WEB-PALVELU HOUSING ENABLER -ARVIOINTIVÄLINEELLE

Sakarias Liukko

Opinnäytetyö
Joulukuu 2009

Mediatekniikka
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) LIUKKO, Sakarias	Julkaisun laji opinnäytetyö	Päivämäärä 1.12.2009
	Sivumäärä 32	Julkaisun kieli suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
WEB-PALVELU HOUSING ENABLER -ARVIOINTIVÄLINEELLE

Koulutusohjelma
Mediatekniikka

Työn ohjaaja(t)
MANNINEN, Pasi

Toimeksiantaja(t)
HATTUNEN, Jaana – ET Elämisen Tuki Oy

Tiivistelmä

Opinnäytteen lähtökohtana oli ET Elämisen Tuki Oy:n tarve korvata esteettömyyden arvioinnissa käytettävä, Housing Enabler -menetelmän apuna toiminut, tietokonesovellus tietoverkkojen välityksellä toimivalla järjestelmällä.

Työssä perehdyttiin Housing Enabler -arviointivälineeseen ja palvelukeskeisen arkkitehtuurin mukaisesti toteutettaviin tietoverkkojen kautta käytettävien palvelujen toteutusteknologioihin Java EE -arkkitehtuurissa. Järjestelmä toteutettiin näiden periaatteiden mukaisesti EJB 3.0 -määrityksen mukaisena web-palveluna. EJB (Enterprise JavaBean) on Java-kieleen perustuva komponenttiteknologia komponenttien rakentamiseksi ja ajamiseksi sovelluspalvelimella.

Työn tuloksena syntyi valmis web-palvelu, jonka avulla voidaan tuoda järjestelmään Housing Enabler -arviointeja, sekä hakea ja muokata järjestelmässä olevia arviointeja. Arvioinnit kirjataan PDF-lomakkeille, jotka voidaan erikseen toteutettavalla asiakassovelluksella ladata järjestelmään verkon välityksellä. Arviointien perusteella voidaan tehdä erilaisia analyysejä asumisympäristöistä. Palveluun voidaan lisätä myöhemmin uusia toiminnallisuuksia ilman että jo toteuttavia toimintoja tarvitsee muuttaa.

Avainsanat (asiasanat)

web-palvelu, palvelukeskeinen arkkitehtuuri, EJB, Housing Enabler, esteettömyys

Muut tiedot

Author(s)
LIUKKO, Sakarias

Type of publication
Bachelor´s Thesis

Date
1.12.2009

Pages
32

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
WEB SERVICE FOR HOUSING ENABLER ASSESSMENT INSTRUMENT

Degree Programme
Media Engineering

Tutor(s)
MANNINEN, Pasi

Assigned by
HATTUNEN, Jaana – ET Elämisen Tuki Ltd

Abstract

The basis of the thesis was the assigner's need to improve the existing computer software used with Housing Enabler instrument. The software was decided to be replaced with a computer system that is accessed via a web browser over a network.

The theory part is about the Housing Enabler assessment instrument and the Java EE web service technologies based on service-oriented architecture. The web service system was implemented according to the guidelines of EJB 3.0 specification. EJB (Enterprise JavaBeans) technology is the server-side component architecture for Java Platform.

The result was a web service, which can be accessed by a client application. The implemented services provide operations for importing and exporting Housing Enabler assessments as well as editing existing assessment data. The assesments are stored to the PDF-forms, which can be imported to system. The assesment data can be used to generate analyses about housing environments.

Keywords
web service, service-oriented architecture, EJB, Housing Enabler, accessibility

Miscellaneous

SISÄLTÖ

KUVIOT

SANASTO

EAR	Enterprise ARchive. JAR-tiedosto, joka sisältää koko Java EE -sovelluksen.
EJB	Enterprise Java Beans, komponenttimalli Java EE -standardissa.
JAR	Java ARchive. Zip-tiedoston kaltainen tiivistetty paketti, jota käytetään yleisesti kaikkien Java-ohjelmien jakeluun.
Java EE	Java Enterprise Edition. Java-standardi ja arkkitehtuuri hajautetun liiketoimintajärjestelmän toteuttamiseen.
JSP	JavaServer Pages. Teknologia, jolla yhdistetään palvelimella suoritettavat Java-ohjelmat web-sivuille.
NetBeans	Sun Microsystems perustama avoimeen lähdekoodiin perustuva kehitysympäristö ja ohjelmointityökalu.
Ohjelmointirajapinta	Käyttöliittymä, jolla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli kommunikoida keskenään.
PDF	Portable Document Format. Adoben kehittämä PostScript-kieleen pohjautuva käyttöjärjestelmäriippumaton, siirrettävä tiedostomuoto.
Servlet	Java EE -palvelimen web-säiliön osuudessa ajettava Java-ohjelma.
WAR	Web Application Archive. Web-sovelluksen jakamisessa käytettävä JAR-tiedosto.
Web-palvelu	Internetissä käytettäviä ja julkaistuja itsenäisiä ja itsensä kuvaavia komponentteja, joihin voidaan kohdistaa kyselyitä, ja

	jotka ovat yhteiskäyttöisiä sekä ajonaikaisesti konfiguroitavissa.
XML	eXtensible Markup Language. Metakieli, jolla määritellään rakenteellisia merkkauskieliä.

1 TYÖN LÄHTÖKOHDAT

1.1 Tehtävät ja taustat

ESKO (Esteetön koti ikääntyneiden ja erityisryhmien asumiseen) -hankkeessa kehitetään Jyväskylän ja pohjoisen Keski-Suomen kuntien alueella ikääntyneiden ja erityisryhmien asukaslähtöistä asumista, ennakoivaa suunnittelua sekä nykyisen asumis-, rakentamis- ja palvelutilanteen arviointia. Sen ensimmäisessä vaiheessa pilotoitiin objektiivisen, eli mitatun asumisen, uusi kansainvälinen Housing Enabler -menetelmä Suomeen hankenimellä Esteetön koti. (ESKO-hanke 2009.)

Hankkeen tavoitteisiin kuului myös menetelmää tukevan web-sovelluksen kehittäminen. Sovelluksen suunnittelusta vastasi Eventizer Oy. Sovelluksen käyttöliittymä toteutettiin Jyväskylän ammattikorkeakoulun opiskelijoiden työharjoitteluna ja web-palvelut opinnäytetyönä. Web-pohjainen toteutus tarjoaa mahdollisuuden käyttää sovellusta ja sieltä löytyviä tietoja mistä tahansa internetiin liitetystä koneesta. Web-palvelut välittävät sovellukselle kaikki tarvittavat tiedot ja mahdollistavat tietojen tallentamisen tietokantaan.

1.2 Toimeksiantaja

ESKO-hankkeessa mukana oleva ET Elämisen Tuki Oy toimi työn toimeksiantajana. ET Elämisen Tuki Oy on keskisuomalainen sosiaali- ja terveysalan hyvinvointipalveluja tuottava yritys, joka tuottaa rakennetun ympäristön esteettömyyden kartoituspalveluja. Esteettömyyskartoituksen avulla kiinteistön saavutettavuutta voidaan muokata paremmaksi ja asuintilat saadaan tukemaan asukkaan toimintakykyä ja itsenäistä selviytymistä. (ET Elämisen Tuki Oy n.d.)

ET Elämisen Tuki Oy käyttää asiakkaiden asuinympäristön fyysisen esteettömyyden arvioinnissa *Housing Enabler* -arviointimenetelmää. Menetelmässä arvioidaan henkilön toimintakykyä ja henkilön asumisympäristön esteitä. Yhdistämällä nämä arviot

voidaan löytää asumisympäristön ongelmallisimmat kohdat. (ET Elämisen Tuki Oy n.d.)

Menetelmää kehitetään edelleen mm. Tanskan Åhlborgin toimintaterapiakoulutuksen, Jyväskylän yliopiston ja Jyväskylän ammattikorkeakoulun yhteisessä kehityshankkeessa. Hankkeen tavoitteen on tuottaa luotettava arviointiväline kansalliseen käyttöön. (ET Elämisen Tuki Oy n.d.)

1.3 Toimeksiantajan vaatimukset ja tarpeet

Toimeksiantaja käyttää esteettömyyden arvioinneissa tietokoneohjelmaa, johon kirjataan arviointilomakkeiden tiedot. Ohjelma yhdistää sekä asiakkaan toimintakyvyn että asuinympäristön tiedot kertoen, mitkä kohdat asuinympäristössä ovat ongelmallisimpia esteettömyyden kannalta. Osana menetelmän kehitystyötä ESKO-hankkeessa haluttiin kehittää sovellusta palvelemaan paremmin toimeksiantajan tarpeita. Tärkeimpiä toimeksiantajan toivomia kehityskohteita olivat

- käyttöoikeus aineistoon organisaatiokohtaisesti
- monitasoisen organisoinnin mahdollistava kategorisointi
- henkilökohtainen työtila käyttäjälle ja
- lomakkeiden täyttäminen arviointitilanteessa ja valmiiden arvioiden lataaminen sovellukseen.

Toinen sovelluksen käyttötarkoitus on tilastollisen tiedon esittäminen tallennetuista arvioinneista. Sovelluksen tuli tarjota erilaisia tilastollisia analyysejä joilla voidaan vertailla eri arviointeja ja löytää näin asiakkaalle paras mahdollinen asumisympäristö.

