Alparslan Ünsal

AUTENTIKOINTI MOBIILISOVELLUKSESSA

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2006

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala	Koulutusohjelma
Tekniikan ja liikenteen ala	Tietotekniikka
Tekijä(t)	
Alparslan Ünsal	
Työn nimi	
Autentikointi mobiilisovelluksessa	
Vaihtoehtoiset ammattiopinnot	**Ohjaaja(t)**
Ohjelmistotekniikka	Raili Simanainen
	Toimeksiantaja
	KajaPro Oy
Aika	**Sivumäärä ja liitteet**
7.4.2006	43+1

Insinöörityössä tehtiin autentikointirajapinta mobiilisovelluksia varten. Autentikointirajapinta tulee osaksi KajaPro Oy:n MCF-arkkitehtuuria. KajaPro on kajaanilainen tietotekniikan yritys, joka on erikoistunut mobiililaiteohjelmointiin Javalla.

KajaPro Oy:llä on toteutettu asiakas/palvelin-ympäristö, jossa asiakaspään muodostaa älypuhelin. Autentikointirajapinta sisältää kaksi eri rajapintaa, joista toinen on tehty mobiilisovelluksen asiakaspuolta varten ja toinen palvelinpuolta varten. Asiakkaiden tunnistamista varten täytyi palvelinpäähän tehdä autentikointirajapinta, jossa tunnistukseen oli käytettävissä laitteen IMEI-koodi, puhelinliittymän IMSI-koodi, käyttäjätunnus/salasana-pari tai niiden yhdistelmä.

Asiakasrajapinnalla käytettiin J2ME-ohjelmointia ja palvelinrajapinnalla käytettiin J2SE-ohjelmointia, JAAS-rajapintaa ja MySQL-tietokantaohjelmaa. Rajapintojen tekemiseen ja testaamiseen käytettiin Sunin NetBeans -ohjelmaa. Tehtyjen rajapintojen testaamiseksi kirjoitettiin demosovellukset, jotka testasivat asiakas- ja palvelinrajapintoja. Insinöörityön tuloksena saatiin MCF-arkkitehtuurin autentikointirajapinnat ja valmis tietokantataulukko.

Kieli	Suomi
Asiasanat	Asiakas/palvelin, autentikointi, JAAS, monikerrosarkkitehtuurit
Säilytyspaikka	☒ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☒ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School	Degree Programme
School of Engineering	Information Technology
Author(s) Alparslan Ünsal	
Title Authentication in Mobile Applications	
Optional Professional Studies Software Engineering	Instructor(s) Raili Simanainen
	Commissioned by KajaPro Oy
Date 7 April 2006	Total Number of Pages and Appendices 43+1

The purpose of this Bachelor's thesis was to design and implement an authentication API for mobile applications. The authentication API will be part of KajaPro's MCF architecture. KajaPro is an IT company in Kajaani, which is specialized in programming mobile devices.

KajaPro has already implemented a client/server environment, where the client side is the mobile phone. The authentication API has two distinct APIs for the client and server sides. During the study, an authentication API for idenfying clients was implemented for the server side, which used for identification the mobile device's IMEI code, the subscriber's IMSI code, the userid/password pair or their combination.

J2ME programming was used during the client's implementation. J2SE programming, JAAS API and the MySQL database program were used for implementing the server. Sun's NetBeans program was used for programming and testing the APIs. Two simple demo applications were produced for testing the client and server APIs. At the end of the study there are two authentication APIs and a database table as a result.

Language of Thesis Finnish

Keywords Authentication, client/server, JAAS, n-tier architecture

Deposited at ☒ Kaktus Database at University of Applied Sciences Library
☒ Library of University of Applied Sciences

ALKUSANAT

Tämä insinöörityö on tehty Kajaanissa toimivalle KajaPro Oy:lle.

Haluan kiittää insinöörityön valvojaa lehtori Raili Simanaista Kajaanin ammattikorkeakoulusta ja kehityspäällikkö Petteri Pyrröä KajaPro Oy:stä. Lisäksi haluan kiittää kieliasun tarkastamisesta lehtori Eero Soinista, yliopettaja Kaisu Korhosta ja opiskelija Leena Impiöä.

Kajaanissa 7.4.2006

Alparslan Ünsal

SISÄLLYSLUETTELO

SYMBOLILUETTELO JA TERMIEN SELITYS

Termi	Selitys
ACL	Access Control List on tietorakenne, joka valvoo pääsyä johonkin resurssiin. Javan acl-paketti tarjoaa tällaisen rajapinnan ja toteutuksen. ACL:n avulla voidaan helposti määrittää käyttäjiä ja ryhmiä ja valvoa näiden pääsyä eri resursseihin.
API	Application Program Interface, sovellusrajapinta. API on joukko määrityksiä, protokollia ja työkaluja sovellusten kehittämiseen ja API tarjoaa yhdenmukaisen toteutustavan sovelluskehittäjille.
Haittaohjelma	Yleiskäsite tietokoneohjelmille, jotka tarkoituksellisesti aiheuttavat ei-toivottuja tapahtumia koneessa tai tietojärjestelmässä.
JDBC	Java Database Connectivity, rajapinta. Se mahdollistaa tietokantayhteyksiä Javassa.
Middleware	Väliohjelmisto. Tavallisesti sijoittuu 2-kerrosarkkitehtuurin kerrosten väliin ja toimii välittäjänä näiden välillä.
Phishing	Sähköiden tunnisteiden varastaminen. Huijaussähköpostit ja vilpilliset Web-sivut, jotka on suunniteltu huijaamaan vastaanottajia luovuttamaan henkilökohtaisia tietoja, kuten luottokorttien numeroita, tilien käyttäjätunnuksia ja salasanoja.
SETI@home	SETI@home:n tarkoituksena on valjastaa Internetiin kytketyt koti-PC:t analysoimaan Arecibon radioteleskoopilla tallennettua dataa ja etsimään siitä mahdollisen elämän merkkejä.

1 JOHDANTO

Insinöörityön tavoitteena oli suunnitella ja tehdä autentikointirajapinta KajaPro Oy:n käyttöön. KajaPro on kajaanilainen tietotekniikan yritys, joka on erikoistunut mobiililaiteohjelmointiin Javalla. KajaPro Oy:llä on toteutettu asiakas/palvelin-ympäristö, jossa asiakaspään muodostaa älypuhelin. Insinöörityön tarkoituksena oli tehdä autentikointirajapinta järjestelmän asiakas- ja palvelinpuolelle. Autentikointirajapinta tuli osaksi KajaPro Oy:n MCF-arkkitehtuuria.

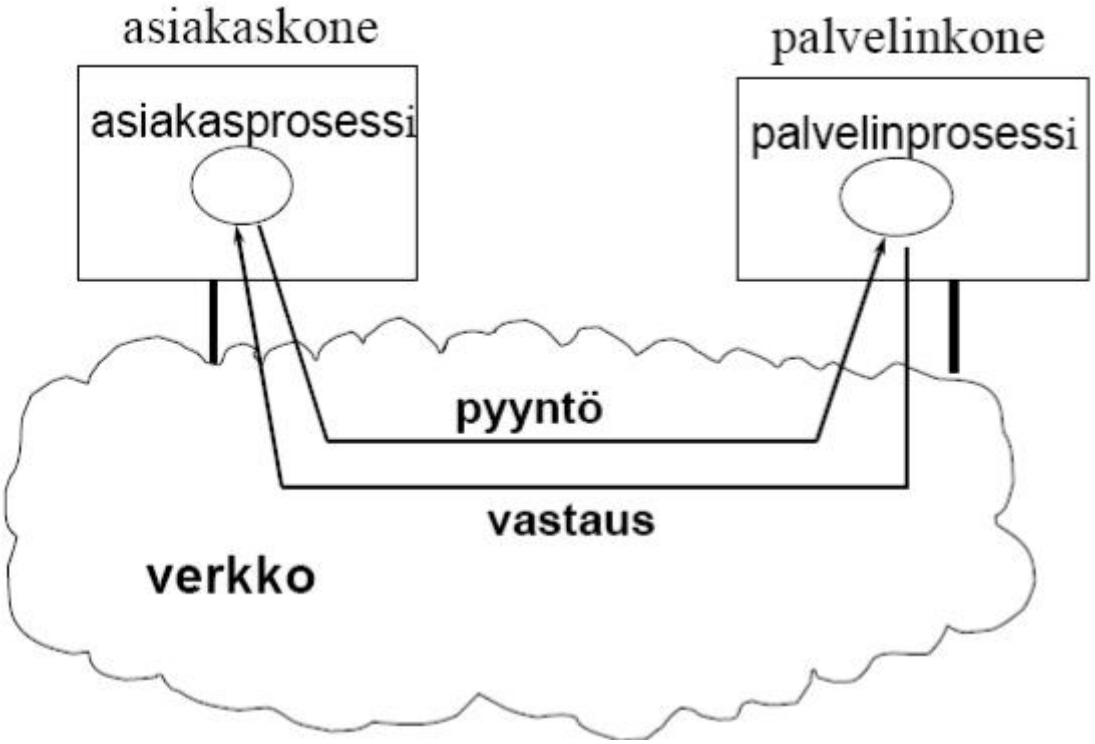
KajaPro Oy:llä ei ole olemassa entuudestaan autentikointitoteutusta. Erityisesti käyttäjän tunnistaminen oli tarpeen järjestelmissä, joissa eri käyttäjillä oli erilaisia oikeuksia. Insinöörityön kohteena oli älypuhelinsovelluksen tunnistaminen palvelin-sovelluksessa. Tunnistusta varten täytyi palvelinpäässä tehdä autentikointi-rajapinta, jossa tunnistukseen oli käytettävissä laitteen IMEI-koodi, puhelinliittymän IMSI-koodi, käyttäjätunnus/salasana-pari tai niiden yhdistelmä.

Insinöörityön alussa kuvataan asiakas-palvelin-arkkitehtuuria ja autentikointia. Sen jälkeen tulevat autentikointirajapintojen toteutus- ja testausosat.

2 SOVELLUSARKKITEHTUURIT

2.1 Asiakas/palvelin-arkkitehtuuri

Kaksi keskenään kommunikoivaa tietokonetta eivät useinkaan ole samanlaisessa asemassa toisiinsa nähden. Toinen tietokone pitää komentoa ja pyytää toiselta tietokoneelta palvelua, esimerkiksi lähettämään tiedoston tai sähköpostiviestin. Tietokonetta, joka suorittaa palveluita, kutsutaan palvelimeksi. Palveluita pyytävää konetta kutsutaan puolestaan asiakkaaksi. Tätä periaatetta kutsutaan asiakas-palvelin-malliksi. [1.] Kuva 1 havainnollistaa asiakas/palvelin-arkkitehtuuria.



Kuva 1. Asiakas/palvelin-arkkitehtuuri [2]

Tavallisen Internet-käyttäjän tietokone toimii aina asiakkaana riippumatta siitä, käytetäänkö tietokonetta sähköpostiviestien lähettämiseen tai vastaanottamiseen, tiedostojen siirtoon tai tietojen hakuun World Wide Webistä. Palvelimena toimiva tietokone ei juuri koskaan ole tavallinen työasema, vaan tehokkaampi tietokone,

joka on tehty palvelemaan useaa samanaikaista käyttäjää. Intranetissä palvelin voi olla toimiston tavallinen verkkopalvelin. [1.]

Teknisesti on tietyissä olosuhteissa mahdollista, että mikä tahansa lähiverkossa oleva tietokone voi toimia palvelimena. Näin ei kuitenkaan yleensä ole asian laita, vaan muun muassa turvallisuus- ja suorituskykysyistä palvelimena toimii tätä tarkoitusta varten oleva tietokone. [1.]

Tietojen siirron suunta ei ratkaise sitä, kumpi kone toimii palvelimena ja kumpi asiakkaana. Asian ratkaisee se, kummalla koneella on aloite siirrossa. Esimerkiksi tiedostosiirrossa yhteyden ottava kone on asiakas, riippumatta siitä, lähettääkö se vai vastaanottaako se tietoa. [1.]

Tiedonsiirron loputtua avattu yhteys kahden koneen välillä puretaan. Vaikka verkossa olisi useampi kone, tiedon siirto tapahtuu aina kahden koneen välillä. Kyseessä on siis aina asiakkaan ja palvelimen välinen tiedonsiirto. [3.]

Pystyäkseen toimimaan tietyn palvelun palvelimena tietokoneelle täytyy asentaa tätä palvelua varten palvelinohjelmisto. Palvelinohjelmisto itsessään voi toimia palvelimena, vaikka laite, jossa se toimii, on sekä palvelin että asiakas. [1.]

