Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Markus Santi, Jere Meriluoto

Yrityksen web-pohjainen CV:iden hallintajärjestelmä

Insinöörityö 3.12.2008

Ohjaaja: tekninen johtaja Atro Tammi
Ohjaava opettaja: yliopettaja Jarkko Vuori

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä	Markus Santi, Jere Meriluoto
Otsikko	Yrityksen web-pohjainen CV:iden hallintajärjestelmä
Sivumäärä	139 sivua
Aika	3.12.2008
Koulutusohjelma	tietotekniikka
Tutkinto	Insinööri (AMK)
Ohjaaja	tekninen johtaja Atro Tammi
Ohjaava opettaja	yliopettaja Jarkko Vuori

Ansioluetteloiden hallinnointi ja ylläpito laajassa, resursseja vuokraavassa organisaatiossa on työlästä sekä hankalaa. Ylläpitoa hankaloittavat erityisesti ansioluetteloita sisältävien tiedostojen määrät sekä formaattierot. Ansioluetteloiden yhtenäisen rakenteen ja ulkonäön puuttuminen voi heikentää organisaation uskottavuutta markkinoilla, aiheuttaen tappioita.

Tässä työssä toteutettiin yrityksen ansioluetteloiden keskitetty hallintajärjestelmä. Erityisesti keskityttiin ansioluetteloiden hallintajärjestelmän suunnitteluun ja toteutukseen, siinä käytettyjen teknologioiden tutkimiseen ja vertailemiseen sekä itse hallintajärjestelmän käyttöön. Ansioluetteloiden hallintajärjestelmä mahdollistaa organisaatiossa ansioluetteloiden tehokkaan hallinnoinnin sekä ylläpidon lisäksi ulkonäöllisesti sekä formaattisesti yhtenevien ansioluetteloiden tuottamisen. Ansioluetteloiden hallinnointia helpotettiin keskittämällä kaikki ansioluettelot yhteen paikkaan, lisäämällä käyttöliittymään erilaisia hakutoimintoja, joiden avulla pystyttiin selaamaan ansioluetteloita tehokkaasti sekä liittämällä käyttöliittymään organisaation työntekijöitä kuvastava resurssien hallinta -työkalu. Hallintajärjestelmän web-pohjaisen käyttöliittymän ansiosta ansioluetteloita voidaan hallita sekä päivittää mistä, milloin ja millä käyttöjärjestelmällä tahansa.

Hallintajärjestelmän toteutuksessa käytetyt teknologiat ovat yleisiä integraatio-sovelluksien suosimia teknologioita, kuten Java-ohjelmointikieli, XML- ja XSL-merkintäkielet, SQL-tietokannat sekä dynaamisten web-sivujen rakentamiseen soveltuva JSP-teknologia. Hallintajärjestelmässä sulautuvat yhteen Javalla rakennettujen tiedonhallintatyökalujen tehokkuus, JSP-sivuilla rakennetun käyttöliittymän dynaamisuus sekä XML-, XSL- ja FOP-teknologioilla tuotettavien ansioluetteloiden tyylikkyys sekä yhdenmuotoisuus. Hallintajärjestelmä sisältää peruskäytön mahdollistavien toimintojen lisäksi sisäänkirjautumisen, erilaiset näkymät käyttäjille sekä ylläpitäjille, tietoturvatoimenpiteitä, lokitusta, virheidenkäsittelyä sekä erilaisia työkaluja ansioluetteloiden hallintaa varten.

Hakusanat	Java, Stripes, JSP, Actionbean, HTML, CSS, XML, XSLT

Helsinki Metropolia University of Applied Sciences **Abstract**

Author	Markus Santi, Jere Meriluoto
Title	A web-based CV management system of a company
Number of Pages	139 pages
Date	3 December 2008
Degree Programme	Tietotekniikka
Degree	Bachelor of Engineering
Instructor	technical supervisor Atro Tammi
Supervisor	principal lecturer Jarkko Vuori

Management of CVs in a large organization that hires its resources is laborious and difficult. Upkeep is made particularly difficult by the amount of files and the differences in format. The lack of a consistent structure and appearance can diminish the organization's credibility on the market, causing losses.

In this program a centralized management system for CVs was created. The focus was on the planning and implementation of the management system, the comparison of possible technologies and the use of the system itself. The CV management system gives the possibility to an organization to effectively manage, upkeep and create CVs that are visually and formally convergent. The management of CVs was made easier by centralizing all the CVs into one location, including a resource manager tool and adding different search features to the user interface, which make the browsing of CVs easy. The management system's web interface makes the management and upkeep of CVs possible from anywhere and from any operating system.

The technologies used to make the management system are common technologies for creating integration applications, such as the Java programming language, XML and XSL markup languages, SQL database and JSP technology for creating dynamic web pages. In the management system the effectiveness of Java-built data processing, a dynamic user interface done by means of of JSP pages and the ability to create both sharp and unformalized CVs using XML, XSL and FOP technologies are fused together. In addition to the basic operations, the management system includes login, different views for different users, data security aspects, logging, error handling and different tools for managing CVs.

Keywords	Java, Stripes, JSP, Actionbean, HTML, CSS, XML, XSLT

Lyhenteet, käsitteet ja määritelmät

ActionBean	JSP-sivujen tapahtumankäsittelyolio
C++	oliopohjainen ohjelmointikieli
C#	oliopohjainen ohjelmointikieli
CSS	Cascading Style Sheet, HTML-sivujen tyylitiedosto
CV	curriculum Vitae, ansioluettelo
DAO	Data Access Object, datakontrolliyksikkö
Eclipse	sovelluskehitin
FOP	Formatting Objects Processor, muotoiluteknologia
GlassFish	sovelluspalvelin
ID	tunnistekoodi
Java	oliopohjainen ohjelmointikieli
JDBC	Java Database Connectivity, Javan tietokantatyökalut
JDOM	Java Document Object Model, Javan XML-työkalu
JSP	Java Server Pages, Javan palvelinsivu
HTML	Hypertext Markup Language, web-sivujen merkintäkieli
Log4J	Java-pohjainen kirjaamisjärjestelmä
MySQL	SQL-tietokannan hallintajärjestelmä
PDF	Portable Document Format, yleinen dokumenttistandardi
Servlet	dynaamisten sivujen tuottamiseen tarkoitettu rajapinta
SOA	Service Oriented Architecture, palvelintaso
SQL	Structured Query Language, tietokantakieli
Stripes	JSP-sivujen tuottamiseen tarkoitettu kehittyneempi teknologia
Struts	Stripesia edeltänyt JSP-sivujen rakentamiskehys

TO	Transfer Object, siirtoyksikkö
VPN	Virtual Private Network, virtuaalinen yksityisverkko
VO	Value Object, arvoyksikkö
XML	Extensible Markup Language, yleinen tiedon merkintäkieli
XSLT	Extensible Stylesheet Language Transformations, XML-merkintäkielen muotoilukieli

Sisällys

1 Johdanto

1.1 Iriba Oy:n esittely

Iriba Oy toteuttaa järjestelmäintegraatio- sekä -kehitysprojekteja asiakkailleen. Yrityksellä ei itsellään ole tuotantoa, vaan liiketoiminta perustuu muiden yritysten projektien tuottamiseen sekä tietoalan ammattilaisten vuokraamiseen. Iriba Oy sai alkunsa vuonna 2003, jolloin kaksi IT-alan ammattilaista, Atro Tammi sekä Lars Andtfolk, perustivat yrityksen palattuaan Suomeen Australiasta. Yritystä perustettaessa liiketoimintaideana oli keskittyä tietojärjestelmien integroimiseen ja kehittämiseen. Yrityksessä on nykyään töissä 12 henkilöä.

Molemmat tämän insinöörityön tekijät, Meriluoto ja Santi, palkattiin Iriba Oy:lle sovelluskehittäjiksi vuoden 2008 elokuussa. Yleisiin integraatioteknologioihin tutustuttaminen haluttiin tiivistää yhden ohjelman suunnitteluun ja toteutukseen. Tämä ohjelma oli ansioluetteloiden hallintajärjestelmä.

1.2 Ansioluetteloiden käsittelyongelma

Iriba lähettää työntekijöitään työhaastatteluun asiakkailleen, jotka tarvitsevat ulkoisia resursseja projekteihinsa. Työntekijällä kuuluu tällöin olla mukanaan ansioluettelo, jossa hänen tekniset taitonsa ovat listattuna. Ansioluettelosta tulee käydä ilmi teknisten taitojen nimet, työntekijän osaamistaso näissä taidoissa, projekti- ja urahistoria, koulutus sekä kielitaito.

Ennen jokainen työntekijä kirjoitti oman ansioluettelonsa johonkin tekstitiedostoon ja lähetti tämän ansioluetteloiden ylläpitäjälle (toimitusjohtaja), joka säilytti kaikkien työntekijöiden ansioluetteloita työkoneellaan. Asiakkaat hakevat yleensä jonkin tietyn teknologian osaajia, jolloin olemassa olevista ansioluetteloista tulee löytää ne henkilöt, jotka sopivat kuvaukseen parhaiten. Lisäksi tuli varmistua siitä, että työntekijä ei ole

toisessa projektissa halutulla työskentelyjaksolla. Ohjelmaan haluttiin liittää näin myös resursseja hallitseva komponentti, joka seuraisi työntekijöiden projektien kestoja ja ilmoittaisi ansioluettelon päivitystarpeesta.

Ansioluetteloiden tutkiminen tietyn taidon löytämiseksi on hidasta, sillä jokainen ansioluettelo sijaitsi eri tiedostossa sekä ansioluettelot erosivat ulkonäöllisesti toisistaan; taidot saattoivat olla eri paikassa kuin toiset eivätkä esimerkiksi aakkosjärjestyksessä.

Tästä toimintamallista haluttiin päästä eroon ja tilalle haluttiin yhdenmuotoinen ansioluetteloiden hallintajärjestelmä, joka mahdollistaisi ansioluetteloiden hallinnan, ylläpidon sekä tulostuksen yhdessä ohjelmassa.

1.3 Ansioluetteloiden hallintajärjestelmän määritelmä

Tehtävänä oli suunnitella ja toteuttaa ohjelma työntekijöiden ansioluetteloiden ylläpitämiseen ja tulostamiseen. Ohjelma toimisi yrityksen palvelimella, ja sitä käytetäisiin graafisen käyttöliittymän avulla. Ansioluettelot tallennettaisiin tietokantaan, josta niitä voitaisiin hakea ja päivittää.

Ohjelma jaettiin eri rajapintoihin ja määrityksiin. Ensinnäkin ohjelma tarvitsisi tehokkaan ja tarpeeksi yksinkertaisen tietokannan, johon työntekijöiden ansioluetteloon tulevat tiedot tallennettaisiin. Ohjelmalla tuli olla tähän tarkoitukseen tehokas tietokannan käsittely-yksikkö, joka kommunikoisi tietokannan kanssa tuottamalla tietokannan ymmärtämiä käskyjä sekä purkamalla tietokannalta saadut tulosjoukot ohjelman ymmärtämään muotoon.

Ohjelman käyttöliittymän tuli mahdollistaa tehokas ja nopea ansioluetteloiden päivitys sekä hallinnointi. Käyttöliittymästä haluttiin myös järjestelmäriippumaton, jotta ohjelmaa pystyttäisiin käyttämään mistä päin maailmaa ja millä käyttöjärjestelmällä tahansa. Käyttöliittymässä tuli olla sisäänkirjautuminen, joka mahdollistaa käyttäjien ja ylläpitäjien eri näkymät sekä alkeellisen tietoturvan. Käyttöliittymän tuli myös olla jatkokehityksen kannalta helposti muokattava.

Ansioluetteloiden ulkoasua haluttiin yhtenäistää samanlaiseksi, jotta esimerkiksi kahden eri ansioluettelon vertailu helpottuisi. Työntekijän tiedot haettaisiin tietokannasta ja muokattaisiin ansioluetteloon sopiviksi. Lopulta ansioluettelo tulostettaisiin yleiseen tiedostoformaattiin, jotta ansioluettelo pystyttäisiin tallentamaan kovalevylle tai tulostamaan paperille sellaisenaan.

Ohjelmaan haluttiin myös lisätä erillinen viestitysosio, johon käyttäjät voivat laittaa vikailmoituksia ja toiveita, sekä resurssienseurantatyökalu, joka mahdollistaa työntekijöiden projektien keston seurannan.

1.4 Työnjako

Ohjelmassa on kaksi selvästi eroteltavaa osuutta: tiedonkanta- sekä käyttöliittymäpuoli. Molemmat insinöörityön tekijät ovat tehneet ohjelman jokaista osa-aluetta. Meriluoto aloitti projektin hieman Santia aikaisemmin, joten hänen aluejakoonsa kuuluvat tietokantapuoli sekä ansioluetteloiden generointi. Santin aluejakona on puolestaan ollut käyttöliittymään liittyvät asiat, ohjelman toiminnallisuuden testaaminen sekä ohjelmasta löytyvien virheiden korjaus.

Työ on pääosin tehty pariohjelmointina eli samalla työasemalla. Koska työn etenemisvaiheet olivat selviä molemmilletekijöille, ei ohjelman kirjoittaminen tuottanut erimielisyyksiä. Pariohjelmointi toimii siten, että toinen rakentaa ohjelmaa toisen tarkkaillessa vierestä. Mahdollisia ongelmia kohdattaessa voidaan keskustella sopivista lähestymis- sekä käsittelytavoista. Kirjoitus- ja ajatusvirheet saadaan myös näin nopeasti karsittua ohjelmasta pois, vaikka kirjoittaja ei niitä itse huomaisikaan. Ohjelmaa on myös rakennettu eri työasemilla ja molempien tekijöiden aikaansaannokset on yhdistetty. Insinöörityön yhteenvedossa on kirjoitettu lyhyt vertailu näiden kahden työskentelytavan esille tuomista kokemuksista ja ajatuksista.

Ohjelmaa tehtäessä ei käytetty mitään varsinaista versionhallintaohjelmistoa, vaan ohjelmasta oli koko ajan tasan yksi versio, jota rakennettiin ja parannettiin tarpeen mukaan. Ohjelman rakentamiseen käytettäviä teknologioita saatiin ehdotuksina kokeneemmilta työntekijöiltä ja tarkan taustatutkimuksen perusteella vaihtoehdot supistettiin muutamaan teknologiaan, joista kokeneempien työntekijöiden kanssa valittiin parhaat. Muuten ohjelman toteuttamistavat olivat melko vapaat.

2 Projektissa käytetyt teknologiat

Tässä luvussa käsitellään ohjelman toteutuksessa käytetyt teknologiat. Useimmat teknologiat ovat hyvin yleisiä ohjelmistoprojekteissa ja koska ohjelma oli oppimisprojekti, haluttiin saada mahdollisimman monta eri teknologiaa sisällytettyä ohjelmaan.

2.1 Java EE

Java Enterprise Edition (Java EE) on oliopohjaisen Java-ohjelmointikielen kehitysalusta, jolla on paljon laajemmat kirjastot kuin tavallisella Java Standard Edition alustalla (Java SE). Java EE on kehitetty helpottamaan palvelupohjaisten ohjelmien (Service Oriented Architecture, SOA) sekä web-pohjaisten ohjelmien kehitystä. Java EE:n tärkeimpiä kirjastoja ovat Java-tietokantatyökalut (Java Database Connectivity Tools, JDBC), metodietäkutsut (Remote Method Invocation, RMI) sekä komponenttimäärittelyt (Java Bean, Servlet, JSP). Java EE tarjoaa tehokkaan sekä vakaan ohjelmointialustan useampikerroksisien palveluohjelmien suunnitteluun ja toteuttamiseen. [1; 2; 3.]

Koska vanhoilla proseduaalisilla ohjelmointikielillä ohjelman toteuttaminen olisi ollut työlästä ja koska haluttiin web-pohjainen käyttöliittymä, tuli valita tehokas ja monen eri teknologian kanssa toimiva oliopohjainen ohjelmointikieli. Oliopohjaisia ohjelmointikieliä on useita, kuten Java, C++ [4.] ja C# [5.]

Valittaessa käytettävää kieltä vaihtoehdoiksi jäivät Java ja C#. C++-kieli poistettiin vaihtoehtolistalta sen ollessa vain vanhempi versio C#:sta.

Verrattaessa Javaa ja C#:a huomataan, että ne ovat hyvin samantapaisia. C#-kielessä on otettu paljon mallia Javasta, minkä huomaa heti kieliä käyttäessä. Koska Java on vanhempi ohjelmointikieli, on siihen liitettävät työkalut ja selvät ohjelmointimenetelmät Javan puolesta puhuvia asioita. C#:n puolesta puhuu se, että sitä tehtäessä on mietitty mitkä asiat vanhemmissa kielissä tulisi korjata. Valmiskirjastot ovat omaa luokkaansa, ja luonnollisesti kielien yhteensopivuus Windows-käyttöjärjestelmän kanssa on erittäin hyvä. Molemmissa ohjelmointikielissä syntaksi on hyvin samantapaista, eli kumpaa tahansa ohjelmointikieltä käyttänyt pystyy kirjoittamaan toisella ilman suurempia ongelmia.

Ohjelmointikieltä valittaessa tietyt asiat painoivat puntarin Javan puolelle. C# on nykyään hyvin suosittu ohjelmointikieli pääteohjelmien teossa suurten valmiskirjastojen ja Windows-käyttöjärjestelmä yhteensopivuuden ansiosta. Hallintajärjestelmä on puolestaan palvelinpuolen sovellus, ja tähän Javassa on panostettu JavaEE-versiossa. Valmiin ja tarkan toimintamallin lisäksi päätökseen suuren vaikutuksen teki se, että molemmat tämän insinöörityön kirjoittaneet ovat ohjelmoineet Javalla selvästi enemmän. Viimeisimpänä ja suurimpana kriteerinä Javan valinnalle oli kuitenkin se, että yrityksemme pääohjelmointikieleksi on valittu Java. Näin ollen oli luonnollista ja suotavaa, että pidämme yhtä yrityksen virallisen linjan kanssa.

2.2 Javan suunnittelumallit

Oliopohjaisen ohjelmoinnin suunnittelumallit tarjoavat ohjelman suunnittelijoille ja kehittäjille tehokkaita yleisratkaisuja, jotka abstrahoivat ohjelman toimintaa sekä yksinkertaistavat sen rakennetta. Suunnittelumallit edesauttavat myös kehittäjien välistä kommunikaatiota tiivistämällä suuria kokonaisuuksia yksinkertaisiin ratkaisuihin. [6.]

Seuraavissa luvuissa käydään läpi vain muutamia suunnittelumalleja, joita ohjelmassa on käytetty. Ohjelmassa käytettyjä suunnittelumalleja ovat datakontrolliyksikkö (DAO), siirtoyksikkö (TO), arvoyksikkö (VO) sekä tehdas (Factory).

2.2.1 Datakontrolliyksikkö

Datakontrolliyksikön (Data Access Object, DAO) tehtävä on abstrahoida ohjelman sekä tietokannan välistä liikennettä. Datakontrolliyksikkö tarjoaa ohjelmalle yksinkertaisia metodeja lisätä, poistaa, päivittää ja hakea dataa tietokannasta. Tietokantakäskyt on kirjoitettu suoraan datakontrolliyksikön sisälle, johon kutsuvalta ohjelmalta saatu data pilkotaan ja sijoitetaan. Tämän jälkeen datakontrolliyksikkö ottaa yhteyden tietokantaan, suorittaa käskyn, tekee mahdollisia toimenpiteitä (käsittelee vastausdataa), sulkee yhteyden ja lähettää vastauksen kutsuvalle ohjelmalle. [6; 7.]

Datakontrolliyksiköiden rakenteen määrittelee datakontrolliyksiköiden rajapinta, jossa on määritetty neljä metodia, joita jokaisen datakontrolliyksikön tulee noudattaa. Metodit ovat samoja kuin tietokannalle tehtävät datan hallintakäskyt eli valinta, poisto, päivitys ja haku. Datakontrolliyksiköiden rajapinta mahdollistaa usean erilaisen tietokannan liittämisen ohjelmaan, sillä itse rajapinta ei ota kantaan tietokannan tyyppiin tai sinne lähetettäviin käskyihin. Tässä ohjelmassa on käytetty pelkästään SQL-tietokantaa, mutta jatkokehityksen vuoksi tietokantarajapinnan tulee olla uudelleenkäytettävä ja helposti muokattava.

2.2.2 Siirtoyksikkö

Siirtoyksikkö (Transfer Object, TO) on yksinkertainen Java-luokka, joka esittää tietokantataulun alkiota. Siirtoyksikön tehtävänä on hakea informaatiota palvelinpuolelta kokonaisuutena, mikä estää informaation paloina hakemisen aiheuttaman hitauden ja palvelinpuolen resurssien turhan käytön. Siirtoyksiköstä voidaan hakea, asettaa ja päivittää informaatiota, joka taas mahdollisesti lähetetään eteenpäin asiakas- tai palvelinpuolelle. [6; 8.]

Ohjelmassa siirtoyksikköä käytetään kokonaisten olioiden siirtämisessä palvelinpuolen ja tietokannan välillä. Työntekijän nimeä haettaessa haetaan kokonainen työntekijätaulun alkio, sillä jos käyttäjä tarvitseekin myöhemmin lisää tietoa kyseisestä työntekijästä, on olio jo palvelinpuolella, mikä estää uudet turhat tietokantahaut. Tietokannan alkiolla ja siirtoyksiköllä on luonnollisesti samat informaatiokentät, ja niihin voidaan laittaa samoja arvoja. Näin voidaan helposti käyttää datakontrolli-yksiköitä sekä siirtoyksiköitä tietokannan hallinnassa. Esimerkiksi kun halutaan lisätä tietokantatauluun uusi alkio (uusi työntekijä tai jokin muu alkio), täytetään vain siirtoyksikköön oikeat tiedot ja lähetetään se datakontrolliyksikölle, joka tekee tarvittavan tietokannan päivityksen. Siirtoyksikössä ei yleensä voi olla dataa muokkaavia metodeja.

Ohjelmasta haluttiin kuitenkin tehdä monikielinen, ja tämän takia siirtoyksikköä tuli hieman muokata ja soveltaa. Siirtoyksikön tuli sisältää kokoelmia eri kielien teksteille. Ohjelman siirtoyksiköt suunniteltiin tietokannan ja palvelinpuolen väliseen kommunikaatioon ja arvoyksiköt, jotka esitellään seuraavassa luvussa, palvelin- ja asiakaspuoleiseen kommunikaatioon. Tämä paransi informaation kontrollia sekä mahdollisti eri osa-alueiden paremman erottamisen toisistaan.

Siirtoyksiköitä luodaan käyttöliittymä- sekä tietokantarajapinnassa. Datakontrolli-yksiköt luovat uusia siirtoyksiköitä vastaanottaessaan dataa tietokannasta. Siirtoyksiköt siirtyvät tietokanta- sekä käyttöliittymärajapinnan välillä. Käyttöliittymästä tullut tiedonhaku- tai tiedonpäivityskäsky aiheuttaa tapahtumaketjun, jossa ensin tarvittavat tiedot sisältävä arvoyksikkö muutetaan siirtoyksiköksi, joka sen jälkeen siirretään tietokantarajapinnalle. Arvoyksikkö voi sisältää esimerkiksi käyttäjän täyttämien syöttökenttien tietoja. Kun siirtoyksiköstä on saatu tarvittava informaatio esimerkiksi arvoyksikköön tai ansioluettelon tulostamiseen tarvittuihin metodeihin, siirtoyksikkö poistetaan.

2.2.3 Arvoyksikkö

Arvoyksikkö (Value Object, VO) on siirtoyksikön (TO) edeltäjä. Ohjelmassa haluttiin kuitenkin ottaa arvoyksiköt mukaan informaation kontrolliin. Kuten edellisessä kappaleessa todettiin, siirtoyksiköt oli tarkoitettu sisältämään usean eri kielen tekstejä ja arvoyksikkö vain yhden. Tietokannan ja palvelinpuolen välissä kulkevista siirtoyksiköistä parsitaan tiedot arvoyksikköön, joka kuljetetaan asiakaspuolelle näytettäväksi. [6; 8.]

Arvoyksikkö rakentuu kentistä, jotka matkivat sitä kuvaavan tietokannan alkion kenttiä, sillä poikkeuksella, että arvoyksikköön voidaan laittaa esimerkiksi eri formaattisia päivämääriä eikä arvoyksikkö sisällä minkään sortin kokoelmia. Arvoyksiköt luodaan käyttöliittymästä tulleen käskyn avulla käyttöliittymärajapinnassa, jossa siitä poimitaan arvot tietokantarajapinnalle menevään siirtoyksikköön. Kun arvoyksikön kaikki tiedot on siirretty siirtoyksikölle, arvoyksikkö poistetaan.

2.2.4 Tehdas

Tehtaan (Factory) tarkoituksena on toimia datakontrolliyksiköiden luojana, tietokannan valitsijana sekä tietokantayhteyksien luojana. Tehdas mahdollistaa ohjelmalle yhdessä datakontrolliyksiköiden kanssa usean eri tietokannan valinnan sekä käytön. Tehtaan sisälle on valmiiksi määritelty tietokantayhteydessä käytettävät ajurit sekä tietokannan osoitteet, käyttäjätunnukset ja salasanat. Datakontrolliyksiköt rakennetaan yleensä pyytämään tehtaalta tietokantaan muodostamaa yhteyttä, jota käytetään datan lähettämisessä ja vastaanottamisessa. [6; 7.]

Ohjelman tehtaan tärkeimpänä tehtävänä on toimia staattisena (”aina läsnä”) luokkana eri datakontrolliyksiköiden luomisessa, eri tietokannoille on luotava kyseisen tietokannan käskyjä generoivia datakontrolliyksiköitä.

2.3 MySQL

MySQL on relaatiotietokantojen hallintajärjestelmä, jota ohjataan SQL-lauseilla (Structured Query Language). MySQL-tietokantoja suositaan niiden nopeuden, luotettavuuden ja useiden ohjelmointikielien kanssa kommunikoimisen mahdollistavan rajapinnan takia. [10; 11.]

Ohjelmiin yleensä rakennetaan oma SQL-rajapinta, joka keskustelee tietokannan kanssa SQL-lauseilla. Internetistä on mahdollista saada ajurit Java-sovellusten ja MySQL-tietokantojen käyttämiseen. MySQL-tietokanta on yleinen sovelluksissa käytettävä tietokanta, joten sen sisällyttäminen projektiin tuntui luontevalta ratkaisulta.

2.4 JSP

Javan palvelinsivut (Java Server Pages, JSP) ovat tärkeä komponentti selainkäyttöliittymien tekemisessä. JSP-sivut hyödyntävät palvelinsovelma (servlet) rajapintaa käyttöliittymän ja tietoa käsittelevän ohjelman välistä kommunikointia varten. JSP-sivujen toiminta perustuu Java-koodin upottamiseen HTML-koodin sekaan. Sivua lukiessa ohjelman komponentti, joka vastaa palvelinsovelman ja palvelimen välisestä viestinnästä, muodostaa sivusta palvelinsovelman, jonka avulla tiedonsiirto käyttöliittymän ja palvelimen välillä mahdollistetaan. [12.]

JSP-sivujen tekeminen on hankalaa, sillä kahden eri ohjelmointikielen sisällyttäminen samaan sivuun tekee ohjelmoinnista sekavaa. Tästä syystä on kehitetty JSP-sivujen valmistamista helpottavia työkaluja. Ohjelmassa käytetään työkalua nimeltä Stripes Framework, josta kerrotaan lisää luvussa 2.7.

2.5 HTML

HTML-kieli (HyperText Markup Language) on yleisesti käytetty merkintäkieli, jolla kuvataan web-sivuja. HTML-kieli ei ole ohjelmointikieli, vaikka niin monesti luullaankin. HTML-kieli käyttää merkintätageja tekstitiedon, kuvien, linkkien ja interaktiivisten lomakkeiden näyttämiseen web-sivuilla. [13; 14.]

Koska web-pohjaisen käyttöliittymän rakentamiseen liittyy aina jollakin tavalla HTML-merkintäkieli, tuli se ottaa mukaan projektissa käytettäviin teknologioihin. Useat eri web-ohjelmointikielet, kuten JSP-sivuissa esiintyvä Java-koodi, pystyvät kyllä tuottamaan dynaamista tietoa sivuille, mutta ne eivät pysty yksin esittämään tietojaan web-sivuilla. Tähän tarkoitukseen tarvitaan HTML-merkintätageja.

2.6 CSS

CSS-kieli (Cascading Style Sheets) on yksinkertainen mekanismi, jolla lisätään muotoilutyylejä web-sivuille. CSS-kielen avulla voidaan määrätä esimerkiksi fontin värejä, painoja sekä sivujen komponenttien tyylejä. CSS-koodi sijaitsee joko HTML-koodiin upotettuna tai se linkataan erillisestä tiedostosta HTML-sivuille. [15; 16.]

Koska HTML-sivujen tyylittely on työlästä ja itseään toistavaa HTML-koodin avulla, otettiin käyttöön CSS-kieli. CSS-kielellä pystyttiin määrittämään koko sivun kattava fontti, taustakuvat, sivun komponenttien asettelu sekä värimaailma.

2.7 Stripes Framework

JSP-sivujen tuoma etu dynaamisuudessa toi käyttäjälle myös ongelmia. Sivujen luominen oli hankalaa, ja näytettävän datan hallinnoiminen monimutkaisissa tilanteissa vaati paljon työtä. Tätä ongelmaa ovat monet eri tahot yrittäneet ratkaista luomalla sivujen rakentamista helpottavia kehyksiä (Framework). Helpoin tapa demonstroida ongelmaa on kuitenkin esimerkin kautta. Kuvassa 1 on esitetty esimerkki JSP-sivusta, johon on HTML-koodin sekaan upotettu Java-koodia <% %> -tagien väliin. Sivu siis suorittaa Java-koodia, josta se saa muuttujia sivulla näytettäviin kohtiin. Näinkin yksinkertaisesta esimerkistä selviää, miksi monimutkaisempien sivujen ja ohjelmapätkien yhdistäminen tuottaa ongelmia; koodi menee yksinkertaisesti liian sekavaksi.

```
<body bgcolor="white">
<font size=5 color="red">
<%! String[] fruits; %>
<jsp:useBean id="foo" scope="page" class="checkbox.CheckTest" />

<jsp:setProperty name="foo" property="fruit" param="fruit" />
<hr>
The checked fruits (got using request) are: <br>
<%
        fruits = request.getParameterValues("fruit");
%>
<ul>
<%
    if (fruits != null) {
          for (int i = 0; i < fruits.length; i++) {
%>
<li>
<%
              out.println (util.HTMLFilter.filter(fruits[i]));
          }
        } else out.println ("none selected");
%>
</ul>
<br>
<hr>
```

Kuva 1. Esimerkki JSP-sivuista, jossa Java-koodia on upotettu HTML-koodin sekaan

Stripes Framework on kehittynyt dynaamisten HTML-sivujen rakentamista varten luotu työkalu, joka mahdollistaa JSP-sivujen kehittyneemmän rakentamisen [17]. Stripes ratkaisee edellä mainitun ongelman tagikirjastolla. Itse JSP-sivuun ei tule mitään Java-koodia, vaan pelkästään HTML-koodia. Koodin kohdat, jotka sisältävät dynaamisia ominaisuuksia, on muutettu Stripes-tageiksi. Näin sivuilla näytettävää dynaamista tietoa saadaan muokattua kirjoittamalla HTML-koodin sekaan Stripes-tageja. Sivun toiminnallisuus ja tapahtumat hoidetaan ActionBean-tapahtumankäsittelyluokissa. Erottamalla nämä kaksi osaa toisistaan saadaan selkeyttä isojen ja monimutkaisten JSP-sivujen tekemiseen. Kuvassa 2 on esimerkki ohjelman JSP-sivusta, jossa käytetään Stripes-tageja ja viitataan sivusta vastaavaan ActionBeaniin.

```
<stripes:form action="/com/iriba/cvgenerator/actionbean/Category.action">

<stripes:hidden name="languages.selectedLanguage" value="${actionBean.languages.selectedLanguage}"/>

<table>
    <tr><td><h3>${actionBean.ski.text}</h3></td></tr>
    <tr><td><h4>Employees with this skill:</h4></td></tr>
    <tr><td>
    <c:forEach items="${actionBean.skillEmpList.skiEmpList}" var="el">
            <tr><td><h4>${el.level}</h4></td></tr>
            <c:forEach items="${el.emps}" var="empvo">
                <tr><td> - ${empvo.name}</td></tr>
            </c:forEach>
            <br>
    </c:forEach>
    </td></tr>
    <tr><td>
    <stripes:link href="/com/iriba/cvgenerator/actionbean/Category.action" event="searchSki">
        <IMG src="${pageContext.request.contextPath}/images/buttons/back_b.gif">
        <stripes:param name="search" value="${actionBean.search}"/>
    </stripes:link>
```

Kuva 2. JSP-sivu, joka on rakennettu Stripes Frameworkilla.

Kuten jo edella todettiin, Stripes Frameworkin tapaisia työkaluja on tehty useita. Suurin osa ratkaisuista on hyvin alkeellisia, eivätkä ne juurikaan vähennä tarvittavan työn määrää. Yleisimmin käytetty työkalu on nimeltään Struts [18]. Stripes on uudempi työkalu, ja se kehitettiin juurikin Strutsissa ilmenneiden ongelmakohtien, kuten sivun interaktiivisuuden ylläpidon, takia. Vaikka itse työkalujen käyttämisessä on paljon yhtäläisyyksiä, Stripesin suurin etu on suurempi abstraktion määrä. Sivujen toimintaa ylläpitävistä asioista ei Stripesissa tarvitse huolehtia.

