Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Ohjelmistotekniikka

2011

Antti Kallio

OHJELMOINTIRAJAPINTA-PALVELUN TOTEUTTAMINEN TIETOVERKOSSA

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

Turun ammattikorkeakoulu

Tietotekniikan koulutusohjelma | Ohjelmistotekniikka

Kesäkuu 2011 | 20

Ohjaaja: TkL Jari-Pekka Paalassalo

Antti Kallio

OHJELMOINTIRAJAPINTAPALVELUN TOTEUTTAMINEN TIETOVERKOSSA

Tässä opinnäytetyössä suunniteltiin ja toteutettiin ohjelmointirajapinta joka mahdollistaa mySync-palvelun hallinnoinnin tietoverkon kautta. Rajapintaa käyttäen palvelun käyttäjät voivat rakentaa omia käyttöliittymiä tai muita työkaluja palvelun hallinnointiin.

Ohjelmointirajapinta toteutettiin REST-arkkitehtuurityylin mukaisesti. Rajapintapalvelu toimii muiden mySync-palveluiden tavoin Java Servlet -palvelimella. Palvelun toteutuksessa käytettiin Restlet-sovelluskehystä, joka hoitaa HTTP-kommunikoinnin perustasolla. Sovelluskehyksen ja arkkitehtuurityylin mukaisesti toteutettiin resurssiluokkia, jotka suorittavat rajapinnan käyttäjän pyytämät toiminnot.

Palvelu käyttää kommunikoinnissa XML-tietorakenteita. Tietorakenteet luodaan pääosin automatisoidusti käyttäen XStream-kirjastoa, joka muuntaa Java-objekteja XML-tietorakenteiksi ja päin vastoin.

Ohjelmointirajapintapalveluun saatiin toteutettua kaikki halutut ominaisuudet, ja se on aktiivisessa tuotantokäytössä.

ASIASANAT:

HTTP, REST, Restlet, Servlet, XStream

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme in Information Technology | Software Engineering

June 2011 | 20

Instructor: Jari-Pekka Paalassalo, Lic. Tech., Principal Lecturer

Antti Kallio

IMPLEMENTATION OF AN APPLICATION PROGRAMMING INTERFACE IN A DATA NETWORK

The subject of this thesis was to desing and implement an application programming interface that enables managing of the mySync-service via data networks. By using this interface, the users of the service can create their own user interfaces or other tools to manage the service.

The application programming interface was designed according to the RESTful architectural style. The interface service operates on a Java Servlet server among other mySync service parts. The service was implemented using the Restlet software framework, which manages basic HTTP communication. Resource classes were implemented according with the framework and the RESTful style to perform the tasks requested by users.

The service uses XML structures to communicate with client applications. The structures are created mainly using automated tools, specifically the XStream library, which transforms Java objects to XML structures and back.

The application programming interface service was implemented with all requirements met and it is in active usage.

KEYWORDS:

HTTP, REST, Restlet, Servlet, XStream

SISÄLTÖ

KUVAT

TAULUKOT

KOODIESIMERKIT

SYMBOLI- JA LYHENNELUETTELO

HTML	kieli jolla www-sivut on koostettu. (Hypertext Markup Language)
JSP	teknologia jolla voidaan tarjota dynaamisesti luotuja HTML-, XML- tai muita dokumentteja. (JavaServer Pages)
RPC	tekniikka joka mahdollistaa ohjelmametodien käytön verkkoyhteyden välityksellä. (Remove Procedure Call)
SOAP	vaihtoehto REST-tyylille verkkopalvelun protokollavalinnaksi. (Simple Object Access Protocol)
URI	ilmaisee resurssin sijainnin verkkopalvelimella. (Uniform Resource Identifier)
W3C	kansainvälinen organisaatio joka kehittää standardeja World Wide Webin käyttöön. (World Wide Web Consortium)
WAR	tiedostopaketti, joka voi sisältää JSP-sivuja, servlet-ohjelmia ja muita tiedostoja. Käytetään palvelimien ohjelmien kokoamiseen. (Web application Archive)
XML	joukko sääntöjä, joilla voidaan luoda määrityksiä dokumenttien rakenteille. (Extensible Markup Language)
XSD	XML-skeemakieli jolla voidaan määrittää XML-dokumenttien muotosäännöt.. (XML Schema Document)

1 JOHDANTO

Sync Oy:n kehittämä mySync-järjestelmä mahdollistaa matkapuhelimien keskitetyn etähallinnan. Järjestelmän ominaisuuksia ovat esimerkiksi puhelimen asetusten määritys, laitteen lukitus, avaus ja tyhjennys komennettaessa sekä kontaktitietojen sekä muiden tietojen hallinta ja synkronointi useiden laitteiden välillä.