2.1.1 ”Thin”- ja ”Thick”-asiakas

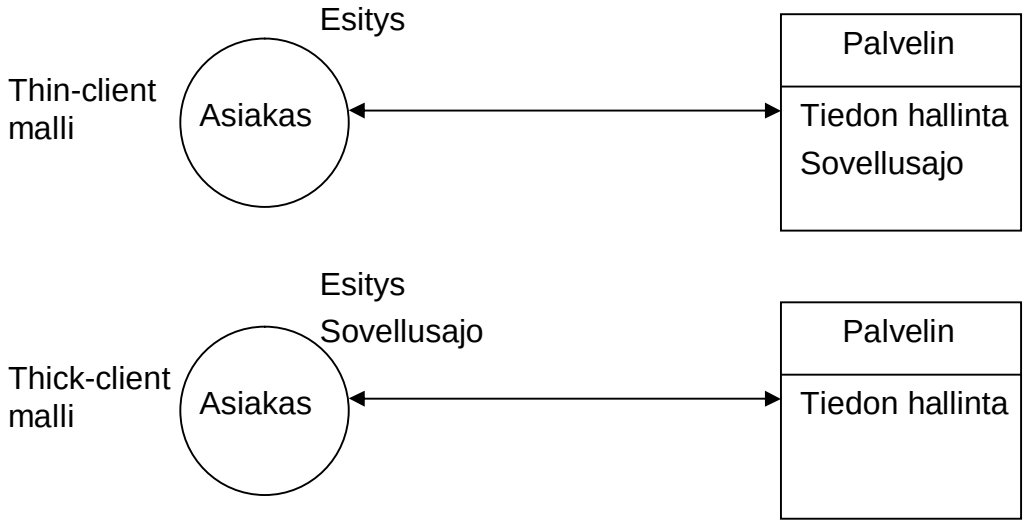
Asiakas/palvelin-sovelluksen suunnittelussa pitää päättää, mitkä sovelluslogiikan osat sijaitsevat asiakaskoneessa ja mitkä palvelinkoneessa. Tämä päätös on ratkaiseva tekijä asiakas- ja palvelinsovelluksien hinnoissa, kuluissa, suorituskyvyissä, tietoturvassa ja joustavuudessa. Päätöksen avulla voidaan kertoa, onko kyseessä ”Thin”- vai ”Thick”-asiakas. [4.]

Ohjelmistomäärittelynä ”Thin”-asiakas on asiakas/palvelin-arkkitehtuurissa oleva asiakaskone, jolla ei ole lainkaan tai on hyvin vähän sovelluslogiikkaa. Laitteistomäärittelynä ”Thin”-asiakas on massamuistiton laite, jolla on rajoittunut laskentateho ja muistimäärä. [5.] Kuva 2 havainnollistaa ”Thin”-asiakas tietokonetta.

Kuva 2. Moderni ”Thin”-asiakas tietokoneen ulkonäkö [4]

”Thin”-asiakkaan koneella on sen verran toiminnallisuutta, että asiakas pystyy aloittamaan verkkoyhteyden tai Web-selain. ”Thin”-asiakkaan koneella on yleensä käyttöliittymäsovellus, ja sen avulla annetaan palvelimelle pyyntöjä. [4.]

Ohjelmistomäärittelyn mukaan ”Thick”-asiakkaalla on käytössä runsaasti sovelluslogiikkaa [5]. Laitteistomäärittelynä ”Thick”-asiakas on suorituskykyinen tietokone tai laite [5]. ”Thick”-asiakas tekee kaikki mahdolliset tehtävät itse ja lähettää palvelimelle vain tiedonvälitystä varten tarvittavia tietoja [4]. ”Thin”- ja ”Thick”-asiakkaiden ero nähdään alla olevasta kuvasta 3.



Kuva 3. ”Thin”- ja ”Thick”-asiakasmallit [6, s.246]

”Thick”-asiakkaiden käytöstä esimerkkinä voidaan esittää hajautettujen järjestelmien projekteja, kuten SETI@home-projekti. SETI@home-projektin asiakkaat pyytävät palvelimelta tietoa, suorittavat pyydetyt laskut ja lähettävät saavutetut ratkaisut palvelimelle. ”Thin”-asiakkaita käytetään tilanteissa, joissa esitellään kaikille esimerkiksi samaa opetusmateriaalia. [4.]

2.1.2 ”Thin”- ja ”Thick”-asiakkaiden edut

”Thin”-asiakkaiden käytössä IT-asiantuntijoiden kulut vähenevät, koska lähes kaikki asiakkaiden tehtävät suoritetaan palvelimessa. Tietokoneiden laitteistolla on vähemmän todennäköisyyttä epäonnistua, koska paikallinen ympäristö on hyvin rajoitettu ja se tarjoaa suojausta haittaohjelmia vastaan. [4.]

”Thin”-asiakkaiden laitteisto on yleensä halvempi, koska asiakkailla ei ole kiintolevyä, sovellusmuistia tai tehokasta suoritinta. ”Thin”-asiakkaiden laitteisto vanhenee hitaammin kuin ”Thick”-asiakkaan laitteistot. ”Thin”-asiakkaiden laitteisto ja ohjelmisto eivät ole arvokkaita varkaille. [4.]

”Thick”-asiakkaat tarvitsevat yleensä vähemmän kaistanleveyttä, koska ne suorittavat lähes kaikki tehtävät itse. Palvelimella ei tarvitse olla niin paljon tehoa kuin ”Thin”-asiakkaiden palvelimella, jolloin palvelin tulee olemaan halvempi. ”Thick”-asiakkailla on parempi multimediasuorituskyky ja ne ovat eri sovelluksien ajossa joustavampia kuin ”Thin”-asiakkaat. [4.]

2.2 Monikerrosarkkitehtuurit

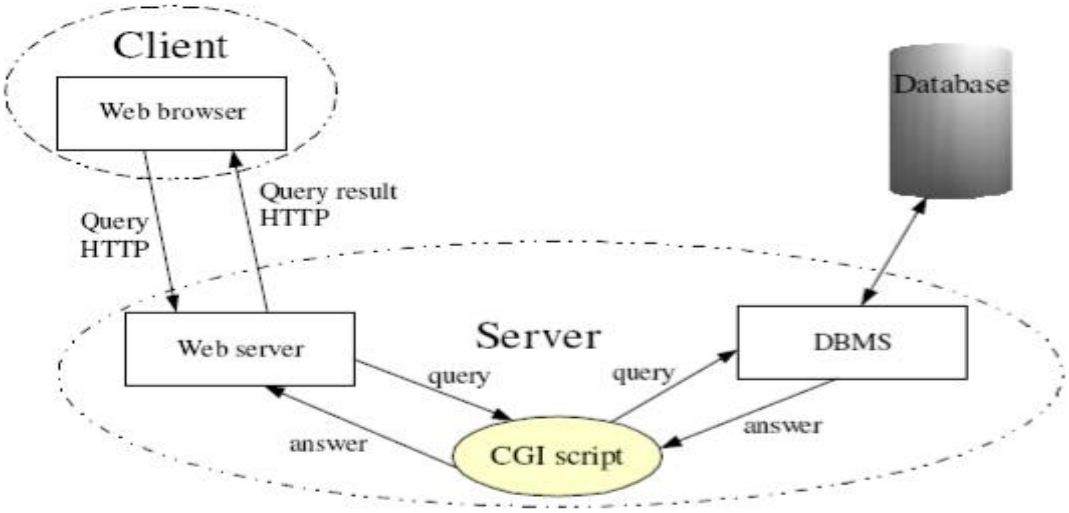
Tavallisella asiakas/palvelin-arkkitehtuurilla on kaksi erityyppistä kerrosta, asiakkaat ja palvelimet. Tästä syystä asiakas/palvelin-arkkitehtuuri on kutsuttu myös kaksikerrosarkkitehtuuriksi. [7.]

Kaksikerrosarkkitehtuuri on tyypillisesti sidottu tiettyyn asiakasratkaisuun. Päivitykset, muutokset jne. heijastuvat usein myös asiakkaiden toimintaan liittyviin vaatimuksiin. [5.]

Kaksikerrosarkkitehtuurissa asiakas ja palvelin ovat suorassa yhteydessä, tämä on nopea mutta joustamaton tapa. Kaksikerrosarkkitehtuuri soveltuu huonosti palvelemaan suuria kutsu-/käyttäjämääriä ja toiminnallisuuden jakaminen toteutuksen jälkeen on hankalaa. [8.]

2.2.1 Kolmikerrosarkkitehtuuri

Jotkut verkot koostuvat kolmesta eri kerroksesta, joita ovat asiakkaat, asiakkaiden pyyntöjä suorittavat sovelluslogiikkakerrokset ja tietokantapalvelimet. Tätä arkkitehtuuria kutsutaan kolmikerrosarkkitehtuuriksi. [7.] Kuva 4 havainnollistaa kolmikerrosarkkitehtuuria.



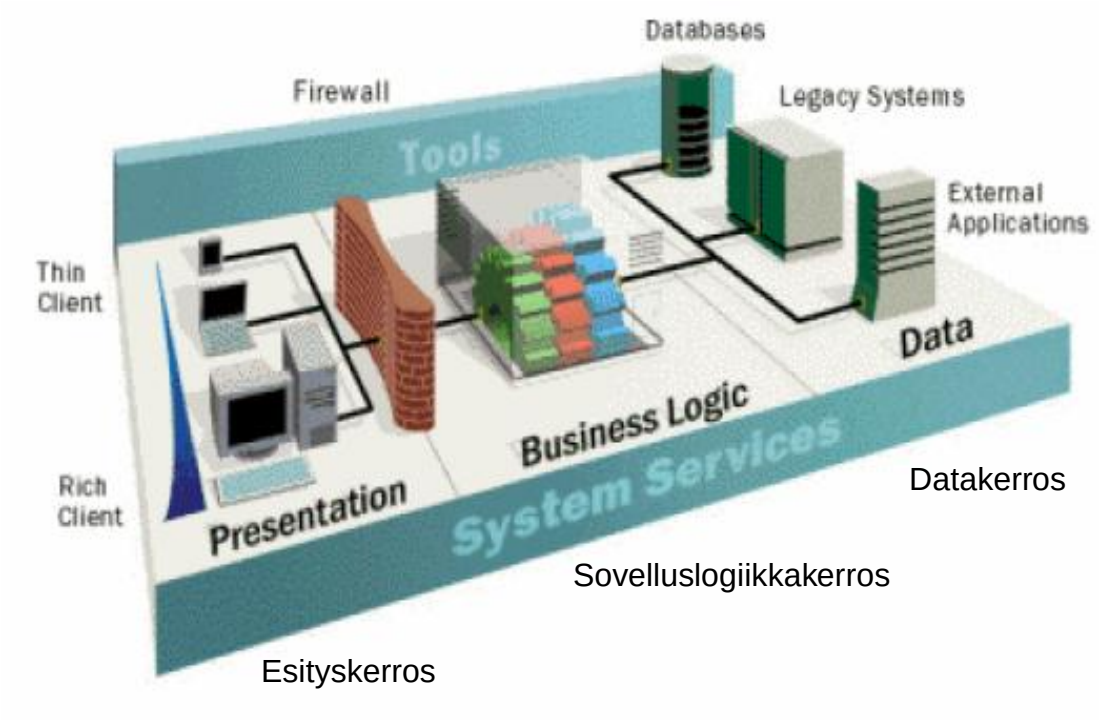
Kuva 4. Esimerkki kolmikerrosarkkitehtuurista [9, s. 877]

Kolmikerrosarkkitehtuurissa erotetaan esityskerros, sovelluslogiikkakerros ja tietolähteet. Asiakas kommunikoi ainoastaan välikerroksen (middle tier) kanssa, joka kommunikoi tietovarastojen, taustasovellusten tai tietolähteiden kanssa. [5.]

Kolmikerrosarkkitehtuuri parantaa perinteiseen kaksikerrosarkkitehtuuriin (asiakas-palvelin) verrattuna ylläpidettävyyttä, skaalautuvuutta, joustavuutta ja uudelleenkäytettävyyttä. [5.]

2.2.2 n-kerrosarkkitehtuurit

n-kerroksisessa monikerrosarkkitehtuurissa voi palveluja olla useita, esimerkiksi erilaisia business-logiikkaa toteuttavia sovelluslogiikkakerroksia ja tietokanta-järjestelmiä. Monikerrosarkkitehtuurin toiminta ja resurssien hallinta on tehokkaampaa kuin kaksikerrosarkkitehtuuri. Kuva 5 havainnollistaa monikerrosarkkitehtuuria.



