TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Matti Rintala

TILAPÄISTUNNUSTEN HALLINTASOVELLUS

Työn ohjaaja Lehtori Erkki Hietalahti
Työn teettäjä Tampereen ammattikorkeakoulu, valvojana DI Petteri Jekunen
Tampere 2008

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka
Rintala, Matti
Tilapäistunnusten hallintasovellus
Tutkintotyö 32 sivua, 5 liitesivua
Työn ohjaaja Lehtori Erkki Hietalahti
Työn teettäjä Tampereen ammattikorkeakoulu, valvojana DI Petteri Jekunen
Huhtikuu 2008
Hakusanat Java, Servlet, Portlet, AJAX

TIIVISTELMÄ

Tampereen ammattikorkeakoulussa vieraileville luennoitsijoille ja muille vieraille, jotka tarvitsevat TAMK:n tietojärjestelmät käyttöönsä, on mahdollista luoda tilapäinen käyttäjätunnus. Tämän opinnäytetyön ohjelmistolla korvataan erikoistaitoja vaativa komentorivipohjainen ohjelma modernilla web-sovelluksella. Helppokäyttöisellä graafisella käyttöliittymällä varustettu sovellus on mahdollista luovuttaa nykyistä suuremman käyttäjäkunnan haltuun, jolloin asiakaspalvelun laatu ja nopeus paranee.

Tilapäistunnusten hallintasovellus integroituu osaksi TAMK:n intranet-portaalia. Tämän vuoksi ohjelmisto on J2EE-pohjainen web-sovellus, jossa yhdistellään portletteja, servlettejä sekä JSP-sivuja. Käyttöliittymän toteutuksessa on käytetty myös JavaScript-kieltä ja sillä toteutettua jMaki-kirjastoa. Ohjelma tarjoaa liitynnät LDAP- ja AD-hakemistoihin sekä MySQL-tietokantaan.

Sovellus vaatii vielä hieman jatkokehitystä, erityisesti käyttöliittymän visuaalisen ilmeen muokkaamista. Tämän opinnäytetyön aihe oli erittäin monipuolinen ja haastava. Työn teknisessä toteutuksessa joutui perehtymään useisiin web-sovelluksissa käytettyihin tekniikoihin. Osa työssä hyödynnetyistä menetelmistä oli suhteellisen uusia, joten lähdemateriaalin löytämisessä oli omat haasteensa.

TAMK University of Applied Scienses
Computer engineering
Software engineering
Rintala, Matti

Temporary account management software
Engineering thesis
32 pages, 5 appendices
Thesis supervisor
Lecturer Erkki Hietalahti
Comissioning company
TAMK, MSc Petteri Jekkunen
April 2008
Keywords
Java, Servlet, Portlet, AJAX

ABSTRACT

TAMK University of Applied Sciences has lots of visiting lectures in a year. These lecturers might need short time access to TAMK's information systems so they need a temporary user account. Currently these accounts are reserved and administrated by using UNIX command line application. Usage of the software requires special knowledge and for that reason only TAMK's Computer Center's personnel are operating it. Purposes of this thesis work are design and implement graphical web application which is easy to use. Easier reservation application can be operated by wider range of users which improves quality of service for visitors.

Graphical temporary account management software integrates as a part of TAMK's intranet portal. That technical requirement limits implementation techniques of the application to Java based web applications. This thesis work's application combines portlets, servlets and JSP pages. Also AJAX features are used to produce more pleasant user interface by using jMaki framework. The application uses LDAP and AD directories and MySQL database as external data sources.

SISÄLLYS

KÄYTETYT TERMIT JA LYHENTEET

LDAP	Lightweight Directory Access Protocol on hakemistopalveluiden käyttämiseen tarkoitettu protokolla.
AD	Active Directory on Microsoftin näkemys LDAP-hakemistosta.
InnoDB	MySQL:n tietokantamoottori, joka tukee transaktioita ja vierasavaimia.
J2EE	Java 2 Enterprise Edition on ohjelmistoalusta, jolla voidaan luoda sovelluspalvelimella suoritettavia web-sovelluksia.
Portlet	Siirrettävä web-sovellus, joka muodostaa osan portaalin käyttöliittymästä.
Servlet	Java web-sovellus, joka suoritetaan sovelluspalvelimella.
JSP	JavaServer Pages on J2EE:n tekniikka, jolla voidaan tuottaa dynaamisia www-sivuja.
AJAX	Asynchronous JavaScript And XML on yleisnimitys tekniikoille, joilla voidaan tuottaa rikkaita käyttöliittymiä www-sovelluksiin.
MVC	Model View Controller -termillä viitataan ohjelmointimalliin, jossa tieto, tiedon esittäminen ja sovelluslogiikka on eriytetty toisistaan.
JNDI	Java Naming and Directory Interface on Javan rajapinta, jolla käytetään esimerkiksi hakemistopalveluita.
JDBC	Java Database Connectivity on ohjelmistorajapinta, jota käytetään tietokantayhteyksissä.

1 JOHDANTO

Tampereen ammattikorkeakoulussa käy vuosittain useita vierailevia luennoitsijoita, jotka tarvitsevat pääsyn TAMK:n tietojärjestelmiin. Tällaisia lyhytaikaisia tarpeita varten vierailijalle on mahdollista antaa käyttöön tilapäinen käyttäjätunnus, jolla mahdollistetaan pääsy tärkeimpiä tietojärjestelmiä. Tilapäistunnusten luomiseen on olemassa komentorivipohjainen sovellus, jonka käyttö vaatii erikoisosaamista. Tästä syystä TAMK:n tietokonekeskuksen henkilöstö huolehtii tunnuksen varaamisesta.

Tässä opinnäytetyössä suunnitellaan komentorivisovelluksen toiminnallisuuden toteuttava Tampereen ammattikorkeakoulun intranettiin integroituva graafinen ohjelma tilapäistunnusten varaamiseen. Suunnittelun lähtökohtana on sovelluksen helppokäyttöisyys, joka mahdollistaa tunnusten myöntämisen hajauttamisen Tietokonekeskuksen organisaation ulkopuolelle. Samalla otetaan ensimmäinen askel nykyisten tunnushallintatyökalujen modernisoimiseksi.

2 TILAPÄISTUNNUSTEN KOMENTORIVISOVELLUS

Tässä luvussa tarkastellaan olemassa olevan tunnusvaraussovelluksen toimintaa ja toteutusta. Tämän työn kannalta oli oleellista kartoittaa lähtökohdat, jotta olemassa oleva toiminnallisuus voidaan toteuttaa uudessa graafisessa versiossa.

2.1 Komentorivisovelluksen yleiskuvaus

Tällä hetkellä tilapäistunnusten hallinnointiin käytetään UNIX-shellissä toimivaa komentorivipohjaista sovellusta. Ohjelmalla pystytään suorittamaan kaikki tarvittavat toimenpiteet, jotka liittyvät tähän tunnustyyppiin. Tunnuksista voidaan eri parametreilla listata vapaat, varatut tai kaikki. Tässä suppeassa tulostusmuodossa vapaista tunnuksista näytetään vain tunnuksen nimi ja toimipiste, jossa se on käytettävissä. Varatuista tunnuksista näytetään myös viimeinen voimassaolopäivä. Lisäparametrilla saatavassa laajassa tulosteessa ilmaistaan myös tiedot tunnuksen tekijästä, käyttäjästä, vastuuhenkilöstä sekä tunnuksen

käyttötarkoitus. Myös yksittäisen tunnuksen tietoja on mahdollista tarkastella. Kuvassa 1 on esitetty esimerkkitulostus yhden tunnuksen tiedoista komentorivisovelluksella tarkasteltuna.

```
delta:~ #> temppi tunnus24 -l

tunnus24, Viimeinen voimassaolopäivä 20080218, Teiskontie, Käyttäjä Matti R, Vastuuhenkilö Matti R, Tekijä root
```

Kuva 1 Komentorivisovelluksen tulostus yhden tunnuksen tiedoista

Tunnushallintajärjestelmässä olevien tietojen tarkastelun lisäksi on olennaista pystyä varaamaan tunnuksia niitä tarvitseville vierailijoille. Varaamiseen tarvitaan tiedot viimeisestä voimassaolopäivästä, tunnuksen vastuuhenkilöstä, käyttäjästä sekä käyttötarkoituksesta. Tieto tunnuksen tekijästä tallentuu automaattisesti varaajan käyttäjätunnuksen muodossa. Varatusta tunnuksesta on mahdollista tulostaa paperiversio käyttäjän muistin tueksi. Esimerkki tunnuksen varaamisesta komentorivisovelluksella on esitetty kuvassa 2, jossa ohjelman nimen perään on annettu varattava tunnus sekä viimeinen voimassaolopäivä.

```
delta:~ #> temppi tunnus24 20080324

--- Syötä tunnuksen vastuullinen henkilö -----
Valitse vaihtoehto:
(1) Matti R
(2) Sirpa Nurminen
..tai kirjoita omavalintainen syöte:
Matti Rintala

--- Syötä tunnuksen käyttäjä -----
Valitse vaihtoehto:
(1) Matti R
(2) Vierailija
..tai kirjoita omavalintainen syöte:
Ville Vierailija

--- Syötä tunnuksen käyttötarkoitus -----
Valitse vaihtoehto:
(1) Tilapäistunnus
(2) Testailua
..tai kirjoita omavalintainen syöte:
Testailua

Päivitettiin tietokantaan.
Kotihakemisto putsattiin.

Käyttäjätunnus on tunnus24 ja salasana enot5asiaZ


Tulostetaanko(cc1) (k/e) ?

e
```

Kuva 2 Tunnuksen varaaminen komentorivisovelluksella

2.2 Komentorivisovelluksen tekninen toteutus

Nykyinen hallintasovellus on toteutettu merkkijonojen manipulointiin hyvin soveltuvalla Perl-ohjelmointikielellä. Tilapäistunnusten hallintaan liittyvä logiikka on kirjoitettu yhteen tiedostoon, mutta lisäksi ohjelma käyttää hyväkseen tunnushallinnan yleisiä Perl-moduuleita. Niissä on toteutettu tunnushallintasovelluksien vaatimia yleisiä komponentteja, joilla voidaan esimerkiksi kommunikoida erilaisten tietovarastojen kanssa. Lisäksi tunnushallinnan moduulit sisältävät tilapäisiä tietorakenteita, joihin voidaan hakea sisältöä paikallista tutkimista tai muokkaamista varten. Tämän työn sovelluksessa ei hyödynnetä olemassa olevia Perl-moduuleita millään tavalla, koska niihin ei ole olemassa rajapintaa intranet-portaalista.

Tilapäistunnuksiin liittyvää tietoa on tallennettu tunnushallintatietokantaan, LDAP- ja AD-hakemistoihin. Näiden ulkoisten tietolähteiden lisäksi rajapintana toimii tiedostojärjestelmä, jonka avulla siivotaan tunnuksen kotihakemisto. Tämä vaihe tehdään tunnuksen varaamisen yhteydessä.

3 GRAAFISEN SOVELLUKSEN ASIAKASVAATIMUKSET

Graafisen sovelluksen tulee toteuttaa vähintään samat ominaisuudet ja toiminnallisuus kuin komentoriviversion. Ohjelmiston käyttöä voidaan laajentaa sellaisille käyttäjille, joille helppokäyttöisyys ja suoraviivaisuus ovat tärkeimpiä ominaisuuksia monipuolisuuden sijasta. Tämän vuoksi käyttöliittymä tulee mahdollisuuksien mukaan suunnitella helppokäyttöiseksi ja käyttäjää opastavaksi. Ohjelmistossa käytettyjen tekniikoiden on oltava TAMK:n intranet-portaalissa toimivia Java-sovelluksia. Lisäksi käyttöliittymän toteutuksessa tulee soveltaa AJAX-tekniikoita, jotta sovellus saadaan toimimaan mahdollisimman nopeasti loppukäyttäjän näkökulmasta tarkasteltaessa.