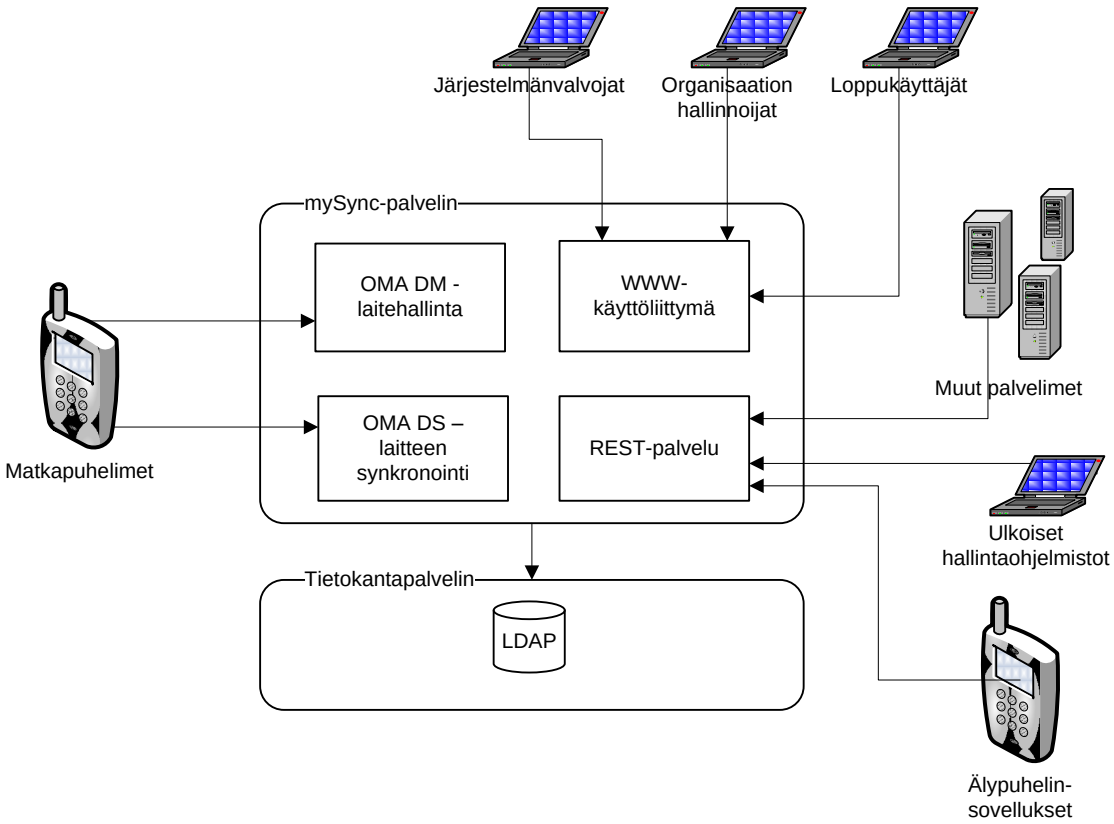
Tässä opinnäytetyössä järjestelmään lisättiin mahdollisuus hallita näitä ominaisuuksia Internetin kautta avoimen ohjelmointirajapinnan avulla. Rajapinta antaa järjestelmää käyttäville tahoille mahdollisuuden toteuttaa uusia hallinta-käyttöliittymiä tai automatisoida toimintaa haluamillaan tavoilla ilman, että itse järjestelmään tarvitsee tehdä muutoksia.

Rajapinnan käyttö on mahdollista järjestelmän olemassa olevilla käyttäjä-tunnuksilla. Opinnäytetyön ohessa luotiin myös dokumentaatio rajapinnan käyttöä varten.

2 ARKKITEHTUURI

2.1 Yleisarkkitehtuuri

Olemassa olevan mySync-järjestelmän arkkitehtuuri perustuu Java Servlet -palvelinohjelmistolla suoritettaviin palveluihin. Erillisiin palveluihin kuuluvat graafinen www-käyttöliittymä, OMA DM -laitehallintapalvelin ja OMA DS -synkronointipalvelin. Järjestelmän ytimenä toimii LDAP-tietokanta. Kaikki palvelut voidaan asentaa uudelle palvelinkoneelle yhdestä asennuspaketista. Kuva 1 havainnollistaa järjestelmän eri palveluiden käyttäjiä. Toteutettava REST-palvelu mahdollistaa uusien hallintamenetelmien toteuttamisen olemassa oleville palvelimille ja aivan uusien käyttötapojen hyödyntämisen.



Kuva 1 mySync-järjestelmän palvelut

Eräs mySyncin sisäisistä osista on ns. core-projekti, joka mahdollistaa tietokannan yksinkertaisen käsittelyn Java-pohjaisista ohjelmista. Siksi toteutettava palvelu on järkevintä toteuttaa Java Servlet -palveluna.

2.2 REST-arkkitehtuurityyli

REST (Representational State Transfer) on termi, jonka Roy Fielding määritti väitöskirjassaan vuonna 2000 [1]. Se ei ole varsinaisesti ohjelmistoarkkitehtuuri, vaan joukko määrityksiä ja rajoitteita, joista muodostuu arkkitehtuurityyli. Rajoitteet, joita noudattamalla luodaan REST-tyylinen sovellus, ovat [2] seuraavat:

- Järjestelmän on oltava asiakas-palvelin-kokonaisuus.
- Sen on oltava tilaton, eli palvelin ei pidä kirjaa asiakkaan tilasta ja jokainen kutsu on itsenäinen.
- Järjestelmän on tuettava välimuisteja sen eri tasoilla.
- Jokaisella resurssilla on oltava yksilöllinen sijainti.
- Sen on skaalauduttava.

Staattiset www-sivustot toteuttavat nämä rajoitteet. Toisaalta suurin osa dynaamisista www-sivustoista ei noudata kaikkia rajoitteita. Yleisin rikkomus on asiakkaan tilan tallennus käyttämällä istuntotunnisteita palvelimella tai evästeissä.

2.3 Vaihtoehtoja REST-tyylille

Monet suosituimmista verkkopalveluista on toteutettu käyttäen REST-tyyliä. Esimerkkejä ovat Twitter, Yahoo ja Flickr. Googlen verkkopalveluista osa on toteutettu SOAP-protokollalla, osa RPC-toiminnoilla [3].

Resursseja käytetään SOAP ja RPC-arkkitehtuuriin perustuvissa palveluissa ohjelmointikielistä tutuilla funktionimillä. Tällaisten palveluiden ohjelmointirajapinta usein voidaankin luoda automatisoidusti olemassa olevista muista rajapinnoista [4]. Haittapuolena verkkoliikenteen määrä voi kasvaa johtuen huonosti optimoiduista komentoviesteistä ja rajapinnan käyttö vaikeutuu johtuen komentoviestien abstraktiotasojen lisääntymisestä.

3 KÄYTETYT TEKNIIKAT

3.1 HTTP-protokolla

HTTP (Hypertext Transfer Protocol) on synkroninen tiedonsiirtoprotokolla [8], jota tavallisimmin käytetään WWW-sivujen ja niillä tarjottavien tiedostojen siirtoon. Protokolla soveltuu kuitenkin myös muihinkin tarkoituksiin. Koska protokolla on yleisessä käytössä Internet-rajapinnoissa, asiakasohjelmien tekijöillä on todennäköisesti kokemusta sen käytöstä. Vaihtoehtoisesti voitaisiin luoda kokonaan uusi protokolla, joka saattaisi käyttää tehokkaammin tiedonsiirtokapasiteettia, mutta olisi epäyhteensopiva monien olemassa olevien työkalujen kanssa.