2.8 XML

XML (eXtensible Markup Language) on W3C:n (World Wide Web Consortium) luoma järjestelmien väliseen tiedonsiirtoon perustuva merkintäkieli. XML mahdollistaa tiedon välittämisen järjestelmästä toiseen yksinkertaisen merkintänsä ansiosta. Tasohierarkiaan perustuva XML asettaa dataa tiettyyn järjestykseen, jota voidaan esittää sen jälkeen helposti esimerkiksi web-sivuilla. XML on myös hyvin vahva järjestelmien väliseen kommunikaation soveltuva sanomienlähetysteknologia. [19; 20.]

Ohjelmassa tuli löytää helppo tapa siirtää työntekijän tiedot ansioluettelon generoivalle osalle. Java-kielellä helpoin tapa luoda PDF-tyyppinen tiedosto oli ensin luoda XML-muotoinen tiedosto ja käyttää FOP-teknologiaa tiedoston muodostamiseen (FOP-teknologiasta lisää myöhemmissä luvuissa).

2.9 XSLT

XSLT (Extensible Stylesheet Language Transformations) on W3C:n luoma standardi XML-merkintäkieli, joka toimii XML-tiedostoille ikään kuin tyylitiedostona (vertaa HTML ja CSS). XSLT ei ota kantaa tiedon esitystapaan (web-sivu, CVS-tiedosto, PDF) eikä alkuperäistä XML-tietoa muuteta, vaan se esitetään eri tavalla tyyliteltynä. [19; 21.]

Ohjelmassa XSLT:tä on käytetty tyyliteltyjen tulostettavien PDF-tiedostojen luomiseen FOP-teknologian avulla. Ohjelmassa tuli olla mahdollisuus tulostaa ansioluettelo valmiiksi määritellylle pohjalle, ja XSLT-teknologia on tehty juuri XML-tyyppistä tietoa varten. Ohjelman vastaanottama XML-sanoma puretaan XSLT-tyylitiedostossa määriteltyihin kohtiin. Ohjelmassa voi olla useita XSLT-tyylitiedostoja, mikä mahdollistaa erilaiset ansioluetteloiden pohjat.

2.10 JDOM

JDOM (Java Document Object Model) on teknologia, joka mahdollistaa XML-tiedostojen luomisen, muokkaamisen ja lukemisen helposti Javan kautta. JDOM:ssa luodaan XML-dokumentti, asetetaan sinne tietotasoja sekä itse dataa. Lopputuloksena on valmis XML-dokumentti. [22.]

Ohjelmassa JDOMia on käytetty työntekijän tietojen muokkaamisessa XML -tiedostoksi siirtoyksiköstä. Vaihtoehtoisena XML-tiedostojen muodostamiskeinona olisi ollut XML-tiedoston kirjoittaminen merkkijonona Java-luokan sisällä, mutta tämä ajatus hylättiin tehottomuutensa vuoksi.

2.11 Apache FOP

Apache FOP on Java-sovellus, joka kääntää XSL-FO (XSL Formatting Objects) -tiedostoja PDF- tai muuhun tulostettavaan muotoon. FOP-teknologiaa käytetään Java-sovelluksissa, joihin annetaan XML-tiedosto, jossa tieto on, sekä XSL-tyylipohja. Ohjelma muodostaa valmiin XSL-FO-tiedoston, joka voidaan suoraan kääntää PDF-muotoiseksi tiedostoksi. [23]

2.12 Log4J

Log4J-teknologia on Java-pohjainen kirjaamistyökalu. Lokin kirjoittaminen on osa hyvin kirjoitettua ohjelmaa, joka mahdollistaa isojenkin ohjelmien tapahtumien ja syntyneiden poikkeuksien seuraamisen sekä virheiden etsimisen. Kirjattavat kohdat on aina kerrottava itse lokituskomponentille. Kirjaamiskomponentti ei itse seuraa ohjelman tapahtumia vaan sitä kutsutaan aina tarvittaessa. Kirjattavat asiat voidaan valitaan määräämällä lokitustaso. Korkein taso on virheet, jolloin lokittaminen tapahtuu poikkeuksen sattumisen yhteydessä. Keskimmäinen taso on tapahtumat, jolloin lokittaminen on täysin ohjelman käyttäjien päätettävissä. Kirjattavaksi lisätään ohjelmassa ne tapahtumat, joiden käytöstä halutaan merkintä lokiin.

Alin taso on virheiden etsimistaso, jonka avulla ohjelmoija voi seurata ohjelman suoritusta ja löytää mahdolliset virheet. Kirjaamistason määrääminen sisältää aina korkeammat tasot, joten esimerkiksi lokitustason määrääminen virheiden etsimiseksi lisää lokiin kaikki ylemmät tasot. Log4J:n valitseminen lokitustyökaluksi juontaa juurensa yrityksen sovelluskehitysvalintoihin. Lisäksi Log4J on yleisimmin käytetty lokitustyökalu Java-ohjelmia tehdessä. [24.]

Ohjelmassa lokitus olisi voitu tehdä itsekirjoitetulla Java-luokalla, mutta valmiiden työkalujen käyttö on tehokkaampaa ja varmempaa. Ohjelmassa lokitustaso on asettettu keskimmäiselle tasolle. Lokitukseen kuuluvat tietokantatransaktioiden tapahtumat. DAO-luokassa lähetettävän tietokantakäskyn lopputulos lokitetaan. Poikkeuksen sattuessa lokitetaan virhe ja onnistunut transaktio lokittaa pelkän tapahtuman. Esimerkki tästä on nähdään kuvassa 3**.**

```
try{
    conn = RDBDAOFactory.createConnection();
    stmt = conn.createStatement();
    stmt.executeUpdate(statement);

    logger.info("All EmployeeSkill with id: " + id + " removed");

    conn.close();
    return true;
}
catch(SQLException e){
    logger.error("All EmployeeSkill with id: " + id + " remove failed!");
    return false;
}
```

Kuva 3. Tietokantatransaktion lokitus.

Kuvassa 3 on esitetty lokituksen käyttöä. Lokitus-oliota kutsutaan info-, error-, tai debug-käskyllä riippuen siitä, minkälainen tapahtuma halutaan lokittaa. Lisäksi annetaan lokitettava viesti. Näiden tietojen perusteella lokittaja kirjoittaa tapahtuman haluttuun tekstitiedostoon. Lokitukset kirjautuvat tiedostoon riveittäin. Lokitusmerkinnässä näkyy aina aikaleima (milloin lokitus tapahtui), lokituksen taso, mikä luokka on käskenyt lokituksen ja itse lokitusviesti.

3 Ansioluetteloiden hallintajärjestelmän rakenne

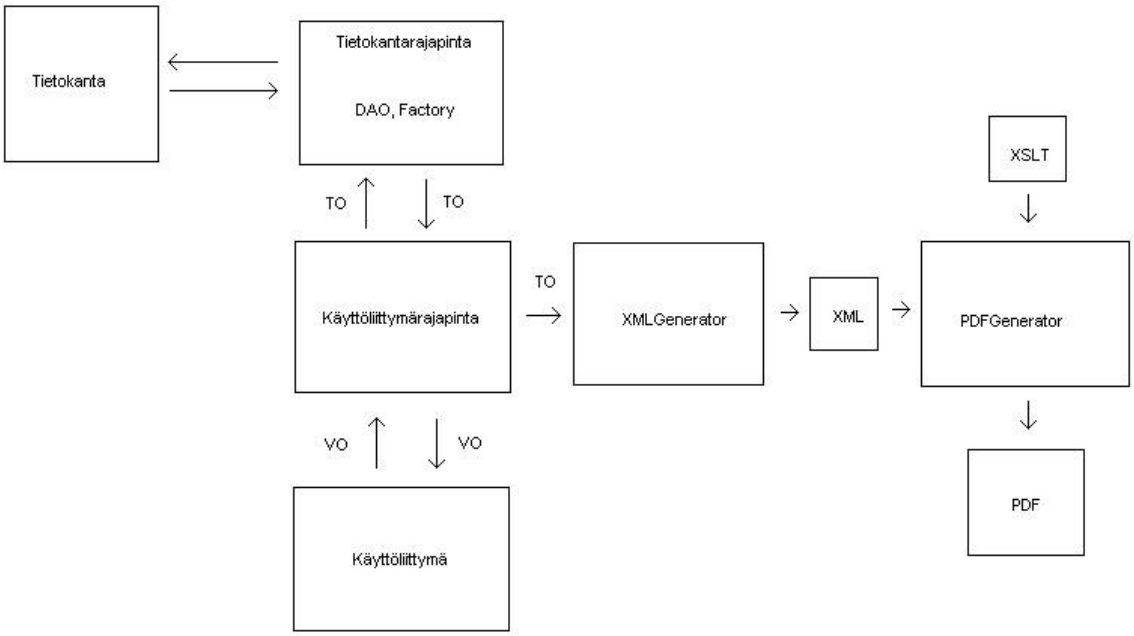
Tässä luvussa käydään lyhyesti läpi hallintajärjestelmän yleinen toiminta ja tietokannan, rajapintojen sekä käyttöliittymän periaatteet ja rakenteet.

3.1 Yleinen toiminta

Ansioluetteloiden hallintajärjestelmä koostuu web-pohjaisesta käyttöliittymästä, Javalla rakennetuista käyttöliittymä- sekä tietokantarajapinnoista sekä niiden Javalla rakennetuista luokista, jotka toimivat datan siirtoyksikköinä sekä datan käsittelijöinä. JSP-sivut toimivat käyttöliittymärajapintana ohjelman ja käyttäjän välillä.

Käyttäjän tehdessä tiedonhaku- tai tiedonpäivityspyynnön käyttöliittymässä käyttöliittymärajapinta kommunikoi tietokantarajapinnan kanssa ja tietokantarajapinta huolehtii datan haku- ja päivitystoiminnoista. Tietokannan ja tietokantarajapinnan välillä kulkeva informaatio koostuu SQL-käskyistä sekä tulosjoukoista. Tietokanta- sekä käyttöliittymärajapinnan välissä kulkeva informaatio on tietokannan tauluja kuvaavassa muodossa eli siirtoyksikkömuodossa (TO). Siirtoyksiköistä voidaan näin hakea haluttuja tietoja eri aikoina. Käyttöliittymärajapinta muuttaa siirtoyksiköt käyttöliittymässä näytettäviksi arvoyksiköiksi, joista haetaan halutut tiedot käyttäjälle näytettäviksi.

Käyttäjän tehdessä CV:n tulostuspyynnön käyttöliittymärajapinta lähettää käsiteltävän siirtoyksikön XML-generaattorille, joka muodostaa XML-tiedoston siirtoyksiköstä. XML-tiedosto lähetetään edelleen PDF-generaattorille, joka hakee tulostettavan PDF:n tyylipohjan XSLT-tiedostosta sekä yhdistää XML-tiedostosta ansioluettelossa näytettävät tiedot tyylipohjalla valmiiksi määriteltyihin kohtiin, muodostaa PDF-tiedoston ja avaa käyttäjälle tulostettavan ansioluettelon PDF-muodossa.



Kuva 4. Ohjelman yleinen toiminta

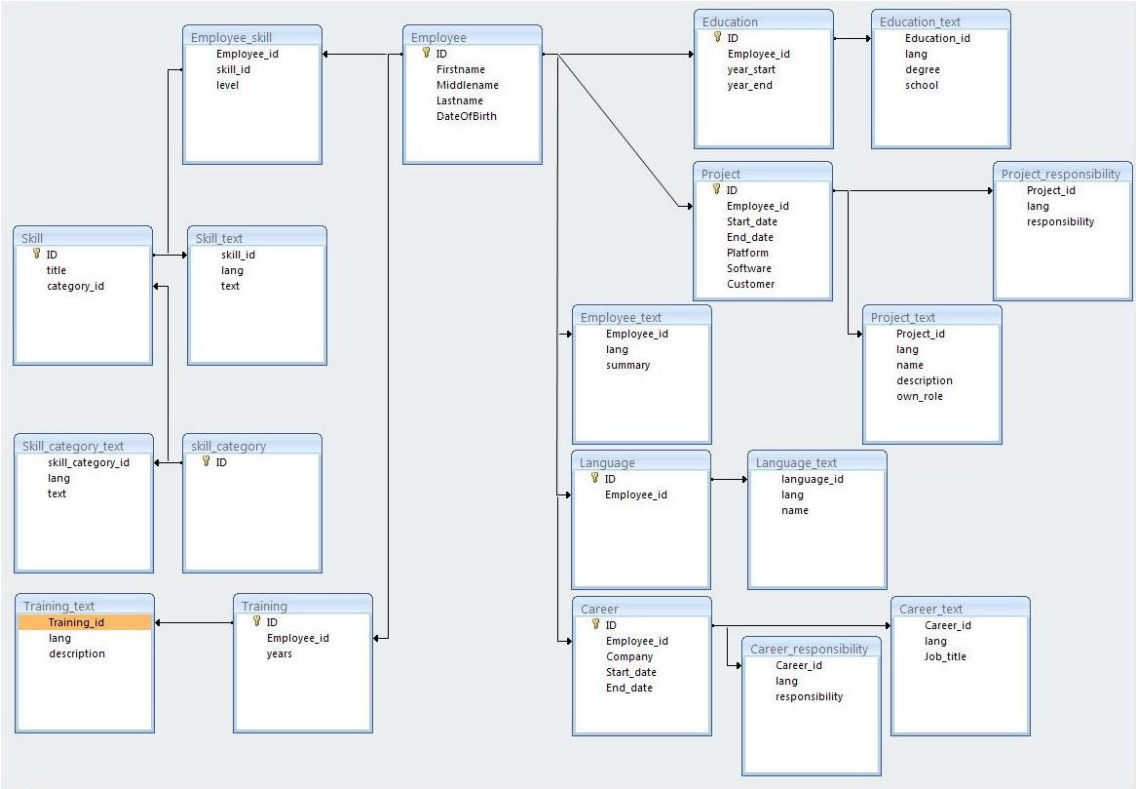
Kuvassa 4 on esitetty ohjelman yleisen toiminnan rakennekaavio. Jokainen laatikko kuvastaa toiminnallista osaa ja laatikoiden välillä kulkevat nuolet merkitsevät informaation kulkua. Nuolien viereen on merkitty informaation kapselointitapa.

3.2 Tietokanta

Tietokannan rakenne

Ohjelmassa haluttiin saada jokaiselle ansioluettelon kohdalle oma tietokantataulu, jotta jokaista kohtaa pystyttäisiin käsittelemään omana kokonaisuutenaan. Esimerkiksi työntekijöillä on projekteja, jonka tietoihin kuuluvat asiakas, projektin nimi, vastuualueet, käytetyt teknologiat jne. Projektia poistettaessa, päivitettäessä tai lisättäessä pystyttiin nyt siis käsittelemään kaikkia siihen liittyviä tietoja yksinkertaisesti käsittelemällä yhtä projektitaulua.

Tietokanta koostuu 21 taulusta, jotka on linkitetty toisiinsa toisten taulujen tunnistekoodien (ID) avulla. Tietokanta on jaettu lnikitettäviin ja yksittäisiin tauluihin, joista linkitettäviä ovat kaikki Employee-päätaulusta riippuvaiset taulut sekä kielitaulut. Yksittäisiä tauluja ovat Employee-taulu sekä yrityksen työntekijöiden taidot sisältävä Skill-taulu.



Kuva 5. Tietokannan rakenne.

Kuvassa 5 on esitetty CV:n hallintaan liittyvien taulujen rakenne sekä taulujen keskinäiset suhteet. Taulun kentän vieressä oleva avain kuvastaa kentän olevan taulun pääavain. Taulusta toiseen osoittavat nuolet osoittavat kenttien suhteita, eli minkä taulun attribuutin tulee olla sama molemmissa tauluissa. Tunnistekoodit (ID) jakautuvat taulun omaan tunnistekoodiin ja työntekijän tunnistekoodiin (Employee_id).

Employee-päätaulu

Employee-päätaulun tarkoituksena on toimia eräänlaisena avaintauluna, joka mahdollistaa työntekijän käsittelyn tietokannassa yhtenä oliona. Employee-päätaulu sisältää tärkeimmät tiedot työntekijästä, kuten nimet, syntymäajan sekä tietokannan generoiman työntekijän tunnistekoodin (ID). Tunnistekoodin avulla kaikki työntekijään liittyvät taulut, voidaan linkittää oikeaan työntekijään. Tietokanta pitää huolen, ettei kahdella työntekijällä voi olla samaa tunnistekoodia.

Ilman Employee-päätaulua työntekijästä ei saa olla mitään muutakaan tietoa tietokannassa. Tällöin, jos työntekijän ansioluettelo halutaan poistaa, tulee Employee-päätaulun mukana poistaa kaikki muut taulut, joihin työntekijän tunnistekoodi viittaa.

Muut taulut

Muut taulut liittyvät työntekijän ansioluettelon tietoihin, lukuun ottamatta yrityksessä yleisesti olevia taitoja ylläpitävä Skill-taulu ja siihen liittyvät kategoriataulut. Ansioluettelon tietoihin liittyvillä tauluilla on kieliriippuvaisia alatauluja, jotka viitataan päätauluihin näille annettujen tunnistetietojen perusteella (päätaululle generoitu oma tunnistekoodi sekä Employee_id, eli työntekijän tunnistekoodi). Päätaulujen datankäsittely on toteutettu tietokantarajapinnassa datakontrolliyksiköillä.

Alataulut

Alataulut sisältävät kieliriippuvaisia tekstejä. Esimerkiksi työntekijän kirjoittama tiivistelmä itsestään voi olla kirjoitettuna usealle eri kielelle, ja tällöin jokaista Employee-taulua kohden on niin monta Employee_text-alataulua kuin kieliä on määritelty. Alataulut kulkevat päätaulujensa elinkaaren mukana, eli alataulut luodaan, poistetaan, päivitetään ja valitaan päätaulujensa kanssa samaan aikaan. Siksi alataulujen datankäsittely on sijoitettu päätaulujen datankäsittelyn sisälle.

Tietokantarajapinta

Tietokantarajapinnan tehtävänä on abstrahoida tietokannan ja käyttöliittymän välillä kulkevan datan käsittelyä. Rajapinnan tehtävänä on pitää tietokantayhteyttä toiminnassa, valita oikea tietokanta, generoida tietokannalle lähetettäviä SQL-lauseita, käsitellä tietokannasta saatua informaatiota ja lähettää käsiteltyä informaatioa takaisin kutsuvalle ohjelmalle.

Tietokantarajapinta koostuu Javalla rakennetuista, SQL-lauseita generoivista datakontrolliyksiköistä, dataa sisältävistä siirtoyksiköistä sekä tietokantayhteyksiä ja datakontrolliyksiköitä generoivasta staattisesta tehdas-luokasta.

Käyttöliittymärajapinta

Käyttöliittymärajapinnan tärkeimpänä tehtävänä on kommunikaatio käyttöliittymän, tietokantarajapinnan sekä PDF-tulostuksen kanssa. Käyttöliittymästä tehdyt pyynnöt kulkevat käyttöliittymärajapinnalle, joka päättelee, mihin pyyntö lähetetään eteenpäin. Käyttöliittymärajapinnan tehtävänä on myös tietokantarajapinnalta saatujen siirtoyksiköiden muokkaaminen arvoyksiköiksi eli käyttöliittymässä esitettävään muotoon sekä arvoyksiköiden muokkaaminen tietokantaan siirrettävään muotoon eli siirtoyksiköiksi.

Käyttöliittymä

Käyttöliittymän tehtävänä on luoda graafinen esitys ohjelmasta käyttäjälle ja mahdollistaa interaktiivisuus käyttäjän sekä ohjelmiston välillä. Käyttöliittymässä käyttäjän painaessa nappia ohjelman suoritus siirtyy käyttöliittymärajapinnalle, joka välittää käskyn eteenpäin määritellylle komponentille. Vastauksen palautettuaan käyttöliittymä jää odottamaan käyttäjän seuraavaa käskyä. Käyttöliittymä rakennettiin

selainpohjaiseksi ja siitä haluttiin myös miellyttävän näköinen, mutta ennen kaikkea tehokas, nopea ja vakaa.

Käyttöliittymässä käyttäjälle näkyvät sivut on rakennettu yhden valmiiseen pohjasivuun päälle, joka sisältää yrityksen logon, dynaamiselle datalle tarkoitetun keskiosion sekä alatunnistaan. Sivupohja liitetään muiden sivujen alkuun, jolloin säästytään sivujen uudelleenkirjoittamiselta. Jokaisella käyttöliittymän sivulla on eri tehtävänsä, jolloin sivun lataamisen alussa pohjasivun määräämälle keskiosalle ladataan haluttu JSP-sivu.

4 Ohjelman rakenne

4.1 Tietokantarajapinta

4.1.1 Yleinen toiminta

Tietokantarajapinnan tehtävänä on lähettää SQL-käskyjä tietokantaan ja muokata sieltä saatua dataa valmiisiin siirtoyksikköihin sopivaksi. Siirtoyksiköt välitetään tämän jälkeen käyttöliittymän kontrolleille, joissa siirtoyksiköt muutetaan arvoyksiköiksi ja välitetään eteenpäin käyttöliittymälle. Tietokantarajapinta on jaettu itse rajapintaan, jossa määritellään datakontrollien metodit, joiden avulla generoidaan SQL-lauseita tietokannalle sekä siirtoyksiköihin, joihin tietokannasta saatu ja tietokantaan menevä data on muokattu sopivaksi.

Tietokantarajapinnan suunnittelu ei rajoitu pelkästään SQL-tietokantoihin, vaan rajapintaa voidaan muokata lisäämällä uusia datakontrolleja eri tietokannoille, esimerkiksi Oracle- ja Cloudspace-tietokannoille. Itse tietokantarajapinta pysyy samana; käyttäjä voi syöttää dataa tietokantaan, lukea dataa tietyin kriteerein, poistaa dataa ja päivittää vanhoja tietoja.

4.1.2 Datan lisäys tietokantaan

Tietokantaan siirrettävä data sijoitetaan ensin siirtoyksikköön (TO), joka lähetetään tehdas-luokalta saatuun datakontrolliyksikköön (DAO). Tämän jälkeen siirtoyksikkö siirretään datakontrolliyksikölle insert-metodilla. Datakontrolliyksikkö saa muodostetun yhteyden tietokantaan tehdas-luokalta, luo alustavan SQL-käskyn (INSERT) normaalina merkkijonona ja lisää siirtoyksiköstä tiedot SQL-lauseeseen oikeisiin kohtiin. Lopulta datakontrolliyksikkö yrittää lähettää generoidun SQL-lauseen tietokannalle valvoen samalla mahdollisia virheitä (tietokantaan ei saatu yhteyttä, SQL-lause ei kelpaa) ja ilmoittaa onnistumisestaan käyttäjälle.

Datakontrolliyksiköiden rakenne muuttuu monimutkaisemmaksi, kun käsitellään yhtä päätaulua ja sen alatauluja. Tällöin jokaisen käskyn jälkeen tulee myös käsitellä alataulua. Kun halutaan lisätä dataa päätauluun sekä alatauluun (jonka tulee olla linkitettynä päätauluun tämän ID:n perusteella), on ensin lisättävä päätaulu tietokantaan, haettava tietokannan generoima tunnistekoodi päätaululle takaisin datakontrolliyksikölle ja lisättävä alataulu tietokantaan käyttäen tätä tunnistekoodia. Eli datakontrolli-yksiköissä, jotka käyttävät sekä päätaulua että alataulua, tulee olla kolme eri SQL-lausetta (päätaulun lisäys, ID:n haku sekä alataulun lisäys). Muut metodit rakennetaan hyvin samalla tavalla, esimerkiksi päätaulua poistettaessa on myös alataulut poistettava päätaulun tunnistekoodin perusteella.

4.1.3 Datan poisto tietokannasta

Kun tietokannasta halutaan poistettavan dataa, luodaan ensin kyseisen datan datakontrolliyksikkö, jolle ilmoitetaan poistettavan taulun tunnistekoodi. Tehtaalta saatu datakontrolliyksikkö käyttää tehtaan luomaa yhteyttä tietokantaan, muodostaa SQL-käskyn (DELETE) ja yrittää lähettää sen tietokantaan. Datakontrolli ilmoittaa tämän jälkeen, onnistuiko poisto. Päätaulua poistettaessa poistetaan myös kaikki siihen viittaavat alataulut.

Employee-päätaulua poistettaessa poistetaan myös kaikki kyseisen taulun tunnistekoodiin (ID) viittaavat taulut. Tämä tapahtuu rakentamalla ensin kaikkien taulujen datakontrolliyksiköt ja aktivoimalla näiden poistometodit, joissa taulut poistetaan työntekijän tunnistekoodin perusteella.

4.1.4 Datan päivitys tietokantaan

Vanhan datan päivityksessä haetaan yleensä ensin tietyin kriteerein haluttu data, muokataan sitä ja päivitetään tietokantaan. Tehdas-luokalta saatu datakontrolliyksikkö ottaa argumenttina päivitettävän siirtoyksikön, jonka ID:n perusteella pystytään päivittämään oikeaa taulua tietokannassa. Datakontrolliyksikkö käyttää tehtaalta saatua yhteyttä, muodostaa SQL-käskyn (UPDATE) ja yrittää päivittää tiedot tietokantaan. Tämän jälkeen datakontrolliyksikkö ilmoittaa onnistumisestaan kutsuvalle ohjelmalle.

4.1.5 Datan valinta tietokannasta

Datan valinta tietokannasta tietyin kriteerein on monimutkaisin metodi koko datakontrolliyksikössä. Ensin luodaan valittavan datan tyyppinen siirtoyksikkö, jonka kenttiä (esim. ID, nimi) muokataan halutunlaisiksi. Tämän jälkeen tehtaalta saatu datakontrolliyksikkö ottaa kyseisen siirtoyksikön ja lähtee generoimaan SQL-käskyä (SELECT).

Koska minkä tahansa kentän mukaan voidaan valita dataa, pitää SQL-lauseen muodostus suorittaa pätkissä. Ensin testataan, onko saadun siirtoyksikön (criteria) kenttä tyhjä (0 tai null, riippuen kentän tyypistä). Jos kenttä ei ole tyhjä, lisätään se hakukriteeriksi SQL-lauseeseen (kentännimi LIKE kentänarvo AND). Testejä jatketaan niin kauan, kunnes kaikki siirtoyksikön kentät on käyty läpi. Lopussa katsotaan, olivatko kaikki kentät tyhjiä. Tällöin generoitu SQL-lause on muotoa SELECT * FROM taulun_nimi, ja sillä valitaan kaikki taulut.

Jos yhdessäkin kentässä oli dataa, lisätään alkuperäiseen SQL-lauseeseen (SELECT * FROM taulun_nimi) ehtokäsky (WHERE), sijoitetaan kaikki kenttähaut peräkkäin SQL-lauseen perään (kentännimi LIKE kentänarvo AND kentännimi LIKE kentänarvo AND) ja lopussa poistetaan lauseesta neljä merkkiä, jotta saadaan viimeinen ”AND”-merkkijono pois. Tämän jälkeen SQL-lause lähetetään tietokannalle, ja datakontrolliyksikkö vastaanottaa tuloskokoelman (resultset), jossa on kaikki tietokannasta saatu data, joka vastasi hakukriteerejä. Datakontrolliyksikkö luo kokoelman siirtoyksiköistä, luo uuden siirtoyksikön yksi kerrallaan, täyttää sen datalla tuloskokoelmasta ja siirtää sen kokoelmaan. Datakontrolliyksikkö palauttaa kokoelman siirtoyksiköistä metodin lopussa.

4.2 Käyttöliittymä

Käyttöliittymä on siis selainkäyttöliittymä, joka tuli tehdä Stripes Frameworkia käyttäen. Käyttöliittymään kuuluu kahdeksan aluetta: JSP-sivut, ActionBean-tapahtumakäsittelijät, arvo-objektikirjasto, EmpManager-luokka, syöttökenttien tarkistus, kielen valinta, tietoturva sekä järjestelmäpoikkeukset.

4.2.1 JSP sivut

Näkyvin osa käyttöliittymästä ovat JSP-sivut. Stripes Frameworkin ansiosta JSP-sivuille ei tule Java-koodia lainkaan, vaan pelkästään stripes-tageja HTML-koodin sekaan. Itse sivun tapahtumat käsitellään ActionBean-luokissa. Tämän seurauksena sivuja tehdessä voi keskittyä vain sivun ulkoasuseikkoihin. JSP-sivut ovat tyyliteltyjä CSS-tiedoston avulla, ja käyttöliittymässä on käytetty pohjapiirrosta, jossa on logo, navigaationapit sekä kielen valinta. Sivuihin on myös lisätty graafisia komponentteja, jotka ovat Meriluodon käsialaa. Muuten sivuissa on lähinnä peruskomponentteja. Tietojen jaottelu on tehty tavallisilla HTML-tauluilla. Haetut tiedot tulostetaan tyyliteltyihin taulukoihin. Muut käyttöön liittyvät objektit (submit-napit, tekstikentät, tekstialueet ja alasvetovalikot) ovat Stripes-tageilla tehtyjä.

4.2.2 ActionBean

Toinen puoli käyttöliittymän interaktiivisuudesta ovat ActionBean-tapahtuman-käsittelyoliot. JSP-sivujen ollessa vapaita Java-koodista, on oltava luokkia, jotka hoitavat tapahtumankäsittelyn. JSP-sivu- ActionBean-pari toimii siten, että jokaisen sivun jokaiselle interaktiiviselle osalle määrätään ActionBean-luokka. Yksi luokka voi toki käsitellä useampaa toimintoa sivuissa. Sivulla oleva nappi tai linkki kutsuu siis ActionBean-luokan jotain metodia, joka hoitaa itse tapahtuman käsittelyn. ActionBean-luokat on jaettu tehtävien mukaan, sillä jokainen luokka hoitaa tietyn osa-alueen käyttöliittymästä. Jotta ActionBean-luokkien rakennetta saataisiin yksinkertaisemmaksi, on ne määrätty periytyväksi BaseActionBean-luokasta, jossa on kaikille sivuille yhteisiä muuttujia ja metodeita.

ActionBeanien käyttäminen sujuu seuraavasti: kun sivu ladataan, se hakee tietoa ActionBeanilta, jonka ActionBean hakee esimerkiksi tietokannasta. Kun sivulta siirrytään toiselle sivulle, tallennetaan sivun ladanneeseen ActionBeaniin kaikki parametrit (esimerkiksi sisäänkirjautuneen henkilön tunnus) sekä mahdollisten syöttökenttien tiedot. Seuraavan sivun lataa uusi ActionBean, jonka edellisen sivun tapahtuman aiheuttaja (nappi tai linkki) on määrännyt.

ActionBeanien välillä siirrettävät tiedot ovat joko parametreina JSP-sivuilla tai tapahtumaa käsittelevässä metodissa. Yleisesti tällä tyylillä ohjelmoiduissa käyttöliittymäsivuissa on kaksi hyvää etua. Ensinnäkin JSP-sivujen koodittomuus helpottaa käyttöliittymän hallittavuutta, sillä ulkoasu sekä tapahtumankäsittely on eroteltu toisistaan. Toiseksi palvelimella olevan ohjelman tarvitsemat palvelinsovelman käsittelyt hoituvat automaattisesti Stripes Frameworkin kautta.

4.2.3 Arvo-objektit

Arvo-objektit ovat olioita, joissa tietokantaan sijoitettu tieto kulkee EmpManager-luokan ja käyttöliittymän välillä. Arvo-objekteihin tallennetaan myös tietoa käyttöliittymässä, kun halutaan tallentaa tai tehdä muutoksia tietokannassa olevaan dataan. Arvo-objekteihin kuuluvat myös List-luokat, jotka hakevat taulukoihin tulostettavat tiedot listamuodossa. Listojen muodostus tapahtuu sekä List-olioissa että EmpManager-oliossa. Muita arvo-objekteihin kuuluvia luokkia ovat Util-luokat, joiden tehtävänä on suorittaa muutos siirto-objektien ja arvo-objektien välillä. Vaikka olennainen sisältö on luokilla sama, joutuu tietoa joissain tapauksissa muokkaamaan. Yleisesti arvo-objektiluokissa kuljetetaan kaikki tieto käyttöliittymän ja EmpManager-luokan välillä.

4.2.4 EmpManager-luokka

EmpManager-luokka on tärkein yksittäinen luokka koko ohjelmassa. Luokka toimii solmukohtana käyttöliittymän ja tietokantapuolen välillä ja hoitaa lisäksi suurimman osan käyttöliittymään tulostettavien taulukoiden tiedonkäsittelystä. Luokassa on siis metodit, jotka lähettävät siirtoyksiköitä tietokantapuolelle sekä metodeja, jotka lähettävät arvoyksikkö-listoja käyttöliittymälle. Arvo- ja siirtoyksiköt ovat muutettavissa toisesta toiseksi Util-luokkien avulla. EmpManager-luokassa on kaikki usean eri tiedon hakemiseen tarkoitetut metodit sekä tiedon järjestelyyn ja kriteerien asettamiseen tarkoiteut metodit.EmpManager-luokan tehtävänä on myös lähettää muodostetut arvoyksikkölistat käyttöliittymälle.