Kuva 5. Monikerrosarkkitehtuuri [10]

Monikerrosarkkitehtuuri erottaa dataa selkeästi asiakasprosesseista ja mahdollistaa datan abstrahoinnin [10]. Tällöin kuormitus voidaan tasapainottaa eri palvelimien välillä paremmalla tavalla. [7.]

Moni- tai kolmikerrosarkkitehtuurissa sisäinen rakenne ja toiminta ovat piilotettuna. Asiakas/palvelin-arkkitehtuurissa palvelinpään muuttaminen edellyttää myös asiakaspään muuttamista ja protokollan muuttuessa, muutokset tulee tehdä sekä asiakas- että palvelinpäähän. Kaksikerrosarkkitehtuurissa asiakaspään muuttaminen ei ole mahdollista. Monikerrosarkkitehtuurissa palvelinpään protokollan tai asiakaspään muuttamiseen tarvitsee päivittää ainoastaan välikerrosta. [8.]

Monikerrosarkkitehtuuri antaa mahdollisuuden datan rakenteen muuttamiseen ja päivittämiseen, ilman että asiakasohjelmia tarvitsee muuttaa. Uusien asiakaslaitteiden ja järjestelmien kehittäminen on mahdollista ilman muutoksia tietokantoihin ja niiden saantimekanismeihin. [10.]

n-kerrosarkkitehtuurin haittana on sen raskas verkkokuormitus. Ohjelmointi ja testaus ovat paljon vaikeampia suorittaa verrattuna kaksikerrosarkkitehtuuriin, koska useamman laitteen pitää viestiä keskenään. [7.]

Monikerrosarkkitehtuuri soveltuu järjestelmiin, jotka ovat laajoja ja palvelevat suuria asiakasvirtoja ja useita erityyppisiä asiakkaita. Se soveltuu järjestelmiin, joissa data ja sovellukset ovat heterogeenisia. n-kerrosarkkitehtuuri on myös sopiva järjestelmiin, jotka muuttuvat usein ja niihin liittyvä data kerätään useista lähteistä. [8.]

3 AUTENTIKOINTI

Autentikointi eli todentaminen on käyttäjien kirjautumisen yhteydessä käytettävä menetelmä, jolla varmistetaan käyttäjien oikeellisuus. Autentikoinnilla varmennetaan käyttäjän (tai palvelun) identiteetti. Tämä voidaan tehdä kolmella eri tavalla tai niiden yhdistelmällä. Ensimmäisenä voidaan käyttää käyttäjän tietoa, jossa esimerkiksi voidaan käyttää salasanaa, PIN-koodia tai salausavainta. Toisena voidaan käyttää käyttäjän omistamaa ainetta, kuten esimerkiksi henkilöllisyystodistusta. Viimeisenä voidaan käyttää käyttäjän fyysistä ominaisuutta, jossa esimerkkinä voi olla sormenjälki, silmän värikalvo tai allekirjoitus. [11.]

3.1 Tietoturvapalvelut ja tietoturvauhat

Tiedon suojaustarve ilmaistaan usein tietoturvapalvelujen avulla. Tiedon luottamuksellisuudella (confidentiality) tarkoitetaan, että vain valtuutetut (authorized) voivat nähdä tiedon sisällön. [12.]

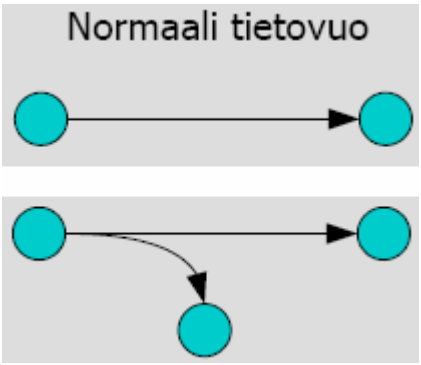
Tiedon eheydellä (integrity) tarkoitetaan, ettei tieto ole muuttunut siirron tai säilytyksen aikana. Myös julkiselta tiedolta voidaan edellyttää eheyttä. Tiedon autenttisuudella (authenticity) tarkoitetaan, että tiedon lähettänyt osapuoli on se, joksi hän itsensä tiedon vastaanottaneelle osapuolelle esittelee. [12.]

Kiistämättömyydellä (non-repudiation) tarkoitetaan, että tiedon lähettäjä ei voi kiistää lähettäneensä tietoa tai että tiedon vastaanottaja ei voi kiistää tiedon vastaanottoa. Käytettävyydellä (availability) tarkoitetaan, että tiedon tulee olla valtuutettujen käytettävissä heidän sitä tarvitessaan. [12.]

Pääsynvalvonnalla (access control) tarkoitetaan, että vain valtuutetuilla on pääsy palveluun ja sen sisältämiin tietoihin. Tarkastettavuudella (auditability) tarkoitetaan, että tietoturvapolitiikan toteutuminen voidaan verifioida. [13, 14.]

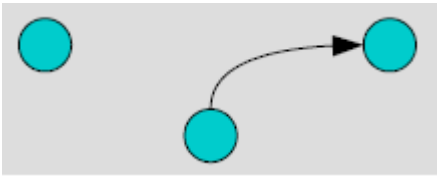
Tietoturvauhkia voidaan luokitella useilla tavoilla. Stallingsin mukaan tietoturvauhat ovat jakaantuneet neljään eri perusluokkaan [14]: sieppaukseen, väärentämiseen, modifiointiin ja keskeytykseen.

Sieppauksella (interception) tarkoitetaan tiedon päätymistä luvattoman osapuolen (henkilö, ohjelmisto, tietokone tms.) haltuun. Sieppaus uhkaa luottamuksellisuutta, ja se voidaan toteuttaa esimerkiksi salakuuntelun (eavesdropping) tai troijalaisen avulla. [12.] Kuvasta 6 nähdään sieppauksen toimintaperiaate.



Kuva 6. Sieppauksen toimintaperiaate [15]

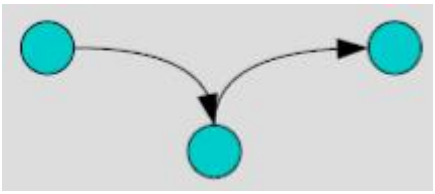
Väärentämisellä tarkoitetaan luvattoman osapuolen tuottamien tietojen hyväksymistä oikeina tietoina. Väärentäminen uhkaa autenttisuutta, ja se voidaan toteuttaa esimerkiksi käyttämällä nauhoitettuja sanomia tai naamioitumalla siepattujen todentamistietojen avulla. [12.] Kuvasta 7 nähdään väärentämisen toimintaperiaate.



Kuva 7. Väärentämisen toimintaperiaate [15]

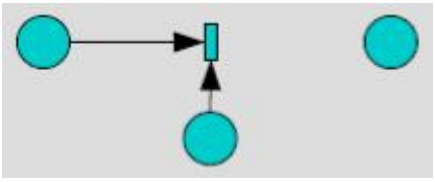
Modifioinnilla (modification) tarkoitetaan tiedon sisällön luvatonta muuttamista. Modifiointi uhkaa tiedon eheyttä, ja se voidaan toteuttaa esimerkiksi henkilö keskellä -hyökkäyksenä (man-in-the-middle), jossa sanomat ohjataan modifiointiin kykenevän ohjelmiston kautta esimerkiksi väärentämällä ensin reitittimien reititys-

tauluja tai nimipalvelun antamia osoitetietoja. [12.] Kuvasta 8 nähdään modifioinnin toimintaperiaate.



Kuva 8. Modifioinnin toimintaperiaate [15]

Keskeytyksellä (interruption) tarkoitetaan järjestelmän toiminnan luvatonta häirintää siten, että järjestelmän luvallinen käyttö estyy. Tämä on uhka käytettävyydelle. Keskeytys voidaan toteuttaa esimerkiksi palvelun esto -hyökkäyksenä (denial-of-service) lähettämällä kohteelle sallittuja pyyntöjä suurella nopeudella. [12.] Kuvasta 9 nähdään keskeytyksen toimintaperiaate.



Kuva 9. Keskeytyksen toimintaperiaate [15]

3.2 Autentikointimekanismit

3.2.1 Salasanat

Salasanat kuuluvat luokkaan ”mitä käyttäjä tietää”. Salasanat ovat yleisin ja vanhin käytössä oleva autentikointimekanismi. Ne olivat käytössä ensimmäisen kerran sotalinjojen pitämisessä. [16.]

Salasanojen käyttötavoista on kaksi eri vaihtoehtoa. Kertakäyttöisessä salasanassa turva perustuu salasanan vaihteluun eikä niiden käytössä verkkoliikennettä tarvita salata. Verkkopankkien salasanalistat ovat kertakäyttöisiä salasanoja. [16.]

Toisena vaihtoehtona ovat pysyvät salasanat. Yleensä käytössä olevat pysyvät salasanat ovat liian helppoja ja niitä tulisi vaihtaa tietyin väliajoin. Pysyviä salasanoja vaihdettaessa salasanan tulee vaihtua kokonaan. Eli salasana ”salasana”:n muuttaminen ”salasana1”:ksi ei ole vaihtamista. [16.]

Salasanojen vahvuudet ovat niiden helppo toteutus ja hinta. Salasanoja voidaan hyödyntää ohjelmistotasolla, silla ne eivät vaadi erillisiä lisälaitteita. Salasanojen heikkoudet johtuvat helppojen salasanojen käytöstä ja salasanat ovat alttiita salakuuntelulle. [16.]

Heikko autentikointi perustuu salasanan tai vastaavan metodin käyttöön. Vahva autentikointi perustuu siihen, ettei salasanoja tai avaimia lähetetä verkon yli. Vahva autentikointi perustuu myös vähintään kahteen autentikointimekanismin käyttöön. [16.]

3.2.2 Toimikortit

Toimikortit kuuluvat luokkaan ”mitä käyttäjällä on”. Toimikortteja on erityyppisiä: älykortit ja haaste/vaste-tyyppiset toimikortit. Älykorteilta löytyy useimmiten prosessori ja käyttäjän henkilökohtaista tietoa. HST-kortti ja JavaCard ovat esimerkkejä älykortista. Haaste/vaste-tyyppisen toimikortin käyttäjä tietää kortin PIN-koodin ja omistaa toimikortin. [16.]

Toimikorttiin voidaan sisällyttää useita toimintoja, eli se voi sisältää pääsyn useisiin kohteisiin. Toimikortit ovat kooltaan pieniä, ja ne ovat saavuttaneet suosiota. Toimikorttien useat toiminnot voivat olla heikkous, koska tällöin niihin voidaan tartuttaa viruksia. Toimikortti yksinään ei ole vahva autentikointimekanismi. Kuvasta 10 nähdään esimerkki HST-kortista. [16.]

Kuva 10. HST-kortti [16]

3.2.3 Biometria

Biometria kuuluu luokkaan ”mitä käyttäjä on”. Biometrian tekniikassa haetaan ihmisestä uniikkeja piirteitä. Esimerkkeinä voivat olla silmän värikalvo, silmän verkkokalvo, sormenjälki, kämmenen muoto, allekirjoitus ja ääni. Sormenjälki on vanhin autentikointimekanismi. Se perustuu sormen iholla olevien juovien tutkimiseen ja on suhteellisen luotettava. Silmän värikalvon kohteena ovat silmän värikalvolla sijaitsevat verisuonet, ja se on tarkin biometrisistä todennustavoista. [16.]

Biometrialla saavutetaan korkeimman tason varmuus todentamisesta. Se on helppo käyttää, koska käyttäjän ei tarvitse kantaa mukanaan esim. älykortteja, avaimia tms. Biometrialla kerätty tieto on todella henkilökohtaista. Useimmat biometriset lukijat ovat kalliita, ja biometria vaatii erillisiä laitteita. [16.]

3.3 Kaksivaiheinen autentikointi

Kaksivaiheinen autentikointi syntyy yhdistämällä kaksi autentikointimekanismia kolmesta mahdollisesta. Esimerkiksi käyttäjällä oleva toimikortti ja sen käyttäjän tietämä salasana, voidaan käyttää yhtä aikaa. Tämän esimerkin autentikointivaiheet ovat, mitä käyttäjällä on ja mitä käyttäjä tietää. Pankkikortti on toinen esimerkki kaksivaiheisesta autentikoinnista. Autentikoinnissa käytetään kortin fyysistä muotoa ja kortin PIN-koodi lähtee tietona. [17.]