HTTP:n käyttö on suhteellisen yksinkertaista. Asiakasohjelma lähettää palvelimelle pyynnön, jossa määritetään HTTP-komennon tyyppi, halutun resurssin sijainti, joukko otsaketietoja ja mahdollisesti viestirunko, jonka sisältö voi olla mitä tahansa, esimerkiksi tekstiä, HTML-sivu, XML-dataa tai jopa binaaridataa. Palvelin vastaa pyyntöön samankaltaisella paketilla, jossa on vastauskoodi, koodin tarkoitusta selittävä teksti, otsaketietoja ja mahdollisesti vapaamuotoinen viestirunko [5]. Vastauskoodi voi kertoa esimerkiksi sen, hyväksyttiinkö pyynnössä ollut käsky vai tapahtuiko jokin virhe. Asiakasohjelma voi sitten jatkaa toimintaansa oman logiikkansa mukaisesti.

Tavallisimmat HTTP-komennot on listattu taulukossa 1. Komennoilla on yleisesti ymmärretyt tarkoitukset, mutta niiden todelliset tarkoitukset ja seuraukset ovat palvelimen toteutuksen varassa.

Taulukko 1 Tavallisimmat HTTP-komennot [5]

HTTP-komento	Selitys
GET	GET on komento jolla haetaan palvelusta tietoa. GET-komento ei saa aiheuttaa palvelussa muutoksia, joten peräkkäiset kutsut antavat aina identtisen vastauksen, ellei jokin muu ole aiheuttanut muutoksia. GET-komento on siis turvallinen suorittaa.

PUT	PUT-komentoa käytetään yleensä viestirungossa olevan tiedon tallentamiseen tai päivittämiseen palveluun.
DELETE	DELETE-komennolla poistetaan resursseja.
POST	POST-komennolla voidaan suorittaa komentoja tai syöttää tietoja palveluun. POST-komennon suoritus aiheuttaa yleensä uusia, erilaisia seurauksia toisin kuin edelliset komennot.

Otsaketiedot sisältävät metatietoa siirrettävästä datasta. Niiden avulla HTTP-komennon ohessa voidaan sisällyttää mm. käyttäjätunnus, viestin sisällön muodon tyyppi ja kieli sekä tietoa viestin sisällön vanhenemisesta.

3.2 Java Servlet

Servletit ovat Java-ohjelmointikielen luokkia joilla laajennetaan palvelin-asiakas-arkkitehtuurin palvelimen toimintoja. Luokka on servlet jos se toteuttaa Java Servet API -rajapinnan. Silloin se pystyy vastaanottamaan pyyntöjä ja reagoimaan niihin. Servlettejä käytetään yleensä tuottamaan HTTP-palveluja, esimerkiksi HTML-sivustoja tai XML-tietopalveluja.

3.3 Restlet-sovelluskehys

Palvelinohjelman rakentaminen käyttäen suoraan Java-ympäristön tarjoamaa HTTP-rajapintaa on toki mahdollista. Toteutettavia osioita olisivat mm. käyttäjän todennusmenetelmät, resurssin sijainnin tunnistaminen ja virheviestien käsittely. Edellä mainitut ovat jokaisen HTTP-perustaisen palvelimen perusosia. Näiden osien toteuttaminen itse viivyttäisi projektia ja mahdollisesti korottaisi ohjelmiston monimutkaisuutta tarpeettomasti. Siksi on yleensä järkevää käyttää valmiita sovelluskirjastoja ja sovelluskehyksiä joita on jo kehitetty pitkään. Sovelluskehyksen käytön etuna ovat virheiden pieni todennäköisyys, nopea käyttöönotto ja valmis kehityskehys johon voidaan liittää oman palvelimen toimintalogiikka.

Restlet [6] oli yksi ensimmäisiä Java-sovelluskehyksiä joka noudatti REST-tyyliä. Se saavutti suosiota nopeasti helppokäyttöisyydellään. Sitä kehittää ranskalainen Noelios Technologies, ja sitä levitetään nykyisin avoimen lähdekoodin lisenssillä [2].

Restlet-kehys mahdollistaa sekä palvelin- että asiakasohjelmien luomisen käyttäen samaa kirjastoa ja samoja resurssiluokkia. Tämä helpottaa silloin myös asiakasohjelmien luomista, kun palvelimella käytetyt resurssiluokat on annettu asiakasohjelman käyttöön. Koska Restlet-kehyksestä on versio Android-matkapuhelimiin, on näin helppo tehdä Android-laitteissa toimiva yhteensopiva asiakasohjelma. Tällainen ohjelma onkin jo toteutettu rajapinnan avulla.

Restlet-kehys hallitsee mm. resurssien sijaintien määrittelyn ja resursseihin liittyviin komentoihin reagoinnin. Ohjelmoijan ei tarvitse huolehtia HTTP-viestien välityksestä. Resurssit määritellään omina luokkinaan. Resurssiluokissa ilmoitetaan annotaatioilla, mitkä metodit vastaavat mitäkin HTTP-komentoa ja mitkä tietomuodot ovat mahdollisia viestirungoissa. Metodit saavat parametrina asiakkaalta vastaanotetun viestirungon ja metodit voivat tarvittaessa lopettaessaan palauttaa viestirungon ja HTTP-koodin, jotka välitetään asiakkaalle.

Koodiesimerkissä 1 on esitelty metodi, joka poistaa yksittäisen laitteen. Metodi on annotaatiolla merkitty vastaamaan HTTP-komentoa DELETE. Metodi ei ota parametrina viestirunkoa eikä palauta viestirunkoa. Se käyttää vain resurssin sijainnissa määritettyjä organisaation ja laitteen tunnuksia. Jos poiston aikana tapahtuu virheitä, metodi lisää vastaukseen virhetietoja.