2 HOUSING ENABLER -ARVIOINTIVÄLINE

2.1 Arviointivälineen kuvaus

Housing Enabler on ikääntyvien ihmisten fyysisen ympäristön esteettömyyden arviointiin kehitetty menetelmä. Menetelmän on kehittänyt vuonna 1979 arkkitehtuurin professori Edward Steinfeld Yhdysvalloissa (Slaug 2001). Ruotsalalainen gerontologian professori Susanne Iwarsson jatkoi menetelmän kehitystä Lundin yliopistossa kollegoineen vuonna 1997. Tieteellisesti kehitetyn menetelmän avulla voidaan objektiivisesti arvioida asuinympäristössä havaittuja ongelmia ikääntyvän asukkaan fyysisen toimintakyvyn kannalta esimerkiksi asunnon suunnittelun, rakentamisen tai ennalta ehkäisevän kotikäynnin yhteydessä. Tavoitteena on, että fyysinen ympäristö tukee ikääntyvän asukkaan itsenäistä selviytymistä hänen asuinympäristössään. (Iwarsson & Slaug 2008, 14.)

Housing Enablerin tärkein ominaisuus on asiakkaan toimintakyvyn arvioiminen suhteessa asuinympäristössä toimintaa vaikeuttaviin esteisiin. Useimmat aiemmat menetelmät esteettömyyden arvioinnissa keskittyvät vain ympäristön esteisiin (ESOK-hanke 2008). Housing Enabler -menetelmässä esteettömyys on subjektiivinen käsite, jota voidaan arvioida vain suhteessa arvioitavan henkilön toimintakykyyn. Täysin toimintakykyinen henkilö ei kohtaa normaaleissa asuinympäristöissä koskaan toimintakyvyn esteitä. Toisaalta taas toiselle henkilölle esteellisyyttä aiheuttavat tekijät voivat olla toiselle välttämättömiä toimintakyvyn säilyttämiseksi. Esimerkiksi pienet tasoerot vaikeuttavat liikuntaesteisen toimintakykyä, mutta näkövammaiselle ne ovat välttämättömiä (Iwarsson & Slaug 2008, 15, 26).

2.2 Arviointiprosessi

Iwarssonin ja Slaugin (2008, 14–15) kuvaamalla Housing Enabler -menetelmällä tapahtuva ympäristön arviointi koostuu kolmesta eri vaiheesta:

1. henkilön toimintakyvyn rajoitteiden ja liikkumisapuvälineiden tarpeen arviointi
2. ympäristökartoitus ja
3. kokonaispisteiden laskeminen.

Arvioinnissa on tärkeää tunnistaa kaikki ongelmakohdat, ja näin ollen arvioijalla onkin suuri vastuu arvioinnin tarkkuudesta. Luotettavan arvioinnin tekemiseen Housing Enabler -menetelmällä vaaditaan toimintaterapeutin pätevyys. Tästä huolimatta myös muut ammattiryhmät voivat hyötyä menetelmästä. (Mts. 25.)

Toimintakyvyn rajoitteiden ja liikkumisapuvälineiden tarpeen arviointi

Housing Enabler -arvioinnin ensimmäisessä vaiheessa kartoitetaan yksilön tai ryhmän tai väestöryhmän toimintakyvyn rajoitteita. Arviointi sisältää 15 kohtaa toimintakyvyn rajoitteista ja liikkumisapuvälineiden tarpeista. Arvioinnissa havainnoidaan arvioitavaa henkilöä omassa asuinympäristössään ja haastatellaan häntä. Arvioinnin tuloksena saadaan asiakkaan toimintakykyprofiili. Jos arvioinnin kohteena on ryhmä tai väestöryhmä, toimintakykyprofiili kuvaa koko ryhmän toimintakyvyn rajoitteita. Tällöin arvioinnissa luotetaan enemmän tilastotietoihin ja muuhun saatavissa olevaan tietoon arvioitavasta ryhmästä. (Mts. 15.)

Ympäristökartoitus

Toisessa vaiheessa, ympäristökartoituksessa, arvioidaan asiakkaan asumisympäristössä olevia ympäristöesteitä. Ympäristökartoitukseen kuuluu 188 kohtaa, jotka jakaantuvat neljään osioon: ulkotiloihin, sisäänkäynteihin, sisätiloihin ja opastukseen ja viestintään. Näistä osioista arvioija ottaa arviointiin mukaan ne osa-alueet, joilla on merkitystä kyseisessä asiakas- ja arviointitilanteessa. Ympäristökartoituksessa lähtökohtana on, että arvioitavassa asuinympäristössä asuva henkilö voi toimia aktiivisesti ja itsenäisesti huolimatta rajoitteista toimintakyvyssään. (Mts. 44.)

Kartoitus tehdään havainnoimalla asuinympäristöä ympäristökartoituksen kohtien mukaan. Jokaiseen kohtaan voi tehdä muistiinpanoja arvioinnin tueksi. Tarvittaessa voidaan piirtää yksinkertaisia kuvia tai kuvata kohde kameralla, jolloin voidaan vielä havainnollistaa lisää arvioitavaa kohdetta. (Mts. 48.)

Kokonaispisteiden laskeminen

Kolmannessa vaiheessa lasketaan arviointikohteille ympäristön esteellisyyttä kuvaavat kokonaispisteet. Jokaiselle ympäristöarvioinnissa olevalle 188 kohdalle on etukäteen määritetty, mihin toimintakyvyn rajoitteisiin ne vaikuttavat ja kuinka paljon asteikolla 1–4. Piste 1 on tarkoittaa vähäistä haittaa ja piste 4 korkeinta ongelmakuormitusta. Jokaisesta asuinympäristöstä löytyneestä esteettömyysongelmasta, joka vaikuttaa arvioitavalla henkilöllä olevaan toimintakyvyn rajoitteeseen, lisätään pisteitä arvioinnin kokonaispisteisiin. Nämä pisteet kertovat, kuinka paljon esteettömyysongelmia asiakkaan toimintakyvyllä kohdataan kyseisessä ympäristössä. Mitä korkeammat pisteet ovat, sitä suurempia ovat esteettömyysongelmat. (Mts. 15.)

2.3 Tulosten analysointi

Housing Enabler -menetelmällä tehtyjen arviointien tuloksista voidaan tehdä monia erilaisia analyyseja. Analyysit voivat perustua numeerisiin Housing Enabler -pisteisiin tai ne voivat olla kuvauksia asiakkaan toimintakyvystä ja asumisympäristöstä. Tällaisia kuvauksia voidaan käyttää pohjana esimerkiksi suunniteltaessa muutostöitä asumisympäristöön. (Ivarsson & Slaug 2008, 49.)

Arvioista tehtäviin numeerisiin Housing Enabler -pisteisiin perustuvilla analyyseilla voidaan saada tilastollista tietoa laajemmin toimintakyvyn rajoitteista ja ympäristön esteistä. Analysoimalla eri arviointitulosten pistemääriä voidaan etsiä esimerkiksi tiettyjen väestöryhmien yleisimpiä toimintakyvyn rajoituksia ja useimmin esiintyviä asuinympäristön ongelmakohtia. Näin saadaan helposti selville, mitkä rajoitteet haittaavat useimmiten esteettömyyttä, ja tällaisia esteitä voidaan ennaltaehkäistä jo asuntojen suunnittelussa. Tuloksien perusteella voidaan myös kerätä tarkkoja tietoja sellaisista asuinympäristöistä, jotka aiheuttavat eniten esteettömyysongelmia tietyssä asunnossa tai asuntoalueella. (Mts. 50.)

Toinen tapa soveltaa arviointien tuloksia on etsiä toimintarajoitteiselle henkilölle esteettömin mahdollinen asuinympäristö. Arvioimalla useita eri asuinympäristöjä ja vertaamalla niiden ympäristöarviointeja henkilön toimintakyvyn arviointiin voidaan nähdä, missä asuinympäristössä esteellisyyden kokonaispisteet jäävät pienimmiksi,

eli asuinympäristössä on kyseisellä henkilöllä vähiten esteettömyysongelmia. (ESKO-hanke 2009.)

Tulosten perusteella voidaan myös arvioida esimerkiksi henkilön ikääntymisen tai sairauden etenemisen aiheuttamia toimintakyvyn rajoitteiden muutoksia tulevaisuudessa luomalla vaihtoehtoisia toimintakyvyn rajoitteiden arviointeja. Näitä arviointeja voidaan verrata asukkaan nykyisen asuinympäristön ympäristökartoitukseen, jolloin nähdään, mitkä esteet tulevat myöhemmin haittaamaan henkilön toimintaa. (Iwarsson & Slaug 2008, 40.)

3 PALVELUKESKEINEN ARKKITEHTUURI JA WEB-PALVELUT

3.1 Palvelukeskeinen arkkitehtuuri

Palvelukeskeinen arkkitehtuuri on arkkitehtuuritason suunnittelumalli, joka jakaa sovelluksen toiminnan erillisiin palveluihin. Palvelukeskeisen arkkitehtuurin avulla eri tietojärjestelmien toiminnot ja prosessit on suunniteltu toimimaan itsenäisinä, avoimina ja joustavina *palveluina*. Näiden palveluiden yhteistoiminta pohjautuu rakenteellisiin ja järjestelmäriippumattomiin palvelukuvauksiin, eli palvelukeskeinen arkkitehtuuri on teknologiariippumaton eikä ota suoraan kantaa toteutusteknologiaan. Palveluiden hyödyntäminen perustuu aina avoimien, standardoitujen tai muutoin vakiintuneiden, rajapintojen ja tekniikoiden käyttöön (Papazoglou 2008, 3, 22–23.)