4 WEB-SOVELLUSTEN TEORIAA

Seuraavissa aliluvuissa luodaan katsaus tämän opinnäytetyön kannalta oleellisiin web-sovellusten komponentteihin. Asiakasvaatimuksista johtuen käsitellyt aiheet koostuvat Java-pohjaisista tekniikoista ja niihin liittyvistä palvelinohjelmistoista.

4.1 Sovelluspalvelin

Sovelluspalvelin on ohjelmisto, joka toteuttaa HTTP-protokollan liikennöinnin lisäksi mahdollisuuden suorittaa eri tekniikoilla toteutettuja palvelinpään sovelluksia. Sovelluspalvelimelta edellytetään, että se toteuttaa Sun Microsystemsin J2EE-spesifikaation riittävän täydellisesti. Ohjelmistokehityksen kannalta sovelluspalvelin helpottaa ohjelmointityötä, koska kehittäjän ei tarvitse huolehtia alemman tason logiikasta. Näin kaikki kehitykseen käytetyt resurssit voidaan keskittää olennaiseen ja hyödyntää sovelluspalvelimen tarjoamia rajapintoja. /2, s. 29–30./

Tyypillinen palvelin tarjoaa erilaisia komponenttisäiliöitä (container), joissa voidaan esimerkiksi suorittaa J2EE-sovelluksia. Tämän työn kannalta oleellisimmat säiliöt ovat servlet- ja portlet-containerit. Lisäksi sovelluspalvelimet saattavat tarjota rajapinnan ulkoisten tietovarastojen käyttöön, esimerkiksi tietokantoja sekä hakemistopalveluita. Palvelimen tarjoamaa tietokantojen yhteysallasta on helppo käyttää kehitettävästä ohjelmistosta, eikä kehittäjien tarvitse miettiä omia ratkaisuja, jolloin suunnittelu on suoraviivaisempaa.

4.2 Servlet-sovellukset

Servlet on palvelimen Java-virtuaalikoneessa suoritettava sovellus. Näitä ohjelmia käytetään tyypillisesti tuottamaan dynaamisia HTML-sivuja sekä suorittamaan operaatioita erilaisten tietovarastojen kanssa. Käyttäjän www-selaimesta palvelimelle tulevat pyynnöt ohjataan servlet-moottorille (servlet engine), joka huolehtii ohjelman suorittamisesta. Yleensä yhtä sovellusta varten on olemassa yksi palvelimella ajettava instanssi. Servletit pystyvät kuitenkin tehokkaaseen toimintaan, koska ne suoritetaan säikeistettyinä. Ohjelmistokehityksessä on erittäin tärkeää huomioida säikeiden mukanaan tuomat erityispiirteet esimerkiksi jaetun muistin osalta. On mahdollista määrätä sovellus toimimaan ainoastaan yhdessä säikeessä, mutta se ei ole yleensä oikea ratkaisu ongelmien välttämiseksi, koska silloin kerrallaan voidaan palvella vain yhtä asiakasta. /2, s. 31–32./

Servlet-säiliö huolehtii servlet-sovellusten elinkaaresta. Elinkaaren ensimmäiset vaiheet muodostuvat sovelluksen lataamisesta sovelluspalvelimen muistiin ja servletin alustuksesta. Nämä vaiheet voidaan suorittaa joko palvelimen käynnistymisen yhteydessä tai ensimmäisen asiakkaan kutsuessa sovellusta. Alustuksen jälkeen ohjelma jää varsinaiseen suoritustilaan toteuttamaan sille osoitettuja palvelupyyntöjä. Servletin elinkaareen viimeiset toimenpiteet suoritetaan yleensä, kun sovelluspalvelin sammutetaan. Nämä vaiheet muodostuvat sovelluksen tuhoamisesta ja roskien keruusta. /2, s. 32–33./

4.3 Portlet-sovellukset

Portaali on web-sovellus, joka toimii ylemmän tason käyttöliitttymäkomponenttina portleteille. Portaali tarjoaa tyypillisesti käyttäjäkohtaista muokattavuutta, kertakirjautumisen (single sign on) sekä visuaalisesti yhtenäisen ilmeen. Portlet on puolestaan yksittäinen Java-sovellus, jolla tuotetaan osa portaalin näkymästä. Tyypillisesti tämä mahdollisesti dynaamisesti muodostettu sisältö koostuu esimerkiksi XHTML-kuvauskielellä tuotetusta koodin osasta. Portaali on vastuussa siitä, että erilaisilla portleteilla tuotettu sisältö muodostaa yhtenäisen kokonaisuuden, esimerkiksi täydellisen HTML-sivun. /4, s. 13./

Asiakkaat, esimerkiksi www-selaimet, kommunikoivat portlettien kanssa pyyntö- ja vastauspareilla (request, response), joita portaali ohjastaa. Servlet-sovellusten tapaan myös portlettien ajonaikaisena ympäristönä toimii erillinen säiliö, jota kutsutaan portlet containeriksi. Säiliö huolehtii sisältämiensä sovellusten elinkaaresta, sekä portaalilta tulevista palvelupyynnöistä. /4, s. 13–14./

Servlet- ja portlet-sovelluksilla on selkeä riippuvuussuhde toisistaan, ja yhdessä JSP-sivujen kanssa ne muodostavat Java-kielellä toteutettavan web-sovelluksen rungon. Edellisessä kappaleessa mainittujen yhtäläisyyksien lisäksi portletit eroavat servleteistä tietyiltä osin. Tärkeimpiä eroavaisuuksia ovat esimerkiksi: Portletit luovat vain osan kuvauskielen tulosteesta, Niitä ei ole sidottu tiettyyn URL-osoitteeseen, Asiakasohjelmat kommunikoivat portaalin kautta ja Sama sovellus voi esiintyä useamman kerran samassa portaalissa. Portlet-sovellus voi hyödyntää helposti servlet-ohjelmia, JSP-sivuja ja JSP-tagi-kirjastoja (JSTL). /4, s. 15–17./

Portleteille on määritelty ikkunan kokoon liittyviä tiloja sekä erilaisia toimintamoodeja. Spesifikaatiossa on määritetty kolme tilaa: view, edit ja help. View on normaali toimintatila, jossa ohjelma ottaa vastaan palvelupyyntöjä ja tuottaa vastauksena osan portaalisivun sisällöstä. Sovelluksen on toteutettava vähintään tämä tila. Edit-moodissa käyttäjä voi muokata portlettia, ja help-moodi tarjoaa apua ohjelman käytöstä. Kaksi viimeisenä mainittua tilaa eivät ole pakollisia. Portaalipalvelin voi sisältää myös valmistajakohtaisia tiloja. Ikkunatiloja

on myös kolme ennalta määritettyä sekä mahdolliset portaalikohtaiset tilat. Normal-tilassa portlet-sovellus saattaa jakaa sivun muiden ohjelmien kanssa, joten tulosteen koko voi olla rajoitettu. Maximized-tilassa portlet on mahdollisesti ainoa sovellus kyseisellä sivulla tai sillä voi olla paljon tilaa käytettävissään. Minimized-tilaan saatetun sovelluksen pitäisi tuottaa mahdollisimman vähän, jos ollenkaan, tulostetta. /4, s. 35–40./

4.4 Käyttöliittymän tekniikat

Tässä työssä käyttöliittymän toteutukseen käytetään Javan JSP-sivuja /5/ dynaamisen HTML-kuvauskielen luomiseen ja jMaki-kirjastoa /6/ AJAX-ominaisuuksien toteuttamiseen. AJAX on lyhenne englaninkielisistä sanoista ”Asynchronous JavaScript And XML” /7/ eli asynkroninen JavaScript ja XML. Tällä kombinaatiolla yhdistellään vanhoja tekniikoita uudella tavalla ja saadaan luotua uudenlaista toiminnallisuutta www-sivuille. Tämän opinnäytetyön käyttöliittymässä hyödynnetään ainoastaan asynkronista tiedonsiirtoa, jolla on mahdollista päivittää vain osa www-sivusta, jolloin vältetään koko sivun uudelleenlataaminen. Käyttäjä kokee tällaisen ominaisuuden parempana käytettävyytenä sekä nopeutuneena toimintana. Tietoa siirretään selaimen ja palvelimen välillä javascript-kielen XMLHttpRequest-oliossa. Varsinaisesta tiedon kuljettamisesta huolehtii HTTP-protokolla /7/.

Dynaamisten www-sivujen tuottamiseen käytetty JavaServer Pages (JSP) -teknologia on osa J2EE-kokonaisuutta. JSP-sivuilla voidaan toteuttaa palvelin- ja alustariippumattomia www-sovelluksia, joiden koodi suoritetaan palvelimella. Vaikka JSP-sivulla on mahdollista luoda sovelluksen toimintalogiikkaa, niin yleensä niitä käytetään käyttöliittymän luomiseen. Näihin sivuihin voidaan upottaa suoraan Java-kielistä koodia, jolla tuotetaan esimerkiksi HTML-kuvauskielen elementtejä. Yhtä hyvin HTML-koodin staattiset osat voidaan kirjoittaa suoraan JSP-sivulle ja dynaamiset osat muodostaa Javalla. Ohjelmistologiikan toteuttamista varten on olemassa tag-kirjastoja, joiden XML-kuvauskieltä muistuttavilla elementeillä saadaan helposti aikaiseksi dynaamista toiminnallisuutta. JavaServer

Pages Standard Tag Library (JSTL)-kirjasto sisältää esimerkiksi tageja, joilla voidaan suorittaa ehtoihin perustuvaa suorituksen haarautumista tai iteraatioita. /5./

AJAX-ominaisuuksien toteuttamiseen tässä opinnäytetyössä käytetään asiakas-palvelin ohjelmistokehyksen (framework) tarjoavaa jMaki-kirjastoa. Tämä javascript-kirjasto sitoo yhteen monia muita kirjastoja tarjoten yhdenmukaisen tavan käsitellä esimerkiksi käyttöliittymältä tulevia tapahtumia ja suorittaa asynkronista tiedonsiirtoa selaimen ja palvelimen välillä. Kirjasto tarjoaa myös joukon pieniä sovelluksia, joita kutsutaan widgeteiksi. Näiden sovelmien joukosta löytyy valmiita komponentteja, joita yhdistelemällä voidaan helposti luoda käyttöliittymiä. /6./

5 KEHITYSYMPÄRISTÖ

Tässä kappaleessa esitellään opinnäytetyön sovelluksen kehittämiseen käytetyt työkalut. Samalla tutustutaan testauksessa ja kehittelyssä käytettyihin tietojärjestelmiin.

5.1 Java

Tämän opinnäytetyön Java-alustana toimi Sun Microsystemsin Java EE 5 SDK Update 4. Samassa paketissa seurasi mukana ohjelmiston suorittamiseen vaadittava sovelluspalvelin nimeltään Sun Java System Application Server 9.1/GlassFish v. 3.0. Portlettien suorittamiseksi oli vielä erikseen asennettava Open Portlet Container versio 2.0 beta -portlet-säiliö.

5.2 Sovelluskehitin

Työn ohjelmointiympäristönä toimi avoimen lähdekoodin sovelluskehitin nimeltään NetBeans IDE 6.0 (www.netbeans.org). Kyseinen työkalu on ominaisuuksiltaan samaa tasoa kuin kaupalliset kilpailijansa. Plugin-arkkitehtuurin myötä sovellukseen voidaan lisätä ominaisuuksia erilaisiin tarpeisiin ja mahdollistaa saman työkalun käyttö useissa, jopa erikielisissä,

ohjelmistoprojekteissa. Tilapäistunnusten hallintasovelluksen kehittämiseen käytettävät tekniikat ovat erinomaisesti tuettuna tässä kehitystyökalussa.

