Tiina Kuisma

LDAP ja käyttäjähallinta

Tietotekniikan
kandidaatintutkielma
15.12.2006

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Tiina Kuisma
Yhteystiedot: tihyvari@cc.jyu.fi
Työn nimi: LDAP ja käyttäjähallinta
Title in English: LDAP and Identity Management
Työ: Tietotekniikan kandidaatintutkielma
Sivumäärä: 26
Tiivistelmä: Tutkielmassa tarkastellaan keskitettyyn käyttäjähallintaan kehitettyjä protokollia sekä standardeja. Tarkemman tarkastelun kohteeksi on valittu nykyään paljon yleistynyt toteutus LDAP eli *Lightweight Directory Access Protocol*.
English abstract: The thesis examines some protocols and standards, which have been developed to solve problems related to the identity management. A closer look is devoted to LDAP i.e. *Lightweight Directory Access Protocol* because it has become a common implementation protocol.
Avainsanat: käyttäjähallinta, SAML, Shibboleth, LDAP, Lightweight Directory Access Protocol, LDAP-mallit, APIt.
Keywords: identity management, SAML, Shibboleth, LDAP, Lightweight Directory Access Protocol, LDAP Models, APIs.

Sisältö

1 Johdanto

Monet yritykset ja organisaatiot kamppailevat nykyään käyttäjähallintaan liittyvien ongelmien parissa. Tietojärjestelmät monimutkaistuvat ja vanhojen järjestelmien rinnalle otetaan käyttöön uusia. Samalla käyttäjien ja käyttöoikeuksien hallinnointi muuttuu yhä haastavammaksi. Sen myötä kasvaa tarve keskitetylle ja hyvin suunnitellulle käyttäjähallinnalle. Tarvittavat toiminnalliset vaatimukset asettavat myös vaatimuksia muun muassa teknologiakehitykselle ja standardoinnille.

Käyttäjähallintaan liittyviä ongelmia voidaan tarkastella niin sovellusten käyttäjien kuin niitä ylläpitävien näkökulmasta. Sovellusten käyttäjän näkökulmasta ongelmaksi muodostuu jokaiseen sovellukseen erikseen sisään kirjautuminen ja sen myötä pahimmillaan useiden käyttäjätunnusten ja salasanojen muistaminen. Salasanojen muistamista hankaloittaa se, että joihinkin järjestelmiin joutuu salasanan vaihtamaan esimerkiksi kolmen kuukauden välein. Useasti tämän seurauksena käyttäjätunnuksia ja salasanoja kirjoitetaan lapuille muistiin ja saatetaan pahimmassa tapauksessa jopa kiinnittää näkyville monitorin kylkeen.

Sovellusten hallinnan näkökulmasta ongelmia aiheuttaa jokaisen sovelluksen omat erilaiset prosessit, toimintatavat ja työkalut käyttäjätietojen hallintaan. Täten uusi sovellus lisää merkittävästi tietohallinnon työntekijöiden työmäärää ja hankaloittaa turhaan organisaation järjestelmien käyttäjien tietojen ja oikeuksien hallintaa kokonaisuutena. Suurimmaksi ongelmaksi yritysmaailmassa muodostuu helposti talosta poislähteneiden työntekijöiden tunnusten sulkeminen. Auki jääneet tunnukset aiheuttavat tietoturvaongelman koko organisaatiossa.

Kandidaatintutkielma tarkastelee lyhyesti käyttäjähallintaa ja sen tueksi kehitetyistä protokollista ja standardeista erityisesti LDAPia. Luvussa 2 käsitellään käyttäjähallintaa yleisesti sekä esitellään lyhyesti muutamia protokollia ja standardeja. Varsinaisesti kandidaatintutkielma painottuu luvusta 3 lähtien esitettävään LDAP-määrittelyyn, koska sillä on ollut merkittävä rooli uusimmissa ja yleistyneimmissä käyttäjähallintaratkaisuissa. Luvussa 3 esitellään LDAPin määrittelyä sekä sen rakennetta. Varsinaisesti LDAPin arkkitehtuuriin perehdytään luvussa 4. Lopuksi luvussa 5 esitellään LDAPin käyttöön avuksi kehitettyjä ohjelmointirajapintoja. Kandidaatintutkielman esimerkeissä on käytetty lähtökohtana Netscapen Java-kielelle toteuttamia rajapintoja.

2 Käyttäjähallinta ja sen protokollat

Käyttäjähallinnassa on Wikipedian määritelmän [10] mukaan kyse organisaatiossa sovituista kokonaisvaltaisista käytänteistä, jotka koskevat niin liiketoiminnan prosesseja, toimintatapoja kuin teknologioitakin. Näiden käytänteiden avulla organisaatio voi edesauttaa ja kontrolloida työntekijöidensä pääsyä kriittisiin järjestelmiin ja resursseihin. Samalla ne voivat suojella salattavia henkilöstö- ja liiketoimintatietoja siten, etteivät ulkopuoliset pääse niihin käsiksi. Ulkopuolisilla voidaan käsittää niin organisaatioon kuulumattomat kuin siihen kuuluvat osapuolet, joiden ei kuitenkaan työnsä puolesta kuulu päästä kyseisiin tietoihin käsiksi. Luku perustuu pääasiassa lähteisiin [1] ja [12].

2.1 Käyttäjähallinnan käsitteitä

Keskitettyyn käyttäjähallintaan on kehitetty erilaisia standardeja ja protokollia. Pääsääntöisesti kehitetyt protokollat koskevat erilaisia autorisointi- ja autentikointiprosesseja. **Autorisoinnilla** tarkoitetaan käyttäjän valtuuttamista eli hänelle määritellään käyttöoikeuksia tarvittaviin sovelluksiin sekä resursseihin. **Autentikoinnilla** puolestaan tarkoitetaan käyttäjän todentamista hänen yrittäessä päästä käsiksi suojattuihin resursseihin. Todentaminen suoritetaan yleensä kysymällä käyttäjätunnusta ja salasanaa.

Protokollassa määritellään yleensä toimenpiteet ja rakenteet, jotka liittyvät tietojen käsittelyyn, tallennukseen ja siirtoon. Siinä määritellään muun muassa tallennettujen tietojen ja tietokantojen formaatit eli tallennusmuodot. Siinä voidaan määritellä myös siirrettävien tietojen esitystavat ja siirtomekanismit.

Uusia protokollia kehitetään jatkuvasti sekä vanhoja jatkokehitetään muuttuvien tarpeiden mukaisiksi, joten tarjonta erilaisten protokollien suhteen on hyvin elävää ja monipuolista. Muutamat protokollat ovat kuitenkin niin sanottuja perusstandardeja, joita hyödynnetään uusien kehittämisessä. Luvuissa 2.2- 2.4 esitellään joitakin käyttäjähallintaan keskeisesti liittyviä protokollia sekä kehitelmiä.