Palvelukeskeisen arkkitehtuurin mallissa keskeisimpänä osana ovat palvelut. Palvelu määritellään mekanismiksi, joka mahdollistaa pääsyn yhteen tai useampaan palvelun rajapinnan kautta tarjottuun kykyyn. Palvelun rajapinta sisältää ohjeet tarjoamiensa kykyjen käyttämiseen. Palvelu on yleensä läpinäkymätön, eli sen toteutus on piilotettu palvelun käyttäjältä lukuun ottamatta palvelun rajapinnan paljastamia tietomalleja ja kykyjen käyttämiseksi tarvittavia tietoja. Palveluntarjoaja tarjoaa palvelua muiden käyttöön, mutta palvelun loppukäyttäjät eivät välttämättä tiedä palveluntarjoajaa.

Palvelujen vaikutukset voivat sisältää pyydetyn tiedon vastauksena saadun tiedon, jaettujen tietoresurssien tilan muutoksen tai näiden kahden yhdistelmän. (MacKenzie, Laskey, McCabe, Brown & Metz 2006, 12.)

Palvelu voidaan jakaa kolmeen eri osaan: palvelun tuottajaan, asiakkaaseen ja palvelurekisteriin. Palvelukeskeinen arkkitehtuuri yhdistää palvelun käyttäjän tarpeet (engl. needs) ja kyvyt (engl. capabilities). Palvelun tuottaja tuottaa palvelun ja julkaisee sen kuvauksen palveluhakemistoon. Palvelun käyttäjä etsii ja hakee palvelua palveluhakemistosta. Palvelun käyttö tapahtuu palvelun tarjoajan ja palvelun käyttäjän välisellä kommunikaatiolla. (MacKenzie ym. 2006, 8–9.)

3.2 Web-palvelu

Web-palvelut ovat ohjelmistojärjestelmiä, jotka mahdollistavat eri sovellusten välisen vuorovaikutuksen tietoverkon yli. Web-palvelut ovat usein internetin kautta julkaistuja ja käytettäviä itsenäisiä komponentteja, joihin voidaan kohdistaa kyselyitä. Web-palvelut ovat kehittyneet yhtenä keskeisenä palvelukeskeisen arkkitehtuurin toteutusteknologiana ja ne on myös useasti virheellisesti rinnastettu toisiinsa. (Papazoglou 2008, 3, 22–23.)

Web-palveluita tarjoavat siihen erikoistuneet komponentit. Näitä komponentteja voidaan tehdä eri kehitysalustoille. Web-palvelut ovat alustariippumattomia, eli komponenttien on pystyttävä kommunikoimaan standardien mukaan riippumatta siitä, mille alustalle ne on toteutettu. (Mts. 16–17.)

Web-palvelun ulkoiset rajapinnat on määritelty ja kuvattu standardoidulla XML-kielellä. Web-palvelut käyttävät XML-kieltä pyyntöjen ja vastausten generointiin. XML-pohjainen ratkaisu mahdollistaa alusta- ja käyttöjärjestelmäriippumattomuuden. Web-palvelujen protokollakenttään kuuluu kolme XML-pohjaista komponenttia: SOAP, WSDL ja UDDI. (Mts. 32–33.)

SOAP (Simple Object Access Protocol)

SOAP on W3C:n kehittämä protokolla, jonka avulla voidaan välittää XML-

muotoisia viestejä. Se on ohjelmistokielestä, alustasta ja käyttöliittymästä riippumaton. (Mts. 120.)

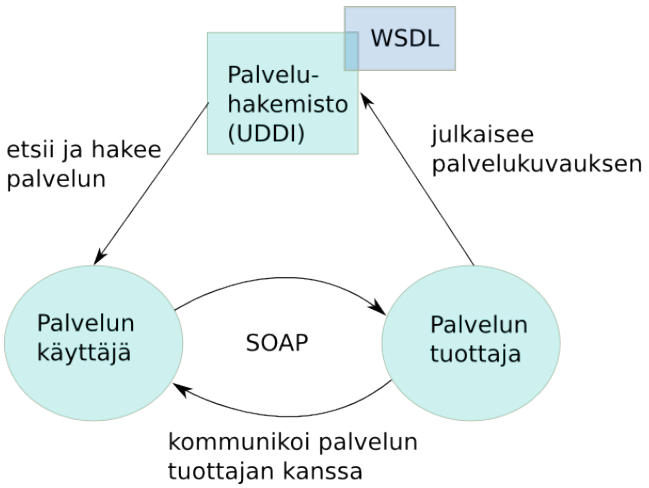
WSDL (Web Service Description Language)

WSDL on W3C:n kehittämä XML-pohjainen kieli, jolla voidaan kuvata web-palvelun käyttöliittymä eli julkinen rajapinta sekä sen toteutus. WSDL sisältää komponenttien kuvauksen, yhteydenottotavat ja ohjeen miten tarjottuja palveluita voidaan käyttää. (Mts. 149.)

UDDI (Universal Description Discovery and Integration)

UDDI on web-palveluiden tietorekisteri, jota käyttämällä asiakas voi etsiä palveluita. Asiakas voi käyttää yksityistä tai julkista tietorekisteriä. Tietorekisterit tarjoavat kaiken tarvittavan tiedon web-palvelusta ja sen käytöstä. (Mts. 177.)

Samoin kuin palvelukeskeisen arkkitehtuurissa myös web-palvelussa ovat osapuolina palvelun tuottaja, palvelun käyttäjä ja palveluhakemisto. Kuviossa 1 on esitetty osapuolien väliset suhteet. Web-palvelun tuottaja kuvaa tarjoamiaan palveluita WSDL-kielellä. Tuottajan ja käyttäjän väliset viestit lähetetään SOAP-protokollan määrittelemässä muodossa. Siirtoprotokollana käytetään usein HTTP-protokollaa. Tarjolla olevat palvelut voidaan lisäksi tallentaa UDDI-rekisteriin. (Mts. 24.)



KUVIO 1. Palvelukeskeisen arkkitehtuurin osapuolet (Papazogolou 2008, 24)

3.3 Web-palvelun hyödyt toteutettavassa järjestelmässä

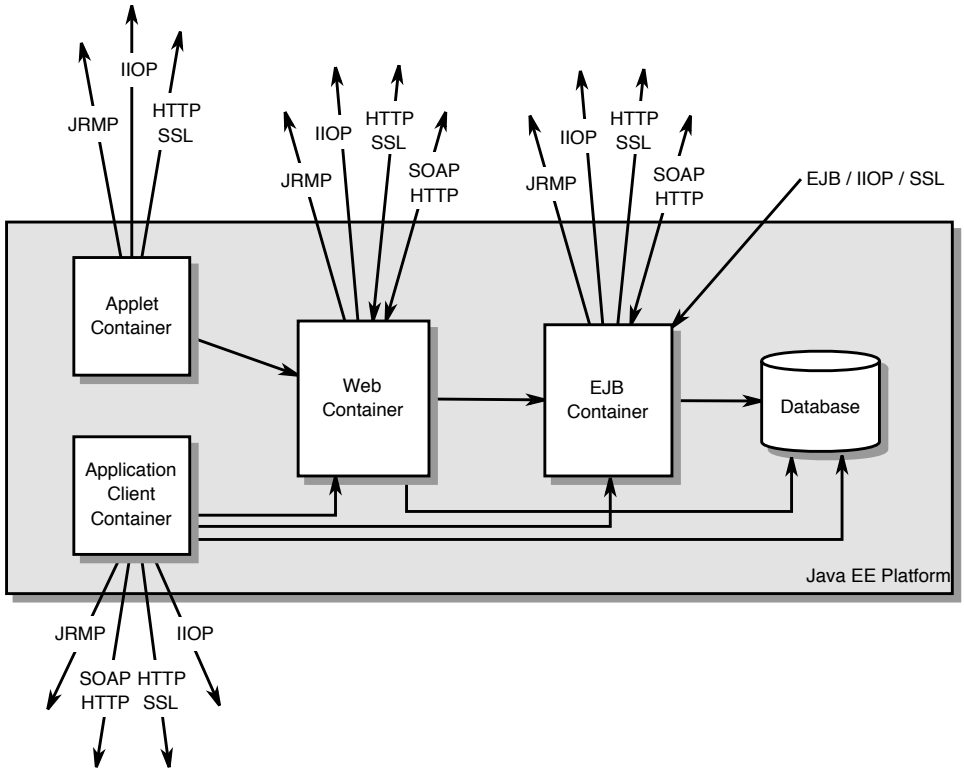
Opinnäytteessä toteutettavan järjestelmän vaatimuksissa (ks. 5.1. Arkkitehtuuriset vaatimukset) määriteltiin, että järjestelmän täytyi tarjota teknologiasta riippumattomat rajapinnat arviointitietojen käyttämiseen. Näin voidaan jatkossa tarjota pääsy järjestelmän tietoihin myös ulkopuolisista järjestelmistä. Web-palvelu ratkaisee yhteensopivuusongelmia yhteisellä tietojen merkkauksella ja linkittämisellä ja tarjoaa alusta- ja käyttöjärjestelmäriippumattoman toteutuksen. Web-palvelun sovelluskomponentit ovat myös helposti uudelleenkäytettäviä, ja palvelun modulaarinen rakenne mahdollistaa uusien komponenttien liittämisen järjestelmään. Näiden web-palvelun tarjoamien hyötyjen vuoksi järjestelmä päätettiin toteuttaa web-palveluna.