Tämän opinnäytetyön sovelluksen testaaminen oli helppoa, koska varsinainen ohjelma voitiin suorittaa suoraan sovelluskehittimestä. Sovellus suoritettiin Sunin Application Serverissä ajettavassa Portlet Containerissa. Helposti asennettavalla sovelluskehittimen lisäkkeellä tunnushallintasovellus saatiin käännettyä ja siirrettyä (deploy) Portlet Containeriin.

5.3 Versionhallinta

Jokaisessa ohjelmistoprojektissa kunnollinen ja helposti käytettävä versionhallintasovellus on ehdoton edellytys. Työssäni tässä tehtävässä oli avoimen lähdekoodin sovellus nimeltään Subversion (www.tigris.org), jonka versionumero oli 1.4.2. Versionhallintasovelluksen tarkoituksena on mahdollistaa ohjelmiston hallittu kehittäminen ja muutosten integrointi olemassa olevaan koodiin. Tällaisella ohjelmistolla voidaan tehdä muutoksia tuotettavaan sovellukseen ja tarvittaessa palauttaa vanhempi tilanne takaisin käyttöön, jos jokin uusi ratkaisu osoittautuu toimimattomaksi.

NetBeans sisältää mahdollisuuden käyttää Subversionia suoraan kehitysympäristöstä, joten versionhallinta oli tässä työssä erittäin helppoa. Tehtyjä muutoksia on myös mahdollista tarkastella erittäin havainnollisesti. Versionhallinnan tietovarasto (repository) sijaitsi Tampereen ammattikorkeakoulun palvelimella (https://svn.tamk.fi), johon oli mahdollista päästä käsiksi myös kotoa.

5.4 Tietokanta

Tunnushallinnan tietokanta sijaitsee avoimen lähdekoodin MySQL-tietokantapalvelimella, jonka versio on 4.0.20.0. Tietokannassa käytetään InnoDB-tyyppisiä tauluja /1, s. 20/, joilla voidaan suorittaa transaktioita sekä huolehtia viite-eheydestä. Tunnushallintasovelluksen testaamista varten oli tarpeen luoda tuotantokannasta erillinen tietokanta. Testikannan taulurakenne on identtinen tuotantoversion kanssa ja siihen tallennettiin tuotantokannan varmuuskopion data.

Testikannan versio MySQL:stä on 5.0.20, joka on merkittävästi uudempi kuin tuotannossa olevan. Lähitulevaisuudessa on kuitenkin tarkoitus siirtää tunnushallinnan tietokanta uudempaan ympäristöön, joten tuoreemman version käyttäminen oli täysin perusteltua.

5.5 Hakemistopalvelimet

Tilapäistunnusten hallintasovellus kommunikoi kahden eri valmistajan hakemistopalvelimen kanssa. Toinen palvelimista on Sunin Directory Server 5.2, joka toimii tietovarastona useille TAMK:n palveluille. Microsoftin Active Directory-palvelin huolehtii Windows-työasemien tiedonsaannista.

6 OHJELMISTON RAKENNE

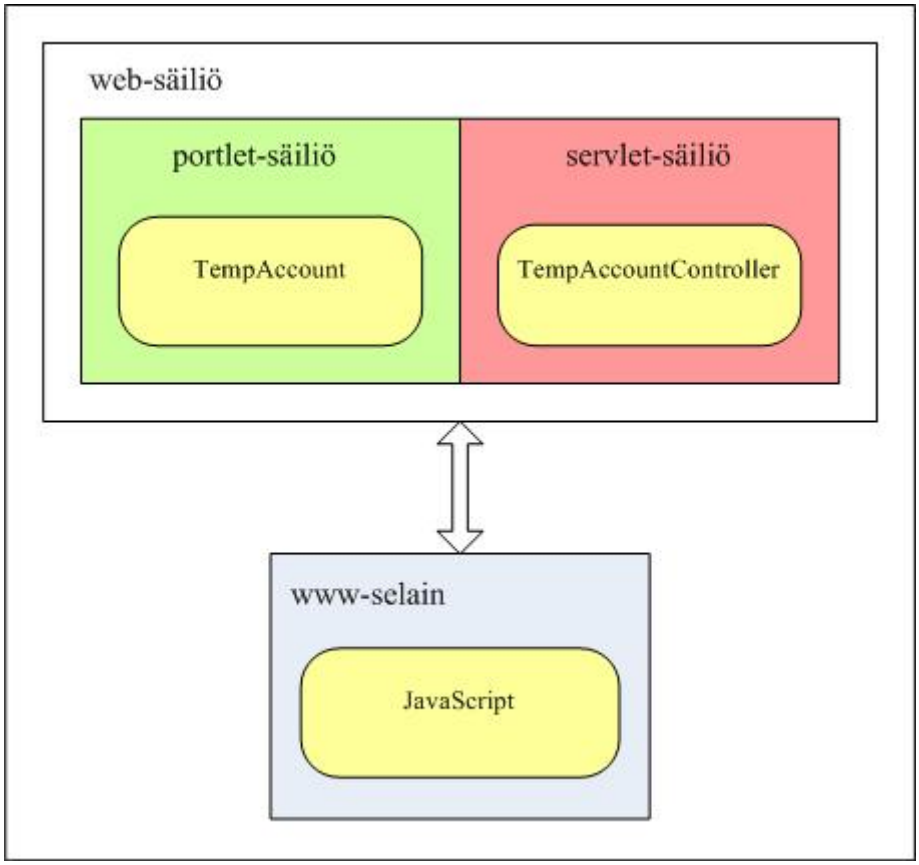
Tässä luvussa on esitetty tämän opinnäytetyön sovelluksen rakenne ja toteutuksen periaatteet. AJAX-tekniikoilla toteutettu käyttöliittymä käsitellään tarkemmin omassa luvussaan, mutta tietyiltä osin tässä luvussa käsitellyssä logiikassa viitataan myös käyttöliittymään.

6.1 Toteutuksen pääperiaatteet

Tilapäistunnusten hallintasovelluksen toteutustekniikkana käytetään Java-ohjelmointikielellä toteutettuja web-sovelluksia. Hallintasovelluksen runko pohjautuu seuraaviin Java-tekniikoihin: portletit, servletit ja JSP-sivut, joita hyödyntämällä pyritään luomaan MVC-mallin mukainen rakenne. Ohjelmiston logiikka, käyttöliittymä ja tietovarastot pyritään siis eriyttämään toisistaan. Kuvassa 3 on esitetty sekä palvelimen että asiakaspään pääkomponentit yleisellä tasolla.

Sovelluksen toimiessa portaalipalvelimella ei ohjelmaan voida suoraan tallentaa tilatietoja. Sama instanssi saattaa palvella yhtä aikaa useita asiakkaita, jolloin tilatieto on tallennettava käyttäjäkohtaisesti. Käytännössä tämä tarkoittaa sitä, että sovelluksen tila on tallennettava käyttäjäkohtaiseen istuntoon (sessio). Istuntotiedon säilytys tapahtuu palvelimella, mutta asiakkaan www-selaimeen

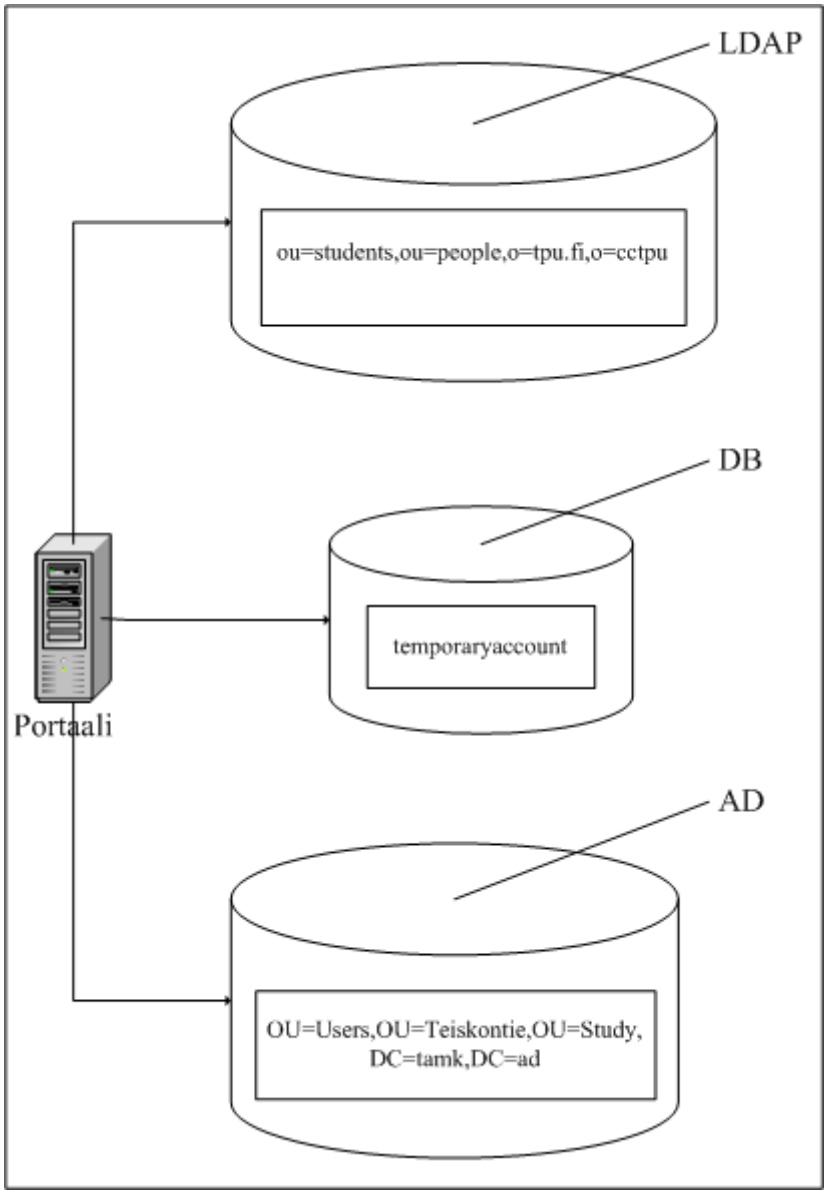
talletetaan eväste (cookie), jolla käyttäjä voidaan tunnistaa. Datan säilöminen istunnossa kuluttaa palvelimen muistia, joten turhan tiedon varastointia on vältettävä resurssien säästämiseksi.



Kuva 3 Sovelluksen pääkomponentit

Usean käyttäjän samanaikainen palvelu sekä palvelimen toiminnan tehostamiseksi suorittama sovelluksen säikeistys, tuovat mukanaan lisää toteutuksessa huomioitavia seikkoja. Pääluokista luodut oliot eivät ole käyttäjäkohtaisia, vaan niiden varaamaa muistia saattaa käyttää samaan aikaan useampi asiakas. Tällöin ei voida olla varmoja siitä, missä järjestyksessä kutakin käyttäjää palvellaan. Tunnushallintasovelluksen toteutuksessa on siis jollain tapaa huolehdittava säieturvallisuudesta. Tässä ohjelmassa ongelma on huomioitu siten, että koko olion näkyvyysalueen muuttujia ei käytetä. Koodaustasolla se tarkoittaa tilapäistiedon tallentamista metodikohtaisiin muuttujiin ja sen jälkeen pysyvästi asiakaskohtaiseen istuntoon. Oliokohtaisia muuttujia olisi mahdollista käyttää siten, että ne lukitaan yhden käyttäjän haltuun kerrallaan Javan synchronized-metodilla. Tällä toteutustavalla sovelluksen käytettävyys häiriintyy, koska muut asiakkaat

joutuvat odottamaan lukon avautumista. Kaikkein huonoin ratkaisu olisi toteuttaa sovellus implementoimalla singleThread-rajapinta, jolloin koko ohjelmaa voisi käyttää ainoastaan yksi asiakas kerrallaan.