Asiantuntijoiden mukaan kaksivaiheinen autentikointi voi alentaa verkossa tapahtuvia varkauksia, ”phishing”-lähetyksiä ja muita verkkopetkutuksia, koska pelkkä salasanan käyttö antaa mahdollisuuden päästä uhrin informaatiolle. Jotkut tietoturvallisuusmenetelmät edellyttävät kolmevaiheista autentikointia, mikä tarkoittaa biometrisen tiedon (esim. sormenjälki) käyttöä toimikortin ja salasanan kanssa. [17.]

3.4 JAAS-autentikointirajapinta

JAAS-rajapinta mahdollistaa käyttäjäkohtaisen autentikoinnin ja pääsynvalvonnan (access control). Java itse mahdollistaa pääsynvalvonnan perustuen siihen, mistä ohjelma on tullut ja kuka sen on allekirjoittanut, sekä haluttaessa ACL:n (Access Control List) avulla. JAAS taas jatkaa näitä ominaisuuksia tuoden mukaan mahdollisuuden valvontaan sen mukaan, kuka ohjelmaa ajaa. JAAS:n rajapinta kattaa siis sekä autentikoinnin että authorisoinnin, ja nämä osiot tavallaan jakavatkin sen kahteen osaan. [18.] Taulukko 1 näyttää JAAS:n luokat ja rajapintaluokat.

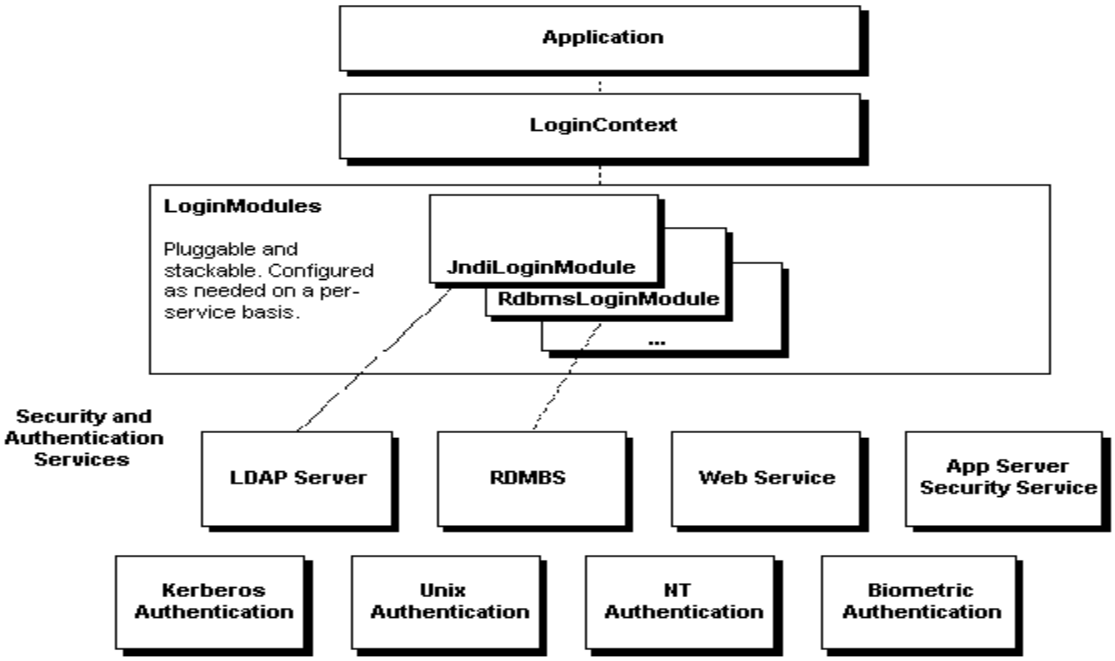
Taulukko 1. JAAS:n luokat ja liitännät [19]

Yleiset	Subject, Principal, credential (credential ei ole luokkaa, mutta voi olla mikä tahansa olio)
Autentikointi	LoginContext, LoginModule, CallbackHandler, Callback
Autorisointi	Policy, AuthPermission, PrivateCredentialPermission

Autentikoinnin osalta JAAS antaa mahdollisuuden määritellä halutun määrän eri autentikointitapoja (toteutuksia) kutakin sovellusta kohden. Eri tavoille määritetään niiden pakollisuustaso (required, requisite, sufficient, optional), mikä vaikuttaa niiden käyttäytymiseen kirjautumisen yhteydessä. Onnistuneen autentikoinnin edellytys on siis kaikkien ohjelman vaatimien autentikointitapojen onnistuminen. Eri tapoja ja niiden pakollisuustasoja voidaan tarvittaessa lisätä tai muuttaa muokkaamalla autentikointi-konfiguraatio -tiedostoa. [18.]

Autentikointi-konfiguraatio -tiedostossa määritetään loginmoduulin ominaisuuksia. Näitä ovat esimerkiksi tietokannan osoite, nimi, salasana ja käyttäjätunnus tai tietokannan ajuri. Tarvittaessa voidaan lisätä uusia loginmoduuleja tai poistaa olemassa olevia. Autentikointi-konfiguraatio -tiedoston paikka määritetään järjestelmän ominaisuudella *javax.security.auth.login.config*. Tämä ominaisuus asetetaan komennolla *java -Djava.security.auth.login.config=jaas.config JaasTest*.

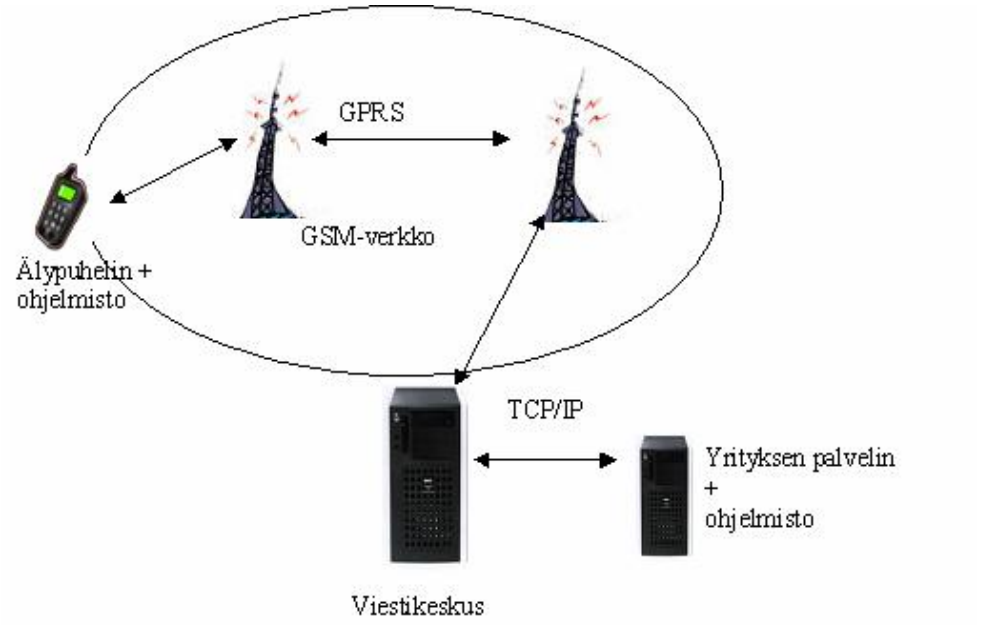
Kuvassa 11 on havainnollistettu JAAS:n ylemmän tason arkkitehtuuria. Sovelluskoodi käsittelee ensisijaisesti ”LoginContext”-luokkaa. ”LoginContext”-luokan alapuolella on dynaamisesti konfiguroitavia loginmoduuleja, jotka hoitavat varsinaista autentikointia. [19.]



Kuva 11. Jaas:n ylemmän tason arkkitehtuuri [19]

4 JÄRJESTELMÄN NYKYTILAN KUVAUS

KajaPro Oy:llä on käytössä useita sovelluksia, jotka toimivat asiakas/palvelin-arkkitehtuurilla. Näissä sovelluksissa asiakaspään laitteena käytetään älypuhelinta (Kuva 12). [20.]



Kuva 12. Asiakas/palvelin-järjestelmä mobiiliverkossa [20]

Käytössä olevien mobiilisovelluksien pohjalta jatketaan mobiiliohjelmistokehyksen kehitystä, joka mahdollistaa mobiilisovelluksien kommunikoinnin palvelimen kanssa sokettiyhteyden kautta. Ohjelmistokehyksen nimi on ”Mobile Client/Server Framework(MCF)”.

Tyypillisesti älypuhelimessa on käyttöliittymä, jonka avulla lähetetään parametroitua tietoa palvelinsovellukselle [20]. Palvelinsovellukselta myös vastaanotetaan tietoa älypuhelimeen [20]. Toistaiseksi järjestelmässä tietoturvaseikkoja on huomioitu vain lähtevien viestien salaamisen kannalta. Tämän insinöörityön kohteena oli älypuhelinsovelluksen tunnistaminen palvelinsovelluksessa.

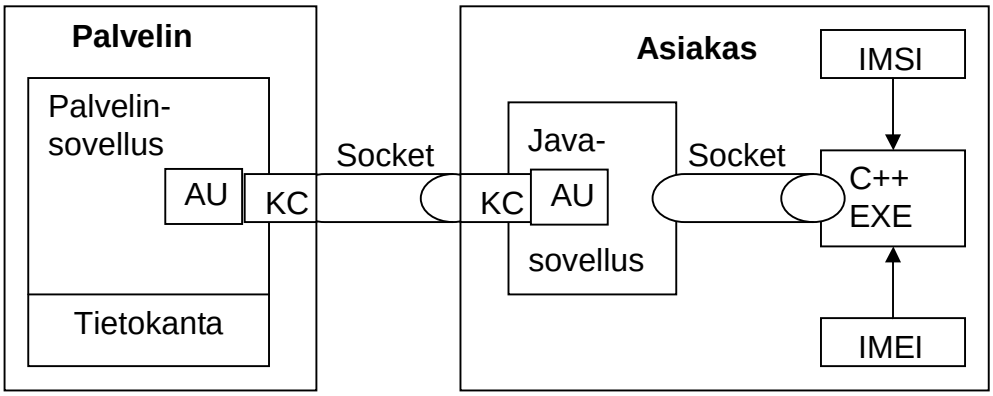
Järjestelmän nykyisessä versiossa yhteys puhelimen ja palvelimen välille on muodostettu sokettiyhteyden avulla. Palvelimelle on toteutettu myös toiminto, joka vaatii asiakkaan kirjautumisen palvelimelle. Tämä palvelintoiminto on kuitenkin vielä keskeneräinen, koska se ainoastaan varmistaa, ettei kaksi samannimistä henkilöä voi kirjautua palvelimelle samanaikaisesti. Toisin sanoen palvelinsovellus sallii kenen tahansa kirjautua palvelimelle. [20.]

Palvelin ei pysty erottamaan, onko asiakas oikeasti puhelinsovellus vai ei. Tällöin tiedon lähettäminen palvelimelle ja vastaanottaminen palvelimelta on mahdollista myös normaalilla pc-sovelluksella, joka mallintaa puhelinsovelluksen toimintaa. Palvelinta ei voi kaataa puhelinsovelluksen avulla, koska kaikki asiakkaat käsitellään eri istuntoina. Asiakkaiden istunnon tilaa myös tarkistetaan tietyn väliajoin. Tämän tarkistuksen avulla palvelimelta poistetaan asiakkaat, jotka eivät ole tehneet mitään tietyllä ajanjaksolla. Jos vastaanotettava tieto aiheuttaa poikkeuksen palvelimella, asiakkaan istunto suljetaan. [20.]

5 AUTENTIKOINTIRAJAPINNAN TOTEUTTAMINEN

Kuvassa 13 esitetään autentikointirajapinnan paikat (AU) MCF-arkkitehtuurissa. Autentikointirajapinta tulee olla palvelimessa aina käytössä. Asiakas lähettää palvelimelle viestiä, jonka jälkeen palvelin ottaa asiakkaan tunnisteen vastaan, tunnistaa asiakkaan autentikointirajapinnan kautta ja lähettää vastauksen asiakkaalle. Asiakkaalla pyritään aikaansaamaan ”Thin”-asiakasmalli.

Kun järjestelmässä lähetetään tietoa asiakkaan ja palvelimen välillä, käytetään autentikointirajapintaa, KajaCrypto salausrajapintaa ja sokettiyhteyttä. Asiakkaan autentikointirajapintaa tarvitaan vain silloin, kun kirjautumisessa tarvitaan käyttäjätunnus/salasana-paria. IMEI- ja IMSI-koodien saanti eivät olleet tämän työn aiheena. Niitä käsitellään Symbian C++ -sovelluksessa toisessa insinöörityössä.