4 WEB-PALVELUN TOTEUTUS JAVALLA

Järjestelmän web-palvelu toteutettiin Java EE:n tarjoaman EJB 3.0 -arkkitehtuurin mukaisesti. Java EE sisältää useita eri kerroksia. Web-palvelu toteutetaan lähinnä Java EE:n EJB-kerroksessa.

4.1 Java EE:n rakenne

Java EE koostuu useista eri kerroksista. Kuviossa 2 esitetään Shannonin (2006, 15) laatima Java EE:n kerrosmalli. Web-palvelun toteuttava Java EE -sovellus käyttää näitä samoja kerroksia ja koostuu yleensä web-kerroksessa ja EJB-kerroksessa olevista moduuleista. Java EE -sovellus pakataan EAR-päätteiseen tiedostoon, joka sisältää web-moduulin WAR-päätteiseen tiedoston ja EJB-moduulin JAR-päätteiseen tiedoston. (Knutson & Kreger 2002 15, 65.)



KUVIO 2. Java EE:n kerrosmalli (Shannon 2006, 15)

Web-kerros toimii rajapintana asiakasohjelmien lähettämille HTTP-kutsuille. Näitä kutsuja kuuntelevat JSP-sivut ja servletit sijaitsevat tyypillisesti juuri web-kerroksessa. Web-kerros voi ottaa vastaan myös web-palvelulle lähetettyjä SOAP-sanomia ja toimia näin web-palvelun rajapintana. (Shannon 2006, 7.)

EJB-kerroksen muodostaa sovelluspalvelin, siinä oleva EJB-säiliö (engl. EJB Container) -alusta ja EJB-säiliössä toimivat EJB-komponentit. Varsinainen web-palvelun toteutus sijaitsee EJB-kerroksessa. Palvelun rajapinta voi sijaita myös EJB-kerroksessa EJB-komponentin toteuttaman rajapinnan avulla. (Mts. 7.)

4.2 EJB-komponenttityypit

EJB-arkkitehtuuri määrittelee kolme erilaista EJB-komponenttityyppiä: istuntopohjaisen komponentin, kohdepohjaisen komponentin ja viestipohjaisen komponentin (DeMichiel & Keith 2006, 35).

4.2.1 Istuntopohjainen komponentti (engl. Session Bean)

Istuntopohjaiset komponentit esittävät toimintoja, jotka suoritetaan asiakasohjelman kutsusta. Komponentti voi olla tilaton (engl. stateless) tai tilallinen (engl. statefull). Tilallinen komponentti pitää muistissa tietoa asiakassovelluksen tilasta. Webpalvelun toteuttava komponentti on aina tilaton. Istuntopohjainen komponentti ei suoraan esitä tietokannassa olevaa dataa, mutta voi käsitellä tietokannan dataa. Istuntopohjaisuus tarkoittaa, että komponentin ilmentymä kestää asiakkaan istunnon ajan ja ilmentymällä voi olla vain yksi asiakas. Istuntopohjaisen komponentin elinaika on usein melko lyhyt. Tyypillisesti EJB-säiliö tarjoaa skaalautuvan ajonaikaisen ympäristön, jossa voi toimia useita istuntopohjaisen komponentin ilmentymiä samanaikaisesti. (DeMichiel & Keith 2006, 35, 61–62)

4.2.2 Kohdepohjainen komponentti (engl. Entity Bean)

Kohdepohjaiset EJB-komponentit esittävät sovelluksen pysyvää, tietokannassa säilytettävää tietoa. Kohdepohjaisen komponentin ilmentymällä voi olla samanaikaisesti useampi. Kohdepohjaisen komponentin ilmentymän elinaika on pitkä, ja usein se kestää niin kauan kuin tietokannassa oleva tieto. Kohdepohjainen komponentti tukee deklaratiivisia transaktioita, eli transaktiot voidaan konfiguroida annotaatioilla ohjelmoinnin sijasta. (DeMichiel & Keith 2006, 36.)

4.2.3 Viestipohjainen komponentti (engl. Message-Driven Bean)

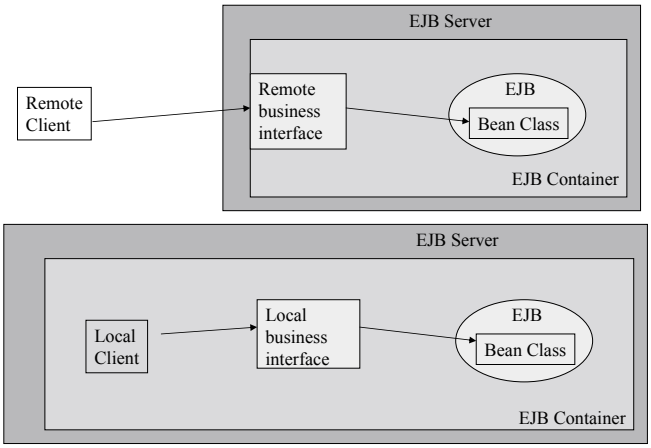
Viestipohjaisen komponentin ilmentymä toimii ainoastaan silloin, kun se saa sanoman. Viestipohjainen komponentti ottaa vastaan sanomia asynkronisesti, eli sanoman lähettäjän ei tarvitse odottaa vastaanottajan vastausta. Komponentti sisältää myös liiketoimintalogiikkaa sanoman käsittelyyn. Viestipohjainen komponentti ei suoraan esitä tietokannassa olevaa dataa, mutta voi käsitellä sitä. Viestipohjaisen komponentin ilmentymä on suhteellisen lyhytikäinen ja on tilaton. Tyypillisesti EJB-säiliö tarjoaa skaalautuvan ajonaikaisen ympäristön, jossa voi toimia useita viestipohjaisen komponentin ilmentymiä samanaikaisesti. (DeMichiel & Keith 2006, 36, 103–104.)

4.3 EJB-komponentti

EJB-komponentti koostuu rajapinnasta, komponenttiluokasta ja mahdollisista muista luokista. Komponenttiin kuuluvat tiedostot pakataan yhteen jakelukelpoiseksi JAR-päätteiseksi tiedostoksi. (Lipitsäinen 2007, 51.)

4.3.1 Komponentin rajapinta

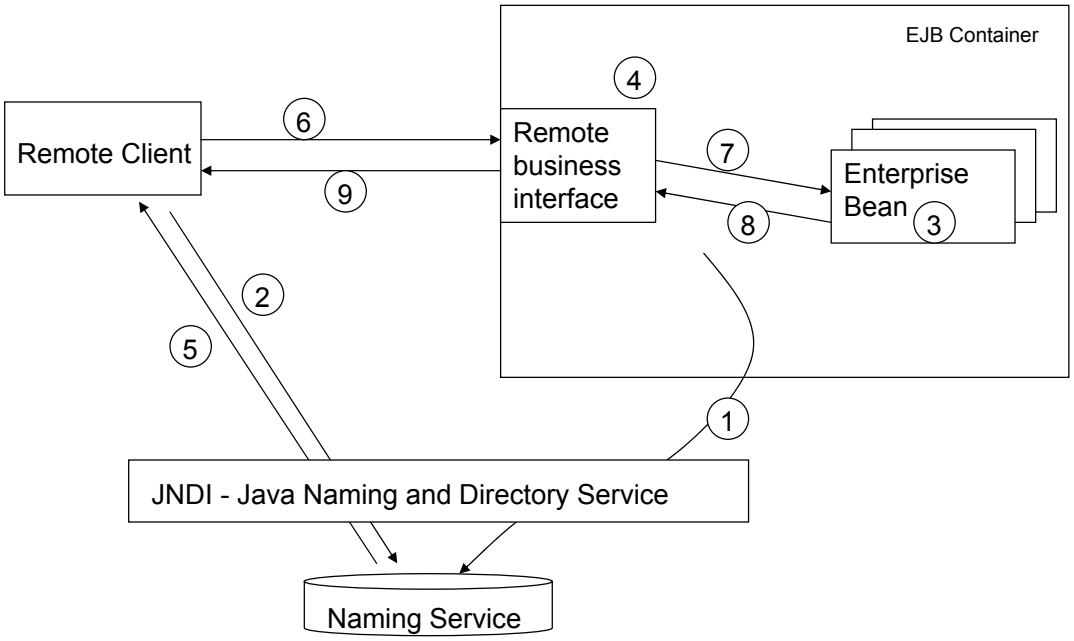
Asiakasohjelmat eivät kommunikoi suoraan EJB-komponentin kanssa vaan kommunikointi tapahtuu rajapintojen kautta. Rajapinta voi olla joko paikallinen rajapinta tai etärajapinta (ks. kuvio 3). Paikallista rajapintaa käytetään silloin kuin asiakasohjelma on samassa JVM (Java Virtual Machine) -koneessa kuin itse komponentti. Muuten on käytettävä etärajapintaa. (Lipitsäinen 2007, 47.)