2.2 SAML

SAML (Security Assertion Markup Language) on XML-standardi (Extensible Markup Language) autorisointi- sekä autentikointitietojen vaihtoon toimialueiden (engl. *domain*) välillä. SAML on OASIS Security Services Technical Committeen kehittämä. Se on kehitetty ratkaisemaan kertakirjautumiseen (engl. *single sign-on*) liittyviä ongelmia [3].

OASIS-määritelmän [1, s. 13] mukaan SAMLin perusosat ovat

- väittämät (engl. *assertions*),
- protokolla (engl. *protocol*),
- bindings ja
- profiilit (engl. *profile*).

SAML-väittämiä siirretään identiteetin tuottajalta eli lähettävältä toimialueelta palvelun tuottajalle eli vastaanottavalle toimialueelle. Väittämät pitävät sisällään tietoa, jonka mukaan vastaanottava toimialue tekee oikeuspäätöksen. Erilaisia väittämiä on autentikointi-, attribuutti- ja autorisointiväittämiä. Autentikointiväittämällä välitetään viesti, jossa todetaan kyseisen osapuolen olevan autentikoitu sovitulla tavalla. Attribuuttiväittämän avulla puolestaan välitetään kyseessä olevaan osapuoleen liittyviä tietoja ja ominaisuuksia. Autorisointiväittämä kertoo, mitä käyttäjät saavat tehdä ja mitä eivät. SAML-väittämiä voidaan myös laajentaa määrittelemällä omia väittämiä tarpeiden mukaan.

SAML määrittelee myös **protokollan**, jolla lähetetään pyyntöjä ja vastauksia väittämien saamiseksi. SAML-protokolla onkin yksinkertainen pyyntö-vastausprotokolla (engl. *request-response*). OASIS [1, s. 21] määrittelee toiminnon siten, että niin sanottu SAML-pyytäjä lähettää SAML-pyyntöviestin vastaajalle, joka vastaavasti lähettää vastausviestin SAML-pyytäjälle. Tietojenvaihto tapahtuu reaaliaikaisesti Web Services -ympäristössä.

SAMLin perusosalle **bindings** ei ole suoraa suomenkielistä käännöstä. Niillä käsitetään niin sanottuja liitoksia (engl. *mappings*), joita käytetään esimerkiksi SAMLin pyyntö-vastausviestinnässä. SAMLin version 1.1 määritelmän ”SOAP over HTTP binding” mukaan SAML-tiedot kuljetetaan SOAP-viesteissä (Simple Object Access Protocol) HTTP:n päällä. SAML SOAP binding määrittelee, kuinka SAMLin pyyntö- ja vastausviestienvaihto on liitetty SOAP-viestinvaihtoon.

OASIS-yhteisön [1, s. 15] mukaan SAMLin **profiilit** määrittelevät, kuinka SAMLin väittämiä, protokollia ja bindingseja yhdistellään ja rajoitetaan. Yhdistelyllä ja

rajoittamisella voidaan saada aikaan parempi toiminnallisuus riippuen käyttötapauksesta. Profiilien avulla määritellään myös, kuinka SAMLia voidaan käyttää erilaisiin tarkoituksiin. Kehitystyö SAMLin ympärillä on vielä kesken, eikä sen käyttöä kaikissa tilanteissa ole vielä määritelty. Siitä onkin yleisesti käytössä tällä hetkellä kolmas versio SAML 2.0, joka hyväksyttiin maaliskuussa 2005.

2.3 Hakemistopalvelut ja Shibboleth

Eräs käyttäjähallintaan kytkeytyvä ongelma on se, etteivät sovellukset pysty hyödyntämään toistensa hakemistoja eli tietokantoja. Tästä johtuen yritykset ajavat eriytettyjä ja itsenäisiä hakemistoja sekä sovelluksia, jotka eivät ole vuorovaikutuksessa saati luottosuhteessa toisiinsa. Tämän seurauksena voi olla, että tietoja joudutaan päivittämään useampaan kertaan, joka puolestaan voi aiheuttaa tallennettavien tietojen ristiriitaisuutta.

Jokaisella sovelluksella on myös oma yksityinen identiteettivarasto, jossa sijaitsee sovelluksen käyttämät käyttäjätunnukset ja salasanat. Jokainen yksityinen sovellushakemisto tarvitsee omat käytänteensä käyttäjien perustamiseen, hallintaan sekä kulunvalvontaan. Tämä aiheuttaa Spencerin [5] mukaan jatkuvia ongelmia yrityksen teknologiainfrastruktuurille. Ratkaisuksi ongelmalle on kehitetty niin sanottuja **hakemistopalveluja**.

SAMLiin perustuen on kehitetty arkkitehtuuri ja avoin lähdekooditoteutus nimeltä Shibboleth. Shibboleth on tarkoitettu hajautettuun autentikointiin sekä pääsynvalvontaan. Shibboleth on erityisesti yliopistomaailmassa käytössä oleva toteutus.

Wikipedian [12, s. 2] määritelmän mukaan Shibbolethin toteutuksessa hyödynnetään niin sanottua yhtenäistettyä identiteettiä. **Yhtenäistetyllä identiteetillä** tarkoitetaan sitä, että yhden toimialueen sisällä oleva käyttäjätieto annetaan toisen toimialueen käyttöön. Tämä mahdollistaa toimialueiden kesken kertakirjautumisen ja vähentää ylläpitäjien tarvetta ylläpitää käyttäjätunnuksia ja salasanoja useassa paikassa.

2.4 Muita protokollia

Myös ohjelmistotalot ovat kehitelleet omia käyttäjähallinnan protokolliaan. Tällaisia ovat muun muassa Microsoftin Passport sekä Microsoftin IBM:n kanssa kehittämä WS-Federation. **WS-Federation** -protokolla toimii WS-Securityn päällä. Se on ke-

hitetty tukemaan erityisesti ajatusta yhtenäistetystä identiteetistä. WS-Federation -erittely määrittää niin sanotun federaation (engl. *federation*), joka on kokoelma alueista, joiden välille on muodostunut luottamussuhde. Luottamussuhteen taso voi vaihdella, mutta pääsääntöisesti se pitää sisällään autentikoinnin ja autorisoinnin.

WS-Federation esittelee palveluja ja määritelmiä, joita käytetään yhtenäistetyn identiteetin mallissa. Nämä palvelut ovat

- Security Token Service,
- Identity Provider,
- Attribute Service ja
- Pseudonym Service.