Koodiesimerkki 1 Laitteen poistava metodi

```
@Delete
public void deleteDevice() {
  String orgCode = getUriSegment("orgCode");
  String deviceCn = getUriSegment("deviceCn");

  OrganisationStore orgStore =
      new LdapOrganisationStore(getSession().getAccountEnvironment());
  try {
    Organisation org = new Organisation(orgCode);
    orgStore.read(org);
    DeviceStore deviceStore = orgStore.getDeviceStore(org);
    Device device = new Device(deviceCn);
    deviceStore.read(device);
    StoreAction.removeDevice(device, deviceStore, orgStore.getGroupStore(org),
        orgStore.getEventLogItemStore(org), getSession().getAccessAccount().getGlobalKey());
    return;
  } catch (DataStoreException e) {
    e.printStackTrace();
    getResponse().setStatus(Status.SERVER_ERROR_INTERNAL, e);
    return;
  } catch (NotFoundException e) {
    getResponse().setStatus(Status.CLIENT_ERROR_NOT_FOUND, e,
        "Organisation or account not found or access denied");
    return;
  }
}
```

Restlet-sovelluskehyksellä toteutettu palvelu voidaan suorittaa sellaisenaan Java Servlet -palvelinohjelmistossa. Toteutetussa rajapinnassa onkin toimittu juuri näin.

3.4 XStream-sovelluskirjasto

XML (Extensible Markup Language) on joukko sääntöjä joilla voidaan luoda dokumentteja, joita pystyvät lukemaan ja kirjoittamaan sekä tietokone-sovellukset että ihmiset. Sitä käytetään laajasti verkkopalveluissa kaikenlaisten tietojen välittämisessä. Toteutetussa rajapinnassa kaikki komennot ja tiedot välitetään XML-viesteinä.

mySync-järjestelmässä tietotyyppejä on useita kymmeniä. XML-tiedostomuotojen määritys käsin jokaiselle on aikaa vievää, ja niiden ylläpito muutosten tapahtuessa vaivalloista. Siksi on helpointa käyttää jotain järjestelmää, joka luo itsenäisesti Java-objekteista XML-dokumentteja, ja vastaavasta XML-dokumenteista Java-objekteja.

XStream [7] on helppokäyttöinen sovelluskirjasto, joka suorittaa muunnoksia Java-objektien ja XML-dokumenttien välillä. Normaalisti luokkien ja muuttujien

nimiä käytetään suoraan XML-dokumenteissa elementtien niminä. XStream mahdollistaa omien nimien määrittelyn jotta elementtien nimet voivat olla siistimpiä. Lisäksi XStream mahdollistaa omien muuntajien määrittelyn, jotta monimutkaisemmista luokista saadaan siistimpiä ja yksinkertaisempia XML-dokumentteja. Taulukossa 2 on esimerkki käyttäjätiliobjektin elementtien nimien siistimisestä.

Taulukko 2 Esimerkki XML-dokumentin elementtien nimien siistimisestä

Ennen siistimistä	Siistimisen jälkeen
<com.trivore.mysync.framework.core.Account> <globalKey>cn=johndoe,ou=account,orgCode=testorg, ou=data,dc=mysync</globalKey> <cn>johndoe</cn> <displayName>Doe, John</displayName> <validTo>1337155098550</validTo> <userClass>endUser</userClass> <givenName>John</givenName> <sn>Doe</sn> <syncMail> <com.trivore.mysync.framework.core.SyncMail> <cn>mail.0</cn> <mail>doe@example.com</mail> <mailProtocol>imap</mailProtocol> </com.trivore.mysync.framework.core.SyncMail> <com.trivore.mysync.framework.core.SyncMail> <cn>mail.1</cn> <mail>doe@gmail.com</mail> <mailProtocol>imap</mailProtocol> </com.trivore.mysync.framework.core.SyncMail> </syncMail> </com.trivore.mysync.framework.core.Account>	<account> <globalKey>cn=johndoe,ou=account,orgCode=test org,ou=data,dc=mysync</globalKey> <cn>johndoe</cn> <displayName>Doe, John</displayName> <validTo>1337154567916</validTo> <userClass>endUser</userClass> <givenName>John</givenName> <sn>Doe</sn> <syncMails> <syncMail> <cn>mail.0</cn> <mail>doe@example.com</mail> <mailProtocol>imap</mailProtocol> </syncMail> <syncMail> <cn>mail.1</cn> <mail>doe@gmail.com</mail> <mailProtocol>imap</mailProtocol> </syncMail> </syncMails> </account>

4 SUUNNITTELU

4.1 Viestirakenteet

Asiakasohjelman ja palvelun välisessä kommunikoinnissa päätettiin käyttää yleisiä XML-dokumentteja. Käytetty XStream-kirjasto mahdollistaa muunnoksen tavallisten Java-objektien ja XML-dokumenttien välillä vaivattomasti. XStream mahdollistaa myös JSON-viestirakenteiden käytön, ja tätä mahdollisuutta on käytetty hyväksi toteutetussa ohjelmointirajapinnassa. JSON-muodon tuen lisäys ei aiheuta juurikaan lisätyötä.

Perusobjektit, kuten ”laite” tai ”kontakti”, riittävät joihinkin toimintoihin, kuten esimerkiksi ”lue laitteen tiedot” tai ”päivitä kontakti”. Monet muut toiminnat tarvitsevat kuitenkin lisäinformaatiota. Esimerkiksi ”lisää uusi laite” vaatii lisätietona mallilaitteen tunnuksen. Tällaisia tapauksia varten on luotu säiliöobjekteja, joihin sisältyy mahdollisten perusobjektien lisäksi lisätietoja.

Jotkin perusobjektit sisältävät niin paljon tietoa, että koko objektin lataaminen tietoverkon kautta voi kuluttaa turhan paljon tiedonsiirtokapasiteettia. Esimerkiksi organisaation kaikkien laitteiden listaaminen yhdessä on tällainen tapaus. Tällaisissa tilanteissa palvelin käyttääkin linkkiobjekteja jotka sisältävät laitteista vain rajallisen määrän tietoa ja linkin resurssiin josta yksittäisen laitteen kaikki tiedot voi hakea.

4.2 Resurssien sijainnit

Resurssit ovat kohteita joiden kautta palvelua voidaan käskyttää. Ohjelmointirajapinnan käyttäjät kutsuvat näitä resursseja HTTP-pyynnöillä. Tavallisilla Internet-sivustoilla resurssit ovat sivuja ja kuvia. Resurssin sijainti määritellään hierarkkisesti käyttäen tekstijonoa, jossa hierarkkiset osat erotellaan kauttaviivoilla. Taulukossa 3 on palvelussa olevia resurssien sijainteja selityksineen. Riippuen HTTP-pyynnössä määritetystä HTTP-komennosta, palvelin reagoi kutsuun eri tavoin.