KUVIO 3. EJB-komponentin rajapinnat (Lipitsäinen 2007, 48)

4.3.2 EJB-komponentin toiminta web-palvelimena

EJB-komponentin toiminta sekä yhteys asiakkaan, palvelun ja taustajärjestelmien välillä on esitetty kuviossa 4.



KUVIO 4. EJB-komponentin toiminta (Lipitsäinen 2007, 49)

Saadakseen yhteyden web-palveluun asiakasohjelma lähettää SOAP-pyynnön palvelun rajapinnalle ja rajapinta kutsuu EJB-säiliössä olevaa web-palvelua. Toiminnot suoritetaan seuraavassa järjestyksessä:

1. Java EE -sovellus ja EJB-komponentti asennetaan EJB-säiliö -alustaan, jolloin säiliö rekisteröi komponentin nimipalveluun JNDI:n (Java Naming and Directory Interface) avulla.
2. Asiakkaan halutessa käyttää web-palvelua löytyy se komponentin nimipalvelusta.
3. EJB-säiliö luo komponentin ilmentymän ja
4. komponentin rajapinnan ilmentymän.
5. EJB-säiliö palauttaa asiakkaalle viittauksen komponentin rajapinnan ilmentymään.
6. Asiakas lähettää web-palvelun operaation ja sen parametrit sisältävän SOAP-pyynnön komponentin rajapinnalle.
7. EJB-säiliö välittää metodikutsun itse komponentille.
8. Komponentti suorittaa operaatiota vastaavan metodin koodin, ja suorituksen jälkeen säiliö välittää metodin paluuarvon rajapinnan ilmentymälle.
9. Rajapinta palauttaa paluuarvon asiakasohjelmalle. (Mts. 50.)

5 TOTEUTETTAVAN JÄRJESTELMÄN VAATIMUKSET

5.1. Arkkitehtuuriset vaatimukset

Eventizer Oy vastasi järjestelmän suunnittelusta kokonaisuudessaan. Poikosen (2009, 6) laatimissa arkkitehtuurisissa vaatimuksissa määritellään tärkeimmät vaatimukset, jotka ohjasivat arkkitehtuurisuunnittelua. Niitä ovat tiedon keruu ilman verkkoyhteyttä, tiedon tuonti sovelluksesta ulkopuolisiin järjestelmiin sekä järjestelmän laajennettavuus ja yhteensopivuus.

Tiedon keruu ilman verkkoyhteyttä

Arvioinnit suoritetaan yleensä arvioitavien henkilöiden ja ympäristöjen luona. Tällöin ei aina ole mahdollista lähettää tietoja suoraan palvelimelle. Arvioijilla pitää olla mahdollisuus tallentaa arvioinnit paikallisesti ja lähettää ne myöhemmin palvelimelle. (Mts. 6.)

Tiedon tuonti sovelluksesta

Järjestelmään talletettuja tietoja tarvitaan myös sovelluksen ulkopuolisissa järjestelmissä. Järjestelmän täytyy tarjota mahdollisuus tietojen tuontiin eri tiedostomuodoissa. (Mts. 6.)

Laajennettavuus

Järjestelmää tullaan toimeksiantajan mukaan hyvin todennäköisesti laajentamaan myöhemmin. Järjestelmän täytyy sen vuoksi olla laajennettavissa ilman isoja muutoksia nyt kehitettävään järjestelmään. (Mts. 6.)

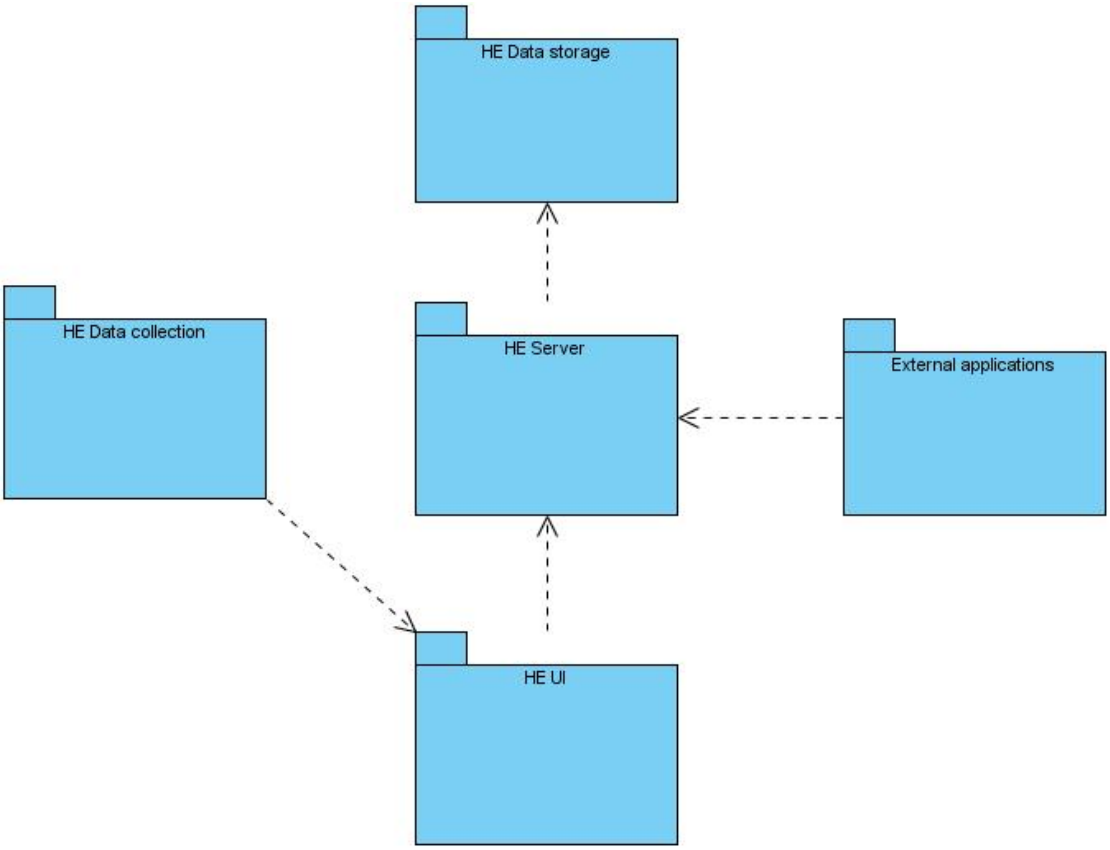
Yhteensopivuus

Järjestelmään talletettuja tietoja voidaan käyttää myös muissa sovelluksissa. Järjestelmän täytyy tarjota teknologiasta riippumattomat rajapinnat tiedon haulle. (Mts. 6.)

5.2 Arkkitehtuurikuvaus

5.2.1 Looginen näkymä

Loogisessa näkymässä (kts. kuvio 5) on kuvattu järjestelmän korkean tason osittamisen loogisiin osiin, kuten tietokantaan, palvelimeen ja käyttöliittymään. Tällä tasolla ei ole otettu vielä kantaa toteutustekniikoihin. Korkeimmalla tasolla järjestelmä koostuu tiedon keruusta (engl. HE Data collection), käyttöliittymästä (engl. HE UI), palvelimesta (engl. HE Server), tietovarastoista (engl. HE Data storage) ja ulkoisista sovelluksista (engl. External applications). (Poikonen 2009, 7.)

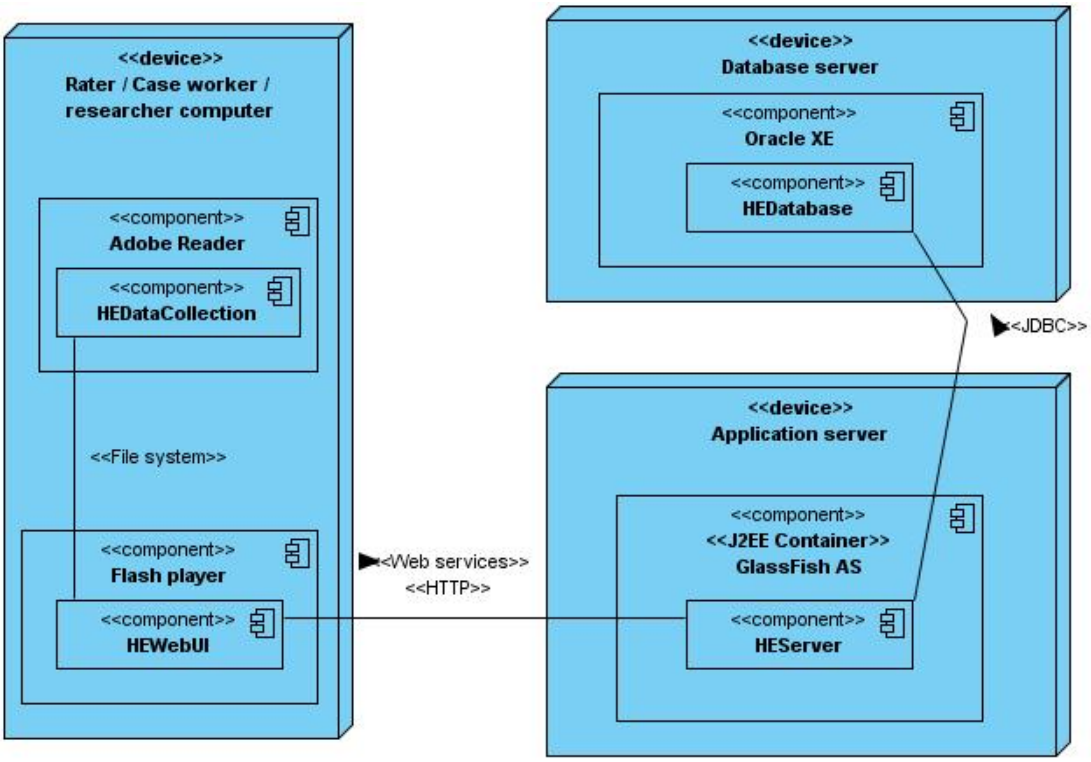


KUVIO 5. Järjestelmän looginen näkymä (Poikonen 2009, 7)