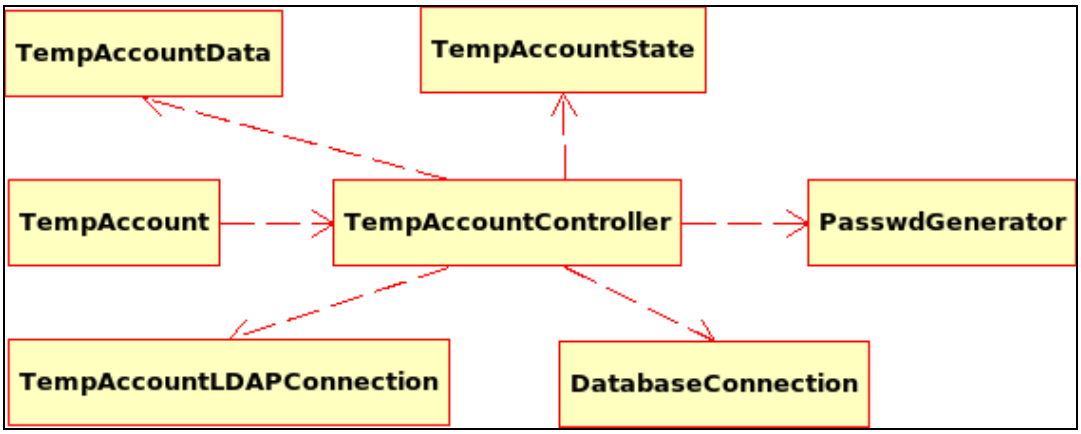
Kuva 4 Sovelluksen käyttämät tietovarastot

Sovelluksen ulkoisina tietovarastoina toimivat MySQL-tietokanta ja LDAP- sekä AD-hakemistopalvelimet. Näiden tietovarastojen liitynnät on esitetty kuvassa 4. Molempiin hakemistoihin päivitetään vain uusi salasana ja tietokantaan enemmän ja tärkeämpää tietoa. Näistä syistä on tärkeintä, että päivitykset tietokantaan onnistuvat tai sitten koko varausprosessi keskeytetään. Tällainen toiminnallisuus on

toteutettu tämän opinnäytetyön sovelluksessa tietokannan tukemilla transaktioilla. LDAP ja AD eivät tue vastaavaa ominaisuutta, mutta jos tietokantaan on tehty onnistunut päivitys, voidaan poikkeustilanteessa salasana vaihtaa myös muilla työkaluilla.

6.2 Pääluokat

Tässä luvussa käsitellään tilapäistunnusten hallintasovelluksen rakennetta yleisesti ja esitellään ohjelman pääluokat. Kuvassa 5 on kuvattu sovelluksen luokkakaavio yleisellä tasolla. Karkeasti jaoteltuna luokat on jaoteltavissa pariin apuluokkaan, ohjauslogiikkaan sekä ulkoisiin tietojärjestelmiin liikennöiviin luokkiin. Kuten kuvasta 5 voi päätellä, luokat vaihtavat tietoa toistensa välillä metodikutsuilla. Web-sovelluksen rakenteesta johtuen yksikään luokka ei sisällä toisen luokan olioita tietojäseninään. Liitteissä 4 ja 5 on esitetty logiikkaluokkien sekä tietovarastoluokkien julkiset rajapinnat.



Kuva 5 Hallintasovelluksen luokkakaavio

TempAccount-luokasta muodostetaan tämän sovelluksen ainoa portlet-instanssi, jolla hoidetaan liityntä portaalipalvelimeen. Tässä luokassa ei ole juurikaan mahdollista toteuttaa logiikkaa käyttöliittymän AJAX-vaatimuksen takia. Käytännössä tämän luokan olio hakee käyttöliittymän päävalikon muodostavan JSP-sivun, johon sitten sovelluksen ohjausluokka päivittää tietoa asynkronisesti.

Tilapäistunnusten hallintasovelluksen ohjauslogiikka hoidetaan TempAccountController-oliossa. Tähän luokkaan liittyy oleellisesti

TempAccountState-luokka, jossa säilytetään istuntoon tallennettavaa käyttäjäkohtaista tilatietoa. Tilaolion metodeilla voidaan asettaa ja lukea ohjelman toimintatilaan liittyvää tietoa. Tilatiedon sekä käyttöliittymältä saadun syötteen perusteella ohjausluokka suorittaa kulloiseenkin tapahtumaan liittyvät toimenpiteet joko suoraan TempAccountController-oliossa tai apuluokkien avulla.

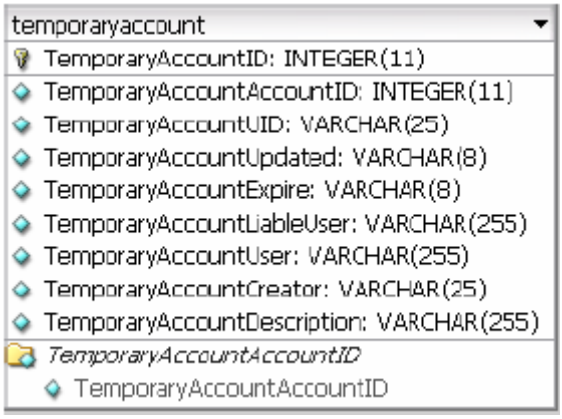
TempAccountData-luokan oliot toimivat tilapäisenä istuntoon tallennettavana tietovarastona, johon sijoitetaan tietokannasta haettu data. Ohjelman suorituksen aikana tähän tilapäiseen tietovarastoon voidaan tehdä tunnuksen varaukseen liittyvät muutokset ennen kuin ne tallennetaan tietokantaan tai hakemistopalvelimille. Kyseisen luokan olio saadaan alustettua tietokantakyselyn tiedoilla käyttäen siihen tarkoitettua metodia. Muilla luokan metodeilla tallennetaan tai haetaan tietoa olion kentistä. Datan validointi suoritetaan sovelluksen muissa luokissa, joten TempAccountData-luokan rakennetta ei tarvitse monimutkaistaa siihen liittyvillä toiminnoilla.

PasswdGenerator-luokka on nimensä mukaisesti suunniteltu salasanojen generointiin. Salasanoissa on pyritty täydellisen satunnaisuuden sijaan siihen, että ne olisivat rakenteeltaan hyviä, mutta silti mahdollisia muistaa. Nämä tavoitteet on saavutettu siten, että muodostettava merkkijono luodaan käyttäen laajoja sanastoja, joista arvotaan suomenkielisiä sanoja. Nämä sanat yhdistetään satunnaisesti arvottuihin yksittäisiin merkkeihin, jolloin kokonainen salasana on muodostettu. Salasanan periaatteellinen rakenne on siis muotoa: <sana1><erikoismerkki1><sana2><erikoismerkki2>. Sanastotiedostoihin päästään käsiksi käyttämällä Javan tiedostoyhteyksiin tarkoitettuja perusluokkia.

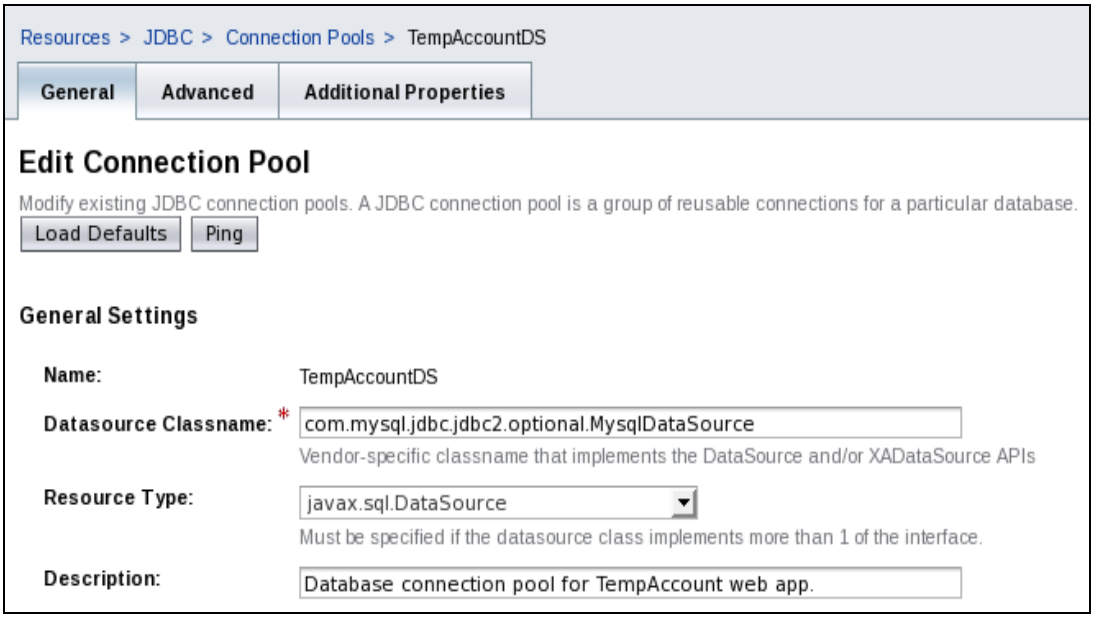
6.3 Tietokantayhteydet

Tämän työn sovelluksen tärkeimpänä tietovarastona toimii tunnushallinnan MySQL-tietokanta. Sen temporaryaccount-taulussa, joka on esitetty kuvassa 6, säilytetään tilapäistunnusten tietoja. Taulun kentät sisältävät seuraavat tiedot: varsinaiseen tunnustauluun viittaavan vierasavaimen, käyttäjätunnuksen, viimeisen muokkausajankohdan, tunnuksen vanhenemispäivän, vastuuhenkilön nimen,

tunnuksen haltijan nimen, tunnuksen luojan nimen ja tunnuksen käyttötarkoituksen kuvauksen. /1, s. 25./ Tämän taulun tietojen perusteella valitaan vapaana olevat tunnukset niiden varaamista varten. Saman taulun tiedoilla valitaan myös varatut tunnukset muokattavaksi.



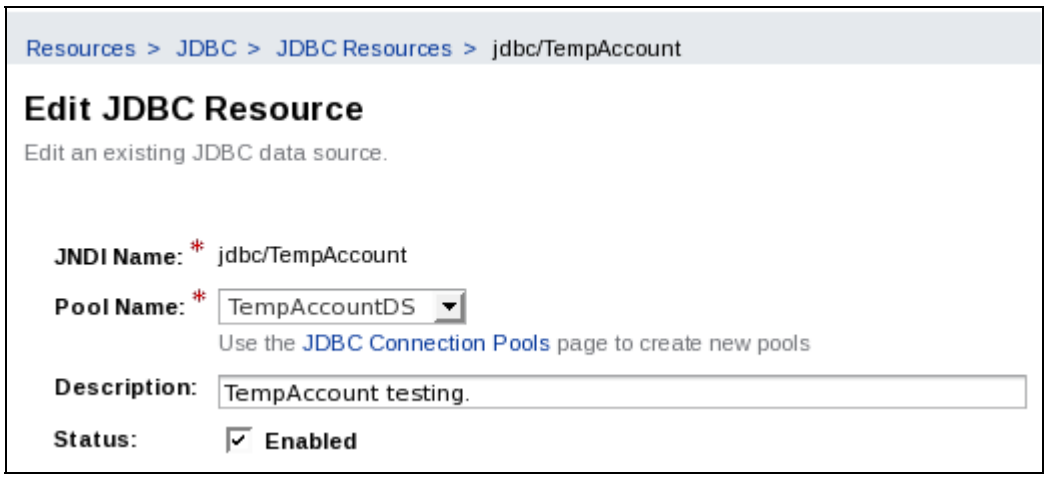
Kuva 6 Tilapäistunnusten tietokantataulun rakenne /1, s. 26./



Kuva 7 Tietokannan yhteysaltaan tärkeimmät määritykset Application Serveriin

Hallintasovelluksen tietokantayhteydet on toteutettu käyttäen sovelluspalvelimen tarjoamaa JDBC-yhteysallasta. Tällaisella ratkaisulla on pyritty siirtämään perustason yhteydenhallinta pois sovelluksesta ja keskittymään enemmän sovelluslogiikkaan. Palvelimeen tulee konfiguroida resurssit tietokantaa varten sekä luoda siihen liittyvä JNDI-nimikonteksti, jolla sovellus käyttää

tietokantayhteyksiä. Tässä työssä käytettävässä sovelluspalvelimessa nämä toimenpiteet suoritetaan graafisen hallintakäyttöliittymän kautta. Kuvassa 7 on ruutukaappaus Application Serverin hallintakonsolista osasta yhteysaltaan määrityksistä ja kuva 8 esittää JNDI-nimen yhdistämistä yhteysaltaaseen.