Taulukko 3 Resurssien sijainteja

Resurssi	Selite
`/org`	Lista organisaatioista joihin lukuoikeus
`/org/{orgCode}`	Organisaation käsittely
`/org/{orgCode}/account`	Organisaation käyttäjätilien käsittely
`/org/{orgCode}/account/{accountCn}`	Yksittäisen käyttäjätilin käsittely
`/org/{orgCode}/account/{accountCn} /pim/{pimType}`	Käyttäjätilin kontakti- ja muiden tietojen käsittely
`/org/{orgCode}/device`	Organisaation laitteiden käsittely
`/org/{orgCode}/device/{deviceCn}`	Yksittäisen laitteen käsittely
`/org/{orgCode}/device/{deviceCn} /task`	Laitteen tukemien toimintojen listaus
`/org/{orgCode}/device/{deviceCn} /task/{taskName}`	Laitteen toiminnon suorittaminen
`/org/{orgCode}/group`	Käyttäjäryhmien käsittely
`/org/{orgCode}/template/account`	Käyttäjämallien käsittely
`/org/{orgCode}/template/device`	Laitemallien käsittely
`/org/{orgCode}/log`	Tapahtumatietojen luku
`/org/{orgCode}/report`	Raporttien listaus
`/org/{orgCode}/report/{reportId}`	Yksittäisen raportin luku
`/deviceinfo`	Laitetyyppien käsittely
`/deviceinfo/{brand}/{cn}`	Yksittäisen laitetyypin käsittely
`/operator`	Puhelinoperaattorien käsittely
`/operator/{c}/{cn}`	Yksittäisen puhelinoperaattorin käsittely

4.3 Suodatus

Monissa yhteyksissä, joissa palvelu palauttaa useita tietorakenteita (esimerkiksi listattaessa laitteita), olisi toivottavaa, että hakutuloksia voitaisiin rajata parametrien mukaisesti. Suodatustoiminnot voidaan lisätä URI:n parametreina. Koska mySync-palvelu tukee suodatettua hakua myös sisäisesti, suodatus-

mahdollisuuden lisäyksestä seuraa suorituskykyparannusta myös palvelimen sisällä.

Suodatinparametrit lisätään URI:in muodossa {parametri}_{suodatin}={arvo}. Suodattimia voidaan lisätä useita yhtä aikaa, jolloin ne yhdistetään käyttämällä loogista funktiota ”AND”. Taulukossa 4 on esimerkkejä suodattimista.

Taulukko 4 Esimerkkejä suodattimista

Suodatin	Arvo vaaditaan	Esimerkki
On yhtä kuin arvo	Kyllä	cn_equals=johndoe
On eri kuin arvo	Kyllä	cn_notequals=johndoe
Sisältää arvon	Kyllä	cn_contains=john
Ei sisällä arvoa	Kyllä	cn_notcontains=john
Alkaa arvolla	Kyllä	cn_startswith=john
Ei ala arvolla	Kyllä	cn_notstartswith=john
Loppuu arvoon	Kyllä	cn_endswith=doe
Ei lopu arvoon	Kyllä	cn_notendswith=doe
On olemassa (ei ole tyhjä)	Ei	cn_exists
Ei ole olemassa (on tyhjä)	Ei	cn_notexists
Suurempi kuin arvo	Kyllä	locked_greaterthan=5
Pienempi kuin arvo	Kyllä	locked_lessthan=5

5 TOTEUTUS

5.1 Käyttäjien tunnistus ja käyttäjäoikeudet

Ohjelmointirajapinnan kautta voidaan päästä käsiksi hyvin yksityisiin tietoihin, joten palvelun käyttöä täytyy voida rajoittaa käyttäjäkohtaisesti. Rajapintaa käytetään samoilla käyttäjätunnuksilla kuin muitakin järjestelmän osia. Hierarkisesti rakennettu tietokanta ja sen sisäiset pääsyoikeusrajoitteet huolehtivat siitä, että eri organisaatioiden käyttäjät eivät pääse käsiksi toisiin organisaatioihin. Tietyt käyttäjät on määritetty oman organisaationsa haltijoiksi, ja heillä on mahdollisuus hallita kaikkia oman organisaation käyttäjiä ja laitteita. Muut, ns. loppukäyttäjätilit, voivat hallita vain omaa tiliään ja omia laitteitaan. Tietokannan rajoitteet helpottavat rajapintapalvelun tekoa, sillä nämä rajoitteet toimivat näkymättömästi taustalla.

Käyttäjien tunnistus tapahtuu HTTP-pyynnön yhteydessä käyttäen ”Basic Authentication” tekniikkaa [8]. Tekniikan etuna on yksinkertainen toteutus sekä palvelimen että asiakasohjelman puolella. Haittapuolena on huono turvallisuus, ellei HTTP-yhteyttä ole salattu SSL-tekniikalla. Koska mySync-palvelut poikkeuksetta käyttävät salausta, tämä ei ole ongelma.

5.2 XML-skeema

Jotta XML-rakenteita voidaan luoda asiakasohjelmissa, niiden rakenne ja sisältö täytyy määritellä tarkasti. Tämä rakenne määritetään XML-skeemassa. Skeema sisältää selityksen ja käyttöparametrit jokaiselle elementille, jota järjestelmän XML-rakenteissa käytetään.

Skeeman määrittelyyn on erilaisia kieliä. DTD (Document Type Definition) on perinteinen määrittelykieli. Se ei kuitenkaan sisällä yhtä paljon määrittelymahdollisuuksia kuin XSD (XML Schema Document), jota W3C suosittelee. Tässä työssä skeema on määritetty XSD-kielellä. Koodiesimerkissä 2 on osa skeemasta. Osassa määritetään laitetyypin mahdolliset tiedot.