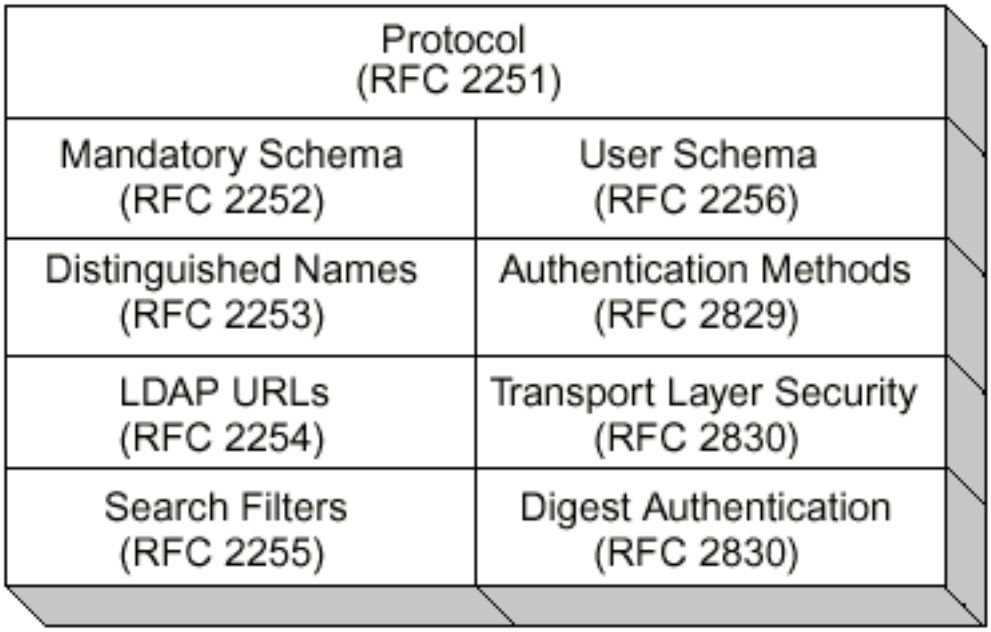
Pfitzmann ja Waidner esittelevät artikkelissaan [2] omaksi ehdotelmakseen protokollien joukkoon BBAE:n (Browser-Based Attribute-Exchange Protocol). He suunnittelivat BBAE:n yhteensopivaksi profiiliksi olemassa oleville standardointikehitelmille, kuten SAML ja WS-Federation. He pyrkivät BBAE:n kehityksessä muita kehitelmiä parempaan tietoturvaan sekä skaalautuvuuteen.

3 LDAP (Lightweight Directory Access Protocol)

LDAP on hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla, joka toimii TCP/IP:n päällä. Se syntyi yksinkertaistettuna vaihtoehtona monimutkaisemmalle X.500-hakemistopalvelulle. Alkuperäinen versio LDAPista kehitettiin Michiganin yliopistossa 1993, mutta nykyään sen kehityksestä vastaa IETF eli International Engineering Task Force. Luku perustuu pääosin lähteisiin [7, s. 20] ja [13, s. 47].

3.1 LDAPin versiot ja ydinstandardit

Ensimmäinen laajasti käytössä ollut versio LDAPista oli LDAPv2, joka kuvattiin RFC 1777:ssä vuonna 1995. LDAPista on nykyään laajasti käytössä versio LDAPv3 ja se kuvataan RFC 2251:ssä [8]. Koska LDAP on paljon muutakin kuin pelkkä protokolla, siihen liittyviä kuvauksia löytyy kymmenistä RFC:eistä. Kuvassa 3.1 on kuvattu niin sanotut LDAP-ydinstandardit versiossa kolme [7, s. 20].



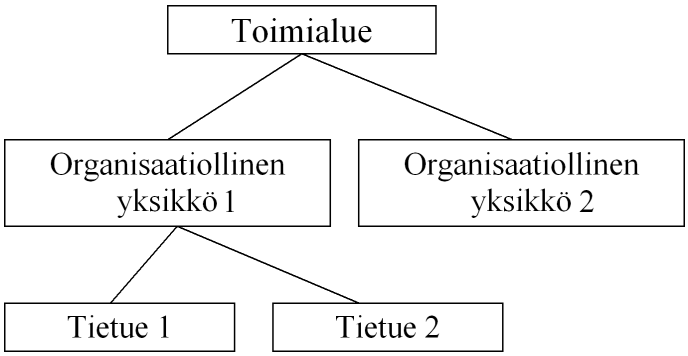
Kuva 3.1: LDAP-ydinstandardien RFC-julkaisut.

3.2 LDAPin osat

Mark Wilcox määrittelee kirjassaan [13, s. 47] LDAPin koostuvan kolmesta osasta, jotka ovat tietomuoto, protokolla ja API eli ohjelmointirajapinta.

Tietomuoto määrittelee, kuinka hakemistotieto tallennetaan ja haetaan. Tietomuoto on suunniteltu olevan sekä monialustainen (engl. *cross-platform*) että monikulttuurinenkin. Se on suunniteltu siten, että sillä on käytössään globaali nimimäärittely, josta käytetään termiä nimiavaruus (engl. *namespace*). Tieto tallennetaan LDAP-palvelimelle sekä hierarkisessa että suhteellisessa muodossa. Hierarkisuudella tarkoitetaan sitä, että tallennettavat tietueet (engl. *entry*), juuritietuetta lukuunottamatta, sijoittuvat puurakenteessa toisen tietueen alapuolelle. Suhteellisuus ilmenee puolestaan siinä, että tietueita voidaan ryhmitellä yhteen.

Ylintä tasoa **LDAP-hierarkian** puurakenteessa kutsutaan toimialueeksi (engl. *domain*). Toimialueita voi olla useampia riippuen toteutuksesta. Puun oksat muodostuvat organisaatiollisista yksiköistä. Yleensä ne ovat organisaation osastoja, mutta ne voivat olla mitä tahansa alajaotteluja kyseiselle organisaatiolle. Tietomuodon määrittelyn mukaan niin sanottuja lehtiä (engl. *leaf*) ovat ne tietueet, jotka eivät ole toimialueita, eikä organisaatiollisia yksikköjä. Kuvassa 3.2 esitetään hyvin yksinkertainen LDAP-hierarkiarakenne.



Kuva 3.2: LDAP-hierarkia.

Protokolla määrittelee, kuinka asiakkaat ja palvelimet ovat vuorovaikutuksessa toistensa kanssa. Asiakkaan ollessa vuorovaikutuksessa LDAP-palvelimen kanssa, se käy läpi kolme perusvaihetta. Nämä vaiheet ovat yhteydenotto palvelimelle, operaatioiden toteuttaminen sekä lopuksi yhteyden sulkeminen. LDAPissa käytettäviä operaatioita esitellään tarkemmin luvusta 4.3 lähtien.

