UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ESTUDO E IMPLEMENTAÇÃO DE AUTORIZAÇÃO NO ACESSO PARA O PORTAL DO HUSM UTILIZANDO O CIBAC

TRABALHO DE GRADUAÇÃO

Leandro Sacchet

Santa Maria, RS, Brasil

2007

ESTUDO E IMPLEMENTAÇÃO DE AUTORIZAÇÃO NO ACESSO PARA O PORTAL DO HUSM UTILIZANDO O CIBAC

por

Leandro Sacchet

Trabalho de Graduação apresentado ao Curso de Ciência da Computação
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Raul Ceretta Nunes

Trabalho de Graduação Nº 229
Santa Maria, RS, Brasil

2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

ESTUDO E IMPLEMENTAÇÃO DE AUTORIZAÇÃO NO ACESSO PARA O PORTAL DO HUSM UTILIZANDO O CIBAC

elaborado por
Leandro Sacchet

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Prof. Dr. Raul Ceretta Nunes
(Presidente/Orientador)

Prof. Antônio Marcos de Oliveira Candia (UFSM)

Esp. Marcos Vinícius Bitencourt de Souza (UFSM)

Santa Maria, 01 de março de 2007.

Ao meu pai
***Domingos Sacchet**,*
à minha mãe
Ivani Teresinha Rosa Sacchet

AGRADECIMENTOS

Ao fim de mais essa etapa, gostaria de agradecer algumas pessoas que foram importantes durante este período. Agradeço:

Primeiramente a Deus, pela vida.

Ao meu amigo e orientador Professor Dr. Raul Ceretta Nunes, por tudo que fez por mim, além de orientar-me neste trabalho também estava sempre pronto à ajudar em outras tarefas inerentes a esta caminhada. Com certeza uma pessoa que marcou minha graduação, por toda sua compreensão e amizade.

Aos meus grandes amigos do CPD que colaboraram muito no decorrer deste trabalho: Eduardo Maikel Müller, Marcos Vinícius Bitencourt de Souza e César Augusto Guerra de Souza. Estavam sempre dispostos a ajudar e com certeza foram muito importantes no desenvolvimento do trabalho. Amigos que podem contar comigo sempre que precisarem.

Aos meus amigos de infância (Júnior, Maurício, Rodrigo, Felipe, Tiago, Daniel...) que não são muitos mas valem por uma multidão. São esses amigos que faço questão de compartilhar os melhores momentos da minha vida. Em fim, amigos que quero levar para sempre.

A minha família. Meu pai (Domingos Sacchet), personalidade forte, mas no fundo tem um coração mole e faz tudo por seus filhos. Te amo muito pai. Minha mãe (Ivani Teresinha Rosa Sacchet), como ser humano, o mais incrível que conheço e como mãe, não tenho nem palavras, simplesmente espetacular. Mãe eu te amo demais. Meus irmãos e melhores amigos (Paulo Sérgio Sacchet e Evanise Sacchet), pessoas que quero estar perto sempre. Manos eu amo muito vocês. Minha noiva (Vanesa de David), mulher pela qual me apaixonei e pela qual quero viver todos os momentos de minha vida. Te amo muito amor. A minha sogrinha que amo muito também (Ledir de David) e a todos da sua família. Meus sobrinhos (Giovanni, Andrielli, Lorenzo e Enzo), eu amo muito vocês, o tio sempre estará por perto quando precisarem. Meus cunhados (Heber e Andrea), pessoas que desejo toda felicidade e sucesso junto aos meus irmãos.

Por fim, quero agradecer aos meus colegas da computação, meus outros grandes amigos do CPD (Francisco Henrique Luder Ripoli, Rodrigo Dewes, Marcelo Kunde, Marcus Pereira, Leonardo Shenkel e Rodrigo Moro) e à todos que torceram por mim nesta caminhada.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

ESTUDO E IMPLEMENTAÇÃO DE AUTORIZAÇÃO NO ACESSO PARA O PORTAL DO HUSM UTILIZANDO O CIBAC

Autor: Leandro Sacchet
Orientador: Prof. Dr. Raul Ceretta Nunes
Local e data da defesa: Santa Maria, 01 de março de 2007.

Atualmente, os assuntos confidencialidade e privacidade são preocupações inevitáveis na área da Tecnologia e Informação, em especial na área médica. Para que tais premissas sejam atendidas, deve-se utilizar um mecanismo de controle de acesso robusto, para dessa forma, enfrentar os desafios de segurança que envolve estas áreas.