Koodiesimerkki 2 Osa XSD-tyyppisestä XML-skeemasta

```
<xs:element name="deviceInfo">
  <xs:complexType>
    <xs:all>
      <xs:element name="globalKey" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="createTimestamp" type="timestamp" maxOccurs="1" minOccurs="0" />
      <xs:element name="modifyTimestamp" type="timestamp" maxOccurs="1" minOccurs="0" />
      <xs:element name="cn" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="displayName" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="fqdn" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="brand" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="manufacturer" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="modelTypeName" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="modelTypeCode" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="hardwareVersion" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="hardwareDate" type="timestamp" maxOccurs="1" minOccurs="0" />
      <xs:element name="firmwareVersion" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="firmwareDate" type="timestamp" maxOccurs="1" minOccurs="0" />
      <xs:element name="softwareVersion" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="softwareDate" type="timestamp" maxOccurs="1" minOccurs="0" />
      <xs:element name="softwareUpdated" type="timestamp" maxOccurs="1" minOccurs="0" />
      <xs:element name="serialNumber" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="deviceType" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="owner" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="manager" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="description" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="statusText" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="seeAlso" type="xs:string" maxOccurs="1" minOccurs="0" />
      <xs:element name="features" maxOccurs="1" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="feature" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element ref="mobileDeviceCapabilities" maxOccurs="1" minOccurs="0" />
    </xs:all>
  </xs:complexType>
</xs:element>
```

XStream-kirjasto, joka muuntaa Java-objektit automaattisesti XML-rakenteiksi, ei käytä skeemaa, mutta sen on silti tuotettava skeeman mukaisia rakenteita. Palvelua rakennettaessa ja ylläpidettäessä on siis huolehdittava XStreamin tuotosten ja skeeman yhtäpitävyydestä.

5.3 Resurssien toteutus

Restlet-kehys sisältää reititinluokkia, joita käytetään URI:n tulkitsemiseen ja oikean resurssin löytämiseen. Reitittimet huolehtivat myös käyttäjän tunnistamisesta kun käytetään resursseja jotka vaativat erityisoikeuksia.

Jokaista resurssia varten on tehty oma luokka, joka sisältää metodit kaikkia haluttuja HTTP-komentoja varten Restlet-kehyksen käyttötavan mukaisesti. Metodit on merkitty annotaatioilla @Get, @Post, @Delete sekä @Put, jotka ilmaisevat Restlet-kehykselle metodien käyttötarkoituksen. Jos metodin on tarkoitus vastaanottaa tietoa HTTP-komennon lähettäjältä, metodille on määritetty parametri josta tiedon voi lukea. Restlet-kehys huolehtii parametrin arvon välittämisestä. Jos HTTP-komennon on tarkoitus palauttaa tietoa, se tapahtuu palauttamalla metodissa tieto upotettuna Representation-luokkaan.

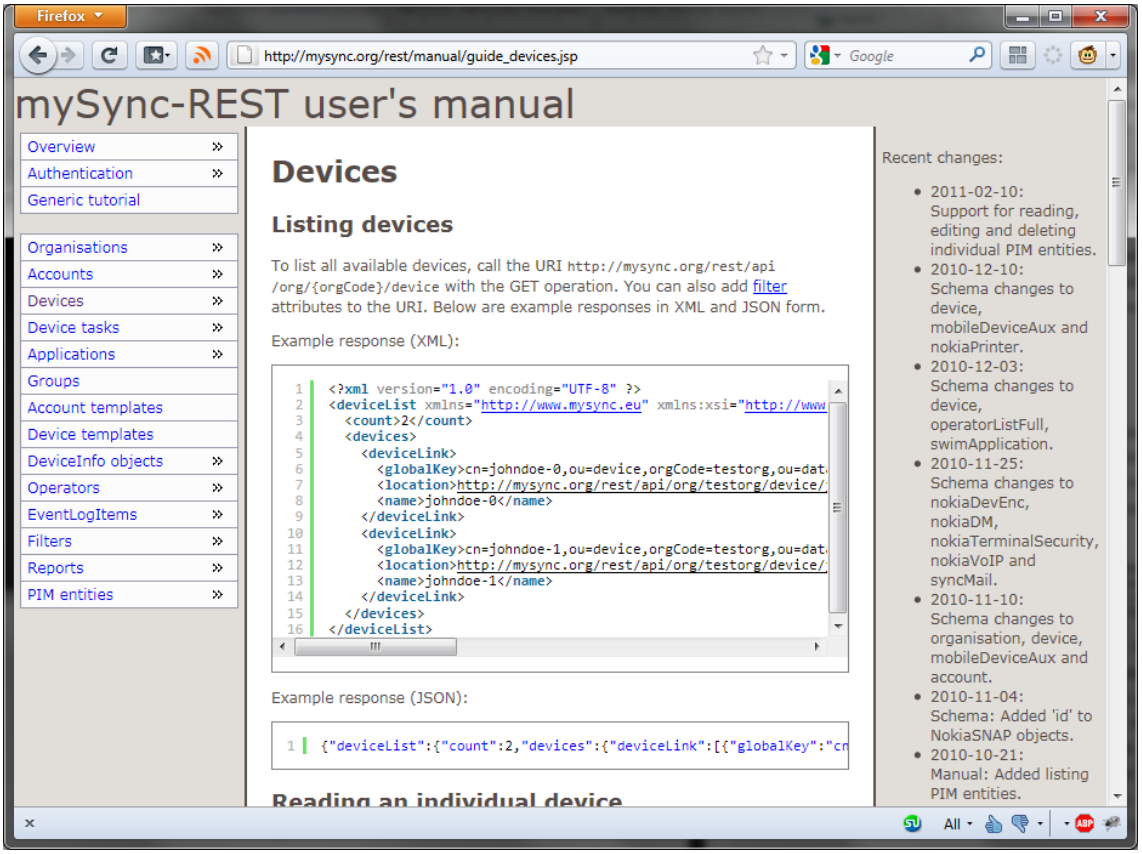
5.4 Ohjeistus

Ohjelmointirajapinta on tarkoitettu ensi kädessä ulkopuolisten ohjelmoijien käyttöön, joten heille on tarjottava riittävä ohjeistus sen käyttöön. REST-tyyppisten palveluiden käyttö ei ole mahdollista ilman kattavaa dokumentointia arkkitehtuurityylin vapaamuotoisuuden takia.

