INCA: Um Serviço de Segurança Para Ambientes de Ensino Colaborativos

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INCA: Um Serviço de Segurança Para Ambientes de Ensino Colaborativos

Thiago de Medeiros Gualberto

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do Título de Mestre em Ciência da Computação.

Orientador: **Prof. Dr. Sérgio Donizetti Zorzo**

SÃO CARLOS
2009

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

G899is	Gualberto, Thiago de Medeiros. INCA : um serviço de segurança para ambientes de ensino colaborativos / Thiago de Medeiros Gualberto. -- São Carlos : UFSCar, 2010. 88 f. Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2009. 1. Computadores - medidas de segurança. 2. e-Learning. 3. Serviços da web. 4. Plataforma MOODLE. 5. Plataforma SAKAI. I. Título. CDD: 005.8 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

"INCA: Um Serviço de Segurança para Ambientes de Ensino Colaborativo"

THIAGO DE MEDEIROS GUALBERTO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Sérgio Donizetti Zorzo
(Orientador - DC/UFSCar)

Prof. Dr. Hélio Crestana Guardia
(DC/UFSCar)

Prof. Dr. Volnys Borges Bernal
(POLI/USP)

São Carlos
Agosto/2009

Que este trabalho seja uma oferta agradável ao coração de Deus, que por sua misericórdia me permitiu realizá-lo.

“Aos cansados ele dá novas forças e enche de energia os fracos. Até os jovens se cansam, e os moços tropeçam e caem; mas os que confiam no Senhor recebem sempre novas forças. Voam nas alturas como águias, correm e não perdem as forças, andam e não se cansam. (Is 40, 29-31)”

Agradecimentos

Primeiramente, a Deus, pelo dom da vida, pela graça de todos os dias encontrar motivos para amá-Lo mais em todas as coisas e, a cada descoberta, poder amá-Lo mais que todas as coisas por meio das quais Ele se revela.

Aos meus pais, por terem me dado a riqueza de conhecimentos mais valiosos do que o melhor dos resultados que esta dissertação pudesse mostrar.

À minha namorada Larissa, pela vivência do amor verdadeiro em Deus que a cada dia descobrimos mais e mais profundo em nós.

A UFSCar, pela grande oportunidade e aprendizado.

Ao meu professor orientador, Dr. Sergio Donizetti Zorzo, pela amizade e confiança adquiridas durante esses anos de convivência e por todo o apoio para a realização deste trabalho. Muito obrigado por ter acreditado em mim.

Aos meus amigos do Grupo de Oração Universitário e da República São Lucas. Obrigado pela amizade e por serem minha família e sustento espiritual em São Carlos durante todo este tempo. O nome de cada um de vocês não cabe aqui, mas o meu coração é grato pela vida de vocês.

Aos funcionários do DC/UFSCar, por todo o auxílio e amizade. Em especial, Maria Cristina Carreira Trevelin, Jorgina Vera de Moraes (Tia Verinha) e Evélton Cardoso de Marco.

Aos amigos do Laboratório GSDR, pelos conselhos, conversas e discussões sobre este trabalho. A ajuda de cada um de vocês foi muito importante. Obrigado, também, pelo companheirismo e amizade. Passamos grandes momentos juntos que levarei por toda a minha vida.

Aos amigos do Laboratório LINCE, por toda a ajuda neste trabalho. Muito obrigado a todos vocês!

A todos os amigos do Departamento de Computação. Muito obrigado pela convivência e pelo crescimento que me proporcionaram. A todos os professores, funcionários e alunos do Departamento de Computação da UFSCar, pela saudável convivência e por todo o apoio dado durante a realização deste trabalho.

Ao meu primo Wálace Érick de Medeiros Moura, por todos os conselhos e ensinamentos. Aprendi muito durante nossas conversas e discussões sobre o meu trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro.

Em fim, agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Resumo

Pesquisas no domínio de *e-Learning* enfocam, principalmente, a maneira como o conteúdo de aprendizado é fornecido aos usuários do sistema, sem considerar os requisitos de segurança na maioria das implementações. Entretanto, as ameaças à segurança não podem ser negligenciadas, principalmente, porque aplicações de *e-Learning* são construídas sobre arquiteturas heterogêneas, distribuídas, abertas e que podem ser vulneráveis à ação de pessoas mal-intencionadas. Este trabalho apresenta INCA (**I**ntegridade, **N**ão Repúdio, **C**onfidencialidade e **A**utenticidade), um serviço que oferece segurança por meio da tecnologia de *Web Services* e que se propõe a atender os requisitos de segurança de ambientes de *e-Learning* desenvolvidos em diferentes arquiteturas e linguagens de programação. A utilização de *Web Services* para oferecer segurança nos ambientes de *e-Learning* complementa as funcionalidades de tais ambientes, isto é, não se limita às plataformas em que o sistema cliente foi desenvolvido. INCA utiliza criptografia e assinatura digital para oferecer integridade, não repúdio, confidencialidade e autenticidade. Por meio dos testes realizados pôde-se constatar que o INCA oferece segurança a ambientes de *e-Learning* de maneira flexível, configurável, escalável e de forma eficiente. As funcionalidades do INCA são apresentadas neste trabalho, com sua empregabilidade em um estudo de caso para a ferramenta *upload* de arquivo, dos ambientes Sakai e Moodle, que foram modificados para que pudessem usar os serviços do INCA. A contribuição deste trabalho é a realização do estudo de caso para os ambientes supracitados, uma vez que tal ferramenta nesses sistemas não possui serviços de segurança. Algumas modificações, em tais ambientes, foram necessárias para avaliar a aplicabilidade do INCA em ambientes diferentes. O estudo de caso evidencia a prova de conceito para o serviço de segurança apresentado neste trabalho.

Palavras-Chave: *e-Learning*, *Web Service*, Serviço de Segurança, Sakai, Moodle.

Abstract

Research efforts on *e-Learning* systems have mainly focused on the provision of learning contents for users, but these efforts have not considered security requirements in the implementation of such systems. However, threats to security may not be neglected, as these applications are usually built using heterogeneous, distributed and open architectures. This feature may make these systems vulnerable to ill-intentioned users. This work presents the INCA (Integrity, Non Repudiation, Confidentiality and Authenticity), a service that provides security through the technology of Web Services and aims at meeting the security requirements of e-Learning environments developed in different architectures and programming languages. The use of *web services* to provide security in *e-Learning* environments will complement the features of such environments, not being limited to platforms in which the client system was developed. INCA uses cryptography and digital signature to provide integrity, non-repudiation, confidentiality and authenticity. Through the tests it could be seen that INCA provides security for e-learning environments in a flexible, configurable, scalable and efficient way. The main features of INCA are presented in this work, with their employability in a case study for the *upload* tool for Sakai and Moodle environments, which were modified so that they could make use of INCA. The contribuition of this work is the case study implementation for the above mentioned environments, since such tool in these systems does not have security services. Such modifications were necessary to evaluate the applicability of the INCA in different environments. The case study shows the proof of concept for the security service presented in this work.

Keywords: *e-Learning*, *Web Service*, Security Service, Sakai, Moodle.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AC	Attribute Certificate
AES	Advanced Encryption Standard
CDAC	Central de Distribuição e Armazenamento de Chaves
CORBA	Common Object Request Broker Architecture
DAC	Discretionary Access Control
DCOM	Distributed Component Object Model
DES	Data Encryption Standard
DRM	Digital rights management
DSA	Digital Signature Algorithm
EaD	Educação a Distância
ECDSA	Elliptic Curve Digital Signature Algorithm
EJBCA	Enterprise JavaBeans Certificate Authority
FCS	Floor Control Security
GPL	General Public License
GPRS	General Packet Radio Service
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
ICP	Infraestrutura de Chave Pública
IDL	Interface Definition Language
INCA	Integridade, Não repúdio, Confidencialidade e Autenticidade
ITU-T	International Telecommunication Union Telecommunication
JADE	Java Agent Development Framework
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JPA	Java Persistence API
KDC	Key Distribution Center
LCR	Lista de Certificados Revogados
LMS	Learning Management Systems
MAC	Mandatory Access Control
MINE	Multimedia Information Network
Moodle	Modular Object-Oriented Dynamic Learning Environment
NIST	National Institute of Standards and Technology
OID	Objeto de Identificação
P3P	Platform for Privacy Preferences Project
PDF	Portable Document Format
PGP	Pretty Good Privacy
PHP	Hypertext Preprocessor
PKI	Public Key Infrastructure
PP	Política de Privacidade
PS	Política de Segurança
RBAC	Role Based Access Control

RMI	Remote Method Invocation
RSA	Rivest Shamir Adelman
RSS	Rich Site Summary
SGA	Sistema de Gestão de Aprendizagem
SMTP	Simple Mail Transport Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
UAB	Universidade Aberta do Brasil
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator
W3S	W3Schools
WSDL	*Web Services* Description Language
XML	Extensible Marckup Language

1 Introdução

Instituições de ensino investem de forma significativa na implantação de ambientes de *e-Learning*. Os benefícios deste processo de ensino-aprendizagem justificam os esforços das organizações para atuarem na Educação a Distância (EaD).

Os ambientes de *e-Learning* são utilizados por várias pessoas. Assim, a segurança torna-se uma exigência fundamental, uma vez que estes sistemas estão sujeitos a ataques (JOSHI et al., 2001b).

No domínio de *e-Learning*, as pesquisas enfocam, principalmente, a maneira como o conteúdo de aprendizado é fornecido aos usuários do sistema, sem considerar – na maioria das implementações – os requisitos de segurança (WEBBER et al., 2006). Porém, as ameaças à segurança não podem ser negligenciadas, principalmente, porque aplicações *e-Learning* são construídas sobre arquiteturas heterogêneas, distribuídas, abertas e podem ser vulneráveis a acessos de pessoas mal-intencionadas.

A segurança no aprendizado eletrônico tem como papel fornecer uma sessão fim-a-fim segura entre o aluno e o ambiente de *e-Learning* (RAITMAN et al., 2005). Aspectos importantes como autenticação, não-repúdio, confidencialidade e integridade dos dados, entre outros, devem ser considerados no sistema.

Um dos motivos que levam um ambiente de *e-Learning* a exigir mecanismos de segurança é a garantia da privacidade do aluno quanto ao seu desempenho acadêmico, de maneira a restringir o acesso a pessoas autorizadas (GRAF, 2002) (GELBORG, 2003). Para um aluno, pode ser importante que somente ele e o professor tenham acesso às suas notas. Além disso, espera-se a garantia de que não seja afetada a integridade de um documento ou mensagem enviada por um aluno ao professor.

Impedir que os alunos fraudem avaliações, não permitir que os serviços fiquem indisponíveis por razões de ataques maliciosos de terceiros, e o fato de que o conteúdo de aprendizagem é proprietário são outros motivos que justificam a utilização de mecanismos de segurança.

A confiança nos ambientes de *e-Learning* é um pré-requisito à sua aceitação por parte dos usuários. Assim, o uso da segurança no aprendizado eletrônico é importante, uma vez que todo o sistema está sujeito a sofrer novas ameaças (RABUZIM et al, 2006). Dessa maneira, percebe-se a necessidade de proteger ambientes de *e-Learning* não apenas com relação a fraudes, mas também com relação à integridade do conteúdo em si.

Analisando a necessidade de segurança em ambientes de *e-Learning*, o trabalho aqui apresentado tem como objetivo oferecer segurança para aplicações desses sistemas.

Observada a existência de diversos ambientes de *e-Learning* implementados em linguagens de programação e arquiteturas diferentes, como, por exemplo, Moodle (MOODLE, 2008), Sakai (SAKAI, 2008), entre outros, apresenta-se aqui o INCA, um serviço que oferece segurança por meio da tecnologia de *Web Services*. A tecnologia de *Web Services* é uma escolha apropriada pelas suas características de independência de plataforma e heterogeneidade entre linguagens (BOOTH et al., 2004). As características de *Web Services* evitam que um serviço de segurança implementado para ser utilizado em uma aplicação precise ser reimplementado ou reestruturado para ser utilizado em outra aplicação. Assim, o INCA pode ser utilizado por ambientes de *e-Learning* independentemente da linguagem e da arquitetura em que estes foram desenvolvidos.

1.1 Motivação

Os domínios de *e-Learning* têm enfocado, principalmente, a maneira como o conteúdo de aprendizado é distribuído e/ou processado, sem considerar as questões de segurança (SAXEMA, 2004) (WEBBER et al., 2007) (RAITMAN et al., 2005). Por isso, o trabalho aqui apresentado contribui para um maior interesse nas pesquisas em relação à segurança nos ambientes de *e-Learning*.

A realização deste trabalho é fundamentada e adaptada da afirmação de Webber et al. (2007), de que a segurança, juntamente com os requisitos do sistema, é um dos aspectos mais importantes que devem ser levados em consideração para suportar a ampla implantação de aplicações de *e-Learning*.

A utilização da tecnologia de *Web Services* justifica-se por duas de suas características: independência de plataforma e heterogeneidade entre linguagens.

A construção do serviço apresentado se fundamentou sobre a necessidade de abordagens que oferecessem segurança para ambientes de ensino colaborativo de maneira flexível, configurável e escalável (GUALBERTO; ZORZO, 2009).

Alguns trabalhos presentes na literatura, como os apresentados por Furnell et al. (1998) e Kambourakis et al. (2007), têm foco na implantação de segurança dentro do ambiente de *e-Learning*, o que pode influenciar na diversificação de sistemas de aprendizado eletrônico existentes, como Sakai, Moodle, AulaNet, entre outros (SAKAI, 2008) (MOODLE, 2008) (AULANET, 2009).

Este trabalho consiste no fornecimento de segurança para ambientes de *e-Learning* em geral, sem levar em consideração a arquitetura ou linguagem de programação utilizada em seu desenvolvimento. Assim, este trabalho não se limita à implantação de segurança no núcleo de um ambiente de *e-Learning*.

1.2 Objetivo

Os principais objetivos do trabalho aqui apresentado são:

- Oferecer segurança a ambientes de ensino colaborativo: implementar técnicas capazes de oferecer segurança nas aplicações de ambiente de *e-Learning*.

- Garantir que a eficiência do ambiente de *e-Learning* seja mantida com o emprego do INCA: garantir que o INCA possa ser executado nas diversas aplicações de ambientes de ensino colaborativo, com atraso nos tempos de resposta que não inviabilize a aplicação.

- Garantir a interoperabilidade na utilização do INCA em ambientes de *e-Learning*: garantir que ambientes de *e-Learning* desenvolvidos sobre linguagens e arquiteturas diferentes possam utilizar o INCA.

- Garantir a flexibilidade na utilização do INCA em ambientes de *e-Learning*: garantir que diferentes serviços e ferramentas de um sistema de aprendizado eletrônico tenham liberdade de uso do INCA, com possibilidade de que somente algumas ferramentas usem o serviço. Por exemplo, uma ferramenta de envio de arquivo está utilizando o INCA enquanto as demais ferramentas e serviços – como *chat*, fórum, *whiteboard*, entre outros – não o utilizam.

1.3 Organização do trabalho

Este trabalho possui 7 capítulos, distribuídos da seguinte maneira:

O capítulo 2 traz a contextualização do que é segurança e apresenta a tecnologia de *Web Service*. Nele são discutidos os serviços de segurança existentes, segundo os padrões internacionais, e os possíveis mecanismos de segurança utilizados para oferecer tais serviços. Também apresenta o conceito de *Web Service* e os padrões utilizados por ele.

No capítulo 3, são apresentados os trabalhos relacionados à proposta desta dissertação, ressaltando as estratégias e abordagens adotadas por cada autor para oferecer segurança aos ambientes de *e-Learning*;

O capítulo 4 descreve o trabalho desenvolvido. Expõe os detalhes do serviço de segurança INCA, os da ICP (Infraestrutura de Chave Pública) e CDAC (Central de Distribuição e Armazenamento de Chaves) utilizados e implementados, respectivamente, e os detalhes de implementação. Ao final, exibe uma comparação dos trabalhos existentes na literatura, a partir dos quais o trabalho aqui apresentado foi desenvolvido, detalhando as comparações e contribuições desta proposta para a academia.

O capítulo 5 descreve os testes realizados que permitiram a validação das operações realizadas pelo serviço. Além disso, detalha as execuções dos testes de carga realizados com a finalidade de avaliar a quantidade de requisições que o serviço pode suportar.

O capítulo 6 descreve o estudo de caso realizado nos ambientes de ensino colaborativo Sakai e Moodle, com o objetivo de validar o trabalho desenvolvido.

Por fim, o capítulo 7 apresenta as conclusões e as propostas para trabalhos futuros.

2 Segurança e Mecanismos de Implementação por *Web Services* em Aplicações de Sistemas de *e-Learning*

Este capítulo descreve a arquitetura básica de *Web Services*, bem como os principais padrões e protocolos envolvidos. Também são apresentados, na seção 2.2, serviços de segurança e mecanismos que possam ser utilizados para que esses serviços possam ser adquiridos.

2.1 A Tecnologia de *Web Services*

Web Services, segundo o W3C, são definidos como "Um sistema de *software* projetado para suportar interação máquina-a-máquina através de uma rede. *Web Services* têm uma interface descrita em um formato processável por máquina (especificamente WSDL). Outros sistemas interagem com um *Web Service* da maneira prescrita por sua descrição usando mensagens SOAP, tipicamente transportadas usando HTTP (*Hypertext Transfer Protocol*) com serialização XML (*Extensible Markup Language*), juntamente com os outros padrões *Web*".

A tecnologia de *Web Services* permite a comunicação entre computadores, tornando acessíveis os dados e serviços, isto é, fazendo com que aplicações possam interoperar, seja através de uma rede interna, seja por meio de padrões *Web* (ARRUDA et al., 2003). *Web Services* foram projetados e construídos com base em padrões abertos, o que torna possível que as funcionalidades de um *Web Service* desenvolvido sejam exportadas para outras aplicações (ARSANJANI et al, 2003). Assim, essa tecnologia pode ser conceituada, de maneira simplificada, como uma arquitetura para a distribuição de serviços, que permite a interoperabilidade entre aplicações e na qual seus componentes são independentes de plataforma (FULLER et al., 2003).

A aplicabilidade de *Web Service* pode ser realizada em vários cenários de integração de aplicações, tais como compartilhamento de dados, comércio em larga escala através da *internet* (FREMANTLE et al., 2002), além de cenários de computação móvel e de grades computacionais.

A utilização dos *Web Services* pode ser vista de duas formas. A primeira é a do ponto de vista de negócio, quando o seu uso pode diminuir os custos operacionais – devido ao

emprego de uma infraestrutura já existente –, melhorar a eficiência operacional (através da comunicação entre processos) e a flexibilidade organizacional – por meio da exposição de serviços. A segunda é a do ponto de vista tecnológico, e, sob esta ótica, os *Web Services* podem trazer a redução da complexidade – por meio do encapsulamento de interfaces e componentes –, o prolongamento da utilização de aplicações legadas e o provimento de uma interoperabilidade a baixo custo, quando as aplicações a serem integradas são heterogêneas ou, ainda, quando a infraestrutura não pode ser modificada.

Dentre as principais características dos *Web Services*, duas se destacam (ARRUDA, 2003): o fraco acoplamento e a interoperabilidade. A primeira significa que o consumidor de um serviço não precisa conhecer qualquer estrutura ou rotina interna para fazer uso dele. Ele precisa apenas ter acesso à interface, que é o meio pelo qual o provedor informa as características do serviço. Dessa maneira, é necessário que o serviço tenha uma descrição possível de ser entendida pelos computadores e que ela seja utilizada para identificar a localização dos serviços e a maneira como estes podem ser processados.

Outra importante característica fornecida pelos *Web Services* é a interoperabilidade, que se dá pelo fato de sua comunicação ser baseada em XML (BRAY, 2004). Dessa maneira, uma aplicação desenvolvida, por exemplo, na linguagem PHP, pode acessar serviços implementados em Java.

Atualmente, existem tecnologias que possibilitam o acesso remoto a serviços distribuídos (STAL, 2002). Contudo, tecnologias como, por exemplo, Corba (Common Object Request Broker Architecture) (CORBA, 2004) e Java RMI (MICROSYSTEMS, 2006), apresentam baixa escalabilidade, baixa capacidade de interoperabilidade e a necessidade de um alto poder de comunicação. Ao contrário de tais tecnologias, os *Web Services* apresentam algumas vantagens, tais como simplicidade, flexibilidade e interoperabilidade, com um baixo custo de processamento, porque utilizam mecanismos como linguagem XML para representação, padronização de suas informações (KLEIJNEN; RAJU, 2003) e o protocolo HTTP (FIELDING et. al., 1999) para transporte.

Assim, devido à existência de sistemas de aprendizado eletrônico desenvolvidos em linguagens e arquiteturas diferentes, os *Web Services* podem ser utilizados, uma vez que oferecem a independência de plataforma e de linguagens.

Arquitetura dos *Web Services*

A arquitetura de utilização de *Web Services* é composta pelos elementos ilustrados na Figura 1 (GOTTSCHALK et al., 2002; KREGER, 2001; ROY; RAMANUJAN, 2001): o Provedor de Serviços, o Cliente (Consumidor de Serviços) e o Registro de Serviços.

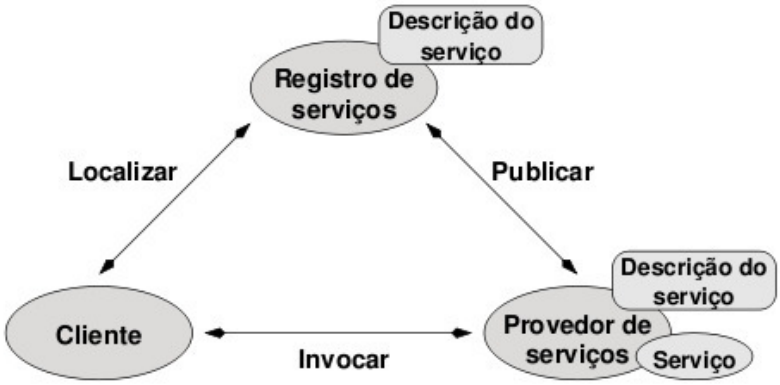

Figura 1 - Arquitetura de Web Services.

O Provedor de Serviços é responsável pela criação de um *Web Service*, cuja tarefa é descrever cada serviço criado e disponibilizar ou publicar tais descrições. Por meio dessas descrições, é possível garantir que um *Web Service* possa ser encontrado e compreendido, pela utilização de mecanismos de busca. Os serviços podem ser acessados pelos clientes, que são os consumidores do serviço. Assim, é importante salientar que, a partir da descrição disponibilizada pelo provedor, o consumidor pode obter as informações necessárias para utilizar um serviço. Além disso, o provedor pode publicar um serviço junto a um Registro de Serviços. Dessa forma, o consumidor pode pesquisar junto ao registro de serviços para descobrir e localizar serviços apropriados para a sua aplicação.

A arquitetura de um *Web Service* pode ser representada por uma pilha de protocolos *Web* que possui quatro camadas principais, conforme a ilustração dada pela Figura 2 (CERAMI, 2002).

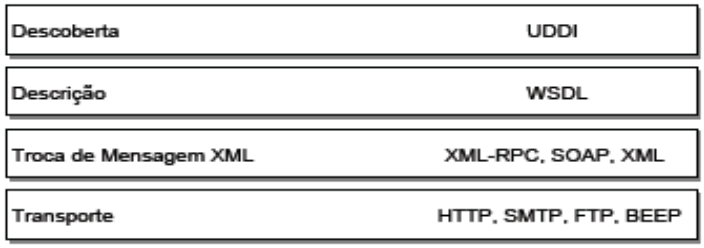

Figura 2 - Pilha de Protocolos Web Service.

A primeira camada é a de transporte, responsável por transportar as mensagens entre as aplicações. A segunda é a troca de mensagem XML, que tem como função codificar as mensagens em um formato XML comum. A terceira camada descreve a interface pública de um *Web Service*. Por fim, a quarta camada é responsável por centralizar os serviços em um registro comum e fornecer funcionalidades de publicação e busca.

Tecnologias para *Web Services*

Os *Web Services* utilizam três tecnologias específicas: Protocolo Simples de Acesso a Objeto (*Simple Object Access Protocol* - SOAP), Linguagem de Descrição de *Web Services* (*Web Services Description Language* - WSDL) e Descrição, Descoberta e Integração Universal (*Universal Description, Discovery and Integration* - UDDI). O SOAP fornece um mecanismo para a comunicação entre os *Web Services* e as aplicações clientes. WSDL é uma linguagem usada para descrever *Web Services*. UDDI permite que os *Web Services* guardem suas características em um registro que pode ser pesquisado por aplicações em busca de serviços. A Figura 3, adaptada de Erl (2004), ilustra os relacionamentos entre as três tecnologias supracitadas.

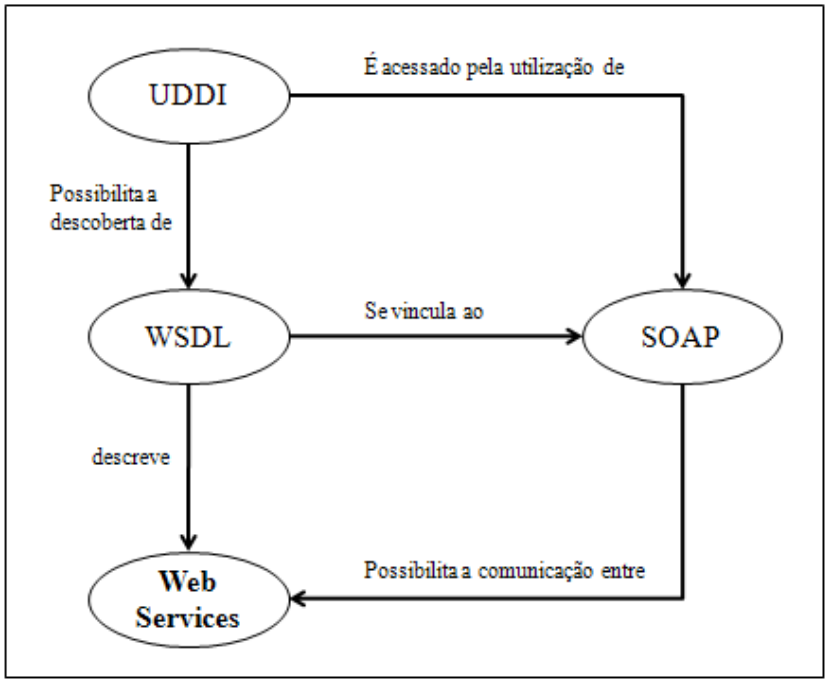

Figura 3 - Relacionamentos entre as tecnologias de Web Services.

O provedor de serviço pode fazer a publicação de seus serviços junto ao UDDI. O serviço de registro UDDI gerencia as informações sobre provedores e as implementações

de seus serviços. Tanto o provedor de serviços quanto os clientes podem acessar o UDDI por meio do protocolo SOAP, com o objetivo de publicar e descobrir, respectivamente, os serviços que lhe interessam. O acesso ao UDDI por parte dos clientes possibilita a descoberta do arquivo WSDL com a descrição do serviço publicado. Por meio desse arquivo, pode-se criar o cliente do serviço publicado no UDDI e acessá-lo pela utilização do protocolo SOAP, possibilitando, assim, a comunicação entre o consumidor e provedor de serviços. Nas próximas subseções, serão descritas estas três tecnologias de forma mais detalhada.

Protocolo SOAP

SOAP é um protocolo simples e leve, utilizado para a troca de dados no formato XML na *Web*. Tipicamente, ele é usado pelas aplicações clientes (consumidores de serviço) para invocar um serviço na *Web*. O SOAP permite o transporte de mensagens XML através de protocolos de alto nível, tais como HTTP, SMTP (*Simple Mail Transport Protocol*), dentre outros, sendo que o HTTP tem sido o mais utilizado. Assim, o SOAP está se tornando, rapidamente, o padrão para a troca de mensagens baseadas em XML (CURBERA et al., 2002).

Existem também outras linguagens que provêm funcionalidades parecidas com as do SOAP, como Java RMI (*Remote Method Invocation*), DCOM (*Distributed Component Object Model*) e CORBA (HENNING, 2008), porém, tal protocolo é escrito em XML, o que lhe permite ser independente de plataforma e linguagem (CERAMI, 2002). Uma mensagem SOAP é um documento XML e sua estrutura é composta por quatro elementos, os quais estão ilustrados pela Figura 4 (W3Schools -W3S) (MITRA; LAFON, 2007).

```
<soap:Envelope xmlns:soap="..." soap:encodingStyle="..."
   <soap:Header>
   ...
   </soap:Header>
   <soap:Body>
      ...
      <soap:Fault>
         ...
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Figura 4 - Estrutura de uma mensagem SOAP.

O primeiro elemento, o *Envelope*, identifica a mensagem SOAP contendo o cabeçalho e o corpo da mensagem. O segundo, o *Header* (cabeçalho), é um elemento opcio-

nal, que contém informações para segurança, roteamento ou as demais necessárias para a correta manipulação da mensagem. O terceiro, *Body* (corpo), é um elemento obrigatório, que tem os dados em XML incluídos na mensagem a ser transmitida; possui informações de chamada e de resposta ao servidor. Por fim, o elemento *Fault*, localizado dentro do elemento *Body*, traz informações dos erros ocorridos no envio da mensagem; este elemento somente está presente nas mensagens de resposta do servidor.

As informações necessárias para invocar um serviço ou refletir os resultados de uma requisição a um serviço são representadas na mensagem SOAP. O exemplo da Figura 5 ilustra um método que recebe dois números como argumentos de entrada e retorna a soma entre eles.

```
...
public int Soma (int num1, int num2){...}
...
```

Figura 5 - Exemplo do método Soma da interface de um serviço.

Na Figura 6, é ilustrada a mensagem SOAP de requisição ao serviço especificado na Figura 5. Informações como *namespace* (linha 2), estilo de codificação (linha 3), nome da interface (linha 5), método acessado (linha 6) e os argumentos passados para o método (linhas 7 e 8) estão representadas na Figura 6. Como resposta a essa mensagem de requisição, o provedor do serviço deverá retornar outra mensagem SOAP, ilustrada pela Figura 7 com o resultado da soma entre os dois números.