API määrittelee, kuinka muut ohjelmat voivat olla vuorovaikutuksessa LDAP-palvelinohjelmiston kanssa. Tarkemmin LDAPiin liittyviä ohjelmointirajapintoja käsitellään luvussa 5.

4 Mallit

LDAP-standardi määrittelee neljä niin sanottua mallia (engl. *model*). Nämä mallit ohjaavat hakemiston käyttöä. Kun hakemistot toteutetaan mallien mukaisesti, voidaan erilaisia hakemistototeutuksia käsitellä yhdenmukaisesti. Joitakin harvemmin käytössä olevia toiminnallisuuksia on LDAP-malleista jätetty pois, mutta muuten ne ovat enimmäkseen samat kuin X.500 -standardissa.

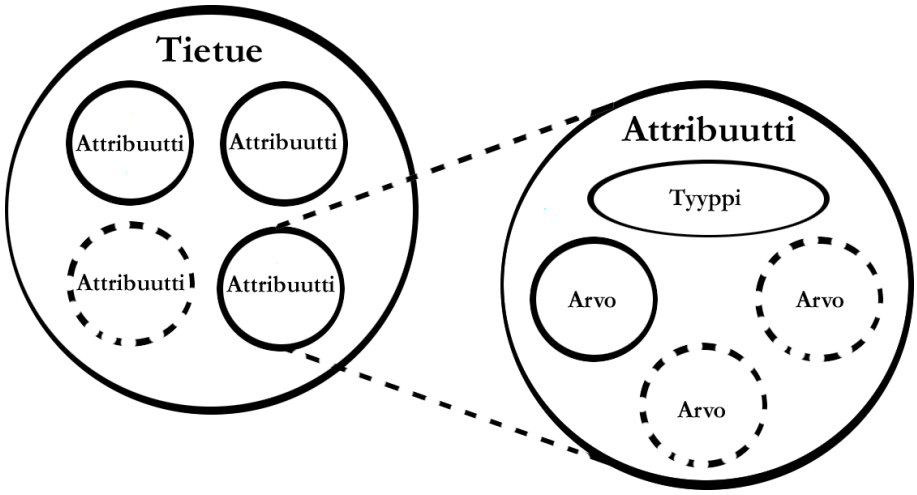
Mallit ovat IBM:n Redbookin [7, luku 2] mukaan seuraavat:

- **Tietomalli** määrittelee, minkälaista tietoa hakemistoon voi tallentaa.
- **Nimeämismalli** määrittelee, kuinka tietueet nimetään ja organisoidaan hakemistossa.
- **Toimintamalli** määrittelee, kuinka hakemistossa oleviin tietoihin voidaan tehdä hakuja ja kuinka sen tietoja ylläpidetään.
- **Tietoturvamalli** määrittelee, kuinka hakemisto voidaan suojata luvattomalta käytöltä.

Luvuissa 4.1- 4.6 edellä mainitut mallit käsitellään tarkemmin. Luku perustuu pääasiassa lähteisiin [7, luku 2] ja [11].

4.1 Tietomalli

Tietomalli määrittelee hakemistoon tallennettavien attribuuttien tyypit sekä attribuuteista koostuvat tietueet. **Tietueet** edustavat reaalimaailman kohteita, kuten ihmisiä, palvelimia ja organisaatioita. Tietue rakentuu attribuuttien kokoelmasta, joka pitää sisällään tietoja kohteista. Jokaisella **attribuutilla** on tyyppi, ja se voi saada yhden tai useamman arvon. Attribuutin tyyppi on yhdistetty syntaksiin kuvaten, minkä muotoista tietoa attribuutti voi saada arvokseen. Syntaksi määrittää myös sen, kuinka attribuutin arvoja vertaillaan, kun hakemistoon suoritetaan hakuja [7, s. 32]. Suhde hakemistotietueen, sen attribuuttien ja niiden arvojen välillä osoitetaan kuvassa 4.1.



Kuva 4.1: Tietueet, attribuutit ja arvot.

Jokaisella hakemistoon määritellyllä tietueella on `objectclass`-erikoisattribuutti. Sen arvo on lista kahdesta tai useammasta niin sanotusta skeemanimestä. Attribuutin `objectclass` määrittely voisi näyttää seuraavalta:

```
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson.
```

Skeemoja käyttämällä määritetään, mitä kohdetyyppiä tietueet edustavat. Tarkemmin LDAP-skeemaa käsitellään luvussa 4.7. Attribuutti `objectclass` määrittää tietueen olioluokan. Olioluokka puolestaan määrittää, mitä attribuutteja tietue voi sisältää sekä mitkä niistä ovat pakollisia. Olioluokka on aina periytetty yhdestä tai useammasta olioluokasta, poikkeuksena ainoastaan kantaluokka, josta käytetään LDAPin yhteydessä nimeä `top` [7, s. 35].

Attribuutit voidaan luokitella joko käyttäjäattribuutteihin tai toiminnallisiin attribuutteihin. **Käyttäjäattribuutit** ovat tavallisia attribuutteja, joiden arvoa voi käyttäjä muuttaa. Esimerkki käyttäjäattribuutista on `cn` (`common name`), jota voidaan käyttää esimerkiksi käyttäjän nimen määrittelyyn. **Toiminnalliset attribuutit** on tarkoitettu ennalta määrättyihin tarkoituksiin, ja ne ohjaavat hakemiston toimintaa tai kuvaavat sen tilaa. Esimerkki toiminnallisesta attribuutista on `createTimeStamp`, joka kertoo tietueen lisäysajan hakemistoon [13, s. 76-78].