5.2.2 Ajonaikainen näkymä ja komponenttinäkymä

Ajonaikainen näkymä kuvaa järjestelmän ajonaikaisen konfiguraation, joka koostuu tietokoneresursseista, komponenteista ja niiden riippuvuuksista. Järjestelmä jakaantuu kolmeen eri tietokoneresurssiin. Kuviossa 6 esitetyt resurssit ovat loppukäyttäjän tietokone, sovelluspalvelin ja tietokantapalvelin. Kaaviossa esitetään myös tärkeim-

mät komponentit, joita ovat HEDataCollection, HEWebUI, HEServer ja HEDatabase. HEDataCollection-komponenttia käytetään tiedon keruussa. Sitä käytetään Adobe Reader -sovelluksen kautta ja se ei ole suoraan yhteydessä HEServer-komponenttiin. HEDataCollection toteutetaan PDF-lomakkeilla, jotka mahdollistavat tiedonkeruun ilman verkkoyhteyttä. (Poikonen 2009, 8.)



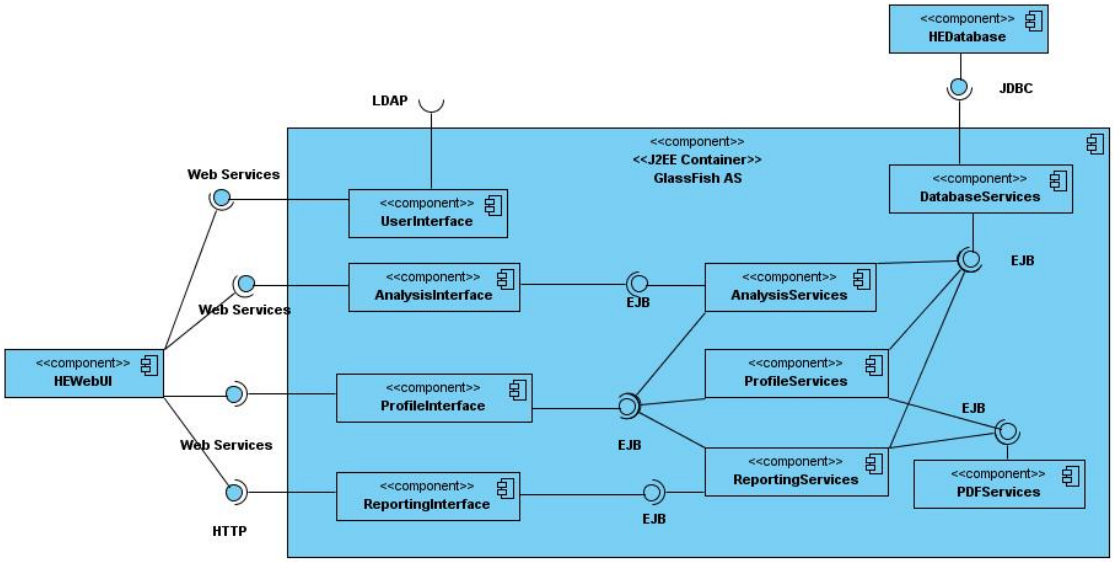
KUVIO 6. Järjestelmän ajonaikainen konfiguraatio (Poikonen 2009, 8)

HEWebUI-komponentti tarjoaa käyttöliittymän HEServerin tarjoaman tiedon lukemiselle ja muokkaamiselle. Se mahdollistaa myös tiedon lähettämisen HEServer-palvelimelle HEDataCollection-komponentilta. HEWebUI toteutetaan Adobe Flex -tekniikalla ja sitä käytetään Flash Player -sovelluksella. (Mts. 9.)

HEDatabase on relaatiotietokanta, joka sisältää kaiken järjestelmään tallennettavan datan. Tietokantaa voidaan käyttää myös samalta koneelta kuin HEServer-komponenttia. HEDatabase toteutetaan Oracle XE -tietokantaan. (Mts. 9.)

5.2.3 Palvelinkomponentit

Palvelin on järjestelmän tärkein osa. Se tarjoaa HTTP- ja web-palvelu -rajapinnat muille komponenteille ja yhteyden tietokantaan. Palvelin toteutetaan EJB-teknologialla. Sovelluspalvelimena käytetään GlassFish AS:ä. Palvelin toteuttaa kuvissa 7 esitetyt web-palvelut: profiilipalvelut, analyysipalvelut ja PDF-palvelut sekä niiden rajapinnat. (Poikonen 2009, 9.)



KUVIO 7. Järjestelmän palvelinkomponentit (Poikonen 2009, 9)

Profiilipalvelut tarjoavat operaatioita profiilidatan hakuun ja muokkaukseen. Ne myös muuntavat profiilidatan oikean muotoiseksi tallennusta varten. Profiilipalveluita käytetään profiilirajapinnan kautta. Profiilirajapinta mahdollistaa profiilien hakemisen, tallentamisen ja muokkaamisen. Rajapinta vastaanottaa web-palvelu -kutsut ja välittää ne profiilipalveluille varsinaista datan käsittelyä varten. (Mts. 10.)

Analyysipalvelut tarjoavat operaatioita analyysidatan hakuun. Analyysipalveluita käytetään analyysirajapinnan kautta. Analyysirajapinta vastaanottaa web-palvelu -kutsut ja välittää ne analyysipalveluille, jossa tieto muunnetaan oikeaan muotoon tietokantahakuja varten. (Mts. 10.)

PDF-palveluita käytetään PDF-dokumenttien hallintaan. Ne purkavat ja täyttävät profiilipalveluiden käyttämät PDF-tiedostot. PDF-palveluilla ei ole julkista rajapintaa, vaan sitä kutsutaan järjestelmän sisältä profiilipalveluista. (Mts. 10.)

6 PALVELUN TOTEUTUS

6.1 Projektityöskentely

6.1.1 Scrum-menetelmä

Projektityöskentely toteutettiin ketterän ohjelmistomenetelmän Scrumin mukaisesti. Scrum-menetelmässä projektiin osallistuvat jaetaan kolmeen eri rooliin: *tuotteen omistaja* (engl. Product Owner), *tiimi* (engl. Scrum Team) ja *Scrum-mestari* (engl. Scrum Master). Tuotteen omistaja edustaa projektin asiakasta. Scrum-tiimi on projektissa työskentelevä itsenäinen muutaman hengen tiimi. Tiimissä on projektipäällikköä vastaava Scrum-mestari, joka hoitaa kommunikoinnin tiimin ulkopuolelle. Scrum-mestarilla ei ole suoraan määräysvaltaa muihin tiimin jäseniin vaan päätökset tehdään yhdessä. (Schwaber 2003, 6–7.)

Tuotteen työlista (engl. Product Backlog) sisältää Scrum-menetelmällä toteutettavan sovelluksen vaatimukset. Tuotteen työlistalta voidaan toteutettavia vaatimuksia viedä julkaisun työlistalle, joka siis sisältää tietyssä julkaisussa toteutettavat vaatimukset. Projekti jaetaan useisiin noin neljän viikon mittaisiin vaiheisiin eli sprintteihin. Sprintti alkaa suunnittelupäivällä, jossa Scrum-tiimi päättää yhdessä tuotteen omistajan kanssa, mitä tuotteen vaatimuksia tullaan toteuttamaan alkavan sprintin aikana. Vaatimuksista tarkennetut tehtävät muodostavat *sprintin työlistan* (engl. Sprint Backlog), mikä toimii myös sprintin runkona. (Mts. 12–13.)

Sprintin kuluessa tiimi pitää päivittäisiä Scrum-palavereita, joissa Scrum-mestari kysyy jokaiselta tiimin jäseneltä kolme kysymystä:

1. Mitä olet tehnyt edellisen Scrum-kokouksen jälkeen?
2. Mitä aiot tehdä seuraavaan Scrum-kokoukseen mennessä?
3. Mikä on hankaloittanut työtäsi?

Sprintin loputtua tiimi esittelee toteutetut sprintin tulokset asiakkaalle. Tämän jälkeen tiimi pitää vielä keskenään arviointikeskustelun, jossa arvioidaan, miten kulunut sprintti onnistui ja mitä voitaisiin tehdä jatkossa paremmin. (Mts. 7–9.)