```
1 <soap:Envelope
2    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4    <soap:Body>
5       <c:calculadora xmlns:c="http://www.schemas.exemplo/calculadora/">
6          <c:Soma>
7             <c:arg1> 2 </c:arg1>
8             <c:arg2> 2 </c:arg2>
9          <c:Soma>
10      </c:calculadora>
11   </soap:Body>
12 </soap:Envelope>
```

Figura 6 - Exemplo da mensagem SOAP requisitando o método Soma.

```
1 <soap:Envelope
2    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4    <soap:Body>
5         <c:SomaResponse xmlns:c="Some-URI">
6              <c:return> 4 </c:return>
7         </c:SomaResponse>
8    </soap:Body>
9 </soap:Envelope>
```

Figura 7 - Exemplo da mensagem SOAP com a resposta da requisição ao método Soma.

SOAP é um protocolo mais complexo que o XML-RPC, mas, pela sua utilização, é possível que aplicações se comuniquem de forma simples e completamente independente da linguagem de programação e plataforma. Além disso, o SOAP suporta uma variedade de protocolos para transporte de suas mensagens, como HTTP, SMTP, dentre outros.

Linguagem para de Descrição de Interfaces (WSDL)

A linguagem WSDL descreve *Web Services* de maneira estruturada, através de um documento XML. O documento XML contém todas as informações necessárias para que possíveis clientes possam utilizá-lo de forma automatizada. Esta tecnologia é fornecida pelo provedor de *Web Services* e permite especificá-los de maneira padronizada e formal. Tal descrição inclui detalhes, como definição de operações suportadas pelo serviço, formato de mensagens de entrada e saída, tipos de dados, protocolo de ligação (*binding*), endereço de rede etc.

O WSDL tem uma função similar à do IDL (*Interface Definition Language*) do CORBA ou à implementação de uma interface remota em Java RMI (GOTTSCHALK et al., 2002). Enquanto o SOAP especifica a comunicação entre um cliente e um servidor, o WSDL descreve os serviços oferecidos.

Considerando que um *Web Service* tenha sido implementado na linguagem de programação Java e que se deseja criar um cliente para acessar tal serviço, pode-se gerar o código do cliente, com o arquivo WSDL, em qualquer linguagem de programação, desde que esta suporte a tecnologia de *Web Services*. Dessa maneira, é possível a comunicação entre arquiteturas diferentes (BOOTH; LIU, 2006). A Figura 8 ilustra a estrutura de um documento WSDL.

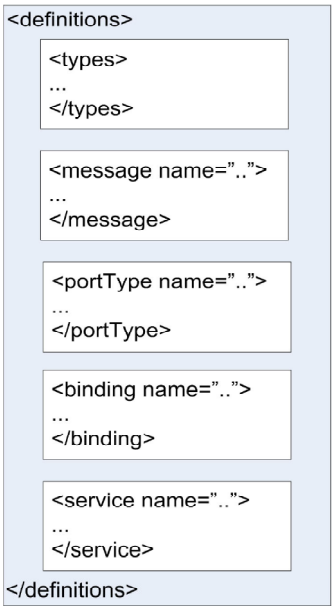

Figura 8 - estrutura de um documento WSDL.

Segundo Weerawarana et al. (2005), um documento WSDL possui um elemento raiz chamado `<definitions>`, que contém uma parte abstrata e outra concreta. A parte abstrata descreve o que o *Web Service* faz, em termos de mensagens criadas e processadas por ele. Os elementos `<types>`, `<message>` e `<portType>` são responsáveis por tal descrição. O elemento `<types>` define todos os tipos de dados utilizados para descrever as mensagens trocadas entre o cliente e o servidor. O elemento `<message>` representa uma definição abstrata dos dados que estão sendo transmitidos – ele é responsável por descrever uma mensagem unidirecional, seja ela uma mensagem de requisição ou uma resposta. Este elemento possui um atributo chamado *name*, que define o nome da mensagem e, além disso, pode conter zero ou mais elementos filhos do tipo *part*, que declaram os argumentos da função. O elemento `<part>` contém o nome e o tipo do argumento. O terceiro elemento, `<portType>` define um *Web Service*, as operações que podem ser realizadas por ele e as mensagens que estão envolvidas. Este elemento combina vários elementos `<messages>` para formar uma operação unidirecional ou bidirecional completa. Há também a possibilidade de combinação de uma mensagem de requisição e uma mensagem de resposta dentro de uma única operação requisição/resposta.

Já a parte concreta define “como” e “onde” acessar o serviço. Os elementos `<binding>` e `<service>` são os responsáveis por essa definição. O elemento `<binding>`, descreve o protocolo de transporte a ser utilizado para cada mensagem definida no elemento `<portType>`. Além disso, este elemento realiza descrições concretas sobre como

o serviço será implementado na rede. Por fim, o elemento <service> define o nome e o endereço para invocar cada uma das operações do serviço especificado, isto é, contém informações da localização do serviço.

Os elementos supracitados descrevem um *Web Service* (BOOTH; LIU, 2006). A seguir, são apresentados alguns exemplos de trechos de código WSDL, que demonstram as definições vistas anteriormente. Na Figura 9, tem-se a ilustração do elemento <portType>, no qual é definida a operação "somar" – tendo a mensagem "somarRequest" como entrada e produzindo a mensagem "somarResponse" como saída. Desta maneira, definimos a operação resquest/response:

```
<wsdl:portType name="ICalculadora">
   <wsdl:operation name="somar" parameterOrder="in0 in1">
      <wsdl:input message="impl:somarRequest" name="somarRequest" />
      <wsdl:output message="impl:somarResponse" name="somarResponse" />
   </wsdl:operation>
</wsdl:portType>
```

Figura 9 - Exemplo do elemento portType.

Na Figura 10, exemplifica-se o elemento <message>, onde são especificados os dados que serão transmitidos entre o cliente e o servidor. Os parâmetros transmitidos para o serviço e a resposta que ele retorna estão presentes no elemento <part>, encontrado dentro do elemento <message>.

```
<wsdl:message name="somarRequest">
   <wsdl:part name="in0" type="xsd:int" />
   <wsdl:part name="in1" type="xsd:int" />
</wsdl:message>
<wsdl:message name="somarResponse">
   <wsdl:part name="somarReturn" type="xsd:int" />
</wsdl:message>
```

Figura 10 - Exemplo do elemento message.

A Figura 11 ilustra o elemento <binding>, que associa o elemento <portType> ao protocolo SOAP através de um elemento de extensão SOAP chamado <wsdlsoap:binding>. Dois parâmetros são fornecidos por este elemento: o protocolo de transporte e o estilo da requisição – que, para o exemplo apresentado, é "rpc".

```
<wsdl:binding name="CalculadoraSoapBinding" type="impl:ICalculadora">
   <wsdlsoap:binding style="rpc"
                     transport="http://schemas.xmlsoap.org/soap/http" />
</wsdl:binding>
```

Figura 11 - Exemplo do elemento binding.

A localização real do serviço é definida nos elementos <service> e <port>, ilustrados na Figura 12. Este último está presente dentro do elemento <port>. Assim, um serviço pode conter várias portas e cada uma delas é específica para o tipo de ligação que foi descrito no elemento <binding>.