Afim de facilitar o acesso às informações clínicas, mantendo porém a confidencialidade e privacidade dos dados disponibilizados, foi proposta a implementação de um portal para o Hospital Universitário de Santa Maria (HUSM). Além do portal, foi necessário dar continuidade ao estudo e implementação do modelo de controle de acesso baseado em informações contextuais (CIBAC), com a inclusão do suporte à regras negativas.

Este trabalho apresenta um estudo e implementação de autorização no acesso para o portal do HUSM utilizando o CIBAC. Também é apresentada a implementação do suporte à regras negativas no CIBAC, como forma de facilitar a descrição de regras de controle de acesso.

Palavras-chave: SOA, Web Services, CIBAC, Portal.

ABSTRACT

Undergraduation Final Work
Undergraduation in Computer Science
Federal University of Santa Maria

STUDY AND IMPLEMENTATION ABOUT AUTHORIZATION IN THE HUSM WEB PORTAL ACCESS USING THE CIBAC

Author: Leandro Sacchet
Advisor: Prof. Dr. Raul Ceretta Nunes

Nowadays, the subjects confidenciality and privacy are inevitable concerns in the Technology and Information area, specially in the medical area. So that such premises are taken care of, a robust access control mechanism must be used to face the challenges of security that involves these areas. With the goal to facilitate the access to clinical information keeping the confidencialidade and privacy of the data, we has implemented a web portal to the Santa Maria Hospital University (HUSM). Beyond the portal, we also has worked on the implementation of negative rules support on the Contextual Information-Based Access Control model (CIBAC). This work presents details of this study and implementation.

Keywords: SOA, Web Services, CIBAC, Web Portal.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ASF	*Apache Software Fundation*
CIBAC	*Contextual Information-Based Access Control*
CSAC	*Context Sensitive Access Control*
CS-RBAC	*Context-Sensitive Role-Based Access Control*
CSS	*Cascading Style Sheet*
C-TMAC	*Contextual Team-Based Access Control*
DAC	*Discretionary Access Control*
DNS	*Domain Name System*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HUSM	Hospital Universitário de Santa Maria
IDL	*Interface Definition Language*
IP	*Internet Protocol*
JEE	*Java Enterprise Edition*
JSP	*JavaServer Pages*
JSTL	*JavaServer Pages Standard Tag Library*
MAC	*Mandatory Access Control*
MVC	*Model View Controller*
OMG	*Object Management Group*
PDP	Ponto de Decisão de Políticas
RBAC	*Role Based Access Control*
RPC	*Remote Procedure Call*
SGML	*Standard Generalized Markup Language*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*

TIC	Tecnologia de Informação e Comunicação
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WSDL	*Web Services Description Language*
XACML	*eXtensible Access Control Markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da Internet e com as facilidades de comunicação e acesso a informações que a mesma proporciona, surgiu a necessidade de criar um portal para o Hospital Universitário de Santa Maria (HUSM). Este sistema deve ser usado para disponibilizar de forma facilitada o acesso a dados clínicos dos pacientes, mantendo porém a confidencialidade e integridade das informações.

Atualmente fala-se muito em Segurança da Informação e como ela deve ser modelada. Segurança da Informação refere-se à proteção existente sobre as informações de uma determinada empresa ou pessoa, isto é, aplica-se tanto as informações corporativas quanto as pessoais.

Para enfrentar os desafios de segurança que envolvem este tipo de sistema, são necessários fortes serviços de autenticação e autorização. Este trabalho explora mecanismos de segurança com enfoque na autorização do usuário.

Para que um mecanismo de autorização funcione conforme o esperado, deve-se usar um bom modelo de controle de acesso. Este modelo é um ponto chave para manter a integridade e confidencialidade dos dados, assim como a privacidade dos pacientes. Para a implementação da arquitetura do portal (seção 4.3) é utilizado, como serviço de autorização, o modelo de controle de acesso baseado em informações contextuais (CIBAC - seção 3.5.1) que foi implementado na forma de uma Arquitetura Orientada a Serviços utilizando *Web Services* (SOUZA, 2006).

Web Services (seção 2.2) é uma tecnologia relativamente nova e tem recebido grande aceitação como uma importante implementação de uma Arquitetura Orientada a Serviços (seção 2.1). Isto porque *Web Services* permite a integração de aplicações extremamente heterogêneas através da Internet. A especificação do *Web Service* é completamente independente da linguagem de programação, sistema operacional, e hardware utilizado, isto

para obter uma baixo acoplamento entre os serviços (ERL, 2005).