Kuva 13. Autentikointirajapinnan paikat MCF-arkkitehtuurissa

Ohjelmistokehityksen tukirankana oli inkrementaalinen ohjelmistokehitys, jonka mukaan ohjelmisto kehitettiin. Ohjelmistoa kehitettiin vaiheittain, siten että aluksi suunniteltiin ja koodattiin vain pieni osa toiminnallisuutta, joka testattiin. Tämän jälkeen suunniteltiin lisää toimintoja, jotka koodattiin ja testattiin. Näin jatkettiin, kunnes koko ohjelmisto oli saatu valmiiksi.

5.1 Vaatimukset

Kun mobiilisovellukselta lähetetään tietoa tai ohjauskomentoja palvelinsovellukselle, tulee jollain tapaa varmistaa, että kyseessä on käyttäjä tai laite, jolloin tietoa tai komentoja voidaan hyväksyä vastaanotettavaksi. Tunnistusta varten täytyy asiakas- ja palvelinpäässä tehdä autentikointirajapinta, jossa tunnistukseen on käytettävissä laitteen IMEI-koodi, puhelinliittymän IMSI-koodi, käyttäjätunnus/salasana-pari tai niiden yhdistelmä.

Toiminnolle on järjestelmässä asetettu seuraavia vaatimuksia, kuten uuden asiakkaan kirjautuessa, otetaan vastaan palvelimessa lähettäjän tunniste ja käyttäjätunnus/salasana-pari. Vastaanotettuja tietoja verrataan tietokannassa oleviin tietoihin. Palvelin hyväksyy tai hylkää kirjautumisen.

Autentikointitavan pitää olla käyttäjän valittavissa. Käyttäjän pitää pystyä valitsemaan, käyttääkö autentikointirajapinta IMEI-, IMSI-koodia vai käyttäjätunnus/salasana-paria autentikointia varten. Jotta saadaan hyvä tietoturvallisuus, autentikoinnin pitää olla kaksivaiheinen, eli tunnisteita ja käyttäjätunnus/salasana-paria pitää käyttää yhdessä.

Ilman autentikointia ulkopuolisen on mahdollista kirjautua järjestelmään sisään esimerkiksi kaapatun viestin avulla. Erityisesti puhelinsovelluksessa autentikoinnilla voitaisiin estää mahdollinen vahingonteko, jos puhelin on hävinnyt. Tämä toiminto suojaisi palvelin- ja puhelinsovellukseen tallennettuja tietoja. Ilman autentikointia olisi mahdollista muuttaa ainakin puhelimeen tallennettuja tietoja ja luultavammin palvelimella sijaitsevia tietoja. Näitä muutoksia ei välttämättä havaittaisi heti ja tästä syystä saattaisi pahassa tapauksessa aiheutua jopa taloudellisia vahinkoja järjestelmää käyttävälle yritykselle. [20.]

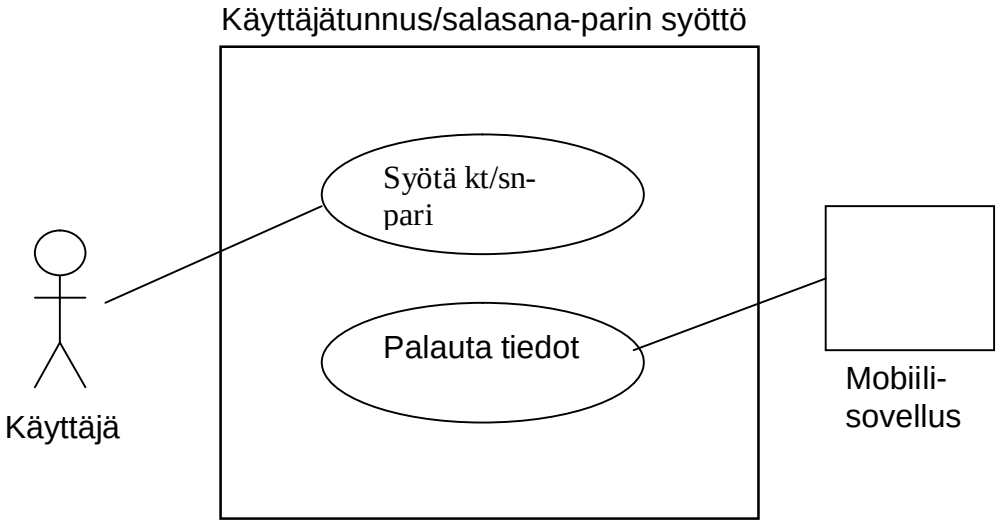
Jotta autentikointirajapinta saadaan toteutettua, täytyy palvelimelle toteuttaa uusi tietokantataulu ja autentikointirajapinta, jossa verrataan saatuja tietoja kannassa oleviin tietoihin. Tähän tauluun määritellään kaikki tarvittavat tiedot asiakkaista. Toiminnon avulla voidaan kerätä muitakin tietoja asiakkaista, kuten yhteyden kesto, yhteyksiä viikossa ja niin edelleen. [20.]

5.2 Autentikointirajapinnan toteutus

Saman rajapinnan käyttö asiakkaalla sekä palvelimella ei ollut mahdollista, koska asiakasrajapinta käyttää MIDP:n käyttöliittymäkirjastoa ja palvelinrajapinta käyttää JAAS-rajapintaa. Tästä syystä autentikointirajapinnan asiakas ja palvelinpuolet toteutettiin erikseen.

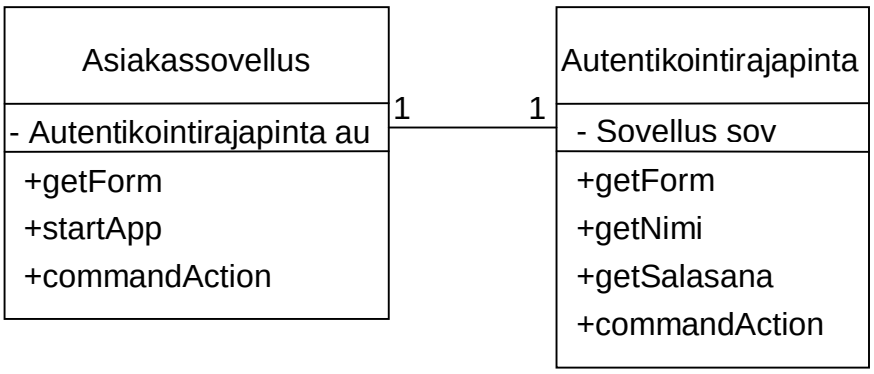
5.2.1 Asiakkaan autentikointirajapinta

Asiakasrajapinnalla ei ole paljoa toiminnallisuutta. Asiakasrajapinta antaa mahdollisuuden syöttää käyttäjätunnus/salasana-pari, MIDP:n käyttöliittymä komponenttien avulla. Tämän jälkeen asiakasrajapinta palauttaa käyttäjätunnus/salasana-parin mobiililaitteen sovellukselle. Kuva 14 havainnollistaa asiakasrajapinnan käyttötapauskaavion.



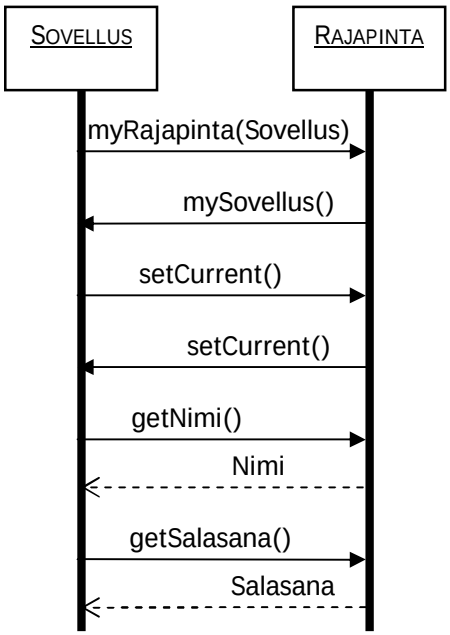
Kuva 14. Asiakasrajapinnan käyttötapauskaavio

Asiakasrajapinta muodostuu yhdestä luokasta. Varsinaisessa mobiilisovelluksen luokassa, pitää muodostaa asiakasrajapinnan luokan instanssi, samalla annetaan sovellusluokan instanssi parametrina. Asiakasrajapintaluokka sisältää MIDP:n käyttöliittymäkirjasto, mutta se ei ole midletluokkaa (Kuva 15).



Kuva 15. Demosovelluksen ja asiakasrajapinnan luokkakaavio

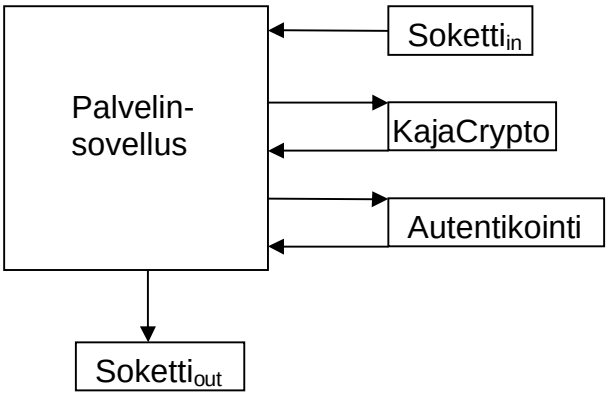
Ennen kuin syötetään käyttäjätunnus ja salasana, vaihdetaan näkymä asiakasrajapinnan näkymäksi. Sen jälkeen palataan sovellusnäkymään ja haetaan sieltä autentikointirajapinnasta syötetyt tiedot (Kuva 16).



Kuva 16. Asiakasrajapinnan sekvenssikaavio

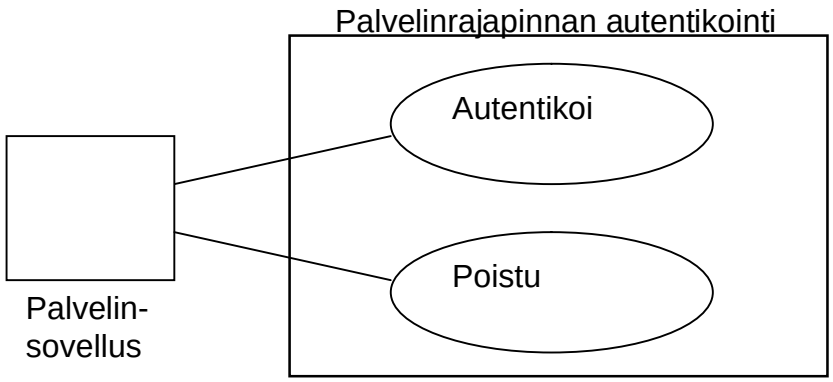
5.2.2 Palvelimen autentikointirajapinta

Palvelinrajapinta käyttää autentikointia varten Javan JAAS-rajapintaa ja samalla mahdollistaa käyttäjäkohtaisen autentikoinnin. Kuvassa 17 havainnollistetaan palvelimen autentikointirajapinnan paikka.



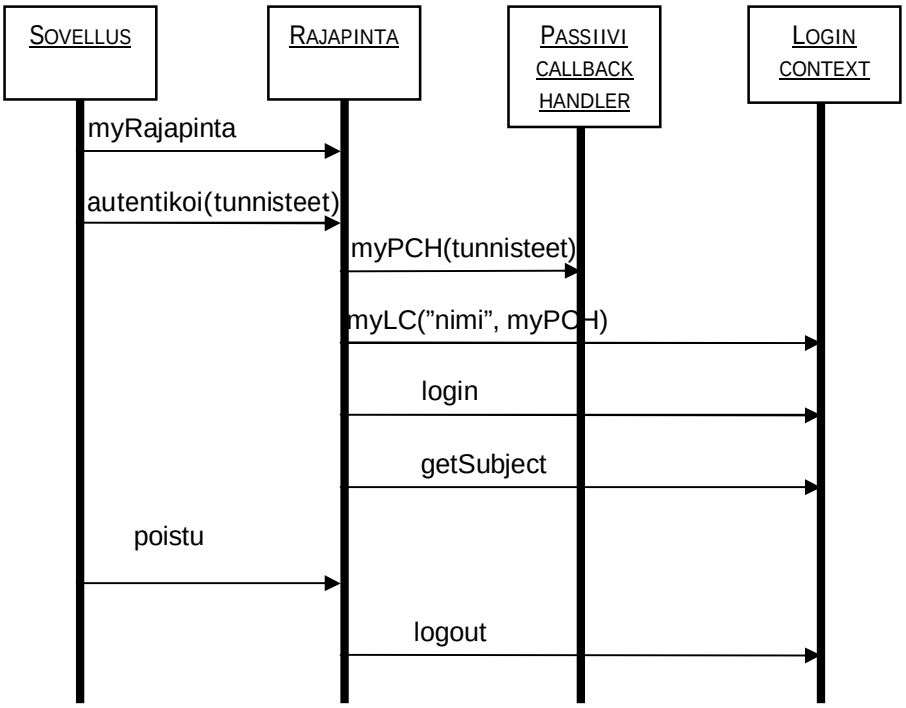
Kuva 17. Autentikointirajapinnan paikka palvelimessa

Tiedot tulevat palvelimelle sokettiyhteyden kautta. Sen jälkeen palvelinsovellus käyttää KajaCrypto rajapintaa salauksen purkamiseksi ja palvelinsovellus lähettää viestistä poimitut käyttäjän tunnisteet autentikointirajapinnalle. Autentikointirajapinta vertaa tunnisteita tietokannassa oleviin tietoihin ja palauttaa vastauksen palvelimelle. Kuva 18 havainnollistaa palvelinrajapinnan käyttötapauskaavion.