Kuva 8 Application Serverin JDBC-resurssin määrityksiä

Altaasta saatua yhteyttä käytetään samalla tavalla kuin mitä tahansa Javan tietokantayhteyttä. Tilapäistunnusten varailutahdin ollessa keskimäärin melko vähäinen, ei koko ajan auki olevia yhteyksiä tarvita ja näin säästetään tietokantapalvelimen resursseja. Tietokantayhteys varataan siis vain tarvittaessa, vaikka tämä onkin hieman hitaampi kuin koko ajan auki olevan yhteyden käyttäminen.

Tämän opinnäytetyön sovelluksessa tietokantayhteyksistä huolehtiva luokka on nimeltään DatabaseConnection. Koska varsinainen liityntä tietokantaan hoidetaan yhteysaltaalla, luokka muodostuu staattisista metodeista. Näillä metodeilla haetaan tiedot käytettävissä olevista käyttäjätunnuksista sekä päivitetään varattavan tunnuksen tiedot kantaan. Kirjoitettaessa käytetään transaktioita, jotta voidaan varmistua, siitä että kaikki muutokset saadaan tehtyä. Ellei päivittäminen onnistu palataan rollbackillä alkuperäiseen tilaan ja tulostetaan loppukäyttäjälle virheilmoitus.

6.4 Hakemistopalveluiden liitynnät

Tilapäistunnusten hallintasovellus tarvitsee tietokantayhteyksien lisäksi liitynnän kahteen hakemistopalvelimeen, jotka ovat Sunin LDAP-palvelin sekä Microsoftin Active Directory. Näihin palvelimiin ei päivitetä muuta tietoa kuin varattavan käyttäjätunnuksen uusi salasana. Tämän vuoksi hakemistopalvelimien yhteyksistä huolehtiva TempAccountLDAPConnection-luokka on rakenteeltaan melko yksinkertainen. Siinä luokassa on kaksi staattista metodia, joilla saadaan asetettua salasana sekä LDAP- että AD-palvelimille.

```
// hashtable for settings
Hashtable hash = new Hashtable();

String ldapUrl = sc.getInitParameter( "ldap-url" );

// directory settings
hash.put( Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory" );
hash.put( Context.PROVIDER_URL, ldapUrl );

try
{

    DirContext ctx = new InitialDirContext( hash );

    // only modification is new password
    ModificationItem[] passMod = new ModificationItem[1];

    passMod[0] = new ModificationItem( DirContext.REPLACE_ATTRIBUTE,
            new BasicAttribute( "userpassword", passwd ) );

    // modify passwd
    ctx.modifyAttributes( uid, passMod );

    // close context
    ctx.close();
```

Kuva 9 Osittainen esimerkki JNDI:n käytöstä LDAP-yhteyden muodostamiseen

Salasanan asetusmetodit käyttävät JNDI-rajapintaa, joka on Javan liityntä esimerkiksi hakemistopalveluita varten. Käytännössä JNDIä käytetään siten, että Hashtable-luokan olioon asetetaan hakemistopalvelimeen liittyvät tiedot, esimerkiksi LDAP-url. Sen jälkeen tiedoilla alustetaan DirContext-luokan olio, jonka metodeilla saadaan suoritettua uuden salasanan asettaminen. Kuvassa 9 on esitetty osa TempAccountLDAPConnection-luokan lähdekoodista, jolla havainnollistetaan paremmin JNDIn käyttöä. Molemmat tähän sovellukseen liittyvät palvelimet vaativat luonnollisesti tietyntasoisen käyttäjätunnuksen, jotta

niiden tietosisältöä voidaan muokata. Päivityksiin vaadittu käyttäjätunnus ja salasana asetetaan samalla kun alustetaan sovelluksen muutkin parametrit.

6.5 Sovelluksen toimintalogiikka

Tässä luvussa käydään läpi hallintasovelluksen toimintalogiikka ja toimintojen vaiheet. Ohjelma voidaan jakaa selkeisiin kokonaisuuksiin, jotka toteuttavat loppukäyttäjälle näkyvän toiminnallisuuden. Toiminnot voidaan karkeasti jaotella seuraavasti: päävalikon näyttäminen, tunnuksen varaaminen, varattujen tunnusten tarkastelu sekä niiden muokkaaminen. Seuraavissa luvuissa syvennytään näihin toimintoihin.

6.5.1 Päävalikon vaiheet

Liitteessä 1 on esitetty sekvenssikaaviona sovelluksen päävalikon generoimiseen käytetyt vaiheet. Käyttäjän avatessa selaimeensa sellaisen portaalin osan, johon tämän tutkintotyön sovellus on määritelty, aloitetaan päävalikon muodostaminen. TempAccount-luokan ilmentymän doView-metodissa haetaan koko sovelluksen päänäkymän muodostama JSP-sivu. Tällä sivulla suoritetaan latauksen yhteydessä jMaki-javascript-kirjaston sisältämä funktio, jossa kutsutaan XMLHttpRequest-oliolla TempAccountController-luokan kutsujenkäsittelijää. Ohjausluokan olio luo käyttäjää varten uuden istunnon sekä tilapäistunnuksia varten tyhjän tietovaraston. Näiden toimenpiteiden jälkeen TempAccountController käyttää RequesDispatcher-luokan oliota hakeakseen päävalikon muodostavan JSP-sivun.

6.5.2 Varauslomakkeen vaiheet

Tilapäistunnuksen varaamiseen käytettävän lomakkeen muodostamiseen johtavat vaiheet on kuvattu liitteessä 2. Tämä toiminto aktivoituu käyttäjän valitessa portletin päävalikosta vaihtoehdoksi tunnuksen varaamisen. TempAccountController ottaa käsittelyyn selaimelta lähetetyn pyynnön ja vaihtaa kyseisen asiakkaan istuntoon tilaksi tiedon haun tietokannasta ja kutsuu siihen käytettyä metodiaan. Varsinainen tietokantayhteys ja SQL-kysely toteutetaan DatabaseConnection-luokan getTempTableData-metodilla. Tämä metodi pyytää uuden yhteyden JDBC-yhteysaltaasta ja suorittaa sen saatuaan kyselyn

tunnushallinnan tietokantaan. Onnistuneen kyselyn tulokset palautetaan ResultSet-oliossa TempAccountControllerille ja tallennetaan käyttäjäkohtaiseen istuntoon TempAccountData-luokan olioina. Kontrolleriolio siirtää seuraavaksi suorituksen JSP-sivulle, joka generoi varattavaksi arvotun tunnuksen tietojen muokkaamiseen tarvittavan html-lomakkeen.

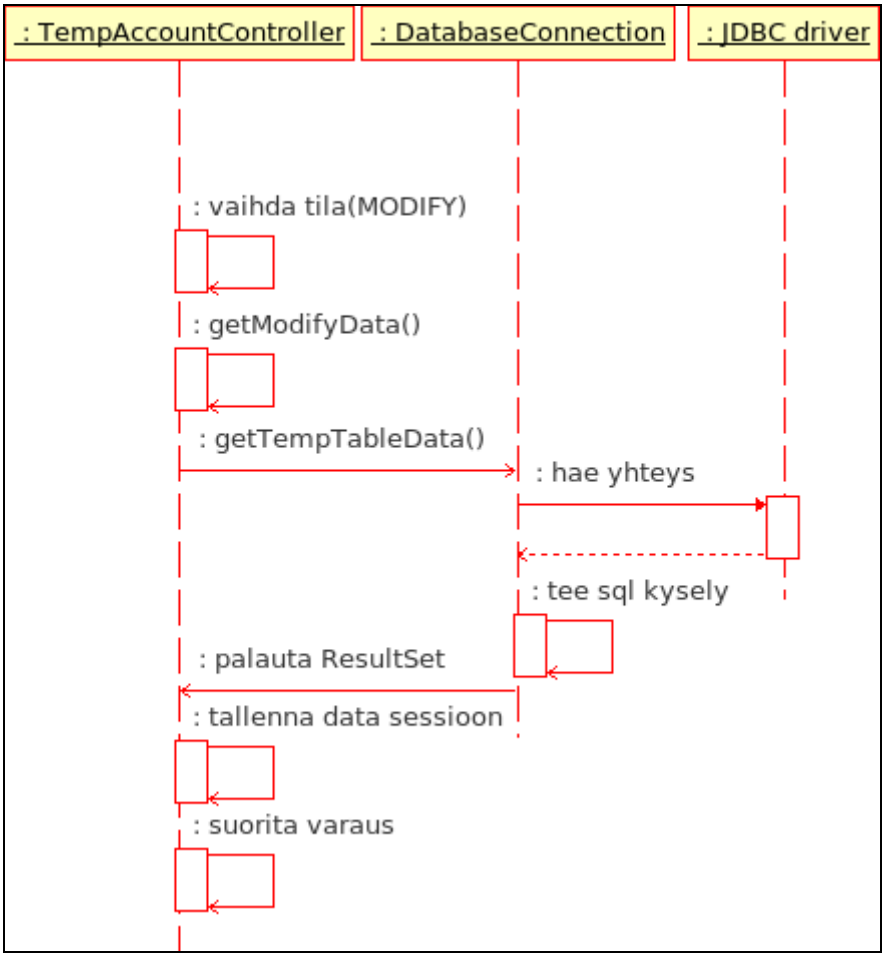
6.5.3 Tunnuksen varaamisen vaiheet

TempAccountControllerin ottaessa vastaan edellisessä vaiheessa muodostetun lomakkeen käsittelykutsun, käynnistyvät liitteessä 3 esitetyt vaiheet tunnuksen varaamiseksi. Ensin käyttäjäkohtaiseen istuntoon vaihdetaan tunnuksen varausta kuvaava tila ja noudetaan lähetetyt tiedot. Seuraavaksi käytetään DatabaseConnection-luokan metodia yhteyden noutamiseen altaasta ja SQL:n update-lauseen suorittamiseen. Onnistuneen tietokannan päivittämisen jälkeen luodaan PasswdGenerator-luokan metodilla uusi salasana, joka asetetaan LDAP- ja AD-hakemistopalvelimelle. TempAccountLDAPConnection-luokka sisältää metodit molempien hakemistopalvelimien salasanan päivittämiseksi. Mikäli nämä operaatiot onnistuvat, generoidaan JSP-sivulla ilmoitus onnistuneesta toimenpiteestä. Varatun käyttäjätunnuksen tiedot ovat saatavilla tulostusta varten saman ilmoitussivun kautta.