Rajapinnan ohessa toteutettiin ohjesivusto (kuva 2), joka on saatavissa ohjelmointirajapintapalvelimelta. Sivusto sisältää myös esimerkkejä XML-viestirakenteista. Koska sivusto on rakennettu JSP-tekniikalla, niihin on voitu upottaa kutsuja suoraan rajapinnassa käytettäviin resursseihin ja XML-muuntajiin.

Sivustolla näytettävät XML-esimerkkilistaukset luodaan aina uudestaan reaaliajassa. Tämä takaa sen, että esimerkit ovat ajanmukaisia. Tällä tavoin toimittaessa muutokset järjestelmän muissa osissa eivät useinkaan vaadi muutosta itse dokumentaatioon.

Myös ohjeissa näytettävät esimerkkilinkit sisältävät aina oikean palvelimen osoitteen. Koska rajapintapalvelua suoritetaan useilla eri palvelimilla, on ymmärrettävyyden kannalta tärkeää, että esimerkeissä on nykyisen palvelimen tunnus.

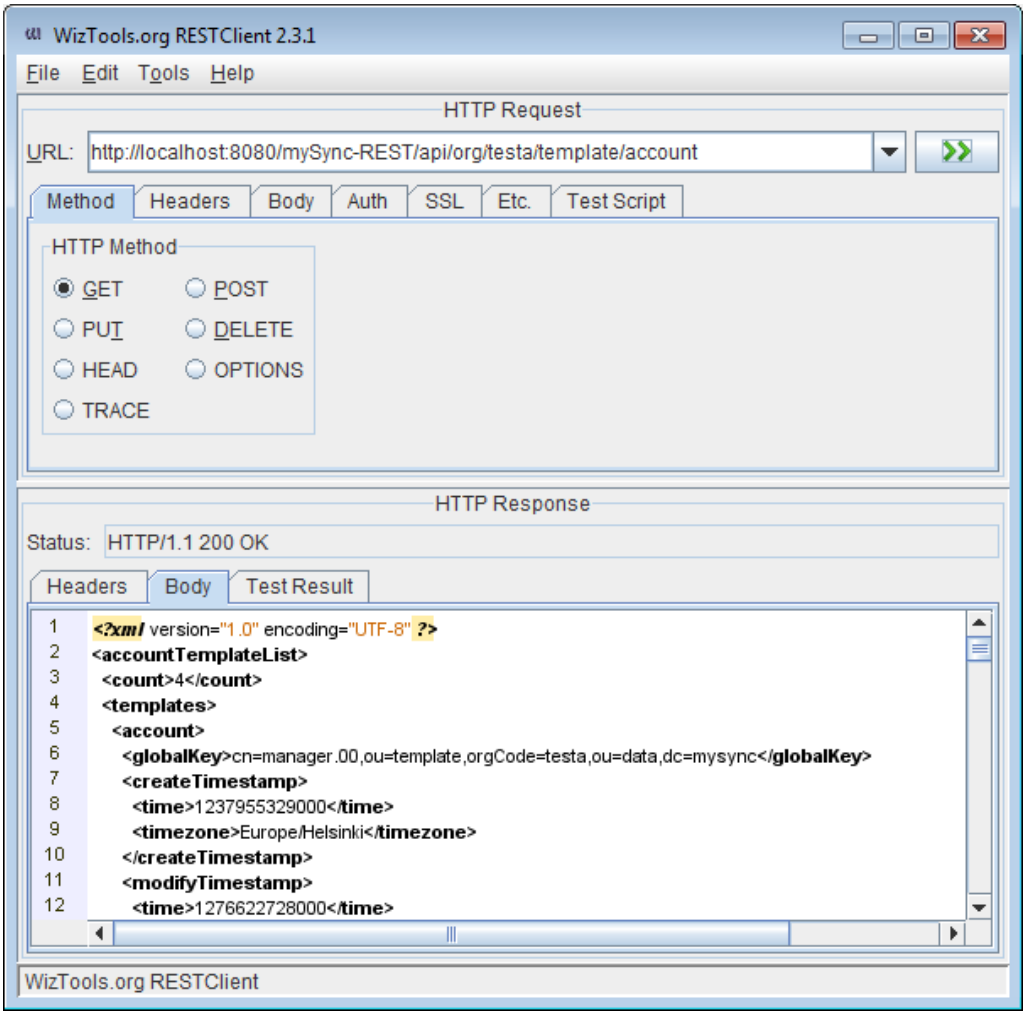


Kuva 2 Kuvakaappaus palvelun ohjeistuksesta

6 TESTAUS JA KÄYTTÖÖNOTTO

6.1 Testaus

Rajapinnan käyttöä testattiin jatkuvasti uusia ominaisuuksia lisättäessä käyttäen RESTClient-sovellusta [9]. RESTClient on yleiskäyttöinen HTTP-protokollaan pohjautuvien palveluiden testaustyökalu (kuva 3). Se mahdollistaa pyyntöjen ominaisuuksien ja viestirungon määrittämisen ja palvelun vastauksen tarkastelun.