Kuva 18. Palvelinrajapinnan käyttötapauskaavio

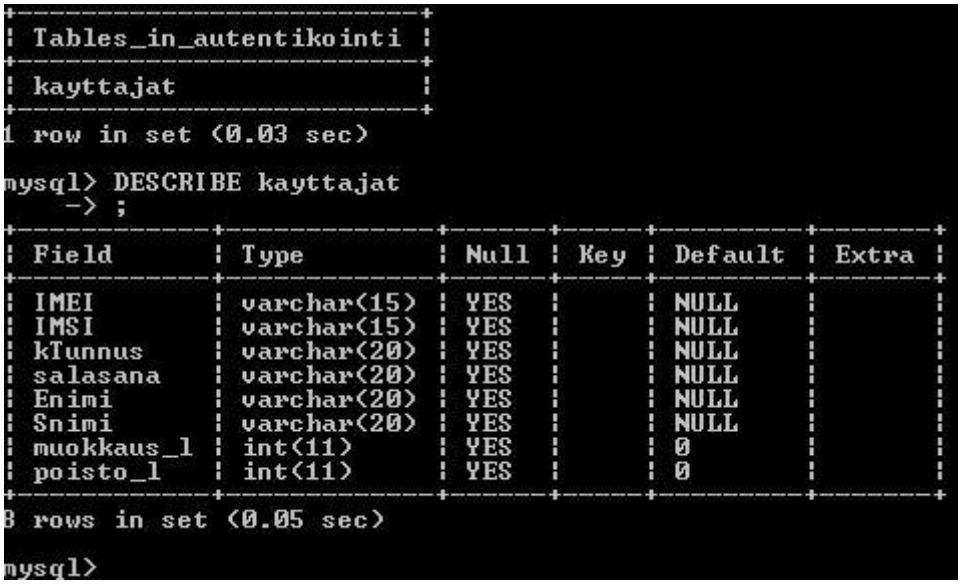
Palvelinrajapinta on Java-luokka, mikä käyttää JAAS-autentikointirajapintaa. Palvelinrajapinnan luokkakaavio löytyy liitteestä A. Palvelinsovelluksen sijalle tehtiin demosovellus, jossa kutsuttiin ”autentikoi”-metodia. Demosovellus antaa ”autentikoi”-metodin tarvitsemat tunnisteet parametrina. Sen jälkeen demosovellus muodosti instanssin palvelinrajapinnasta. Kuva 19 havainnollistaa palvelinrajapinnan sekvenssikaavion.



Kuva 19. Palvelinrajapinnan sekvenssikaavio

Kun kutsuttiin ”autentikoi”-metodi, palvelinrajapinta muodosti instanssit ”LoginContext”- ja ”passiiviCallBackHandler”-luokista ja yritti kirjautua JAAS-rajapinnan kautta. Kirjautumisessa verrataan tunnisteita tietokannassa oleviin tietoihin. Kirjautumisen onnistuessa, rajapinta ottaa käyttäjän tiedot ”getSubject”-metodilla.

Käyttäjien tietojen säilyttämiseksi muodostettiin palvelimessa käyttäjätietokanta käyttäen MySQL-tietokantaohjelmaa. Tietokannan yhteyksien muodostamiseen käytettiin JDBC-rajapintaa ja MySQL-”connector”-ajuria. Palvelinrajapinnan taulukko muodostuu kuvan 20 mukaisista attribuuteista.



```
+--------------------------+
| Tables_in_autentikointi  |
+--------------------------+
| kayttajat                |
+--------------------------+
1 row in set (0.03 sec)

mysql> DESCRIBE kayttajat
    -> ;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| IMEI       | varchar(15) | YES  |     | NULL    |       |
| IMSI       | varchar(15) | YES  |     | NULL    |       |
| kTunnus    | varchar(20) | YES  |     | NULL    |       |
| salasana   | varchar(20) | YES  |     | NULL    |       |
| Enimi      | varchar(20) | YES  |     | NULL    |       |
| Snimi      | varchar(20) | YES  |     | NULL    |       |
| muokkaus_l | int(11)     | YES  |     | 0       |       |
| poisto_l   | int(11)     | YES  |     | 0       |       |
+------------+-------------+------+-----+---------+-------+
8 rows in set (0.05 sec)

mysql>
```

Kuva 20. ”kayttajat”-taulukon attribuutit

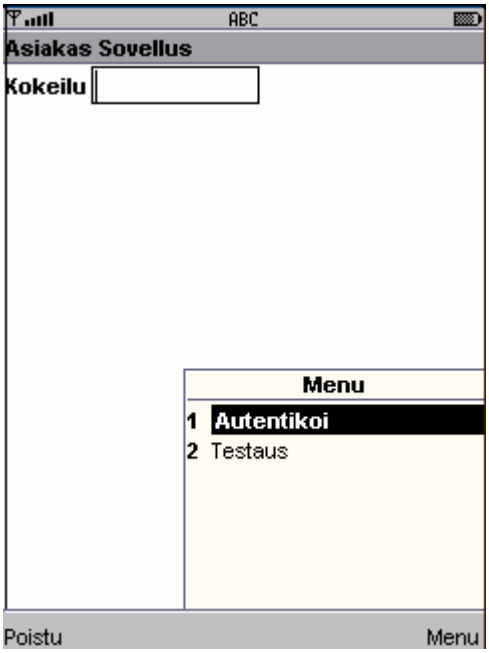
Käyttäjä pystyy valitsemaan autentikointitavan tekemällä ilmoituksen yhteyshenkilölle. Esimerkiksi käyttäjä Matti Meikäläisen halutessa käyttää autentikoinnissa IMEI-koodia ja käyttäjä/salasana-paria, taulukkoon kirjoitetaan käyttäjän IMEI-koodi, käyttäjätunnus/salasana-pari. Taulukon IMSI-koodi attribuutille kirjoitetaan ”ei_AUta” arvo.

6 AUTENTIKOINTIRAJAPINNAN TESTAUS

Molempien rajapintojen testaamiseksi suunniteltiin ja ohjelmoitiin yksinkertaiset asiakas- ja palvelindemosovellukset. Demosovelluksista puuttuivat soketti-yhteydet, koska ne eivät olleet tämän työn aiheena ja MCF-arkkitehtuurin uusi viestikehys ei ollut vielä valmis. Vanhasta viestikehyksestä puuttuivat käyttäjän tunnisteet. Demosovelluksissa syötettiin tietoja rajapinnalle, jonka jälkeen rajapinnat suorittivat annetut tehtävät.

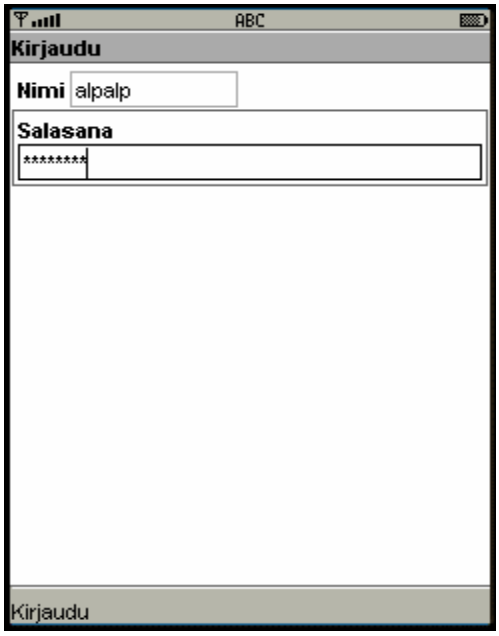
6.1 Asiakasrajapinta

Asiakasrajapinta testattiin NetBeansin emulaattorilla sekä Nokian 6670 -mobiililaitteella. Testausta varten tehtiin demosovellusluokka. Demosovelluksen valikossa olivat ”autentikoi”- ja ”testaus”-valinnat (Kuva 21).



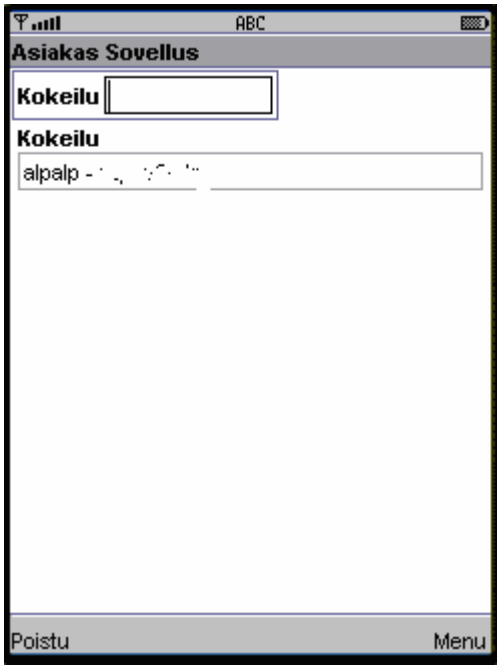
Kuva 21. Demosovelluksen valikko

Valitsemalla ”autentikoi”-valinta saatiin näkyviin autentikointirajapinnan käyttöliittymä, missä syötettiin käyttäjätunnus/salasana-pari (Kuva 22).



Kuva 22. Autentikointirajapinnan käyttöliittymä

Kirjautumisen jälkeen näytöllä oli asiakassovelluksen käyttöliittymä. Valikosta valittiin ”testaus”-valinta. Sovellus haki autentikointirajapinnasta tarvittavan käyttäjätunnus/salasana-parin ja sen jälkeen tulosti ne näytölle (Kuva 23).



Kuva 23. Käyttäjätunnus/salasana-pari demosovelluksessa

Asiakasrajapinta testattiin eri testitapauksilla. Testaukseen sisältyi kaksi yksikkötestausta, kaksi integrointitestausta ja yksi järjestelmätestaus. Asiakasrajapinnan testitapaukset olivat taulukon 2 mukaisia.