6.5.4 Varatun tunnuksen muokkaaminen

Varatun tunnuksen muokkaaminen käynnistyy, kun käyttäjä suorittaa toiminnon valinnan päävalikosta. Muokkausprosessin vaiheet on esitetty kuvan 10 sekvenssikaaviossa, josta on jätetty pois www-selaimen ja portaalin osuudet. Palvelupyynnön tullessa TempAccountControllerille, vaihdetaan käyttäjäkohtaisen istunnon tila muokkausta vastaavaksi. Tämän jälkeen suoritetaan metodikutsu, jossa suoritetaan tiedon haku tietokannasta DatabaseConnection-luokan metodilla. Tietokantayhteys muodostetaan samalla tavalla kuin tunnuksen varaamisen yhteydessä. Ainoana erona on se, että SQL-kyselyllä haetaan tällä hetkellä varattuna olevat tunnukset. Tietokannasta haettu tieto tallennetaan istuntoon ja sovelluksen suorittaminen ohjataan samalle käyttöliittymäkomponentille kuin varauksen yhteydessä. Ainoa ero varaamiseen verrattuna tästä vaiheesta eteenpäin

on se, että voimassaolopäivää ei voi muokata. Tämä seikka huomioidaan sekä käyttöliittymässä että palvelimella suoritettavassa muokatun tiedon validoinnissa.



Kuva 10 Varatun tunnuksen muokkaamisen sekvenssikaavio

6.6 Ohjelman konfigurointi

Sovelluksen asetusten muokkaaminen ilman uudelleenkääntämistä on tärkeä ominaisuus. Koodiin upotetut asetukset ovat huonon ohjelmointitavan mukaisia, joten on oltava jokin tapa muokata ohjelman asetuksia koskematta koodiin. Portlet-sovelluksilla on olemassa kaksi kuvaustiedostoa, joissa määritellään parametreja sovelluspalvelinta varten sekä mahdollisesti sovelluksen asetuksia. Nämä tiedostot ovat nimeltään portlet.xml ja web.xml: ensimmäinen on käytössä ainoastaan portlet-sovelluksissa, jälkimmäistä käytetään myös servlettien yhteydessä.

```
<context-param>
    <description>LDAP url</description>
    <param-name>ldap-url</param-name>
    <param-value>ldap://eta.tpu.fi:389/ou=people,o=tpu.fi,o=cctpu</param-value>
</context-param>
<context-param>
    <description>Password generator dictionary file</description>
    <param-name>passwd-file-3</param-name>
    <param-value>/home/maza/words3</param-value>
</context-param>
```

Kuva 11 Osa web.xml-tiedoston kustomointiparametreista

Tilapäistunnusten hallintasovelluksessa ei ole kovinkaan paljon muuttuvia parametreja, joten ei ole mielekästä käyttää muita kustomointimahdollisuuksia kuin web.xml-tiedostoa. Kuvassa 11 on esitetty osa tästä tiedostosta, jossa määritellään ohjelman asetuksia. Parametrit on määritelty <context-param>-elementissä, joka sisältää parametrin kuvauksen, nimen sekä arvon. Ylemmässä parametrissa asetetaan LDAP-palvelimen URL ja alemmassa salasanageneraattorin käyttämän sanaston polku. Taulukossa 1 on kuvattu hallintasovelluksen käyttämät asetukset selityksineen.

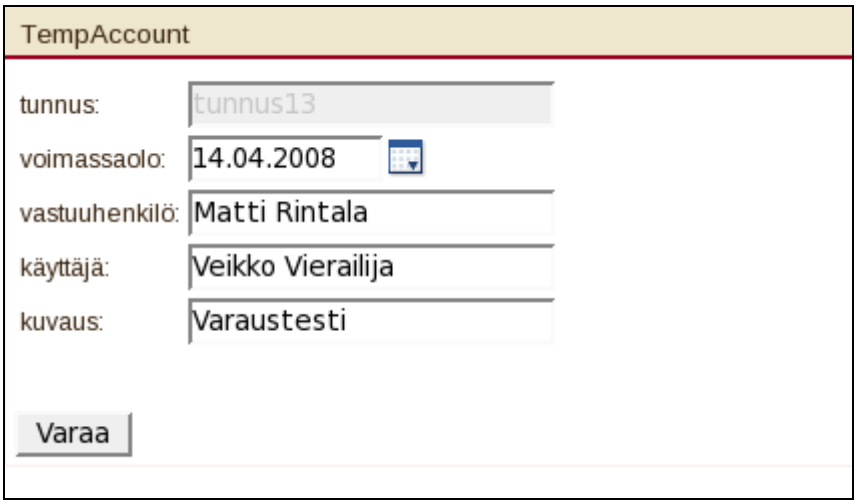
Parametrin nimi	**Parametrin kuvaus**
database-name	JDBC-yhteysallas
ldap-url	LDAP url
ad-url	Active Directory url
passwd-file-3	Polku kolmen merkin pituiseen sanastoon
passwd-file-4	Neljän merkin sanasto
passwd-file-5	Viiden merkin sanasto
ldap-username	LDAPin käyttäjätunnus
ldap-password	LDAPin salasana
ad-username	Active Directoryn käyttäjätunnus
ad-password	Active Directoryn salasana

Taulukko 1 Web.xml:ssä määriteltävät parametrit

7 KÄYTTÖLIITTYMÄ

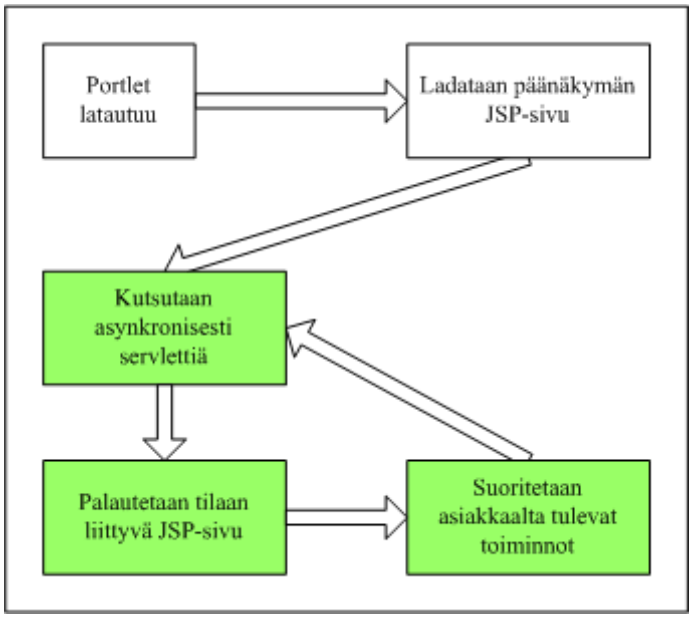
Tilapäistunnusten hallintasovelluksen käyttöliittymä on osa kokonaista portaalisivua. Käyttäjän www-selaimessa käyttöliittymän osat esitetään HTML-kuvauskielen mukaisilla elementeillä. Dynaaminen sisältö generoidaan käyttäen JSP-sivuja ja JavaScriptiä, jotka kommunikoivat taustalla toimivan ohjauslogiikan kanssa. Jotta koko portaalisivua ei ladattaisi uudelleen jokaisen palvelimen ja

selaimen välisen liikennöintitapahtuman yhteydessä, on käyttöliittymässä hyödynnetty AJAXin asynkronista tiedonsiirto-ominaisuutta. Tämän toiminnallisuuden ansiosta sovellus toimii hieman nopeammin ja on miellyttävämpi käyttää. Kuvassa 12 on näytönkaappaus tunnuksen varausnäkymän tietojenmuokkausdialogista.



Kuva 12 Varattavan tunnuksen tietojen muokkaus

Teknisesti käyttöliittymä on toteutettu siten, että sovelluksen ensimmäisen latauksen yhteydessä käyttäjälle esitetään JSP-sivu, jota käytetään koko sovelluksen päänäkymänä. Kyseinen sivu sisältää tavallista JavaScript-koodia sekä jMaki-kirjaston funktioita, HTML-elementtejä ja JSTL-tageja. Kuvassa 13 on esitetty sekä pääsivun latautuminen että sovelluksen toiminnallisuuden muodostava palveluvaihe. Pääsivun yhteen HTML-kielen div-elementtiin sijoitetaan varsinaiset käyttöliittymän näkymät. Sisällön asettaminen tapahtuu käyttäen JavaScript-koodissa DOM-mallin innerHTML-elementtiä. Käyttöliittymä kommunikoi TempAccountController-luokan kanssa käyttäen jMaki-kirjaston funktiota, joka abstraktoi XMLHttpRequest-olion käyttöön liittyvät seikat ohjelmoijalle. HTML-kielen tasolla sovellusta tarkasteltaessa käyttöliittymä muodostuu lomakkeesta, jonka submit-toiminne suorittaa JavaScript-koodin. Koodissa hoidetaan AJAX-liikennöinti sovelluksen palvelimella sijaitsevan logiikan kanssa.

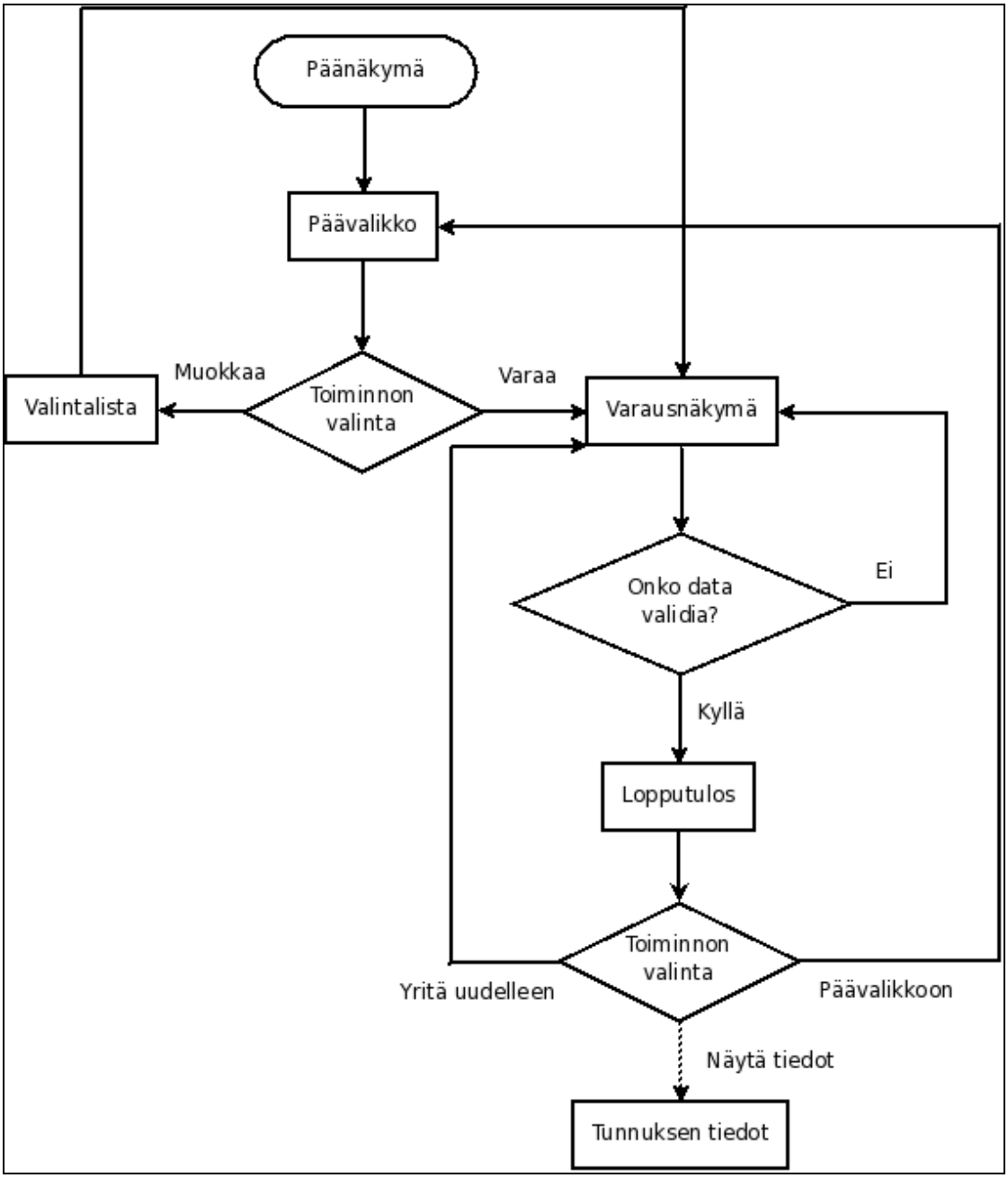


Kuva 13 Sovelluksen käyttöliittymän latautuminen ja palveluvaihe

Tämän opinnäytetyön sovelluksen käyttöliittymän osat on koodattu erillisiksi JSP-sivuiksi. Näin yksittäistä näkymää, esimerkiksi päävalikkoa, voidaan muokata vaikuttamatta muihin sovelluksen osiin. JSP-sivuilla voidaan kätevästi käyttää sovelluslogiikalta tulevaa dataa sivun dynaamiseen luomiseen. Esimerkiksi käyttäjäkohtaisesta istunnosta saadaan suoraan haettua yksittäisen tunnuksen tiedot, jotka voidaan sitten esittää HTML-elementtien avulla muokkaamista varten.