4.2 Nimeämismalli

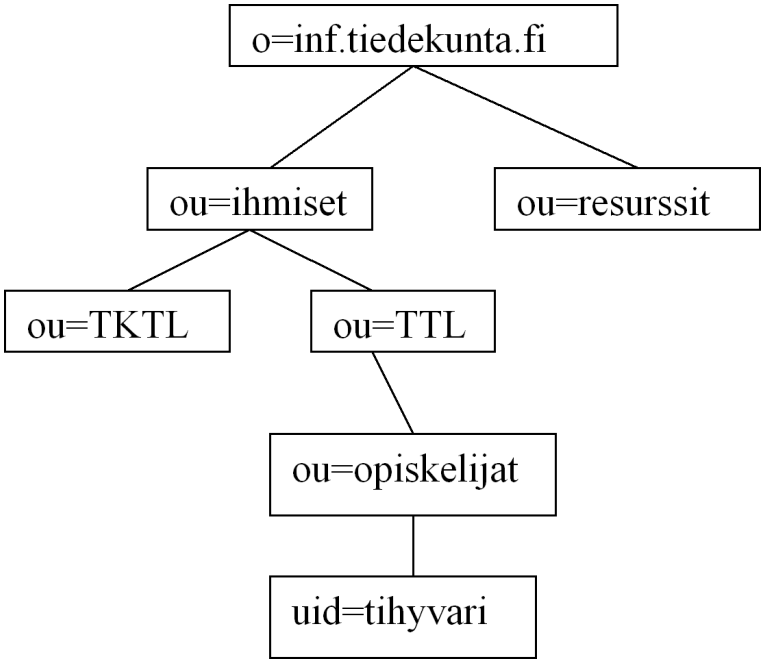
Nimeämismallin mukaan tietueet järjestetään hakemistoon puumaiseen rakenteeseen, jota kutsutaan lyhenteellä DIT eli Directory Information Tree. Tietueiden järjestys hakemistopuussa perustuu niiden yksikäsitteisiin nimiin. Tietueen **yksikäsitteinen nimi** eli DN (engl. *distinguished name*) muodostetaan yhdisteenä tietueen nimestä ja tätä hakemistopuussa edeltävien tietueiden nimistä.

Yksikäsitteinen nimi koostuu siis useasta paikallisesti yksikäsitteisestä nimestä, RDN (engl. *relative distinguished name*) [7, s. 42]. Jokainen RDN vastaa solmua hakemistopuussa alkaen juurisolmusta ja päätyen hakemistotietueeseen. Yksinkertaisimmassa ja yleisimmässä tapauksessa RDN on muotoa

```
<attribuutin nimi>=<arvo>.
```

Esimerkki hakemistopuusta esitetään kuvassa 4.2. Kuvan alimmainen hakemistotietue saa yksikäsitteiseksi nimeksi

```
uid=tihyvari, ou=opiskelijat, ou=TTL, ou=ihmiset,
o=inf.tiedekunta.fi.
```



Kuva 4.2: Esimerkki hakemistopuusta (DIT).

Ylläolevassa hakemistopuussa määritellyn opiskelijan tiedot voisi tietuemäärittelynä näyttää seuraavalta:

```
dn: uid:tihyvari, ou=opiskelijat, ou=TTL, ou=ihmiset,
o=inf.tiedekunta.fi
cn=Tiina Kuisma
sn=Kuisma
givenname: Tiina
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: intOrgPerson
ou: opiskelijat
uid: tihyvari
mail: tihyvari@cc.jyu.fi
```

Hakemistotietueiden nimillä on muutamia ominaisuuksia. Nimet muun muassa esitetään joko merkkijonona tai URL-muodossa. Ominaisuuksiin kuuluu myös, että nimillä on yhtenäinen syntaksi, eivätkä nimiavaruuden rajat ilmene nimistä. Nimien tarkka syntaksi määritellään standardissa RFC 2253 [9]. Poikkeuksen puumaiseen hakemistoon tekevät niin sanotut aliastietueet. Tällaiset tietueet ovat viittauksia toiseen tietueeseen, jotka sijaitsevat joko saman tai eri palvelimen hakemistopuussa [7, s. 35].

4.3 Toimintamallin autentikointioperaatiot

Toimintamalli kuvaa LDAP-hakemistoon suoritettavat operaatiot. Mallin sisältämät toiminnot voidaan jakaa autentikointi-, kysely- ja päivitysoperaatioihin.

Autentikointioperaatioita ovat `bind`, `unbind` ja `abandon`. `bind`-operaatiolla asiakas suorittaa tunnistautumisen ja yhteyden avaamisen LDAP-palvelimelle. Se ei tarkoita samaa kuin yhteydenottaminen palvelimelle, vaan lähinnä autentikoitumista serverille. Kaikki käyttöoikeudet LDAP-tietoon perustuvat siihen, miten yhteys on avattu. Yksinkertainen yhteys lähettää käyttäjän DN:n ja salasanan selkokielisenä. Yhteys kuitenkin suositellaan suojattavan esimerkiksi käyttäen TLS:ä (Transport Layer Security) [11].

Yhteyden täytyy täsmätä annettuun tietueeseen palvelimella ja useimmissa tapauksissa tämä tietue on henkilökohtainen tietue. LDAP-palvelin vertaa annettua

salasanaa `userPassword`-attribuuttiin, joka löytyy annetusta tietueesta. Jos yksikäsitteinen nimi ja salasana jätetään tyhjiksi, niin yhteydestä tulee ns. anonyymi yhteys, eli yhteyttä ei muodosteta nimettynä käyttäjänä [7, s. 54]. Tämän tyyppistä yhteyttä käsitellään lisää luvussa 4.6.

Yhteyden avaaminen ja tunnistautuminen LDAP-palvelimelle Java-kieltä käyttäen voisi näyttää esimerkiksi seuraavalta:

```
LDAPConnection ldap=new LDAPConnection();
ldap.connect("ldap.somewhere.com",389);
ldap.authenticate(uid=tihyvari, ou=opiskelijat,
o=inf.tiedekunta.fi, "sprain");
```

Ensimmäisellä rivillä luodaan viite `LDAPConnection`-objektiin, jonka jälkeen voidaan vasta avata varsinainen yhteys `connect`-metodilla. Lopuksi autentikoidutaan käyttäjänä `tihyvari`.

`unbind`-operaatiolla asiakas luopuu tärkeistä operaatioista ja sulkee yhteyden. Se ei kuitenkaan ole vastakohta `bind`-operaatiolle, sillä se tekee vain operaatioista luopumisen ja yhteyden sulkemisen. `abandon`-operaatiolla asiakas voi pyytää palvelinta keskeyttämään käynnissä olevan operaation. Tätä voi hyödyntää esimerkiksi tilanteessa, jossa asiakas ei ole enää kiinnostunut tekemänsä operaation vastauksesta [11]. Javalla yhteyden sulkeminen tapahtuu `disconnect`-metodilla seuraavasti:

```
ldap.disconnect();
```

4.4 Toimintamallin kyselyoperaatiot

Kyselyoperaatioita ovat `search` ja `compare`. `search` on kaikkein yleisin operaatio. Sitä käyttämällä asiakas pyytää LDAP-palvelinta etsimään hakemistopuusta hakukriteeriensä mukaista tietoa tietojen lukua ja listausta varten. Luvulle ja listaukselle ei ole erikseen operaatioita, vaan ne sisältyvät `search`-operaatioon.