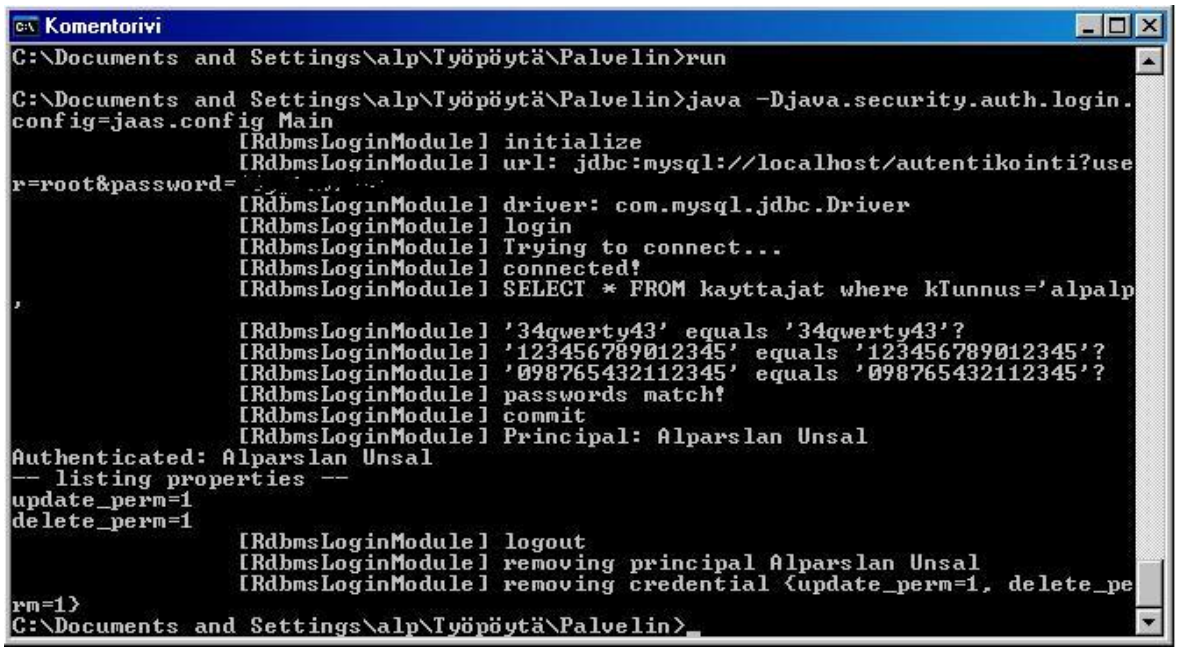
Taulukko 2. Asiakasrajapinnan testitapaukset

Toiminto	Testitapaus	Hyväksymiskriteerit
Käyttäjätunnus/salasanaparin syöttö	Tulostetaan syötetyt tiedot. Yksikkötestaus	Syötetyt tiedot tulostuvat mobiililaitteen näytölle
Demosovelluksen valikon toimivuus	Tarkistetaan, että toimiiko demosovelluksen valikko. Yksikkötestaus	Valitun valikon toiminnallisuus käynnistyy
Näkymän vaihto	Tarkistetaan, että muuuttuko näkymä rajapinnan ja demosovelluksen välillä. Integrointitestaus	Näkymä vaihtuu rajapinnan ja demosovelluksen välillä
Nimen ja salasanan haku	Tulostetaan nimi ja salasana haun jälkeen. Integrointitestaus	Haetut tiedot tulostuvat demosovelluksessa
Demosovelluksen ja asiakasrajapinnan toimivuus	Tarkistetaan, että täyttäkö ohjelmisto vaaditut vaatimukset. Järjestelmätestaus	Ohjelmisto toimii suunnittelun mukaisesti

6.2 Palvelinrajapinta

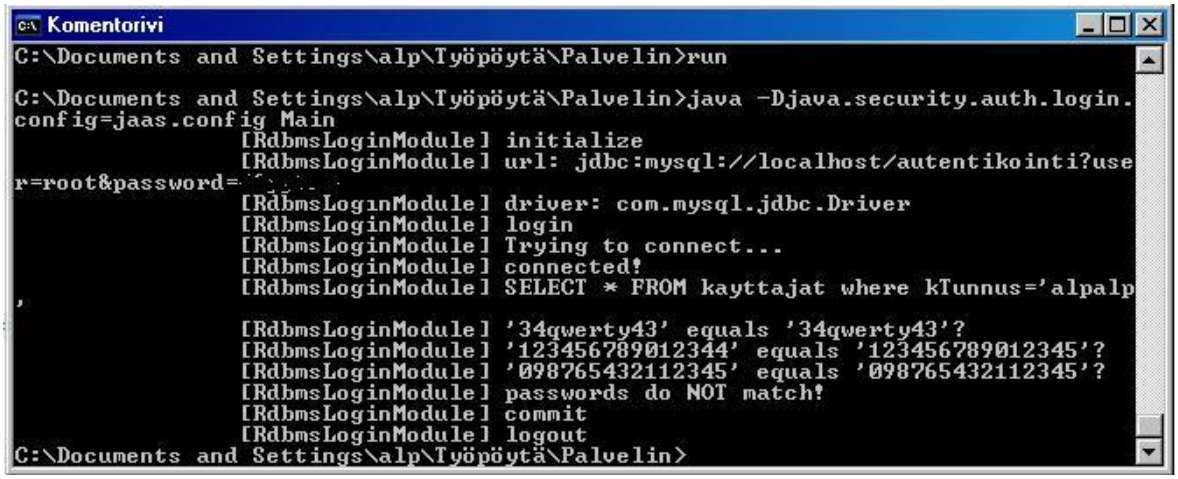
Palvelinrajapinta testattiin komentorivillä ”loginmoduuli”-luokan tarjoaman ”debug”-muuttujan avulla. Testausta varten tehtiin demosovellusluokka, joka käynnisti palvelinrajapinnan ja kutsui ”autentikoi”-metodia. Demosovellus antaa ”autentikoi”-metodin tarvitsemat tunnisteet parametrina. Todellisuudessa ”autentikoi”-metodin tarvitsemien parametrien pitää tulla palvelimelle sokettiyhteyden kautta ja sen jälkeen palvelinsovellus kutsuu ”autentikoi”-metodia. Autentikoinnin onnistuessa tulostetaan asiakkaan tiedot.

Tietokannan ”loginmoduuli”-luokan tarjoamalle ”debug”-muuttujalle voidaan antaa konfigurointi tiedostossa ”True”- tai ”False”-arvo. Jos muuttujan arvo on asetettu ”True”:ksi, silloin ”loginmoduuli”-luokka tulostaa ohjelman vaiheiden onnistumisesta kuten kuvassa 25. Kuvan tilanteessa autentikoinnissa käytetään IMEI-koodia, IMSI-koodia ja käyttäjätunnus/salasana-paria.



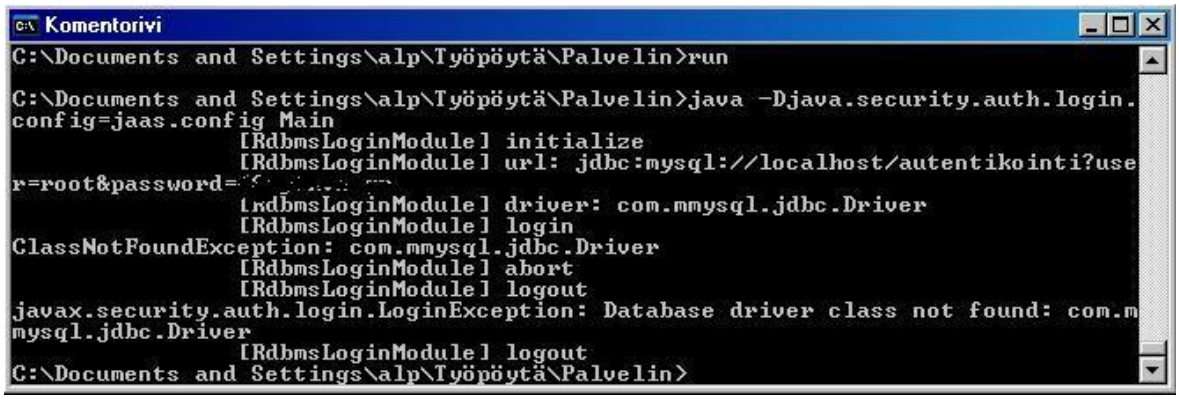
Kuva 25. Onnistunut autentikointi

”debug”-muuttujan avulla virhetilanteissa voidaan helposti seurata ohjelman kulkua. Kuva 26 havainnollistaa tilanteen, kun autentikointi ei onnistunut.



Kuva 26. Ei-onnistunut autentikointi

”debug”-muuttuja antaa myös mahdollisuuden seurata virhetilanteita. Esimerkkinä kuva 27 havainnollistaa tietokanta-ajurivirhetilanteen.



Kuva 27. Tietokanta-ajurivirhe

Palvelinrajapinta testattiin eri testitapauksilla. Testaukseen sisältyi kolme yksikkötestausta, kolme integrointitestausta ja yksi järjestelmätestaus. Palvelinrajapinnan testitapaukset olivat taulukon 3 mukaisia.

Taulukko 3. Palvelinrajapinnan testitapaukset

Toiminto	Testitapaus	Hyväksymiskriteerit
Pystyykö loginmoduuli ottaa yhteyttä tietokantaan?	Seurataan loginmoduulin debuggaus lauseita. Yksikkötestaus	Loginmoduulin debuggaus lauseet ilmoittavat yhteydestä
Pystyykö loginmoduuli verrata käyttäjän tunnisteet tietokannassa oleviin?	Seurataan loginmoduulin debuggaus lauseita. Yksikkötestaus	Loginmoduulin debuggaus lauseet ilmoittavat vertauksen tuloksesta
Pystyykö loginmoduuli poistamaan käyttäjän tiedot?	Seurataan loginmoduulin debuggaus lauseita. Yksikkötestaus	Loginmoduulin debuggaus lauseet ilmoittavat käyttäjän tietojen poistamisesta
Pystyykö ”autentikoi”-metodi käynnistämään JAAS rajapinta?	Seurataan loginmoduulin debuggaus lauseita. Integraatiotestaus	Loginmoduulin debuggaus lauseet tulostuvat
Autentikoi metodin paluuarvot	Tulostetaan metodin palauttamia arvoja. Integraatiotestaus	Metodin paluuarvot ovat odotettuja
Poistumisen toimivuus	Seurataan loginmoduulin debuggaus lauseita. Integraatiotestaus	Loginmoduulin debuggaus lauseet ilmoittavat poistumisesta
Demosovelluksen, palvelinrajapinnan ja JAAS rajapinnan yhteistoimivuus	Tarkistetaan, että täyttäkö ohjelmisto vaaditut vaatimukset. Järjestelmätestaus	Ohjelmisto toimii suunnittelun mukaisesti

7 TULOKSET JA TULOSTEN TARKASTELU

Molemmat rajapinnat vastasivat hyvin asetettuja tavoitteita, vaikka palvelinrajapinnalla oli ongelmia NetBeansin asetuksien kanssa.

7.1 Asiakasrajapinta

Asiakasrajapinta vastasi hyvin asetettuja tavoitteita. Asiakkaan demosovellus pystyi käynnistämään asiakasrajapinnan. Asiakasrajapinnalta syötettiin käyttäjätunnus/salasana-pari ja asiakasrajapinnan metodit palauttivat tunnisteet demosovellukselle tulostettavaksi. Asiakasrajapinnan testitulokset olivat taulukon 4 mukaisia.

Taulukko 4. Asiakasrajapinnan testitulokset

Toiminto	Testitapaus	Hyväksymiskriteerit	Tulos
Käyttäjätunnus/salasana-parin syöttö	Tulostetaan syötetyt tiedot. Yksikkötestaus	Syötetyt tiedot tulostuvat mobiililaitteen näytölle	Syötetyt tiedot tulostuivat mobiililaitteen näytölle
Demosovelluksen valikon toimivuus	Tarkistetaan, että toimiiko demosovelluksen valikko. Yksikkötestaus	Valitun valikon toiminnallisuus käynnistyy	Valitun valikon toiminnallisuus käynnistyi
Näkymän vaihto	Tarkistetaan, että muuuttuko näkymä rajapinnan demosovelluksen välillä. Integrointitestaus	Näkymä vaihtuu rajapinnan ja demosovelluksen välillä	Näkymä vaihtui rajapinnan ja demosovelluksen välillä
Nimen ja salasanan haku	Tulostetaan nimi ja salasana haun jälkeen. Integrointitestaus	Haetut tiedot tulostuvat demosovelluksessa	Haetut tiedot tulostuivat demosovelluksessa
Demosovelluksen ja asiakasrajapinnan toimivuus	Tarkistetaan, että täyttäkö ohjelmisto vaaditut vaatimukset. Järjestelmätestaus	Ohjelmisto toimii suunnittelun mukaan	Ohjelmisto toimi suunnittelun mukaisesti

7.2 Palvelinrajapinta

Palvelinrajapinta vastasi hyvin asetettuja tavoitteita. Palvelimen demosovellus pystyi käynnistämään palvelinrajapinnan ja palvelinrajapinta otti vastaan demosovelluksen lähettämät tunnisteet. Sen jälkeen palvelinrajapinta vertasi asiakkaan tunnisteita tietokannassa oleviin tietoihin. Palvelinrajapinnan testitulokset olivat taulukon 5 mukaisia.