```
<wsdl:service name="ICalculadoraService">
   <wsdl:port binding="impl:CalculadoraSoapBinding" name="Calculadora">
      <wsdlsoap:address
      location="http://127.0.0.1:8080/axis/services/Calculadora" />
   </wsdl:port>
</wsdl:service>
```

Figura 12 - Exemplo do elemento service.

Devido à utilização de apenas tipos primitivos (int), não houve a geração do elemento <type> no documento WSDL.

De posse dos elementos supracitados, têm-se informações suficientes para descrever o modo pelo qual um cliente deve invocar e interagir com o *Web Service*. Contudo, as implementações por trás dos *Web Services* podem ser qualquer coisa, desde que o remetente e o receptor tenham a mesma descrição do serviço, isto é, o mesmo arquivo WSDL (BOOTH et al., 2004).

Registro para a Publicação e Localização de Serviços (UDDI)

O UDDI é uma especificação técnica que tem o objetivo de encontrar, acrescentar e alterar *Web Services* de forma rápida e fácil (CLEMENT, 2004).

O registro UDDI define um esquema sob o qual os provedores podem descrever metadados a respeito de seus *Web Services*. Ele pode ser utilizado tanto por um provedor de serviço, para publicar seus serviços oferecidos, como pelos clientes – para descobrir os

serviços que estão disponíveis e obter as informações necessárias para utilizá-los. Resumidamente, o UDDI é uma espécie de lista de *Web Services* públicos fornecidos.

Os dados contidos em um UDDI dividem-se em três categorias: a *White Pages*, que inclui informações básicas sobre uma organização, como informação de contato, endereço, além de outros tipos de informações; a *Yellow Pages*, em que são incluídos dados gerais de classificação da organização ou do serviço oferecido – nessa categoria, as informações são organizadas pela natureza específica do produto ou por regiões geográficas; e a *Green Pages*, que inclui informações técnicas sobre um *Web Service*, como um endereço para a invocação do serviço e um ponteiro para uma especificação externa. Em poucas palavras, essa categoria explica como fazer a comunicação com os serviços.

O UDDI é implementado como um *Web Service*, que utiliza SOAP para a troca de mensagens, e a interação com o UDDI é feita através de um conjunto pré-definido de interfaces SOAP. A Figura 13 elucida a estrutura do registro UDDI.

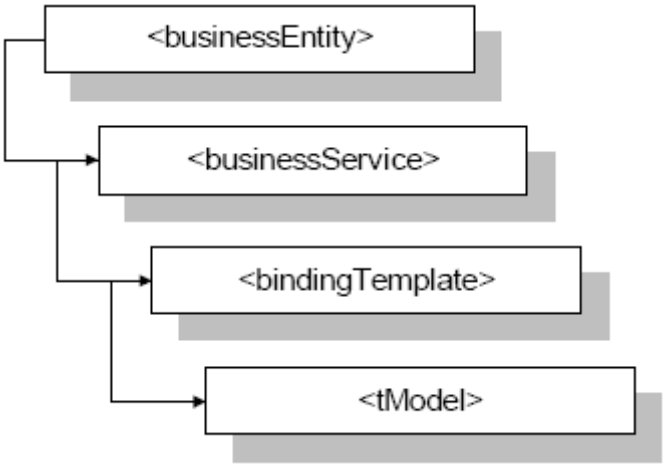

Figura 13 - Estrutura do UDDI.

O primeiro elemento da estrutura do UDDI é o `<businessEntity>`, que possui informações sobre a organização que publica o *Web Service*. Essas informações podem ser o nome da organização e os dados de contato, e podem incluir os serviços oferecidos e suas respectivas URLs. Este elemento contém um ou mais `<businessService>`.

O segundo elemento, `<businessService>`, possui a descrição de cada serviço oferecido. Informações como nome do serviço estão presentes nessa estrutura. Este elemento contém um ou mais `<bindingTemplate>`.

O elemento `<bindingTemplate>` possui informações descritivas sobre o modo pelo qual invocar um serviço. Este elemento referencia um ou mais `<tModel>`.

O elemento <tModel> possui informações sobre especificações técnicas de um serviço como, por exemplo, localização, interface, tipo de serviço, dentre outros. Podem ser referenciados por vários <bindingTemplate>.

Através desta estrutura UDDI, é possível definir a maneira de publicar e descobrir as informações sobre os *Web Services* de forma fácil, rápida e dinâmica, por meio de uma conexão com a *internet* para que outros clientes também possam utilizar a mesma estrutura. Entre outras coisas, o UDDI também permite que registros operacionais sejam mantidos por diferentes propósitos em diferentes contextos (CLEMENT, 2004).

2.2 Segurança

No aprendizado eletrônico, a necessidade de segurança das informações e dos usuários é essencial. Um bom mecanismo de autenticação, de autorização, de armazenamento seguro de provas, de garantia de o estudante ser quem realmente afirma, entre outras, são fundamentais para a garantia de segurança das aplicações em um ambiente de aprendizado eletrônico.

Nesta seção, serão apresentados alguns serviços de segurança que podem ser alcançados pelo uso de mecanismo de segurança. Utilizados adequadamente, estes serviços podem garantir a proteção necessária para o ambiente em que são aplicados.

Serviços de Segurança

O ITU-T (*International Telecommunication Union - Telecommunication*) define serviço de segurança como um serviço fornecido por uma camada de comunicação de sistemas abertos, que garante a segurança adequada dos sistemas e dos dados trocados (ITU-T, 1991).

Na recomendação X.800 do ITU-T, os serviços de segurança são divididos em cinco categorias: confidencialidade e integridade dos dados, não repúdio, autenticação e controle de acesso. Além dos serviços de segurança, também são apresentados mecanismos que implementam tais serviços e que serão discutidos nesta subseção. Além disso, devido a essa recomendação ser desenvolvida como padrão internacional, diversas organizações e indivíduos desenvolveram suas aplicações de segurança baseando-se nessa padronização.

Os serviços de segurança descritos a seguir são serviços básicos. Na prática, serviços e mecanismos de segurança podem ser combinados de forma a satisfazer à política de

segurança e/ou às exigências do usuário ou do sistema. Os mecanismos de segurança particulares podem ser usados para implementar combinações dos serviços básicos de segurança (STALLINGS, 2007).

Confidencialidade dos Dados

A confidencialidade é a proteção da informação contra o acesso por indivíduos, entidades ou processos sem autorização (MENEZES, 1997). Em relação ao conteúdo de uma transmissão de dados, podem ser identificados vários níveis de proteção. O serviço mais amplo protege todos os dados transmitidos entre dois usuários durante um período de tempo (ISO, 1989).

Também podem ser definidas algumas formas restritas deste serviço, incluindo a proteção de uma única mensagem ou, até mesmo, de campos específicos dentro de uma mensagem (STALLINGS, 2006). Outro aspecto da confidencialidade é a proteção do fluxo de tráfego por meio do uso de chaves de sessão – o que exige que um atacante não seja capaz de observar a origem e o destino, frequência, duração ou outras características do tráfego em uma comunicação.

Tipicamente, a confidencialidade é alcançada com criptografia. Um esquema de encriptação proficiente impede que um adversário recupere e/ou apreenda informações parciais sobre uma mensagem encriptada (CHÁVEZ et al., 2005).

Integridade dos Dados

A integridade dos dados é a propriedade na qual os dados não são alterados sem autorização. É a garantia de que os dados recebidos são exatamente aqueles enviados por uma entidade autorizada, ou seja, não sofreu nenhuma operação que invalida sua integridade, como modificação, inserção, eliminação, ou repetição (BERTINO; SANDHU, 2005) (ISO, 1989). A integridade dos dados inclui a noção de que itens de dados estejam completos.

Alguns requisitos presentes na literatura para o controle da integridade incluem (BYUN et al., 2006) (SANDHU, 1993): o controle de fluxo de informação, que previne a contaminação (ou influência) de dados com maior integridade por dados com menor integridade; a verificação de dados, que garante que os dados verificados sejam fornecidos apenas para determinadas operações; a prevenção da fraude e erro, necessária para garantir que apenas os dados legítimos sejam introduzidos para os sistemas de informação; por fim, o requisito de validação de dados autônoma, que tenta manter e/ou melhorar a integridade (ou confiança) dos dados, independentemente do acesso a eles. Embora a seleção destes requisitos seja subje-

tiva, a aplicação de cada um deles sozinho não é suficiente para preservar a integridade dos dados.

Da mesma forma que a confidencialidade, a integridade pode ser aplicada a um fluxo de mensagens, a uma única mensagem, ou a campos selecionados dentro de uma mensagem (STALLINGS, 2006).

Um serviço de integridade orientado à conexão, que trata de um fluxo de mensagens, assegura que estas são recebidas no mesmo estado em que foram enviadas, sem duplicação, inserção, modificação, reordenação ou repetições. Por outro lado, um serviço de integridade sem conexão, que trata de mensagem individual sem levar em conta qualquer contexto maior, fornece proteção, na maioria das vezes, somente contra modificação de mensagem.

O serviço de integridade dos dados também pode ser oferecido com e sem recuperação (CHÁVEZ et al., 2005). Devido à possibilidade de o serviço de integridade estar relacionado a ataques ativos, a preocupação recai sobre a detecção ao invés da prevenção. Se uma violação de integridade é detectada, o serviço pode simplesmente informá-la; posteriormente, para fazer a recuperação a partir da violação, é exigida alguma outra parte do *software* ou uma intervenção humana.

Não Repúdio

O não repúdio é uma das questões de segurança fundamentais existentes em ambientes eletrônicos (ONIEVA et al., 2008). Este serviço impede que tanto o emissor quanto o receptor neguem a emissão/recebimento de uma mensagem transmitida. Assim, alguns exemplos incluem: não repúdio de origem, destino e submissão. O primeiro deles opera quando uma mensagem é enviada e o receptor pode provar a autenticidade do remetente, ou seja, que o remetente da mensagem foi quem realmente a enviou. O segundo, não repúdio de destino, atua quando uma mensagem é recebida e o remetente pode provar que o receptor efetivamente recebeu a mensagem e que sua integridade não foi comprometida (MENEZES, 1997) – este exemplo é o complemento do não repúdio de origem (ZHOU; GOLLMANN, 1997). O terceiro e último exemplo, não repúdio de submissão, garante para o autor de uma mensagem ou documento eletrônico que seus dados forram submetidos ao agente de entrega para a distribuição.

É possível encontrar várias situações nas quais é necessário ou se pode aplicar o serviço de não repúdio: compras pela *internet*, assinatura em contratos, *e-mails* certificados ou, mais geralmente, em qualquer troca realizada por meio de redes digitais. Por exemplo, uma entidade pode autorizar a compra de determinado produto por outra entidade e depois

negar que tal autorização foi concedida. Um processo envolvendo uma terceira parte confiável é necessário para resolver a disputa. Desta maneira, pode-se perceber que o não repúdio deve ser verificável por um terceiro.

O não repúdio pode ser fornecido pelo uso de assinaturas digitais – tema que será explanado mais detalhadamente ainda neste capítulo.

Autenticação

O serviço de autenticação está preocupado em garantir a autenticidade das pessoas ou organizações envolvidas na comunicação e na autoria de documentos eletrônicos. Sua função é garantir ao destinatário que a origem da mensagem ou documento eletrônico é verdadeira, assim como a identidade do remetente (MENEZES, 1997).

No caso de uma interação contínua, como a conexão de um terminal a um *host*, dois aspectos estão envolvidos: primeiro, ao iniciar uma conexão, o serviço garante que as duas entidades são autênticas, ou seja, cada entidade é realmente quem afirma ser; segundo, o serviço deve garantir que a conexão não será interferida de modo que um terceiro não se passe por uma das partes com finalidade de transmissão ou recepção não autorizada.

A autenticação é um serviço que atua sobre a identificação, aplicando-se a ambas as entidades e à própria informação. Assim, o aspecto da criptografia relacionado à legitimidade é normalmente dividido em duas grandes classes: autenticação da entidade e autenticação de origem dos dados. A primeira delas fornece uma confirmação da identificação de uma entidade (por exemplo, uma pessoa, um terminal de computador, um cartão de crédito etc). Este tipo de serviço de autenticação é fornecido para uso do estabelecimento ou durante a fase de transferência dos dados de uma conexão, para confirmar a identidade de uma ou mais entidades conectadas a uma ou mais diferentes entidades. O serviço de autenticação pode oferecer a confiança a uma entidade não confiável.

A segunda, a autenticação de origem dos dados, fornece uma confirmação da origem de uma unidade de dados. Este tipo de autenticação não oferece proteção contra a duplicação ou a alteração de unidades de dados, apenas suporta aplicações como correio eletrônico, onde não há nenhuma interação prévia entre as entidades comunicantes.

Controle de Acesso

O controle de acesso pode ser definido como a capacidade de limitar e controlar o acesso a qualquer tipo de recurso e aplicações de um sistema, fornecendo proteção contra a utilização não autorizada de recursos acessíveis (DZIERZAWSKI et al., 1999). Geral-

mente, é necessária a identificação do usuário para verificar quais são os seus direitos de acesso sobre os recursos do sistema.

Este serviço pode ser aplicado a vários tipos de acesso, como, por exemplo, o uso de um recurso de comunicações, a leitura, a escrita ou o apagamento de um recurso de informação, a execução de uma aplicação, entre outros (STALLINGS, 2006).

Os principais mecanismos de controle de acesso são: o Controle de Acesso Discricionário, o Controle de Acesso Obrigatório e o Controle de Acesso Baseado em Papéis (NCSC, 1985) (SANDHU; MUNAWER, 1998) (OSBORN et al., 2000) (SANDHU et al., 1996) (FERRAIOLO; KUHN, 1992).

O Controle de Acesso Discricionário, conhecido como DAC (*Discretionary Access Control*), define controles de acesso básicos para recursos de um sistema, isto é, formas de restrição do acesso a recursos baseadas na identidade dos indivíduos ou dos grupos aos quais eles pertencem. De modo geral, o dono do recurso decide quem está autorizado a acessar o recurso e qual privilégio terá sobre ele. O DAC tem sido visto como um mecanismo tecnicamente correto para as necessidades de segurança em setores comerciais e governamentais (FERRAIOLO; KUHN, 1992).

O Controle de Acesso Obrigatório, conhecido como MAC (*Mandatory Access Control*), é outra forma de controle de acesso. Neste mecanismo, quem determina as políticas de acesso é o sistema, e não o dono do recurso, como é feito no DAC. O MAC é utilizado nas áreas onde os dados são altamente sensíveis, como a área militar ou a do governo (FERRAIOLO; KUHN, 1992).

O Controle de Acesso Baseado em Papéis, conhecido como RBAC (*Role Based Access Control*), atribui direitos para papéis específicos, não para indivíduos, ou seja, o acesso a recursos de um determinado sistema é limitado de acordo com o papel do indivíduo dentro de uma organização.

O RBAC define três regras, a seguir: atribuição de papéis, autorização de papéis e autorização de transações. Na primeira regra, um indivíduo só pode executar uma operação desde que possua um papel adequado a ele atribuído. Na autorização de papéis, assim como na regra anterior, os usuários só possuem papéis aos quais foram autorizados. Por fim, a autorização de transações, juntamente com as duas regras anteriores, garante que um usuário possa executar somente as transações a que estiver autorizado.

O RBAC está chamando a atenção, particularmente, para aplicações comerciais, devido à sua potencialidade para reduzir a complexidade e os custos de administração de segurança em grandes aplicações em rede (BARKLEY et al., 1997). De acordo com Joshi et

al. (2001b), este mecanismo tem grande relevância para tratar os requisitos complexos de segurança em aplicações baseadas na *Web*. Além disso, o RBAC é também uma solução para fornecer aspectos de segurança em uma infraestrutura governamental digital multidomínio (JOSHI et al., 2001a).

Os tipos de serviços de segurança utilizados em aplicações de sistemas de *e-Learning* podem ser definidos diante das características e necessidades destes sistemas.

Mecanismos de Segurança

Os serviços de segurança apresentados na subseção anterior podem ser obtidos por meio de mecanismos de segurança que funcionam de forma independente ou em conjunto com outros mecanismos de segurança.

Dentre os mecanismos de segurança presentes na literatura, podem-se listar alguns que serão utilizados no escopo deste trabalho.

Criptografia

A criptografia é o ato de transformar uma informação em uma forma incompreensível sem uma nova transformação. Este processo ocorre por meio de um programa de computador que realiza operações matemáticas, de maneira a codificar a informação – transformando-a em um texto cifrado – e, depois, decodificá-la.

Existem dois tipos de criptografia: simétrica e assimétrica. Na criptografia simétrica, utiliza-se a mesma chave para codificar e decodificar a mensagem. A criptografia assimétrica, mais conhecida como criptografia de chave pública (*Public Key Infrastructure - PKI*), faz uso de um par de chaves distintas para codificar e decodificar uma mensagem. Uma delas é de domínio público, distribuída livremente para todos os correspondentes – via *e-mail* ou de outras formas; a outra, de domínio privado, é de conhecimento exclusivo de seu dono. A chave pública é obtida a partir da chave privada correspondente.

Na criptografia simétrica, a chave utilizada para codificar a mensagem é a mesma que a decodifica, isto é, sua segurança restringe-se apenas ao segredo da chave entre os usuários legítimos. Faz-se necessária, então, a utilização de um canal de comunicação efetivamente seguro entre o emissor e receptor, que seja imune a qualquer intervenção de terceiros. Uma forma de estabelecer um canal de comunicação realmente seguro é pelo uso do mecanismo de distribuição de chaves desenvolvido por Diffie-Hellman. Este mecanismo, ao con-

trário da *Key Distribution Center* (KDC), não utiliza uma chave comum para a distribuição segura das chaves simétricas.

O modelo simétrico é utilizado, principalmente, em aplicações onde existe somente um emissor e um receptor, como, por exemplo, um terminal bancário conectado à sua central de dados.

Um exemplo muito conhecido que oferece melhor entendimento é o de Alice e Bob (MENESES, 1997). Alice tem posse de uma mensagem e deseja enviá-la a Bob, de forma que só ele tenha acesso a ela. Com posse de uma chave simétrica, Alice cifra sua mensagem, a ponto de que seu conteúdo possa ser visto somente por quem tem posse da chave simétrica. Eve, um desconhecido, deseja ter acesso à mensagem, mas não o consegue por não possuir a chave simétrica. A mensagem é, então, enviada a Bob, de forma que somente quando ele tiver posse da chave secreta, utilizada por Alice para cifrar a mensagem, conseguirá decifrá-la. A Figura 14 ilustra o processo de codificação e decodificação de uma mensagem por meio da criptografia simétrica.

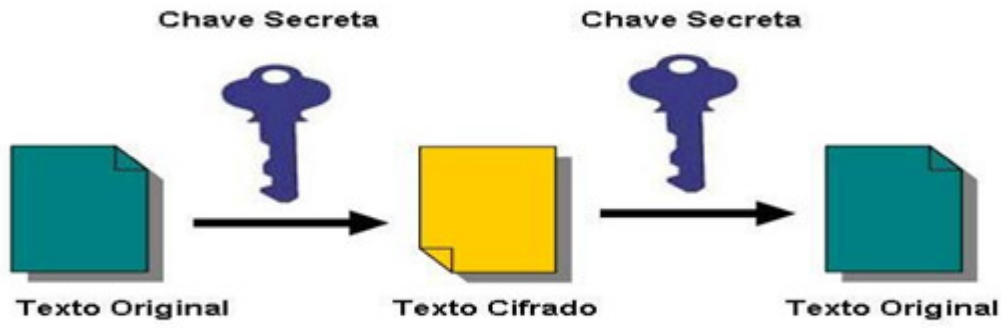

Figura 14 – Criptografia simétrica.

Um dos algoritmos simétricos mais populares é o AES – *Advanced Encryption Standard* – anunciado pelo NIST (*National Institute os Standards and Technology*) como um padrão de criptografia. Além do algoritmo de criptografia simétrica citado, existem outros, como Meneses(1997):

•**DES – *Data Encryptation Standard***: Desenvolvido pela IBM e adotado pelo governo americano em 1977, como padrão de codificação para o uso de seus órgãos governamentais não relacionados à segurança nacional.

•**Triple-DES**: Algoritmo considerado frágil, mas ainda usado. Após a criação do DES, desenvolveu-se um método (Triple-DES) para melhorar sua segurança. Tal objetivo é alcançado aplicando-se o DES três vezes, mas com duas chaves diferentes.

•**Madryga**: Desenvolvido em 1984, este algoritmo trabalha com blocos de 8 *bits*, e não utiliza transposições. Trata-se de um algoritmo considerado bem fraco, pois suas operações estão baseadas em ou-exclusivo e deslocamento de *bits*, e ele não aceita efeito avalanche, ou seja, o fato de a mudança de um *bit* no texto normal modificar metade dos *bits* no texto cifrado.

Em 1976, foi introduzido por Whitfield Diffie e Martin Hellman o conceito de criptossistemas de chave pública ou chave assimétrica (DIFFIE; HELLMAN, 1976). Aceitava-se como natural o fato de que as chaves de cifragem e decifragem tivessem de ser mantidas secretas, fossem elas iguais ou diferentes. Porém, um dos grandes problemas era, justamente, o da distribuição de chaves por canais inseguros, o que exigia uma grande quantidade de troca de dados.

Os modelos criptográficos assimétricos utilizam duas chaves distintas para codificar e decodificar uma mensagem. Uma delas é de domínio público, distribuída livremente para todos os correspondentes – via *e-mail* ou por meio de outras formas; e a outra, de domínio privado, é de conhecimento exclusivo de seu dono. O processo de codificação e decodificação através de criptografia assimétrica é ilustrado na Figura 15.

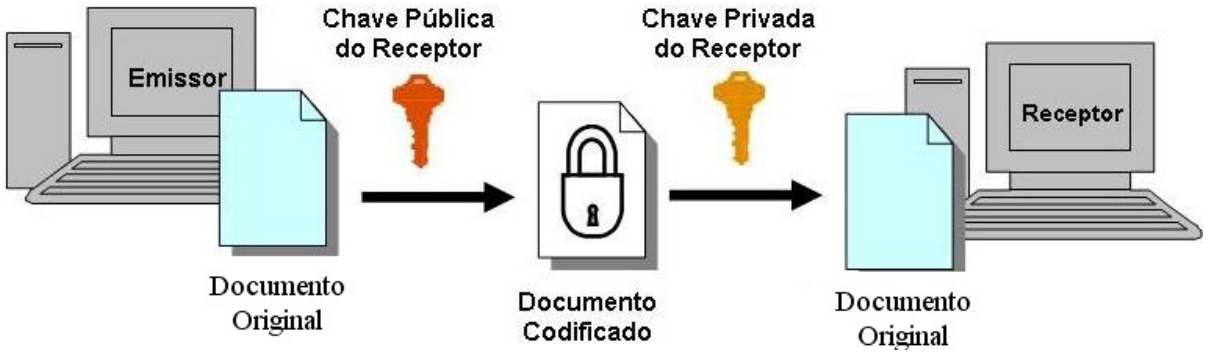

Figura 15 - Criptografia assimétrica.

Um emissor X que deseja enviar um documento e/ou mensagem codificada para um receptor Y precisa, primeiramente, adquirir a chave pública de Y para, então, codificar o documento e enviá-lo. O emissor Y, ao receber o documento, através de sua chave privada, consegue decodificar o documento codificado e obter o documento original novamente.

Assinatura Digital

O mecanismo de assinatura digital permite que um usuário com posse de sua chave privada possa assinar digitalmente uma mensagem ou documento. Assim, o destinatário

da mensagem só poderá verificar a assinatura realizada adquirindo a chave pública correspondente à chave privada que assinou a mensagem. Essa verificação pode ser realizada por meio de um *software* criptográfico, por exemplo. Com o uso desse mecanismo de segurança, pode-se garantir autenticidade, integridade e não repúdio de uma mensagem.