`compare`-operaatiolla sen sijaan voidaan hakemistopalvelimelta kysyä, onko jollakin attribuutilla tiettyä arvoa. Operaatio palauttaa arvon `TRUE` tai `FALSE` riippuen siitä, löytyykö tietueesta kyseistä arvoa. Sen sijaan `search`-operaatio palauttaisi vastauksen `not found`, jos haettavaa attribuuttia ei löydy. `compare`-operaation vastaus `FALSE` osoittaisi sen, että tietue kuitenkin on olemassa, mutta sillä vain ei ole kyseiseen attribuuttiin täsmäävää arvoa [7, s. 51].

`search`-operaatiolle annetaan kahdeksan erilaista parametriä. Nämä parametrit ovat Wikipedian määritelmän [11] mukaan seuraavat:

`baseObject` määrittää haun lähtökohdan yksikäsitteisen tietueen nimellä ilmoittaen, mistä hakemistopuun kohdasta haku aloitetaan.

`scope` määrittää haun laajuuden antamalla haun vaikutusalueen hakemistopuussa. Vaihtoehdolla `baseObject` haetaan vain nimettyä tietuetta, `singleLevel` hakee tietueista heti alapuolella olevista ja `wholeSubtree` määrittää haun kohteeksi koko puun alkaen DN:stä, joka on määritelty aloitustietueeksi.

`filter` määrittää hakusuodattimen ilmoittaen, minkälaiset tietueet palautetaan.

`deferAliases` määrittää aliaksen uudelleenohjauksen kertoen, seurataanko ja miten seurataan mahdollisia aliastietueita.

`attributes` määrittää attribuuttilistana, mitkä attribuutit palautetaan tulostietueissa.

`sizeLimit` määrittää kokorajoituksena palautettavien tietueiden määrän.

`timeLimit` määrittää aikarajoituksena ajan, joka hakuun voidaan käyttää.

`typesOnly` on tyyppirajoitus, jolla voidaan määrätä palautettavan vain attribuuttien tyypit, ei arvoja.

`search`-operaatio Java-kieltä käyttäen voisi esimerkiksi näyttää seuraavalta:

```
...
//Alustetaan tarvittavat muuttujat parametrimäärittelyyn.
String base = "o=inf.tiedekunta.fi";
int scope = LDAPConnection.SCOPE_SUB;
String filter = "(cn=Tiina Kuisma)";
//Muodostetaan lista, johon attribuutit otetaan talteen.
String attrs [] = {"cn","uid","mail"};
//Suoritetaan haku antaen parametrinä myös attrs-lista.
//Viimeisellä parametrilla määritellään, että palautetaan
   sekä attribuutit että niiden arvot.
```

```
LDAPSearchResults res = ld.search(base, scope, filter,
   attrs, false);
...
```

Esimerkin koodi ei toimi suoraan esitellyllä tavalla. Siinä on esitelty vain joitakin tarvittavia muuttujamäärittelyjä, joita haun suorittamiseen tarvittaisiin. Esimerkissä on käytetty hakulähteenä luvussa 4.2 esiteltävää hakemistopuuta ja sen tietuemäärittelyä. Haku palauttaisi tuloksena tiedot seuraavasti:

```
DN: uid=tihyvari, ou=opiskelijat, ou=TTL, ou=ihmiset,
o=inf.tiedekunta.fi
cn: Tiina Kuisma
uid: tihyvari
mail: tihyvari@cc.jyu.fi.
```

4.5 Toimintamallin päivitysoperaatiot

Päivitysoperaatioita ovat `add`, `delete`, `modify` ja `modifyRDN`. Päivitysoperaatiot ovat atomisia, toisin sanoen ne joko suoritetaan onnistuneesti kokonaan tai niitä ei suoriteta ollenkaan.

`add`-operaatiolla voidaan lisätä uusia tietueita hakemistoon. `add`-operaatiolle annetaan parametreina lisättävän tietueen yksikäsitteinen nimi sekä siihen liitettävät attribuutit. Liitteessä A on esimerkki tietueen lisäyksestä käyttäen Java-kieltä.

`delete`-operaatiolla voidaan poistaa olemassa oleva tietue hakemistosta. Vain lehtisolmuja voidaan poistaa eli poistettavalla solmulla ei saa olla lapsisolmuja. Liitteen A lisäystä vastaava esimerkki tietueen poistosta on nähtävissä liitteessä B.

`modify`-operaatiota voidaan käyttää attribuuttien ja arvojen muuttamiseen kyseisessä tietueessa. Sitä käyttämällä voidaan lisätä uusia attribuutteja sekä poistaa tai muuttaa olemassa olevia. `Modify RDN` -operaatiolla voidaan tietue siirtää toiseen paikkaan hakemistopuussa. Jos tietueella on mahdollinen alipuu, niin se siirtyy myös siirron yhteydessä. Wikipedian [11] mukaan tietueita ei voi kuitenkaan siirtää palvelinrajojen yli.

4.6 Tietoturvamalli

Tietoturvamalli perustuu `bind`-operaatioon. LDAP on yhteyspohjainen protokolla (engl. *connection-oriented*). Siinä siis asiakas avaa yhteyden autentikoitumalla palvelimelle ja suorittaa haluamansa operaatiot yhteyden aikana.

Tietoturvamallin mukaan on olemassa useampi tapa käyttää `bind`-operaatiota yhteyden avaamiseen. Yksi mahdollisuus on, että asiakas autentikoi itsensä antamalla yksikäsitteisen nimen sekä salasanan. Esimerkki tästä löytyy luvusta 4.3. Jos **yksikäsitteinen nimi** ja **salasana** täsmäävät hakemistossa oleviin tietoihin, palvelin toteaa autentikoinnin onnistuneen.

Toisessa tavassa tunnistetietoja ei anneta, jolloin LDAP-palvelin olettaa yhteyden syntyneen anonyyminä. Tällöin asiakkaalla on usein hyvin vähän, tai ei ollenkaan oikeuksia. **Anonyymejä autentikointeja** varten määritellään erikseen käyttöoikeustasot [7, s. 54]. Autentikoituminen ilman yksikäsitteistä nimeä ja salasanaa näyttäisi seuraavalta:

```
ldap.authenticate("","");
```

LDAPv3:ssa lisämahdollisuutena on tullut ns. **SASL-tunnistus** (Simple Authentication and Security Layer). SASL on yleinen autentikointimalli, jossa on käytettävissä useita erilaisia autentikointimetodeja. Tällainen on esimerkiksi Kerberos. SASL:ssa yhteysprotokollat, kuten LDAP tai IMAP, kuvataan profiileilla. Jokainen profiili käsitetään protokollan laajennukseksi, joka sallii protokollan ja SASL:n yhteistoiminnan [7, s. 55].