Taulukko 5. Palvelinrajapinnan testitulokset

Toiminto	Testitapaus	Hyväksymiskriteerit	Tulos
Pystyykö loginmoduuli ottaa yhteyttä tietokantaan?	Seurataan loginmoduulin debuggaus lauseita. Yksikkötestaus	Loginmoduulin debuggaus lauseet ilmoittavat yhteydestä	Loginmoduulin debuggaus lauseet ilmoittivat yhteydestä
Pystyykö loginmoduuli verrata käyttäjän tunnisteet tietokannassa oleviin?	Seurataan loginmoduulin debuggaus lauseita. Yksikkötestaus	Loginmoduulin debuggaus lauseet ilmoittavat vertauksen tuloksesta	Loginmoduulin debuggaus lauseet ilmoittivat vertauksen tuloksesta
Pystyykö loginmoduuli poistamaan käyttäjän tiedot?	Seurataan loginmoduulin debuggaus lauseita. Yksikkötestaus	Loginmoduulin debuggaus lauseet ilmoittavat käyttäjän tietojen poistamisesta	Loginmoduulin debuggaus lauseet ilmoittivat käyttäjän tietojen poistamisesta
Pystyykö autentikoi metodi käynnistämään JAAS rajapinta?	Seurataan loginmoduulin debuggaus lauseita. Integraatiotestaus	Loginmoduulin debuggaus lauseet tulostuvat	Loginmoduulin debuggaus lauseet tulostuivat
Autentikoi metodin paluuarvot	Tulostetaan metodin palauttamia arvoja. Integraatiotestaus	Metodin paluuarvot ovat odotettuja	Metodin paluuarvot olivat odotettuja
Poistumisen toimivuus	Seurataan loginmoduulin debuggaus lauseita. Integraatiotestaus	Loginmoduulin debuggaus lauseet ilmoittavat poistumisesta	Loginmoduulin debuggaus lauseet ilmoittivat poistumisesta
Demosovelluksen, palvelinrajapinnan ja JAAS rajapinnan yhteistoimivuus	Tarkistetaan, että täyttäkö ohjelmisto vaaditut vaatimukset. Järjestelmätestaus	Ohjelmisto toimii suunnittelun mukaan	Ohjelmisto toimi suunnittelun mukaisesti

8 YHTEENVETO

Insinöörityön tavoitteena oli tehdä autentikointirajapinta mobiili asiakas/palvelinjarjestelmään. Asiakkaalla sekä palvelimella piti olla eri autentikointirajapinnat. Näihin tavoitteisiin päästiin hyvin. Asiakas sekä palvelinautentikointirajapinnat ovat valmiina ja toimivat suunnittelun mukaan.

Autentikointirajapinnan toteutuksessa käytettiin MIDP-, J2SE-ohjelmointia ja JAAS-rajapintaa. Sisään kirjautuvan asiakkaan tietoja verrataan palvelimella sijaitsevan MySQL-tietokannassa oleviin tietoihin. Palvelin myöntää tai hylkää kirjautumisen tilanteen mukaan.

Työ oli haastava, koska uutta asiaa tuli paljon. Javaan, MIDP:iin, NetBeansiin, JAAS:iin ja MySQL:iin tutustuminen sekä sille ohjelmien tekeminen tuotti töitä. Työssä välittyi Java-ohjelmoinnista hyvä kuva. Insinöörityö valmistui aikataulussa, mutta asiakasrajapinnan toteutus myöhästyi viikon. Työssä esiintyneet ongelmat hidastivat työn edistymistä. Asiakasrajapinnassa mobiililaitteen näkymän vaihto oli ongelmallinen, mutta se saatiin toimimaan. Palvelinrajapinta toimi moitteettomasti, kun käännettiin ohjelmisto tavallisella Java (javac) kääntäjällä. NetBeans kehitysympäristössä JDBC-ajuri ei toiminut, vaikka lisättiin sen tarvitsema kirjasto. Palvelinrajapinta tuotti ongelmia, kun ohjelmisto kuului tiettyyn pakkaukseen. Järjestelmän tietoturvaominaisuutta, joka käynnisti palvelinrajapinnan, ei pystytty asentamaan.

Osa vaatimuksista muuttui hieman välillä, mutta tästä ei aiheutunut haittaa, koska varsinaista asiakasta ei ollut. Insinöörityön tuloksena saadut rajapinnat ovat esimerkki KajaPro:lle, jonka pohjalta rajapintaa voidaan kehittää eteenpäin.

JAAS-rajapinta antaa hyvän mahdollisuuden jatkokehitykseen, koska JAAS:llä voidaan tehdä autorisointirajapinta, missä annetaan käyttäjille oikeuksia. Palvelimen tietokannalle voidaan kehittää moduuleja, jotka lisäävät, poistavat tai muokkaavat asiakastietoja. Lisäksi autentikointi on myös mahdollista toteuttaa MAC-osoitteen kautta, kuten Nokian uusilla kommunikaattoreilla.

LÄHDELUETTELO

1 Ei tekijää. Ei päiväystä. Tietoverkot ja niiden rakenne, asiakas-palvelin-käsite. [WWW-dokumentti]. <http://www.pagina.fi/esim/00975/>. (Luettu 15.3.2006).

2 Marttinen L. Asiakas/palvelin-mallin kuva. Tietoliikenne kursin luentomateriaali. [PDF-dokumentti]. <http://www.cs.helsinki.fi/u/marttine/tili/tIk04/luennot/ch1bw2.pdf> (Luettu 18.3.2006).

3 Ei tekijää. Ei päiväystä. Kuinka Internet/WWW toimii? [WWW-dokumentti]. <http://www.htk.fi/kirjasto/inetopas/net_o.htm>. (Luettu 15.3.2006).

4 Wikipedia. Viimeksi muutettu 16.3.2006. Thin Client. [WWW-dokumentti]. <http://en.wikipedia.org/wiki/Thin_client>

5 Seppänen V. Kevät 2005. Asiakas/palvelin-arkkitehtuuri ja tietoverkot. Tietoverkot kursin luentomateriaali <http://www.mit.jyu.fi/wikstrom/opetus/itk115/luennot/itk115-kalvot-1.pdf> [PDF-dokumentti].

6 Sommerville I. Software Engineering. Kuudes painos. Addison Wesley, 2001. 693 s. ISBN 0-201-39815-X

7 Wikipedia. Viimeksi muutettu 13.3.2006. Client-server. [WWW-dokumentti]. <http://en.wikipedia.org/wiki/Client_server>

8 Seppänen V. Johdatusta asiakas-palvelin-malliin. Tietoverkot kursin luentomateriaali [PPT-dokumentti].

9 Elmashri, Navathe, Fundamentals of Database Systems, Addison Wesley

10 Seppänen V. Syksy 2002. Asiakas/palvelin-arkkitehtuuri ja tietoverkot. Tietoverkot kursin luentomateriaali [PDF-dokumentti]. <http://www.mit.jyu.fi/wikstrom/opetus/itk115/luennot/itk115-14_17_4slides.pdf>

11 Wikipedia. Viimeksi muutettu 9.11.2005. Autentikointi. [WWW-dokumentti]. <http://fi.wikipedia.org/wiki/Autentikointi>

12 Laiho R. 22.2.2001. Hajautetun järjestelmän tietoturva. Helsingin yliopiston tietojenkäsittelytieteen laitoksen seminaari. [PDF-dokumentti]. <http://myy.helia.fi/~ritvalai/220201.pdf>

13 Fraser, B. (editor). Site Security Handbook. RFC 2196, Syyskuu 1997

14 Stallings W. Cryptography and Network Security: Principles and Practice. Second edition. Prentice-Hall, 1998

15 Seppälä J. 12.4.2005. Automaation Tietoturva. Tietoverkkopohjainen automaatio kursin luentomateriaali. [PDF-dokumentti]. <http://www.ac.tut.fi/aci/courses/7601000/pdf/Averkot_tietoturva.pdf>

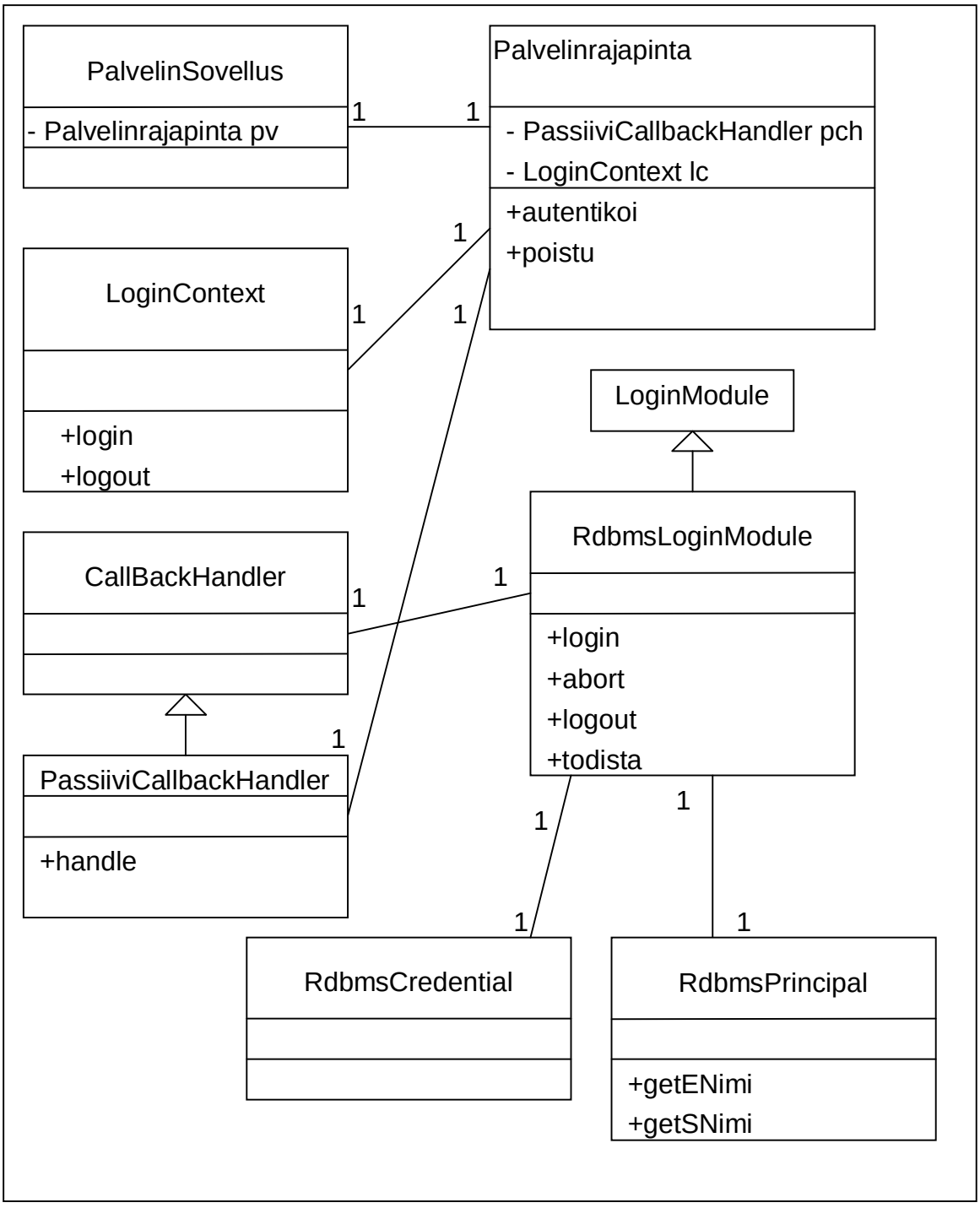
16 Sohlman T. 2.4.2003. Käyttäjän autentikointimekanismit. Mediatekniikan seminaari

17 Search Security. Viimeksi muutettu 19.7.2004. Two factor authentication. [WWW-dokumentti]. <http://searchsecurity.techtarget.com/sDefinition/0,290660,-sid14_gci992919,00.html>

18 Koistinen P. Viimeksi muutettu 13.12.2001. Javan Tietoturvaominaisuudet. Joensuun yliopiston laudaturseminaari. [WWW-dokumentti]. <http://www.cs.joensuu.fi/~pkoistin/laudaturseminaari_pk.html>

19 Musser J, Feuer P. Viimeksi muutettu 13.9.2002. All that JAAS, scalable Java security with JAAS. [WWW-dokumentti]. <http://www.javaworld.com/javaworld/jw-09-2002/jw-0913-jaas_p.html>

20 Määttä S. Älypuhelin tietoturvallisena päätelaitteena. Syksy 2005. Kajaanin ammattikorkeakoulun opinnäytetyö.


PalvelinSovellus
- Palvelinrajapinta pv
Palvelinrajapinta
- PassiiviCallbackHandler pch
- LoginContext lc
+autentikoi
+poistu
LoginContext
+login
+logout
LoginModule
RdbmsLoginModule
+login
+abort
+logout
+todista
CallBackHandler
PassiiviCallbackHandler
+handle
RdbmsCredential
RdbmsPrincipal
+getENimi
+getSNimi
1