Kuvassa 14 on esitetty käyttöliittymän näkymien kulkukaavio, josta selviää sovelluksen eri vaiheissa esitetyt dialogit. Päävalikossa suoritetaan toiminnon valinta, josta haaraudutaan käyttäjän päätöksen perusteella seuraavaan näkymään. Varausnäkymässä suoritetaan vapaan tunnuksen arvonta ja annetaan mahdollisuus muokata varauksessa vaaditut tiedot. Varausyrityksessä sovelluksen ohjauslogiikalla tarkistetaan tiedon oikeellisuus ja yritetään varata tunnus tai palautetaan tiedot korjattavaksi virheilmoituksen kanssa. Onnistuneen varauksen jälkeen käyttäjälle tulostetaan tieto siitä ja näytetään varattuun tunnukseen liittyvät tiedot. Kulkukaaviossa katkoviivalla osoitetaan erilliseen ikkunaan avattava näkymä, jossa tiedot esitetään tunnuksen haltijalle tulostettavassa muodossa. Ellei varaus onnistu tietojen validoinnin jälkeen, jossakin järjestelmässä on vikatilanne.

Tämän vuoksi loppukäyttäjälle annetaan mahdollisuus yrittää varausprosessia uudelleen, jos esimerkiksi tilapäinen häiriö olisi poistunut.



Kuva 14 Käyttöliittymän kulkukaavio

8 SOVELLUKSEN TIETOTURVA

Käyttäjätunnuksiin liittyvät tiedot ovat luottamuksellisia, joten tietoturvasta huolehtiminen on erittäin tärkeää. Tunnuksiin liittyvää tietoa kuljetetaan verkossa

portaalipalvelimen ja käyttäjän www-selaimen välillä, joten selväkielisen tiedon lähettämistä tulee välttää. Tilapäistunnusten hallintasovelluksen toimiessa osana portaalia ei ole mahdollista määrittää sovelluskohtaista salausta. Tältä osin sovellus on siis portaalipalvelimen käyttämän tai käyttämättömän salauksen varassa.

Sovellustason tietoturva on myös erittäin oleellinen asia, jotta ohjelma toimii luotettavasti ja sen käyttämät tietovarastot pysyvät kunnossa. Hallintasovelluksen suunnittelun lähtökohtana on, että sen käyttämissä tietovarastoissa oleva data on oikeellista. Tämän olettamuksen seurauksena esimerkiksi tietokannasta haettavan tiedon järkevyyttä ei tarkisteta erityisen tarkasti. Koska ohjelman käyttöliittymä sijaitsee käyttäjän www-selaimessa, on erittäin tärkeää huolehtia sieltä tulevan syötteen järkevyydestä. On täysin mahdollista tulkita selaimessa ajettavaa koodia ja muokata se toimimaan haitallisesti. Tästä syystä sovelluksen toteutuksessa suoritetaan vahva tiedon validointi palvelinpäässä. Näin voidaan varmistua siitä, ettei haitallista tietoa tallenneta esimerkiksi tietokantaan.

TAMK:n intranet-portaalissa tiettyjen sovellusten näkyminen käyttäjille voidaan hoitaa roolipohjaisesti: tietyn roolin omaavat käyttäjät saavat näkyviin rooliinsa sidotut kanavat. Tilapäistunnusten hallintasovellus on tarkoitettu rajatulle käyttäjäryhmälle, joten sen käyttö vaatii roolipohjaisia oikeuksia. Sovelluksen portaalissa esiintymisen lisäksi on myös oleellista huolehtia siitä, ettei ohjelman komponentteja pääse suoraan käyttämään muut kuin sallitut käyttäjät. Portaaliin liittyvän kertakirjautumisjärjestelmän avulla sallitut käyttäjät todennetaan jokaisen palvelukutsun yhteydessä.

9 LOPPUPÄÄTELMÄT JA JATKOKEHITYS

Tämän opinnäytetyön aihe osoittautui erittäin monipuoliseksi ja todella haastavaksi. Vaikka Java-ohjelmointi olikin työn tekijälle ennestään tuttua, niin kaikki web-sovelluksissa käytettävät tekniikat olivat uusia asioita. Laajan ohjelmointitekniikoiden kirjon takia oli yhden työn parissa mahdotonta perehtyä kovinkaan syvällisesti jokaiseen aiheeseen. Toisaalta tällainen monipuolinen läpileikkaus eri web-sovellusten tekniikoihin loi hyvän perustan osaamisen

syventämiselle tulevaisuudessa. Useiden tekniikoiden yleinen tuntemus antaa myös hyvän kuvan siitä, mitä milläkin niistä voidaan tehdä.

Osa tässä työssä käytetyistä tekniikoista on ollut pidempään käytössä kuin toiset ja se on havaittavissa ohjelmistorajapintojen kypsyydessä sekä dokumentaatiossa. Vanhempina tekniikoina servleteistä ja JSP:stä on saatavilla paljon lähdemateriaalia ja esimerkkejä. Sitä vastoin portletit, ja varsinkin niihin yhdistetyt AJAX-ominaisuudet, aiheuttivat paljon päänvaivaa. Niihin liittyvää lähdeaineistoa oli myös melko niukasti saatavilla, jolloin valmiin tiedon vähyydestä aiheutui haasteita. Samalla kävi selväksi, että Javalla sekä muilla tekniikoilla toteutettavat www-sovellukset elävät jatkuvassa muutoksessa ja nykyiset tekniikat vanhentuvat nopeasti.

Tilapäistunnusten hallintasovellus vaatii vielä hieman jatkokehitystä ja tiukan testauksen läpikäymisen. Näiden teknisten vaiheiden lisäksi uuden sovelluksen jalkauttaminen loppukäyttäjille on yksi haastavista tehtävistä. Tekniseen puoleen liittyen käyttöliittymän hiominen ja mahdolliset muut pienet parannukset ovat melko yksinkertaisia toimenpiteitä. Sitä vastoin merkittävän haasteen muodostaa käyttäjätunnusten kotihakemistojen sisällön siivoaminen siten, että kaikkien tietojärjestelmien tietoturva säilyy hyvällä tasolla.

LÄHTEET

1 Moisio, Teemu, Tampereen ammattikorkeakoulun tunnushallinnan tietokanta. Insinöörityö. Tampereen ammattikorkeakoulu. Tietotekniikka. Tampere 2005. 34 s. + 2 liites.

2 Ahonen, Tero; Hämeen-Anttila, Tapio; Åstrand, Kim, JavaServlets, 1, painos. Docendo Finland Oy, Jyväskylä 2003. 382 s.

3 Lee, Rosanna; The JNDI Tutorial; [www-sivu]; [viitattu 26.2.2008]; Saatavissa: http://java.sun.com/products/jndi/tutorial/index.html

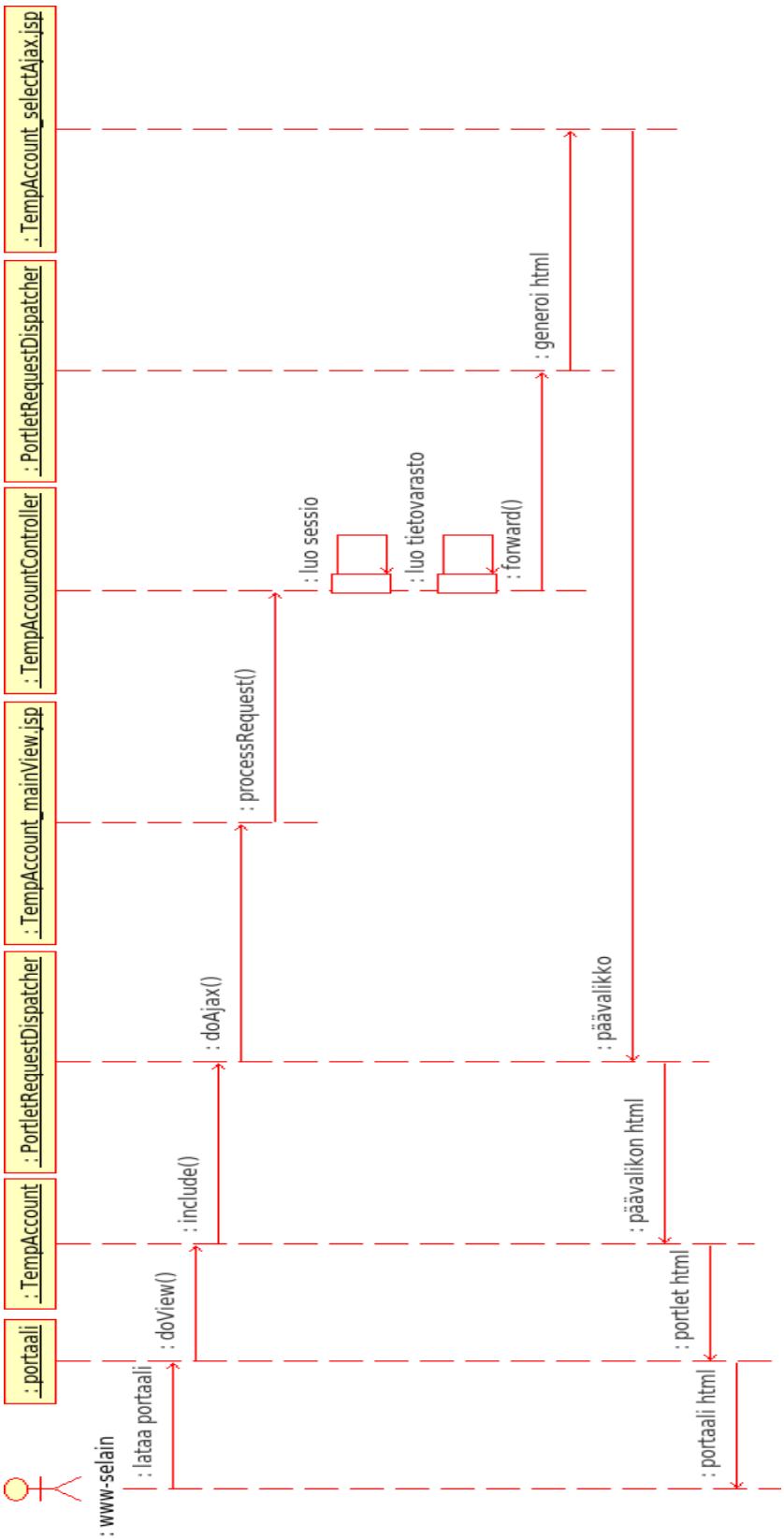
4 Alejandro, Abdelnur; Stefan Hepper, Java Portlet Specification version 1.0, October 7 2003; Saatavissa: http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html