4.2.5 Syöttökenttien tarkistus

Tietokantaan talletettavien tietoalkioiden muodossa ja määrässä on rajoituksia. Täytettäessä jotakin tietoalkiota, esimerkiksi työntekijän koulutusta, on aina olemassa pakollisia kohtia (esimerkiksi koulun nimi). Lisäksi on varmistuttava, että muun muassa päivämäärät, numerot ja tekstit tallentuvat oikeassa muodossa tietokantaan. Näiden kohtien varmistamiseksi syöttökenttiin on lisätty tarkistus. Tarkistusvaihtoehtoja oli kaksi: merkki kerrallaan -tarkistus, jossa käyttäjän antamaa informaatiota syöttökenttiin tarkistettiin reaaliaikaisesti jokaisen vastaanotetun merkin jälkeen, sekä koko syöttökentän sisältämän informaation tarkistus vasta lähetysvaiheessa. Ohjelmassa päädyttiin käyttämään jälkimmäistä keinoa, koska kyseinen tapa oli järjestelmälle kevyempi sekä yksinkertaisempi.

Stripes Frameworkissa on valmiina kaksi tapaa tehdä tarkistuksia. Ensimmäinen ja yleisempi on lisätä vahvistussääntöjä ActionBean-luokkiin. Syöttökentästä tulevan tiedon tulee täyttää annetut ehdot, tai käyttäjälle ilmestyy virheilmoitus kyseisestä kentästä. Ehdot voivat olla esimerkiksi kentän pituudessa ja pakollisuudessa. Toinen ja varmempi tapa on Stripes-poikkeusten käyttäminen. Ohjelmassa tätä keinoa on käytetty päivämäärien käsittelyssä. Käyttöliittymien kentät ovat tekstikenttiä, ja niihin syötetyn tiedon on oltava oikeassa muodossa. Vääränlaisen tiedon sijoittaminen päivämäärämuuttujaan aiheuttaa poikkeuksen joko käyttöliittymässä tai viimeistään tietokantaan tallennettaessa. Näin käyttöliittymästä saadun lausekkeen sisällön sijoittaminen päivämääräolioon asetetaan try-catch-pariin ja syötettäessä vääränlaista tietoa heitetään Stripes-poikkeus. Poikkeus on tavallinen Java-poikkeus, mutta Stripes käsittelee sen automaattisesti lisäämällä virheilmoituksen käyttöliittymään.

4.2.6 Kielien valinta

Ohjelmaan oli lisättävä toiminto, jolla pystyy vaihtamaan käsiteltävää kieltä. Kyse ei ole käyttöliittymän kielestä vaan tulostettavien ansioluetteloiden kielestä. Oletuskieli ansioluetteloiden kirjoittamisessa on englanti. Jokaiselle tietokantaan talletettuun alkioon on löydyttävä toinen alkio suomen kielellä.

Käyttöliittymän sivupohjassa on kielenvalintakohta, josta muuttamalla kieltä kaikki haut, hakujen talletukset sekä PDF-tulostus muutetaan toiselle kielelle. Ohjelma siirtyy samalle sivulle, missä käyttäjä painoi nappia. Toiminnon toteuttamiseksi kielen vaihdolle on tehty oma ActionBean-luokka sekä tietokantaan kielikohtaiset taulut. Käytössä oleva kieli siirretään parametrina JSP-sivuja vaihdettaessa. Käyttöliittymässä on myös sivuja, joissa kielen vaihto ei vaikuta toimintaan. Näiden sivujen tapauksissa kielenvaihto aiheuttaa nollaustoimenpiteen, jossa kieli asetetaan englanniksi ja käyttäjä siirtyy pääsivulle.

4.2.7 Tietoturva

Ohjelman tietoturvaa on parannettu kahdella tavalla. Ensinnäkin ohjelmassa on sisäänkirjautuminen, jossa ohjelman ensimmäiseen versioon haluttiin kovakoodatut käyttäjätunnukset ainoastaan yrityksen työntekijöille. Koska verkkovälitteisessä käytössä on erilaisia tietoturvariskejä, on ohjelmaa käyttääkseen aluksi kirjauduttava yrityksen palvelimelle. Palvelimella on palomuuri, jonka läpäiseminen vaatii VPN-yhteyden (Virtual Private Network). Yhteyden saa ainoastaan VPN-client ohjelman avulla. Ohjelma luo salatun tunnelin käyttäjän ja palvelimen välille. [25.] Seurauksena palvelimen käyttö on turvassa mahdollisilta tietoturvariskeiltä. Samasta syystä johtuen sisäänkirjautumistoiminto on ohjelmassa suhteellisen alkeellinen. Asiasta kerrotaan tarkemmin luvussa 9.2.

Toiseksi sisäänkirjautumisen yhteyteen on lisätty tietoturvasuodatin. Selaimen osoitekenttään pystyy kirjoittamaan minkä tahansa käyttöliittymän sivun, jolloin olisi mahdollista tehdä muutoksia tietokantaan ilman lupaa. Tämä on estetty tietoturva-suodattimen avulla. Suodatin tarkistaa, onko käyttäjä kirjautunut sisään ja onko käyttäjällä tarvittavat oikeudet sivulle, jolle tämä yrittää päästä. Tarkistuksen toteuttaa Security Filter -olio, johon on listattu sallitut sivut ja tapahtumat. Aina listaan kuulumattomien sivujen tai näiden tapahtumankäsittelijän kutsuminen aiheuttaa tarkistuksen.

4.2.8 Järjestelmäpoikkeukset

Järjestelmäpoikkeukset ovat peruspoikkeuksia, jotka aiheutuu tilanteessa, jossa tietokantaan talletettaessa tapahtuu virhe. Seula ei kumminkaan ole täydellinen ja on mahdollista, että tietokannan talletukseen pääsee virheellinen arvo. Käyttöliittymää tämä ei kaada, mutta ilman järjestelmäpoikkeuksia ongelma ei ilmenisi mitenkään käyttäjälle.

Poikkeus aiheutuu aina DAO-tasolla, ja siirretään ylöspäin aina ActionBean-luokille asti, jossa poikkeukset käsitellään. Kaikki metodit, jotka käyttävät tietokantaa jollain tavoin sisältävät try-catch-yhdistelmän. Saadessaan poikkeuksen ohjelma siirtyy virhesivulle, joka on peruskäyttöliittymäsivu pohjapiirroksen ja ohjelmanappien kanssa. Sivulle ilmestyy ilmoitus siitä, mikä aiheutti poikkeuksen. Sivulta käyttäjä voi jatkaa ohjelman käyttöä normaalisti.

4.2.9 Resurssinhallinta

Resurssinhallintatyökalu on yksi ohjelmaan tehdyistä lisäyksistä ohjelman päätarkoituksen, eli ansioluetteloiden käsittelyn, lisäksi. Yrityksen toimitusjohtaja tulostaa ohjelman avulla ansioluetteloita mainostaessaan työntekijöitämme mahdollisiin asiakasyrityksiin. Luvattaessa henkilöitä näille yrityksille on tiedettävä työntekijän sen hetkinen projektitilanne. Ratkaisu ongelmaan on resurssinhallintayökalu, johon toimitusjohtaja voi kirjata ylös, missä kukin työntekijä on milläkin hetkellä töissä, työntekijästä laskutettava tuntihinta sekä aika, jolloin työntekijä vapautuu.

Resurssinhallinta luo näkymän työntekijöistä ja heidän projektitilanteestaan. Vaikka tämä on toimitusjohtajalle tehty työkalu, voivat kaikki yrityksen työntekijät käydä katsomassa yrityksen projektien tilanteita. Muille työntekijöille näkyvä näkymä tosin on suppeampi kuin toimitusjohtajalle. Työntekijöillä ei ole mahdollisuutta muuttaa tietoja, eivätkä he voi nähdä työntekijöistä laskutettavaa tuntihintaa.

5 Tulostettavan CV:n generointi

CV:n generointi lähtee käyntiin, kun käyttäjä painaa ”View CV” -nappia käyttöliittymässä. Tällöin käyttöliittymärajapinta lähettää kyseisen työntekijän tiedot XML-generaattorille, joka tuottaa työntekijästä XML-tiedoston. XML-tiedosto yhdistetään XSLT-tyylitiedoston kanssa ja muutetaan FOP-teknologialla valmiiksi PDF-muotoiseksi tiedostoksi.

Siirtoyksikön muuttaminen XML-muotoiseksi

XML-generaattori luo ensin valmiin XML-dokumentin, johon sen jälkeen upotetaan eri tietotasoja. Tietotasot auttavat tiedon hallinnassa sekä helpottavat datan lukemista. Ansioluettelon kieli otetaan tässä kohtaa huomioon. XML-generaattori saa parametrinä tulostettavan CV:n kielen ja muuttaa ansioluettelossa olevat otsikot sen mukaan. Siirtoyksiköstä siirretään tiedot XML-tietotasoille yksi kerrallaan ja lopulta saadaan valmis XML-dokumentti.

CV:n XSLT-tyylipohja

XSLT-tiedostossa on ennalta määriteltyjä paikkoja ansioluettelon tiedoille sekä tyylitelty näitä kohtia halutunlaisiksi, esimerkiksi työntekijän etu- ja sukunimi painetaan lihavoidulla tekstillä. XSLT-tiedostossa määritellään taulukoita, tasoja sekä tekstikenttiä, aivan kuten HTML- ja CSS-kielissä. XSLT mahdollistaa myös erinäisien ehtolauseiden käytön, jolla voidaan suorittaa erilaisia toimenpiteitä, jos dataa ei esimerkiksi löydykään XML-tiedostosta.

XSL:n sekä XML:n yhdistäminen tuottaa XSL Formatting Objectin (XSL-FO), joka näyttää samalta kuin alkuperäinen XSLT-tiedosto, mutta määrättyihin kohtiin on haettu XML-tiedostosta haluttu data. XSL-FO-tiedosto on valmis käsiteltäväksi, eli siitä voidaan esimerkiksi tulostaa PDF-muotoinen tiedosto ohjelman PDF-generaattorilla.

PDF-muotoisen tiedoston generointi

Ansioluettelon tulostus tapahtuu PDF-generaattorissa. PDF-generaattori ottaa parametrina sisäänsä työntekijän tiedot sisältävän XML-tiedoston ja hakee XSL-tiedoston valmiiksi määritellystä paikasta. XML- ja XSL-tiedostojen yhdistämisen jälkeen generaattori käyttää FOP-teknologiaa muodostaakseen PDF-muotoisen ansioluettelon. Valmiiksi generoitu PDF-muotoisen ansioluettelon kuva löytyy liitteestä 1.

6 Ohjelman käyttö

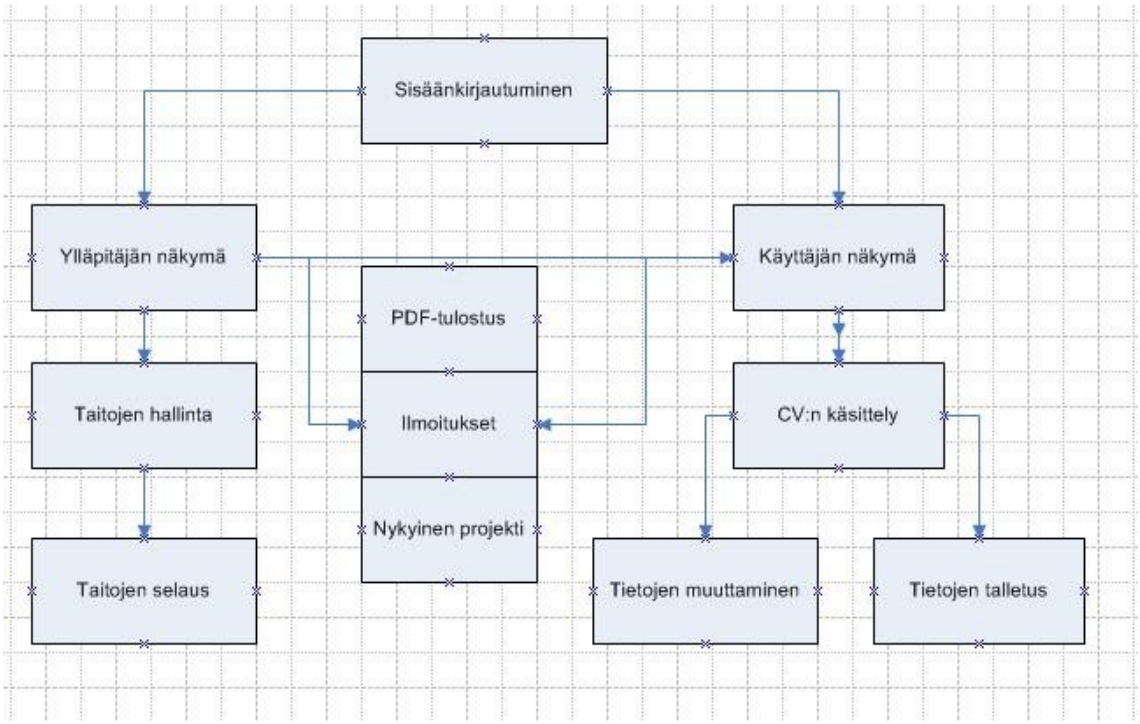
Tässä luvussa käydään läpi ohjelman käyttöä ja siihen liittyviä seikkoja. Ensimmäiseksi käyttöliittymää tehtäessä on päätettävä periaatteet käyttöliittymän suunnittelua varten. Valittavia asioita ovat muun muassa sivujen ulkoasu ja se, miten se vaikuttaa ohjelman käyttämiseen, värimaailma ja mitä vaikutuksia eri väreillä voi olla sekä kuvien vaikutus ohjelman asiallisuuteen ja käyttöön. Käyttöliittymää tehdessämme päätimme, että käyttöliittymän tulee noudattaa yleistä selkeyttä sekä helppoa käyttöä painottavaa selainsivuasetelmaa.

Käyttöliittymän sivuille lisättiin sivupohja navigointinappeineen, joista käyttäjä pystyy nopeasti siirtymään ohjelman osasta toiseen. Ansioluetteloiden muuttaminen on ohjelman pääalue eli alue, jota käytetään eniten. Tähän osaan teimme selkeät kuvalliset napit eri kohtien löytämiseksi. Lisäksi ansioluettelon eri kohtien tiedot näytetään aina taulukoidusti ja helpottavia komponentteja käyttäen (esimerkiksi alasvetovalikot).

Sivujen informaatiosisältö haluttiin pitää mahdollisimman vähäisenä, jotta muutosten ja lisäysten tekeminen sujuisi nopeasti ja helposti. Käyttöliittymän jokainen sivu sisältää mahdollisimman vähän tietoa, tieto esitetään mahdollisimman selkeällä tavalla ja koko ohjelmalle on yhteinen linja, jota kaikki sivut noudattavat. Värit ja kuvat tekevät suuren vaikutuksen käyttöliittymän yleisilmeeseen.

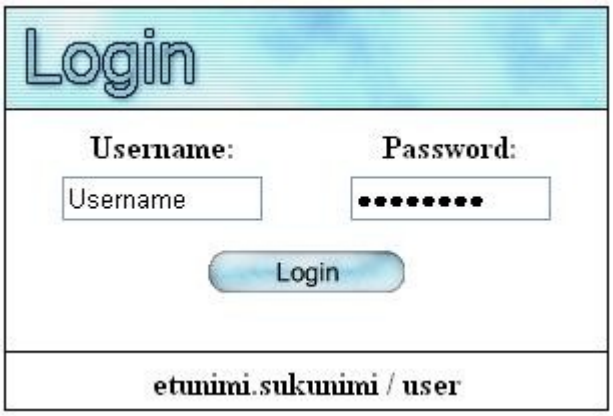
Graafisia virikkeitä on hyvä ohjelmassa olla, sillä se helpottaa ohjelman käyttöä ja antaa ohjelmasta laadukkaaman kuvan. Varjopuolena vääränlaiset tai liialliset graafiset virikkeet saattavat vähentää ohjelman uskottavuutta. Värimaailmaa sekä kuvia valitessamme päätimme tehdä mahdollisimman neutraalin ja rauhallisen ilmapiirin ohjelman käyttäjälle. Värimaailma on vaalea, mikä yleisesti antaa rauhoittavaa tunnetta aggressiivisuuden sijaan. Kuvista ja napeista haluttiin informatiivisia, selkeitä ja virikkeellisiä ilman, että ohjelman asiallisuus kärsii. Käyttöliittymän ulkoasua näytettiin ohjelman loppukäyttäjille, joiden mielipiteet vaikuttivat lopulliseen ulkoasuun.

Ansioluetteloissa on osia, jotka ovat vapaasti lisättäviä, kuten koulutus ja projektit. Joissakin asioissa haluttiin kumminkin yhtenäistää ihmisten ansioluetteloita, esimerkiksi työntekijöiden taidoissa ei haluttu erilaisuuksia kirjoitustyylin tai pienten seikkojen vuoksi. Tämä oli pääsyy kahteen näkymään. Pelkästään ylläpitäjän oikeuksilla kirjautuva pystyy muuttamaan tai lisäämään taitoja. Käyttäjät pystyvät lisäämään ylläpitäjän määräämistä taidoista omaan listaansa ne, jotka katsoo osaavansa. Toinen syy eri näkymille oli yrityksemme toimitusjohtajan tarvitsemat työkalut. Taitoselaaja (Skill Browser) sekä resurssienhallinta (Resource View) ovat taitojen ylläpitämisen (Skill Manager) lisäksi vain toimitusjohtajan käytettävissä. Ohjelman kahden eri näkymän navigointikaavio on esitetty kuvassa 6.



Kuva 6. Käyttöliittymän navigointikaavio

Ohjelman käyttäminen alkaa sisäänkirjautumisesta, johon käyttäjä kirjoittaa käyttäjätunnuksensa ja salasanansa. Login-nappia painettaessa ohjelmaan rakennettu sisäänkirjautumisolio tarkistaa käyttäjätunnuksen sekä salasanan. Sisäänkirjautumis-ikkuna on esitetty kuvassa 7.

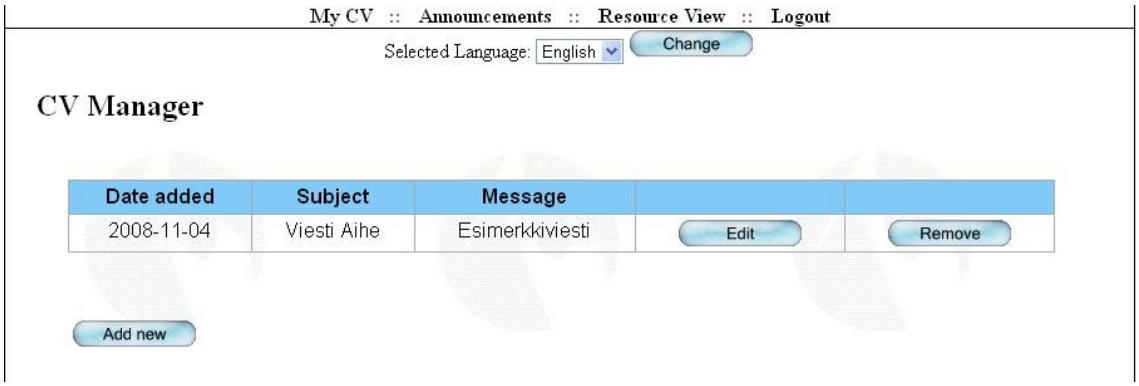


Kuva 7. Sisäänkirjautumisikkuna.

Käyttäjänimi on aina etunimi.sukunimi-muotoinen. Salasana on kovakoodattuna ohjelmaan. Sisäänkirjautumisen jälkeen ohjelma siirtyy eri sivuille, riippuen käyttäjän oikeuksista. Tavalliset käyttäjät näkevät vain oman ansioluettelonsa muokkaamiseen tarvittavat sivut. Ylläpitäjä taas siirtyy työntekijälistaukseen, ja hänellä on myös käytettävissään kaikki navigaatiopalkin napit, joista pääsee käyttämään lisäominaisuuksia, kuten taitoselainta. Ylläpitäjällä on myös oikeudet muokata kenen tahansa työntekijän CV:tä. Tämä tapahtuisi käytännössä vain, jos ansioluetteloita tulostettaessa huomattaisiin kirjoitusvirhe ja haluttaisiin korjata se itse nopeasti. Ohjelma on suunniteltu niin, että jos sisäänkirjautuvalla käyttäjällä ei ole luotuna ansioluetteloa, käyttäjä ohjataan ensin sivulle, josta aloitetaan ansioluettelon luominen. Näkymistä ja toiminnoista kerrotaan tarkemmin osissa 6.1 ja 6.2. Lähes kaikille sivuille yhteinen osa on pohjapiirros, joka näkyy kuvassa 8.

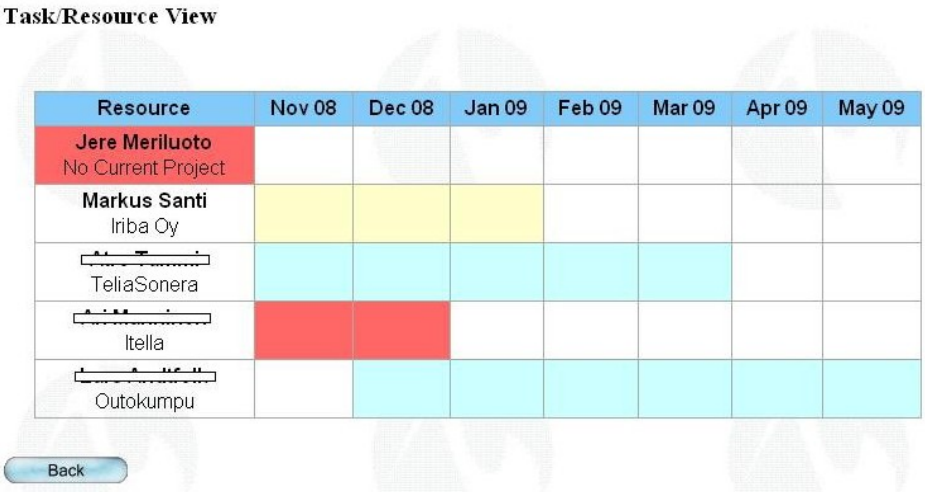
Kuva 8. Käyttäjän perusnäkymä.

Kuvassa 8 näkyy sivujen pohjapiirrokseen kuuluva Iriba CV Manager -logo, navigaatiopalkki, kielenvalintakohta sekä vasemmassa reunassa olevat käyttöliittymänapit, jotka ovat lähes joka sivulla käytössä. Käyttöliittymänappien oikealle puolelle ladataan aina haluttu JSP-sivu, jossa voidaan esimerkiksi muokata, lisätä tai poistaa tietoja ansioluettelosta. Muita näkymille yhteisiä sivuja ovat tiedotussivu (Announcements) ja resurssinäkymä (Resource View).



Kuva 9. Tiedotussivun näkymä.

Tiedotussivu on yleinen viestisivu, johon voi jättää vikailmoituksia, kehitystoivomuksia tai yleisiä viestejä. Tiedotussivun näkymästä esimerkki on kuvassa 8. Resurssinhallinta-työkalusta löytyy kaksi näkymää, joista tavallisen käyttäjän näkymä on kuvassa 10.

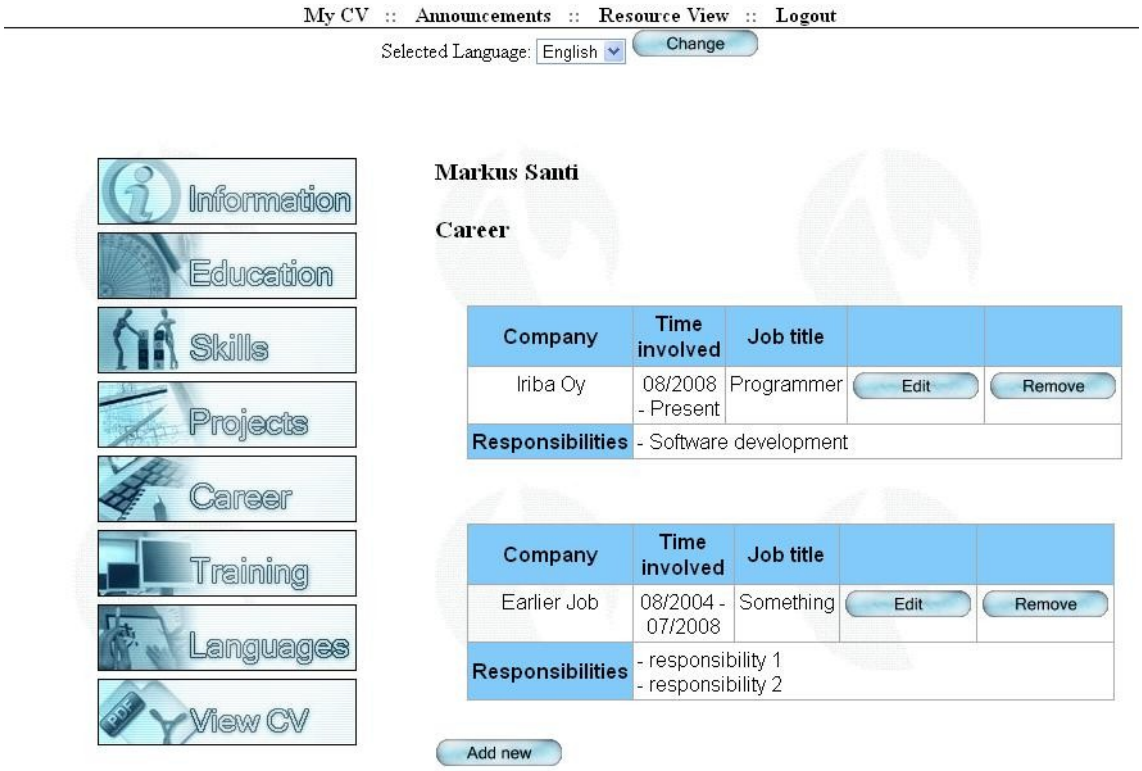


Kuva 10. Resurssinäkymä.

Resurssinäkymässä on näkyvissä työntekijöiden tämänhetkiset sijainnit. Näkymässä ilmenee myös, miten pitkään henkilö on projektissa töissä. Tämä on toimitusjohtajan työkalu; ylläpitäjän oikeuksilla tietoja näkee enemmän ja tietoja voi myös muuttaa. Kaikilla käyttäjillä on kumminkin nähtävissä perusnäkymä.

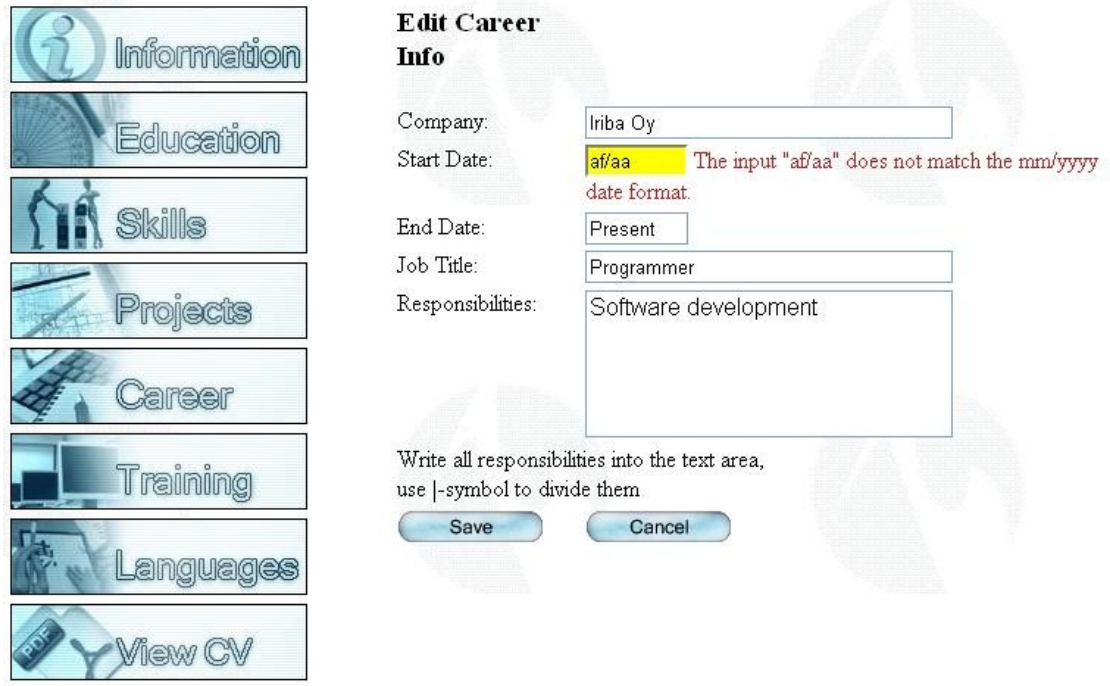
6.1 Käyttäjän näkymä

Käyttäjän näkymän perussivu näkyy kuvassa 8. Oikealla näkyvistä napeista voi navigoida ansioluettelon tietoja käsitteleville sivuille. Suurin osa ansioluettelon tietoja käsittelevistä sivuista käyttävät samaa kaavaa: painamalla nappia pääsee sivulle, missä näytetään jo syötetyt tiedot. Tietoja pystyy muuttamaan ja poistamaan niille tarkoitetuista napeista painamalla, jolloin käyttäjä siirretään tiedon muuttamiseen tarkoitetulle sivulle. Niistä palataan jälleen syötettyjen tietojen sivulle. Yksikään kohta ei ole aivan identtinen, sillä kenttien määrä vaihtelee ja joillakin sivuilla on alasvetoikkunoita tekstikenttien lisäksi. Kuvissa 11 ja 12 näkyy esimerkkejä perustoiminnallisuudesta. Esimerkkinä on käytetty uratietojen näyttämis- sekä muokkaamissivuja.



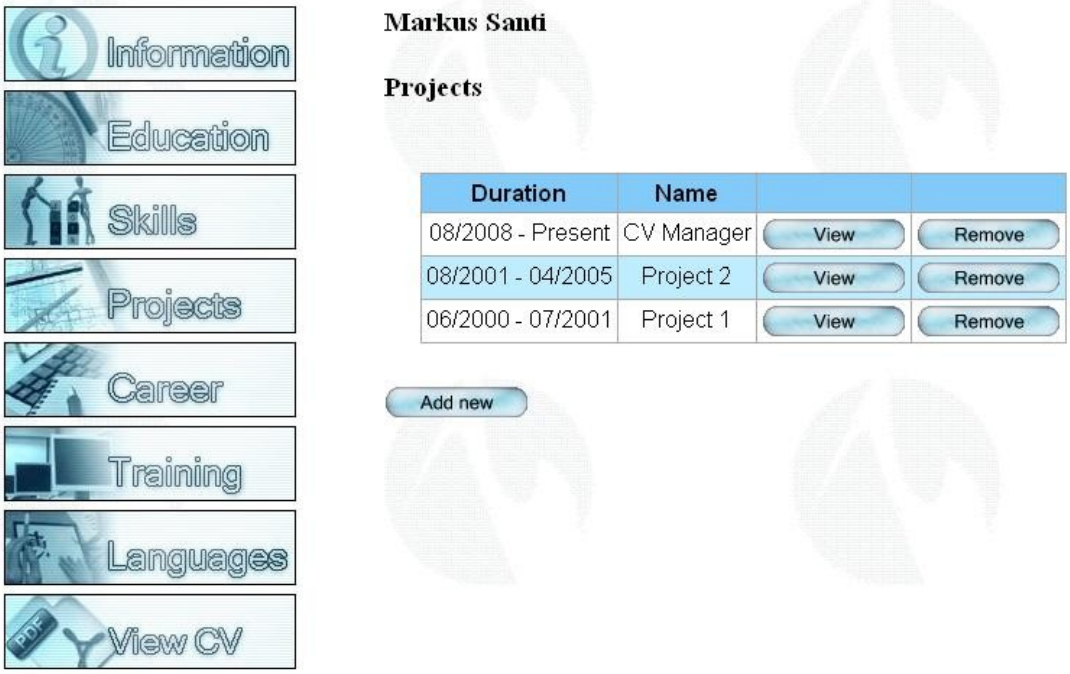
Kuva 11. Uratietojen näkymä.

Tietokantaan syötetyt uratiedot näkyvät taulussa ja ovat aina aikajärjestyksessä riippumatta syöttöjärjestyksestä. Add new -napista voi lisätä uuden uratiedon, Edit-nappi siirtää käyttäjän muutossivulle ja Remove-nappi poistaa alkion.



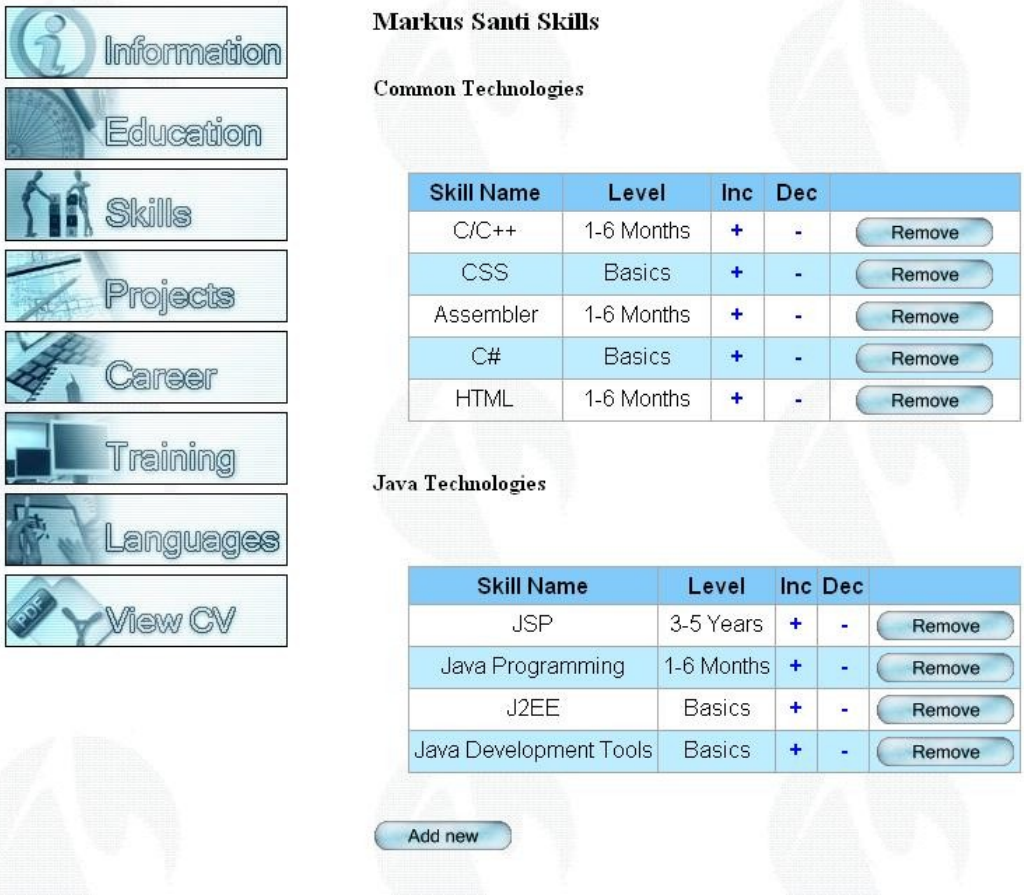
Kuva 12. Luodun uratiedon muuttaminen.