No processo para geração de uma assinatura digital, primeiramente, é calculado o valor *hash* da mensagem, também conhecido como resumo criptográfico, utilizando um algoritmo, como por exemplo, MD5, SHA1, SHA256, entre outros. O *hash* de uma mensagem é uma sequência de *bits* com um tamanho fixo. Os algoritmos utilizados para calcular o *hash* aplicam transformações matemáticas tais que, se apenas um simples *bit* da mensagem de entrada é alterado, obtém-se um valor de *hash* completamente diferente. Dessa maneira, garante-se a integridade da mensagem assinada digitalmente.

Após o calcular o valor do resumo criptográfico da mensagem, o *hash* é cifrado com a chave privada do emissor da mensagem. Com isso, tem-se a assinatura digital da mensagem. A autenticidade e não-repúdio da mensagem assinada digitalmente se baseiam no conhecimento da chave, isto é, o dono da chave privada é o autor da mensagem. Portanto, o sigilo da chave privada é de suma importância. Para isso, são utilizados algoritmos como RSA, DSA (*Digital Signature Algorithm*) e o ECDSA (*Elliptic Curve Digital Signature Algorithm*). A Figura 16 ilustra o processo de geração de uma assinatura digital.

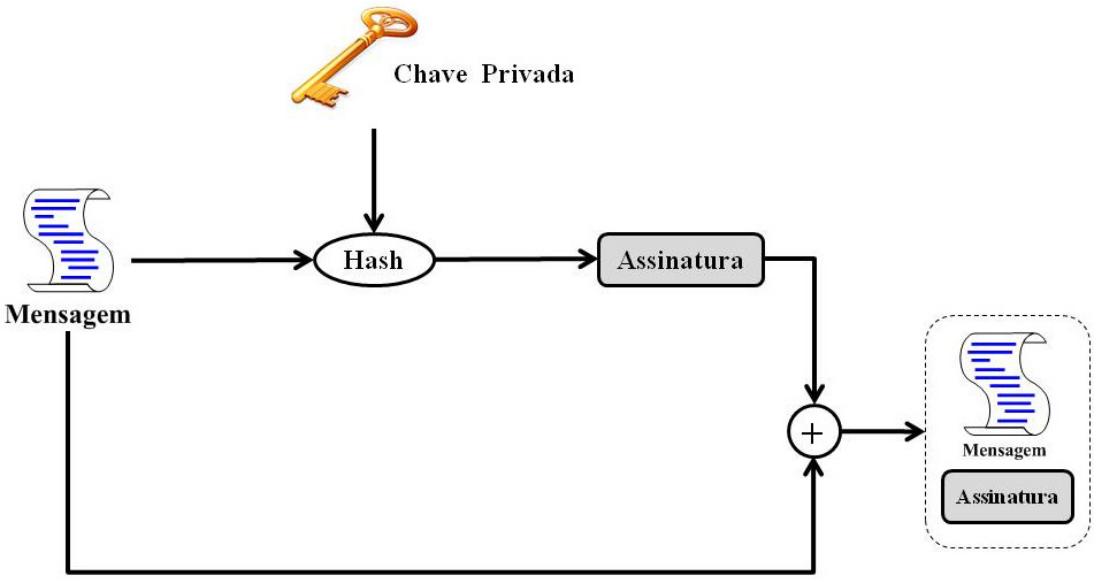

Figura 16 – Processo de geração de uma assinatura digital.

A tecnologia de assinatura digital permite ao receptor de uma mensagem verificar quem gerou a assinatura (autoria) e se a sua integridade não foi afetada. Pelo processo de

verificação de uma assinatura digital, é possível determinar se uma mensagem foi assinada com chave privada correspondente a uma dada chave pública. É necessário que se tenha um modo seguro de se obter a chave pública do emissor que realizou a assinatura digital. Assim, é possível verificar se uma dada mensagem foi realmente assinada pelo emissor.

Do ponto de vista técnico, na verificação de uma assinatura digital calcula-se, primeiramente, o *hash* da mensagem assinada. Para isso, é utilizado o mesmo algoritmo de *hash* utilizado durante o processo de geração da assinatura.

Depois, a assinatura digital é decifrada através da chave pública correspondente à chave privada utilizada para assinar a mensagem. Como resultado, é obtido o valor original do *hash* calculado a partir da mensagem original durante a criação da assinatura digital.

Finalmente, compara-se o valor corrente do *hash* obtido na primeira etapa da verificação com o valor original do *hash* obtido na segunda etapa. A verificação é bem sucedida se os dois valores forem idênticos. Assim, é possível afirmar que a mensagem foi assinada com a chave privada que corresponde à chave pública usada na verificação. Caso contrário, se os valores forem diferentes, a assinatura digital é inválida. A Figura 17 ilustra o processo de verificação da assinatura.

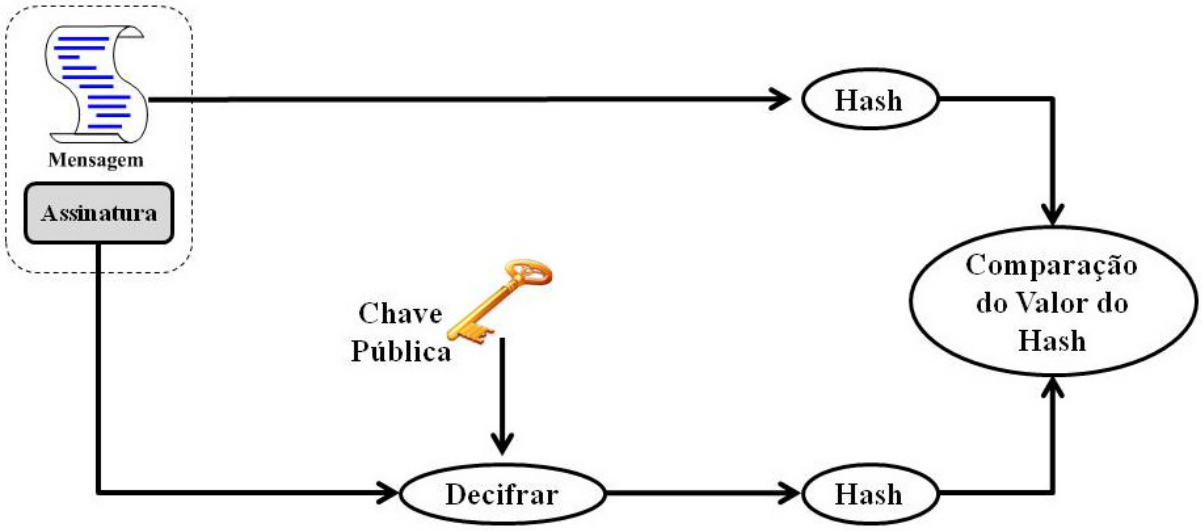

Figura 17 – Processo de verificação de uma assinatura digital.

Neste capítulo, foram explanados os serviços de segurança presentes em padrões internacionais e a tecnologia de *Web Services*, utilizada no escopo deste trabalho, para implementar tais serviços. Os serviços disponibilizados neste trabalho proporcionam a integridade e a confidencialidade dos dados e o não repúdio, e são fornecidos por mecanismos de segurança de assinatura digital e criptografia.

3 Trabalhos Relacionados

A utilização de tecnologias de segurança para satisfazer a necessidade de altos níveis de confidencialidade e privacidade em sistemas de *e-Learning* se deve às informações que são armazenadas nos ambientes de ensino colaborativos, como notas, provas, entre outras.

Apresentam-se, a seguir, alguns trabalhos relacionados com segurança em sistemas *e-Learning*, comparando-os com a proposta apresentada nesta dissertação e identificando as contribuições realizadas.

3.1 Personalização da Segurança para *Internet* e *Web Services*

Yee e Korba (2007) apresentaram uma abordagem para personalização da segurança que permite que um provedor de serviços na *Web* ou na *internet* e os clientes possam estabelecer uma negociação sobre um acordo de personalização da política de segurança (PS).

Uma PS é uma especificação de quais medidas de segurança serão usadas para proteger os serviços dos ataques à segurança. A utilização de PS é uma maneira flexível e eficaz de gerenciar a segurança nos serviços da *internet*.

O fornecedor de um serviço pode fazer uso de uma PS para especificar as medidas de segurança que serão administradas. Porém, esta PS pode não corresponder às preferências de segurança de um consumidor do serviço. A solução para este problema, segundo Yee e Korba (2007), foi permitir que o consumidor, durante a negociação, personalizasse sua PS com as medidas de segurança presentes na PS do fornecedor.

A abordagem de negociação da PS apresentada por Yee e Korba (2007) foi baseada em agentes não-autônomos que satisfazem os requisitos de negociação apresentados pelos autores. Para tanto, propuseram que a negociação da PS, iniciada após o consumidor escolher o serviço desejado, fosse o primeiro de dois estágios. O segundo estágio – a negociação da Política de Privacidade (PP) – é iniciado somente quando o primeiro tem êxito. Ou seja, enquanto o fornecedor e o consumidor do serviço não chegarem a um acordo quanto à PS, o estágio de negociação da PP não é iniciado. O estágio da negociação da PP passa pelo mesmo processo da negociação da PS. Contudo, a execução de um serviço ocorre somente quando a negociação da PP é concluída com sucesso.

O protótipo apresentado por Yee e Korba (2007) para a negociação de PS estende o utilizado para a negociação da política de privacidade apresentado em Yee e Korba (2003). Este protótipo é baseado em uma arquitetura *peer-to-peer*, programada em JADE (*Java Agent Development Framework*), e permite que o consumidor e o fornecedor se comuniquem entre si, pela *internet*, iniciando e continuando uma sessão de negociação.

Algumas pequenas modificações foram necessárias no protótipo apresentado em Yee e Korba (2003) para a negociação da PS. Primeiramente, foi preciso fornecer uma janela de ajuda para os consumidores que precisassem aprender sobre um serviço ou mecanismo de segurança em particular. Posteriormente, procedeu-se com o aperfeiçoamento do mecanismo de seleção de usuário, para permitir a seleção de escolhas múltiplas necessárias para alguns serviços de segurança – como autenticação – e para a negociação de conjuntos de mecanismos de segurança.

Com o propósito de ajudar os consumidores que não sabem qual serviço ou mecanismos de segurança escolher, Yee e Korba (2007) também propuseram um esquema de ajuda *online*, através do qual os fornecedores de serviço armazenam as PS's usadas com seus respectivos serviços. Uma Autoridade de Serviço (AS) coleta, periodicamente, as políticas de segurança juntamente com os tipos de serviços, data e nome do fornecedor para o qual ela foi aplicada. De posse destes dados, a AS constrói uma tabela usando os serviços e mecanismos de segurança de cada PS, juntamente com suas respectivas informações quanto à violação da segurança e seus impactos. Durante o processo de negociação, um consumidor que precise de ajuda para escolher um serviço/mecanismo de segurança, pode solicitar à AS o fornecedor do serviço, a pontuação dos serviços/mecanismos de segurança armazenados nela e os serviços e mecanismos de segurança correspondentes a este serviço. Como resultado dessa solicitação, a menor pontuação do produto da quantidade do número de violações de segurança e a média do impacto dessas violações correspondem ao serviço e mecanismo de segurança mais eficaz para o consumidor.

Yee e Korba (2007) listam alguns pontos fortes de sua abordagem, como: ser simples e fácil de usar, por meio de interfaces apropriadas; satisfazer as exigências de segurança pessoal de cada usuário e fornecer ajuda *online* para que os usuários façam suas escolhas durante as negociações. Entretanto, além do esquema de ajuda não poder ser escalável quanto ao número de usuários e, possivelmente, ser vulnerável a influências maliciosas, não há validação para um domínio específico.

3.2 Plataforma Multiagente para tratar questões de segurança em ambientes de *e-Learning*

No trabalho de Webber et al. (2007), são apresentados alguns resultados da intersecção de três áreas tecnológicas: *e-Learning*, Sistemas Multiagentes e padrões/normas para melhorar o desenvolvimento de sistemas seguros – com a proposta de implementação de uma plataforma multiagente, que consiste em uma infraestrutura distribuída para tratar questões de segurança. Essa plataforma também foi utilizada para o desenvolvimento de ambientes de *e-Learning*.

Os autores apresentam alguns requisitos de segurança de aplicações *e-Learning*, como controle de acesso (identificação, autenticação e autorização), focalizando três cenários: o do professor, o do processo de avaliação e o da colaboração entre os participantes.

O requisito de segurança relacionado ao primeiro cenário é o controle de acesso. Os professores podem gerenciar o ambiente e o aprendizado dos estudantes por meio de suas interações, de forma que possam ajustar as ferramentas de comunicação e os grupos de estudantes, monitorar suas atividades, realizar avaliações e oferecer ajuda. Diante desse contexto, é essencial que o ambiente de aprendizado identifique a real identidade dos estudantes.

Em relação ao segundo cenário, os autores destacam a necessidade de diferentes níveis de requisitos de segurança, como identificação, autenticidade, autorização, privacidade e confidencialidade, uma vez que o cenário de avaliação é um dos processos mais complexos de implementação e administração em um ambiente de *e-Learning*. Também são apresentados requisitos como integridade e armazenamento dos dados, de maneira a prevenir a interceptação, cópia ou modificação de mensagens e o acesso não autorizado dos dados armazenados, respectivamente.

A questão de segurança relacionada ao terceiro e último cenário, como a necessidade de registro e proteção da comunicação contra acessos não autorizados, é fundamental para a interação colaborativa entre os usuários, um importante aspecto do processo de aprendizado.

Analisando os três cenários, os autores identificaram algumas ameaças para aplicações de *e-Learning*, como acesso a dados não autorizados, acesso a dados não autorizados durante a codificação (criptografia), ausência de codificação, negação de serviço, ausência ou insuficiência de mecanismos de agentes de autenticação, instalações incorretas de *soft-*

ware, utilização de linguagens de programação inseguras, senhas inseguras, entre outros. Todas essas ameaças estão relacionadas com a integridade, a privacidade e o controle de acesso.

Os autores também afirmam que questões de segurança têm sido raramente consideradas. Elas não são encaradas como uma preocupação central, uma vez que aplicações de *e-Learning* são construídas sobre arquiteturas abertas, heterogêneas e distribuídas. Eles ainda afirmam que a segurança é um dos aspectos que mais devem ser levados em conta para a implantação de aplicações *e-Learning*.

3.3 Segurança em Ambientes de *e-Learning Online*

Raitman et al. (2005) analisaram a função da segurança em ambientes *e-Learning*, em particular, quanto aos aspectos sociais de segurança e a importância da identidade. Apresentaram um estudo de caso conduzido para testar o senso de segurança experimentado pelos estudantes enquanto usavam a plataforma *Wiki* como meio de colaboração *online* em um ambiente de educação superior. Fez-se uma comparação entre dois estudos nessa ferramenta, por meio de um grupo que utilizou um mecanismo de acesso por *login*/senha e outro que utilizou anonimato. Os autores escolheram essa ferramenta por ser um *site Web* completamente editável, de fácil acesso e que permite aos estudantes sentirem autoridade e controle no ambiente que estão utilizando.

No mecanismo de controle de acesso por meio de *login* do usuário, foram utilizados assinatura e *timestamp* com a finalidade de avaliar os estudantes, classificando-os quanto às suas ações. Essa característica mostrou-se interessante, uma vez que o estudante, ao visualizar a página – mesmo de forma não destacada – podia averiguar o *timestamp* para verificar quando a última alteração na *Wiki* havia sido realizada e reconhecer o usuário responsável por ela. Raitman et al. (2005) também perceberam que a preocupação quanto à exclusão e à edição de conteúdo da ferramenta *Wiki* foi bem destacada pelos estudantes, mas não ocorreu qualquer incidente que validasse suas atividades. O receio dos estudantes de perder algum trabalho ou duplicar suas entradas foi suficiente para convencê-los a acreditar que o ambiente *Wiki* era pouco confiável.

Quanto ao segundo estudo de caso, em que os usuários tinham total controle sobre suas ações, não foi realizada qualquer ação que prejudicasse a *Wiki* ou outro usuário. Os usuários poderiam tanto apagar os conteúdos da *Wiki* como utilizar linguagem imprópria – suas ações não teriam responsabilidade nem poderia haver acusações por alguma atitude. Mesmo assim, eles não se comportaram dessa maneira.

Raitman et al. (2005) mostraram a importância de estimular o senso de segurança em um ambiente de *e-Learning online*. Os alunos foram capazes de induzir integridade e confiança em suas pesquisas e colaborações. No entanto, para a própria segurança dos usuários, eles preferiram o *login* de usuário. Os usuários se sentiram muito mais seguros e confiantes em um ambiente que geralmente é inseguro, exposto e vulnerável.

3.4 Tecnologias de Segurança e Privacidade para Aplicações de Educação a Distância

Lin et al. (2004) apresentaram várias aplicações de Educação a Distância, desenvolvidas no laboratório *Multimedia Information Network* (MINE), e mostraram como a segurança e a privacidade podem ser integradas nessas aplicações. Eles explanaram como o desenvolvimento de ferramentas, combinado com segurança e privacidade, pode oferecer um ambiente de *e-Learning* seguro, eficiente e efetivo.

Em um primeiro momento, os autores expuseram o requisito de autenticação. Tanto para estudantes quanto para instrutores, a autenticação é importante para que um usuário seja identificado antes de utilizar o sistema.

A primeira solução possível para a autenticação do estudante foi feita através de *login* e senha. Se o objetivo é ter uma autenticação mais rigorosa, é necessário o uso de PKI e de assinaturas digitais. No entanto, devido à elevada quantidade de estudantes em um ambiente de Educação a Distância, o gerenciamento de chave que envolve uma PKI pode tornar cara essa abordagem. Uma das alternativas para essa situação é a utilização de PGP (*Pretty Good Privacy*). Outra possível solução é o uso do protocolo *Secure Socket Layer* (SSL), que oferece um canal seguro, de maneira que as trocas de informações não sejam facilmente interceptadas e entendidas.

Quanto à autenticação do instrutor, os autores consideram-na muito importante, devido a que os instrutores têm diferentes papéis e privilégios de acesso a diversas informações de um sistema de aprendizado, como material didático, informações e registro de desempenho dos estudantes. Ao contrário dos alunos, o número de instrutores é razoavelmente estável e baixo. Assim, as organizações podem fazer uso de abordagens de chave pública para a autenticação do instrutor.

Um segundo requisito apresentado por Lin et al. (2004) é a garantia da privacidade. Em um ambiente de EaD, alguns dados são confidenciais e desobrigados de revelação. Algumas aplicações definidas pelos autores precisam coletar informações de desempenho do

aprendizado dos estudantes enquanto navegam pelo sistema de EaD. A partir dessa coleta, as aplicações geram "tutoriais" pessoais para os estudantes. Porém, isto poderia violar a privacidade dos usuários.

Segundo as leis de privacidade (OECD, 1980), o operador de um sistema pode exigir um acordo com o estudante para coletar suas informações de desempenho de aprendizado. Contudo, uma solução apresentada pelos autores para a garantia da privacidade do estudante foi o uso de Política de Privacidade (PP). Uma PP conteria informações sobre quais dados seriam coletados, com que finalidade eles seriam usados, por quanto tempo seriam mantidos, se são ou não compartilhados com outras pessoas e como eles são protegidos. Uma maneira de expressar uma política é através de uma linguagem de descrição de política eletrônica, como por exemplo, *Plataform for Privacy Preferences* (P3P). Porém, o uso de P3P é bastante limitado, uma vez que só se declara o modo como funciona o serviço de aprendizagem ao invés de fazer ajustes para as preferências individuais dos estudantes. Uma abordagem de política mais flexível seria o uso de uma PP para cada estudante, assegurando quais dados privados ele deseja partilhar e sob quais condições. Outra maneira seria a negociação entre o fornecedor do sistema de Educação a Distância (EaD) – no caso, a universidade – e os estudantes, alcançando um acordo sobre a coleção de informações de desempenho de aprendizado.

Quanto à privacidade da comunicação, o estudante deveria ser capaz de fazer uma pergunta ao instrutor em particular. Neste caso, Lin et al. (2004) evidenciam a possibilidade de se usar SSL, já que este método cria um canal seguro entre o estudante e o instrutor. Assim, todos os fluxos de comunicação serão criptografados, garantindo a privacidade da rede.

Lin et al. (2004) também destacam a necessidade de integridade dos dados. Eles afirmam que, para se obter resultados mais confiáveis em exames ou testes, deve-se preservar a integridade dos dados. Na educação tradicional, os exames são feitos em uma sala de aula com a presença do professor, para que os estudantes sejam monitorados durante o exame. Na Educação a Distância, os exames são tratados de maneira diferente, já que não se tem a presença física de um professor. Neste caso, existe não só a possibilidade de um aluno fazer o exame do outro, como também a de "atacantes" poderem ter atitudes como ver, roubar e também modificar as respostas dos alunos durante a transmissão ou armazenamento dos exames.

Uma possível solução é o monitoramento das atividades dos estudantes, tais como postura e digitação, por meio do processamento de imagens de vídeos dos alunos, além do monitoramento do tempo e das características de erro de digitação, respectivamente. Esta

abordagem é considerada promissora por Lin et al. (2004), pois o monitoramento das características de digitação do estudante e sua comparação com os monitoramentos anteriores pode oferecer, ao longo do tempo, um meio útil de avaliar a possibilidade de um disfarce de estudante. Quanto à transmissão de exames, a proteção da privacidade e da integridade dos dados, pode-se obtê-las através do uso de SSL. Entretanto, o agente do exame deveria firmar as respostas da avaliação com a assinatura eletrônica do estudante e a assinatura do agente, garantindo a integridade fim-a-fim. As respostas dos exames podem ser protegidas, enquanto armazenadas, pelo uso de criptografia simétrica, por exemplo, AES ou Triple DES. No entanto, é necessário cuidado na distribuição e proteção das chaves. Neste caso, as chaves simétricas seriam protegidas com o uso de chave assimétrica, por exemplo, PGP ou PKI. Seria possível, ainda, proceder por meio de uma contínua verificação da identidade do estudante, com o uso da biometria.

Outro destaque se faz para a proteção de direitos autorais tanto em *courseware* – *software* para uso educacional – quanto em uma apresentação. A proteção dos direitos autorais do *courseware* é um assunto importante tanto na educação tradicional quanto na EaD. Porém, no mundo digital, a proteção dos direitos de imagem é mais difícil devido à facilidade de se fazer cópias de um material digital. Assim, um sistema de Educação a Distância deve proteger o *courseware* de cópias não autorizadas durante o armazenamento, a transmissão e a apresentação. Segundo Lin et al. (2004), a proteção durante o armazenamento e a transmissão pode ser alcançada pelo uso de encriptação. Quanto à proteção durante a apresentação, trata-se, segundo os autores, de um empreendimento bastante desafiador, já que um estudante pode, simplesmente, gravar as apresentações através da captura de tela, recortando e colando as ações ou, ainda, usando uma filmadora. Uma reação extrema para esta possível conduta poderia ser o monitoramento das atividades dos estudantes com uma câmara de vídeo. Porém, isto seria altamente invasivo e poderia limitar o curso *online*, destruindo algumas vantagens da EaD. Dessa maneira, uma condição-chave para o gerenciamento de direitos autorais é a proteção e contagem de material privado por direitos autorais. Os autores citam o uso do *Digital Rights Management* (DRM) para o gerenciamento de direitos digitais, que oferece contagem e proteção de conteúdo. O DRM pode ser empregado a fim de aplicar permissões associadas com o material da apresentação, isto é, uma permissão pode ser associada com a reprodução. Também seria possível declarar que o material só poderia ser visto por um determinado grupo de pessoas.

Por fim, Lin et al. (2004) explicitam a Segurança de Controle de Fluxo – *Floor Control Security* (FCS). O Controle de Fluxo – *Floor Control* – é um mecanismo pelo qual o