6.1.2 Projektin vastuu- ja vaihejako

Projektin toimeksiantajan edustaja toimi projektissa tuotteen omistajan roolissa. Scrum-mestari oli projektin suunnittelusta vastannut ohjelmistosuunnittelija Eventizer Oy:sta. Varsinaiseen Scrum-tiimiin kuului Scrum-mestarin lisäksi kaksi käyttöliittymästä vastaavaa ohjelmojaa, tietokantaohjelmoija ja palvelinpuolen ohjelmoija. Lisäksi Scrum-mestari toteutetti sovelluksessa käytettävät PDF-muotoiset arviointilomakkeet ja osallistui myös jonkin verran palvelimen toteutukseen. Sovelluksen testaukseen saatiin resursseja toimeksiantajalta sekä Esteetön Koti -hankkeesta Jyväskylän ammattikorkeakoulun edustajalta.

Projekti jaettiin kolmeen noin kuukauden mittaiseen sprinttiin. Ensimmäisessä vaiheessa laitettiin projektin perusrakenteet. Toisessa vaiheessa toteutettiin profiilien käsittely ja kolmannessa vaiheessa tietojen analysointi.

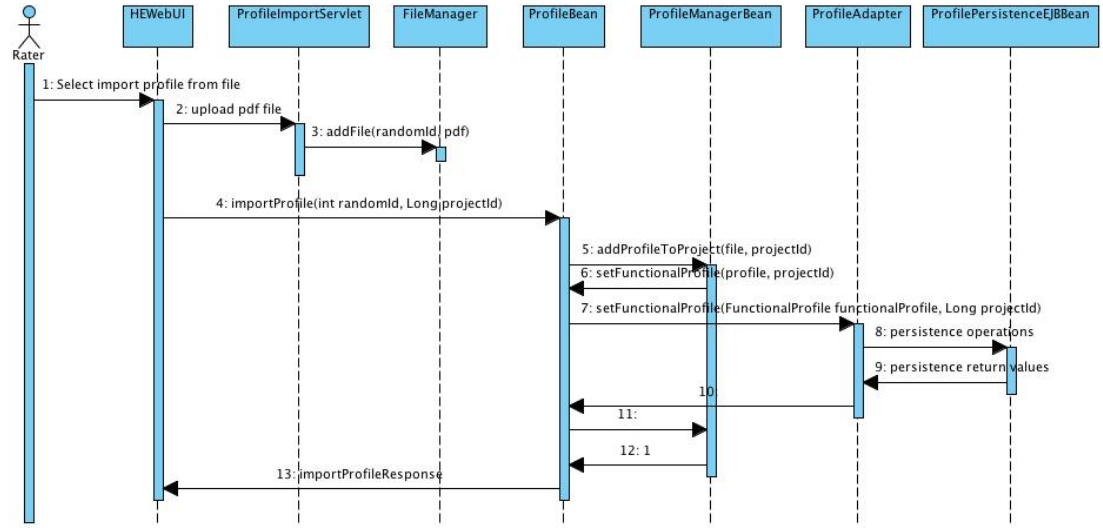
6.2 Palvelujen rajapintojen toteutus

HE-palvelin tarjoaa HTTP- ja web-palvelu -rajapinnat muille järjestelmän komponenteille. Palvelin jakautuu neljään eri osioon: profiilipalveluihin, raportointipalveluihin, analysointipalveluihin ja PDF-palveluihin. Näistä profiilipalvelut, analysointipalvelut ja raportointipalvelut näkyvät ulospäin WSDL-rajapintoina. PDF-palveluita kutsutaan vain palvelimen sisältä.

Koska web-palvelut toteutettiin EJB 3.0-arkkitehtuurin mukaisesti, ei erillisiä rajapintakomponentteja tarvinnut luoda, vaan web-palvelut toteuttavien EJB-luokkien etärajapinnat toimivat palvelujen ulkoisina rajapintoina. Näissä rajapinnoissa määritellään EJB-luokan toteuttamat web-palvelu -operaatiot.

EJB-luokkien etärajapintojen lisäksi palvelussa on myös servletinä toteutettuja rajapintoja. Servletit toimivat rajapintoina tiedostonsiirtoa vaativille web-palvelu

-operaatiolle. Tällaisissa operaatiossa palvelimelle lähetetään kaksi erillistä kutsua. Ensimmäisessä kutsussa lähetetään tiedosto palvelimelle servletin kautta. Tämän jälkeen lähetetään web-palvelu -kutsu EJB-luokan etärajapinnalle. Kuviossa 8 on esimerkki tällaisesta kutsusta.



KUVIO 8. Sekvenssikaavio profiilin tuomisesta palvelimelle (Poikonen 2009, 14)

Profiilin tuonti PDF-lomakkeelta palvelimelle tapahtuu kutsumalla asiakasohjelmasta servletiä, joka vastaanottaa lähetetyn PDF-tiedoston. Servlet tallettaa tiedoston FileManager-luokan avulla, jonka jälkeen asiakasohjelma kutsuu ProfileBeanin importProfile-metodia. ProfileBean vastaa tiedon käsittelystä ja ProfileAdapterilla oikeaan muotoon muutetut tiedot tallennetaan lopuksi ProfilePersistenceEJBBeanilla.

6.3 Web-palveluiden toteutus

Kaikki järjestelmän web-palvelut toteutettiin tilattomina istuntopohjaisina EJB-komponentteina, jotka toteuttavat erillisen luokkaa varten tehdyn rajapinnan määrittelemät operaatiot.

6.3.1 Profiilipalvelu

Profiilipalvelu toteutettiin erillisellä EJB-komponetilla, ProfileBeanilla. Projektin ensimmäisessä vaiheessa ProfileBeaniin toteutettiin kansioita ja projekteja koskevat metodit:

- getProjects
- getFolders
- setProject
- setFolder
- removeProject ja
- removeFolder.

Näitä kaikkia metodeja kutsutaan suoraan asiakasohjelmasta. Kutsuttaessa operaatioita WSDL:n kautta luodaan palvelimella aina uusi ProfileBean-ilmentymä, joka suorittaa kyseistä operaatiota vastaavan metodin. ProfileBean huolehtii datan käsittelystä, ja jokaisen metodin alussa luodaan instanssi adapteriluokasta, jossa injektoidaan ProfilePersistenceEJBBean persistointia varten. ProfileBean kutsuu adapteri-luokkaa, joka muuttaa web-palvelussa käytettävät tietotyypit entiteeteiksi eli kohdepohjaisiksi EJB-komponenteiksi ja välittää persistointipyynnöt ProfilePersistenceEJBBeanille.

Toisessa vaiheessa toteutettiin profiilien käsittely sovelluksessa. Käyttäjän täytyi pystyä tuomaan PDF-lomakkeelle täytettyjä arviointiprofiileja järjestelmään sekä katsella ja muokata profiileja sovelluksessa. Näille toiminnoille tehtiin web-palvelu -operaatiot toteuttavat metodit ProfileBean-komponenttiin. Tässä vaiheessa lisättyjä metodeja olivat:

- copyMoveProfile
- getProfiles
- getFunctionalProfile
- getEnvironmentalProfile
- setEnvironmentalProfile
- setFunctionalProfile
- removeProfile ja
- importProfile.

Myös näissä metodeissa menetellään samalla tavalla kuin edellisenkin vaiheen metodeissa, eli asiakasohjelma kutsuu suoraan ProfileBeanin metodeja joissa suorite-

taan varsinainen tiedon käsittely. Tämän jälkeen tiedot muunnetaan oikeaan muotoon ProfileAdapterilla ja tallennetaan ProfilePersistenceEJBBeanilla.

6.3.2 Analyysipalvelu

Analyyseihin liittyvät metodit luotiin Analysis-moduuliin ja sinne luotiin vastaavat luokat kuin Profile-moduulissakin, eli AnalysisBean, AnalysisAdapter ja AnalysisPersistenceEJBBean. AnalysisBeanille luotiin metodit:

- getScores
- getStatistics
- getFunctionalLimitations
- getOrderedObstacles
- setProfilesToAnalysis
- getAnalysisProfiles ja
- getAnalysisFolders.

AnalysisBeanin operaatiot ovat huomattavasti yksinkertaisempia kuin ProfileBeanin, sillä suurin osa operaatiosta on vain tiedon hakua tietokannassa olevista näkymistä. Entiteettien luonti oli taas hieman monimutkaisempaa kuin Profile-moduulissa, sillä tietokannan näkymät eivät suoraan soveltuneet entiteeteiksi, vaan näkymien tiedot jaettiin asiakasohjelmalle sopiviksi osiksi jo palvelimella.

6.4 Taustajärjestelmät

Järjestelmässä ei ole yhteyksiä erillisiin taustajärjestelmiin. Ainoat yhteydet web-palvelun toteuttavista EJB-komponenteista ulospäin ovat tietokantaoperaatioiden JDBC-yhteydet tietokantaan. Tietokantayhteydet toteutettiin entiteettien avulla. Tarvittavat entiteetit pystyttiin tuomaan EJB-moduuliin suoraan tietokannasta NetBeansin tuonti-toiminnolla. Entiteettien suhteisiin täytyi kuitenkin tehdä pieniä muutoksia koska kaikkea tietoa ei haluttu hakea kerralla tietokannasta (ks. liite 1).