Tietoja muutettaessa ruudulle ilmestyy kuvan 12 näköinen näkymä. Muutettavan tiedon sisältö tulee ruudulle ja tekstikenttiin voi tehdä muutoksia. Uutta uratietoa luodessa näkymä on vastaava, mutta kentät ovat tyhjiä eikä ohjelmanappeja näy vasemmalla. Virheellisen tiedon syöttämisen tapahtuessa ohjelma ei suostu tallettamaan muutoksia vaan näyttää virheellisen kohdan värjäämällä tekstikentän pohjan keltaiseksi ja antamalla sen viereen virheilmoituksen. Jokainen tietoa muuttava tai tietoa lisäävä sivu on tehty samalla periaatteella.



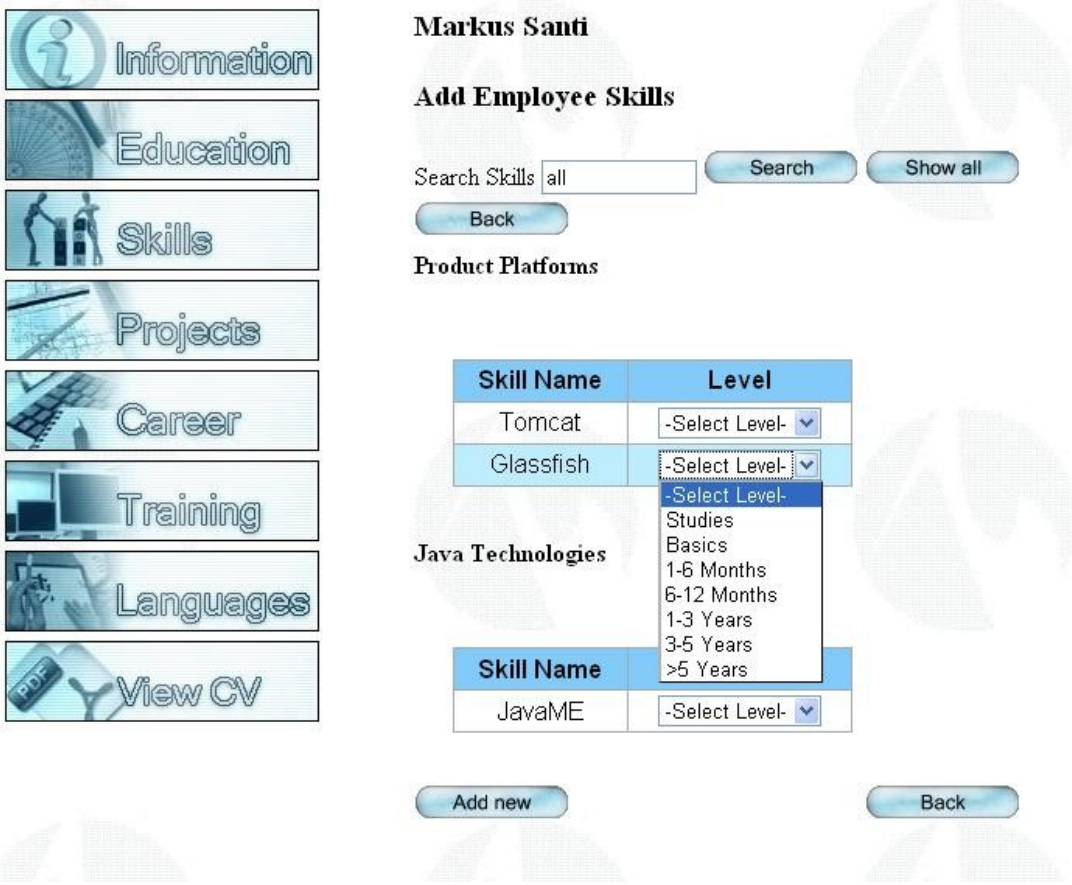
Kuva 13. Projektien listanäkymä.

Projects-nappia painaessa ei näe listausta projekteista tietoineen vaan ainut näkyvä tieto on projektin nimi ja sen kesto. Projektitietoja on sen verran paljon, että niiden käsittelyyn on lisätty ylimääräinen sivu, jossa listataan projektit päivämäärien mukaan järjestettynä. View-nappia painettaessa pääsee sivulle, josta näkee projektin kaikki tiedot ja pääsee muokkaamaan projektien tietoja.



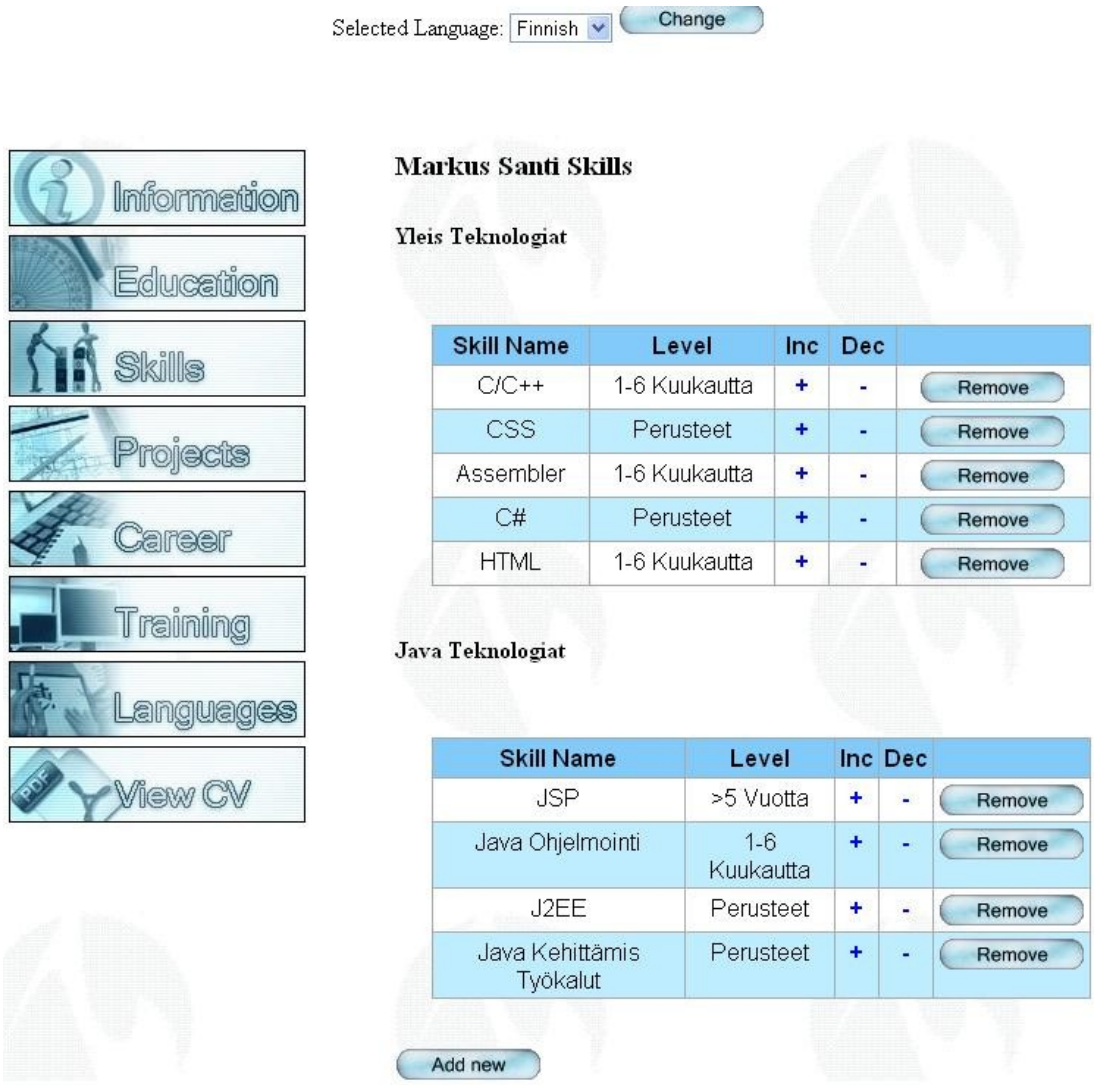
Kuva 14. Työntekijän taitojen näyttämissivu.

Työntekijän taitoja kuvaava sivu on olennaisesti erilainen osa ansioluetteloa tehtäessä. Taidot sekä kategoriat, joihin taidot kuuluvat, ovat ennalta määrättyjä. Käyttäjä pystyy vain lisäämään valmiista listasta itselleen taitoja ja asettamaan niihin oman osaamistasonsa. Taidot tulevat näkyviin kategorioittain, ja ne on järjestetty aakkosjärjestykseen. Plus (+)- ja miinus (-) -napeista pystyy muuttamaan suoraan osaamistasoaan. Painettaessa Add new -nappia käyttäjä siirretään taitolistaukseen, jossa näkyvät taidot, joita käyttäjä ei ole vielä lisännyt itselleen. Taitoa lisättäessä on valittava osaamistaso ja taitoja voi lisätä useampia kerrallaan. Taitojen lisäysnäkymä on esitetty kuvassa 15.



Kuva 15. Työntekijän taitojen lisäämisnäkymä.

Kaikki taidot, joissa on muutettu osaamistasoa (level) siirtyvät työntekijän taitoihin painamalla Add new -nappia. Back-nappia painamalla pääsee takaisin käyttäjän taitoja näyttävälle sivulle. Lisäksi sivulla on hakutoiminto, jolla voi hakea taitoja nimen perusteella. Jos taitoja on hyvin paljon, hakutoiminto nopeuttaa tiettyjen taitojen etsimistä. Haku on muodoltaan ns. ”sisältää”-haku, eli se ei palauta tarkkoja osumia, vaan kaikki osumat, jotka sisälsivät hakusanan.



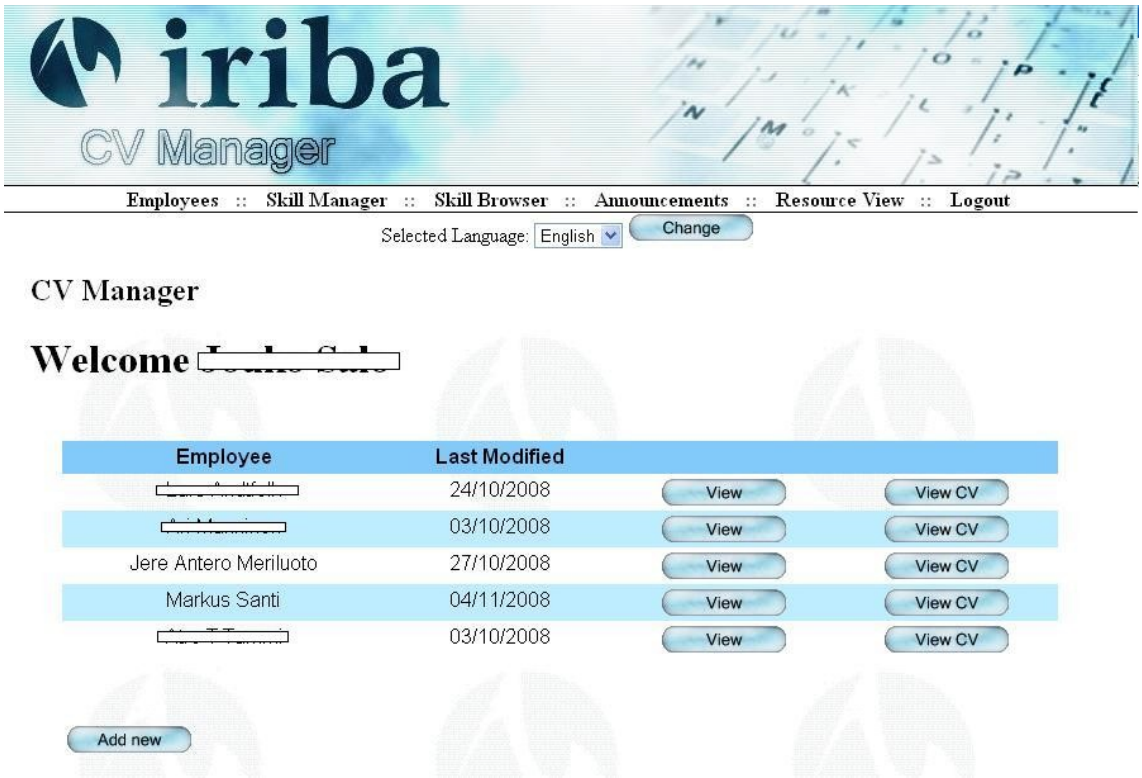
Kuva 16. Esimerkki kielen vaihtamisesta.

Kuvassa 16 näkyy esimerkki siitä, mitä kielen vaihtamisesta seuraa. Näkymä on sama kuin kuvassa 15, mutta kieli on vaihtunut suomeksi. Taitojen nimet, kategoriat ja taitotasot ovat kaikki muuttuneet suomenkielisiksi. Kielen vaihto toimii vastaavalla tavalla kaikissa sivuissa. Jos kieltä vaihdettaessa käsitellään esimerkiksi koulutusta, muuttuu tekstikenttien sisältö valitunkielisiksi. Kirjoitettavat tiedot on tosin kirjoitettava itse molemmille kielille, jos käyttäjä haluaa saada ansioluettelonsa myös suomeksi. Helpoin tapa tähän on tietoja luotaessa aluksi kirjoittaa englanniksi, tallentaa muutokset, vaihtaa kieltä ja luoda uudestaan kyseinen osio, jossa kohdat kirjoitetaan suomen kielellä myös ja tallennetaan. Seurauksena tietoalkio on talletettu tietokantaan molemmilla kielillä.

Vasemmalla olevista napeista alin (View CV) avaa selaimeen uuden välilehden ja avaa siihen ansioluettelon PDF-muodossa. Ansioluettelo tulostuu sillä kielellä, joka oli ennen napin painallusta valittuna. PDF-välilehdessä voi halutessa joko tulostaa ansioluettelon paperille tai tallettaa sen koneelle PDF-tiedostoksi.

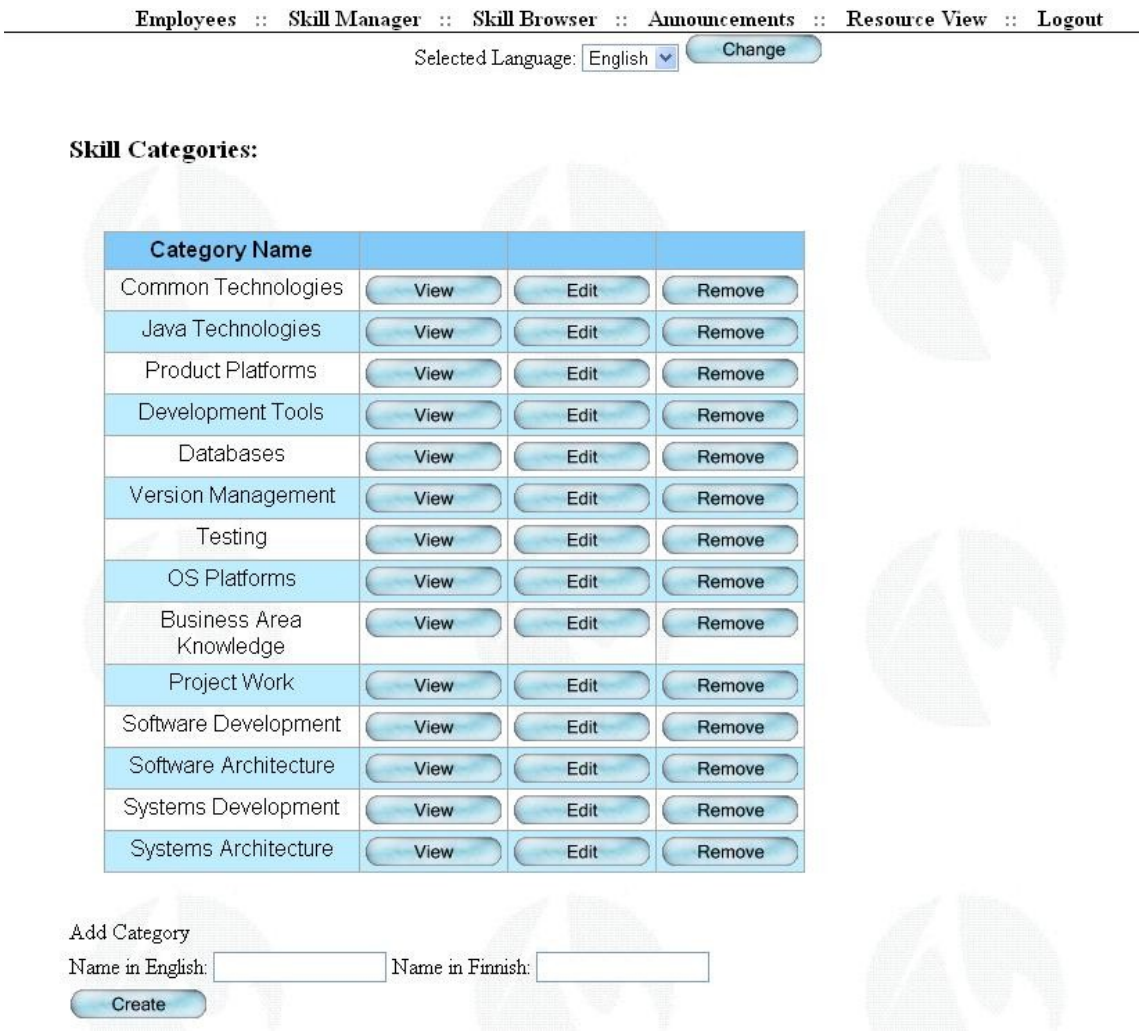
6.2 Ylläpitäjän näkymä

Kuten jo aikaisemmin todettiin, ylläpitäjän näkymä ohjelmassa on laajempi kuin käyttäjän. Käyttäjä näkee vain omaa ansioluetteloaan koskevat asiat, kun taas ylläpitäjä pystyy muuttamaan kaikkien työntekijöiden ansioluetteloita. Lisäksi navigointipalkkiin ilmestyy ylimääräisiä linkkejä, jotka ovat ylläpitäjien käytettävissä olevia työkaluja. Tässä luvussa käydään läpi vain ylläpitäjille näkyvät sivut, joita tavalliset käyttäjät eivät näe.



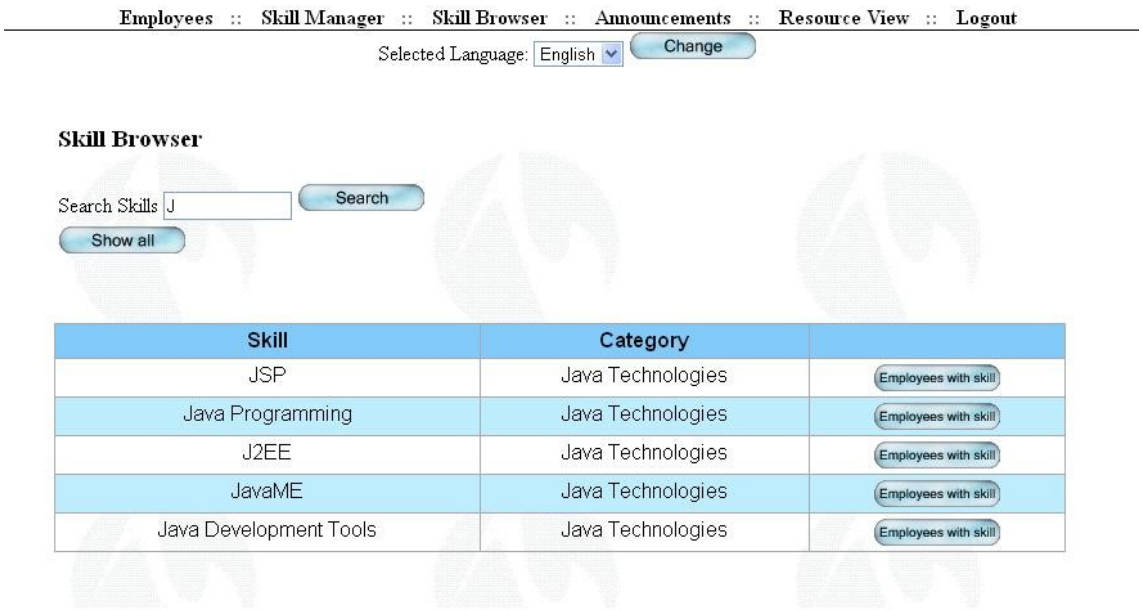
Kuva 17. Ylläpitäjän aloitussivu.

Kuvassa 17 näkyy ylläpitäjän aloitussivu. View CV -nappi on sama kuin käyttäjillä. Ylläpitäjä pystyy katsomaan suoraan talletettuja ansioluetteloita. View-napista pääsee käyttäjän ansioluettelon käsittelysivulle, joka on sama kuin tavallisten käyttäjien perussivu. Työntekijöiden taulukossa työntekijät näkyvät aakkosjärjestyksessä sekä taulukossa on viimeksi muokattu -kenttä. Tämä kenttä kertoo, milloin käyttäjä on viimeksi tehnyt muutoksia tai lisäyksiä ansioluetteloonsa. Yksi tärkeä osa ansioluetteloa on se, että se on ajan tasalla. Tästä ansioluetteloita tarvitseva toimitusjohtaja pystyy asian helposti tarkistamaan.



Kuva 18. Taitojen ylläpidon pääsivu.

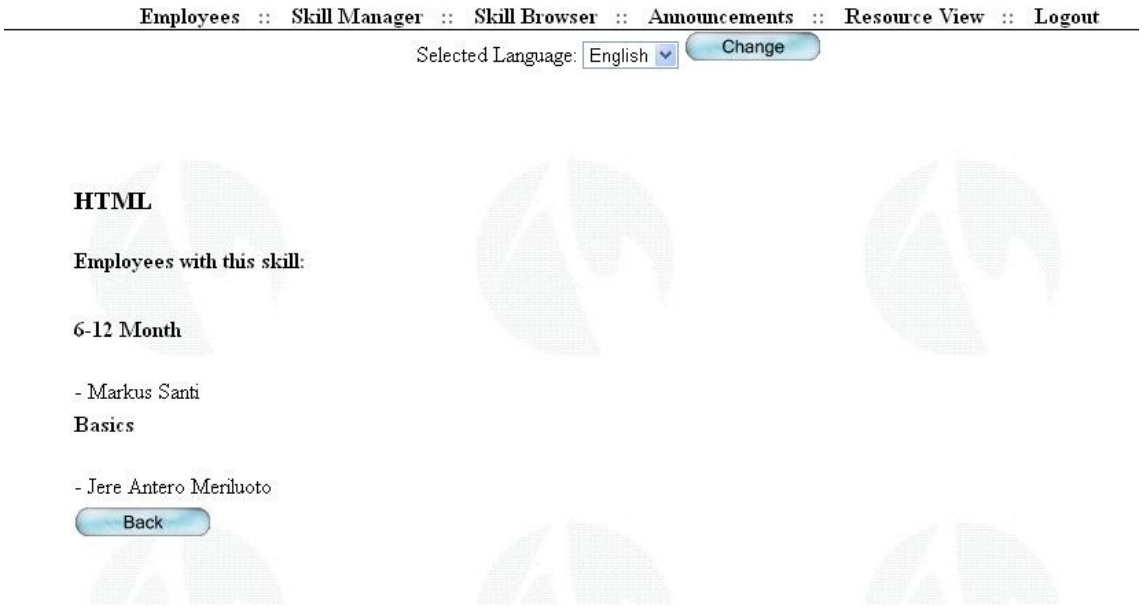
Painamalla navigointilinkeistä Skill Manager -linkkiä pääsee taitojen ylläpitosivulle. Näillä sivuilla ylläpitäjä määrää mitkä taidot ovat työntekijöiden valittavissa. Kategorioiden lisääminen on asetettu suoraan samalle sivulle ja täyttämällä tekstikentät sekä painamalla Create-nappia listan loppuun ilmestyy uusi kategoria. Kategorian sisältämiä taitoja pääsee katsomaan View-napista. Tällöin avautuu samantapainen näkymä, jossa on valitun kategorian sisältämät taidot. Kategorian sisällä taitojen lisääminen toimii samalla periaatteella kuin itse kategorioiden lisääminen. Sekä kategorioiden että taitojen muuttaminen onnistuu Edit-nappia painamalla, jolloin avautuu ohjelmalle tyypillinen tekstikenttänäkymä ja talletusmahdollisuus.



Kuva 19. Taitojen selaussivu.

Seuraava ylläpitäjien työkalu on taitojen selaussivut, joihin pääsee navigointilinkistä Skill Browser. Selaussivulla aukeaa ensimmäisenä hakuvalikko. Samaan tapaan kuin työntekijän taitoja lisätessä haku toimii ”sisältää”-periaatteella, eli kaikki taidot, joissa hakusana esiintyy, palautuvat ruudulle. Tämä työkalu lisättiin toimitusjohtajan pyynnöstä. Jos mahdollinen asiakas soittaa ja kysyy, onko yrityksellä jonkun tietyn taidon osaajia, on taitojen selaussivuilta helppo selvittää tilanne. Jokaisen taidon oikealla puolella on aluksi kategoria, johon se kuuluu, ja Employees with skill -nappi.

Painamalla nappia sivulle tulostuvat kaikki työntekijät, jotka ovat lisänneet tämän taidon itselleen. Tulostus on osaamistason mukaan järjestyksessä. Näkymästä on esitetty esimerkki kuvassa 20.



Kuva 20. Näkymä haetun taidon omaavista työntekijöistä.

7 Kehitys- ja testiympäristö

7.1 Kehitysympäristö

Kehitysympäristönä on käytetty Iriba Oy:n kehittämää perustestiympäristöä. Ympäristö koostuu Eclipse Ganymede -sovelluskehittimestä [26], MySQL-tietokantasovelluksesta [27] sekä GlassFish web-palvelimesta [28]. Ohjelma kirjoitettiin Java-ohjelmointikielellä ja käyttöliittymää tehtäessä oli käytettävä Stripes Frameworkia. Kehitysympäristö on yrityksemme kokeneempien ohjelmoijien valitsema. Stripes lisättiin ympäristöön helpottamaan käyttöliittymien tekemistä, jonka seurauksena GlassFish oli luonteva vaihtoehto web-palvelimeksi, koska Apache Tomcat [29] ei tue kaikkia tarvittavia ominaisuuksia, joita kehitetty ohjelma olisi tarvinnut.

7.2 Testitietokanta

Testitietokanta pystytettiin samalla päätteellä, jossa ohjelmaa rakennettiin. Testitietokantaa testattiin lähettämällä ja kyselemällä dataa tietokannan testiohjelmalla. Tietokannan sisällön tutkimiseen käytettiin MySQL Query -selainta, joka mahdollistaa graafisen taulujen sisällön tutkimisen, konsolipäätteen sekä tietokantataulujen helpon luomisen. Näin pystyttiin testaamaan ohjelman datakontrolliyksiköiden tuottamia SQL-lauseita.

7.3 Testaus

7.3.1 Tietokannan testiohjelma

Ohjelman tietokantapuoli toteutettiin ennen käyttöliittymää, joten tietokannan testaaminen piti suorittaa omalla ohjelmallaan. Tietokannan ja tietokantaa käsittelevien luokkien toiminnan testaamiseksi kirjoitettiin konsolilla ajettava Java -ohjelma. Ohjelmassa oli peräkkäin tietokantaa käsitteleviä metodeja käskyinä. MySQL Query Browser -ohjelman avulla tarkistettiin testien tulokset. Testiohjelmassa käytiin jokainen taulu aluksi yksitellen läpi mahdolliset virheet ja ongelmat korjaten. Jokaisen taulun yksittäisen testaamisen jälkeen siirryttiin koko tietokannan testaamiseen.

Yksittäisten taulujen testaamisessa aluksi tauluihin sijoitettiin kaksi alkiota, tämän jälkeen testattiin hakua, päivitettiin dataa tietokantaan, testattiin hakua uudestaan ja lopuksi poistettiin tiedot tietokannasta.

Kaikkia ongelmia kyseinen testitapa ei poistanut, mutta käyttöliittymän puuttumisen vuoksi tämä oli nopein ja tehokkain tapa testata tietokannan yleistoimivuutta. Lopullinen tietokannan testaaminen suoritettiin käyttöliittymän välityksellä, samalla kun koko ohjelma oli testattavana.

7.3.2 Käyttöliittymän testaus

Käyttöliittymän testaus oli tietokannan testaamista monimutkaisempi asia. Käyttöliittymä käyttää montaa luokkaa jokaisen käskyn suorittamiseen. Lisäksi virheilmoituksesta ei aina saatu selville ongelman alkuperäistä aiheuttajaa. Käyttöliittymää kirjoitettaessa ohjelmaa ajettiin Eclipse-sovelluskehittimen sisällä. Näin saatiin käyttöön debug-moodi, jonka avulla pystyttiin virheen sattuessa pysäyttämään ohjelman ajo mihin tahansa kohtaan koodissa. Tämä työkalu helpotti paljon virheiden etsimistä. Koska käyttöliittymä on suhteellisen laaja, ei ohjelmoija yksin voi löytää kaikkia pikkuvirheitä.

Ensimmäisen version valmistuessa ohjelmaa käytti pari testikäyttäjää. Käyttöliittymän virheet tulevat ainoastaan testaamalla esille, ja testikäyttäjien avulla ohjelmasta saatiin esille virheitä, joita itse tekijät eivät löytäneet.

8 Java paketit ja -luokat

Insinöörityön liitteenä on ohjelman eri luokkien lähdekoodia. Koko projektin lähdekoodin sisällyttäminen insinöörityöhön olisi ollut työlästä sekä tehotonta, joten päätimme esittää luokkien perusrakenteita. Lähdekoodin selaamisen helpottamiseksi tässä luvussa käydään läpi ohjelman rakenne ja ohjelmassa esiintyvät tiedostot. Ohjelman luokat on jaettu paketteihin luokkien toiminnan mukaan. Tietyssä paketissa on aina tiettyä ohjelman osaa käsittelevät luokat. Jokaista ohjelman tiedostoa ei käydä yksitellen läpi, sillä moni luokka tekee samaa asiaa, mutta eri osa-alueelle. Luokkien tarkemmat tehtävät selviävät lähdekoodista kommenttien avulla.

com.iriba.cvgenerator.conn

Conn-paketin tärkein tiedosto on RDBDAOFactory.java. Tämän luokan avulla muodostetaan yhteys tietokannan ja ohjelman välille. Lisäksi pakettiin on sijoitettu käyttöliittymän kieliluokat Language.java sekä Languages.java. Viimeinen tiedosto on Level.java, joka on ohjelmassa esiintyvien alasvetovalikoiden käsittelemistä varten tehty luokka.

com.iriba.cvgenerator.actionbean

ActionBean-paketissa on kaikki käyttöliittymän tapahtuman- ja tiedonkäsittelyyn liittyvät luokat. Suurin osa paketin tiedostoista ovat tapahtumankäsittelyluokkia. Luokat erottaa ActionBean.java-päätteestä. Päätteen edessä olevasta nimestä selviää, mihin osaan käyttöliittymässä luokka liittyy. Manage-alkuiset luokat käsittelevät tiedon

talletusta tietokantaan, muut luokat tiedon hakua ja käyttöliittymään liittyviä toimintoja. Esimerkkinä LoginActionBean.java käsittelee käyttöliittymän sisäänkirjautumista.

ActionBean-luokat käyttävät context-nimistä luokkaa, johon voi tallettaa pysyvää tietoa, joka säilyy aina siirryttäessä yhdeltä tapahtumankäsittelijältä toiselle. Ohjelman toimintaa varten oli kirjoitettava oma context-luokka CVGActionBeanContext.java. Tähän luokkaan on talletettu kaikki tiedot, jotka halutaan säilyttää koko ohjelman käytön ajan, esimerkiksi sisäänkirjautuneet henkilön tiedot, käyttötaso ja valittu kieli.