acesso a um objeto compartilhado é gerenciado, como, por exemplo, controlar o acesso a uma *Whiteboard* colaborativa (um único usuário, por vez, pode utilizar) ou determinar quem tem o direito de falar em uma sessão de *chat*. O *Floor Control* pode ser aplicado a qualquer permissão de acesso específico de um objeto, mas normalmente se refere ao direito de um usuário editar um objeto, enquanto todos os outros usuários somente são capazes de vê-lo. Alguns exemplos de *Floor Control* são: permitir que todos os usuários ao mesmo tempo tenham permissão para editar um objeto, ter um moderador que decide quem tem permissão de alterar o objeto ou a utilização de um *token* que é passado entre os usuários dando permissão para cada um deles alterar o objeto.

A segurança do controle de fluxo é necessária especificamente para atividades síncronas nos ambientes de EaD. Sem uma proteção adequada, as atividades síncronas de comunicação poderiam ser feitas por quem pretende bloquear a comunicação. Dessa maneira, a confiança dos usuários nas ferramentas de aprendizado pode ser afetada se tais ferramentas não funcionarem corretamente.

Os autores apresentaram uma situação em que o *token* com o direito de falar é passado para um aluno e existe a necessidade de garantir que o aluno corrente recebeu o *token*. Segundo eles, é possível solucionar este problema a partir da utilização de protocolos de segurança. Esses protocolos poderiam ser adaptados para garantir a transmissão correta dos *tokens*.

Lin et al. (2004) apresentaram diferentes aplicações *e-Learning* e mecanismos de segurança que poderiam ser usados em determinadas aplicações. Porém, não integraram as tecnologias de segurança e privacidade nessas aplicações.

3.5 Aplicação de PKI em ambientes de *e-Learning* e *m-Learning*

Kambourakis et al. (2007) afirmam que PKI e Certificados de Atributo (AC's) podem proporcionar um *framework* adequado para suportar serviços de autenticação e autorização de maneira eficaz, oferecendo confiança mútua tanto para os alunos quanto para os fornecedores de serviço.

Levando em consideração os requisitos de PKI para o aprendizado a distância *online*, Kambourakis et al. (2007) discutem a potencial aplicação de AC's em um modelo de confiança proposto. Foram apresentadas interações de confiança entre aprendizes e fornecedo-

res no aprendizado eletrônico, demonstrando que mecanismos de segurança robustos e efetivo controle de confiança podem ser obtidos e implementados.

Kambourakis et al. (2007) apresentaram uma arquitetura geral praticamente segura, isto é, um modelo de confiança capaz de suportar atividades de *e-Learning* e *m-Learning*. O esquema proposto utiliza chaves públicas e lógica RBAC para comunicações seguras, além de outros procedimentos entre todos os componentes ativos, enquanto oferece, ao mesmo tempo, máxima escalabilidade, flexibilidade e redução de custos administrativos, especialmente quanto ao aumento do número de participantes.

A viabilidade das soluções foi avaliada em termos de tempo de resposta de serviços nos sistemas de educação móvel, usando um dispositivo móvel de poucos recursos, como aparelho de teste e uma rede GPRS (*General Packet Radio Service*) com largura de banda limitada. Os resultados mostraram que a emissão de AC é viável em termos de tempo de serviço enquanto, simultaneamente, pode distribuir soluções flexíveis e escaláveis tanto para os aprendizes quanto para os provedores de *e-Learning*.

Embora a arquitetura proposta por Kambourakis et al. (2007) seja usada, principalmente, para prover autenticação, autorização, não repúdio de serviços de origem e integridade e confidencialidade das mensagens, outros elementos-chave podem ser efetivamente mantidos usando a mesma infraestrutura e modelo: a não manipulação de avaliação dos testes, a proteção do material do *courseware*, a distribuição segura do material de teste, entre outros.

Neste capítulo, foram apresentados alguns trabalhos encontrados na literatura, que foram utilizados para o desenvolvimento do trabalho aqui apresentado. Nos trabalhos citados nas seções anteriores existem algumas características semelhantes e também características com abordagens diferentes do serviço de segurança desenvolvido, que será explanado no próximo capítulo.

4 INCA: Um Serviço de Segurança para Ambientes de Ensino Colaborativo

Neste capítulo, é apresentado o serviço INCA (**I**ntegridade, **N**ão repúdio, **C**onfidencialidade e **A**utenticidade). Este serviço visa a oferecer segurança para ambientes de ensino colaborativos, independentemente da linguagem/arquitetura em que foram implementados. Também será apresentada a Infraestrutura de Chaves Públicas (ICP) e a Central de Distribuição e Armazenamento de Chaves (CDAC), utilizadas pelo INCA para adquirir as chaves públicas e as chaves simétricas, respectivamente.

4.1 Visão Geral

O INCA foi projetado para prover segurança em ambientes de *e-Learning*. Em sua implementação, procurou-se satisfazer alguns requisitos:

Eficiência: Mesmo com o uso de mecanismos de segurança em sua implementação, a utilização do INCA não deve gerar ou provocar atrasos nos tempos de resposta quando empregado pelas ferramentas ou serviços da aplicação de um ambiente de *e-Learning*.

Interoperabilidade: O INCA foi implementado por meio da tecnologia de *Web Services*, o que permite suporte a diferentes ambientes de *e-Learning*, uma vez que um *Web Service* é independente da arquitetura ou linguagem de programação em que o seu cliente foi desenvolvido.

Flexibilidade: É possível ter módulos do ambiente de *e-Learning* – como o serviço de envio de arquivo – utilizando o serviço, sem que os demais módulos sofram interferência.

Escalabilidade: Com o aumento dos usuários acessando os recursos das aplicações, é necessário que a eficiência do INCA seja mantida.

INCA é disponibilizado na *Web* e sua arquitetura geral é ilustrada pela Figura 18. Nesta arquitetura, um ambiente de ensino colaborativo possui um banco de dados e aplicações diversas. A criação das aplicações vai ao encontro das necessidades de uma organização, seja ela uma universidade, empresa, entre outras. Essas aplicações podem ser: um curso de direção defensiva, um curso de Inglês, um curso de computação, dentre outros.

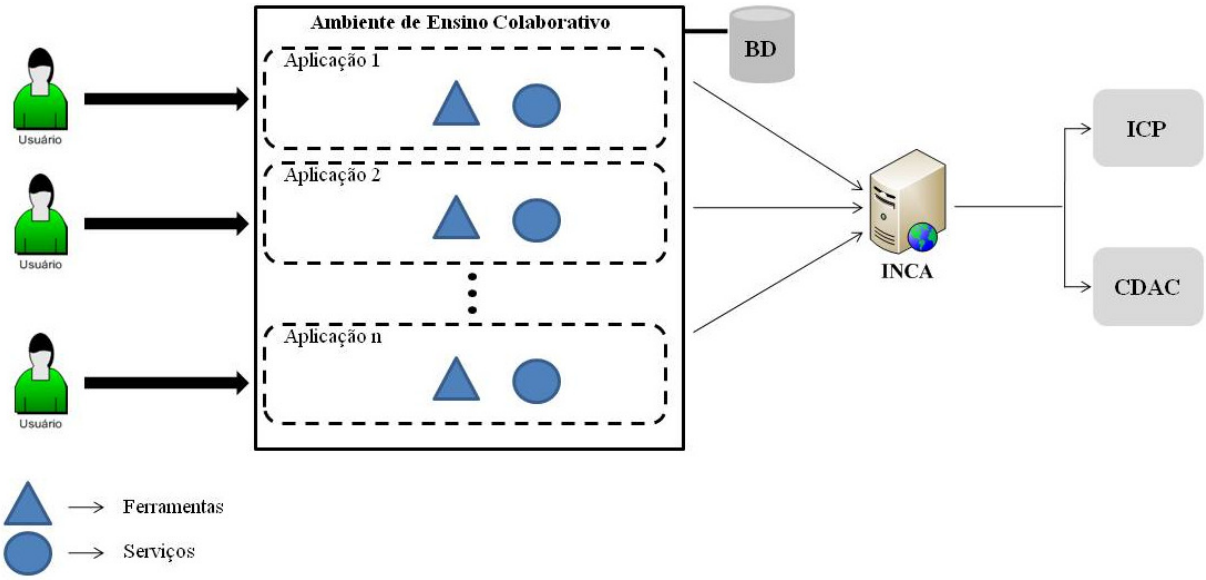

Figura 18 - Ambiente de utilização do INCA.

Para que uma aplicação use o INCA, é necessário que ela realize requisições ao serviço a fim de que ele possa ser ativado. Assim, ao utilizar as funcionalidades de uma aplicação no ambiente de ensino colaborativo, um usuário poderá fazer uso do INCA. O INCA, por sua vez, realiza a comunicação com a ICP e a CDAC por requisições de execuções.

4.2 Serviço de Segurança INCA

O INCA é um serviço criado para oferecer segurança a ambientes de *e-Learning*, de maneira que complemente as funcionalidades já existentes nos ambientes de *e-Learning* atuais (GUALBERTO et al., 2009).

O acesso ao INCA é realizado por meio de *Web Services*. Uma das contribuições deste trabalho é tirar vantagem das características dos *Web Services*, como independência de plataforma e heterogeneidade entre linguagens. Assim, o INCA pode oferecer segurança para as aplicações dos ambientes de *e-Learning* sem levar em consideração a arquitetura em que o sistema foi desenvolvido, bem como as linguagens de programação utilizadas em suas implementações.

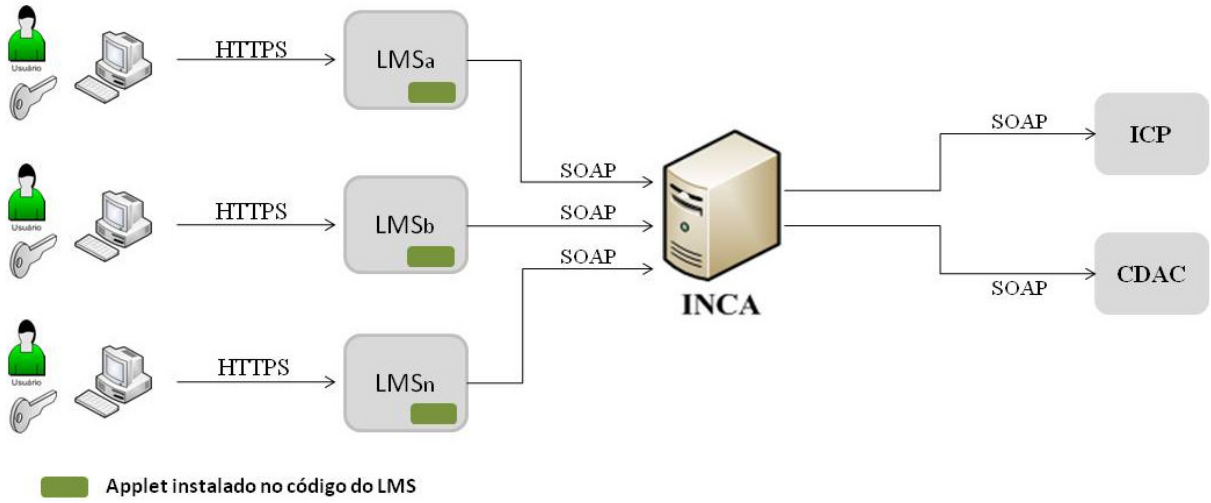

Figura 19- Funcionamento do INCA.

A Figura 19 ilustra o funcionamento do INCA, onde os usuários possuem uma chave privada e realizam acessos, por HTTPS (*Secure Hypertext Transfer Protocol*), à aplicação no ambiente de ensino colaborativo. Cada usuário possui um certificado digital emitido pela autoridade certificadora responsável por emitir e gerenciar os certificados utilizados pelo INCA.

Dentro do código do ambiente de *e-Learning* é incorporado um *Applet* – pequena aplicação escrita em linguagem Java e inserida em uma página *Web* – que permite que um usuário possa selecionar uma operação que ele deseja realizar, como assinaturas digitais, verificação de assinatura, codificação e decodificação. Dessa maneira, quando um usuário desejar assinar um arquivo digitalmente ou decodificar, o *Applet* é trazido para a máquina do cliente e este pode realizar as operações criptográficas citadas anteriormente. Assim, é possível garantir que essas operações, efetuadas pela utilização da chave privada do usuário, ocorram do lado do cliente, garantindo, assim, a irretratabilidade (o serviço de não repúdio). Caso o usuário deseje codificar ou verificar uma assinatura digital, o *Applet* realiza, também, requisições SOAP ao INCA. Nessas requisições, o nome dos usuários, as mensagens, os arquivos, dentre outras informações, são passadas ao INCA para que tais operações possam ser realizadas, isto é, para que um envelope digital possa ser criado e uma assinatura possa ser verificada.

Constantemente, o INCA realiza solicitações à ICP para verificar se o certificado digital, associado ao nome do usuário, está revogado ou não. Por meio dessas solicitações à ICP, o INCA pode adquirir as chaves públicas dos usuários para codificar as chaves simétricas dos usuários. A CDAC também recebe solicitações do INCA para que este adquira

as chaves simétricas dos usuários, como pode ser visto nas seções 4.3 e 4.4, respectivamente. As chaves simétricas e públicas são utilizadas pelo INCA para codificar a mensagem ou arquivo de um usuário e para codificar as chaves simétricas dos usuários, respectivamente. Já as chaves privadas são utilizadas pelo próprio usuário, através do *Applet* incorporado no código do ambiente de *e-Learning*, para realizar assinaturas digitais em mensagens ou arquivos.

Portanto, as questões de segurança, como integridade dos dados, não repúdio, confidencialidade dos dados e autenticidade, podem ser oferecidas a ambientes de ensino colaborativos, uma vez que nem todas as ferramentas/serviços destes ambientes oferecem requisitos de segurança.

Funcionalidades do INCA

INCA oferece quatro serviços de segurança: integridade, confidencialidade dos dados, o não repúdio e autenticidade, por meio de mecanismos como criptografia de chave pública e de chave simétrica e assinatura digital. O serviço de confidencialidade é oferecido através de envelope digital (HOUSLEY, 2004), onde o INCA utiliza uma chave de criptografia simétrica de um usuário para codificar as informações e, com a chave pública deste mesmo usuário, o INCA codifica a sua chave simétrica utilizada anteriormente. A decodificação da chave simétrica e do arquivo do usuário é realizada por um *Applet* que roda do lado do cliente. Em relação aos serviços de autenticidade, integridade e não repúdio, estes são oferecidos através de um *Applet* que realiza assinatura digital. A incorporação deste *Applet* ocorre no código do ambiente de *e-Learning* e tem como objetivo garantir que a assinatura digital seja realizada do lado do cliente (*browser* do usuário). A verificação de tal assinatura é realizada pelo INCA.

Para melhor visualização do serviço desenvolvido, apresenta-se, aqui, um possível cenário nos ambientes de *e-Learning* em que o uso do INCA é adequado: o envio de arquivo. Neste caso, INCA pode oferecer a confidencialidade e integridade de um documento. Também é possível garantir a identificação do aluno que enviou o trabalho ou prova (irretratabilidade). A certeza garantida aos alunos de que a integridade de seus trabalhos e a privacidade na entrega ao professor foi preservada, além de que aqueles que enviaram os trabalhos são quem afirmam ser, são essenciais para que os usuários de aplicações em ambientes de *e-Learning* sintam-se mais à vontade para utilizá-lo. Por essa razão, INCA pode ser aplicado em vários cenários de uma aplicação dentro de um ambiente de *e-Learning*.

Segurança do INCA

O INCA é um serviço que oferece segurança a ambiente de *e-Learning*. Por este motivo, também é importante garantir a segurança do INCA como um todo e a segurança no momento do uso do INCA por uma aplicação cliente.

Para se obter tal segurança, foi utilizada uma especificação chamada *Web Service Security* (NADALIN et al, 2006). Esta especificação estende o protocolo SOAP de forma a implantar mecanismos de segurança, como autenticação, políticas de acesso e criptografia para a construção de *Web Services* seguros. Assim, possibilitou-se definir restrições de acesso aos dados e serviços oferecidos, procurando garantir a segurança do INCA. A segurança na comunicação usuário-aplicação (ambiente de *e-Learning*) ocorre por meio de SSL.

No código do INCA, foi implantada a biblioteca *WS-Security*. Dessa forma, foi possível obter uma conexão segura entre as funcionalidades das aplicações dos ambientes de *e-Learning*, que são considerados clientes do INCA, e o INCA, garantindo uma segurança fim-a-fim não só no nível de transporte como também no nível de mensagem.

Implementação do INCA

Para oferecer serviços de segurança, o INCA foi implementado como um *Web Service* na linguagem Java.

Para oferecer integridade, não repúdio, confidencialidade e autenticidade, foram utilizadas as tecnologias *Java Security* e a API criptográfica *Bounce Castle*, que também é um provedor das especificações *Java Cryptography Extension* (JCE) e *Java Cryptography Architecture* (JCA). A tabela a seguir descreve as operações criptográficas realizadas pelo INCA para oferecer os quatro elementos de segurança supracitados.

Tabela 1 - Tecnologias utilizadas para oferecer as questões de segurança

	Criptografia de Chave Pública	**Assinatura Digital**	**Criptografia de Chave Secreta**
Integridade	✓	✓	✓
Não Repúdio		✓	
Confidencialidade	✓		✓
Autenticidade	✓	✓	

Para fornecer os elementos de segurança listados na tabela anterior, foi utilizado o algoritmo RSA na criptografia de chave pública, com chaves de tamanho 1024 *bits*. Para

realizar assinaturas digitais, utilizou-se SHA1 com RSA como algoritmo de assinatura, também com chaves de 1024 *bits*. Para assinar documentos digitalmente, foi utilizada a biblioteca iText (ITEXT, 2009), que permite realizar assinaturas digitais em arquivos no formato PDF (*Portable Document Format*). Para a criptografia simétrica, foi utilizado o algoritmo AES, com chaves de 256 *bits*.

4.3 Infraestrutura de Chaves Públicas (ICP)

INCA oferece os serviços de segurança, citados anteriormente, por meio de criptografia de chave pública e assinatura digital. Assim, o INCA utiliza certificados digitais para realizar operações como: codificação, decodificação, assinatura digital e verificação da assinatura. Para gerenciar os certificados digitais, utilizados pelo INCA, foi utilizado o *Enterprise JavaBeans Certificate Authority* – EJBCA (EJBCA, 2009) como autoridade certificadora.

Por mais que a sua utilização na literatura não seja tão sólida quanto a da OpenCA (OPENCA, 2009) e da OpenSSL (OPENSSL, 2009), a adoção do EJBCA advém do fato de este ter alta performance, ter independência de plataforma, ser flexível, ser implementado em Java, ser robusto e estar baseado em componentes, o que permite ser utilizado de forma independente ou integrado em uma aplicação J2EE.

O que mais influenciou sua adoção neste projeto foram as três primeiras características citadas acima. O EJBCA pode ter uma interface de administração e integração, isto é, pode funcionar como um *Web Service*. Assim, foi possível garantir que a CA utilizada tenha características de desempenho parecidas com as do INCA.

O EJBCA foi configurado para gerar certificados no formato X.509v3, que possuem informações como o nome do proprietário do certificado, uma chave pública, o nome da CA e a assinatura da CA. Estes certificados foram configurados com a validade de 4 meses, uma vez que este é o tempo de duração de um semestre acadêmico. Os pares de chaves foram gerados com o tamanho de 1024 *bits* e, para isso, foi utilizado o algoritmo de assinatura SHA1 com RSA.

O INCA realiza, constantemente, requisições SOAP ao EJBCA a fim de adquirir as chaves públicas associadas ao nome dos usuários das aplicações dos ambientes de *e-Learning*. Considerando que o EJBCA pode ser representado em uma arquitetura multicamada distribuída, a Figura 20 mostra tal representação. Nesta figura, é possível demonstrar onde o INCA atua nos módulos de sua arquitetura.

O INCA realiza verificações junto à LCR (Lista de Certificados Revogados) para averiguar se um certificado digital está revogado ou não, com o objetivo de obter a chave pública presente no certificado de um determinado usuário. Estas requisições são realizadas por meio de mensagens SOAP. Quando a CA recebe uma requisição do INCA, ela realiza uma operação de resposta com a chave pública do usuário. É importante lembrar a necessidade de uma comunicação segura entre o INCA e o EJBCA; para tanto, foi utilizado o protocolo SSL para se obter a comunicação segura entre estas duas entidades.

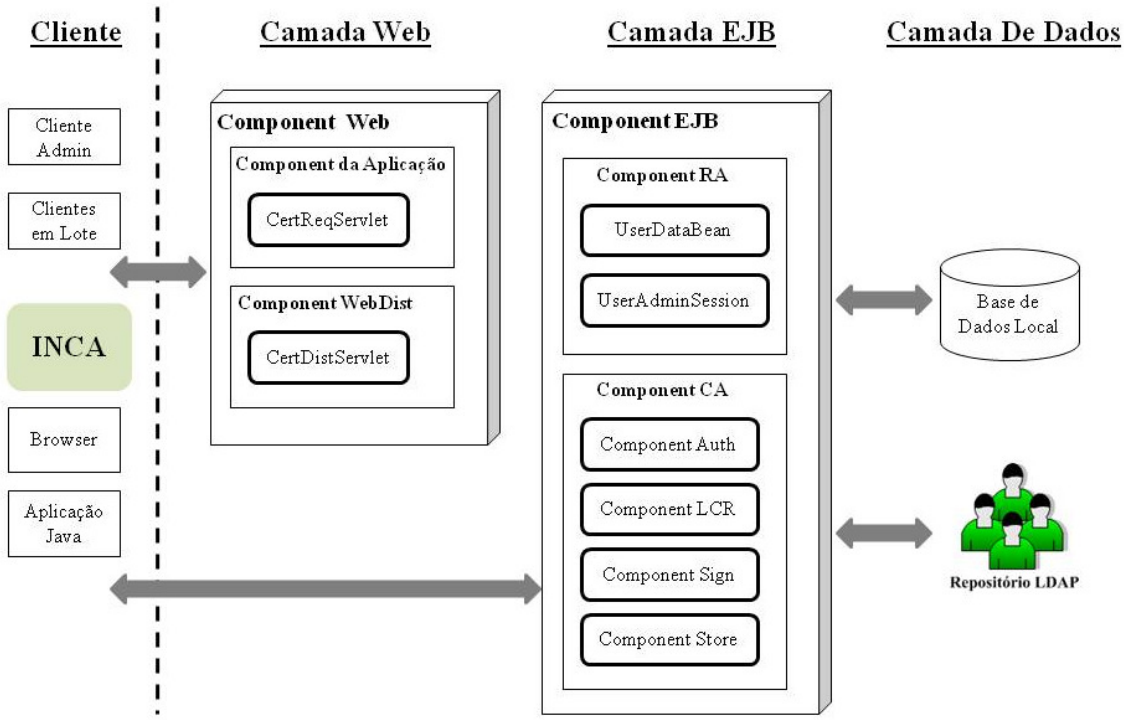

Figura 20 – Representação da interação do INCA com a arquitetura do EJBCA. Adaptado de EJBCA, 2009.

4.4 Central de Distribuição e Armazenamento de Chaves (CDAC)

Utilizada pelas operações de criptografia simétrica realizadas pelo INCA, a CDAC é uma entidade responsável por gerar e distribuir chaves simétricas, atuando como uma terceira parte confiável.

A CDAC fica disponível como um *Web Service*; desse modo, é possível manter as características como flexibilidade, independência de plataforma e suporte à escalabilidade presentes nas outras duas entidades (INCA e EJBCA). Portanto, pode-se ter um ambiente to-

talmente distribuído e escalável, uma vez que estas três entidades foram implementadas e instaladas como um *Web Service*. Na Figura 21, é ilustrada a arquitetura CDAC.

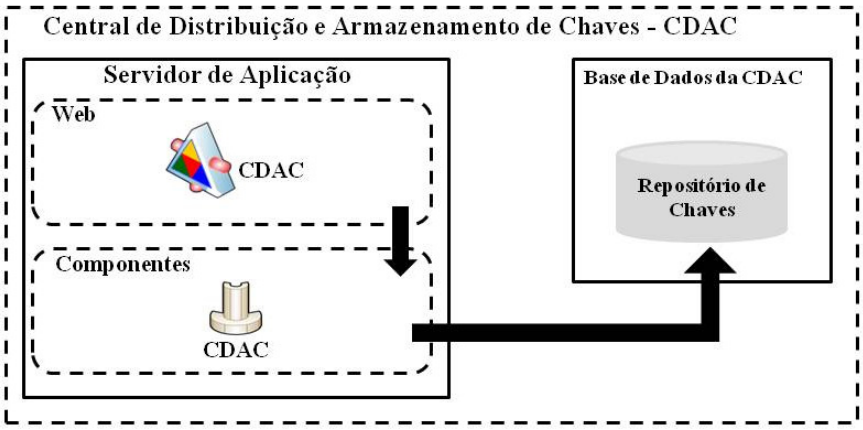

Figura 21 - Arquitetura CDAC.

Funcionamento CDAC

Uma Central de Distribuição de Chaves – *Key Distribution Center* (KDC) – utiliza uma chave mestra para a comunicação com um usuário, de maneira que este possa obter uma chave de sessão temporária para estabelecer uma sessão segura com outro usuário. Entretanto, a CDAC utiliza apenas uma chave simétrica para cada par de usuários.

As principais funcionalidades da CDAC são: gerar e distribuir chaves simétricas; realizar operações de busca; inserir e remover as chaves presentes na base de dados. A Figura 22 ilustra o funcionamento da CDAC utilizada pelo INCA.

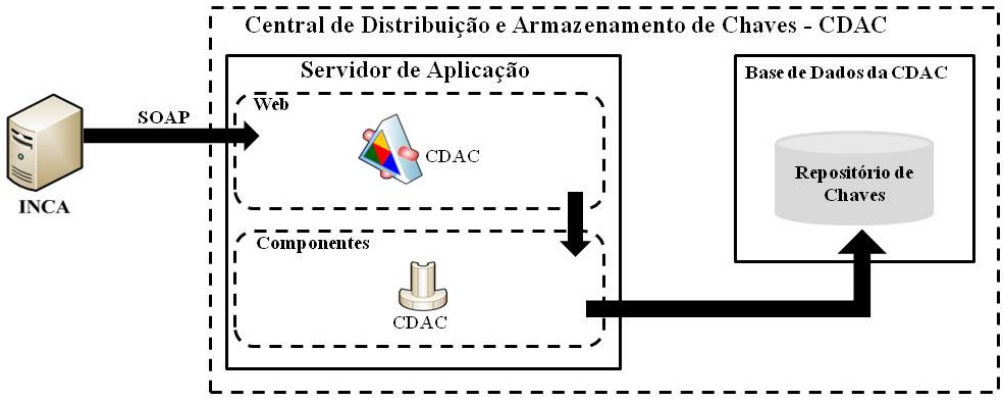

Figura 22 - Funcionamento da CDAC sendo utilizada pelo INCA.

O INCA realiza requisições à CDAC para que a chave de um determinado usuário possa ser adquirida. Esta chave é transmitida por meio de mensagens SOAP, sem a ne-

cessidade de gerar uma chave de sessão para cada comunicação entre os usuários, uma vez que a confiança na comunicação existe, somente, entre o INCA e a CDAC. A segurança na comunicação entre estas entidades é obtida com a utilização da especificação *WS-Security*.

Implementação CDAC

Como a CDAC é disponibilizada como um *Web Service* e implementada em Java, a adoção dessa tecnologia se faz devido ao fato de que ela suporta escalabilidade e tem um padrão na comunicação, já que o INCA também foi implementado como um *Web Service.*

A segurança na comunicação entre o INCA e a CDAC é obtida por meio do padrão *WS-Security*, a partir de cuja utilização é possível garantir uma segurança fim-a-fim entre ambas as partes.

A API *Bouncy Castle* foi utilizada para a geração das chaves simétricas de tamanho 256 *bits* com o algoritmo de criptografia simétrico AES. Essas chaves são armazenadas em um Banco de Dados *MySql* (versão 5), e a CDAC utiliza a tecnologia JPA (*Java Persistence API*) para fazer persistência à base de dados em que as chaves são armazenadas.

4.5 Relação dos Trabalhos Relacionados com o Presente Trabalho

Embora apresente abordagens diferentes, este trabalho compartilha algumas características de trabalhos citados nas seções anteriores.

Yee e Korba (2007) apresentaram uma abordagem de negociação consumidor-fornecedor que realiza a personalização da segurança, além de um novo método de fornecer ajuda durante a negociação. Com a intenção de aumentar a segurança exigida nos ambientes de serviços da *internet*, Yee e Korba (2007) adicionaram quatro tipos de serviços de segurança, além dos presentes nos padrões ISO e ITU-T: *log* seguro, certificação, *Malware Detection* e Monitoramento de Aplicações. Yee e Korba (2007) apresentaram, também, um protótipo baseado em uma arquitetura em JADE para a negociação e a personalização da segurança entre consumidor e fornecedor.

Em relação ao trabalho de Yee e Korba (2007), o presente trabalho visa a implementar um serviço que ofereça segurança para ambientes de ensino colaborativos. Como Yee e Korba (2007) já implementaram um protótipo em JADE para ser utilizado em serviços na *internet* de maneira geral, um dos objetivos deste trabalho é realizar um estudo de caso nos

ambientes de ensino colaborativo Sakai e Moodle, e analisar seu comportamento nestes ambientes.

Webber et al. (2007) implementaram uma plataforma multiagente que consiste em uma infraestrutura distribuída para tratar importantes assuntos de segurança, e que também foi utilizada para o desenvolvimento de ambientes de *e-Learning*.

Com relação ao trabalho de Webber et al. (2007), este trabalho focaliza aspectos de segurança em ambientes de *e-Learning* por meio de uma arquitetura orientada a serviços – os *Web Services*. Assim, por causa das características dos *Web Services* de independência de plataforma e heterogeneidade entre linguagens, podem ser oferecidos serviços de segurança a ambientes de *e-Learning* desenvolvidos em linguagens ou arquiteturas diferentes.