4.7 Skeema

Tietokantojen yhteydessä käytetään skeemoja niiden rakenteen määrittelyyn. Myös LDAPissa skeemoja käytetään tähän tarkoitukseen. LDAPissa skeema on kokoelma käytetyistä olioluokka- ja attribuuttimäärityksistä, jonka avulla määritellään LDAP-hakemistoon tallennettavia olioita ja attribuutteja. Se kuvaa olioluokkien periytymisjärjestyksen sekä attribuuttien syntaksit. Esimerkki LDAPin skeemamäärittelystä Javan objekteille on nähtävissä RFC 2713:ssa [4].

Kun hakemistoon lisätään uusi tai muutetaan olemassa olevaa tietuetta, niin palvelin tarkistaa ensin hakemiston skeemasta, minkä tyyppisestä tietueesta kulloinkin on kyse. Jos tietue on skeeman mukainen, suorittaa palvelin halutun operaation. Muussa tapauksessa palvelin palauttaa virheilmoituksen yritetystä operaatiosta. Skeemaa käyttämällä pystytään IBM:n Redbookin [7, s. 37-38] mukaan vähentä-

mään duplikaattien määrää sekä ylläpitämään tallennettavan tiedon johdonmukaisuutta ja laatua.

LDAPissa **olioluokat** voidaan jakaa IBM:n Redbookin [7, s. 37] mukaan abstrakteihin, rakenteellisiin tai avustaviin luokkiin. Abstraktia luokkaa käytetään niin sanottuna mallina (engl. *template*), kun luodaan muita luokkia. Hakemistotietuetta ei voida toteuttaa abstraktista luokasta, vaan siihen tarkoitukseen tarvitaan rakenteellinen luokka. Avustavaa luokkaa voidaan käyttää rakenteellisen luokan laajennukseen. Laajennuksena käyttö mahdollistaa sen, ettei rakenteellisen luokan skeemamäärittelyä tarvitse itsessään muuttaa. Avustavasta luokasta ei voida toteuttaa kuitenkaan tietuetta yksinään, vaan tällaisen tietueen täytyy olla liitettynä rakenteellisesta olioluokasta toteutettuun hakemistotietueeseen.

5 Ohjelmointirajapinnat

API (Application Programming Interface) eli ohjelmointirajapinta on standardi rajapinta, jota ohjelmoijat käyttävät ohjelmoidessaan sovelluksia. APIn käyttö helpottaa ohjelmoijia, sillä heidän ei tarvitse kiinnittää huomiota LDAPin varsinaiseen toteutukseen. Luku perustuu pääasiassa lähteisiin [6, s. 26-96] ja [13, s. 59-66].

5.1 LDAPin APIt

Michiganin yliopisto kehitti ensimmäisen LDAP-APIn. Siinä käytettiin kielenä C:tä, ja se oli alunperin kehitetty Unix/Linux-alustalle [13, s. 59]. Kyseinen API on nykyäänkin laajasti käytössä, ja sitä on laajennettu koskemaan DOS-, Windows- sekä Macintosh-alustoja.

Netscape on julkaissut sittemmin omat SDK:t (Software Development Kit) C:lle, Javalle ja Perlille. SDK on kirjastoista ja työkaluista koostuva kokonaisuus, jonka avulla voidaan kehittää sovelluksia tietylle alustalle. Netscapen julkaisut on toteutettu avoimen lähdekoodin periaatteella, joten ne ovat kaikkien vapaasti saatavilla.

Oman lisänsä ohjelmointirajapintojen joukkoon tuovat Wilcoxin [13, s. 59-66] mukaan Sunin JNDI (Java Naming and Directory Interfaces) sekä Microsoftin ADSI (Active Directory Service Interfaces). Ne mahdollistavat ohjelmoijan yhteydenoton erilaisiin verkko- ja hakemistopalveluihin.

5.2 LDAPin ydin-APIt

Thompsonin määritelmän [6, s. 26] mukaan LDAPia koskien on kolme niin sanottua ydin-APIa. Nämä APIt koskevat operaatioluokittelua eli autentikointia, kyselyä ja päivitystä.

Autentikointi-APIa käytetään, kun asiakas ottaa yhteyden LDAP-palvelimeen. Autentikointi-APIn yhteydessä käytetään seuraavia kutsuja:

`ldap_open_s` luo ja alustaa yhteydenmuodostuksen, jonka jälkeen se vasta varsinaisesti avaa yhteyden LDAP-palvelimeen.

`ldap_bind_s` autentikoi asiakkaan LDAP-palvelimelle edellisen kutsun jälkeen.

ldap_unbind_s päättää yhteyden. Sillä vapautetaan kaikki LDAP-yhteyden käytössä olleet resurssit, sekä sitä voidan käyttää myös yhteyden käsittelyn keskeyttämiseen.

Kysely-APIa käyttäen voidaan kysellä tietoja, jotka sijaitsevat DITissä eli hakemiston puumaisessa rakenteessa. Niitä voidaan kutsua useammalla tavalla. Niitä voi kutsua esimerkiksi käyttäjä käytössä olevan käyttöliittymän avulla tai ADSI-sovelluksella suoraan. `Ldap_search_s`-kutsu on esimerkki kysely-APIin liittyvästä kutsusta.

Päivitys-APIt mahdollistavat sen, että asiakas voi muuttaa DITissä olevia tietoja. Rajapinnan avulla asiakas voi lisätä uusia tietueita tai muokata ja poistaa olemassa olevia. Esimerkkinä päivityskutsusta on `Ldap_add_s`, jonka avulla lisätään uusi tietue DITiin.

6 Yhteenveto

Käyttäjähallinta luo monelle yritykselle ja organisaatiolle haasteita. Sovellusten ja sen myötä hakemistojen määrä on monessa organisaatiossa kasvanut niin, että se aiheuttaa konkreettisia ongelmia organisaation käyttäjähallinnalle ja ylläpidolle. Suureksi ongelmaksi muodostuu helposti yksinkertaisesti käyttäjätunnusten ja -oikeuksien ylläpitäminen. Näiden ongelmien ratkaisuun on kuitenkin kehitetty erilaisia apukeinoja ja kehitystyö jatkuu yhä.

LDAP on eräs apukeino keskitettyyn käyttäjähallintaan. LDAP on laajasti levinnyt ja selkeästi tällä hetkellä yleisin käytössä oleva hakemistopalvelu. Sen suosiota kasvattaa sen helppous ja yksinkertaisuus ainakin verrattuna edeltäjäänsä X.500.