Paketista löytyy myös kaksi yksittäistä tiedostoa. Ensimmäinen on EmpManager.java. Kuten aikasemmin jo todettiin, tämä luokka toimii ohjelman eri puolien solmukohtana ja hoitaa suurimman osan tiedonkäsittelystä. Toinen luokka on SecurityFilter.java, joka hoitaa tietoturvasuodatuksen ohjelmassa. Tämä luokka mahdollistaa sen, että ilman sisäänkirjautumista ohjelmaa ei voi käyttää.

com.iriba.cvgenerator.pdf

PDF-paketista löytyvät kaikki luokat, jotka liittyvät ansioluetteloiden muuntamiseen PDF-muotoon. Paketti koostuu kolmesta tiedostosta. CVStyle1.xsl on tyylitiedosto, jossa on määritettynä tulostettavan sivun ulkoasu. XMLGenerator.java luo XML-tiedoston tietokannasta haettujen tietojen perusteella. PDFGenerator.java on luokka, jota kutsutaan ansioluetteloita luotaessa.

com.iriba.cvgenerator.to

TO-paketti eli siirto-objektit (Transfer Object) sisältää kaikki TO-luokat, joiden avulla siirretään tietoa tietokannan ja ohjelman välillä. Paketissa ei ole muuta kuin jokaista tietokannan taulua vastaava TO-luokka.

com.iriba.cvgenerator.vo

VO-paketti eli arvo-objektit (Value Object) sisältää kaikki VO-luokat, joita käytetään tiedonsiirtämiseen ohjelman ja käyttöliittymän välillä. Jokaiselle siirto-objektille löytyy vastaava arvo-objekti. Jokaista arvo-objektia kohden on kolme luokkaa: perus-VO-luokat, joihin talletetaan siirto-objekteista tieto käyttöliittymässä näytettävään muotoon, List-luokat, jotka hoitavat käyttöliittymän taulukoiden sisällön hakemisen, sekä Util-luokat, jotka suorittavat muunnokset siirto-objektista arvo-objektiksi ja toisinpäin. Lisäksi paketissa on ylimääräisiä arvo-objektiluokkia, joihin talletetaan tietty eri tauluja yhdistävä informaatio, esimerkkinä tästä EmpSkillListVo.java, joka kerää työntekijän taitojen listan kategorioittain. Tämän tiedon kokoamiseksi on käytävä läpi kolme eri taulua.

com.iriba.cvgenerator.dao

DAO-paketissa ovat kaikki tiedostot, jotka hoitavat itse tietokannan käsittelyn. Yhteyden tietokantaan antaa conn-paketin tehdas-luokka, mutta itse käskyt tietokannalle välittävät tämän paketin datakontrolliyksiköt. Jokaiselle tietokannan päätaululle löytyy omat RDB(nimi)DAO.java- sekä (nimi)DAO.java-tiedostot. Ensimmäisessä ovat itse käskyt, jotka ovat suunniteltuja MySQL-tietokannalle. Jälkimmäinen ryhmä koostuu rajapinnoista, joiden avulla RDB-luokkia kutsutaan.

Web Content

Web content -hakemistossa ovat kaikki käyttöliittymän ulkoasuun liittyvät tiedostot kuten sivujen tyylitiedosto IribaStyle.css, sekä kaikki graafiset komponentit. Kuvat ovat talletettuna images-alihakemistoon. Buttons.jsp sisältää käyttöliittymän sivussa näkyvän navigaatiovalikon, jota tarvitaan useissa sivuissa. Layout-alihakemistossa sijaitsee käyttöliittymän käyttämä pohjapiirros (logo, alatunniste).

Suurin osa Web Content -hakemistosta löytyvistä tiedostoista on käyttöliittymän JSP-sivuja. Jokainen sivu, joka käyttöliittymässä on, on kirjoitettu omaan tiedostoonsa. JSP-tiedoston nimestä saadaan selville, mikä sivu se käyttöliittymässä on. Add-alkuiset sivut ovat ansioluetteloiden osia lisääviä sivuja, vastaavasti EditXXXInfo-alkuiset ovat olemassa olevan tiedon muokkaamista varten tehtyjä sivuja. Emp-päätteiset sivut ovat käyttöliittymässä eri ansioluettelon sisältöjen näyttämiseen tarkoitettuja sivuja. Lyhenteet sivujen nimissä tarkoittavat seuraavaa: Ann = Announcements, Car = Career, Cat = Category, Edu = Education, Emp = Employee, Lan = Languages, Pro = Project, Ski = Skill ja Tra = Training.

Kaikki JSP-sivut koostuvat samoista osista. Tiedoston alussa on aina määritetty ne tagi-kirjastot, joita sivu tarvitsee. Tämän jälkeen sivu koostuu HTML-koodista, johon on sijoitettu stripes-tageja toiminnallisia osia varten (napit, tekstikentät yms.)

Muut

Ohjelmassa on myös yksittäisiä tiedostoja, joille oman paketin tekeminen ei ollut tarpeellista. Järjestelmäpoikkeus SystemException.java löytyy com.iriba.cvgenerator-hakemistosta. Muita tiedostoja ovat properties-päätteiset tiedostot src-hakemistossa. Näissä tiedostoissa on määrätty ohjelmaan lisättyjen toimintojen ominaisuudet. StripesResources.properties-tiedostossa määrätään Stripes Frameworkin ominaisuudet. Tämän lisäksi tiedostoon on lisätty käyttöliittymän vahvistussääntöihin liittyviä toimintoja ja virheilmoituksia.

9 Yhteenveto

Ansioluetteloiden hallintajärjestelmä loi yhtenäisen rakenteen Iriba Oy:n työntekijöiden ansioluetteloille. Hallintajärjestelmän avulla ansioluetteloiden päivittäminen ja hallitseminen helpottuivat huomattavasti. Varsinkin tiettyjen taitojen ja teknologioiden osaajien hakeminen nopeutui yli kymmenkertaisesti, kun voitiin yksinkertaisesti täyttää hakukenttä ja painaa nappia, jolloin ohjelma automaattisesti etsi tietoja kaikista ansioluetteloista sen sijaan, että tietoa etsittäisiin käsipelillä eri dokumenteista.

Ansioluetteloiden lukuisista ulkomuodoista ja formaateista päästiin eroon, sillä ohjelma tuottaa yhdenmuotoisia ja -näköisiä ansioluetteloita yhdessä formaatissa. Ansioluetteloita ei myöskään tarvitse enää säilyttää isoja määriä tietokoneen kovalevyllä eri tiedostoissa, vaan kaikki on kätevästi yrityksen palvelimella pyörivässä tietokannassa.

Pariohjelmointi todettiin tehokkaaksi tavaksi ohjelmien toteuttamisessa. Kirjoitus- sekä ajatusvirheet saatiin nopeasti korjattua pois, kun koodia tutki kaksi silmäparia. Myöskin toteutuksessa olevien osien suunnittelu nopeutui molempien seuratessa toteutusta herkeämättä. Tiedettiin siis aina missä mennään, mitä pitäisi tehdä ja pystyttiin keskustelemaan siitä, miten saavutettaisiin haluttu lopputulos. Myös uusien teknologioiden opiskelu kahdestaan helpotti suunnattomasti. Monesti vastaan tuli tilanne, jossa toinen oli oppinut jostakin asiasta toisen puolen ja toinen toisen. Tällöin pystyttiin yhdistämään opitut asiat ja saatiin yhteinen käsitys esimerkiksi uuden kielen rakenteesta. Vaikka pariohjelmointi kuluttaa tuplamäärän resursseja (kaksi ohjelmoijaa, kaksi palkkaa), tuottavuus on silti yleensä korkeampaa kuin yksinohjelmoinnissa.

Työssä tehtiin myös yksin tehtäviä osioita, kuten ansioluettelon tulostuksen hoitava XML- ja PDF-generaattori. Meriluodon tehdessä ansioluettelon tulostamiseen tarvittavia komponentteja, Santi rakensi käyttöliittymää. Lopulta palaset haluttiin

liimata yhteen, jotta käyttöliittymästä voitaisiin pyytää tulostettavaa ansioluetteloa. Käyttöliittymän ja ansioluetteloiden tulostuskomponentin yhdistämisessä meni jonkin aikaa, sillä molempien piti opetella komponenttien käyttäytymistapa, eli mitä argumentteja ansioluetteloiden tulostuskomponenteille lähetetään, mitä se palauttaa, mihin käyttöliittymässä voidaan liittää kyseinen komponentti ja miten tulos näytetään käyttäjälle. Esimerkkinä tästä toimi hyvin tiedon siirto tulostuskomponenteille, jotka halusivat ansioluettelon tiedot siirtoyksikköinä. Käyttöliittymärajapintaan tuli tällöin rakentaa metodeja, jotka hakisivat kaikki käyttäjän tiedot tietokannasta ja lähettäisivät ne kokoelmana tulostuskomponentille, joka prosessoisi tiedon ja lähettäisi takaisin tulostettavan PDF:n.

Molemmissa työskentelytavoissa oli omat hyvät ja huonot puolensa. Yksintyöskentelyn hyviä puolia oli rauhassa miettiminen ja huonona puolena mahdollisten ongelmatilanteiden käsittely, jolloin piti joko itse miettiä ratkaisua pitkään tai lähteä etsimään apua. Parityöskentelyn hyviä puolia olivat juuri tehokkuus, ongelmaratkaisukyky sekä sosiaalinen kanssakäyminen ja huonona puolena se, ettei aina päässyt itse kirjoittamaan ohjelmaa. Pariohjelmoinnissa korostuivat molempien osapuolien vahvuudet, esimerkiksi Santin aikaisempi Java-kokemus ja Meriluodon HTML- ja CSS-teknologioiden kokemus.

Tehokkuusnäkökulmaa tarkasteltaessa on otettava huomioon, että itse ohjelman kirjoittaminen vie vain pienen osan ohjelman tekemiseen käytetystä ajasta. Ongelmatilanteiden ratkominen sekä virheiden etsiminen vie suurimman osan ajasta. Parityöskentely nopeuttaa virheiden löytämistä sekä korjaamista nostaen näin tehokkuutta. Vaikka kahden ohjelmoijan kustannukset ovat kaksinkertaiset yhteen verrattuna, tietyissä tapauksissa työhön käytetty aika voi vähentyä alle puoleen. Tässä projektissa vallitsi tuttu ja oppimista painottava ilmapiiri, jonka takia katsoimme tarpeettomaksi käyttää virallisia ohjelmointimenetelmiä, eli valmiita kaavoja, kuinka pariohjelmointia tulisi käyttää tehokkaasti.

Ansioluetteloiden hallintajärjestelmä toimi loistavana tutustumisprojektina integraatiosovelluksissa käytettävien teknologioiden sekä web-ohjelmoinnin maailmaan. Projektissa opittiin monen uuden teknologian perusteita, käyttötapoja sekä vahvistettiin tietämystä jo tunnetuista teknologioista. Eri teknologioiden hyödyntäminen useissa komponenteissa halutun toiminnallisuuden saamiseksi opetti useiden eri lähestymistapojen etsimistä ja vertailua.

Erityisen tärkeinä asioina opittiin oliopohjaisten sovellusten rakentamista Java-kielellä, tietokantojen rakentamiseen, ylläpitoon ja hallintaan käytettävää kieltä (SQL) sekä tietokantojen hallintaan käytettäviä valmiita graafisia sovelluksia (MySQL Query Browser).

9.1 Käyttöönotto

Ohjelman ensimmäinen versio on siirretty yrityksen palvelimelle, ja testikäyttäjät ovat etsineet ohjelmasta korjattavia ohjelmointivirheitä. Santi, jolle ohjelmointivirheiden korjaaminen määrättiin, on tehnyt korjauksia ohjelmaan testikäyttäjien antaman palautteen perusteella. Tavoitteena on, että ohjelmasta tulee tämän vuoden aikana Iriban ansioluetteloiden virallinen käsittelijä.

9.2 Jatkokehitys

Projektissa toteutettu ohjelma on ensimmäinen versio ansioluetteloiden hallintajärjestelmästä. Ohjelmaa kehitetään ja korjataan sitä mukaa, kuin virheitä tai parannettavaa löytyy. Ohjelman ympärillä on kehitelty ajatusta yrityksen omasta portaalista, johon yhdistettäisiin erilaisia toimintoja, kuten sähköposti, yleinen viestialue, foorumi ja projektien versionhallintaohjelmisto. Ohjelman käyttöliittymä antaa hyvän pohjarakenteen mahdollisille kehityskomponenteille tarjoamalla valmiin käyttöliittymärajapinnan.

Ensimmäinen jatkokehitystoimenpide olisi yhtenäistää sisäänkirjautuminen yrityksen palvelimen sisäänkirjautumisen kanssa. Tämä tarkoittaa sitä, että ohjelma saisi suoraan palvelimelle kirjautumisesta tiedot ohjelman sisäänkirjautumiseen. Tämän takia ohjelmaan tehtiin vain yksinkertainen sisäänkirjautuminen. Muu jatkokehitys liittyy ohjelman toiminnallisten osien parantamiseen tai muuttamiseen. Nämä jatkokehitystehtävät ovat kaikki opetusmielessä tehtäviä muutoksia, eivätkä niin kriittisiä ohjelman toiminnalle.

Lähteet

1 Horstmann Cay S., Cornell Gary. Core Java Volume I - Fundamentals. Sun Microsystems. 1999.

2 Horstmann Cay S., Cornell Gary. Core Java Volume II - Advanced Features. Sun Microsystems. 2000.

3 Java 2 Platform Standard Edition 5.0 API Specification. (WWW-dokumentti.) Sun Microsystems. <http://java.sun.com/j2se/1.5.0/docs/api/> Luettu 8.2008 - 9.2008.

4 Stroustrup Bjarne. C++ Programming Language Third Edition. Addison Westley. 1997

5 Troelsen Andrew. C# And the .NET Platform Second Edition. Apress. 2003.

6 Alur Deepak, Crupi John, Malks Dan. Core J2EE Patterns. Sun Microsystems. 2003.

7 Data Access Object. (WWW-dokumentti.) Sun Microsystems. <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html> 20.5.2005. Luettu 8.2008

8 Data Access Object. (WWW-dokumentti.) Sun Microsystems. <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html> 20.5.2005. Luettu 8.2008

9 Transfer Object. (WWW-dokumentti.) Sun Microsystems. <http://java.sun.com/blueprints/patterns/TransferObject.html> 2002. Luettu 8.2008

10 DuBois Paul. My SQL CookBook 2nd Edition. O'Reilly Media. 2006.

11 MySQL. (WWW-dokumentti.) Wikipedia. <http://fi.wikipedia.org/wiki/Mysql> Luettu 8.2008

12 JSP Tutorial. (WWW-dokumentti.) JSPtut. <http://www.jsptut.com/>. Luettu 9.2008.

13 Introduction to HTML. (WWW-dokumentti.) W3Schools. <http://w3schools.com/html/default.asp> Luettu 9.2008

14 HTML. (WWW-dokumentti) Wikipedia. <http://en.wikipedia.org/wiki/HTML> Luettu 9.2008

15 Bartlett Kyle. CSS in 24 Hours 2nd Edition. Sams Publishing. 2006.

16 CSS. (WWW-dokumentti) W3Schools.
<http://w3schools.com/css/default.asp> Luettu 9.2008

17 Stripes Framework. (WWW-dokumentti.) Contegix.
<http://www.stripesframework.org/display/stripes/Home>. 26.9.2007.
Luettu 9.2008.

18 Apache Struts. (WWW-dokumentti.) Apache.
<http://struts.apache.org/> Luettu 9.2008

19 Gardner John Robert, Rendon Zarella L. XSLT & XPATH A Guide to XML
Transformation. Pretice Hall. 2002.

20 XML. (WWW-dokumentti.) World Wide Web Consortium.
<http://www.w3.org/XML/> Luettu 9.2008

21 XSLT. (WWW-dokumentti) World Wide Web Consortium.
<http://www.w3.org/TR/xslt> Luettu 9.2008

22 JDOM. (WWW-dokumentti.) World Wide Web Consortium.
<http://www.jdom.org>. Luettu 9.2008

23 Apache FOP. (WWW-dokumentti.) Apache.
<http://xmlgraphics.apache.org/fop/> Luettu 9.2008

24 Apache Log4J. (WWW-dokumentti.) Apache.
<http://logging.apache.org/log4j/1.2/index.html> Luettu 10.2008

25 Cisco VPN Client. (WWW-dokumentti) Cisco.
<http://www.cisco.com/en/US/products/sw/secursw/ps2308/>. Luettu 10.2008.

26 Eclipse Ganymede. (WWW-dokumentti.) Eclipse.
<http://www.eclipse.org/ganymede/> Luettu 8.2008

27 MySQL Community Server 6.0. (WWW-dokumentti.) MySQL.
<http://dev.mysql.com/downloads/mysql/6.0.html> Luettu 8.2008

28 Glassfish - Open Source Application Server. (WWW-dokumentti.) Java developers.
<https://glassfish.dev.java.net/> Luettu 9.2008

29 Apache Tomcat. (WWW-dokumentti.) Apache.
<http://tomcat.apache.org/> Luettu 8.2008

Liite 1: PDF-muotoinen ansioluettelo

Curriculum Vitae

Jere Meriluoto

Backgrounds

Year of birth	1986
Education	150 of 240 Studypoints. Major in programming., Metropolia University of Applied Sciences, 2006 - 2009

Summary

Working experience in Java since 2008. Basic knowledge of many basic programming languages and platforms. Good knowledge of Microsoft Windows OS.

Skill summary

Tools, techniques and products	**Experience**
Common Technologies	
C/C++	1-6 Months
HTML	Studies
CSS	6-12 Months
Java Technolgies	
JSP	Basics
Java Programming	1-6 Months
Product Platforms	
Glassfish	1-6 Months

Project experience

Project Name: Earlier project **Customer:** Customer1 **Time involved:** 3/2007 - 0/2008	**Description of project:** Basic project **Platform:** Basic project **Software:** Java **Own role:** Programmer **Responsibilities:** - many

Project Name: Old Project **Customer:** None **Time involved:** 0/2000 - 9/2001	**Description of project:** Very old project **Platform:** Very old project **Software:** Visual Basic **Own role:** Programmer **Responsibilities:** - programming

Career history

Iriba Oy 7/2008 - 4/1944	**Job title:** Programmer **Main responsibilities:** - Software development

Training

2002	Course of advanced CSS

Language skills

Swedish	Good
Finnish	Native Speaker
English	Fluent

Liite 2: Siirtoyksikön lähdekoodi

CategoryTO.java

```
package com.iriba.cvgenerator.to;

import java.util.ArrayList;
import java.util.Iterator;

import com.iriba.cvgenerator.conn.Language;
import com.iriba.cvgenerator.conn.Languages;

/*
 * CategoryTO
 *
 * TO päätteiset luokat ovat Transfer Object:a. Luokkia käytetään tiedon
 * siirtämiseen tietokannalta tietoa käsittelevälle ohjelmalle. Luokat
 * sisältävät vain tietyn taulun sisällön, ja näille muuttujille get- ja
 * set-metodit.
 */

public class CategoryTO {

        private int id;
        ArrayList<SkillTO> SkillTOList;
        ArrayList<Category_textTO> Category_textTOList;

        public CategoryTO() {
                SkillTOList = new ArrayList<SkillTO>();
                Category_textTOList = new ArrayList<Category_textTO>();

                for (Iterator iterator = new
                        Languages().getAllLanguages().iterator();
                        iterator.hasNext();) {
                        Language language = (Language) iterator.next();

                        Category_textTO temp = new Category_textTO();
                        temp.setLang(language.getId());
                        Category_textTOList.add(temp);
                }
        }

        public void setID(int i) {
                id = i;
        }

        public int getID() {
                return id;
        }

        public int addSkill(SkillTO sto) {

                SkillTOList.add(sto);

                return id;
        }

        public void setSkillTOList(ArrayList<SkillTO> sl) {
                SkillTOList=sl;
        }
        public ArrayList<SkillTO> getSkillToList() {
                return SkillTOList;
```

```
	}

	public void removeSkill(SkillTO sto) {
		SkillTOList.remove(sto);
	}

	public void setCategoryText(String l, String s) {
		for (int i = 0; i <= Category_textTOList.size() - 1; i++) {
			if (Category_textTOList.get(i).getLang().equals(l))
				Category_textTOList.get(i).setText(s);
		}
	}

	public String getCategoryText(String l) {
		String skillText = "";
		for (int i = 0; i <= Category_textTOList.size() - 1; i++) {
			if (Category_textTOList.get(i).getLang().equals(l))
				skillText =
Category_textTOList.get(i).getText();
		}
		return skillText;
	}

}
```

Liite 3: Erilaisten arvoyksiköiden lähdekoodia

CPViewVO.java

```
package com.iriba.cvgenerator.vo;

import java.util.ArrayList;
import java.util.Calendar;

/*
 * CPViewVO
 *
 * On luokka, joka sisältää CurrentProject osioon tulostettavan
 * taulukon kaikken tiedon. Luokka sisältää myös listana jokaisen
 * työntekijän CurrentProject-tiedot.
 *
 * Luokka muodostaa taulun yläriville aina nykyisen päivämäärän
 * mukaisen näkymän.
 */

public class CPViewVO {

        private int month;
        private String year;
        private String nyear;
        private ArrayList<CurrentProjectVO> cpList;


        public CPViewVO() {
                Calendar c = Calendar.getInstance();
                month = c.get(Calendar.MONTH);
                year = ""+c.get(Calendar.YEAR);
                nyear = ""+(c.get(Calendar.YEAR)+1);

        }

        public ArrayList<CurrentProjectVO> getResource() {
                return cpList;
        }
        public void setResource(ArrayList<CurrentProjectVO> cl) {
                cpList=cl;
        }

        public String getMonthName(int m) {
                if(m==0) return "Jan";
                else if(m==1) return "Feb";
                else if(m==2) return "Mar";
                else if(m==3) return "Apr";
                else if(m==4) return "May";
                else if(m==5) return "Jun";
                else if(m==6) return "Jul";
                else if(m==7) return "Aug";
                else if(m==8) return "Sep";
                else if(m==9) return "Oct";
                else if(m==10) return "Nov";
                else return "Dec";
        }

        public String getM1() {
                return getMonthName(month)+" "+year.substring(2);
        }
        public String getM2() {
```

```
                int i = month+1;
                if(i>11) return getMonthName(i-12)+" "+nyear.substring(2);
                else return getMonthName(i)+" "+year.substring(2);
        }
        public String getM3() {
                int i = month+2;
                if(i>11) return getMonthName(i-12)+" "+nyear.substring(2);
                else return getMonthName(i)+" "+year.substring(2);
        }
        public String getM4() {
                int i = month+3;
                if(i>11) return getMonthName(i-12)+" "+nyear.substring(2);
                else return getMonthName(i)+" "+year.substring(2);
        }
        public String getM5() {
                int i = month+4;
                if(i>11) return getMonthName(i-12)+" "+nyear.substring(2);
                else return getMonthName(i)+" "+year.substring(2);
        }
        public String getM6() {
                int i = month+5;
                if(i>11) return getMonthName(i-12)+" "+nyear.substring(2);
                else return getMonthName(i)+" "+year.substring(2);
        }
        public String getM7() {
                int i = month+6;
                if(i>11) return getMonthName(i-12)+" "+nyear.substring(2);
                else return getMonthName(i)+" "+year.substring(2);
        }

}
```

CurrentProjectUtil.java

```
package com.iriba.cvgenerator.vo;

import com.iriba.cvgenerator.to.CurrentProjectTO;

/*
 * CurrentProjectUtil
 *
 * Util päätteiset metodit muuttavat tietyn TO olion
 * VO olioksi tai VO olion vastaavasti TO olioksi.
 */

public class CurrentProjectUtil {

        public static CurrentProjectVO convertToVO(CurrentProjectTO cpTO) {
                return new CurrentProjectVO(cpTO.getID(),
cpTO.getEmployeeID(),
                                cpTO.getEndDate(),cpTO.getCompany(),
cpTO.getProject(),
                                cpTO.getStartDate(), cpTO.getPrice());
        }

        public static CurrentProjectTO convertToTO(CurrentProjectVO cpVO) {
                CurrentProjectTO to = new CurrentProjectTO();
                to.setID(cpVO.getID());
                to.setEmployeeID(cpVO.getEmployeeID());
                to.setEndDate(cpVO.getEndDate());
                to.setCompany(cpVO.getCompany());
                to.setProject(cpVO.getProject());
                to.setStartDate(cpVO.getStartDate());
                to.setPrice(cpVO.getPrice());
                return to;
```

```
        }

}
```

CurrentProjectVO.java

```
package com.iriba.cvgenerator.vo;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

/*
 * CurrentProjectVO
 *
 * VO päätteiset luokat ovat Value Object:a. Samaan tapaan
 * kuin tietokannoilta siirretään ohjelmalle tietoa TO olioissa,
 * siirryy ohjelmalta käyttöliittymälle tieto VO olioissa. Joissain
 * VO luokissa saattaa olla jotain tietojenkäsittelyä.
 */

public class CurrentProjectVO {
        private int id;
        private int employee_id;
        private Date end_date;
        private Date start_date;
        private int price;
        private String company;
        private String project;
        private String name;
        private int amount;
        private int[] monthview = {0,0,0,0,0,0,0};

        public CurrentProjectVO(){}

        public CurrentProjectVO(int id, int employee_id,Date ed, String c,
String p,Date sd,int pr){
                this.id = id;
                this.employee_id = employee_id;
                end_date=ed;
                company=c;
                project=p;
                start_date=sd;
                price=pr;
        }

        public void setID(int id){
                this.id = id;
        }
        public int getID(){
                return id;
        }

        public void setEmployeeID(int employee_id){
                this.employee_id = employee_id;
        }
        public int getEmployeeID(){
                return employee_id;
        }

        public void setPrice(int p) {
                price=p;
        }
        public int getPrice(){
```

```
                return price;
        }

        public void setEndDate(Date ed){
                end_date=ed;;
        }
        public Date getEndDate(){
                return end_date;
        }
        public void setEnd(String ed) {
                if (ed==null || ed.equals("Undetermined")) {
                        GregorianCalendar cl = new
GregorianCalendar(1944,4,4);
                        Date temp = new Date();
                        temp = cl.getTime();
                        end_date=temp;
                }
                else {
                        int d = Integer.parseInt(ed.substring(0,2));
                        int m = Integer.parseInt(ed.substring(3,5))-1;
                        int y = Integer.parseInt(ed.substring(6));
                        GregorianCalendar cl = new GregorianCalendar(y,m,d);
                        Date temp = new Date();
                        temp = cl.getTime();
                        end_date=temp;
                }
        }
        public String getEnd() {
                Calendar cl = Calendar.getInstance();
                cl.setTime(end_date);
                if(cl.get(Calendar.YEAR)==1944)return "Undetermined";
                else {
                        SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
                        return sdf.format(end_date).toString();
                }
        }

        public void setStartDate(Date sd){
                start_date=sd;;
        }
        public Date getStartDate(){
                return start_date;
        }
        public void setStart(String ed) {
                if (ed==null || ed.equals("dd/mm/yyyy")) {
                        GregorianCalendar cl = new
GregorianCalendar(1944,4,4);
                        Date temp = new Date();
                        temp = cl.getTime();
                        start_date=temp;
                }
                else {
                        int d = Integer.parseInt(ed.substring(0,2));
                        int m = Integer.parseInt(ed.substring(3,5))-1;
                        int y = Integer.parseInt(ed.substring(6));
                        GregorianCalendar cl = new GregorianCalendar(y,m,d);
                        Date temp = new Date();
                        temp = cl.getTime();
                        start_date=temp;
                }
        }
        public String getStart() {
                Calendar cl = Calendar.getInstance();
                cl.setTime(start_date);
```

```
                if(cl.get(Calendar.YEAR)==1944)return "dd/mm/yyyy";
                else {
                        SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
                        return sdf.format(start_date).toString();
                }
        }

        public void setCompany(String c){
                if(c==null) company="No Current Project";
                else company=c;
        }
        public String getCompany(){
                return company;
        }

        public void setProject(String p){
                if(p==null) project="No Project!!";
                else project=p;
        }
        public String getProject(){
                if(project.equals("No Project!!")) return "";
                else return project;
        }

        public void setName(String n) {
                name=n;
        }
        public String getName() {
                return name;
        }
        public String getInfo() {
                if(company==null) return "No Project";
                if(project==null) return company;
                return company+" "+project;
        }

        public int[] getMonthview() {
                if (company.equals("No Current Project")) {
                        amount = -1;
                        return monthview;
                }
                else {
                        amount = 0;
                        int s = 0,e = 0;
                        Date now = new Date();
                        SimpleDateFormat sdf = new SimpleDateFormat("MM");
                        SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy");
                        int n = Integer.parseInt(sdf.format(now));
                        if(start_date.compareTo(now)>0) {
                                int sm =
Integer.parseInt(sdf.format(start_date));
                                if(sm<n) s=(sm+12-n);
                                else s=(sm-n);
                        }
                        if (Integer.parseInt(sdf2.format(end_date))==1944)
e=7;
                        else if (end_date.compareTo(now)<0) e=0;
                        else {
                                int em =
Integer.parseInt(sdf.format(end_date));
                                if(em<n) e=(em+13-n);
                                else e=(em+1-n);
                        }
                        for(int i=0;i<7;i++) {
```

```
                                if(s < 1 && e > 0) {
                                        monthview[i] = 1;
                                        e--;
                                        amount++;
                                }
                                else {
                                        monthview[i] = 0;
                                        s--;
                                        e--;
                                        if(e>1) amount++;
                                }
                        }

                        return monthview;
                }
        }


        public int getAmount() {
                return amount;
        }

}
```

EmpSkiCatVO.java

```
package com.iriba.cvgenerator.vo;

import java.util.ArrayList;
import com.iriba.cvgenerator.vo.EmpSkillVO;

/*
 * EmpSkiCatVO
 *
 * On luokka joka sisältää työntekijän taidot tietyssä kategoriassa.
 * Käyttöliittymä tulostaa aina työntekijän taidot kategorioittain.
 */

public class EmpSkiCatVO {

        private String categoryName;
        private int[] skiID;
        private ArrayList<EmpSkillVO> catEmpSkiList = new
ArrayList<EmpSkillVO>();

        public EmpSkiCatVO() {
        }
        public EmpSkiCatVO(String cn, ArrayList<EmpSkillVO> l) {
                categoryName = cn;
                catEmpSkiList = l;
        }

        public String getCategoryName() {
                return categoryName;
        }
        public void setCategoryname(String cn) {
                categoryName = cn;
        }

        public int[] getSkiID() {
                return skiID;
        }
        public void setSkiID(int[] t) {
                skiID=t;
        }
```

```
        public ArrayList<EmpSkillVO> getCatEmpSkiList() {
                return catEmpSkiList;
        }
        public void setCatEmpSkiList(ArrayList<EmpSkillVO> l) {
                catEmpSkiList=l;
        }

        public void addToList(EmpSkillVO esv) {
                catEmpSkiList.add(esv);
        }

}
```

EmpSkillListVo.java

```
package com.iriba.cvgenerator.vo;

import java.util.ArrayList;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.actionbean.EmpManager;
import com.iriba.cvgenerator.to.EmployeeSkillTO;

/*
 * EmpSkillListVO
 *
 * ListVO päättieset luokat luovat käyttöliittymälle tulostettvan
 * listan. Luokat keräävät kaikki tietyn taulun alkiot ja muodostavat
 * näistä listan. Käyttöliittymä käy listaa läpi luodessa taulukoita.
 */

public class EmpSkillListVO {

        private EmpManager em = new EmpManager();
        private String lang;
        private int id;
        private String search="all";


        public ArrayList<EmpSkiCatVO> getEmpSkiList() throws SystemException {
                em.setID(id);
                ArrayList<EmployeeSkillTO> skills=em.getAllEmpSki(id);
                int[] skiID = new int[skills.size()];
                for (int i=0;i<=skills.size()-1;i++) {
                        skiID[i]=skills.get(i).getSkillID();
                }
                ArrayList<EmpSkiCatVO> returnlist =
em.getEmpCatSkiList(skiID,lang);
                return returnlist;
        }

        /*
         *getEmpSkiMissList metodi luo listan taidoista,
         *joita työntekijällä EI ole.
         */

        public ArrayList<EmpSkiCatVO> getEmpSkiMissList() throws
SystemException {
                ArrayList<EmployeeSkillTO> skills=em.getAllEmpSki(id);
                int[] skiID = new int[skills.size()];
                for (int i=0;i<=skills.size()-1;i++) {
                        skiID[i]=skills.get(i).getSkillID();
                }
                return em.getEmpCatSkiMissList(skiID, "en",search);
```

```
        }


        public void setID(int i){
                id = i;
        }
        public int getID(){
                return id;
        }
        public void setSearch(String s) {
                search=s;
        }
        public void setLang(String l) {
                lang=l;
        }
}
```