Raitman et al. (2005) analisaram a segurança da plataforma *Wiki* por meio de dois mecanismos: *login* e anonimato. Porém, o trabalho aqui proposto visa a aplicar e avaliar a segurança de outras ferramentas e serviços do ambiente Sakai e Moodle.

Lin et al. (2004) propuseram uma série de aplicações de EaD desenvolvidas em seu laboratório e mostraram como a segurança e a privacidade podem ser integradas nessas aplicações. Eles apresentaram requisitos de privacidade e segurança, juntamente com possíveis soluções para satisfazer às exigências de cada aplicação. Porém, os autores não apresentaram resultados da utilização dessas "possíveis" soluções.

Kambourakis et al. (2007) discutiram a potencial aplicação de certificados de atributo (AC's) num modelo de confiança proposto. A aplicação de AC's para suportar *m-Learning* (*mobile-Learning*) é apresentada por meio de um teste experimental que utiliza um dispositivo móvel, com poucos recursos, em redes GPRS. Porém, os autores não analisaram a utilização do modelo de confiança proposto em um ambiente de *e-Learning*, tampouco enfocaram as diversas arquiteturas e linguagens de programação em que os ambientes de *e-Learning* podem ser desenvolvidos e implementados; por isso, o modelo de confiança pode não se adaptar a um determinado sistema. Ao contrário do trabalho de Kambourakis, este trabalho visa a oferecer segurança independente do ambiente de *e-Learning*, o que foi possibilitado pelo fato de utilizar-se a tecnologia de *Web Service* para oferecer segurança a tais sistemas.

5 Experimentos, Resultados e Análises

Este capítulo apresenta uma série de experimentos realizados com o serviço INCA apresentado neste trabalho. Os testes permitiram a avaliação de aspectos como: a qualidade dos dados, o tempo de resposta das requisições, a escalabilidade e o "*stress*" do servidor. A seguir, são descritos os resultados obtidos em cada avaliação.

5.1 Ambiente de Execução

O ambiente de testes foi composto por quatro máquinas de igual configuração de *hardware* (INTEL Core Duo 3.0 GHz, 2 GB de RAM, conectadas por um *switch* de 1GB/s, formando uma rede local). Estas máquinas foram configuradas com os seguintes ambientes de *software*: Ubuntu 8.10 (EJBCA, INCA, CDAC) e SlackWare 12.1 (Client-SoapUI). Todas as máquinas possuíam uma máquina virtual Java da SUN, versão 1.6. A topologia de rede utilizada nos experimentos é apresentada pela Figura 23.

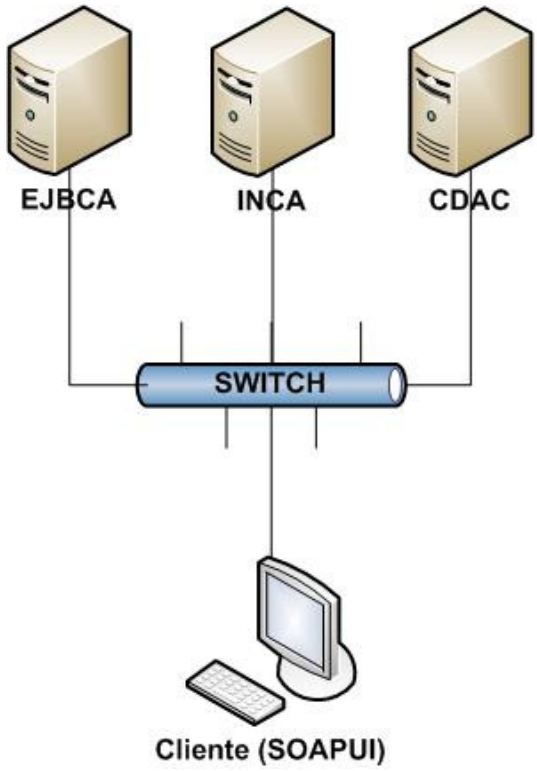

Figura 23 - Topologia de rede.

5.2 Analisando o Desempenho do INCA

Esta seção apresenta uma avaliação experimental do INCA. Os testes aqui realizados foram os de unidade e qualidade dos dados, e os de carga e escalabilidade. Para realizar este conjunto de testes foi instalada em uma máquina cliente a ferramenta SoapUI (SOAPUI, 2009), que permite a realização de testes automatizados com *Web Services*. Após esta instalação, a ferramenta SoapUI foi inicializada e a opção `File/New WSDL Project` foi selecionada para ativar o arquivo WSDL do INCA. Esta opção carrega e analisa o WSDL e gera requisições para cada método do INCA. Este *software* recolheu os dados da capacidade de requisições que acessaram o INCA, o tempo médio gasto para as requisições e as informações das requisições deferidas ou indeferidas durante o período pré-determinado. O teste de escalabilidade demonstrou resposta satisfatória e eficiente de aceitação de acesso ao INCA.

Teste de Unidade e Qualidade dos Dados

O primeiro conjunto de testes foi o de unidade e qualidade dos dados. Cada operação criptográfica do INCA foi testada com o objetivo de verificar sua validação, isto é, certificar se cada uma delas faz, realmente, o que afirma fazer. Este teste foi realizado nas operações de codificação com chave pública/simétrica e assinaturas digitais, onde cada uma delas recebeu requisições unitárias durante 5 minutos. Nessas requisições foram enviados: um arquivo e uma chave simétrica, para realizar a codificação simétrica; a chave simétrica citada anteriormente para realizar uma codificação com a chave pública de um usuário; e um arquivo e a chave privada para a realização de uma assinatura digital.

Após os três primeiros testes, foram realizadas operações complementares – decodificação com chave pública/simétrica e verificação da assinatura – a partir dos resultados obtidos nas operações iniciais, a fim de validar a qualidade dos dados que foram encriptados e assinados nos testes iniciais. Para a decodificação por criptografia simétrica foi enviada a chave simétrica de um usuário. Já em relação à decodificação por meio de chave pública, foi enviada a chave privada de um usuário. Por fim, foi enviada a chave pública de um usuário para verificar a validade e veracidade de uma assinatura digital realizada. Com este teste, pôde-se perceber que todas as operações realizadas pelo INCA cumprem o que afirmam fazer.

Teste de Carga e Escalabilidade

Os testes realizados anteriormente foram efetuados apenas para a verificação da qualidade dos dados obtidos em cada funcionalidade do INCA. No entanto, não são conclusivos no que diz respeito à utilização do INCA com sobrecarga de requisições. Alguns testes foram efetuados visando a uma avaliação concreta da capacidade do INCA com relação ao número de requisições recebidas e o tempo de processamento dessas requisições verificado com sobrecarga. A realização desses testes tinha por objetivo validar o desempenho do INCA, isto é, garantir que ele não "pararia" com a quantidade de requisições recebidas e que o seu comportamento não seria afetado quando executado para vários testes.

Para a execução desses testes foi utilizada a ferramenta SoapUI, de forma a simular clientes *Web Services* utilizando o INCA. Essa ferramenta era capaz de efetuar requisições para cada uma das seis funcionalidades oferecidas pelo INCA, de maneira concorrente, baseado na utilização de *threads*. Nessa ferramenta, foi adotada, para cada funcionalidade, uma estratégia, chamada *Simple*, a partir da qual foi possível variar o número de *threads* durante um determinado período de tempo.

Um aspecto importante a ser definido nesse momento do teste foi a determinação da quantidade de *threads* adotadas para realizar as requisições ao INCA. Para determinar um valor ideal a ser utilizado nos testes, foram executados testes iniciais com um número variável de uma a trinta e duas *threads*. Definiu-se que o número ideal de *threads* realizando requisições deveria ser composto por duas, quatro, oito, dezesseis e trinta e duas *threads*, respectivamente. Cada conjunto de *threads* era responsável por fazer requisições a uma funcionalidade específica do INCA durante o período de tempo pré-determinado na estratégia *simple* da ferramenta SoapUI. A adoção de tais quantidades de *threads* teve como objetivo avaliar o comportamento do INCA quando o número de *threads* aumentava de maneira considerável.

Uma vez definidos os valores para os números de *threads* ideais para a execução dos testes, outro aspecto importante precisou ser definido. Para serem utilizados nas requisições ao INCA foram necessários 4 parâmetros específicos: um par de chaves, pública e privada, de tamanho 1024 *bits*, uma chave simétrica de 256 *bits* e dois arquivos no formato pdf. O primeiro, arquivo "puro", de tamanho 954 bytes e o segundo, o arquivo puro assinado digitalmente, de tamanho 4,9 KB.

Definidos todos os parâmetros a serem utilizados nos testes de escalabilidade, seguiu-se, então, à execução dos mencionados testes. Cada um dos cinco elementos do conjunto de *threads* foi testado, realizando requisições, em cada uma das funcionalidades do IN-

CA durante o período de cinco minutos. Os testes foram executados sem interferências externas de rede que pudessem ocasionar atrasos resultantes de tráfego indesejado e, além disso, foram repetidos por seis vezes, em cada elemento do conjunto de *threads* de uma funcionalidade específica do INCA, para evitar alguma outra possível interferência externa. Neste caso, foram realizados – por exemplo, na função de codificação simétrica – seis testes com duas *threads*, seis testes com quatro *threads* e, assim sucessivamente até trinta e duas *threads*. Esse processo se repetiu para as outras cinco funcionalidades do INCA.

Na Figura 24, é ilustrado um gráfico que representa para cada funcionalidade do INCA, a variação de número de *threads* e o número total de requisições processadas pelo INCA. Esse último valor é a média de cada um dos seis testes realizados.

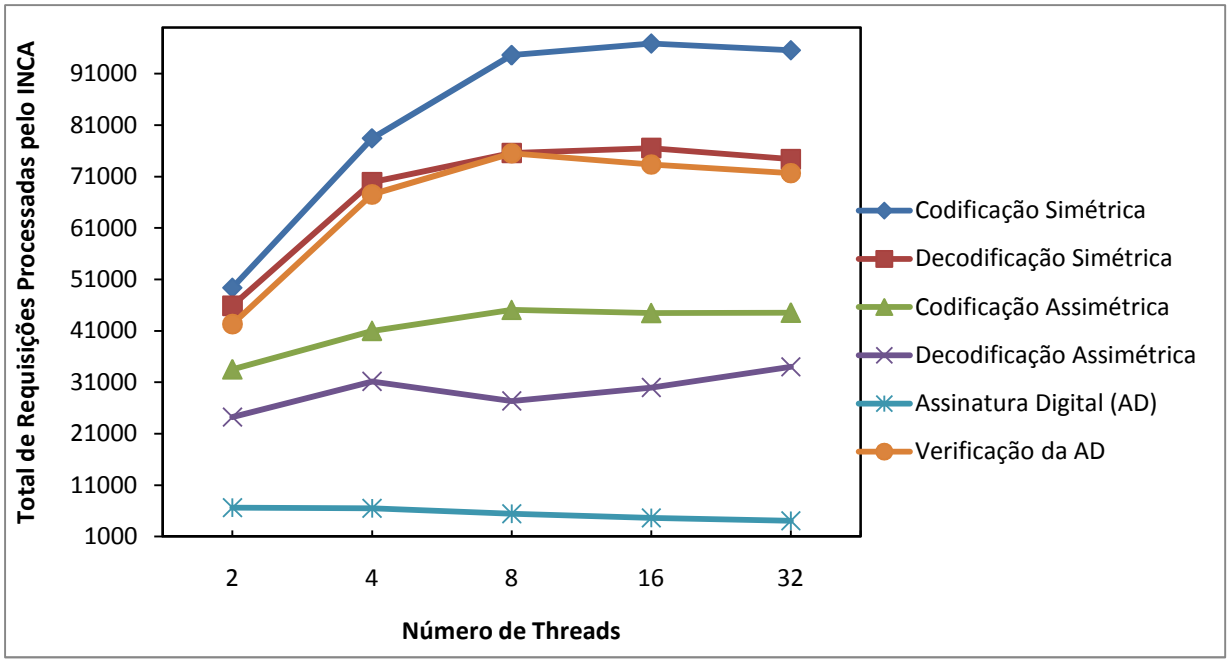

Figura 24 – Gráfico comparativo do total de requisições processadas pelo INCA com o número de *threads* que geraram tais requisições.

Nesse gráfico pode-se tirar algumas conclusões sobre os dados nele representados. Para as funcionalidades de codificação e decodificação simétrica, a quantidade de requisições processadas aumentou na medida em que o número de *threads* também cresceu. Entre 16 e 32 *threads* ocorreu um decréscimo de 1,33% (1290,5 requisições) e 2,84% (2174,5 requisições) em cada uma das funções acima, respectivamente, o que pode ser desconsiderado, uma vez que a quantidade de *threads* eram duas vezes maior.

Em relação às funcionalidades de codificação e decodificação assimétrica é possível perceber, na primeira função, um decréscimo de requisições processadas entre 8 e 32

threads. Mesmo assim, percebe-se que este fato pode ser desconsiderado, uma vez que as requisições geradas entre 8 e 32 *threads* diminuíram cerca de 1,22% (549,6 requisições). Já a segunda funcionalidade processou mais requisições quando estas foram realizadas por 32 *threads*.

Nesse mesmo gráfico, percebe-se uma queda considerável de requisições processadas pela funcionalidade de Assinatura Digital, em relação às outras funcionalidades. Isso ocorre devido aos três processos realizados em uma assinatura em arquivos no formato pdf que são: gerar o *hash* do arquivo pdf a ser assinado, encriptar o *hash* com a chave privada do usuário e, por fim, a inserção da assinatura (*enveloped signature*) no arquivo original. Em relação à funcionalidade de verificação de uma assinatura digital é possível perceber que ocorreu um decréscimo de 5,10% (3850,5) das requisições processadas entre 8 e 32 *threads*.

É importante ressaltar que todas as requisições geradas pelos conjuntos de *threads* e processadas pelas funcionalidades do INCA ocorreram com sucesso, isto é, o INCA não perdeu ou deixou de processar nenhuma dos milhares de requisições recebidas, ou seja, 0% de erro nas requisições processadas – resultado satisfatório com relação ao primeiro conjunto de testes realizados na seção anterior.

Os resultados ilustrados na Figura 24 podem ser mais bem analisados nas Tabelas 2, 3 e 4 que apresentam, para cada funcionalidade do INCA, a quantidade média de requisições processadas (QMRP) por quantidade de *threads* e o desvio padrão (DP). Além desses dois conjuntos de dados, a tabela também explicita a quantidade média de requisições processadas por segundo (QMRPs) e o tempo médio (em Milissegundos) da quantidade de requisições processadas (TMQRP) em QMPR. Pode ser verificado que, mesmo com desvio padrão considerável em alguns casos – como, por exemplo, a funcionalidade de codificação simétrica –, a quantidade de requisições processadas e o tempo médio de processamento das requisições observado são satisfatórios.

Tabela 2 - Dados das funcionalidades de Criptografia Simétrica

	Codificação Simétrica				**Decodificação Simétrica**			
№ de Threads	**QMRP**	**DP (%)**	**QMRPs**	**TMQRP**	**QMRP**	**DP (%)**	**QMRPs**	**TMQRP**
2	49369,5	3,768845	164,6	11,3	45872,7	0,0549	152,9	12,4
4	78417,7	6,916676	261,4	14,7	69925	0,1532	233,1	16,6
8	94598,5	3,552616	315,3	24,7	75578,3	0,1338	251,9	31,1
16	96847,7	2,889525	322,8	48,9	76546	0,2792	255,2	62
32	95557,2	1,238244	318,5	99,7	74371,3	0,6919	247,9	128,4

Tabela 3 - Dados das funcionalidades de Criptografia Assimétrica

	Codificação Assimétrica				Decodificação Assimétrica			
№ de Threads	**QMRP**	**DP (%)**	**QMRPs**	**TMQRP**	**QMRP**	**DP (%)**	**QMRPs**	**TMQRP**
2	33520,2	0,000356	111,7	17,3	24235,2	0,003767	80,8	24
4	41012,2	0,002065	136,7	28,6	31173,7	0,004425	103,9	37,9
8	45089,3	0,001083	150,3	52,6	27373,7	0,005269	91,2	87
16	44478,7	0,003117	148,3	107,2	29989,3	0,008164	100	159,3
32	44539,7	0,00415	148,5	214,7	33995,7	0,005188	113,3	281,3

Tabela 4 - Dados das funcionalidades de Assinatura Digital

	Assinatura Digital				Verificação da Assinatura Digital			
№ de Threads	**QMRP**	**DP (%)**	**QMRPs**	**TMQRP**	**QMRP**	**DP (%)**	**QMRPs**	**TMQRP**
2	6614,7	0,658879	22	90	42314,7	1,858031	141	13,5
4	6503,2	0,807341	21,7	183,4	67527	5,981627	225,1	17,2
8	5438,2	0,385967	18,1	439,9	75531,5	3,958324	251,8	31,2
16	4645,8	1,051012	15,5	1031,6	73361,2	1,108084	244,5	64,7
32	4092	1,222483	13,6	2343	71681	1,466433	238,9	133,1

Com o teste de carga e escalabilidade foi possível comprovar que o INCA se comporta de maneira estável, mesmo com a carga de requisições simultâneas a que foi submetido durante os testes realizados. Os testes foram realizados em um ambiente de execução não dedicado a essa aplicação. Se utilizarmos servidores diferentes para a oferta das funcionalidades do INCA, obteremos resultados melhores, pois os servidores não estarão gerenciando outras aplicações. Vale observar que as características dos servidores influenciam nos resultados testados e, com servidores com maior poder de processamento teremos resultados ainda melhores.

A realização destes testes possibilitou verificar que o INCA apresentou resultados de tempo de resposta adequados para a sua utilização em um ambiente de educação a distância, com a oferta de serviços de segurança de integridade, não-repúdio, confidencialidade dos dados e autenticidade.

Foi verificado também que, além de oferecer segurança com tempo de resposta adequado, cada uma das operações do INCA processou as requisições/resposta em conformidade com o que afirmava fazer.

No próximo capítulo, será apresentado um estudo de caso realizado nas funcionalidades de duas aplicações construídas nos ambientes de ensino colaborativo Sakai e Moodle, respectivamente.

6 Estudo de Caso

O objetivo deste estudo de caso é implementar as funcionalidades do INCA e a sua empregabilidade no serviço de *upload* de arquivos dos ambientes de ensino colaborativo Sakai e Moodle.

6.1 Representação do Cenário do Estudo de Caso

O cenário para a realização do estudo de caso foi composto por cinco máquinas com a mesma configuração de *hardware* (INTEL Core Duo 3.0 GHz, 2 GB de RAM). Estas máquinas foram configuradas com o ambiente de *software* Ubuntu 8.10. A topologia de rede utilizada neste estudo de caso é apresentada pela Figura 25.

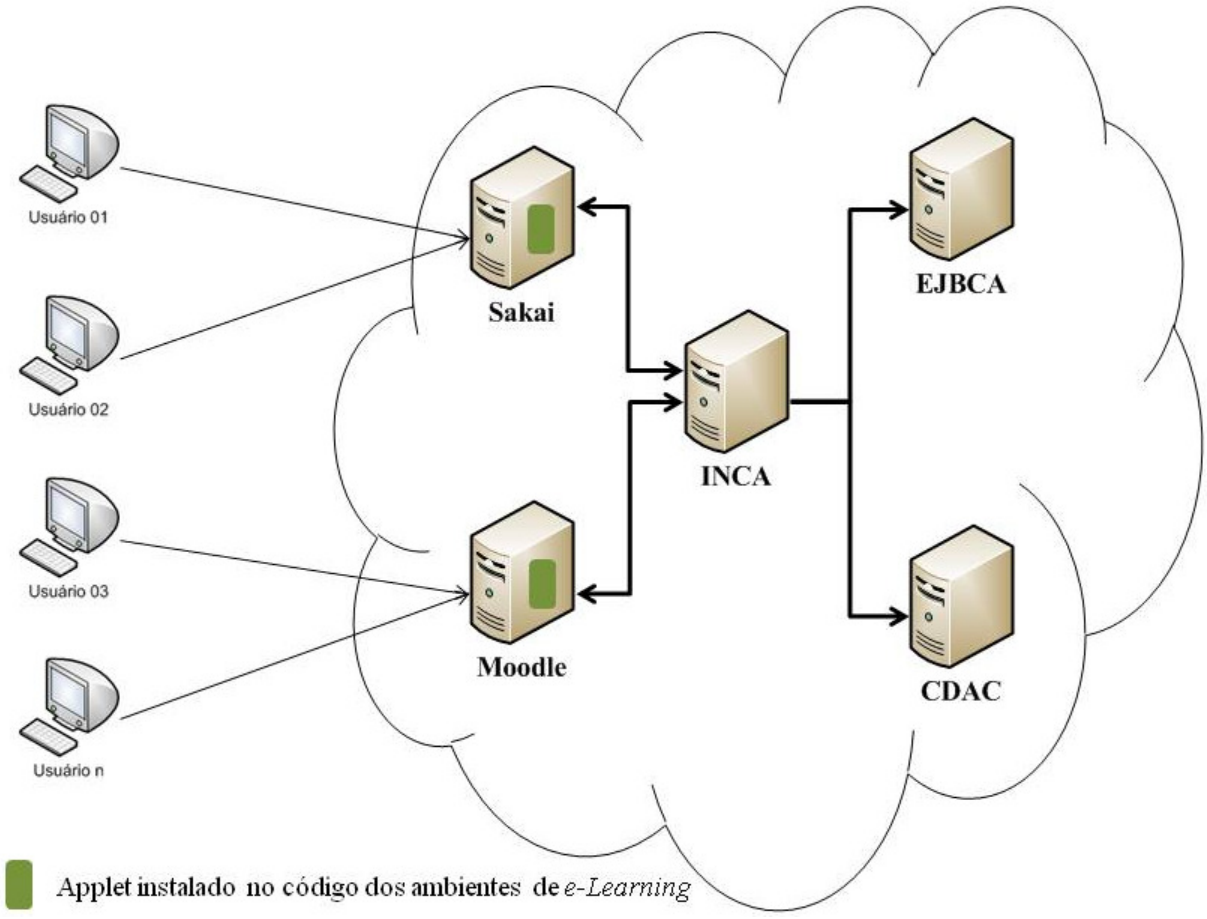

Figura 25 - Topologia de rede do estudo de caso.

6.2 Descrição do Estudo de Caso

De maneira a apoiar ou complementar o processo de aprendizagem dos alunos, alguns ambientes de *e-Learning* oferecem serviços e ferramentas, tais como *upload* de arquivos, envio de mensagens, avaliações, entrega de notas, entre outros. Ferramentas como o *chat* ou sala de bate-papo também são fornecidas por esses ambientes.

O estudo de caso deste trabalho foi aplicado no serviço de *upload* de arquivo dos ambientes de ensino colaborativo Sakai e Moodle, sendo que a adoção do ambiente Sakai no estudo de caso advém do fato de ele ser totalmente escrito em Java, além de seus serviços e ferramentas serem componentizados – o que permite o seu reuso. O fato de o Sakai ser orientado a serviços foi fundamental para o uso do serviço apresentado neste trabalho. Por outro lado, a adoção do Moodle se deve ao fato de ele ser escrito em uma linguagem de programação diferente – o PHP (*Hypertext Preprocessor*). A Figura 26 ilustra a arquitetura geral do estudo de caso realizado.

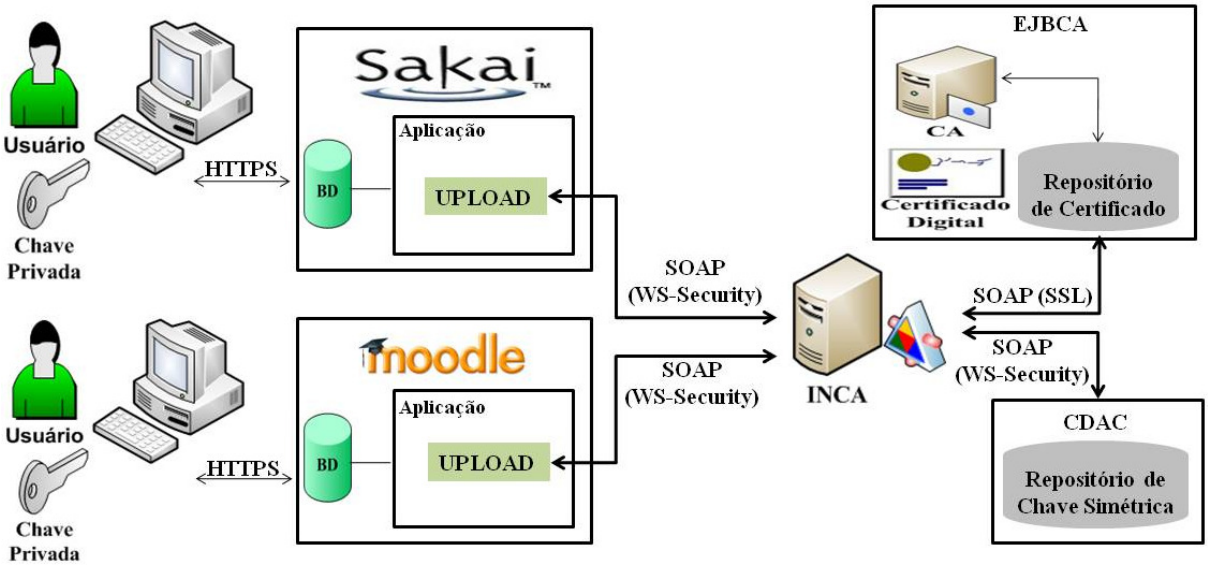

Figura 26 - Arquitetura geral do estudo de caso.

Na representação desta figura, vários usuários acessam os ambientes de ensino colaborativos adotados. Estes ambientes realizam requisições SOAP ao INCA a fim de ativar os serviços de segurança oferecidos por ele. O INCA, constantemente, realiza requisições à EJBCA e na CDAC a fim de adquirir as chaves públicas e chaves simétricas, respectivamente, dos usuários dos ambientes de *e-Learning*.

Os procedimentos necessários para a criação de uma aplicação tanto no ambiente Sakai quanto no Moodle estão detalhados no Apêndice A.

As próximas subseções descrevem os ajustes para que a ferramenta *Podcasts* do Sakai utilize as questões de segurança oferecidas pelo INCA.

6.3 Utilização do INCA no Sakai

O serviço de *upload* de arquivos é utilizado de maneira a permitir que professores e alunos compartilhem documentos dentro de uma aplicação. Este serviço funciona como um recurso para permitir que se faça o *upload* de vários tipos de arquivo, como relatório dos alunos, lista de exercícios, trabalhos, avaliações, entre outros. A adoção deste serviço para a realização do estudo de caso se dá pelo fato de ser um serviço assíncrono e por não existir serviços de segurança eficientes incorporados ao Sakai.

Uso da Ferramenta *Podcasts*

O serviço de *upload* de arquivo no ambiente Sakai pode ser realizado de várias maneiras, sendo aplicado em diversas ferramentas como *Drop Box, Resources* e *Podcasts*. Esta última foi adotada para a realização do estudo de caso.

Com o *Podcasts*, um usuário (professor ou aluno) pode enviar um arquivo, como por exemplo, provas, relatórios, listas de exercícios, entre outros, para o ambiente Sakai. Para isso, são necessários três passos. Primeiro, o usuário precisa selecionar o arquivo que ele deseja enviar; segundo, definir a data de publicação do arquivo e; por fim, definir o nome em que o arquivo será identificado no sistema. Após realizar estes passos, o usuário clica no botão `Add` para que o arquivo possa ser enviado para o sistema (ver Figura 39).

Para que a ferramenta *Podcasts* fizesse uso das funcionalidades do INCA foi necessária a realização de três procedimentos. O primeiro procedimento foi obter o arquivo WSDL, que descreve o INCA. Este arquivo pôde ser obtido pelo acesso à URL (*Uniform Resource Locator*) http://200.18.98.100:8084/ServidorServicos-v2/INCAv2Service?wsdl, que exibe o código WSDL do INCA. O conteúdo deste arquivo é apresentado no Apêndice B.

O segundo procedimento foi incorporar um *applet* na ferramenta *Podcasts* do ambiente de *e-Learning* Sakai. Este *applet* permite que o usuário escolha se deseja assinar digitalmente, verificar a assinatura, codificar ou decodificar um arquivo. A implantação deste *applet* teve como objetivo que a assinatura digital e a decodificação fossem realizadas do lado do cliente (*browser* do usuário do ambiente de *e-Learning*). Assim, quando um usuário selecionar um arquivo para ser enviado para a *Podcasts*, abre-se uma janela (applet) em seu navegador e o usuário pode selecionar se ele deseja assinar ou decodificar um arquivo. Na primei-

ra opção, o usuário seleciona uma chave privada e um arquivo para ser assinado digitalmente. Na segunda opção, o usuário seleciona o seu nome - caso esteja presente na lista de usuários para o qual o envelope digital foi criado - e sua chave privada - utilizada para decodificar a chave simétrica do usuário. Pela utilização da chave simétrica o envelope digital é, então, decodificado. Dessa maneira, é possível garantir o serviço de não repúdio.

Já as operações de codificação e verificação de uma assinatura são realizadas no INCA. Para isso, foram incorporadas requisições SOAP no código do *Applet* para que as operações citadas anteriormente possam ser realizadas.

Neste procedimento, também foi utilizada a biblioteca *WS-Security* para garantir a segurança na comunicação entre a ferramenta *Podcasts* e o INCA. Dessa maneira, entre o usuário e o ambiente de *e-Learning* tem-se uma comunicação segura por meio de SSL, e entre esse ambiente (ferramenta *Podcasts*) e o INCA, a comunicação segura ocorre pela utilização de *WS-Security*.

O terceiro e último procedimento foi fazer com que a ferramenta *Podcasts* utilize o INCA. O primeiro passo foi interromper a execução do servidor *Web Tomcat* com o comando "**>./shutdown**". Em seguida, foi necessário limpar os códigos da ferramenta *Podcasts* do Sakai (cln), construir os novos códigos (bld) e, por fim, fazer o *deploy* (dpl). Estas ações podem ser realizadas pela execução do comando "**>maven cln bld dpl**" dentro do diretório da *Podcasts*, presente na estrutura de diretórios do Sakai. Com a execução do comando anterior não foi necessário recompilar todo o código do Sakai, mas sim os códigos presentes no diretório do *Podcasts* onde foram realizadas as modificações. Por fim, o *Tomcat* foi reinicializado com o comando "**>./startup**" e as modificações realizadas puderam ser utilizadas.

O detalhamento do funcionamento da *Podcasts* após as modificações realizadas é ilustrado pelas Figuras 27 e 28, que exibem os passos da ferramenta *Podcasts* com o emprego das funcionalidades de assinatura digital/verificação da assinatura e criptografia assimétrica-simétrica, respectivamente.

Na Figura 27, são representados os serviços de integridade, não repúdio e autenticidade (autoria) por meio do mecanismo de assinatura digital.

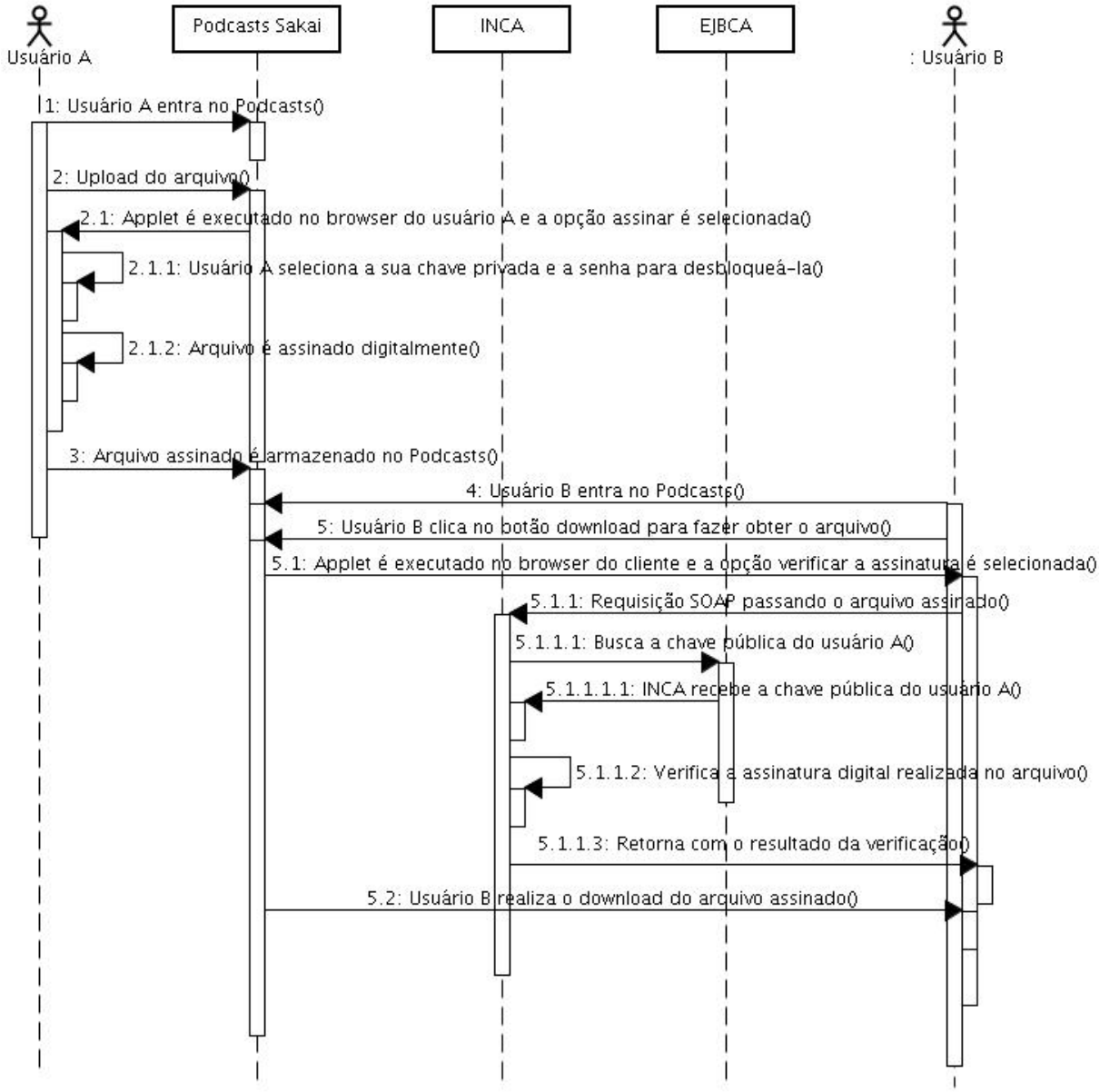

Figura 27 - Diagrama de Sequência do uso de assinatura digital e verificação da assinatura na ferramenta *Podcasts* do Sakai.

O acesso ao Sakai ocorre por meio de HTTPS, isto é, existe um canal seguro na comunicação entre os usuários e as funcionalidades de uma aplicação deste ambiente de *e-Learning*.

