene ensimmäisellä kerralla läpi, vaan virheitä ilmenee (kirjoitusvirheitä ja ajatusmokia lähinnä).
- Projektisuunnitelmaa ei ole vielä päivitetty, joten se pitää tehdä.

13.05.2011

- Kirjallinen oppari aloitettu ja teoriaa kirjoitettu hieman ja päiväkirjaa päivitetty
- Tekeminen jatkuu AppFusen kanssa sekä XML:ää opetellessa

27.05.2011 (henk. koht. miitti Turussa)

- Käytiin yhdessä läpi AppFusen luomista. Ei kuitenkaan saatu vieläkään toimimaan kunnolla, joten sen parissa työskentely jatkuu.

10.06.2011

- AppFuse edelleenkin ongelmana. Sovittiin palaveri AppFusen osaajan kanssa ensi tiistaiksi. Nyt ei ole enää kovin paljon aikaa tapella pelkästään AppFusen kanssa.

14.06.2011 (palaveri Jakub Danekin kanssa)

- Jakub antoi muutamia vinkkejä AppFusen toiminnasta ja niitä lähden seuraavaksi testaamaan.

21.06.2011

- Jakubin ohjeiden mukaan AppFuse saatiin jo lähes toimimaan ja itse työkalua pääsee ohjelmoimaan pian. Lisäksi XML:ää luettu eteenpäin.

28.06-01.07.2011 (viikko Espoon toimistolla)

- AppFuse saatiin toimimaan ja ohjelmointi lähti hyvin liikkeelle. Sovellus osaa nyt lukea XML-muodossa olevan tiedoston, mutta ei vielä lisää sitä tietokantaan.
- Seuraavaksi lisää ohjelmoimista. Ohjelman pitäisi lisätä artikkeli tietokantaan ja lisäksi pitää ohjelmoida lehden sivut, jotka käyttäjä näkee

19.07.2011

- Oppari on oikein hyvässä mallissa. Ohjelmointi on melkein valmis ja kirjallista opparia hieman kirjoitettu lisää.
- Seuraavaksi ohjelmointi valmiiksi, dokumentti lehden toiminnasta print screenien avulla ja kirjallisen opparin kimppuun.

28.07.2011

- Ohjelmointi valmiina ja eka versio kirjallisesta opparista valmiina ja Ollin kommenttien perusteella parantelen sitä lisää. Täytyy lisätä mm. sovelluksen arkkitehtuurista.
- Seuraavaksi viimeistelyt kirjalliseen oppariin ja opparin liitteet kuntoon
- 1.8. pitää kirjallinen oppari palauttaa kouluun tarkastettavaksi ja samalla viikolla myös tapaaminen Espoon toimistolla.