Liite 4: Tehdas-luokan lähdekoodi

RDBDAOFactory.java

```
package com.iriba.cvgenerator.conn;

import java.sql.*;

import org.apache.log4j.Logger;

import com.iriba.cvgenerator.dao.AnnouncementDAO;
import com.iriba.cvgenerator.dao.CareerDAO;
import com.iriba.cvgenerator.dao.CategoryDAO;
import com.iriba.cvgenerator.dao.CurrentProjectDAO;
import com.iriba.cvgenerator.dao.EducationDAO;
import com.iriba.cvgenerator.dao.EmployeeDAO;
import com.iriba.cvgenerator.dao.EmployeeSkillDAO;
import com.iriba.cvgenerator.dao.LanguageDAO;
import com.iriba.cvgenerator.dao.ProjectDAO;
import com.iriba.cvgenerator.dao.RDBAnnouncementDAO;
import com.iriba.cvgenerator.dao.RDBCareerDAO;
import com.iriba.cvgenerator.dao.RDBCategoryDAO;
import com.iriba.cvgenerator.dao.RDBCurrentProjectDAO;
import com.iriba.cvgenerator.dao.RDBEducationDAO;
import com.iriba.cvgenerator.dao.RDBEmployeeDAO;
import com.iriba.cvgenerator.dao.RDBEmployeeSkillDAO;
import com.iriba.cvgenerator.dao.RDBLanguageDAO;
import com.iriba.cvgenerator.dao.RDBProjectDAO;
import com.iriba.cvgenerator.dao.RDBSkillDAO;
import com.iriba.cvgenerator.dao.RDBTrainingDAO;
import com.iriba.cvgenerator.dao.SkillDAO;
import com.iriba.cvgenerator.dao.TrainingDAO;

/*
 * RDBDAOFactory luo yhteyden tietokantaan.
 */


public class RDBDAOFactory {

        /*
         * Kaikki yhteyden luomiseen tarvittavat tiedot.
         */

  public static final String DRIVER=
    "com.mysql.jdbc.Driver";
  public static final String DBURL=
    "jdbc:mysql://localhost:3306/cvgeneratordb";
  private static Logger logger = Logger.getLogger(RDBDAOFactory.class);

  /*
   * Yhteyden luominen.
   */

  public static Connection createConnection() {
    try{
        Class.forName(DRIVER);
        return DriverManager.getConnection(DBURL,"cvgenerator","cvpass");
    }
    catch(SQLException e){
        logger.error("Connection to database failed!");
        return null;
    }
```

```
    catch(ClassNotFoundException e){
        logger.error("Class not found exception at database connection");
        return null;
    }
  }

  /*
   * Jokaiselle DAO luokalle on oma metodi.
   */

  public static EmployeeDAO getEmployeeDAO() {
    return new RDBEmployeeDAO();
  }

  public static EducationDAO getEducationDAO(){
          return new RDBEducationDAO();
  }

  public static ProjectDAO getProjectDAO() {
          return new RDBProjectDAO();
  }

  public static CareerDAO getCareerDAO(){
          return new RDBCareerDAO();
  }

  public static TrainingDAO getTrainingDAO(){
          return new RDBTrainingDAO();
  }

  public static LanguageDAO getLanguageDAO(){
          return new RDBLanguageDAO();
  }

  public static CategoryDAO getCategoryDAO() {
          return new RDBCategoryDAO();
  }

  public static SkillDAO getSkillDAO() {
          return new RDBSkillDAO();
  }

  public static EmployeeSkillDAO getEmployeeSkillDAO(){
          return new RDBEmployeeSkillDAO();
  }

  public static AnnouncementDAO getAnnouncementDAO(){
          return new RDBAnnouncementDAO();
  }

  public static CurrentProjectDAO getCurrentProjectDAO() {
          return new RDBCurrentProjectDAO();
  }
}
```

Liite 5: Datakontrolliyksikön ja sen rajapinnan lähdekoodia

EmployeeDAO.java

```
package com.iriba.cvgenerator.dao;

import java.util.ArrayList;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.to.EmployeeTO;

/*
 * Rajapinta työntekijä-tietokannan käytölle.
 */

public interface EmployeeDAO {
          public int insertEmployee(EmployeeTO e) throws SystemException;
          public boolean deleteEmployee(int id) throws SystemException;
          public boolean updateEmployee(EmployeeTO e) throws SystemException;
          public ArrayList <EmployeeTO>selectEmployeeTO(EmployeeTO criteria)
throws SystemException;
}
```

RDBEmployeeDAO.java

```
package com.iriba.cvgenerator.dao;

import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Iterator;

import org.apache.log4j.Logger;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.conn.Language;
import com.iriba.cvgenerator.conn.Languages;
import com.iriba.cvgenerator.conn.RDBDAOFactory;
import com.iriba.cvgenerator.to.EmployeeTO;

/*
 * RDB DAO luokat suorittavat tietokantakäskyt
 * annetun yhteyden yli.
 */

public class RDBEmployeeDAO implements EmployeeDAO {

        /*
         * muuttujat
         */

        private String statement;
        private Statement stmt;
        private SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        private EducationDAO eduDAO;
        private EmployeeSkillDAO ESDAO;
        private CareerDAO carDAO;
        private TrainingDAO traDAO;
        private LanguageDAO lanDAO;
        private ProjectDAO proDAO;
        private static Logger logger = Logger.getLogger(RDBEmployeeDAO.class);
```

```
        /*
         * poikkeuksena muihin RDB DAO luokkiin joissain metodeissa
         * RDBEmployeeDAO päivittää myös muita tauluja. (esimerkiksi
         * työntekijää poistaessa on poistettava myös hänen kaikki muut
         * tiedot)
         */

        public RDBEmployeeDAO() {
                carDAO = RDBDAOFactory.getCareerDAO();
                lanDAO = RDBDAOFactory.getLanguageDAO();
                proDAO = RDBDAOFactory.getProjectDAO();
                eduDAO = RDBDAOFactory.getEducationDAO();
                traDAO = RDBDAOFactory.getTrainingDAO();
                ESDAO = RDBDAOFactory.getEmployeeSkillDAO();
        }

        /*
         * insert-metodi lisää uuden taulun tietokantaan.
         *
         * @parametri:
         * TransferObject olio jossa on luotavan taulun teidot.
         *
         * @palautusarvo:
         * palauttaa juuri luodin taulun tunnusluvun.
         */

        public int insertEmployee(EmployeeTO newEmployee) throws
SystemException {

                statement = "INSERT into employee (id, firstname, middlename,
lastname, ";
                statement += "yearofbirth, lastmodified) values
(?,?,?,?,?,?)";

                Connection conn = null;
                try {
                        conn = RDBDAOFactory.createConnection();
                        PreparedStatement pstmt =
conn.prepareStatement(statement);
                        pstmt.setInt(1, newEmployee.getID());
                        pstmt.setString(2, newEmployee.getFirstname());
                        if(newEmployee.getMiddlename()!= null)
                                pstmt.setString(3,
newEmployee.getMiddlename());
                        else pstmt.setString(3, "");
                        pstmt.setString(4, newEmployee.getLastname());
                        pstmt.setInt(5, newEmployee.getYearOfBirth());
                        pstmt.setDate(6, new
java.sql.Date(newEmployee.getLastModified()
                                        .getTime()));

                        pstmt.executeUpdate();

                        logger.info("New Employee created");

                        statement = "SELECT MAX(id) FROM employee;";
                        stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(statement);
                        rs.next();
                        newEmployee.setID(rs.getInt("MAX(id)"));

                        // Employee_text sivutaulut
                        for (Iterator iterator = new
Languages().getAllLanguages()
                                        .iterator(); iterator.hasNext();) {
```

```
                                Language language = (Language)
iterator.next();
                                statement = "INSERT into employee_text
(employee_id, lang, summary) ";
                                statement += "values (" + newEmployee.getID()
+ ",'"
                                                + language.getId() + "',";
                                statement += " '" +
newEmployee.getSummary(language.getId())
                                                + "')";

                                stmt = conn.createStatement();
                                stmt.executeUpdate(statement);

                                logger.info("New Employee_text created");
                        }
                        return newEmployee.getID();
                } catch (SQLException e) {
                        logger.error("Creating new Employee failed!");
                        return -1;
                }
                finally {
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Adding of a
new Employee failed.", e);
                                }
                }

        }

        /*
         * delete-metodi poistaa taulun tietokannasta.
         *
         * @parametri:
         * id = poistettavan taulun työntekijä-tunnusluku
         *
         * @palautusarvo:
         * totuusarvo siitä, onnistuiko poisto.
         *
         * Muilla tauluilla on myös deleteSingle metodi, joka poistaa
         * taulun annetun taulu-tunnusluvun perusteella. Näillä kahdella
         * metodilla on siis mahdollista poistaa kaikki työntekijän tietyt
         * taulut. Vaihtoehtoisesti pelkästään työntekijän joku tietty taulu.
         */

        public boolean deleteEmployee(int id) throws SystemException {

                statement = "DELETE employee, employee_text FROM employee,
employee_text ";
                statement += "WHERE employee.id=" + id
                                + " AND employee_text.employee_ID=" + id +
";";

                Connection conn = null;
                try {
                        conn = RDBDAOFactory.createConnection();
                        stmt = conn.createStatement();
                        stmt.executeUpdate(statement);

                        logger.info("Employee " + id + " removed");

                        eduDAO.deleteEducation(id);
```

```
                        traDAO.deleteTraining(id);
                        lanDAO.deleteLanguage(id);
                        carDAO.deleteCareer(id);
                        ESDAO.deleteEmployeeSkill(id);
                        proDAO.deleteProject(id);

                        conn.close();
                        return true;
                } catch (SQLException e) {
                        logger.error("Employee " + id + " remove failed!");
                        return false;
                }
                finally {
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Deleting
Employee " + id + " failed.", e);
                                }
                }
        }

        /*
         * update metodi päivittää taulun tietoja. Annetun TransferObject:n
         * tunnusluvun perusteella talletetaan muut tiedot.
         *
         * @parametri:
         * TransferObject, jossa on muutettavan taulun tiedot.
         *
         * @palautusarvo:
         * totuusarvo siitä, onnistuiko tietojen muuttaminen vai ei.
         */

        public boolean updateEmployee(EmployeeTO newEmployee) throws
SystemException {

                statement = "UPDATE employee SET id=?, firstname=?,
middlename=?, ";
                statement += "lastname=?, yearofbirth=?, lastmodified=? WHERE
id=?;";

                Connection conn = null;
                try {

                        conn = RDBDAOFactory.createConnection();
                        PreparedStatement pstmt =
conn.prepareStatement(statement);
                        pstmt.setInt(1, newEmployee.getID());
                        pstmt.setString(2, newEmployee.getFirstname());
                        if(newEmployee.getMiddlename()!= null)
                                pstmt.setString(3,
newEmployee.getMiddlename());
                        else pstmt.setString(3, "");
                        pstmt.setString(4, newEmployee.getLastname());
                        pstmt.setInt(5, newEmployee.getYearOfBirth());
                        pstmt.setDate(6, new
java.sql.Date(newEmployee.getLastModified()
                                        .getTime()));
                        pstmt.setInt(7, newEmployee.getID());

                        pstmt.executeUpdate();

                        logger.info("Employee " + newEmployee.getID() +"
updated");
```

```
                        // Employee_text sivutaulut

                        String lang=newEmployee.getEditLanguage();

                        statement = "UPDATE employee_text SET summary=? WHERE
employee_id=? AND lang=?";

                        pstmt = conn.prepareStatement(statement);
                        pstmt.setString(1, newEmployee.getSummary(lang));
                        pstmt.setInt(2, newEmployee.getID());
                        pstmt.setString(3, lang);

                        pstmt.executeUpdate();


                        conn.close();
                        return true;
                } catch (SQLException e) {
                        logger.error("Employee " + newEmployee.getID() +"
update failed!");
                        return false;
                }
                finally {
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Updating
Employee " + newEmployee.getID() + " failed.", e);
                                }
                }
        }

        /*
         * select metodi luo listan kaikista tauluista jotka täyttävät tietyn
kriterian.
         *
         * @parametri:
         * TransferObject-olio. Oliolla on monta eri ominaisuutta, niin
muodostaessa
         * tietokannalle hakulauseketta, metodi katsoo TO-olion kaikki
ominaisuudet läpi.
         * Metodin saadessa tyhjä olio, palautusarvona saa kaikki taulun
alkiot.
         *
         * @palautusarvo:
         * TO-olio lista kaikista hakua täsmäävistä tauluista.
         */

        public ArrayList<EmployeeTO> selectEmployeeTO(EmployeeTO criteria)
throws SystemException {

                statement = "SELECT * FROM employee ";
                StringBuffer sb = new StringBuffer();

                if (criteria.getFirstname() != null)
                        sb.append("firstname LIKE '" + criteria.getFirstname()
+ "' AND ");
                if (criteria.getMiddlename() != null)
                        sb
                                        .append("middlename LIKE '" +
criteria.getMiddlename()
                                                        + "' AND ");
                if (criteria.getLastname() != null)
```

```
                        sb.append("lastname LIKE '" + criteria.getLastname() +
"' AND ");
                if (criteria.getID() != 0)
                        sb.append("id LIKE " + criteria.getID() + " AND ");
                if (criteria.getYearOfBirth() != 0)
                        sb.append("yearofbirth LIKE '"
                                        +
sdf.format(criteria.getYearOfBirth()) + "' AND ");
                if (criteria.getLastModified() != null)
                        sb.append("lastmodified LIKE '" +
criteria.getLastModified()
                                        + "' AND ");

                if (sb.length() > 0) {
                        statement += "WHERE ";
                        statement += sb.toString().substring(0,
sb.toString().length() - 4);
                }
                statement += " ORDER BY lastname";

                Connection conn = null;
                try {
                        ArrayList<EmployeeTO> empList = new
ArrayList<EmployeeTO>();

                        conn = RDBDAOFactory.createConnection();
                        stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(statement);

                        while (rs.next()) {
                                EmployeeTO temp = new EmployeeTO();
                                temp.setFirstname(rs.getString("firstname"));
                                temp.setMiddlename(rs.getString("middlename"))
;
                                temp.setLastname(rs.getString("lastname"));
                                temp.setYearOfBirth((rs.getInt("yearofbirth"))
);
                                temp.setID(rs.getInt("id"));
                                temp.setLastModified(rs.getDate("lastmodified"
));
                                for (Iterator iterator = new
Languages().getAllLanguages()
                                                .iterator();
iterator.hasNext();) {
                                        Language language = (Language)
iterator.next();
                                        temp.setSummary(language.getId(), "");
                                }

                                empList.add(temp);
                        }
                        // Employee_text sivutaulut
                        for (int i = 0; i <= empList.size() - 1; i++) {
                                for (Iterator iterator = new
Languages().getAllLanguages()
                                                .iterator();
iterator.hasNext();) {
                                        Language language = (Language)
iterator.next();

                                        statement = "SELECT * FROM
employee_text WHERE employee_id LIKE ";
                                        statement += empList.get(i).getID() +
" AND lang LIKE '"
```

```
                                                        + language.getId() +
"';";

                                        stmt = conn.createStatement();
                                        rs = stmt.executeQuery(statement);

                                        while (rs.next()) {
                                                empList.get(i).setSummary(lang
uage.getId(),
                                                                        rs.getString("
summary"));
                                        }
                                }
                        }

                        conn.close();
                        return empList;
                } catch (SQLException e) {
                        logger.error("Select Employee - search failed!");
                        return null;
                }
                finally {
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Employee
search failed.", e);
                                }
                }
        }

}
```

RDBProjectDAO.java

```
package com.iriba.cvgenerator.dao;

import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Iterator;

import org.apache.log4j.Logger;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.conn.Language;
import com.iriba.cvgenerator.conn.Languages;
import com.iriba.cvgenerator.conn.RDBDAOFactory;
import com.iriba.cvgenerator.to.ProjectTO;

/*
 * RDBProjectDAO käsittelee projekti-tietokantaa.
 * metodien tarkemmat määritykset tiedostossa RDBEmployeeDAO
 */

public class RDBProjectDAO implements ProjectDAO {

        private String statement;
        private Statement stmt;
        private SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        private Connection conn;
        private static Logger logger = Logger.getLogger(RDBProjectDAO.class);

        public RDBProjectDAO() {
```

```
	}

	public int insertProject(ProjectTO newProject) throws SystemException
{
		statement = "INSERT into project (employee_id, id, start_date,
end_date, ";
		statement += " platform, software, customer) values
(?,?,?,?,?,?,?)";

		try{
			conn = RDBDAOFactory.createConnection();
			PreparedStatement pstmt =
conn.prepareStatement(statement);
			pstmt.setInt(1, newProject.getEmployeeID());
			pstmt.setInt(2, newProject.getID());
			pstmt.setDate(3, new
java.sql.Date(newProject.getStartDate().getTime()));
			pstmt.setDate(4, new
java.sql.Date(newProject.getEndDate().getTime()));
			pstmt.setString(5, newProject.getPlatform());
			pstmt.setString(6, newProject.getSoftware());
			pstmt.setString(7, newProject.getCustomer());

			pstmt.executeUpdate();

			logger.info("New Project created");

			statement = "SELECT MAX(id) FROM project;";
			stmt =
RDBDAOFactory.createConnection().createStatement();
			ResultSet rs = stmt.executeQuery(statement);
			rs.next();
			newProject.setID(rs.getInt("MAX(id)"));

			// Project_text sivutaulut
			for (Iterator iterator = new
Languages().getAllLanguages()
					.iterator(); iterator.hasNext();) {
				Language language = (Language)
iterator.next();
				statement = "INSERT into project_text
(project_id, lang, name, description, own_role) ";
				statement += "values ("+newProject.getID()
+",'"+language.getId()+"',";
				statement += "
'"+newProject.getName(language.getId())+"',";
				statement += "
'"+newProject.getDescription(language.getId())+"',";
				statement += "
'"+newProject.getOwnRole(language.getId())+"')";

				stmt = conn.createStatement();
				stmt.executeUpdate(statement);

				System.out.println("NEW PROJECT_TEXT CREATED
IN LANG: "+language.getId());
				System.out.println("WITH NAME: " +
newProject.getName(language.getId()));
				System.out.println("WITH DESCRIPTION: " +
newProject.getDescription(language.getId()));
				System.out.println("WITH OWN ROLE: " +
newProject.getOwnRole(language.getId()));
```

```
                        }

                        // Project_responsibility sivutaulut
                        for (Iterator iterator = new
Languages().getAllLanguages()
                                        .iterator(); iterator.hasNext();) {
                                Language language = (Language)
iterator.next();
                                statement = "INSERT into
project_responsibility (project_id, lang, responsibility) ";
                                statement += "values ("+newProject.getID()
+",'"+language.getId()+"',";
                                statement += "
'"+newProject.getResponsibility(language.getId())+"')";

                                stmt = conn.createStatement();
                                stmt.executeUpdate(statement);

                                logger.info("New Project_text created");
                        }
                        conn.close();
                        return newProject.getID();
                }
                catch(SQLException e){
                        logger.error("Creating new Project failed!");
                        return -1;
                }
                finally {
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Adding of a
new Project failed.", e);
                                }
                }
        }

        public boolean deleteProject(int id) throws SystemException {

                statement = "DELETE project, project_text,
project_responsibility ";
                statement += "FROM project, project_text,
project_responsibility ";
                statement += "WHERE project.employee_id ="+id+" AND
project_text.project_id=project.id AND project_responsibility.project_id
=project.id";

                try{
                        conn = RDBDAOFactory.createConnection();
                        stmt = conn.createStatement();
                        stmt.executeUpdate(statement);

                        logger.info("All Projects with id: " + id + "
removed");

                        conn.close();
                        return true;
                }
                catch(SQLException e){
                        logger.error("All Projects with id: " + id + " remove
failed!");
                        return false;
                }
                finally {
```

```
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Deleting
all Projects with Employee id " + id + " failed.", e);
                                }
                }
        }
        public boolean deleteSingleProject(int id) throws SystemException {

                        statement = "DELETE project, project_text,
project_responsibility ";
                        statement += "FROM project, project_text,
project_responsibility ";
                        statement += "WHERE project.id ="+id+" AND
project_text.project_id="+id+" AND project_responsibility.project_id ="+id+"";

                        try{
                                conn = RDBDAOFactory.createConnection();
                                stmt = conn.createStatement();
                                stmt.executeUpdate(statement);

                                logger.info("Project " + id + " removed");

                                conn.close();
                                return true;
                        }
                        catch(SQLException e){
                                logger.error("Project " + id + " remove
failed!");
                                return false;
                        }
                        finally {
                                if (conn != null)
                                        try {
                                                conn.close();
                                        } catch (SQLException e) {
                                                throw new
SystemException("Deleting Project " + id + " failed.", e);
                                        }
                        }
                }

        public boolean updateProject(ProjectTO newProject) throws
SystemException {

                statement = "UPDATE project SET employee_id=?, id=?,
start_date=?, end_date=?, platform=?, ";
                statement += "software=?, customer=? WHERE id=?;";

                try{

                        conn = RDBDAOFactory.createConnection();
                        PreparedStatement pstmt =
conn.prepareStatement(statement);
                        pstmt.setInt(1, newProject.getEmployeeID());
                        pstmt.setInt(2, newProject.getID());
                        pstmt.setDate(3, new
java.sql.Date(newProject.getStartDate().getTime()));
                        pstmt.setDate(4, new
java.sql.Date(newProject.getEndDate().getTime()));
                        pstmt.setString(5, newProject.getPlatform());
                        pstmt.setString(6, newProject.getSoftware());
                        pstmt.setString(7, newProject.getCustomer());
```

```
                        pstmt.setInt(8, newProject.getID());

                        pstmt.executeUpdate();

                        logger.info("Project " + newProject.getID() +"
updated");

                        // Project_text sivutaulut
                        String lang=newProject.getEditLanguage();

                        statement = "UPDATE project_text SET name=?,
description=?, own_role=?" +
                                                " WHERE project_id=? AND
lang=?";


                        pstmt = conn.prepareStatement(statement);
                        pstmt.setString(1, newProject.getName(lang));
                        pstmt.setString(2, newProject.getDescription(lang));
                        pstmt.setString(3, newProject.getOwnRole(lang));
                        pstmt.setInt(4, newProject.getID());
                        pstmt.setString(5, lang);

                        pstmt.executeUpdate();


                        statement = "UPDATE project_responsibility SET
responsibility=?" +
                        " WHERE project_id=? AND lang=?";

                        pstmt = conn.prepareStatement(statement);
                        pstmt.setString(1,
newProject.getResponsibility(lang));
                        pstmt.setInt(2, newProject.getID());
                        pstmt.setString(3, lang);


                        pstmt.executeUpdate();

                        conn.close();
                        return true;
                        }

                        catch(SQLException e){
                                logger.error("Project " + newProject.getID()
+" update failed!");
                                return false;
                        }
                        finally {
                                if (conn != null)
                                        try {
                                                conn.close();
                                        } catch (SQLException e) {
                                                throw new
SystemException("Updating Project " + newProject.getID() + " failed.", e);
                                        }
                        }
        }


        public ArrayList <ProjectTO>selectProjectTO(ProjectTO criteria) throws
SystemException {
```

```
                statement = "SELECT * FROM project ";
                StringBuffer sb = new StringBuffer();

                if(criteria.getEmployeeID() != 0)
                        sb.append("employee_id LIKE "+criteria.getEmployeeID()
+" AND ");
                if(criteria.getID() != 0)
                        sb.append("id LIKE "+criteria.getID()+" AND ");
                if(criteria.getStartDate() != null)
                        sb.append("start_date LIKE
'"+sdf.format(criteria.getStartDate())+"' AND ");
                if(criteria.getEndDate() != null)
                        sb.append("end_date LIKE
'"+sdf.format(criteria.getEndDate())+"' AND ");
                if(criteria.getPlatform() != null)
                        sb.append("platform LIKE '"+criteria.getPlatform()+"'
AND ");
                if(criteria.getSoftware() != null)
                        sb.append("software LIKE '"+criteria.getSoftware()+"'
AND ");
                if(criteria.getCustomer() != null)
                        sb.append("customer LIKE '"+criteria.getCustomer()+"'
AND ");


                if(sb.length() > 0) {
                        statement += "WHERE ";
                        statement += sb.toString().substring(0,
sb.toString().length()-4);
                }
                statement += " ORDER BY start_date DESC";
                try{
                        ArrayList <ProjectTO>proList = new ArrayList
<ProjectTO>();

                        conn = RDBDAOFactory.createConnection();
                        stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(statement);

                        while(rs.next()){
                                ProjectTO temp = new ProjectTO();
                                temp.setEmployeeID(rs.getInt("employee_id"));
                                temp.setID(rs.getInt("id"));
                                temp.setStartDate(rs.getDate("start_date"));
                                temp.setEndDate((rs.getDate("end_date")));
                                temp.setPlatform(rs.getString("platform"));
                                temp.setSoftware(rs.getString("software"));
                                temp.setCustomer(rs.getString("customer"));

                                for (Iterator iterator = new
Languages().getAllLanguages()
                                                .iterator();
iterator.hasNext();) {
                                        Language language = (Language)
iterator.next();
                                        temp.setName(language.getId(), "");
                                        temp.setDescription(language.getId(),
"");
                                        temp.setOwnRole(language.getId(), "");
                                        temp.setResponsibility(language.getId(
), "");

                                }
```

```
                                proList.add(temp);
                        }
                        // Project_text sivutaulut
                        for(int i = 0; i <= proList.size()-1; i++){
                                for (Iterator iterator = new
Languages().getAllLanguages()
                                                .iterator();
iterator.hasNext();) {
                                        Language language = (Language)
iterator.next();

                                        statement = "SELECT * FROM
project_text WHERE project_id LIKE ";
                                        statement += proList.get(i).getID()+"
AND lang LIKE '" + language.getId()+"';";

                                        stmt = conn.createStatement();
                                        rs = stmt.executeQuery(statement);

                                        while(rs.next()){
                                                proList.get(i).setName(languag
e.getId(), rs.getString("name"));
                                                proList.get(i).setDescription(
language.getId(), rs.getString("description"));
                                                proList.get(i).setOwnRole(lang
uage.getId(), rs.getString("own_role"));

                                        }

                                        statement ="SELECT * FROM
project_responsibility WHERE project_id LIKE ";
                                        statement += proList.get(i).getID()+"
AND lang LIKE '" + language.getId()+"';";

                                        rs = stmt.executeQuery(statement);

                                        while(rs.next()) {
                                                proList.get(i).setResponsibili
ty(language.getId(), rs.getString("responsibility"));
                                        }
                                }
                        }

                        conn.close();
                        return proList;
                }
                catch(SQLException e){
                        logger.error("Select Project - search failed!");
                        return null;
                }
                finally {
                        if (conn != null)
                                try {
                                        conn.close();
                                } catch (SQLException e) {
                                        throw new SystemException("Project
search failed.", e);
                                }
                }}}
```

Liite 6: XML-generaattorin lähdekoodi

XMLGenerator.java

```
package com.iriba.cvgenerator.pdf;

import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import org.jdom.*;
import org.jdom.output.DOMOutputter;
import org.jdom.output.XMLOutputter;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.vo.*;
import com.iriba.cvgenerator.actionbean.EmpManager;


public class XMLGenerator {

        private EmployeeVO emp;
        private ArrayList <EducationVO> eduList;
        private ArrayList <CareerVO> carList;
        private ArrayList <ProjectVO> proList;
        private ArrayList <LanguageVO> lanList;
        private ArrayList <EmpSkiCatVO> escList;
        private ArrayList <TrainingVO> traList;
        private EmpManager em = new EmpManager();



        public String GenerateXML(int empID, String lang) throws
SystemException {

                EducationList edList = new EducationList();
                CareerListVO caList = new CareerListVO();
                ProjectListVO prList = new ProjectListVO();
                LanguageListVO laList = new LanguageListVO();
                EmpSkillListVO esList = new EmpSkillListVO();
                TrainingListVO trList = new TrainingListVO();

                emp = EmployeeUtil.convertToVO(em.getCurrentEmployee(empID),
lang);
                edList.setID(empID);
                caList.setID(empID);
                prList.setID(empID);
                laList.setID(empID);
                esList.setID(empID);
                trList.setID(empID);
                if(edList.getEduList() != null) {
                        edList.setLang(lang);
                        eduList = edList.getEduList();
                }
                if(caList.getCarList() != null){
                        caList.setLang(lang);
                        carList = caList.getCarList();
                }
                if(prList.getProList() != null){
                        prList.setLang(lang);
                        proList = prList.getProList();
                }
                if(laList.getLanList() != null) {
                        laList.setLang(lang);
```

```
                        lanList = laList.getLanList();
                }
                if(esList.getEmpSkiList() != null) {
                        esList.setLang(lang);
                        escList = esList.getEmpSkiList();
                }
                if(trList.getTraList() != null){
                        trList.setLang(lang);
                        traList = trList.getTraList();
                }

                Calendar cl = Calendar.getInstance();
                Calendar cl2 = Calendar.getInstance();

                Element root = new Element("data");

                // Name
                if(emp.getFirstname() != null && emp.getLastname() != null){
                        Element name = new Element("name");
                    Element name2 = new Element("name");

                    name.addContent(name2);
                    name2.addContent(emp.getFirstname()+"
"+emp.getLastname());
                        root.addContent(name);

                }

            // Backgrounds
                if(!eduList.isEmpty() && emp.getYearOfBirth() != 0){
                    Element bgs = new Element("backgrounds");
                    Element bgheader = new Element("header");
                            if(lang.equals("en"))
bgheader.addContent("Backgrounds");
                            else if(lang.equals("fi"))
bgheader.addContent("Taustatiedot");
                            bgs.addContent(bgheader);

                        if(emp.getYearOfBirth() != 0){
                            Element dob = new Element("dateofbirth");
                                dob.addContent(""+emp.getYearOfBirth());
                                bgs.addContent(dob);
                        }

                        if(!eduList.isEmpty()){
                                for(int i=0; i<=eduList.size()-1;i++){
                                        Element edu = new
Element("education");
                                        Element deg = new Element("degree");
                                                deg.addContent(eduList.get(i).
getDegree()+", ");
                                        Element sch = new Element("school");
                                                sch.addContent(eduList.get(i).
getSchool()+", ");
                                        Element etime = new Element("time");
                                        etime.addContent(eduList.get(i).getYea
rStart()+" - "+eduList.get(i).getYearEnd());

                                        String majo = "Major: ";
                                        if(lang.equals("fi")) majo = "Pääaine:
";
                                        Element maj = new Element("major");
                                                maj.addContent(majo +
eduList.get(i).getMajor() + ", ");
```

```
                                        edu.addContent(deg);
                                        edu.addContent(sch);
                                        edu.addContent(maj);
                                        edu.addContent(etime);
                                        bgs.addContent(edu);
                                }
                        }
                        root.addContent(bgs);

                }


                // Summary
                if(emp.getSummary() != null){
                        Element sum = new Element("summary");
                        Element sumheader = new Element("header");
                                if(lang.equals("en"))
sumheader.addContent("Summary");
                             else if(lang.equals("fi"))
sumheader.addContent("Tiivistelmä");
                        Element sumtxt = new Element("summarytext");
                                sumtxt.addContent(emp.getSummary());

                        sum.addContent(sumheader);
                        sum.addContent(sumtxt);
                        root.addContent(sum);
                }
                // Skill summary
                if(!escList.isEmpty()){
                        Element ssum = new Element("skillsummary");
                        Element ssheader = new Element("header");
                                if(lang.equals("en"))
ssheader.addContent("Skill summary");
                             else if(lang.equals("fi"))
ssheader.addContent("Taidot");
                                ssum.addContent(ssheader);

                        for(int i=0; i<=escList.size()-1; i++){
                                Element cat = new Element("category");
                                Element catn = new Element("categoryname");
                                        catn.addContent(escList.get(i).getCate
goryName());
                                        ArrayList <EmpSkillVO> tempList =
escList.get(i).getCatEmpSkiList();
                                for(int j=0; j<=tempList.size()-1;j++){
                                        Element ski = new Element("skill");
                                        Element skin = new
Element("skillname");
                                                skin.addContent(tempList.get(j
).getSkillName());
                                        Element slev = new Element("level");
                                                slev.addContent(tempList.get(j
).getLevelName());
                                                ski.addContent(skin);
                                                ski.addContent(slev);

                                                cat.addContent(ski);
                                }

                                        cat.addContent(catn);
                                        ssum.addContent(cat);
                        }
                        root.addContent(ssum);
                }
                // Project experience
```

```
                if(!proList.isEmpty()){
                        Element proe = new Element("projectexperience");
                        Element proh = new Element("header");
                                if(lang.equals("en")) proh.addContent("Project
experience");
                            else if(lang.equals("fi"))
proh.addContent("Projektit");
                                proe.addContent(proh);
                        for(int i=0; i<=proList.size()-1;i++){
                                Element pro = new Element("project");
                                Element pron = new Element("projectname");
                                        pron.addContent(proList.get(i).getName
());
                                Element cus = new Element("customer");
                                        cus.addContent(proList.get(i).getCusto
mer());
                                Element ptime = new Element("time");
                                        cl.setTime(proList.get(i).getStartDate
());
                                        cl2.setTime(proList.get(i).getEndDate(
));
                                        String endDate;
                                        if(proList.get(i).getEndDate() ==
null){
                                                endDate = "Present";
                                        }
                                        else {
                                                cl2.setTime(proList.get(i).get
EndDate());
                                                endDate =
cl2.get(Calendar.MONTH)+"/"+cl2.get(Calendar.YEAR);
                                        }
                                        ptime.addContent(cl.get(Calendar.MONTH
)+"/"+cl.get(Calendar.YEAR)+" - " + endDate);
                                Element des = new Element("description");
                                        des.addContent(proList.get(i).getDescr
iption());
                                Element pla = new Element("platform");
                                        pla.addContent(proList.get(i).getDescr
iption());
                                Element sof = new Element("software");
                                        sof.addContent(proList.get(i).getSoftw
are());
                                Element own = new Element("ownrole");
                                        own.addContent(proList.get(i).getOwnro
le());
                                ProjectVO temp = proList.get(i);
                                        for(int j=0;
j<=temp.getReslities().size()-1;j++){
                                                Element pres = new
Element("responsibility");
                                                pres.addContent(temp.getReslit
ies().get(j));
                                                pro.addContent(pres);
                                        }
                                        pro.addContent(pron);
                                        pro.addContent(cus);
                                        pro.addContent(ptime);
                                        pro.addContent(des);
                                        pro.addContent(pla);
                                        pro.addContent(sof);
                                        pro.addContent(own);

                                        proe.addContent(pro);
```

```
                        }
                        root.addContent(proe);
                }

                // Career history
                if(!carList.isEmpty()){
                        Element ch = new Element("careerhistory");
                        Element carh = new Element("header");
                                if(lang.equals("en")) carh.addContent("Career
history");
                            else if(lang.equals("fi"))
carh.addContent("Urahistoria");
                                ch.addContent(carh);
                        for(int i=0; i<=carList.size()-1;i++){
                                Element car = new Element("career");
                                Element comn = new Element("companyname");
                                        comn.addContent(carList.get(i).getComp
any());
                                Element ctim = new Element("time");
                                        cl.setTime(carList.get(i).getStartDate
());

                                        String endDate;
                                        if(carList.get(i).getEndDate() ==
null){
                                                endDate = "Present";
                                        }
                                        else {
                                                cl2.setTime(carList.get(i).get
EndDate());
                                                endDate =
cl2.get(Calendar.MONTH)+"/"+cl2.get(Calendar.YEAR);
                                        }
                                        ctim.addContent(cl.get(Calendar.MONTH)
+"/"+cl.get(Calendar.YEAR)+" - " + endDate);
                                Element jobt = new Element("jobtitle");
                                        jobt.addContent(carList.get(i).getJobT
itle());
                                        CareerVO temp = carList.get(i);
                                for(int j=0; j<=temp.getReslities().size()-
1;j++){
                                        Element cres = new
Element("responsibility");
                                        cres.addContent(temp.getReslities().ge
t(j));
                                        car.addContent(cres);
                                }
                                car.addContent(comn);
                                car.addContent(ctim);
                                car.addContent(jobt);


                                ch.addContent(car);
                        }
                        root.addContent(ch);
                }

                // Trainings
                if(!traList.isEmpty()){
                        Element tras = new Element("trainings");
                        Element trah = new Element("header");
                                if(lang.equals("en"))
trah.addContent("Training");
                            else if(lang.equals("fi"))
trah.addContent("Kurssit");
```

```
                        tras.addContent(trah);
                    for(int i=0; i<=traList.size()-1;i++){
                        Element tra = new Element("training");
                        Element tyea = new Element("year");
                            tyea.addContent(""+traList.get(i).getY
ears());
                        Element tdes = new Element("description");
                            tdes.addContent(traList.get(i).getDesc
ription());

                    tra.addContent(tyea);
                    tra.addContent(tdes);

                    tras.addContent(tra);
                    }
                    root.addContent(tras);
                }

                // Language skills
                if(!lanList.isEmpty()){
                        Element lans = new Element("languageskills");
                        Element lanh = new Element("header");
                            if(lang.equals("en"))
lanh.addContent("Language skills");
                                else if(lang.equals("fi"))
lanh.addContent("Kielitaito");
                            lans.addContent(lanh);
                            for(int i=0; i<=lanList.size()-1;i++){
                    Element lan = new Element("language");
                    Element lnam = new Element("name");
                            lnam.addContent(lanList.get(i).getName());
                    Element llev = new Element("level");
                            llev.addContent(lanList.get(i).getLanguageLeve
l(lang));
                    lan.addContent(lnam);
                    lan.addContent(llev);
                    lans.addContent(lan);
                    }
                            root.addContent(lans);
                }
                // Root adds

            Document doc = new Document(root);

            try {
                FileOutputStream out = new
FileOutputStream("EmployeeXML.xml");
                XMLOutputter serializer = new XMLOutputter();
                serializer.output(doc, out);
                out.flush();
                out.close();
              }
              catch (IOException e) {
                System.err.println(e);
              }

              DOMOutputter domOut = new DOMOutputter();

              // org.jdom.Documet -> org.w3c.Document
              try {
                PDFGenerator gen = new PDFGenerator();
                String pdfPath = "/"+emp.getFirstname()+"_"+emp.getLastname()
+"-CV.PDF";
              return gen.GeneratePDF(domOut.output(doc), pdfPath);
                } catch (JDOMException e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                return "fail";
        }
}
```