O usuário A, já autenticado no ambiente de *e-Learning*, entra no *Podcasts*. Em seguida, o usuário fornece as seguintes informações: seleciona o arquivo que deseja submeter ao sistema, a sua data de publicação e o título em que ele será identificado na ferramenta. Após este processo, o usuário A clica no botão *Add* para realizar o *upload* do arquivo.

Antes de o arquivo ser armazenado na base de dados do sistema, o código do *applet* é, então, trazido para a máquina do usuário, executado, e a opção “Assinar arquivo

digitalmente" é selecionada pelo mesmo. A tela do *applet* é redirecionada para outra tela que contém as informações definidas pelo usuário A na ferramenta. Além dessas informações, o usuário precisa selecionar a sua chave privada e a senha para desbloqueá-la. Antes de o arquivo selecionado ser assinado digitalmente, é verificado se o certificado digital associado à chave privada está presente na lista de certificados revogados. Se o certificado estiver revogado, o arquivo não é assinado e o método *cancelUpload* do código da ferramenta *Podcasts* é chamado. Caso contrário, o arquivo é assinado e armazenado no sistema. A assinatura digital realizada neste arquivo é do tipo *enveloped signature* (LIOY; RAMUNNO, 2004).

Um usuário B, já autenticado no ambiente de ensino colaborativo e que deseja fazer o *download* do arquivo, armazenado no *Podcasts*, entra na ferramenta e clica no botão *Download*.

Antes de o usuário B fazer o *download* do arquivo assinado digitalmente, o código do *applet* é, novamente, executado na máquina do usuário e a opção "Verificar assinatura digital" é selecionada. Neste momento, é realizada uma requisição SOAP ao INCA, enviando o arquivo assinado digitalmente. O INCA busca, na EJBCA, a chave pública associada ao certificado digital presente na assinatura do arquivo. De posse dessa chave pública, o INCA verifica a assinatura digital realizada no arquivo e retorna o resultado desta verificação ao *applet* que está rodando no browser do usuário. O *applet* mostra para o usuário uma mensagem informando se a assinatura, presente no arquivo, foi realizada por um certificado digital válido e, por fim, o usuário B adquire o arquivo.

Na Figura 28, é representado o serviço de confidencialidade (sigilo) por meio do mecanismo de criptografia assimétrica e simétrica.

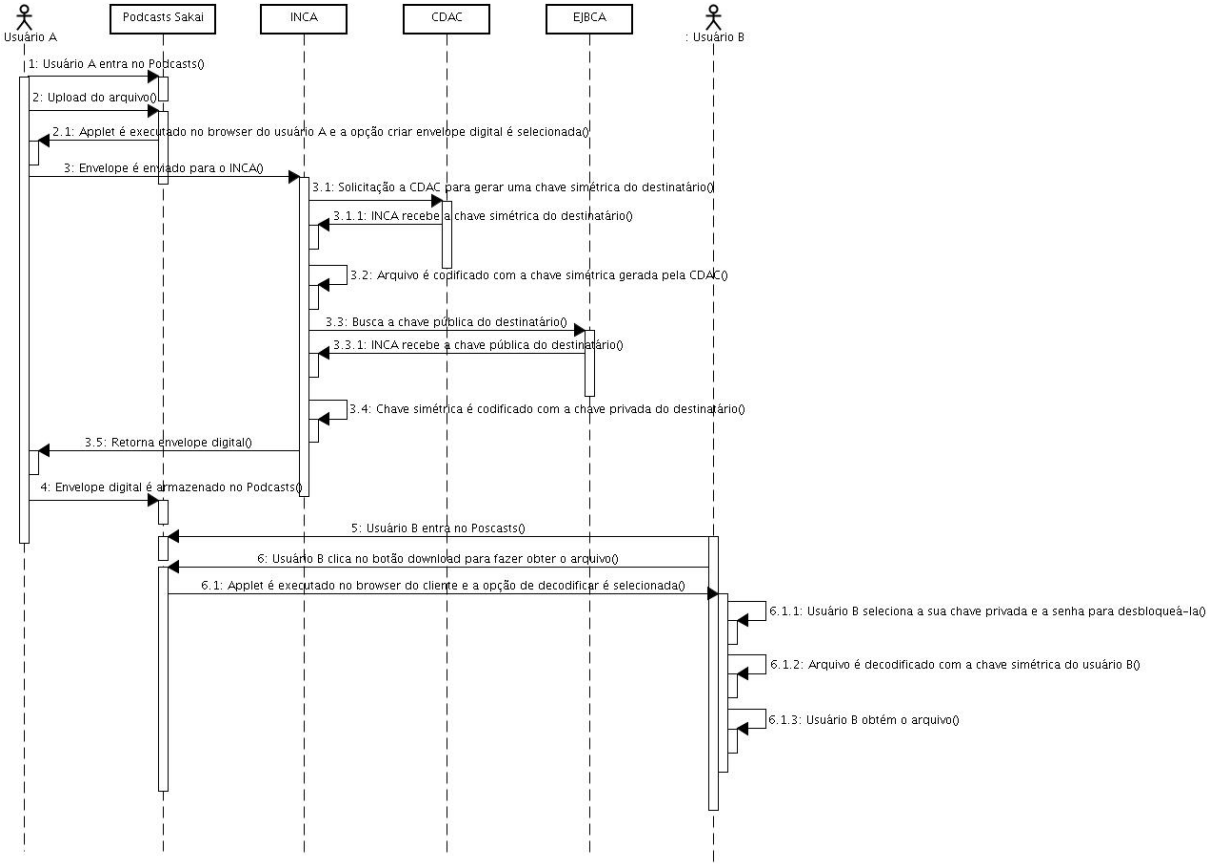

Figura 28 - Diagrama de Sequência do uso de criptografia assimétrica/simétrica na ferramenta *Podcasts* do Sakai.

Nos passos 1, 2 e 2.1 da figura 28 ocorrem os mesmos passos dos da figura 27. Porém, a opção selecionada foi "Criar envelope digital". A tela do *applet* é redirecionada para outra tela que contém as informações definidas pelo usuário A na ferramenta.

Além dessas informações, é possível selecionar os destinatários que, quando acessarem a *Podcasts*, poderão realizar o *download* do arquivo. Após selecionar os destinatários, um envelope digital contendo o arquivo, os destinatários e o dono (usuário A) do envelope é enviado por meio de uma requisição SOAP ao INCA para que possa ser criado. A segurança nesse acesso ao INCA é obtida pela utilização da biblioteca *WS-Security*. Dessa maneira, tem-se uma seção fim-a-fim segura entre o usuário e o INCA, uma vez que existe um canal seguro entre o usuário e o *Podcasts* – pela utilização de SSL - e entre o *Podcasts* e o INCA – pela utilização de WS-Security. O INCA realiza uma solicitação à CDAC para gerar uma chave simétrica do primeiro destinatário selecionado. Com esta chave, o INCA codifica o arquivo. Após esta codificação, o INCA busca na EJBCA a chave pública do primeiro destinatário e codifica a chave simétrica, gerada pela CDAC anteriormente, utilizada na codificação do arquivo. A quantidade de execuções entre o passo 3.1 e 3.4 é relacionada à quantidade

de destinatários selecionados pelo dono (usuário A) do envelope. Após este passo, o INCA cria um envelope digital do tipo CMS *envelopedData* (HOUSLEY, 1999) e envia de volta ao *applet*. Em seguida, o envelope digital é armazenado na base de dados do sistema.

Um usuário B, já autenticado no ambiente de ensino colaborativo, que deseja fazer o *download* do arquivo armazenado no *Podcasts*, entra na ferramenta e clica no botão *Download*.

Antes de o usuário B fazer o *download* do arquivo, presente no envelope digital, o *applet* é executado, novamente, no *browser* do cliente e a opção decodificar é selecionada. A tela do *applet* é redirecionada para outra tela que contém as informações definidas pelo usuário A na ferramenta. Além dessas informações, o usuário B precisa selecionar a sua chave privada e a senha para desbloqueá-la. O *applet*, por meio da chave privada do usuário B, decodifica a chave simétrica deste mesmo usuário que está codificada no envelope digital. Após esta decodificação, o *applet* utiliza a chave simétrica para decodificar o arquivo e o usuário B pode, então, obtê-lo em seu formato original.

6.4 Utilização do INCA no Moodle

Para que a ferramenta de envio de arquivo do Moodle utilize o INCA foi necessária a realização de dois procedimentos. O primeiro procedimento visa a obter o arquivo WSDL que descreve o INCA e o segundo, à implantação do *applet*, com as operações de assinatura digital, decodificação, codificação e verificação da assinatura. As operações de assinatura digital e decodificação precisam ser implementadas para que possam ser realizadas no *browser* do cliente. Neste mesmo *applet* foram incorporadas requisições SOAP de maneira que a codificação e a verificação da assinatura ocorressem no INCA.

Ao contrário do ambiente Sakai, no Moodle não foi necessário recompilar nenhuma parte do código. Como o PHP é uma linguagem interpretada, não foi necessário reinicializar o servidor em que o Moodle está hospedado. As alterações foram realizadas no código e, em seguida, foi executado “ctrl+alt+F5”, o comando de atualização de conteúdo, de tal forma que a página visualizada seja recarregada exibindo as modificações realizadas.

O estudo de caso na ferramenta de *upload* de arquivo do Moodle foi realizado. Portanto, devido à sua semelhança com a ferramenta *Podcast* do Sakai, não será descrito detalhadamente.

A realização deste estudo de caso teve como objetivo realizar a prova de conceito do serviço INCA apresentado neste trabalho. Com tal estudo, foi possível analisar a viabilidade da utilização do INCA tanto no ambiente de ensino colaborativo Sakai, quanto no Moodle, uma vez que, neste estudo de caso, puderam-se descrever as modificações e, enfim, todo o processo necessário para que tais ambientes possam utilizar o INCA de maneira efetiva.

O estudo de caso apresentado exibe a prova de conceito para o INCA em ambientes de ensino colaborativos. A contribuição do INCA no estudo de caso direcionou aos sistemas Moodle e Sakai, especificamente na ferramenta de *upload* de arquivo presente nesses sistemas.

No próximo capítulo, são apresentadas as conclusões gerais deste trabalho e os apontamentos para trabalhos futuros.

7 Conclusão e Trabalhos Futuros

O principal objetivo buscado neste trabalho foi oferecer segurança para os ambientes de ensino colaborativo, independentemente da arquitetura em que foram desenvolvidos. Assim, a principal contribuição do trabalho foi apresentar o INCA, um serviço que oferece segurança para diferentes sistemas de *e-Learning*. Entretanto, diversas outras contribuições obtidas durante o progresso do trabalho colaboraram para a agregação de valor a esta contribuição principal.

O estudo de caso realizado foi fundamental para a realização da prova de conceito do INCA. Assim, foi possível analisar a aplicabilidade do INCA integrado a diferentes ferramentas de diferentes sistemas de aprendizado eletrônico, tais como Sakai e Moodle. Dessa maneira, foi possível perceber características de flexibilidade, configuração e interoperabilidade oferecidas pelo INCA. A característica de flexibilidade foi alcançada devido à sua utilização na ferramenta assíncrona de *upload* de arquivos. O funcionamento dos demais módulos do Sakai e Moodle não sofreu interferência. Em relação à característica de configuração, a utilização do INCA nos sistemas de *e-Learning* supracitados permitiu a escolha de qual mecanismo de segurança utilizar para garantir o serviço de segurança desejado. A realização de tal estudo de caso também permitiu a interoperabilidade oferecida pelo INCA, uma vez que este estudo foi realizado em sistemas de *e-Learning* desenvolvidos em linguagens e arquiteturas diferentes.

Em particular, as pesquisas sobre a segurança em sistemas de *e-Learning* propostas na literatura mostraram que este assunto em tais sistemas não pode ser negligenciado. A abordagem adotada neste trabalho não foi muito encontrada na literatura, uma vez que grande parte dos trabalhos foca a implantação de segurança dentro desses sistemas de aprendizado e algumas das abordagens utilizadas são efetuadas por meio de agentes de *software*, limitando, assim, a necessidade de a segurança ser fornecida a sistemas desenvolvidos na mesma arquitetura e linguagem.

Outra característica que pôde ser encontrada no INCA foi a escalabilidade e a eficiência, uma vez que os testes realizados possibilitaram a avaliação do comportamento do INCA. Pela utilização da estratégia *Simple*, foi possível simular o comportamento que o INCA poderá ter nos momentos em que o acesso a ele for muito alto ou baixo. Portanto, o teste de escalabilidade demonstrou resposta satisfatória e eficiente de aceitação de acesso ao INCA.

Finalmente, a partir dos resultados alcançados, por meio dos testes e da análise do estudo de caso realizado, foi possível obter algumas conclusões sobre a segurança em ambientes de ensino colaborativo. Por meio de tais experimentos, pôde-se constatar que o INCA oferece segurança a sistemas de aprendizado eletrônico de maneira flexível, configurável, escalável e de forma eficiente.

A avaliação do INCA foi feita em uma ferramenta de dois ambientes de *e-Learning*. Tal avaliação serviu para a prova de conceito, mas não há garantias da facilidade de usabilidade em outras ferramentas de tais ambientes ou mesmo em outros ambientes.

Na implementação do INCA, foi utilizada a especificação *WS-Security* para tratar a segurança do INCA, mas há outras formas de implementar um *Web Services* seguro, como as especificações *WS-SecurityPolicy*, *WS-Authorization*, *WS-SecureConversation*, dentre outras. A implementação de tais especificações é possível, embora a complexidade de implementação não tenha sido relatada neste trabalho.

A realização deste trabalho criou diversas perspectivas para a realização de trabalhos futuros como continuação desta pesquisa, tais como a aplicação do INCA nas outras ferramentas e serviços presentes nos sistemas de *e-Learning*, como *WhiteBoard*, fórum, avaliações, entre outras e a avaliação de seu comportamento.

Outra possibilidade de trabalho futuro é a avaliação de mecanismos de privacidade por meio de *Web Services*, como, por exemplo, anonimato, pseudônimo, entre outros, aplicados em diferentes módulos dos sistemas de *e-Learning*.

Por fim, a implementação dos serviços de segurança de controle de acesso e autenticação por meio de *Web Services* e a realização de um estudo de caso em diferentes sistemas de aprendizado eletrônico também são uma possibilidade de trabalho futuro.

8 Referências Bibliográficas

ARRUDA JR, C.R.E. **Context Kernel: um Web Service baseado nas dimensões de informação de contexto**. 2003. 85p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, São Carlos, 2003.

ARSANJANI, A.; HAILPERN, B.; MARTIN, J.; TARR, P. Web services: Promises and compromises. **ACM Queue**, v. 1, n. 1, p. 48–58. 2003.

AULANET. Plataforma LMS. Disponível em: <http://www.aulanet.pt/>. Acessado em 20 de Maio de 2009.

BARKLEY, J.; CINCOTTA, A.; FERRAIOLO, D.; GAVRILLA, S.; KUHN, R. Role Based Access Control for the World Wide Web. In: 20TH NATIONAL COMPUTER SECURITY CONFERENCE, 1997. **Proceedings…** Baltimore, MD, USA, 1997. p. 1–11.

BERTINO, E.; SANDHU, R. Database Security - Concepts, Approaches, and Challenges. **IEEE Transaction on dependable and secure computing**. v. 2, n. 1, p. 2-19, jan/mar. 2005.

BOOTH, D.; LIU, C. **Web Services Description Language (WSDL) Version 2.0 Part 0: Primer**. World Wide Web Consortium (W3C) Recommendation, 2006. Disponível em: <http://www.w3.org/TR/2006/CR-wsdl20-primer-20060327/>. Acesso em: 16 de Abril de 2008.

BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E. CHAMPION, M. FERRIS, C. ORCHARD, D. **Web Services Architecture**. World Wide Web Consortium (W3C) Working Group Note, v. 11, 2004. Disponível em: < http://www.w3.org/TR/ws-arch/ >. Acesso em: 15 Abril de 2008.

BRAY, T. **Extensible Markup Language (XML) 1.0**. W3C Recommendation, 2004. Disponível em: <http://www.renderx.com/demos/xmlspec/xml/REC-xml-20040204.pdf>. Acesso em: 30 de Outubro de 2008.

BYUN, J.; SOHN, Y.; BERTINO, E. Systematic Control and Management of Data Integrity. In: 11TH ACM SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES, 2006. **Proceedings…** Lake Tahoe, California, USA, 2006. p. 101-110.

CERAMI, E. **Web Services Essentials: Distributed Applications with XML-RPC, Soap, UDDI and Wsdl**. O'Reilly & Associates, Inc. Sebastopol, CA, USA, 2002. 304 p.

CHÁVEZ, M. L.; ROSETE, C. H.; HENRÍQUEZ, F. R. Achieving Confidentiality Security Service for CAN. In: 15TH INTERNATIONAL CONFERENCE ON ELECTRONICS, COMMUNICATIONS AND COMPUTERS (CONIELECOMP), 2005. **Proceedings…** Puebla, Mexico, 2005. p. 166-170.

CLEMENT, L. **Universal Description Discovery and Integration (UDDI) Version 3.0.2**. OASIS UDDI Specification TC, 2004. Disponível em: <http://www.uddi.org/pubs/uddi_v3.htm>. Acessado em: 21 de Junho de 2008.

CORBA. **Common Object Request Broker Architecture**: Core Specification. Object Management Group, 2004, v3.0.3. Disponível em: <http://www.omg.org/docs/formal/04-03-12.pdf>.

CURBERA, F.; DUFTLER, M.; KHALAF, F.; NAGY, W.; MUKHI, N.; WEERAWARANA, S.; CENTER, I.B.M.T.J.W.R.; HEIGHTS, Y. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. **IEEE Internet computing**. v. 6, n. 2, p. 86-93, mar/abr. 2002.

DIFFIE, W.; HELLMAN, M. New Directions in Cryptography. **IEEE Transactions on Information Theory**, v. 22, n. 6, p. 644-654, Novembro. 1976.

DZIERZAWSKI, D.; ALLEN, B.; HAMILTON, F.C. Automatic access controls in the defense message system (DMS). In: IEEE MILITARY COMMUNICATIONS CONFERENCE (MILCOM), v. 2, 1999. **Proceedings…** Atlantic City, NJ, USA, 1999. p. 1262–1266.

EJBCA. **Enterprise JavaBeans Certificate Authority**. Disponível em: <http://www.ejbca.org/>. Acessado em 09 de Junho de 2009.

ERL, T. **Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services**. Pretice Hall PTR, Upper Saddle River, NJ, USA, 2004.

FERRAIOLO, D. F.; KUHN, D. R. Role based access control. In: 15TH NATIONAL COMPUTER SECURITY CONFERENCE, 1992. **Proceedings…** Baltimore, MD, USA, 1992. p. 544–554.

FIELDING, R.; GETTYS, J.; MOGUL, JC.; FRYSTYK, H. MASINTER, L.; LEACH, P.; BERNERS-LEE, T. **Hypertext Transfer Protocol (HTTP) 1.1**. Request for Comments (RFC) 2616, 1999. Disponível em: <http://www.ietf.org/rfc/rfc2616.txt>. Acessado em: 19 de Junho de 2009.

FULLER, J.; FUECKS, H.; EGERVARI, K.; WATERS, B.; SOLIN, D. STEPHENS, J.; REYNOLDS, L. **Professional PHP web services**. Wrox Press. Birmingham, 2003. 550 p.

FURNELL, S.M.; ONIONS, P.D.; BLEIMANN, U.; GOJNY, U.; KNAHL, M.; RODER, H.K.; SANDERS, P.W. A security framework for online distance learning and training. **Internet Research: Electronic Networking Applications and Policy**. v. 8 n. 3, pp. 236-242. 1998.

FREMANTLE, P.; WEERAWANA, S.; KHALAF, R. Enterprise Services, **Communications of the ACM**, v. 45 n. 10, p.77 - 82, 2002.

GELBORD, B. On the use of PKI technologies for secure and private e-learning environments. In: 4TH INTERNATIONAL CONFERENCE CONFERENCE ON COMPUTER SYSTEMS AND TECHNOLOGIES: E-LEARNING, 2003. **Proceedings…** Rousse, Bulgaria, p. 568-572, 2003.

GOTTSCHALK, K.; GRAHAM, S.; KREGER, H.; SNELL, J. Introduction to web services architecture. **IBM Systems Journal**, v. 41, p. 170–177, 2002.

GRAF, F. Providing security for eLearning. **Computers & Graphics**, v. 26, n. 2, p. 355–365, 2002.

GUALBERTO, T. M.; ABIB, S.; ZORZO, S. D. INCA: A Security Service for Collaborative Learning Environments. In: INTERNATIONAL CONFERENCE ON EDUCATION TECHNOLOGY AND COMPUTER (ICETC), 2009. **Proceedings…** Cingapura, IEEE Computer Society, 2009. p. 111-115.

GUALBERTO, T. M.; ZORZO, S. D. Incorporating Flexible, Configurable and Scalable Security to the Education Collaborative Environments. In: 39TH ANNUAL FRONTIERS IN EDUCATION (FIE) CONFERENCE, 2009, San Antônio, Texas, USA. 978-1-4244-4714-5.

HENNING, M. The rise and fall of CORBA. **Communications of the ACM**. v. 51, n. 8, p. 52-57, Agosto, 2008.

HOUSLEY, R. **Cryptographic Message Syntax (CMS)**. Internet Engineering Task Force (IETF). July, 2004.

ISO. International organization for standardization. **ISO 7498-2**, Information processing systems open systems interconnection basic reference model part 2: Security architecture. p. 32. 1989.

ITEXT. Biblioteca Java iText para gerar arquivos PDF. Disponível em: <http://www.lowagie.com/iText/index.html>. Acessado em 23 de Maio de 2009.

ITU-T. Security architecture for open systems interconnection for CCITT applications. Recommendation x.800. p. 48. 1991.

JOSHI, J.; GHAFOOR, A.; AREF, W.G.; SPAFFORD, E.H. Digital government security infrastructure design challenges. **IEEE Computer**, v. 34, n. 2, p. 66–72, 2001a.

JOSHI, J.; AREF, W.G.; GHAFOOR, A.; SPAFFORD, E.H. Security models for web-based applications. **Communications of the ACM**, v. 44, n. 2, p. 38–44, 2001b.

KAMBOURAKIS, G.; KONTONI, D.P.N.; ROUSKAS, A.; GRITZALIS, S. A PKI approach for deploying modern secure distributed e-learning and m-learning environments. **Computers & Education**, v. 48, n. 1, p. 1–16, 2007.

KLEIJNEN, S.; RAJU, S. An Open Web Services Architecture. **ACM Queue**, v. 1, n.1, p. 38–46, 2003.

KREGER , H. **Web services conceptual architecture (WSCA 1.0)**. IBM Software Group, 2001. Disponível em: <http://www.cs.uoi.gr/~zarras/mdw-ws/WebServicesConceptualArchitectu2.pdf>. Acesso em: 15 de Maio de 2009.

LIN, N.; KORBA, L.; YEE, G.; SHIH, T.K.; LIN, H.W. Security and privacy technologies for distance education applications. In: 18TH INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS (AINA), 2004. v. 1. **Proceedings…** Fukuoka, Japan, 2004. p. 580–585.

LIOY, A.; G. RAMUNNO. Multiple Electronic Signatures on Multiple Documents. In: 1th INTERNATIONAL WORKSHOP ON ELECTRONIC GOVERNMENT AND COMMERCE: DESIGN, MODELING, ANALYSIS AND SECURITY (EGCDMAS), 2004. **Proceedings...** Setubal, Portugal, 2004, p. 24-34.

MENEZES, A.; OORSCHOT, P. V.; VANSTONE, S. **Handbook of Applied Cryptography**. Estados Unidos: CRC Press. 1997. 816p.

MICROSYSTEMS, S. **Java Remote Method Invocation Specification**. Sun Microsystems, Palo Alto, CA, 2006. Disponível em: <http://java.sun.com/javase/6/docs/technotes/guides/rmi/index.html>. Acesso em: 29 de Maio de 2009.

MITRA, N. e LAFON, Y. **SOAP Version 1.2 Part 0: Primer (Second Edition)**. W3C Recommendation, 2007. Disponível em: <http://www.w3.org/TR/soap12-part0/>. Accesado em: 19 de Junho de 2008.

MOODLE. Modular Object-Oriented Dynamic Learning Environment. Disponível em: < http://moodle.org/ >. Acessado em: 18 de Junho de 2008.

NADALIN, A.; KALER, C.; MONZILLO, R.; HALLAM-BAKER, P. **Web Services Security: SOAP Message Security 1.1**. OASIS Standard Specification, 2006. Disponível em: <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>. Acessado em: 19 de Junho de 2008.

NCSC. Department of Defense Trusted Computer Systems Evaluation Criteria. DoD National Computer Security Center. DoD 5200.28-STD, 1985.

OECD. Organization for Economic Co-Operation and Development. Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. September, 1980. Disponível em: <http://www.oecd.org/document/18/0,3343,en_2649_201185_1815186_1_1_1_1,00.html>. Acessado em Agosto de 2008.

ONIEVA, J. A.; ZHOU, J.; LOPEZ, J. Multiparty non-repudiation: A survey. **ACM Computing Surveys**. v. 41, n. 1, p. 1-43, 2008.

OPENCA. OpenCA PKI Research Labs. Disponível em: <http://www.openca.org/>. Acessado em 13 de Abril de 2009.

OPENSSL. Implementação open source dos protocolos SSL e TLS. Disponível em: <http://www.openssl.org/>. Acessado em 15 de Abril de 2009.

OSBORN, S.; SANDHU, R.; MUNAWER, Q. Configuring role-based access control to enforce mandatory and discretionary access control policies. **ACM Transactions on Information and System Security**, New York, NY, USA. v. 3, n. 2, p. 85–106, 2000.

RABUZIN, K.; BACA, M.; SAJKO, M. E-learning: Biometrics as a Security Factor. In: INTERNATIONAL MULTI-CONFERENCE ON COMPUTING IN GLOBAL INFORMATION TECHNOLOGY (ICCGI), 2006. **Proceedings…** Bucharest, Romênia, 2006, p. 1-6.

RAITMAN, R.; NGO, L.; AUGAR, N.; ZHOU, W. Security in the Online E-Learning Environment. In: FIFTH IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES (ICALT), 2005. **Proceedings…** Kaohsiung, Taiwan, 2005. p. 702–706.

ROY, J.; RAMANUJAN, A. **Understanding web services**. **IT Professional**, IEEE Computer Society, Los Alamitos, CA, USA, v. 3, n. 6, p. 69–73, 2001.

SAKAI: Collaborative and Learning Environment for Education. Disponível em: <http://bugs.sakaiproject.org/confluence/display/DOC/Release+Documentation>. Acessado em 06 de Maio de 2008.