Tietokannassa Housing Enabler -menetelmän pistearvoja sisältävät taulut ovat staattisia ja niihin kohdistetaan vain hakuoperaatioita. Muut taulut sisältävät muokattavaa

tietoa, joihin kohdistetaan sekä haku, että muokkausoperaatioita. Nämä taulut sisältävät Housing Enabler -pisteiden laskemisessa käytettävää tietoa. HE_PARTY_T-taulu on käyttäjien hallinnointia varten. Sisäänkirjautumista ja yhteyden varmennusta ei kuitenkaan toteutettu vielä tässä projektissa joten taulusta ei luotu entiteettiä. HE_PROFILE_DATA_SECTION_T on liitostaulu HE_PROFILE_DATA_T- ja HE_SECTION_T-tauluille, joiden välillä on monesta moneen suhde. Entiteettien suhteet voidaan määritellä EJB:ssä suoraan annotaatioilla, joten myöskään tätä liitostaulua ei toteutettu entiteettinä EJB-moduulissa.

6.5 Tietoturva

HE-palvelimen käyttöä ei ole rajoitettu sisäiseen verkkoon, eli palvelimeen voidaan olla yhteydessä mistä tahansa internetin kautta. Tästä johtuen asiakassovelluksen ja palvelimen välinen yhteys pitää pystyä varmentamaan. Varmennusta ei kuitenkaan toteutettu tässä projektissa, vaan se sisällytettiin projektin jatkokehitykseen.

7 ARVIOINTI JA YHTEENVETO

7.1 Yleinen arvio toteutuksesta

Kaikki projektin tavoitteet saavutettiin, ts. tavoitteena ollut järjestelmä saatiin toteutettua kokonaisuudessaan. Testauspalvelimen käyttöönoton siirtymisestä johtuen testausta jouduttiin siirtämään hieman myöhemmäksi, mikä myöhästytti projektin lopullista valmistumista pari viikkoa.

Projektin toteutuksessa ei käytetty Scrum-menetelmää kirjaimellisesti aiemmin esitetyn mukaan (vrt. 6.1.1 Scrum-menetelmä). Projekti jaettiin kolmeen eri sprinttiin, jotka suunniteltiin ja katselmoitiin erikseen. Päivittäisiä Scrum-palavereja ei pidetty, koska Scrum-mestarille ei oltu resurssoitu aikaa projektin hallintaan juurikaan. Projektiryhmä työskenteli myös eri aikoina ja fyysisesti toisista erillään, joten päivittäisen kaikille sopivan ajankohdan löytyminen olisi ollut melko haasteellista.

Väljästä käyttötavasta huolimatta Scrum-menetelmän valinnasta oli useita hyötyjä projektin toteutuksessa. Konkreettisimmat hyödyt tulivat esille työnjaon joustavuudessa. Projektin toteutus oli lähes kokonaan JAMK:n opiskelijoiden vastuulla ja tehtävien vaatimaa aikaa oli vaikeaa arvioida etukäteen. Scrum-menetelmän kautta voitiin jokaisen vaiheen jälkeen arvioida työn etenemistä yhdessä. Esimerkiksi kun projektin toisen vaiheen jälkeen huomattiin profiilipalveluiden vaativan arvioitua enemmän aikaa, ja vastaavasti käyttöliittymän toteutuksen etenevän arvioitua nopeammin, voitiin hyvin helposti siirtää toinen käyttöliittymäohjelmoijista toteuttamaan analyysipalveluita. Näin kaikki vaatimukset saatiin toteutettua ajallaan.

Toinen merkittävä etu Scrum-menetelmän käyttämisestä oli, että toimeksiantaja pääsi nopeasti näkemään ja kokeilemaan järjestelmän ensimmäisiä toimivia osia. Tämä auttoi saamaan palautetta järjestelmästä jokaisen vaiheen jälkeen ja se tarjosi mahdollisuuden muokata vaatimuksia vielä projektin aikanakin.

Kolmas Scrum-menetelmän hyöty nousee tiiviistä yhteydenpidosta projektin eri henkilöiden välillä. Läpinäkyvyys ja tiivis kommunikointi takasivat sen, että mm. tietokantamäärittelyissä esiin nousseet ongelmat havaittiin nopeasti, jolloin ongelmiin päästiin paneutumaan välittömästi.

7.2 Toteutuksessa havaitut ongelmat

Alkuperäisen projektiaikataulun puitteissa ei suunnitelmista huolimatta saatu käyttöön lainkaan erillistä testipalvelinta, joten testausta ei ehditty oikeastaan aloittamaan vielä projektin aikana. Testipalvelimen puuttumisesta johtuen havaittuja vikoja jouduttiin korjaamaan vielä viimeisen sprintin jälkeen, ja projektin päätös venyi näin muutaman viikon aiottua pidempään. Korvaavaa palvelinta ei ollut saatavilla, joten projektiryhmä ei voinut asialle mitään.

Asioiden esittäminen tarpeeksi selkeästi toimeksiantajalle osoittautui myös haastavaksi. Epäselvyyksiä oli mm. web-sovelluksen ja työpöytäsovelluksen eroissa sekä sovelluksen käyttöliittymän ja arviointilomakkeiden erottamisessa toisistaan. Käytännössä yhteisen ammatillisen kielen puuttuminen vaikutti työstä saatavan pa-

lautteen määrään, toimeksiantaja ei kyennyt tietoteknisen osuuden arviointiin omatoimisesti.

7.3 Jatkokehitys

Projektin tuloksena saatiin alkuperäisten toiveiden mukainen, periaatteessa valmis järjestelmä. Toimeksiantajalla oli kuitenkin vielä kehitystoiveita, joita tullaan lisäämään järjestelmään jatkoprojekteissa. Vaikka järjestelmä on valmis käyttöönotettavaksi, yksi asiakkaan tärkeimpiä kehitystarpeita sovellukselle oli käyttöoikeus organisaatiokohtaisesti. Tämän projektin resurssit eivät riittäneet käyttäjähallinnan toteutukseen. Käyttäjähallinnan toteutus onkin tarkoitus aloittaa heti tämän projektin päätyttyä. Järjestelmän suunnittelussa ja toteutuksessa on otettu huomioon käyttäjähallinta jo alusta alkaen siten, että käyttäjähallinnan lisäys onnistuu mahdollisimman pienillä muutoksilla jo kehitettyyn järjestelmään.

Toinen järjestelmän käyttöönottoa haittaava ongelma on tuotantopalvelimen puute. Lopullista tuotantopalvelinta ei asennettu tämän projektin aikana, koska toimeksiantajalla ei ollut vielä palvelinympäristöä kehitettynä järjestelmää varten. Projektin aikana kehitys tapahtui Eventizer Oy:n kehityspalvelimella. Järjestelmän saaminen tuotantokäyttöön vaatii järjestelmän asentamisen tuotantokäyttöön tarkoitetulle palvelimelle.

LÄHTEET

DeMichiel, L. & Keith, M. 2006. JSR 220: Enterprise JavaBeansTM,Version 3.0 EJB Core Contracts and Requirements. Sun Microsystems. Viitattu 12.11.2009. Http://jcp.org/aboutJava/communityprocess/final/jsr220/, Download.

ESKO-hanke. 2009. ESKO esteetön koti ikääntyneiden ja erityisryhmien asumiseen -hanke. Viitattu 31.8.2009. Http://www.jamk.fi/tutkimus/projekteja/esteetonasuminen/esko.

ESOK-hanke. 2008. Susanne Iwarsson interview. Viitattu 31.10.2009. Http://esok.jyu.fi/esittely/en/material/enabler/.

ET Elämisen Tuki Oy. n.d. RAES-rakennuksen esteettömyyden kartoitus. Viitattu 30.9.2009. Http://www.elamisentuki.fi.

Iwarsson, S. & Slaug, B. 2008. Housing Enabler. Lund: KFS i Lund.

Knutson, J. & Kreger, H. 2002. Web Services for Java EE, Version 1.0. IBM Corporation. Viitattu 12.11.2009. Http://jcp.org/aboutJava/communityprocess/final/jsr109/, Download.

Lipitsäinen, A. 2007. Java EE -arkkitehtuuri. Kurssimateriaali 30.8.2007. HAAGA-HELIA ammattikorkeakoulu. Viitattu 19.11.2009. Http://myy.haaga-helia.fi/~ict4td020/ict4td020b/arkkitehtuuri/j2ee.pdf.

MacKenzie, M., Laskey, K., McCabe, F., Brown, P. & Metz, R. 2006. Reference Model for Service Oriented Architecture 1.0. Viitattu 15.9.2009. Http://www.oasis-open.org/committees/download.php/19679/soarm-cs.pdf.

Papazoglou, M. 2008. Web Services: Principles and Technology. Harlow: Pearson Education.

Poikonen, M. 2009. ESKO-project Architecture design. Eventizer Oy.

Schwaber, K. 2004. Agile Project Management with Scrum. Washington: Microsoft Press.

Shannon, B. 2006. Java™ Platform, Enterprise Edition (Java EE) Specification, v5. Sun Microsystems. Viitattu 12.11.2009. Http://jcp.org/aboutJava/communityprocess/final/jsr244/, Download.

Slaug, B. 2001. Who invented the Enabler concept? Viitattu 12.9.2009. Http://www.enabler.nu, FAQ.

LIITTEET

Liite 1. Järjestelmän tietokantakaavio.