LDAP-standardi määrittelee neljä niin sanottua mallia, jotka ohjaavat hakemiston käyttöä. Mallit ovat tietomalli, nimeämismalli, toimintamalli ja tietoturvamalli. Näistä toimintamallissa määritellään, mitä operaatioita LDAPin yhteydessä on käytettävissä. Operaatiot voidaan luokitella autentikointi-, kysely- ja päivitysoperaatioihin. Autentikointioperaatiot koskevat yhteyden avausta, tunnistautumista ja yhteyden sulkemista LDAP-palvelimelle. Kyselyoperaatioilla voidaan suorittaa esimerkiksi erilaisia kyselyjä hakemistopalvelimelle sekä päivitysoperaatioilla mahdollisia lisäys-, muokkaus- ja poistotoimintoja.

LDAPin yleisyyttä ja käyttölaajuutta tukevat usealle ohjelmointikielelle tarjolla olevat ohjelmointirajapinnat. Niitä löytyy muun muassa C:lle, Javalle ja Perlille.

Lähteet

[1] OASIS, *OASIS Security Services (SAML) TC*, saatavilla HTML-muodossa <URL: http://www.oasis-open.org/committees/security/>, OASIS Open, 2006.

[2] Pfitzmann B. and Waidner M., *Federated Identity-Management Protocols –Where User Authentication Protocols May Go*, Springer-Verlag, Berlin-Heidelberg, 2004.

[3] Rosencrance L., *SAML Secures Web Services*, saatavilla HTML-muodossa <URL: http://www.computerworld.com/developmenttopics/development/webdev/story/0,10801,73712,00.html>, Computerworld Inc., 2006.

[4] Ryan V., Seligman S. and Lee R., *RFC 2713 Schema for Representing Java(tm) Objects in an LDAP Directory*, saatavilla txt-muodossa <URL: http://www.ietf.org/rfc/rfc2713.txt>, The Internet Society, 1999.

[5] Spencer L.C., *An Introduction to Identity Management*, SANS Security Essentials Certification Practical Assignment version 1.4b option 1, SANS Institute, 2003.

[6] Thompson D., *Understanding LDAP*, saatavilla doc-muodossa <URL: http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/ldap.asp>, Microsoft Corporation, 2004.

[7] Tuttle S., Ehlenberger A., Gorthi R., Leiserson J., Macbeth R., Owen N., Ranahandola S., Storrs M. and Yang C., *Understanding LDAP Design and Implementation*, saatavilla pdf-muodossa <URL: http://www.redbooks.ibm.com/redbooks/pdfs/sg244986.pdf>, International Technical Support Organization, IBM Redbook, 2004.

[8] Wahl M., Howes T. and Kille S., *RFC 2251 Lightweight Directory Access Protocol (v3)*, saatavilla txt-muodossa <URL: http://www.ietf.org/rfc/rfc2251.txt>, The Internet Society, 1997.

[9] Wahl M., Howes T. and Kille S., *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names)*, saatavilla txt-muodossa <URL: http://www.ietf.org/rfc/rfc2253.txt>, The Internet Society, 1997.

[10] Wikipedia, *Identity Management*, saatavilla HTML-muodossa <URL: http://en.wikipedia.org/wiki/Identity_management/>, viitattu 26.6.2006.

[11] Wikipedia, *Lightweight Directory Access Protocol*, saatavilla HTML-muodossa <URL: http://en.wikipedia.org/wiki/LDAP>, viitattu 14.8.2006.

[12] Wikipedia, *Shibboleth (Internet2)*, saatavilla HTML-muodossa <URL: http://en.wikipedia.org/wiki/Shibboleth_(Internet2)>, viitattu 2.10.2006.

[13] Wilcox M., *Implementing LDAP*, Wrox Press, 1999.

A Tietueen lisäysluokka

```
import netscape.ldap.*;
import java.io.*;
import java.util.*;

public class add{
    public static void main(String args[]){

        /* Define variables for the server and binding. */
        String host = "localhost";
        int port = 389;
        String dn = "uid=meikama, ou=yllapitaja,
        o=inf.tiedekunta.fi";
        String pwd = "s1n1n3en";

        /* Define variables for adding new entry. */
        String new_dn = "uid=tihyvari, ou=opiskelijat,
        o=inf.tiedekunta.fi";

        String objectclass_values [] = {"top", "person",
        "organizationalperson", inetorgperson"};

        String cn_values [] = {"Tiina Kuisma"};
        String sn_values [] = {"Kuisma"};
        String givenname_values [] = {"Tiina"};
        String ou_values [] = {"ihmiset", "TTL", "opiskelijat"};
        String uid_values [] = {"tihyvari"};
        String mail_values [] = {"tihyvari@cc.jyu.fi"};

        /* Define the attributes first as null. */
        LDAPAttributeSet attribute_set = null;
        LDAPAttribute attribute = null;
        LDAPEntry entry = null;
```

```
LDAPConnection ld = null;

/* Try to make a connection. */
try{
ld = new LDAPConnection();

/* Must bind as a user with rights to write to
the server. */
ld.connect(host,port,dn,pwd);

/* Define the attributes. */
attribute_set = new LDAPAttributeSet();
attribute = new LDAPAttribute("objectclass",
objectclass_values);

attribute_set = add(attribute);
attribute = new LDAPAttribute("cn", cn_values);
attribute_set = add(attribute);
attribute = new LDAPAttribute("sn", sn_values);
attribute_set = add(attribute);
attribute = new LDAPAttribute("givenname",
givenname_values);
attribute_set = add(attribute);
attribute = new LDAPAttribute("mail", mail_values);
attribute_set = add(attribute);

attribute = new LDAPAttribute("ou", ou_values);
attribute_set = add(attribute);
attribute = new LDAPAttribute("uid", uid_values);
attribute_set = add(attribute);

/* Create the entry object. */
entry = new LDAPEntry(new_dn, attrib_set);

/* Add the object. */
ld.add(entry);
```

```
            if(ld != null){
                ld.disconnect();
                }
            }catch(LDAPRxception e){
                e.printStackTrace();
            }
        }
    }
```

B Tietueen poistoluokka

```
import netscape.ldap.*;
import java.io.*;
import java.util.*;

public class del{
    public static void main(String args[]){

        /* Define variables for the server and binding. */
        String host = "localhost";
        int port = 389;
        String dn = "uid=meikama, ou=yllapitaja,
        o=inf.tiedekunta.fi";
        String pwd = "s1n1n3en";

        String entry_dn = "uid=tihyvari, ou=opiskelijat,
        o=inf.tiedekunta.fi";

        /* Try to make a connection. */
        LDAPConnection ld = null;
        try{
        ld = new LDAPConnection();

        /* Must bind as a user with rights to write to
        the server. */
        ld.connect(host, port, dn, pwd);

        /* Delete the object. */
        ld.delete(entry_dn);
        if (ld != null) {
            ld.disconnect();
            }
```

```
        }catch(LDAPException e) {
            e.printStackTrace();
        }
    }
}
```