Liite 7: XSL-tyylitiedoston lähdekoodi

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">

 <xsl:template match="/">

 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

   <fo:layout-master-set>
     <fo:simple-page-master master-name="simple"
                  page-height="29.7cm"
                  page-width="21cm"
                  margin-top="1cm"
                  margin-bottom="2cm"
                  margin-left="2.5cm"
                  margin-right="1.5cm">
       <fo:region-body margin-top="0cm"/>
       <fo:region-before extent="3cm"/>
       <fo:region-after extent="1.5cm"/>
     </fo:simple-page-master>
   </fo:layout-master-set>


   <fo:page-sequence master-reference="simple">
        <fo:flow flow-name="xsl-region-body">
                 <xsl:apply-templates select="data"/>
        </fo:flow>
   </fo:page-sequence>

  </fo:root>
 </xsl:template>

<!-- DATA-ALUE -->
<xsl:template match="name">
        <fo:block font-style="italic">Curriculum Vitae</fo:block>
       <fo:block font-size="24pt"
                 font-family="sans-serif"
                 line-height="24pt"
                     font-weight="bold"
                     margin-top="15px">
                <xsl:value-of select="name"/>

       </fo:block>
<fo:block> </fo:block>
</xsl:template>

 <xsl:template match="backgrounds">
       <fo:block>
                <xsl:apply-templates select="header"/>

                <fo:table border-collapse="separate" height="auto" border-
color="black" border-style="solid" border-width="0pt" table-layout="fixed"
width="100%">

      <fo:table-column column-width="20%"/>
        <fo:table-column column-width="80%"/>

      <fo:table-body>
        <fo:table-row>
```

```
          <fo:table-cell padding="1mm"><fo:block>Year of
birth</fo:block></fo:table-cell>
          <fo:table-cell padding="1mm"><fo:block><xsl:value-of
select="dateofbirth"/></fo:block></fo:table-cell>
        </fo:table-row>


        <fo:table-row>
          <fo:table-cell
padding="1mm"><fo:block>Education</fo:block></fo:table-cell>

<fo:table-cell padding="1mm">

<xsl:for-each select="education">
<fo:block><xsl:value-of select="."/></fo:block>
</xsl:for-each>
            </fo:table-cell>

        </fo:table-row>


      </fo:table-body>
</fo:table>



       </fo:block>
 </xsl:template>

 <xsl:template match="summary">
       <fo:block white-space-collpase = "false">
        <xsl:apply-templates select="header"/>
        <xsl:apply-templates select="summarytext" disable-output-
escaping="yes"/>
       </fo:block>
 </xsl:template>

<xsl:template match="skillsummary">
        <fo:block>
        <xsl:apply-templates select="header"/>
        <fo:table border-collapse="separate" height="auto" border-
color="black" border-style="solid" border-width="0pt" table-layout="fixed"
width="100%">

      <fo:table-column column-width="50%"/>
        <fo:table-column column-width="50%"/>

        <fo:table-header>
         <fo:table-row>
          <fo:table-cell>
            <fo:block font-weight="bold">Tools, techniques and
products</fo:block>
           </fo:table-cell>
           <fo:table-cell>
             <fo:block font-weight="bold">Experience</fo:block>
           </fo:table-cell>
         </fo:table-row>
        </fo:table-header>

      <fo:table-body>
<xsl:for-each select="category">
        <fo:table-row keep-together.within-page="always">
          <fo:table-cell padding-top="5mm" padding-bottom="3mm" font-
size="14pt"><fo:block><xsl:value-of
select="categoryname"/></fo:block></fo:table-cell>
```

```
        </fo:table-row>
<xsl:for-each select="skill">
        <fo:table-row>
          <fo:table-cell><fo:block><xsl:value-of
select="skillname"/></fo:block></fo:table-cell>
          <fo:table-cell><fo:block><xsl:value-of select="level"/></fo:block></
fo:table-cell>
        </fo:table-row>
</xsl:for-each>
</xsl:for-each>
      </fo:table-body>
</fo:table>


        </fo:block>

</xsl:template>

<xsl:template match="projectexperience">
        <fo:block>
                <xsl:apply-templates select="header"/>

<fo:table border-collapse="collapse" height="auto" border-color="black"
border-style="solid" border-width="1pt" table-layout="fixed" width="100%">

      <fo:table-column column-width="40%"/>
        <fo:table-column column-width="60%"/>



      <fo:table-body>
<xsl:for-each select="project">
        <fo:table-row border-color="black" border-style="solid" border-
width="1pt" keep-together.within-page="always">
          <fo:table-cell padding="1mm">
<fo:block font-weight="bold">
Project Name:
</fo:block>
<fo:block>
<xsl:value-of select="projectname"/>
</fo:block>
<fo:block font-weight="bold">
Customer:
</fo:block>
<fo:block>
<xsl:value-of select="customer"/>
</fo:block>
<fo:block font-weight="bold">
Time involved:
</fo:block>
<fo:block>
<xsl:value-of select="time"/>

</fo:block>

</fo:table-cell>
          <fo:table-cell padding="1mm">
<fo:block font-weight="bold">
Description of project:
</fo:block>
<fo:block>
<xsl:value-of select="description"/>
</fo:block>
<fo:block font-weight="bold">
Platform:
```

```
</fo:block>
<fo:block>
<xsl:value-of select="platform"/>
</fo:block>
<fo:block font-weight="bold">
Software:
</fo:block>
<fo:block>
<xsl:value-of select="software"/>
</fo:block>
<fo:block font-weight="bold">
Own role:
</fo:block>
<fo:block>
<xsl:value-of select="ownrole"/>
</fo:block>
<fo:block font-weight="bold">
Responsibilities:
</fo:block>
<fo:block>
<xsl:for-each select="responsibility">
<fo:list-block>

<fo:list-item>
 <fo:list-item-label>
   <fo:block></fo:block>
 </fo:list-item-label>
 <fo:list-item-body>
   <fo:block>- <xsl:value-of select="."/></fo:block>
 </fo:list-item-body>
</fo:list-item>


</fo:list-block>
</xsl:for-each>
</fo:block>

</fo:table-cell>
        </fo:table-row>
</xsl:for-each>
      </fo:table-body>
</fo:table>

        </fo:block>
</xsl:template>

<xsl:template match="careerhistory">
        <fo:block>
                <xsl:apply-templates select="header"/>
<fo:table border-collapse="collapse" height="auto" border-color="black"
border-style="solid" border-width="1pt" table-layout="fixed" width="100%">

      <fo:table-column column-width="40%"/>
        <fo:table-column column-width="60%"/>



      <fo:table-body>
<xsl:for-each select="career">
        <fo:table-row border-color="black" border-style="solid" border-
width="1pt" keep-together.within-page="always">
          <fo:table-cell padding="1mm">
<fo:block font-weight="bold">
<xsl:value-of select="companyname"/>
</fo:block>
```

```
<fo:block>
<xsl:value-of select="time"/>
</fo:block>

</fo:table-cell>
          <fo:table-cell padding="1mm">
<fo:block font-weight="bold">
Job title:
</fo:block>
<fo:block>
<xsl:value-of select="jobtitle"/>
</fo:block>
<fo:block font-weight="bold">
Main responsibilities:
</fo:block>
<fo:block>
<xsl:for-each select="responsibility">
<fo:list-block>

<fo:list-item>
 <fo:list-item-label>
   <fo:block></fo:block>
 </fo:list-item-label>
 <fo:list-item-body>
   <fo:block>- <xsl:value-of select="."/></fo:block>
 </fo:list-item-body>
</fo:list-item>



</fo:list-block>
</xsl:for-each>
</fo:block>

</fo:table-cell>
        </fo:table-row>
</xsl:for-each>
      </fo:table-body>
</fo:table>

        </fo:block>
</xsl:template>

<xsl:template match="trainings">
        <fo:block>
                <xsl:apply-templates select="header"/>

                <fo:table border-collapse="separate" height="auto" border-
color="black" border-style="solid" border-width="0pt" table-layout="fixed"
width="100%">

      <fo:table-column column-width="20%"/>
        <fo:table-column column-width="80%"/>

      <fo:table-body>
<xsl:for-each select="training">
        <fo:table-row keep-together.within-page="always">
          <fo:table-cell padding="1mm"><fo:block><xsl:value-of
select="year"/></fo:block></fo:table-cell>
          <fo:table-cell padding="1mm"><fo:block><xsl:value-of
select="description"/></fo:block></fo:table-cell>
        </fo:table-row>
</xsl:for-each>
      </fo:table-body>
</fo:table>
```

```
        </fo:block>
</xsl:template>

<xsl:template match="languageskills">
        <fo:block>
                <xsl:apply-templates select="header"/>

                <fo:table border-collapse="separate" height="auto" border-
color="black" border-style="solid" border-width="0pt" table-layout="fixed"
width="100%">

      <fo:table-column column-width="20%"/>
        <fo:table-column column-width="80%"/>

      <fo:table-body>
<xsl:for-each select="language">
        <fo:table-row keep-together.within-page="always">
          <fo:table-cell padding="1mm"><fo:block><xsl:value-of
select="name"/></fo:block></fo:table-cell>
          <fo:table-cell padding="1mm"><fo:block><xsl:value-of select="level"/
></fo:block></fo:table-cell>
        </fo:table-row>
</xsl:for-each>
      </fo:table-body>
</fo:table>
        </fo:block>


</xsl:template>


<!-- TEMPLATET -->
<!-- Headerit -->
 <xsl:template match="header">
       <fo:block font-size="18pt"
             font-family="sans-serif"
             line-height="24pt"
             space-after.optimum="15pt"
             background-color="lightgray"
             color="black"
             text-align="left"
                 margin-top="20px">
         <xsl:value-of select="."/>
       </fo:block>
 </xsl:template>

<xsl:template match="data">
    <fo:block font-size="11pt" font-family="sans-serif" color="black">

    <xsl:apply-templates/>
    </fo:block>
</xsl:template>

 </xsl:stylesheet>
```

Liite 8: PDF-generaattorin lähdekoodi

```
package com.iriba.cvgenerator.pdf;

import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.*;
import javax.xml.transform.sax.*;

import org.apache.fop.apps.*;
import org.w3c.dom.Document;

/*
 * PDFGenerator muodostaa PDF tiedoston annetusta XSL tiedostosta.
 */

public class PDFGenerator {

    public String GeneratePDF(Document xmlfile, String pdf){

        try {

            // Setup directories
            //File baseDir = new File(".");

            // Setup input and output files
            //File xmlfile = new File(baseDir, "EmployeeXML.xml");
            File xsltfile = new File("CVStyle1.xsl");
            File pdffile = new File(pdf);


            // configure fopFactory as desired
            FopFactory fopFactory = FopFactory.newInstance();

            FOUserAgent foUserAgent = fopFactory.newFOUserAgent();
            // configure foUserAgent as desired

            // Setup output
            OutputStream out = new java.io.FileOutputStream(pdffile);
            out = new java.io.BufferedOutputStream(out);

            try {
                // Construct fop with desired output format
                Fop fop = fopFactory.newFop(MimeConstants.MIME_PDF,
foUserAgent, out);

                // Setup XSLT
                TransformerFactory factory = TransformerFactory.newInstance();
                Transformer transformer = factory.newTransformer(new
StreamSource(xsltfile));

                // Set the value of a <param> in the stylesheet
                transformer.setParameter("versionParam", "2.0");

                // Setup input for XSLT transformation
                //Source src = new StreamSource(xmlfile);
                Source src = new DOMSource(xmlfile);

                // Resulting SAX events (the generated FO) must be piped
through to FOP
                Result res = new SAXResult(fop.getDefaultHandler());
```

```
                // Start XSLT transformation and FOP processing
                transformer.transform(src, res);
                out.close();
                return pdf;
            } finally {
                out.close();

            }


        } catch (Exception e) {
            e.printStackTrace(System.err);
            System.exit(-1);
        }
        return "fail";
    }

}
```

Liite 9: EmpManager-luokan lähdekoodi

EmpManager.java

```
package com.iriba.cvgenerator.actionbean;

import java.util.ArrayList;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.conn.*;
import com.iriba.cvgenerator.dao.*;
import com.iriba.cvgenerator.to.*;
import com.iriba.cvgenerator.vo.*;

/*
 * EmpManager on tietokannan ja käyttöliittymän välillä
 * tapahtuvan tiedonsiirron solmukohta. Kaikki tieto suuntaan
 * tai toiseen kulkee EmpManager:n kautta.
 */

public class EmpManager {

        /*
         * Muuttujat
         */

        private int id;

        private EmployeeDAO empDAO;
        private EducationDAO eduDAO;
        private ProjectDAO proDAO;
        private SkillDAO skiDAO;
        private TrainingDAO traDAO;
        private CareerDAO carDAO;
        private LanguageDAO lanDAO;
        private CategoryDAO catDAO;
        private EmployeeSkillDAO esDAO;
        private AnnouncementDAO annDAO;

        /*
         * Konstruktori on tyhjä, koska olion luomista
         * tapahtuu paljon.
         */

        public EmpManager(){
        }

        /*
         * update-metodit tallettavat tietokantaan muutoksia
         * tauluihin. Jokaiselle taululle on oma metodi.
         */

        public void update(EmployeeTO emp) throws SystemException {
                empDAO = RDBDAOFactory.getEmployeeDAO();
                empDAO.updateEmployee(emp);
        }
        public void updateEducation(EducationTO edu) throws SystemException {
                eduDAO = RDBDAOFactory.getEducationDAO();
                eduDAO.updateEducation(edu);
        }
        public void updateTraining(TrainingTO tra) throws SystemException {
                traDAO = RDBDAOFactory.getTrainingDAO();
```

```
                traDAO.updateTraining(tra);
        }
        public void updateLanguage(LanguageTO lan) throws SystemException {
                lanDAO = RDBDAOFactory.getLanguageDAO();
                lanDAO.updateLanguage(lan);
        }
        public void updateCareer(CareerTO car) throws SystemException {
                carDAO = RDBDAOFactory.getCareerDAO();
                carDAO.updateCareer(car);
        }
        public void updateProject(ProjectTO pro) throws SystemException {
                proDAO = RDBDAOFactory.getProjectDAO();
                proDAO.updateProject(pro);
        }
        public void updateCategory(CategoryTO cat) throws SystemException {
                catDAO = RDBDAOFactory.getCategoryDAO();
                catDAO.updateCategory(cat);
        }
        public void updateSkill(SkillTO ski) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                skiDAO.updateSkill(ski);
        }
        public void updateEmpSkill(EmployeeSkillTO est) throws SystemException
{
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                esDAO.updateEmployeeSkill(est);
        }
        public void updateAnnouncement(AnnouncementTO ann) throws
SystemException {
                annDAO = RDBDAOFactory.getAnnouncementDAO();
                annDAO.updateAnnouncement(ann);
        }
        public void updateCurrentProject(CurrentProjectTO cpt) throws
SystemException {
                CurrentProjectDAO cpDAO =
RDBDAOFactory.getCurrentProjectDAO();
                cpDAO.updateCurrentProject(cpt);
        }

        /*
         * create-metodit luovat uuden taulun tietokantaan.
         *
         * create (työntekijän luomiseen käytetty metodi) luo
         * uudelle työntekijälle myös samalla CurrentProject-taulun.
         */

        public int create(EmployeeTO emp) throws SystemException {
                empDAO = RDBDAOFactory.getEmployeeDAO();
                int ret = empDAO.insertEmployee(emp);
                createCurrentProject(ret);
                return ret;
        }
        public void createEdu(EducationTO edu) throws SystemException {
                eduDAO = RDBDAOFactory.getEducationDAO();
                eduDAO.insertEducation(edu);
        }
        public void createTra(TrainingTO tra) throws SystemException {
                traDAO = RDBDAOFactory.getTrainingDAO();
                traDAO.insertTraining(tra);
        }
        public void createLan(LanguageTO lan) throws SystemException {
                lanDAO = RDBDAOFactory.getLanguageDAO();
                lanDAO.insertLanguage(lan);
        }
        public void createCar(CareerTO car) throws SystemException {
```

```
                carDAO = RDBDAOFactory.getCareerDAO();
                carDAO.insertCareer(car);
        }
        public void createPro(ProjectTO pro) throws SystemException {
                proDAO = RDBDAOFactory.getProjectDAO();
                proDAO.insertProject(pro);
        }
        public void createCat(CategoryTO cat) throws SystemException {
                catDAO = RDBDAOFactory.getCategoryDAO();
                catDAO.insertCategory(cat);
        }
        public void createSki(SkillTO ski) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                skiDAO.insertSkill(ski);
        }
        public void createEmpSki(EmployeeSkillTO est) throws SystemException {
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                esDAO.insertEmployeeSkill(est);
        }
        public void createAnnouncement(AnnouncementTO ann) throws
SystemException {
                annDAO = RDBDAOFactory.getAnnouncementDAO();
                annDAO.insertAnnouncement(ann);
        }
        public void createCurrentProject(int id) throws SystemException {
                CurrentProjectDAO cpDAO =
RDBDAOFactory.getCurrentProjectDAO();
                cpDAO.insertCurrentProject(id);
        }

        /*
         * remove metodit vastaavasti poistavat tauluja.
         * poistaminen tapahtuu aina tietokannassa tietueelle
         * annetun id-numeron perusteella.
         */

        public void remove(int id) throws SystemException {
                empDAO = RDBDAOFactory.getEmployeeDAO();
                removeCurrentProject(id);
                empDAO.deleteEmployee(id);
        }
        public void removeEdu(int id) throws SystemException {
                eduDAO = RDBDAOFactory.getEducationDAO();
                eduDAO.deleteSingleEducation(id);
        }
        public void removeTra(int id) throws SystemException {
                traDAO = RDBDAOFactory.getTrainingDAO();
                traDAO.deleteSingleTraining(id);
        }
        public void removeLan(int id) throws SystemException {
                lanDAO = RDBDAOFactory.getLanguageDAO();
                lanDAO.deleteSingleLanguage(id);
        }
        public void removeCar(int id) throws SystemException {
                carDAO = RDBDAOFactory.getCareerDAO();
                carDAO.deleteSingleCareer(id);
        }
        public void removePro(int id) throws SystemException {
                proDAO = RDBDAOFactory.getProjectDAO();
                proDAO.deleteSingleProject(id);
        }
        public void removeCat(int id) throws SystemException {
                catDAO = RDBDAOFactory.getCategoryDAO();
                catDAO.deleteCategory(id);
        }
```

```
        public void removeSki(int id) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                skiDAO.deleteSkill(id);
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                esDAO.deleteEmployeeSkill(id);
        }
        public void removeEmpSki(int id,int sid) throws SystemException {
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                esDAO.deleteSingleEmployeeSkill(id,sid);
        }
        public void removeAnnouncement(int id) throws SystemException {
                annDAO = RDBDAOFactory.getAnnouncementDAO();
                annDAO.deleteAnnouncement(id);
        }
        public void removeCurrentProject(int id) throws SystemException {
                CurrentProjectDAO cpDAO =
RDBDAOFactory.getCurrentProjectDAO();
                cpDAO.deleteCurrentProject(id);
        }

        /*
         * getAll metodit palauttaa halutun alkion taulusta
         * kaiken sisällön.
         */

        public ArrayList<EmployeeTO> getAllEmp() throws SystemException {
                empDAO = RDBDAOFactory.getEmployeeDAO();
                return empDAO.selectEmployeeTO(new EmployeeTO());
        }
        public ArrayList<EducationTO> getAllEmpEdu() throws SystemException {
                eduDAO = RDBDAOFactory.getEducationDAO();
                EducationTO temp = new EducationTO();
                temp.setEmployeeID(id);
                return eduDAO.selectEducationTO(temp);
        }
        public ArrayList<TrainingTO> getAllEmpTra() throws SystemException {
                traDAO = RDBDAOFactory.getTrainingDAO();
                TrainingTO temp = new TrainingTO();
                temp.setEmployeeID(id);
                return traDAO.selectTrainingTO(temp);
        }
        public ArrayList<LanguageTO> getAllEmpLan() throws SystemException {
                lanDAO = RDBDAOFactory.getLanguageDAO();
                LanguageTO temp = new LanguageTO();
                temp.setEmployeeID(id);
                return lanDAO.selectLanguageTO(temp);
        }
        public ArrayList<CareerTO> getAllEmpCar() throws SystemException {
                carDAO = RDBDAOFactory.getCareerDAO();
                CareerTO temp = new CareerTO();
                temp.setEmployeeID(id);
                return carDAO.selectCareerTO(temp);
        }
        public ArrayList<ProjectTO> getAllEmpPro() throws SystemException {
                proDAO = RDBDAOFactory.getProjectDAO();
                ProjectTO temp = new ProjectTO();
                temp.setEmployeeID(id);
                return proDAO.selectProjectTO(temp);
         }
        public ArrayList<CategoryTO> getAllCat() throws SystemException {
                catDAO = RDBDAOFactory.getCategoryDAO();
                return catDAO.selectCategoryTO(new CategoryTO());
         }
        public ArrayList<SkillTO> getAllCatSki(int id) throws SystemException
{
```

```
                skiDAO = RDBDAOFactory.getSkillDAO();
                SkillTO temp = new SkillTO();
                temp.setCategoryID(id);
                return skiDAO.selectSkillTO(temp);
        }
        public ArrayList<AnnouncementTO> getAllAnn() throws SystemException {
                annDAO = RDBDAOFactory.getAnnouncementDAO();
                AnnouncementTO temp = new AnnouncementTO();
                return annDAO.selectAnnouncementTO(temp);
        }

        /*
         * getSkiSearch metodi palauttaa listan taidoista, jotka
         * täyttävät annetun hakukriteerin.
         *
         * @parametrit:
         * search = annettu hakukriteeri.
         * lang = käytössä oleva kieli.
         *
         * @palautusarvo:
         * lista haun täyttävistä taidoista.
         */

        public ArrayList<SkillTO> getSkiSearch(String search, String lang)
throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                SkillTO temp = new SkillTO();
                if(search.equals("all")) {
                        return skiDAO.selectSkillTO(temp);
                }
                else {
                        ArrayList<SkillTO> sort = skiDAO.selectSkillTO(temp);
                        ArrayList<SkillTO> ret = new ArrayList<SkillTO>();
                        for (int i=0;i<=sort.size()-1;i++) {
                                if(sort.get(i).getSkillText(lang).toLowerCase(
).contains(search.toLowerCase())){
                                        ret.add(sort.get(i));
                                }
                        }
                        return ret;
                }
        }

        /*
         * getAllSkiEmp metodi palauttaa listan kaikista tietyn
         * taidon omaavista työntekijöistä.
         *
         * @parametrit:
         * id = käsiteltävän taidon tunnusnumero.
         * lang = käytössä oleva kieli.
         *
         * @palautusarvo:
         * lista työntekijän taidoista.
         */

        public ArrayList<SkillEmpVO> getAllSkiEmp(int id, String lang) throws
SystemException {
                String[] levels = {"Studies","Basics","1-6 Month","6-12
Month","1-3 Years","3-5 Years",">5 Years"};
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                EmployeeSkillTO eskill = new EmployeeSkillTO();
                eskill.setSkillID(id);
                ArrayList<EmployeeSkillTO> temp =
esDAO.selectEmployeeSkillTO(eskill);
```

```
                ArrayList<SkillEmpVO> returnlist = new
ArrayList<SkillEmpVO>();
                empDAO = RDBDAOFactory.getEmployeeDAO();
                EmployeeTO eto;
                for (int i=6;i>=0;i--) {
                        boolean check = false;
                        SkillEmpVO esv = new SkillEmpVO();
                        for (int j=0;j<=temp.size()-1;j++) {
                                if (temp.get(j).getLevel()==i) {
                                        check=true;
                                        eto = new EmployeeTO();
                                        eto.setID(temp.get(j).getID());
                                        eto =
empDAO.selectEmployeeTO(eto).get(0);
                                        esv.addToList(EmployeeUtil.convertToVO
(eto, lang));
                                }
                        }
                        if (check) {
                                esv.setLevel(levels[i]);
                                returnlist.add(esv);
                        }
                }
                return returnlist;
        }

        /*
         * getAllEmpSki metodi palauttaa listan työntekijän taidoista
         *
         * @parametri:
         * id = käsiteltävän työntekijän tunnusnumero.
         */

        public ArrayList<EmployeeSkillTO> getAllEmpSki(int id) throws
SystemException {
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                EmployeeSkillTO temp = new EmployeeSkillTO();
                temp.setID(id);
                return esDAO.selectEmployeeSkillTO(temp);
        }

        /*
         * getResViewList metodi palauttaa resurssinäkymässä tulostettavan
         * taulukon sisällön määrittämiseen tarvittavat tiedot.
         *
         * @palautusarvo:
         * taulukon sisältöä kuvaava olio.
         */

        public CPViewVO getResViewList() throws SystemException {
                CPViewVO ret = new CPViewVO();
                CurrentProjectDAO cpDAO =
RDBDAOFactory.getCurrentProjectDAO();
                ArrayList<CurrentProjectTO> list =
cpDAO.selectCurrentProjectTO(new CurrentProjectTO());
                ArrayList<CurrentProjectVO> voList = new
ArrayList<CurrentProjectVO>();
                for (int i=0;i<=list.size()-1;i++) {
                        empDAO = RDBDAOFactory.getEmployeeDAO();
                        EmployeeTO empto = new EmployeeTO();
                        empto.setID(list.get(i).getEmployeeID());
                        empto = empDAO.selectEmployeeTO(empto).get(0);
                        CurrentProjectVO temp =
CurrentProjectUtil.convertToVO(list.get(i));
                        temp.setName(empto.getNormalName());
```

```
                        voList.add(temp);
                }
                ret.setResource(voList);
                return ret;
        }

        /*
         * getEmpCatSkiMissList metodi palauttaa listan työntekijän taidoista,
         * joita työntekijällä ei ole.
         *
         * @parametrit:
         * ids = taulukko haettavien taitojen tunnistusnumeroista.
         * lang = käytössä oleva kieli.
         * search = annettu hakukriteeri.
         *
         * @palautusarvo:
         * lista työntekijältä puuttuvista taidoista.
         */

        public ArrayList<EmpSkiCatVO> getEmpCatSkiMissList(int[] ids, String
lang,String search) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                ArrayList<SkillTO> skilist = skiDAO.selectSkillTO(new
SkillTO());
                ArrayList<SkillTO> skimisslisttemp = new ArrayList<SkillTO>();
                ArrayList<SkillTO> skimisslist = new ArrayList<SkillTO>();
                for (int i=0;i<=skilist.size()-1;i++) {
                        if (checkTable(ids, skilist.get(i).getID()));
                        else skimisslisttemp.add(skilist.get(i));
                }
                if(search.equals("all"))skimisslist=skimisslisttemp;
                else {
                        for (int i=0;i<=skimisslisttemp.size()-1;i++) {
                                if(skimisslisttemp.get(i).getSkillText(lang).t
oLowerCase().contains(search.toLowerCase())) {
                                        skimisslist.add(skimisslisttemp.get(i)
);
                                }
                        }
                }
                ArrayList<EmpSkiCatVO> returnlist = new
ArrayList<EmpSkiCatVO>();
                catDAO = RDBDAOFactory.getCategoryDAO();
                for (int i=0;i<=skimisslist.size()-1;i++) {
                        CategoryTO cattemp = new CategoryTO();
                        cattemp.setID(skimisslist.get(i).getCategoryID());
                        cattemp = catDAO.selectCategoryTO(cattemp).get(0);
                        String category=cattemp.getCategoryText(lang);
                        int index=checkList(category,returnlist);
                        if(index<0) {
                                EmpSkiCatVO escv = new EmpSkiCatVO();
                                escv.setCategoryname(cattemp.getCategoryText(l
ang));
                                EmpSkillVO esv = new EmpSkillVO();
                                esv.setEmployeeID(id);
                                esv.setSkillID(skimisslist.get(i).getID());
                                esv.setSkillName(skimisslist.get(i).getSkillTe
xt(lang));
                                escv.addToList(esv);
                                returnlist.add(escv);
                        }
                        else {
                                EmpSkillVO esv = new EmpSkillVO();
                                esv.setSkillID(skimisslist.get(i).getID());
```

```
                                esv.setSkillName(skimisslist.get(i).getSkillTe
xt(lang));
                                returnlist.get(index).addToList(esv);
                        }
                }
                return sortSkills(returnlist);
        }

        /*
         * checkTable metodi tarkistaa onko taulukossa tiettyä
         * alkiota. getEmpCatSkiMissList-metodi käyttää tätä metodia.
         *
         * @parametrit:
         * ids = taulukko josta etsitään alkiota
         * id = mitä taulukosta etsitään.
         *
         * @palautusarvo:
         * totuusarvo löytyikö alkio vai ei.
         */

        public boolean checkTable(int[] ids,int id) {
                boolean ret = false;
                for (int i=0;i<=ids.length-1;i++) {
                        if(ids[i]==id) ret=true;
                }
                return ret;
        }

        /*
         * getEmpCatSkiList metodi palauttaa listan työntekijän
         * omistamista taidoista.
         *
         * @parametrit:
         * ids = taulukko taitojen tunnusluvuista.
         * lang = käytössä oleva kieli.
         *
         * @palautusarvo
         * lista työntekijän taidoista.
         */

        public ArrayList<EmpSkiCatVO> getEmpCatSkiList(int[] ids, String lang)
throws SystemException {
                ArrayList<EmpSkiCatVO> returnlist = new
ArrayList<EmpSkiCatVO>();
                catDAO = RDBDAOFactory.getCategoryDAO();
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                for (int i=0;i<=ids.length-1;i++) {
                        CategoryTO cattemp = new CategoryTO();
                        cattemp.setID(getCatID(ids[i]));
                        cattemp = catDAO.selectCategoryTO(cattemp).get(0);
                        String category=cattemp.getCategoryText(lang);
                        EmployeeSkillTO estemp = new EmployeeSkillTO();
                        estemp.setSkillID(ids[i]);
                        estemp.setID(id);
                        estemp = esDAO.selectEmployeeSkillTO(estemp).get(0);
                        int index=checkList(category,returnlist);
                        if(index<0) {
                                EmpSkiCatVO escv= new EmpSkiCatVO();
                                escv.setCategoryname(cattemp.getCategoryText(l
ang));
                                escv.addToList(EmpSkillUtil.convertToVO(estemp
, lang));
                                returnlist.add(escv);
                        }
                        else {
```