5 JavaServer Pages Technology; [www-sivu]; [viitattu 5.3.2008]; Saatavissa: http://java.sun.com/products/jsp/

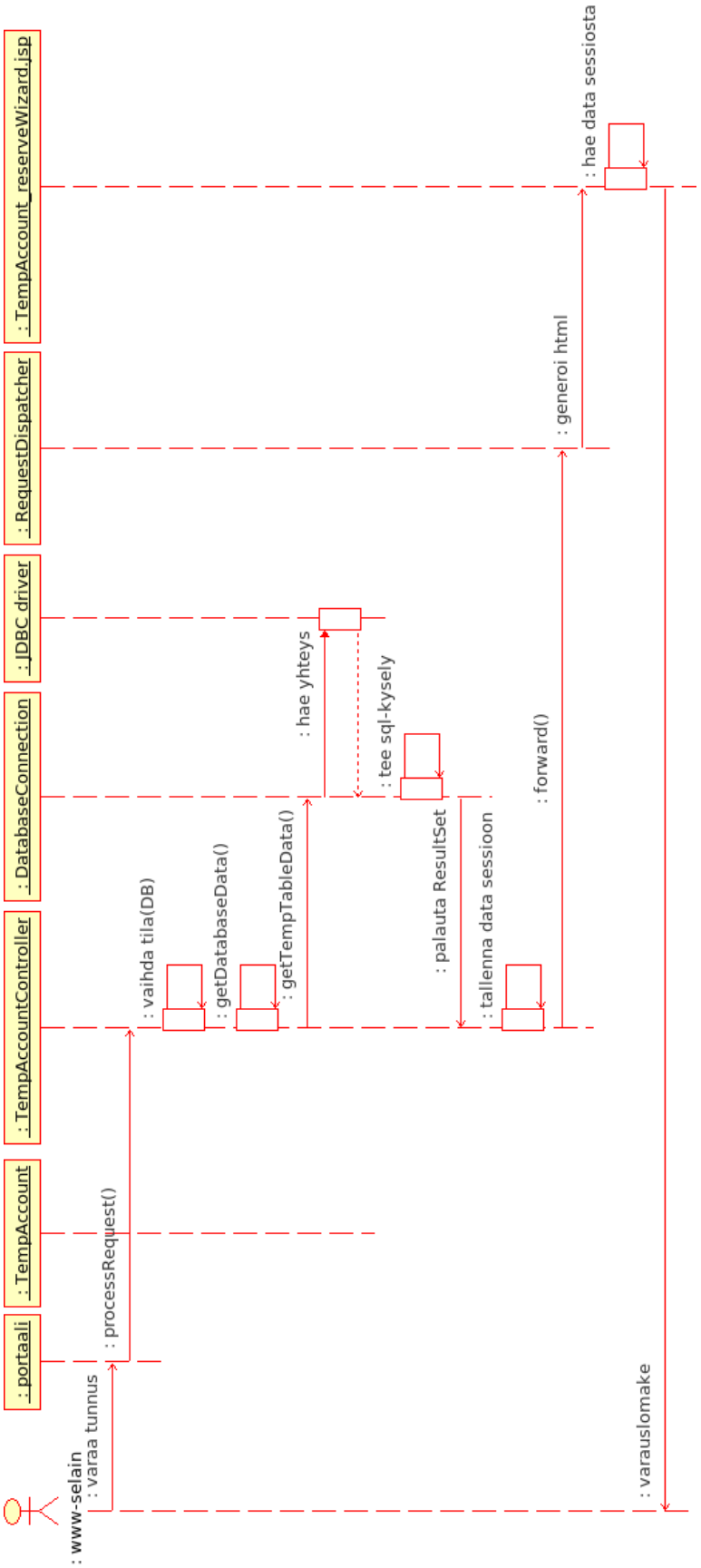
6 jMaki Home; [www-sivu]; [viitattu 8.3.2008]; Saatavissa: http://jmaki.com/

7 AJAX Tutorial; [www-sivu]; [viitattu 11.3.2008]; Saatavissa: http://www.w3schools.com/ajax/default.asp

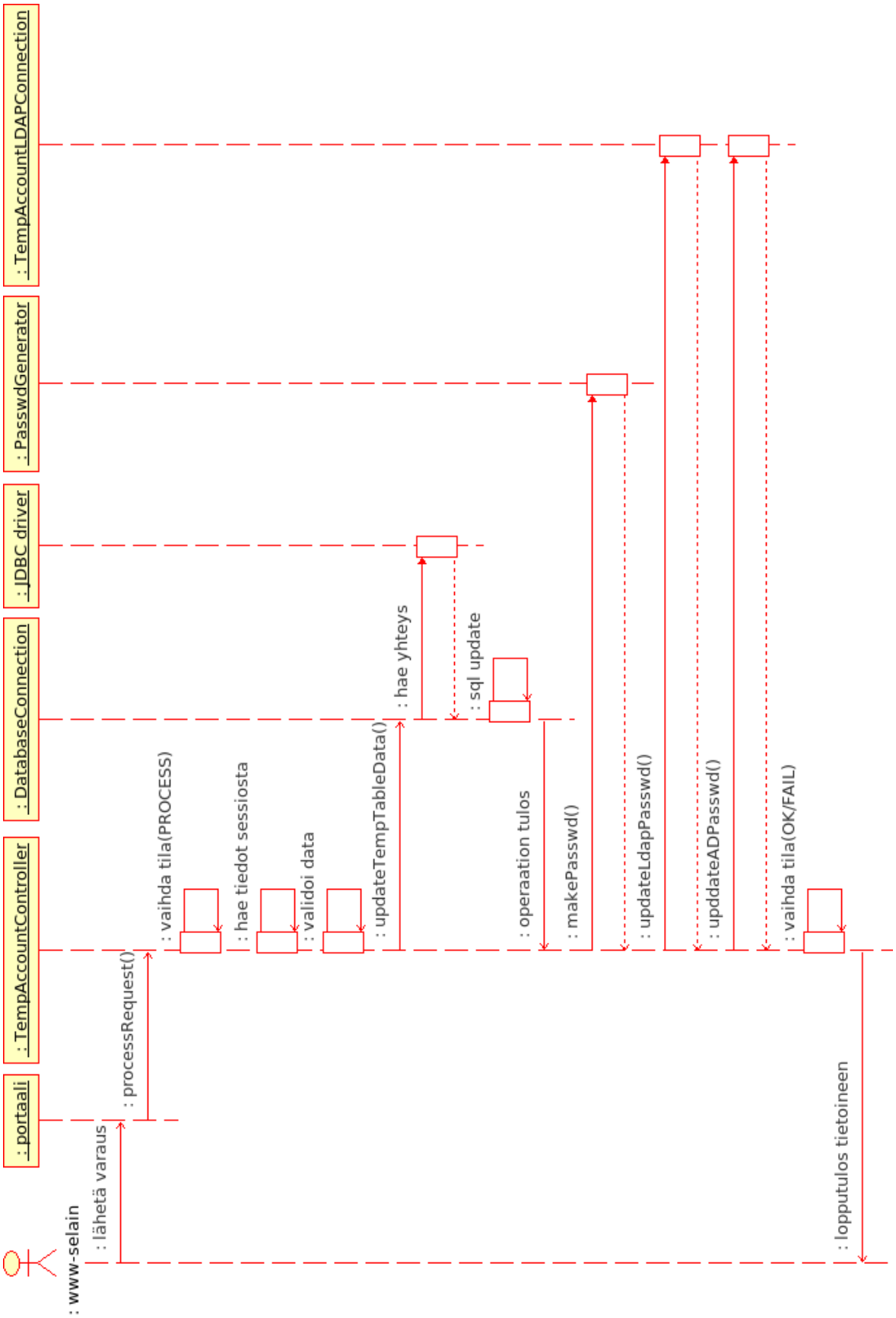
LIITE 1 PÄÄVALIKON SEKVENSSIKAAVIO



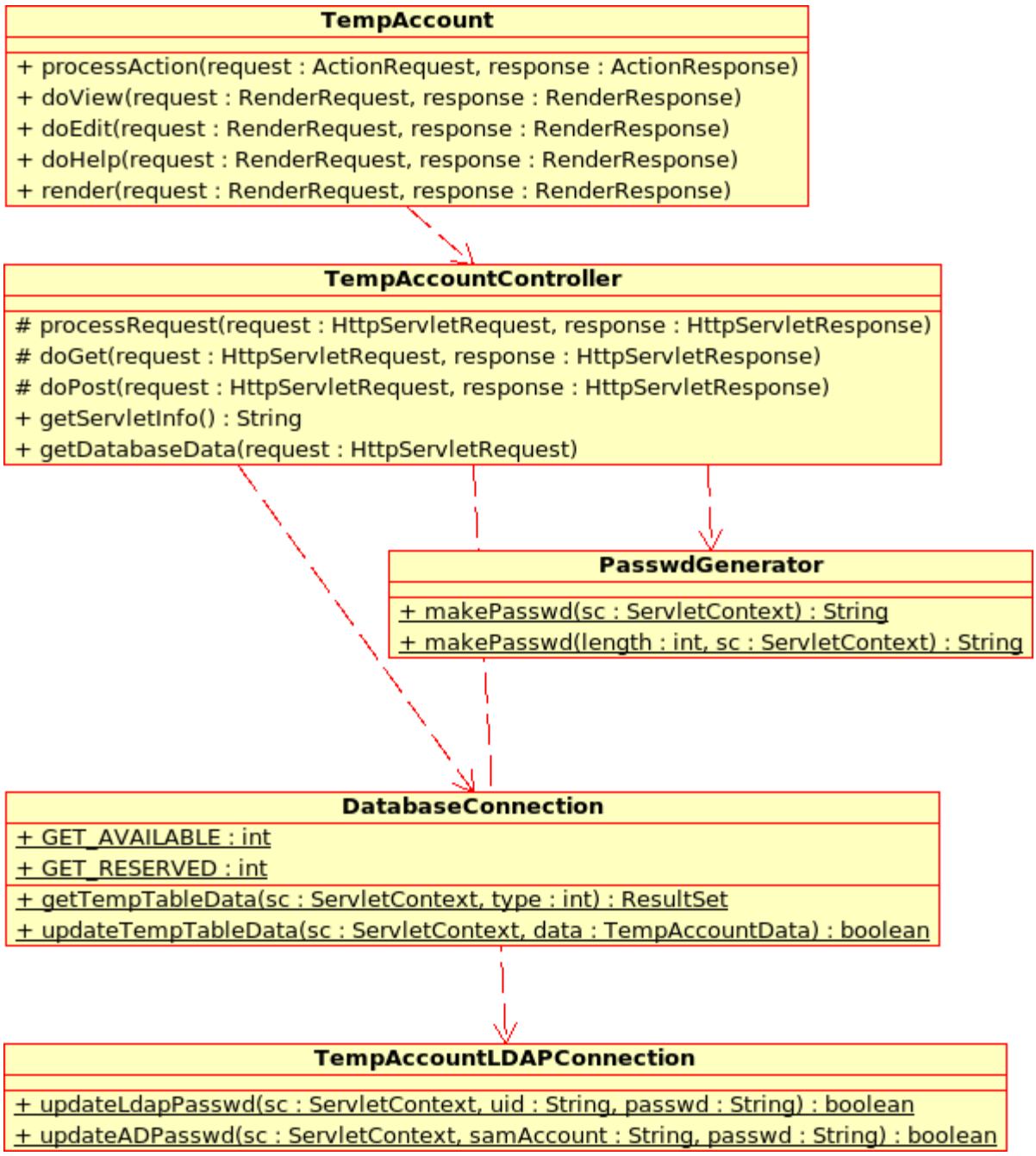
LIITE 2 VARAUSLOMAKKEEN SEKVENSSIKAAVIO



LIITE 3 TUNNUKSEN VARAAMISEN SEKVENSSIKAAVIO



LIITE 4 SOVELLUSLOGIIKAN LUOKKAKAAVIO



LIITE 5 SOVELLUKSEN TIETOVARASTOLUOKAT