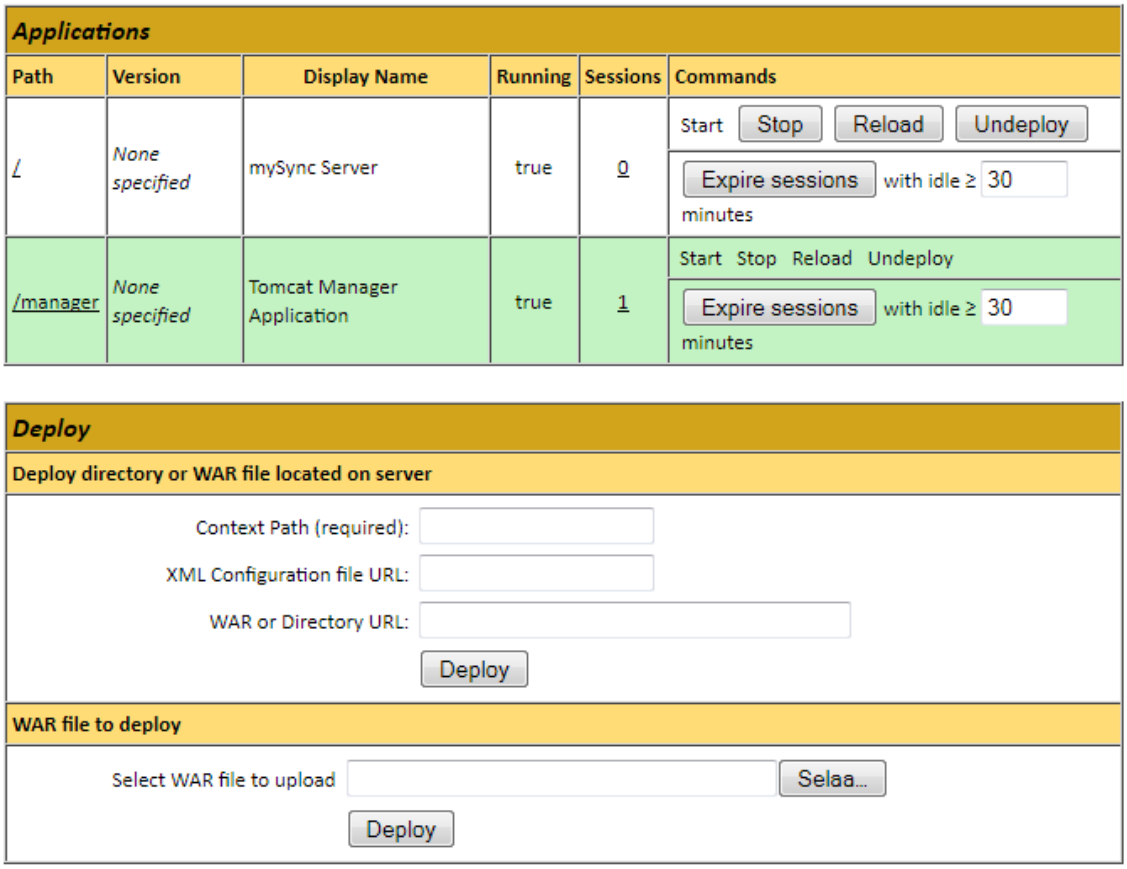
Kuva 3 Kuvakaappaus RESTClient-sovelluksesta

Automaattista toiminnan testausta varten tehtiin myös joukko JUnit-yksikkötestejä. Näiden testien suorittaminen auttaa havaitsemaan ongelmia, joita voi tulla muutoksia tehdessä itse rajapintapalveluun tai sen käyttämiin sovelluskirjastoihin.

6.2 Palvelun käyttöönotto

Rajapintapalvelu on liitetty mySync-järjestelmän asennuspaketin rakentavaan järjestelmään. Asennuspakettiin kootaan kaikki mySync-palvelun toteuttavat palvelut. Paketti luodaan ohjelmallisesti käyttäen Apache Ant -ohjelmistoa joka kääntää palveluiden lähdekoodin Java-tavukoodiksi ja kokoaa ne WAR-paketeiksi joita voidaan suorittaa Servlet-palvelimessa. Toteutettu palvelu tulee käyttöön automaattisesti mySync-järjestelmän asennuksen tai päivityksen yhteydessä.

Järjestelmän täydellinen asennus tapahtuu purkamalla asennuspaketti Linux Debian -palvelimelle ja suorittamalla asennuksen suorittava komento. Palveluiden osittaisen päivityksen voi tehdä käyttämällä Apache Tomcat -palvelimen hallintatyökaluja. Yksi työkaluista on palvelimen oma graafinen WWW-käyttöliittymä (kuva 4). Se mahdollistaa kaikkien servlet-palveluiden päivittämisen yhdellä WAR-paketilla.



Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	mySync Server	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):
XML Configuration file URL:
WAR or Directory URL:
Deploy

WAR file to deploy

Select WAR file to upload Selaa...
Deploy

Kuva 4 Apache Tomcat -palvelimen päivitystyökalu

7 YHTEENVETO

Työssä suunniteltiin ja toteutettiin ohjelmointirajapinta, jonka tarkoitus on mahdollistaa mySync-arkkitehtuurin hallinnointi loppukäyttäjän ehdoilla. Rajapintaa käytetään jo sisäisesti palvelinten väliseen kommunikointiin.

Työtä aloitettaessa tärkeää oli löytää oikeat työkalut projektin nopeaan käynnistämiseen. Sovelluskehys Restlet ja XML-muunnoskirjasto XStream hoitivat suuren osan tällaisen palvelun vaatimasta työstä.

Rajapintaa on työn tekemisen jälkeen laajennettu sisältämään lisää toimintoja. Käytetty Restlet-sovelluskehys on mahdollistanut laajennusosien lisäyksen todella vaivattomasti. Tämä on osoittanut sovelluskehyksen valinnan osuneen oikeaan.

LÄHTEET

[1] Fielding, R. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* [www-dokumentti]
Saatavilla http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (luettu 16.5.2011)

[2] Sandoval, J. 2009. *RESTful Java Web Services*. Packt Publishing Ltd.

[3] Singh, T. 2009 *REST vs. SOAP - The Right WebService* [www-dokumentti]
Saatavilla http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/ (luettu 5.6.2011)

[4] Richardson L., Ruby S. 2007 *RESTful Web Services*. O'Reilly Media, Inc.

[5] Burke, B. 2010. *RESTful Java with JAX-RS*. O'Reilly Media, Inc.

[6] *Restlet web framework*. [www-dokumentti]
Saatavilla http://www.restlet.org/about/introduction (luettu 16.5.2011)

[7] Bangalore, R. 2008 *Use XStream to serialize Java objects into XML*. [www-dokumentti]
Saatavilla https://www.ibm.com/developerworks/java/library/x-xstream/ (luettu 16.5.2011)

[8] Berners-Lee, T., Fielding, R., Frystyk, H. 1996 *Hypertext Transfer Protocol -- HTTP/1.0* [www-dokumentti]
Saatavilla http://www.ietf.org/rfc/rfc1945.txt (luettu 16.5.2011)

[9] *RESTClient*. [www-dokumentti]
Saatavilla: http://code.google.com/p/rest-client/ (luettu 16.5.2011)