SANDHU, R. On five definitions of data integrity. In IFIP WG11.3 Working Conference on Database Security VII. **Proceedings…** Workshop on Database Security, 1993.

SANDHU, R.; COYNE, E.J.; FEINSTEIN, H.L.; YOUMAN, C.E. **Role-based access control models**. IEEE Computer. v. 29, n. 2, p. 38–47, Fevereiro, 1996.

SANDHU, R.; MUNAWER, Q. How to do discretionary access control using roles. In: RBAC '98: THIRD ACM WORKSHOP ON ROLE-BASED ACCESS CONTROL, 1998. **Proceedings...** Fairfax, Virginia, USA, 1998. p. 47–54.

SAXEMA, R. Security and online content management: balancing access and security. In: 12TH BIENNIAL CONFERENCE AND EXHIBITION, **Victorian Association for Library Automation (VALA)**, Melbourne, Australia. 2004.

SOAPUI. **The Web Services Testing tool**. Disponível em: <http://www.soapui.org/>. Acessado em 12 de Junho de 2009.

STAL, M. Web services: beyond component-based computing. **Communications of the ACM**, v. 45, n. 10, p. 71-76, Outubro. 2002.

STALLINGS, W. **Network Security Essentials: Applications and Standards**. Prentice Hall. 2007. p. 432.

STALLINGS, W. **Cryptography And Network Security: Principles and Practice**. Prentice Hall. 2006. p. 680.

WEBBER, C.; MARIA DE FATIMA, W.P.; LIMA, M.E.; RIBEIRO, A.M. Adding Security to a Multiagent Learning Platform. In: FIRST INTERNATIONAL CONFERENCE ON AVAILABILITY, RELIABILITY AND SECURITY (ARES), 2006. **Proceedings…** Vienna, Austria, 2006, p. 887–894.

WEBBER, C.; MARIA DE FATIMA, W.P.; CASA, M.E.; RIBEIRO, A.M. Towards Secure e-Learning Applications: a Multiagent Platform. **Journal of Software**. v. 2. n. 1. p. 1-10, 2007

WEERAWARANA, S.; CURBERA, F.; LEYMANN, F.; STOREY, T.; FERGUSON, D.F. **Web Services Platform Architecture: Soap, WSDL, WS-Policy, WS-Addressing, WS-**

Bpel, WS-Reliable Messaging and More. Prentice Hall Professional Technical Reference. 2005. p. 416.

YEE, G.; KORBA, L. The Negotiation of Privacy Policies in Distance Education. In: 14TH IRMA INTERNATIONAL CONFERENCE. **Proceedings…** Philadelphia, Pennsylvania, 2003, p. 18-21.

YEE, G.; KORBA, L. Security Personalization for Internet and Web Services. **International Journal of Web Services (IJWSR)**, v. 5, n. 1, p. 1-23, 2007.

ZHOU, J.; GOLLMANN, D. Evidence and non repudiation. **Journal of Network and Computer Applications**, v. 20, n. 3, pp. 267-281. 1997.

Apêndice A – Processo de Criação de uma Aplicação nos Ambientes de Ensino Colaborativo Sakai e Moodle

Este apêndice descreve o processo para a criação de uma aplicação nos ambientes de ensino colaborativo Sakai e Moodle que contenham o serviço de *upload* de arquivo.

Criação de uma aplicação no ambiente Sakai

O Sakai é um sistema de gerenciamento de aprendizado e trabalho cooperativo suportado por computador. Seu principal objetivo foi criar um sistema aberto, ao qual pudessem ser incorporadas diversas ferramentas.

A plataforma Sakai inclui a maioria das funcionalidades existentes num Sistema de Gestão de Aprendizagem (SGA), como gestão e distribuição de conteúdos, fóruns de discussão, salas de bate-papo, *upload* de trabalhos, exercícios *online*, entre outros.

Além dessas funcionalidades, a plataforma de aprendizagem colaborativa Sakai dispõe de suporte para o desenvolvimento de pesquisas e projetos, através de um conjunto de ferramentas, o qual inclui listas de distribuição de correio eletrônico, *wikis*, arquivos, e até mesmo leitores de RSS (*Rich Site Summary*).

Para o estudo de caso descrito no capítulo 6, foi instalada a versão 2.4 do Sakai. Esta versão é sustentada pelas tecnologias: Java SE 1.5.0, Tomcat 5.5.23, Maven v1.0.2 e MySQL 4.1.12.

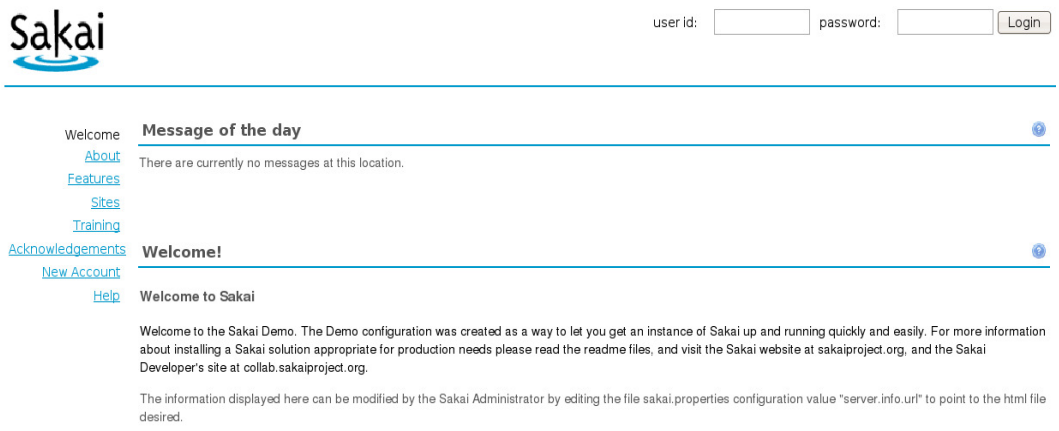

Figura 29 - Interface inicial do Sakai.

A seguir, serão mostrados todos os procedimentos necessários para se criar uma aplicação no Sakai. A Figura 29 ilustra a página principal do Sakai utilizada para os usuários obterem informações gerais do ambiente e, também, "*logar*" no sistema.

Para se criar uma aplicação no Sakai, foi necessário "entrar" no ambiente como administrador. A Figura 30 ilustra a página do administrador Sakai. Esta página possui todas as informações necessárias para gerenciar todo o ambiente Sakai, desde os usuários e aplicações já existentes, como também a criação de novos usuários e aplicações no sistema.

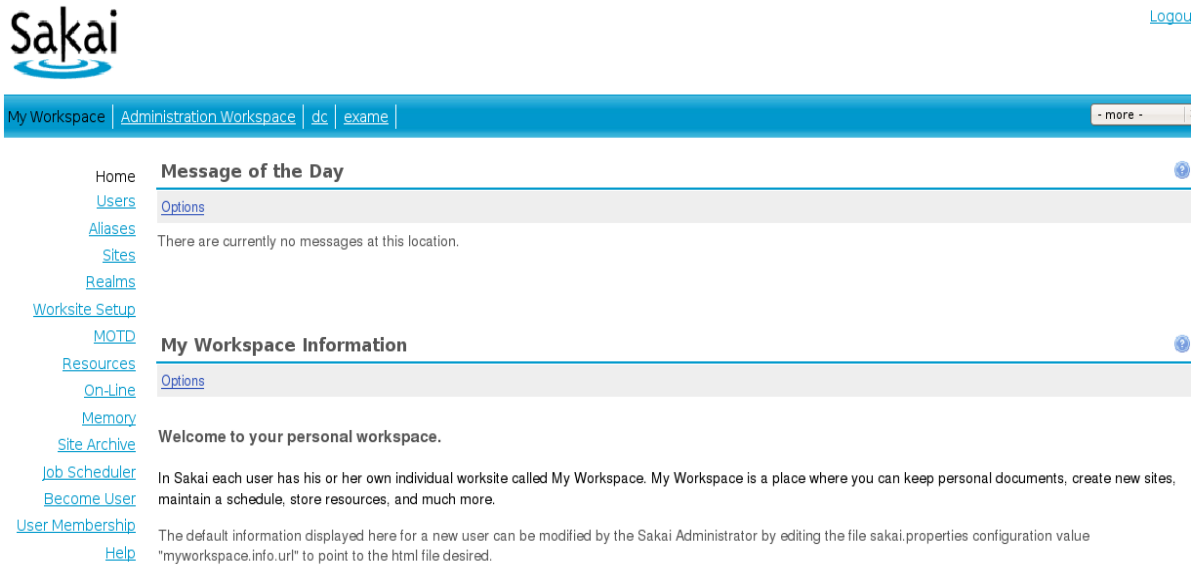

Figura 30 - Interface inicial do Administrador Sakai.

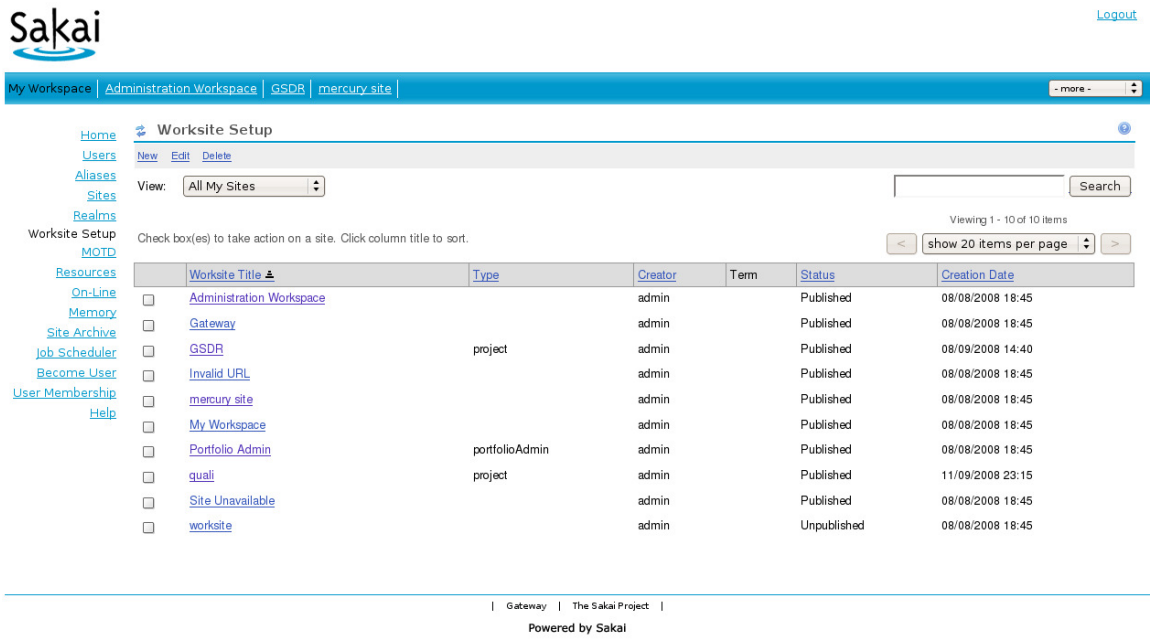

Figura 31 - Interface com as aplicações criadas no ambiente Sakai.

Através do *link* “Worksite Setup” é iniciado o processo de criação de novas aplicações no ambiente Sakai. O conteúdo deste *link* é ilustrado na Figura 31. Nesta página, é possível criar, editar e apagar aplicações. Nela, também são listadas todas as aplicações que já estão criadas no ambiente Sakai, como as aplicações GSDR, quali, Portifolio Admin, entre outras.

Após clicar em *New*, para criar uma nova aplicação, o usuário é direcionado para a página ilustrada na Figura 32. Nesta página, é definido o tipo de aplicação que se deseja criar, “*course website*” ou “*project website*”. Para este trabalho, foi definido “*Project website*”, uma vez que o outro tipo de aplicação estava indisponível na distribuição do Sakai utilizada.

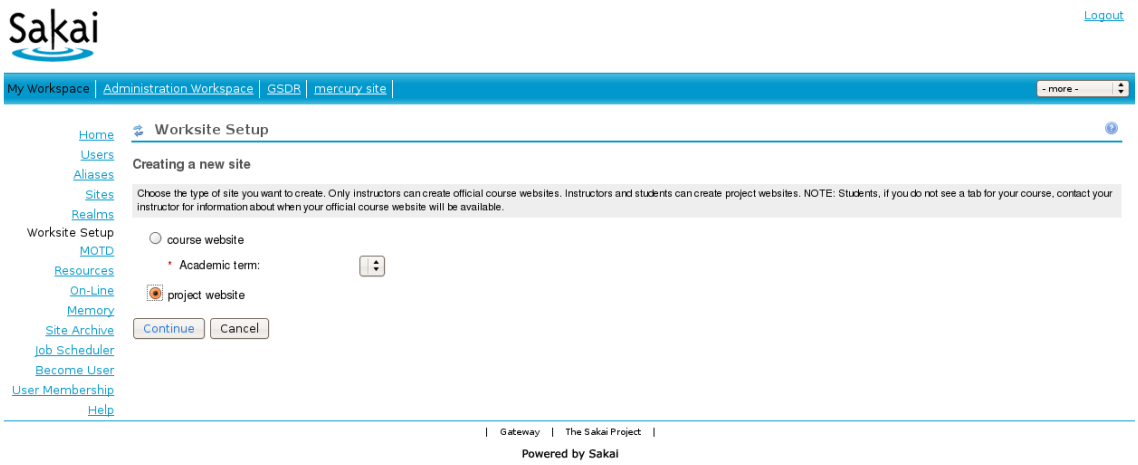

Figura 32 - Interface com os tipos de aplicações que podem ser criadas.

Depois de escolher o tipo de aplicação, foi necessário definir as informações básicas da aplicação, como seu título e sua descrição. O nome da aplicação foi definido como INCA e, em sua descrição, foi feita uma explicação de que ela seria utilizada para o estudo de caso com serviço apresentado neste trabalho. A Figura 33 ilustra a página que possui essas informações.

Definindo o nome e a descrição da aplicação, é o momento de escolher as ferramentas que estarão presentes na aplicação. Na Figura 34, são ilustradas as ferramentas que podem ser fornecidas pelo sistema Sakai, como por exemplo, *Blogger*, *Email Archieve*, *Evaluations*, Portfólios, *Fórum*, *Chat Room*, *Drop Box*, entre outros. Para a realização do estudo de caso, foi criada uma aplicação com a ferramenta *Podcasts* – que oferece o serviço de *upload* de arquivo e que é utilizada para o compartilhamento de arquivos entre os usuários da aplicação.

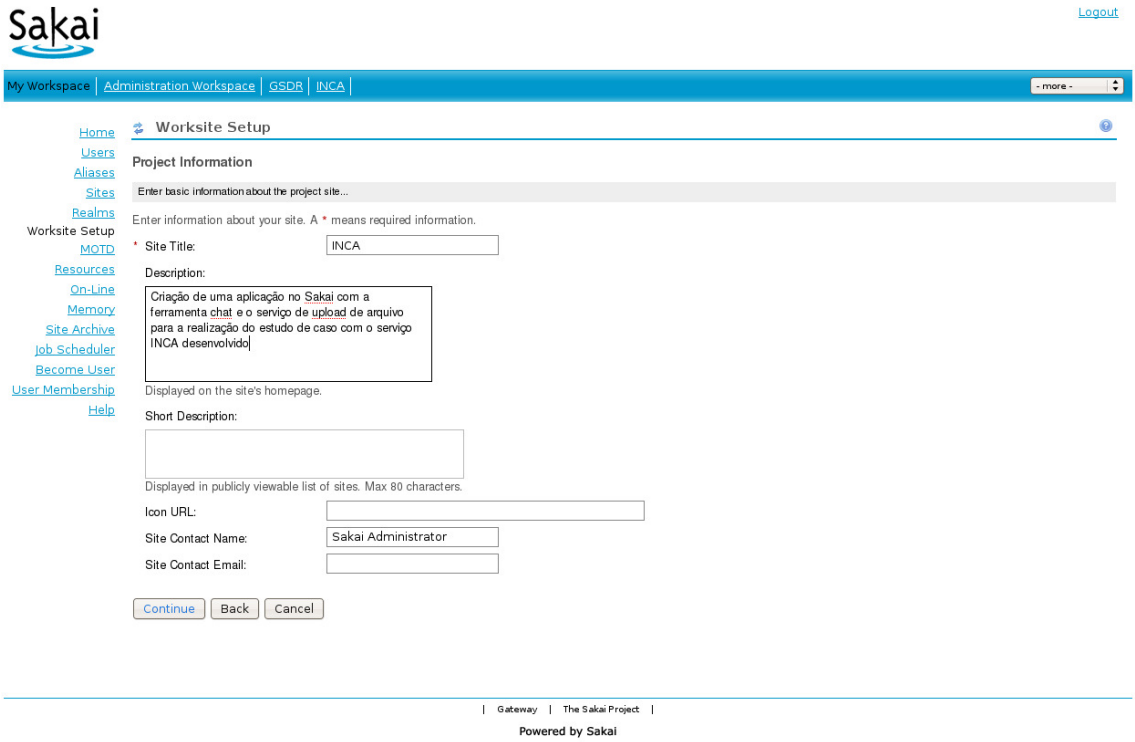

Figura 33 - Interface para a definição de informações básicas da aplicação.

Figura 34 - Interface com as ferramentas oferecidas pelo Sakai.

O último passo para a criação da aplicação – ilustrado na Figura 35 – foi definir as opções de acesso à aplicação.

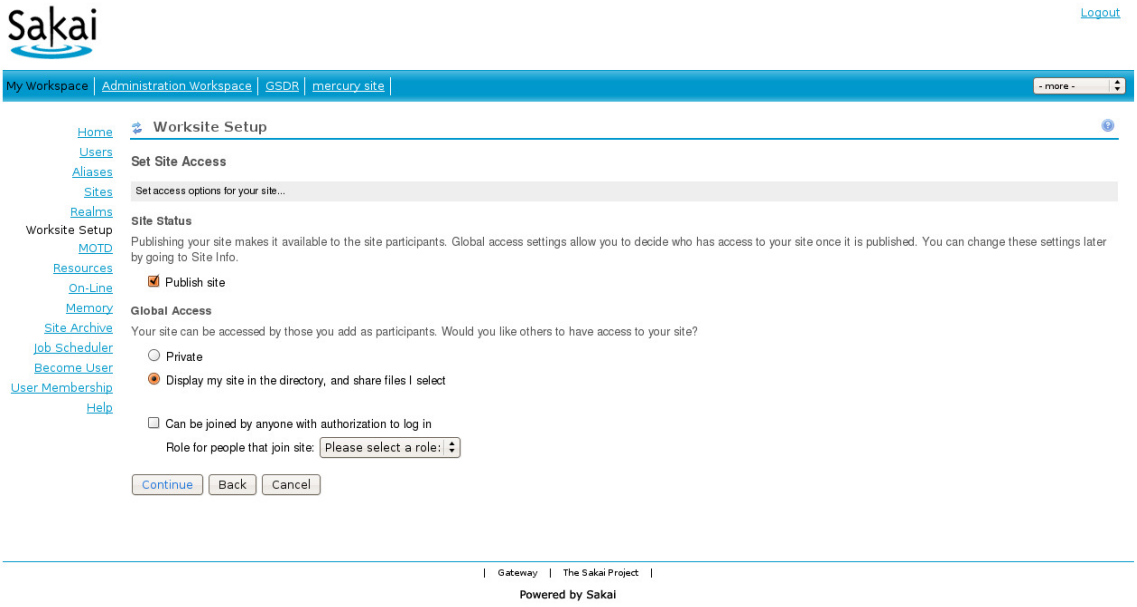

Figura 35 - Interface com opções de acesso à aplicação.

Após a definição de como será o acesso à aplicação, é ilustrada, na Figura 36, a página de confirmação das configurações da aplicação. Esta página possui informações, como o nome e descrição da aplicação, as ferramentas presentes, entre outras.

Por fim, é só "clicar" no botão *Create Site* para que a aplicação seja criada.

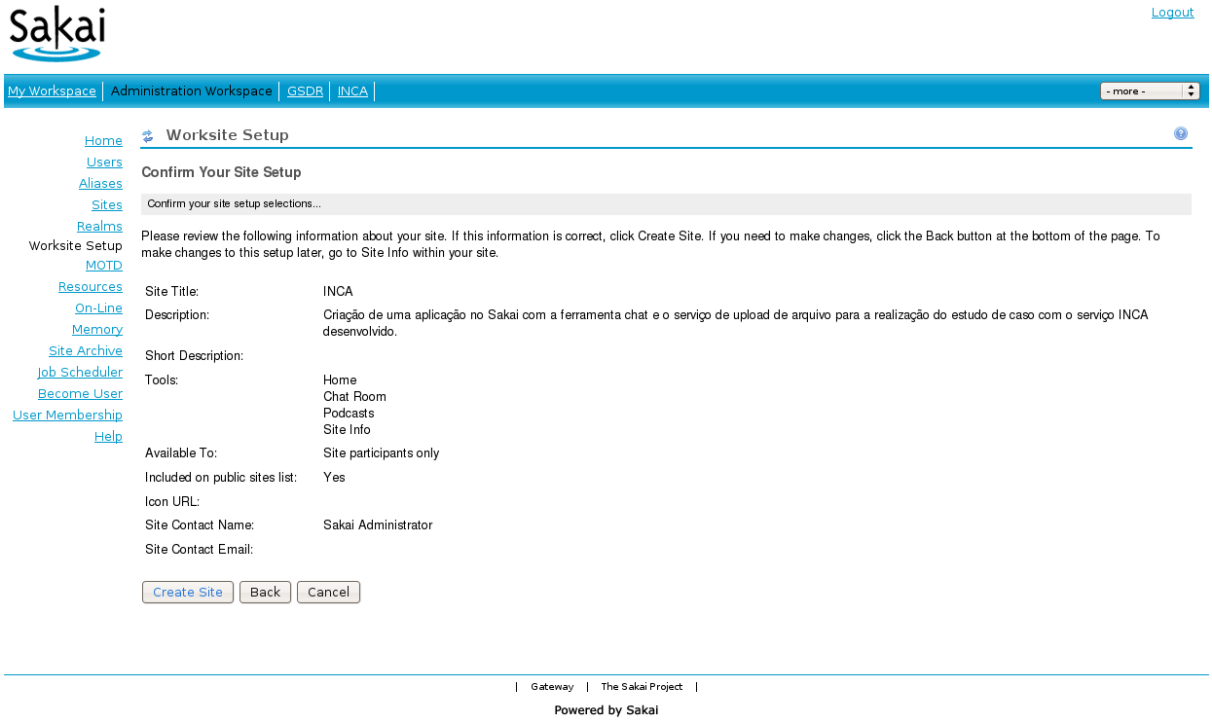

Figura 36 - Interface para confirmação das configurações definidas na criação da aplicação.

Depois de criar a aplicação, é possível visualizar na página com a lista das aplicações já criadas no Sakai (Figura 37) que a aplicação INCA já está disponível para ser utilizada.

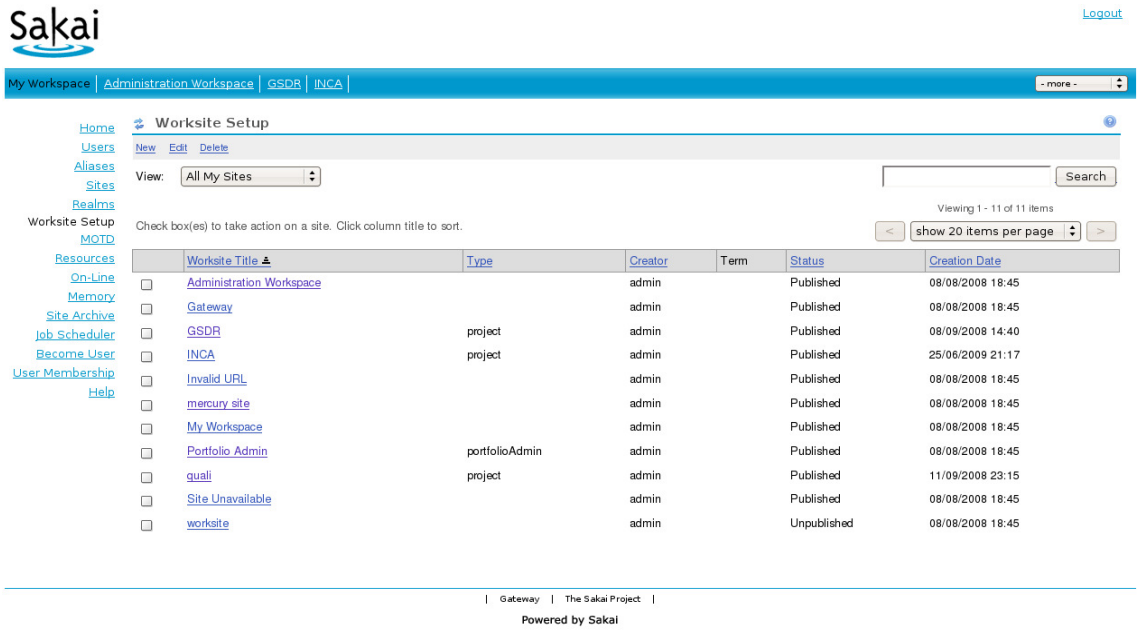

Figura 37 - Interface com a aplicação INCA dentro do ambiente Sakai.

Na Figura 38, é ilustrada a ferramenta *Podcasts*. Nesta ferramenta, é possível adicionar arquivos, criar pasta, adicionar links da *Web* e criar documentos.

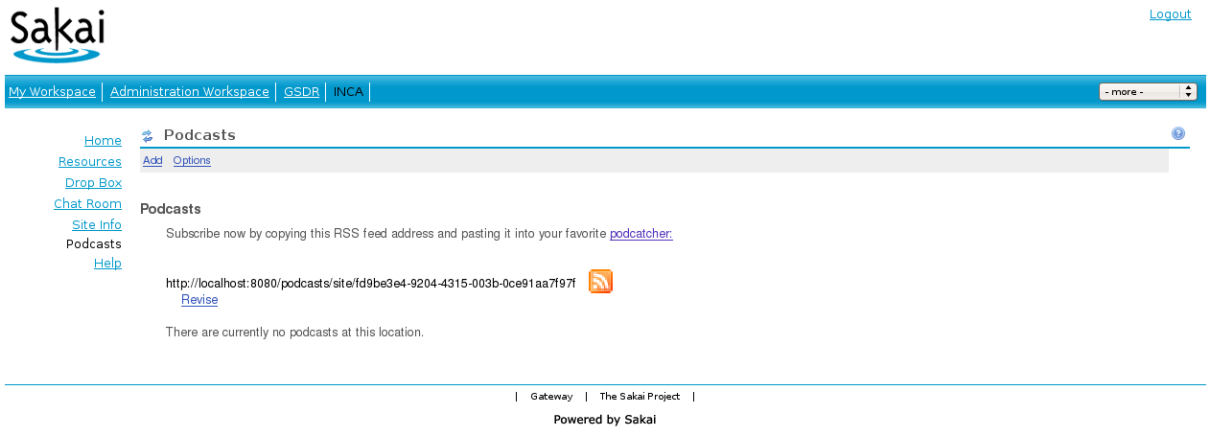

Figura 38 - Interface da ferramenta *Podcasts* da aplicação criada.

Na Figura 39, é ilustrada a interface para fazer *upload* de arquivos na ferramenta *Podcasts*. O usuário pode selecionar o arquivo, a data de publicação, o título do arquivo e, caso necessário, uma descrição do objetivo pelo o qual o arquivo está disponível no sistema.

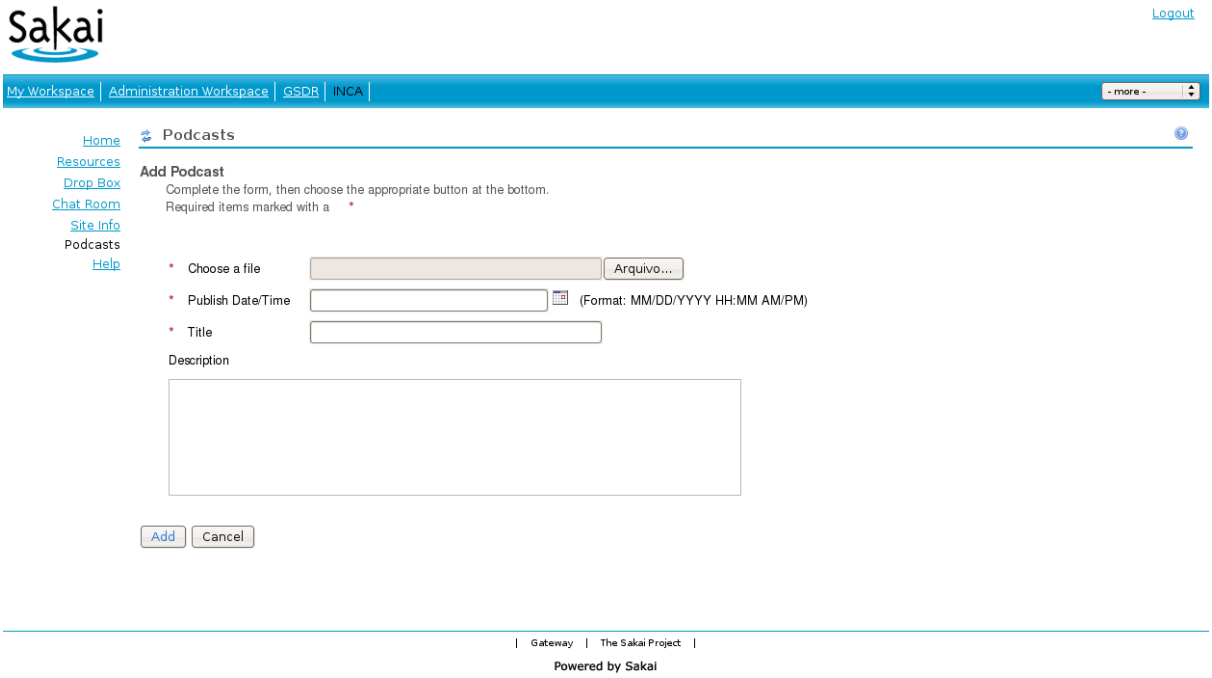

Figura 39 – Interface para fazer *upload* de arquivos na ferramenta *Podcasts*.

Na próxima subseção, é descrito o processo de criação de uma aplicação no ambiente de ensino colaborativo Moodle.

Criação de uma aplicação no ambiente Moodle

O Moodle (*Modular Object-Oriented Dynamic Learning Environment*) é um ambiente de ensino colaborativo desenvolvido em PHP, para permitir a criação e a administração de cursos na *Web*. Por ser um sistema de código aberto, livre e gratuito, este sistema pode ser baixado, utilizado, modificado e distribuído seguindo os termos estabelecidos pela licença *General Public License* (GPL).

Instituições de ensino estão adaptando o Moodle a seus próprios conteúdos, não apenas para cursos totalmente a distância, mas também como apoio aos cursos presenciais. Ainda no contexto educacional, o Moodle vem sendo utilizado para outras atividades, como treinamento de professores, formação de grupos de estudo e, até mesmo, para o desenvolvimento de projetos. Não se restringindo a este contexto, o Moodle tem sido utilizado por empresas privadas, ONGs e grupos independentes, com o objetivo de interagir de maneira colaborativa na *Web*.

Este sistema de aprendizado eletrônico possui ferramentas de comunicação, avaliação, disponibilidade, administração e organização de conteúdos. Os recursos disponíveis para o desenvolvimento das atividades neste ambiente são: *Chat*, Avaliação do Curso, Diário, Fórum, *Wiki*, *upload* de arquivos, entre outros.

Para o presente estudo de caso foi instalada a versão 1.8 do Moodle. Esta versão é sustentada pelas tecnologias Apache2, PHP 5 e MySQL 5.

A seguir, serão mostrados todos os procedimentos necessários para se criar uma aplicação no Moodle. A Figura 40 ilustra a página principal do Moodle, onde são listadas as aplicações que estão disponíveis.

Figura 40 - Interface inicial do Moodle.

Na Figura 41, é ilustrada a página de *login* do Moodle. Para que se possa criar uma aplicação neste ambiente foi necessário "logar" como administrador do sistema.

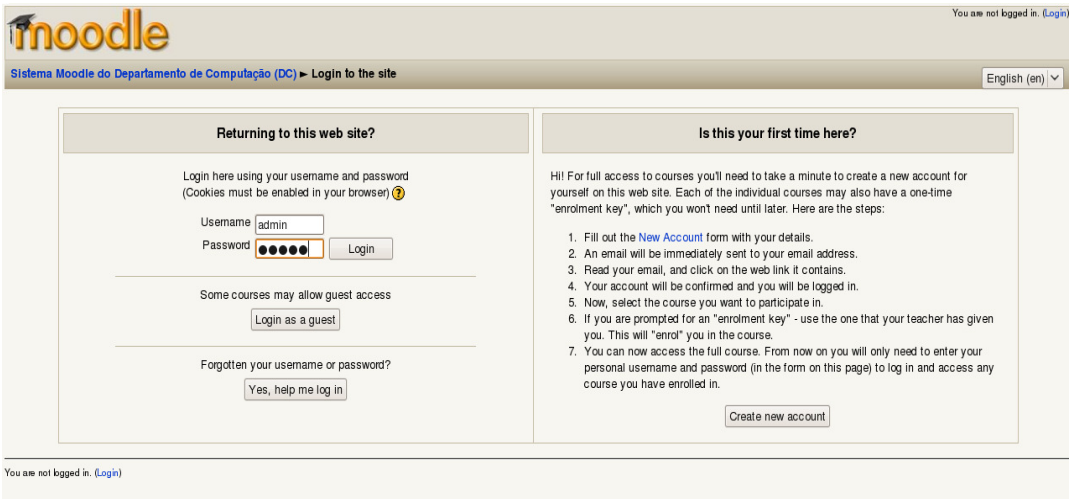

Figura 41 - Interface de *login* do Moodle.

A Figura 42 ilustra a interface inicial do Administrador Moodle. Como administrador, foi possível criar novas aplicações, adicionar/editar usuários, definir a aparência do ambiente, configurar módulos do sistema, entre outros.

Figura 42 - Interface inicial do Administrador Moodle.

Após clicar em adicionar cursos, o sistema foi direcionado para a página ilustrada na Figura 43. Nesta página, foi possível definir a categoria da aplicação que, no contexto deste estudo de caso, foi definida como "Mestrado". Na Figura 44, foi possível perceber a existência da nova categoria.

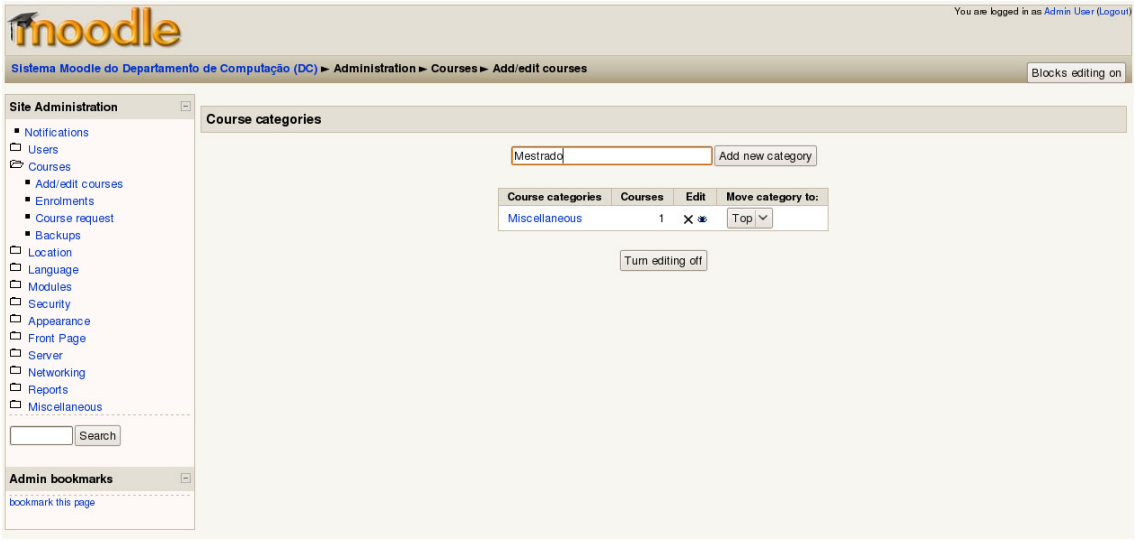

Figura 43 - Interface para a definição da categoria da aplicação.

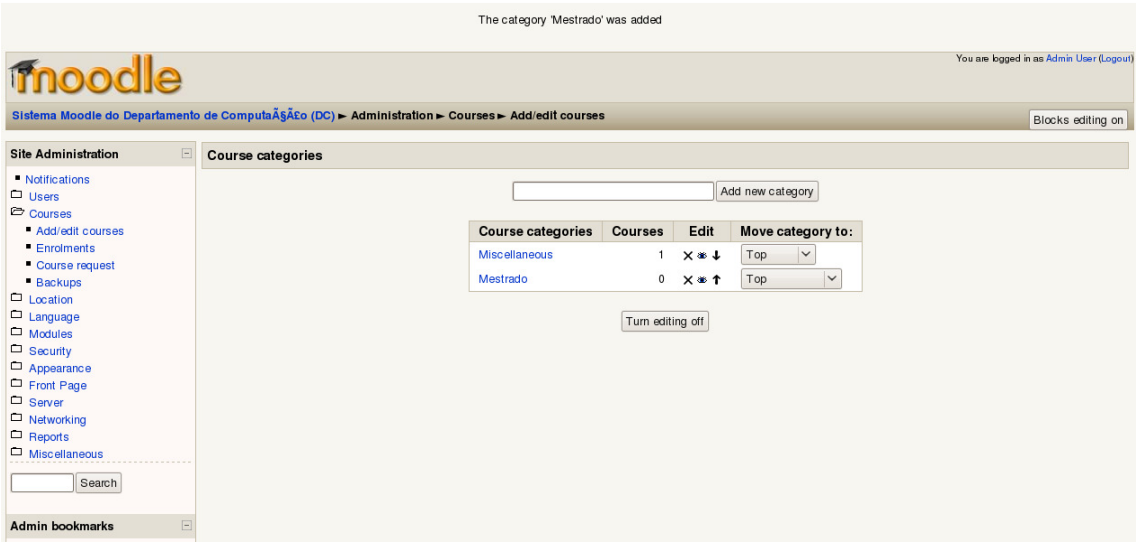

Figura 44 - Interface com a categoria mestrado criada.

Após definir a nova categoria, o sistema Moodle foi direcionado para a página ilustrada na Figura 45, onde foi possível iniciar o processo para a criação de uma aplicação para a categoria “Mestrado”.

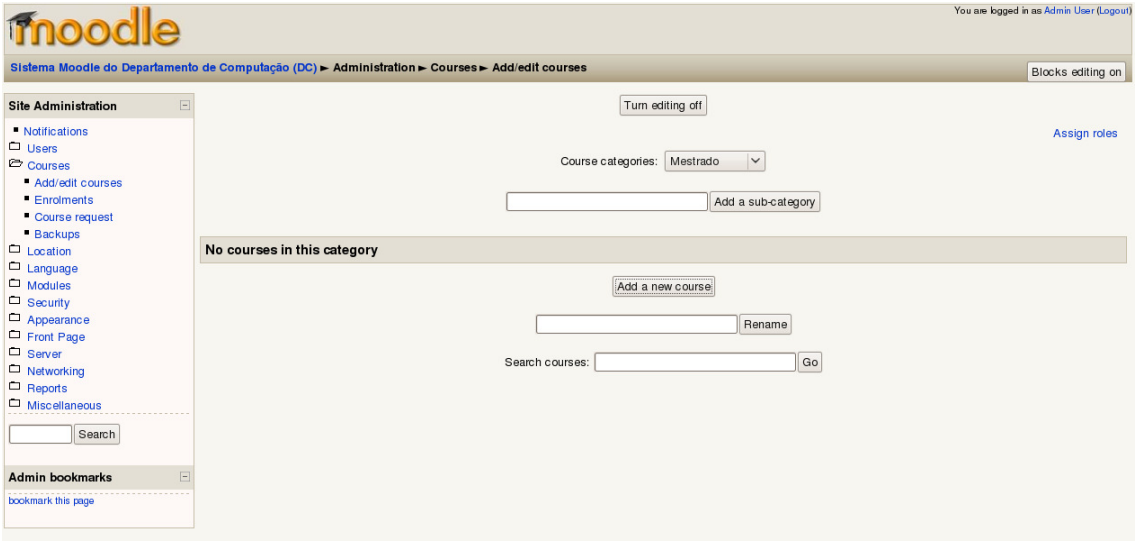

Figura 45 - Interface para a criação de uma aplicação.

A interface de configuração da aplicação a ser criada é ilustrada na Figura 46. Nesta página, foi possível definir o nome completo da aplicação (Estudo de Caso Moodle), nome curto (Teste - INCA), ID da aplicação, formato da aplicação, número de semanas, tamanho máximo dos arquivos utilizados para *upload* no sistema, entre outros. O tipo de registro da aplicação, a definição da possibilidade de criação de grupos de usuários e o público

alvo da aplicação, isto é, para quem ela está sendo criada, também pode ser definido nesta página.

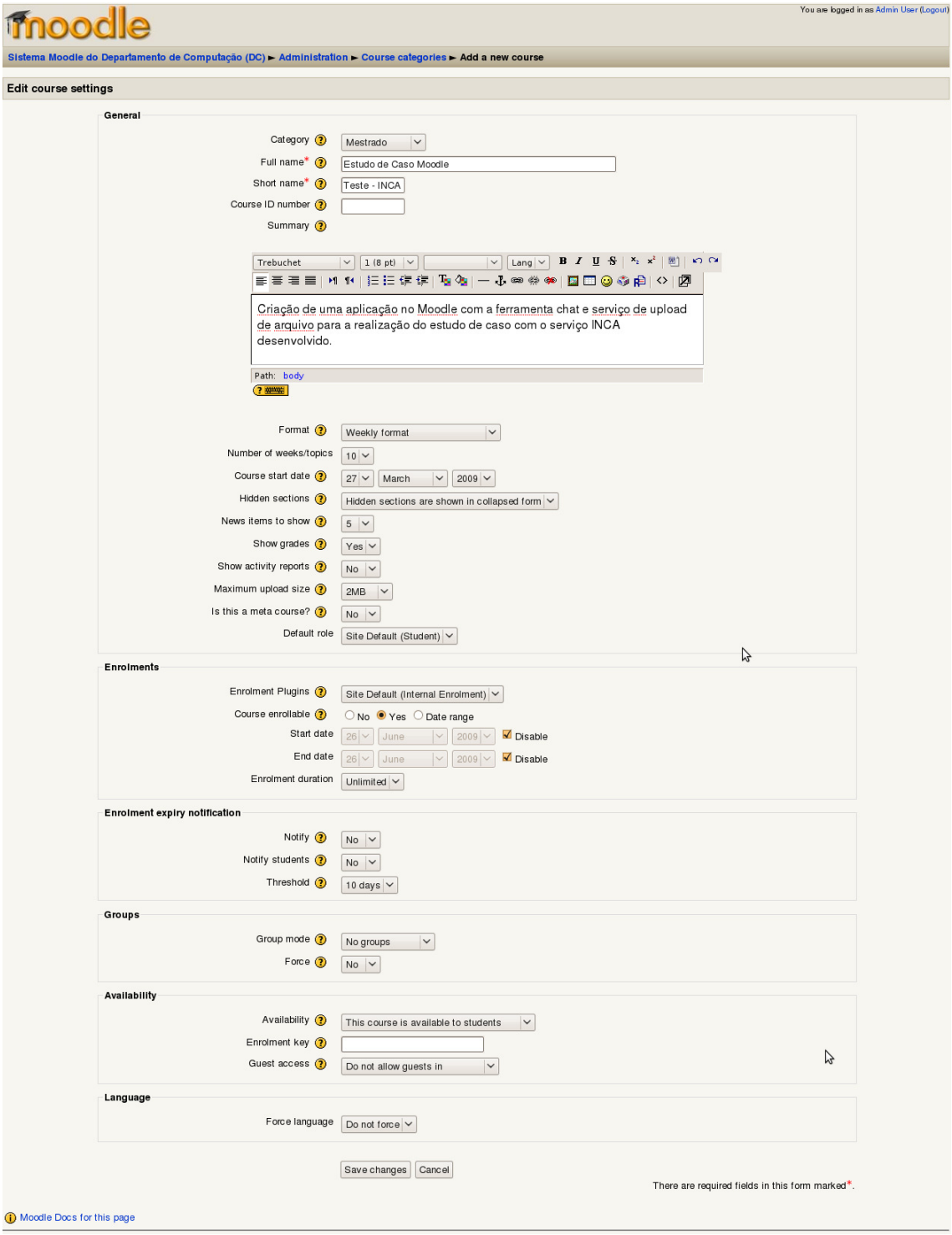

Figura 46 - Interface de configuração da aplicação a ser criada.

Após salvar as configurações ilustradas na figura anterior, o sistema é direcionado para uma interface, ilustrada na Figura 47, para que se possa atribuir as regras aos parti-

cipantes da aplicação. Nesta página, é possível definir as regras do administrador, criador do curso, professor, tutor, aluno e visitante.

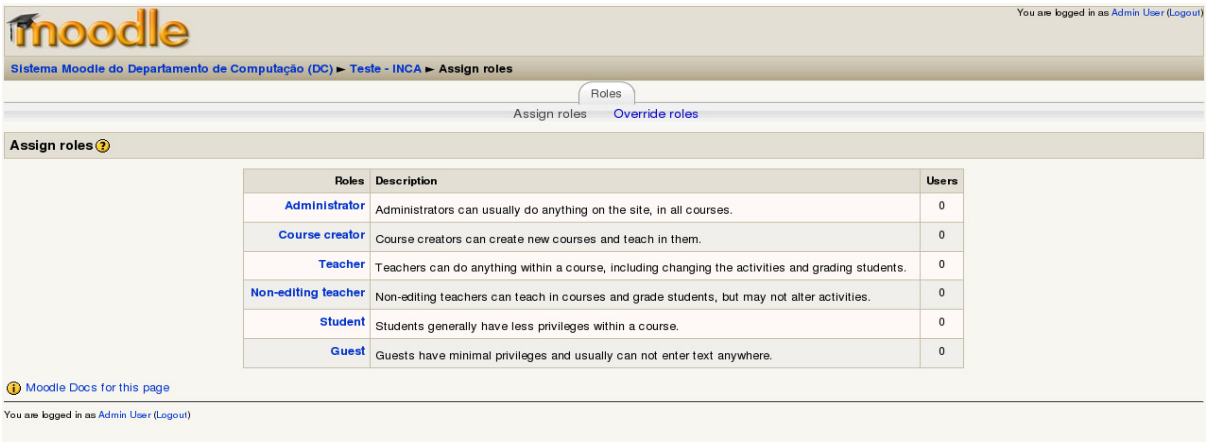

Figura 47 - Interface para atribuir regras aos participantes da aplicação.

Depois de definir as regras dos participantes da aplicação, é ilustrada, na Figura 48, a página principal da aplicação criada. Nesta página, é possível visualizar os participantes da mesma aplicação, ter acesso às notícias configuradas pelo administrador ou pelo responsável da aplicação, definir a criação de grupos de usuários dentro da aplicação, criar questionários para a avaliação do curso ou do próprio sistema, fazer *backup* da aplicação, entre outras ações.

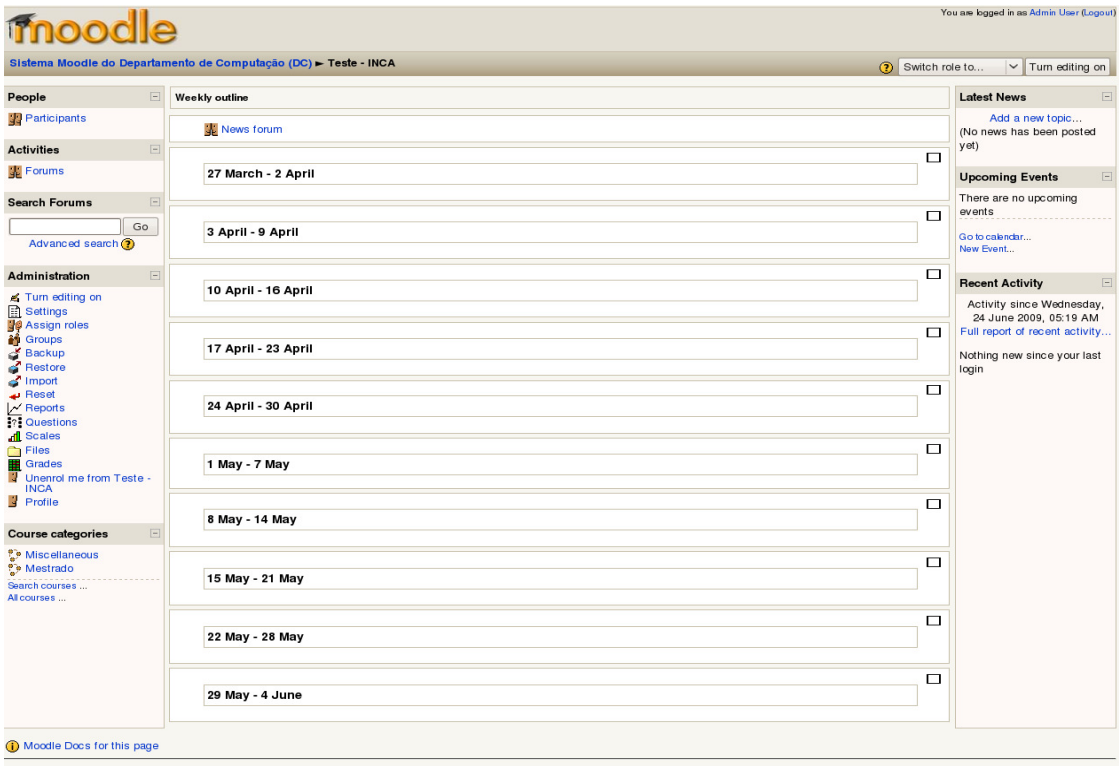

Figura 48 - Interface da aplicação Teste-INCA criada.

Ao contrário do ambiente Sakai, em que o administrador pode definir as ferramentas que estarão presentes na aplicação durante seu processo de criação, uma aplicação criada no Moodle não possui, de início, ferramentas como *chat*, *upload* de arquivos, fórum, entre outras.

Todas essas ferramentas, citadas anteriormente, estão disponíveis para qualquer que seja a aplicação criada. Neste caso, fica a cargo do professor ativar uma ferramenta para que ela possa ser utilizada na aplicação.

Na Figura 49, é ilustrado o processo para adicionar a ferramenta de *upload* de arquivo. O nome de tal ferramenta, sua descrição, tamanho máximo dos arquivos que podem fazer *upload*, entre outras características, pode ser configurada nesta página.

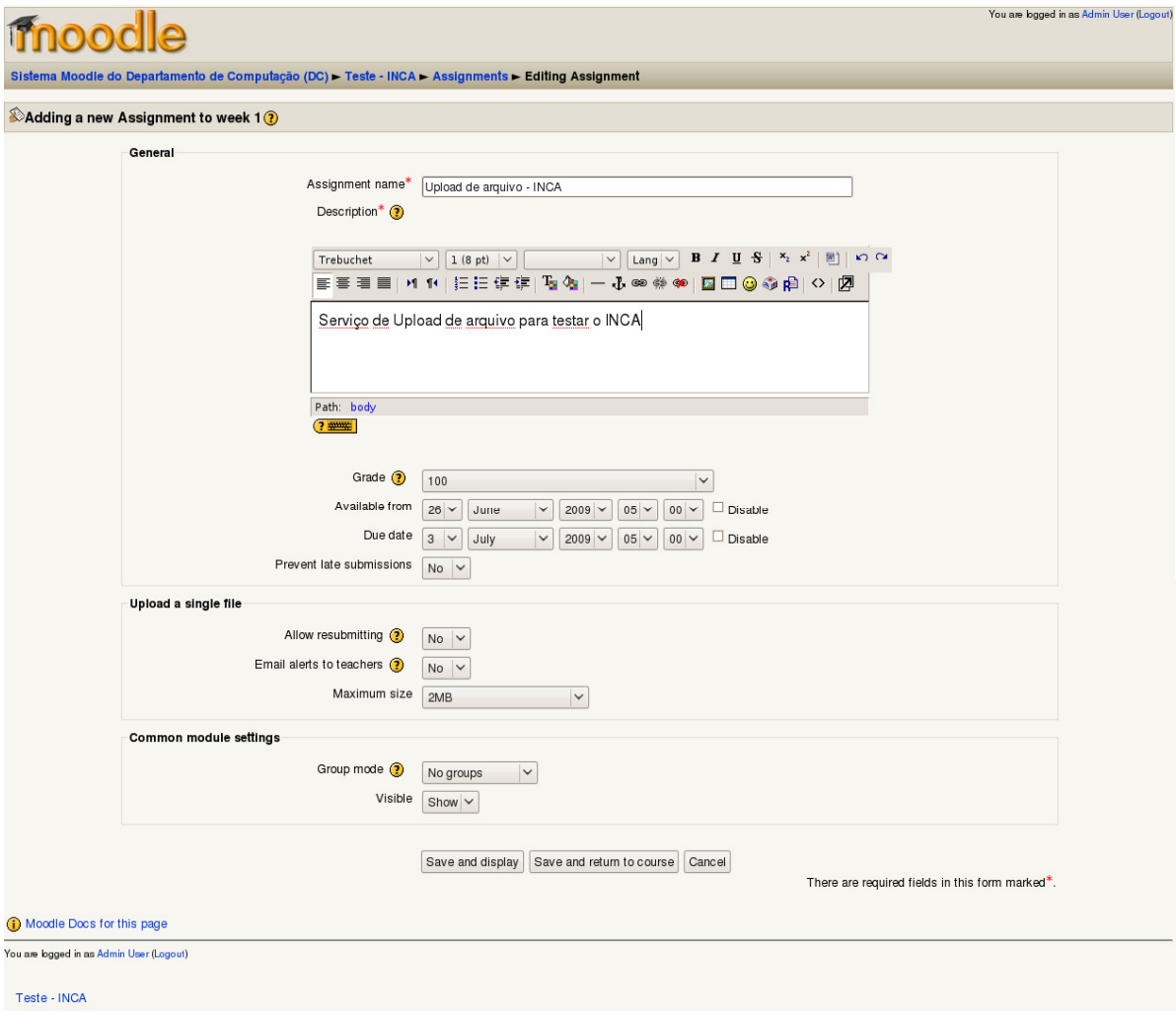

Figura 49 - Interface para adicionar a ferramenta de *upload* de arquivo à aplicação Teste - INCA.

Após adicionar a ferramenta de *upload* de arquivo, o sistema Moodle foi direcionado para a página da ferramenta, operação que é ilustrada pela Figura 50. Nela, os usuários podem selecionar um arquivo que está em seu computador e realizar seu *upload* no sistema.

Figura 50 - Interface para fazer *upload* de arquivo na aplicação Teste - INCA.

Na Figura 51, é ilustrada a página da ferramenta *upload* de arquivo com o arquivo IMPORTANTE.txt disponível no sistema.

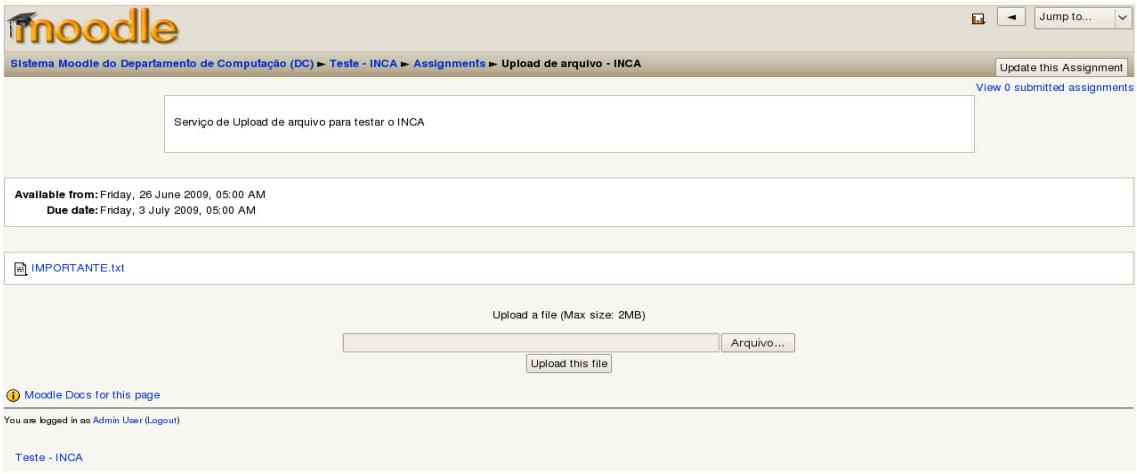

Figura 51 - Interface com o *upload* realizado com sucesso.

Após adicionar a ferramenta de *upload*, tem-se uma aplicação criada no ambiente Moodle para realizar o estudo de caso da utilização de tais ferramentas do serviço INCA apresentado neste trabalho. A página principal da aplicação com as duas ferramentas criadas é ilustrada na Figura 52.

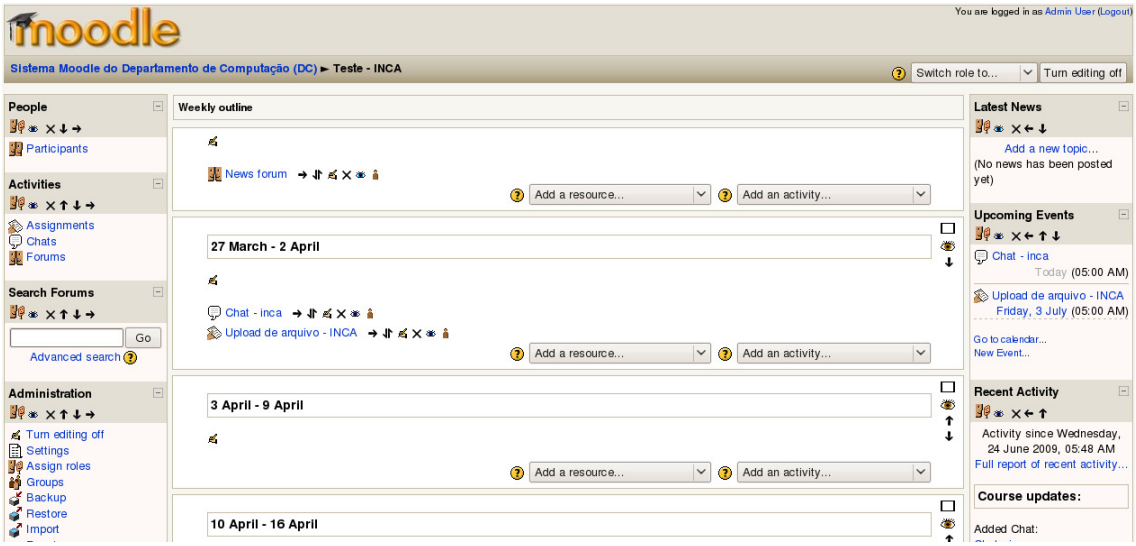

Figura 52 - Interface com a presença da ferramenta de *upload* de arquivo na aplicação Teste - INCA.

No Apêndice A, foram apresentados os procedimentos necessários para a criação de uma aplicação nos ambientes de ensino colaborativos Sakai e Moodle que contenham a ferramenta de *upload* de arquivo.

Apêndice B – Código do WSDL do INCA

Apresenta-se, a seguir, o código do arquivo WSDL do INCA. Informações como o *namespace* do serviço, os nomes dos métodos do INCA, os tipos de dados dos seus respectivos argumentos, o nome das mensagens de entrada e saída de cada método, dentre outras, são representadas neste arquivo.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--
 Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-
WS RI 2.1.3.1-hudson-417-SNAPSHOT.
-->

<definitions targetNamespace="http://SecurityServices/"
name="INCAv2Service">

      <types>

            <xsd:schema>
            <xsd:import namespace="http://SecurityServices/" schemaLoca-
tion="http://localhost:8080/ServidorServicos-v2/INCAv2Service?xsd=1"/>
            </xsd:schema>
      </types>

      <message name="codifica">
            <part name="parameters" element="tns:codifica"/>
      </message>

      <message name="codificaResponse">
            <part name="parameters" element="tns:codificaResponse"/>
      </message>

      <message name="Exception">
            <part name="fault" element="tns:Exception"/>
      </message>

      <message name="decodifica">
            <part name="parameters" element="tns:decodifica"/>
      </message>

      <message name="decodificaResponse">
            <part name="parameters" element="tns:decodificaResponse"/>
      </message>

      <message name="criaAD">
            <part name="parameters" element="tns:criaAD"/>
      </message>

      <message name="criaADResponse">
            <part name="parameters" element="tns:criaADResponse"/>
      </message>

      <message name="verificaAD">
            <part name="parameters" element="tns:verificaAD"/>
```

```
	</message>

	<message name="verificaADResponse">
		<part name="parameters" element="tns:verificaADResponse"/>
	</message>

	<message name="cifraSimetrico">
		<part name="parameters" element="tns:cifraSimetrico"/>
	</message>

	<message name="cifraSimetricoResponse">
		<part name="parameters" element="tns:cifraSimetricoResponse"/>
	</message>

	<message name="decifraSimetrico">
		<part name="parameters" element="tns:decifraSimetrico"/>
	</message>

	<message name="decifraSimetricoResponse">
		<part name="parameters" ele-
ment="tns:decifraSimetricoResponse"/>
	</message>

	<portType name="INCAv2">

		<operation name="codifica">
			<input message="tns:codifica"/>
			<output message="tns:codificaResponse"/>
			<fault message="tns:Exception" name="Exception"/>
		</operation>

		<operation name="decodifica">
			<input message="tns:decodifica"/>
			<output message="tns:decodificaResponse"/>
			<fault message="tns:Exception" name="Exception"/>
		</operation>

		<operation name="criaAD">
			<input message="tns:criaAD"/>
			<output message="tns:criaADResponse"/>
			<fault message="tns:Exception" name="Exception"/>
		</operation>

		<operation name="verificaAD">
			<input message="tns:verificaAD"/>
			<output message="tns:verificaADResponse"/>
			<fault message="tns:Exception" name="Exception"/>
		</operation>

		<operation name="cifraSimetrico">
			<input message="tns:cifraSimetrico"/>
			<output message="tns:cifraSimetricoResponse"/>
			<fault message="tns:Exception" name="Exception"/>
		</operation>

		<operation name="decifraSimetrico">
			<input message="tns:decifraSimetrico"/>
			<output message="tns:decifraSimetricoResponse"/>
			<fault message="tns:Exception" name="Exception"/>
		</operation>
	</portType>
```

```
      <binding name="INCAv2PortBinding" type="tns:INCAv2">
            <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>

            <operation name="codifica">
                  <soap:operation soapAction=""/>

                  <input>
                        <soap:body use="literal"/>
                  </input>

                  <output>
                        <soap:body use="literal"/>
                  </output>

                  <fault name="Exception">
                        <soap:fault name="Exception" use="literal"/>
                  </fault>
            </operation>

            <operation name="decodifica">
                  <soap:operation soapAction=""/>

                  <input>
                        <soap:body use="literal"/>
                  </input>

                  <output>
                        <soap:body use="literal"/>
                  </output>

                  <fault name="Exception">
                        <soap:fault name="Exception" use="literal"/>
                  </fault>
            </operation>

            <operation name="criaAD">
                  <soap:operation soapAction=""/>

                  <input>
                        <soap:body use="literal"/>
                  </input>

                  <output>
                        <soap:body use="literal"/>
                  </output>

                  <fault name="Exception">
                        <soap:fault name="Exception" use="literal"/>
                  </fault>
            </operation>

            <operation name="verificaAD">
                  <soap:operation soapAction=""/>

                  <input>
                        <soap:body use="literal"/>
                  </input>

                  <output>
```

```
                        <soap:body use="literal"/>
                  </output>

                  <fault name="Exception">
                        <soap:fault name="Exception" use="literal"/>
                  </fault>
            </operation>

            <operation name="cifraSimetrico">
                  <soap:operation soapAction=""/>

                  <input>
                        <soap:body use="literal"/>
                  </input>

                  <output>
                        <soap:body use="literal"/>
                  </output>

                  <fault name="Exception">
                        <soap:fault name="Exception" use="literal"/>
                  </fault>
            </operation>

            <operation name="decifraSimetrico">
                  <soap:operation soapAction=""/>

                  <input>
                        <soap:body use="literal"/>
                  </input>

                  <output>
                        <soap:body use="literal"/>
                  </output>

                  <fault name="Exception">
                        <soap:fault name="Exception" use="literal"/>
                  </fault>
            </operation>
      </binding>

      <service name="INCAv2Service">

            <port name="INCAv2Port" binding="tns:INCAv2PortBinding">
                  <soap:address loca-
tion="http://localhost:8080/ServidorServicos-v2/INCAv2Service"/>
            </port>
      </service>
</definitions>
```