```
                                returnlist.get(index).addToList(EmpSkillUtil.c
onvertToVO(estemp,lang));
                        }
                }
                return sortSkills(returnlist);
        }

        /*
         * sortSkill() ja sortSki() metodit järjestävät categorian
         * taidot aakkosjärjestykseen.
         *
         * @palautusarvo: aakkosjärjestyksessä oleva lista.
         */

        public ArrayList<EmpSkiCatVO> sortSkills(ArrayList<EmpSkiCatVO> list)
{
                ArrayList<EmpSkiCatVO> returnlist = new
ArrayList<EmpSkiCatVO>();
                for (int i = 0; i < list.size(); i++) {
                        returnlist.add(sortSki(list.get(i)));
                }
                return returnlist;
        }

        public EmpSkiCatVO sortSki(EmpSkiCatVO esc) {
                ArrayList<EmpSkillVO> list = esc.getCatEmpSkiList();
                ArrayList<EmpSkillVO> returnlist = new
ArrayList<EmpSkillVO>();
                while (list.size() > 1) {
                        int index = 0;
                        for (int i = 1; i < list.size(); i++ ) {
                                if
(list.get(index).getSkillName().toLowerCase().compareTo(
                                                list.get(i).getSkillName().toL
owerCase()) > 0) index = i;
                        }
                        returnlist.add(list.get(index));
                        list.remove(index);
                }
                returnlist.add(list.get(0));
                esc.setCatEmpSkiList(returnlist);
                return esc;
        }


        /*
         * checkList tarkistaa onko palautettavassa listassa jo tietty
         * kategoria. Jos on, se palauttaa indeksin, jos ei metodi palauttaa
         * arvon -1.
         *
         * @parametrit:
         * category = kategorian nimi, jota etsitään.
         * list = lista, josta etsitään.
         *
         * @palautusarvo:
         * numeroarvo joka kertoo löytyneen indeksin tai -1.
         */

        public int checkList(String category, ArrayList<EmpSkiCatVO> list) {
                        for (int i=0;i<=list.size()-1;i++) {
                                if(list.get(i).getCategoryName().equals(catego
ry)) {
                                        return i;
                                }
                        }
```

```
                return -1;
        }

        /*
         * get Name metodit palauttavat haettavan alkion nimen.
         *
         * @parametrit:
         * id = haettavan alkion tunnusnumero.
         * l = käytössä oleva kieli.
         *
         * @palautusarvo:
         * haetun alkion nimi.
         */

        public String getSkillName(int id,String l) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                SkillTO temp = new SkillTO();
                temp.setID(id);
                temp = skiDAO.selectSkillTO(temp).get(0);
                return temp.getSkillText(l);
        }

        /*
         * getCatID palauttaa kategorian tunnusluvun annetun
         * taidon tunnusluvun avulla.
         *
         * @parametri:
         * id = taidon tunnusluku.
         *
         * @palautusarvo:
         * etsityn kategorian tunnusluku.
         */

        public int getCatID(int id) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                SkillTO temp = new SkillTO();
                temp.setID(id);
                temp = skiDAO.selectSkillTO(temp).get(0);
                return temp.getCategoryID();
        }
        public String getCatName(int id, String lang) throws SystemException {
                catDAO = RDBDAOFactory.getCategoryDAO();
                CategoryTO temp = new CategoryTO();
                temp.setID(id);
                temp = catDAO.selectCategoryTO(temp).get(0);
                return temp.getCategoryText(lang);
        }

        /*
         * getCurrent - metdit palauttavat tietyn tietoalkion
         * kaikki tiedot tietoalkion tunnusluvun avulla.
         *
         * @parametri:
         * id = haettavan alkion tunnusluku.
         *
         * @palautusarvo:
         * TO-olio, jossa on kaikki alkion tiedot.
         */

        public EmployeeTO getCurrentEmployee(int id) throws SystemException {
                 empDAO = RDBDAOFactory.getEmployeeDAO();
                 EmployeeTO employee=new EmployeeTO();
                 employee.setID(id);
                 ArrayList<EmployeeTO> employees =
empDAO.selectEmployeeTO(employee);
```

```
                return employees.get(0);
        }
        public EducationTO getCurrentEducation(int id) throws SystemException
{
                eduDAO = RDBDAOFactory.getEducationDAO();
                EducationTO education=new EducationTO();
                education.setID(id);
                ArrayList<EducationTO> educations =
eduDAO.selectEducationTO(education);
                return educations.get(0);
        }
        public TrainingTO getCurrentTraining(int id) throws SystemException {
                traDAO = RDBDAOFactory.getTrainingDAO();
                TrainingTO training=new TrainingTO();
                training.setID(id);
                ArrayList<TrainingTO> trainings =
traDAO.selectTrainingTO(training);
                return trainings.get(0);
        }
        public LanguageTO getCurrentLanguage(int id) throws SystemException {
                lanDAO = RDBDAOFactory.getLanguageDAO();
                LanguageTO language=new LanguageTO();
                language.setID(id);
                ArrayList<LanguageTO> languages =
lanDAO.selectLanguageTO(language);
                return languages.get(0);
        }
        public CareerTO getCurrentCareer(int id) throws SystemException {
                carDAO = RDBDAOFactory.getCareerDAO();
                CareerTO career=new CareerTO();
                career.setID(id);
                ArrayList<CareerTO> careers = carDAO.selectCareerTO(career);
                return careers.get(0);
        }
        public ProjectTO getCurrentProject(int id) throws SystemException {
                proDAO = RDBDAOFactory.getProjectDAO();
                ProjectTO project=new ProjectTO();
                project.setID(id);
                ArrayList<ProjectTO> projects =
proDAO.selectProjectTO(project);
                return projects.get(0);
        }
        public CategoryTO getCurrentCategory(int id) throws SystemException {
                catDAO = RDBDAOFactory.getCategoryDAO();
                CategoryTO category=new CategoryTO();
                category.setID(id);
                ArrayList<CategoryTO> categorys =
catDAO.selectCategoryTO(category);
                return categorys.get(0);
        }
        public SkillTO getCurrentSkill(int id) throws SystemException {
                skiDAO = RDBDAOFactory.getSkillDAO();
                SkillTO skill=new SkillTO();
                skill.setID(id);
                ArrayList<SkillTO> skills = skiDAO.selectSkillTO(skill);
                return skills.get(0);
        }
        public EmployeeSkillTO getCurrentEmpSkill(int id,int sid) throws
SystemException {
                esDAO = RDBDAOFactory.getEmployeeSkillDAO();
                EmployeeSkillTO empskill=new EmployeeSkillTO();
                empskill.setID(id);
                empskill.setSkillID(sid);
                empskill = esDAO.selectEmployeeSkillTO(empskill).get(0);
                return empskill;
```

```
        }
        public AnnouncementTO getCurrentAnnouncement(int id) throws
SystemException {
                annDAO = RDBDAOFactory.getAnnouncementDAO();
                AnnouncementTO ann=new AnnouncementTO();
                ann.setID(id);
                ArrayList<AnnouncementTO> anns =
annDAO.selectAnnouncementTO(ann);
                return anns.get(0);
        }
        public CurrentProjectTO getCurrentCProject(int id) throws
SystemException {
                CurrentProjectDAO cpDAO =
RDBDAOFactory.getCurrentProjectDAO();
                CurrentProjectTO cpt = new CurrentProjectTO();
                cpt.setEmployeeID(id);
                return cpDAO.selectCurrentProjectTO(cpt).get(0);
        }

        public void setID(int id) {
                this.id=id;
        }

}
```

Liite 10: BaseActionbean-luokan lähdekoodi

BaseActionBean.java

```
package com.iriba.cvgenerator.actionbean;

import java.util.Date;

import org.apache.log4j.Logger;

import net.sourceforge.stripes.action.ActionBean;
import net.sourceforge.stripes.action.ActionBeanContext;
import net.sourceforge.stripes.action.RedirectResolution;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.conn.Languages;
import com.iriba.cvgenerator.dao.RDBEducationDAO;
import com.iriba.cvgenerator.vo.AnnouncementListVO;
import com.iriba.cvgenerator.vo.CareerListVO;
import com.iriba.cvgenerator.vo.CareerVO;
import com.iriba.cvgenerator.vo.CategoryListVO;
import com.iriba.cvgenerator.vo.CategoryVO;
import com.iriba.cvgenerator.vo.EducationList;
import com.iriba.cvgenerator.vo.EducationVO;
import com.iriba.cvgenerator.vo.EmpSkillListVO;
import com.iriba.cvgenerator.vo.EmployeeListVO;
import com.iriba.cvgenerator.vo.EmployeeUtil;
import com.iriba.cvgenerator.vo.EmployeeVO;
import com.iriba.cvgenerator.vo.LanguageListVO;
import com.iriba.cvgenerator.vo.LanguageVO;
import com.iriba.cvgenerator.vo.ProjectListVO;
import com.iriba.cvgenerator.vo.ProjectVO;
import com.iriba.cvgenerator.vo.SkillListVO;
import com.iriba.cvgenerator.vo.SkillVO;
import com.iriba.cvgenerator.vo.TrainingListVO;
import com.iriba.cvgenerator.vo.TrainingVO;

/*
 * BaseActionBean on kaikkien actionBeanien ylä-luokka.
 * BaseActionBeaniin on koottu kaikki actionbeanien
 * tarvitsemat yleiset muuttujat ja - metodit. Lisäksi
 * BaseActionBeanissa on actionbeaninlle pakolliset
 * ominaisuudet (context).
 */

public class BaseActionBean implements ActionBean {

        /*
         * Muuttujina aliluokkien käyttöön jokainen VO, manager,
         * valittu kieli, hakuehto ja context.
         */

        private CVGActionBeanContext context;
        private EmpManager employeeManager = new EmpManager();
        private Languages languages = new Languages();
        private EmployeeVO emp;
        private String search;
        private CategoryVO cat;
        private SkillVO ski;
        private ProjectVO pro;
        private EducationVO edu;
        private CareerVO car;
        private TrainingVO tra;
```

```
        private LanguageVO lan;
        private static Logger logger =
Logger.getLogger(RDBEducationDAO.class);

        public LanguageVO getLan() {
                return lan;
        }
        public void setLan(LanguageVO lan) {
                this.lan=lan;
        }

        public TrainingVO getTra() {
                return tra;
        }
        public void setTra(TrainingVO tra) {
                this.tra=tra;
        }

        public CareerVO getCar() {
                return car;
        }
        public void setCar(CareerVO car) {
                this.car=car;
        }

        public EducationVO getEdu() {
                return edu;
        }
        public void setEdu(EducationVO e) {
                edu=e;
        }

        public void setPro(ProjectVO pro){
                this.pro = pro;
        }
        public ProjectVO getPro(){
                return pro;
        }

        public EmployeeVO getEmp() {
                return emp;
        }
        public void setEmp(EmployeeVO emp) {
                this.emp = emp;
        }

        public CVGActionBeanContext getContext() {
                return context;
        }


        public void setContext(ActionBeanContext context) {
                if (!(context instanceof CVGActionBeanContext)) {
                        System.out.println("ActionBean Context was not of type
CVGActionBeanContext!!!");
                        throw new IllegalArgumentException("ActionBean Context
was not of type CVGActionBeanContext");
                }

                this.context = (CVGActionBeanContext) context;
        }

        public void setLanguages(Languages languages) {
                this.languages = languages;
        }
```

```
        public Languages getLanguages() {
                return languages;
        }



        public EmpManager getEmployeeManager() {
                return employeeManager;
        }

        public void setEmployeeManager(EmpManager em) {
                this.employeeManager = em;
        }


        public String getSelectedLanguage() {
                return getLanguages().getSelectedLanguage();
        }

        /*
         * getRedirect metodi on kaikkien resoluutioni-metoideiden
         * käyttämä metodi. Resoluution palauttaminen siirtää
         * käyttöjärjestelmän uudelle sivulle. Käyttämällä ne tämän
         * metodin kautta uusi sivu tulee talletettua contextiin.
         */

        public RedirectResolution getRedirect(String url) {
                if(url.equals("/Error.jsp"))
getLanguages().setSelectedLanguage("en");
                getContext().getServletContext().setAttribute("currentPage",
url);
                return new RedirectResolution(url).flash(this);
        }


        protected String getCurrentPage() {
                return (String)
getContext().getServletContext().getAttribute("currentPage");
        }


        protected void setCurrentPage(String url) {
                getContext().getServletContext().setAttribute("currentPage",
url);
        }


        /*
         * getJotainList-metodit ovat käytössä, kun käyttöliittymä
         * avaa sivun, jossa tulee listata kaikki käsiteltävän aiheen
         * alkiot. Jokaiselle käyttöliittymän muodostamalle listalle on
         * oma getList-metodi.
         *
         * @Palautusarvo:
         * käyttölittymään tulostettava lista.
         */


        public EducationList getEducationList() {
                EducationList edul = new EducationList();
                edul.setLang(getSelectedLanguage());
                edul.setID(getEmp().getID());
                return edul;
        }
```

```
public TrainingListVO getTrainingList() {
        TrainingListVO tral = new TrainingListVO();
        tral.setLang(getSelectedLanguage());
        tral.setID(getEmp().getID());
        return tral;
}
public LanguageListVO getLanguageList() {
        LanguageListVO lanl = new LanguageListVO();
        lanl.setLang(getSelectedLanguage());
        lanl.setID(getEmp().getID());
        return lanl;
}
public CareerListVO getCareerList() {
        CareerListVO carl = new CareerListVO();
        carl.setLang(getSelectedLanguage());
        carl.setID(getEmp().getID());
        return carl;
}
public ProjectListVO getProjectList() {
        ProjectListVO prol = new ProjectListVO();
        prol.setLang(getSelectedLanguage());
        prol.setID(getEmp().getID());
        return prol;
}
public CategoryListVO getCategoryList() {
        CategoryListVO catl = new CategoryListVO();
        return catl;
}

public SkillListVO getSkillList() {
        SkillListVO skil = new SkillListVO();
        skil.setID(cat.getID());
        return skil;
}

public SkillListVO getSkillSearchList() {
        SkillListVO skil = new SkillListVO();
        skil.setSearch(search);
        return skil;
}

public EmpSkillListVO getEmpSkillList() {
        EmpSkillListVO esl = new EmpSkillListVO();
        esl.setLang(getSelectedLanguage());
        esl.setID(emp.getID());
        esl.setSearch(search);
        return esl;
}

public EmployeeListVO getSkillEmpList() {
        EmployeeListVO el = new EmployeeListVO();
        el.setLang(getSelectedLanguage());
        el.setSID(ski.getID());
        return el;
}

public AnnouncementListVO getAnnouncementList(){
        return new AnnouncementListVO();
}

public void setSearch(String s) {
        search = s;
}
public String getSearch() {
        return search;
```

```
	}

	public void setCat(CategoryVO cat){
		this.cat = cat;
	}
	public CategoryVO getCat(){
		return cat;
	}

	public void setSki(SkillVO ski){
		this.ski = ski;
	}
	public SkillVO getSki(){
		return ski;
	}

	/*
	 * updateLastModified metodi päivittää yksittäisen
	 * työntekijän "viimeiseksi päivitetty"-tietoa. Tätä
	 * metodia kutsutaan aina, kun luodaan uusia tauluja
	 * tai päivitetään tietoja.
	 */

	public void updateLastModified() {
		try {
			setEmp(EmployeeUtil.convertToVO(getEmployeeManager().g
etCurrentEmployee(
					getEmp().getID()),
getSelectedLanguage()));
			getEmp().setLastModified(new Date());
			getEmployeeManager().update(EmployeeUtil.convertToTO(g
etEmp(), getSelectedLanguage()));
		}
		catch (SystemException e) {
			logger.error("Update Last Modified caused an
Exception");
		}
	}

	public Date getNewDate(){
		return new Date();
	}
}
```

Liite 11: CVGActionBeanContext-luokan lähdekoodi

CVGActionBeanContext.java

```
package com.iriba.cvgenerator.actionbean;

import com.iriba.cvgenerator.vo.EmployeeVO;

import net.sourceforge.stripes.action.ActionBeanContext;

/*
 * CVGActionBeanContext periytyy ActionBean-luokkien
 * käyttämästä ActionBeanContext:sta. Ohjelma käyttää
 * tätä contextia, jotta siihen voi lisätä omia atribuutteja.
 *
 * Contextissa on aina get-set pari atribuuteista mitä
 * halutaan, että contexti pitää muistissa.
 */

public class CVGActionBeanContext extends ActionBeanContext {

        public EmployeeVO getUser(){
                return (EmployeeVO)
getRequest().getSession().getAttribute("user");
        }
        public void setUser(EmployeeVO currentUser){
                getRequest().getSession().setAttribute("user", currentUser);
        }

        public String getUserLevel(){
                return (String)
getRequest().getSession().getAttribute("userLevel");
        }

        public void setUserLevel(String currentUserLevel){
                getRequest().getSession().setAttribute("userLevel",
currentUserLevel);
        }



        public EmployeeVO getEmpVO() {
         return (EmployeeVO) getRequest().getSession().getAttribute("EmpVo");

   }

   public void setEmpVO(EmployeeVO vo) {
       getRequest().getSession().setAttribute("EmpVO", vo);
   }

   public int getEmpID(){
        return (Integer) getRequest().getSession().getAttribute("EmpID");
   }
   public void setEmpID(int id){
        getRequest().getSession().setAttribute("EmpID", new Integer(id));
   }

   public void setError(String error) {
           getRequest().getSession().setAttribute("error", error);
   }
   public String getError() {
           return (String) getRequest().getSession().getAttribute("error");
   }
```

```
    public void logout() {
        getRequest().getSession().invalidate();
    }
}
```

Liite 12: LoginActionBean- sekä LogOutActionBean-luokan lähdekoodi

LoginActionBean.java

```
package com.iriba.cvgenerator.actionbean;

import java.util.ArrayList;
import java.util.StringTokenizer;

import com.iriba.cvgenerator.SystemException;
import com.iriba.cvgenerator.conn.RDBDAOFactory;
import com.iriba.cvgenerator.to.EmployeeTO;
import com.iriba.cvgenerator.vo.EmployeeUtil;
import com.iriba.cvgenerator.vo.EmployeeVO;

import net.sourceforge.stripes.action.RedirectResolution;
import net.sourceforge.stripes.action.Resolution;
import net.sourceforge.stripes.validation.LocalizableError;
import net.sourceforge.stripes.validation.Validate;
import net.sourceforge.stripes.validation.ValidationError;

/*
 * LoginActionBean hoitaa ohjelman sisäänkirjautumisessa
 * tehtävät toimenpiteet ja käskyt.
 */

public class LoginActionBean extends BaseActionBean {
    @Validate(required=true)
    private String username;

    @Validate(required=true)
    private String password;

    /*
     * Tehtävänannon mukaan mahdolliset käyttäjät ja heidän
     * salasanat ovat kovakoodattuna tässä tiedostossa.
     */

    private String[] users =
    {
                "jere.meriluoto",
                "markus.santi",
    };
    private String[] passwords =
    {
                "admin",
                "user",
    };

    public void setUsername(String username) { this.username = username; }

    public String getUsername() { return username; }

    public void setPassword(String password) { this.password = password; }

    public String getPassword() { return password; }

    /*
     * login metodi suorittaa tarkistuksen ja sisäänkirjautumiseen liittyvän
     * tiedonkäsittelyn. userLevel muuttujan asettaminen vaikuttaa oikeuksiin
     * mitä käyttäjällä tulee olemaan ohjelmassa.
     */
```

```
    public Resolution login() {

        int index=-1;
        // Etsii työntekijää.
        for(int i=0; i<=users.length-1;i++){
                if(getUsername().equals(users[i])) index = i;
        }

        // Jos työntekijää ei löytynyt.
        if (index == -1) {
            ValidationError error = new
LocalizableError("usernameDoesNotExist");
            getContext().getValidationErrors().add("username", error);
            return getContext().getSourcePageResolution();
        }
        // Työntekijä löytyi, mutta salasana on väärä.
        else if (!getPassword().equals(passwords[index])) {
            ValidationError error = new LocalizableError("incorrectPassword");
            getContext().getValidationErrors().add("password", error);
            return getContext().getSourcePageResolution();
        }
        // Sisäänkirjautuminen onnistui.
        EmployeeVO currentUser;
        String userLevel="user";
        if(users[index].equals("jouko.salo")){
                currentUser = new EmployeeVO(0,"Jouko", null,
"Salo",0,null,null);
                userLevel="admin";
                getContext().setUserLevel(userLevel);
                getContext().setUser(currentUser);
            return new
RedirectResolution("/com/iriba/cvgenerator/actionbean/Employee.action");
        }
        else {
                StringTokenizer name = new StringTokenizer(users[index], ".");
                String fname = name.nextToken();
                String lname = name.nextToken();

                // Hakee työntekijän tiedot tietokannasta.
                EmployeeTO criteria = new EmployeeTO(0,
fname,null,lname,0,null);
                try {
                        ArrayList <EmployeeTO> allEmp =
RDBDAOFactory.getEmployeeDAO().selectEmployeeTO(criteria);

                        if(!allEmp.isEmpty()){
                                currentUser =
EmployeeUtil.convertToVO(allEmp.get(0), "en");
                                getContext().setUserLevel(userLevel);
                        getContext().setUser(currentUser);
                        return new RedirectResolution("/com/iriba/cvgenerator/
actionbean/Employee.action?edit=&amp;emp.ID="+currentUser.getID());
                        }
                        else {
                                // Työntekijä ei ollut tietokannassa, luo uusi
CV.
                                getContext().setUser(EmployeeUtil.convertToVO(
criteria, "en"));
                                getContext().setUserLevel(userLevel);
                                return new RedirectResolution("/AddEmp.jsp");

                        }
                }
                catch (SystemException e) {
                        getContext().setError(e.getMessage());
```

```
                        return getRedirect("/Error.jsp");
                }
        }
    }
}
```

LogoutActionBean.java

```
package com.iriba.cvgenerator.actionbean;

import net.sourceforge.stripes.action.RedirectResolution;
import net.sourceforge.stripes.action.Resolution;

/*
 * LogoutActionBean hoitaa tapahtumat uloskirjautumis
 * nappia painettaessa.
 */


public class LogoutActionBean extends BaseActionBean {
    public Resolution logout() throws Exception {
        getContext().logout();
        return new RedirectResolution("/Login.jsp");
    }
}
```

Liite 13: SecurityFilter-luokan lähdekoodi

SecurityFilter.java

```
package com.iriba.cvgenerator.actionbean;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/*
 * SecurityFilter luokka tarkistaa sisäänkirjautumisen.
 * Ohjelmassa on ilman sisäänkirjautumista pelkästään login-sivu
 * salittu. Jos jollekkin muulle sivulle yrittää päästä (kirjoittamalla
 * sen manuaalisesti) heittää ohjelma käyttäjän login-sivulle.
 */

public class SecurityFilter implements Filter {
    private static Set<String> publicUrls = new HashSet<String>();

    static {
        publicUrls.add("/Login.jsp");
        publicUrls.add("/com/iriba/cvgenerator/actionbean/Login.action");
    }


    public void init(FilterConfig filterConfig) throws ServletException { }

    public void doFilter(ServletRequest servletRequest,
                         ServletResponse servletResponse,
                         FilterChain filterChain) throws IOException,
ServletException {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;

        if (request.getSession().getAttribute("user") != null) {
            filterChain.doFilter(request, response);
        }
        else if ( isPublicResource(request) ) {
            filterChain.doFilter(request, response);
        }
        else {
            response.sendRedirect(
                    request.getContextPath() + "/Login.jsp");
        }
    }


    protected boolean isPublicResource(HttpServletRequest request) {
        String resource = request.getServletPath();

        return publicUrls.contains(request.getServletPath())
```

```
                || (!resource.endsWith(".jsp") && !
resource.endsWith(".action"));
    }


    public void destroy() { }
}
```

Liite 14: Järjestelmäpoikkeus-luokan lähdekoodi

SystemException.java

```
package com.iriba.cvgenerator;

/*
 * SystemException on järjestelmän tietokantatransaktioille
 * tehty perus poikkeus.
 */

public class SystemException extends Throwable {


        private SystemException() {
        }


        public SystemException(String arg0) {
                super(arg0);
        }


        public SystemException(Throwable arg0) {
                super(arg0);
        }


        public SystemException(String arg0, Throwable arg1) {
                super(arg0, arg1);
        }

}
```

Liite 15: Esimerkki tietoja näyttävästä JSP-sivusta

EditCVs.jsp

```
<!--EditCVs-sivulla näkyy työntekijän perusnäkymä -->

<%@ taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>



<stripes:layout-render name="/layout/layout.jsp" >
    <stripes:layout-component name="content">



                <stripes:form
action="/com/iriba/cvgenerator/actionbean/Employee.action">

                <stripes:hidden name="languages.selectedLanguage" value="$
{actionBean.languages.selectedLanguage}"/>

                <stripes:hidden name="emp.ID" value="${actionBean.emp.ID}"/>

                <table>
                        <tr>
                                <td>

                                         <jsp:include page="/Buttons.jsp"/>
                                </td>
                                <td>
                                        <h3>${actionBean.emp.name}</h3>
                                        <table class="display">
                                        <tr><th>Year of Birth:</th><td>$
{actionBean.emp.yearOfBirth}</td></tr>
                                        <tr><th>Summary:</th>
                                        <td style="text-align: left;"
class="texttd">${actionBean.emp.summary}</td></tr>
                                        <tr><th>Last Modified:</th><td>$
{actionBean.emp.modified}</td></tr>
                                        <tr><td>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Employee.action" event="editEmp">
                                         <IMG src="$
{pageContext.request.contextPath}/images/buttons/edit_b.gif"/>
                                                <stripes:param name="emp.ID"
value="${actionBean.emp.ID}"/>
                                                <stripes:param
name="languages.selectedLanguage" value="$
{actionBean.languages.selectedLanguage}"/>
                                </stripes:link>
                                        </td>
                                        </tr>
                                        </table>
                                        <c:choose>
                                        <c:when test="${userLevel ==
'admin'}">
                                        <stripes:link
href="/com/iriba/cvgenerator/actionbean/Employee.action" event="removeEmp">
```

```
                                        <IMG src="$
{pageContext.request.contextPath}/images/buttons/remove_b.gif"/>
                                                <stripes:param name="emp.ID"
value="${actionBean.emp.ID}"/>
                                                <stripes:param
name="languages.selectedLanguage" value="$
{actionBean.languages.selectedLanguage}"/>
                            </stripes:link>
                                        </c:when>
                                        </c:choose>
                                </td>

                        </tr>


                </table>
                </stripes:form>

            </stripes:layout-component>
</stripes:layout-render>
```

Liite 16: Esimerkki tietoja muuttavasta JSP-sivusta

```
EditProInfo.jsp

<!--EditProInfo-sivulla voi muuttaa työntekijän projekti-alkion sisältää -->

<%@ taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>



<stripes:layout-render name="/layout/layout.jsp" >
    <stripes:layout-component name="content">


                <stripes:form
action="/com/iriba/cvgenerator/actionbean/ManagePro.action">

                <table>
                        <tr>
                                <td>
                                        <jsp:include page="/Buttons.jsp"/>
                                </td>
                                <td>
                                        <table>
                                        <tr><td><h3>Edit
Project</h3></td></tr>
                                        <tr><td>Name:</td><td><stripes:text
name="pro.name" value = "${actionBean.pro.name}" id="long"/>
                                                <stripes:errors
field="pro.name"/></td></tr>
                                        <tr><td>Start
Date:</td><td><stripes:text name="start" value = "$
{actionBean.pro.start}"id="short"/>
                                                <stripes:errors field="start"/
></td></tr>
                                        <tr><td>End
Date:</td><td><stripes:text name="end" value = "$
{actionBean.pro.end}"id="short"/>
                                                <stripes:errors
field="end"/></td></tr>
                                        <tr><td>Platform:</td><td><stripes:tex
t name="pro.platform" value = "${actionBean.pro.platform}"id="long"/>
                                                <stripes:errors
field="pro.platform"/></td></tr>
                                        <tr><td>Software:</td><td><stripes:tex
t name="pro.software" value = "${actionBean.pro.software}"id="long"/>
                                                <stripes:errors
field="pro.software"/></td></tr>
                                        <tr><td>Customer:</td><td><stripes:tex
t name="pro.customer" value = "${actionBean.pro.customer}"id="long"/>
                                                <stripes:errors
field="pro.customer"/></td></tr>
                                        <tr><td>Own
Role:</td><td><stripes:text name="pro.ownrole" value = "$
{actionBean.pro.ownrole}"id="long"/>
                                                <stripes:errors
field="pro.ownrole"/></td></tr>
```

```
                                        <tr><td>Description:</td><td>
<stripes:textarea name="pro.description" value = "$
{actionBean.pro.description}"/>
                                                <stripes:errors
field="pro.description"/></td></tr>
                                        <tr><td>Responsibilities:</td><td><str
ipes:textarea name="pro.responsibilities" value = "$
{actionBean.pro.reslities}"/>
                                                <stripes:errors
field="pro.responsibilities"/></td></tr>

                                        <tr><td></td><td>Write all
responsibilities into the text area, use |-symbol to divide them</td></tr>
                                        <tr><td>
                                                <input name="savePro"
type="image" src="$
{pageContext.request.contextPath}/images/buttons/save_b.gif">
                                                        <stripes:param
name="pro.ID" value="${actionBean.pro.ID}"/>
                                                        <stripes:param
name="pro.employeeID" value="${actionBean.pro.employeeID}"/>
                                                        <stripes:param
name="emp.ID" value="${actionBean.pro.employeeID}"/>
                                                        <stripes:param
name="languages.selectedLanguage" value="$
{actionBean.languages.selectedLanguage}"/>
                                                </td>
                                                <td>
                                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Project.action" event="cancelEdit">
                                                        <IMG src="$
{pageContext.request.contextPath}/images/buttons/cancel_b.gif"/>
                                                        <stripes:param
name="emp.ID" value="${actionBean.pro.employeeID}"/>
                                                        <stripes:param
name="pro.ID" value="${actionBean.pro.ID}"/>
                                                        <stripes:param
name="languages.selectedLanguage" value="$
{actionBean.languages.selectedLanguage}"/>
                                                </stripes:link>
                                        </td></tr>
                                        </table>
                </table>
                </stripes:form>

            </stripes:layout-component>
</stripes:layout-render>
```

Liite 17: Sivujen pohjasivun lähdekoodi

layout.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<stripes:layout-definition>

<HTML>
<HEAD>
        <TITLE>IRIBA OY CV MANAGER</TITLE>
        <LINK HREF="${pageContext.request.contextPath}/iribaStyle.css"
REL="stylesheet" TYPE="text/css" />
</HEAD>
<BODY topmargin=0>

        <jsp:useBean  id="empManager" scope="session"
                     class="com.iriba.cvgenerator.actionbean.EmpManager"/>

        <DIV ID="container">



        <DIV ID="logo">
                        <IMG SRC="$
{pageContext.request.contextPath}/images/newlogo.jpg">
        </DIV>

        <DIV ID="navigation">
                <CENTER>
                <c:choose>
                        <c:when test="${userLevel == 'admin'}">
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Employee.action"
event="listEmployees">Employees</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Category.action"
event="skillManager">Skill Manager</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Category.action"
event="skillBrowser">Skill Browser</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Announcement.action"
event="listAnnouncements">Announcements</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/ManageCP.action"
event="viewCP">Resource View</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Logout.action" event="logout"> Logout
</stripes:link>
                        </c:when>
                        <c:otherwise>
```

```
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Employee.action" event="edit">My
CV</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Announcement.action"
event="listAnnouncements">Announcements</stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/ManageCP.action"
event="viewCP">Resource View
                                        <stripes:param name="emp.ID" value="$
{actionBean.emp.ID}"/>
                                </stripes:link>
                                <b>::</b>
                                <stripes:link
href="/com/iriba/cvgenerator/actionbean/Logout.action" event="logout"> Logout
</stripes:link>
                        </c:otherwise>
                </c:choose>
                </CENTER>
        </DIV>
        <CENTER>
    <stripes:form
action="/com/iriba/cvgenerator/actionbean/SelectLanguage.action">
      <stripes:label for="languages.selectedLanguage"/>:
                <stripes:select name="languages.selectedLanguage" value="$
{actionBean.selectedLanguage}">
                <stripes:options-collection collection="$
{actionBean.languages.allLanguages}"
                                                        label="name"
value="id"/>
        </stripes:select>
                <stripes:hidden name="emp.ID" value="${actionBean.emp.ID}"/>
                <stripes:hidden name="pro.ID" value="${actionBean.pro.ID}"/>
                <stripes:hidden name="edu.ID" value="${actionBean.edu.ID}"/>
                <stripes:hidden name="car.ID" value="${actionBean.car.ID}"/>
                <stripes:hidden name="tra.ID" value="${actionBean.tra.ID}"/>
                <stripes:hidden name="lan.ID" value="${actionBean.lan.ID}"/>
                <input name="changeLanguage" type="image" src="$
{pageContext.request.contextPath}/images/buttons/change_b.gif">
        </stripes:form>
        </CENTER>

        <DIV ID="form">
                 <stripes:layout-component name="content"/>
        </DIV>

        <DIV ID="footer">
                <CENTER>
                © 2008 Iriba Oy
                </CENTER>
        </DIV>


</DIV>
</BODY>
</HTML>
</stripes:layout-definition>
```