Enfim, este projeto apresenta a implementação do portal para o HUSM e, também dá continuidade ao estudo e implementação do CIBAC, explicando o que deve ser incluído e alterado para que o mesmo passe a suportar regras negativas, mostrando as novas representações e os algoritmos utilizados.

1.1 Justificativa

O estudo e implementação envolvida no portal do HUSM é uma necessidade, pois não há atualmente nenhuma facilidade de acesso às informações clínicas. Um portal *web* com informações sobre os pacientes pode melhorar muito principalmente o atendimento ao paciente.

Como se trata de um sistema *web* e, levando em consideração que algumas informações e dados disponibilizados são restritos a pessoas devidamente autorizadas, deve haver um controle de acesso robusto a estas informações.

Todo o estudo dos mecanismos de autorização realizados neste projeto tem por objetivo resolver tais problemas, impedindo usuários não autorizados de acessar informações e funcionalidades que não lhe dizem respeito.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo fazer um estudo mais profundo sobre segurança da informação no contexto do CIBAC. Deve ser explorado o uso de mecanismos de autorização baseados em contexto para controlar o acesso *web* à dados clínicos, o que é uma necessidade para o portal do HUSM.

1.2.2 Objetivos Específicos

Para atender ao objetivo geral, têm-se os seguintes objetivos específicos:

- estudar os conceitos envolvidos em uma Arquitetura Orientada a Serviços;
- estudar os conceitos envolvidos na aplicação de Web Services;
- estudar os modelos de controle de acesso existentes atualmente;
- entender o funcionamento do CIBAC;

- implementar um portal para acesso às informações;
- utilizar e adequar o CIBAC ao portal, para desta forma controlar o acesso aos dados e funcionalidades;

O restante deste texto está organizado em 5 capítulos:

- **Capítulo 2 - Revisão da Literatura:** apresenta as principais características dos conceitos e tecnologias estudadas para realização deste trabalho.
- **Capítulo 3 - Modelos de Controle de Acesso:** apresenta conceitos e características de alguns modelos de controle de acesso. Também comenta sobre a linguagem utilizada atualmente pelo CIBAC na descrição de políticas de acesso.
- **Capítulo 4 - Portal do HUSM:** apresenta detalhes do modelo do portal para o HUSM. Também explica a implementação do mesmo assim como as tecnologias e políticas utilizadas. Além disso, há uma breve descrição de como foi feito o registro de auditoria no processo de autorização entre o portal e o CIBAC e, também é apresentado a aplicação alvo do SIE a ser migrada para o ambiente *web*.
- **Capítulo 5 - Regras Negativas no CIBAC:** detalha como o CIBAC está atualmente, ou seja, explica como são representadas as regras positivas e também descreve o algoritmo utilizado na avaliação de tais regras. É explicado o que deve ser incluído e alterado para que o mesmo passe a suportar regras negativas, mostrando as novas representações e os algoritmos utilizados.
- **Capítulo 6 - Conclusão:** apresenta as considerações finais referentes ao trabalho.

2 REVISÃO DA LITERATURA

Este capítulo contextualiza a pesquisa bibliográfica e expõe as principais características de uma Arquitetura Orientada a Serviços. Também é feita uma explicação sobre os padrões relacionados com a tecnologia *Web Services*, necessários para uma melhor compreensão do funcionamento do modelo CIBAC (detalhado na seção 3.5.1).

2.1 Arquitetura Orientada a Serviços

Arquitetura de software é um conceito bastante abstrato, o que dá margem a uma série de definições. Uma arquitetura de software trata basicamente de como os componentes fundamentais de um sistema se relacionam intrinsecamente e extrinsecamente (IEEE, 2000).

Uma Arquitetura Orientada a Serviços (SOA - *Service Oriented Architecture*) tem como seu componente fundamental o conceito de serviços. Serviço pode ser definido como uma função de um sistema computacional que é disponibilizado para outro sistema na forma de um serviço, sendo que este deve funcionar de forma independente do estado de outros e deve possuir interface bem definida.

A SOA (PAPAZOGLOU, 2003) trata-se de um outro tipo de abordagem no desenvolvimento de *software* no qual as aplicações passam a ser construídas e reorganizadas como provedoras de serviços (ou operações) específicos e bem definidos.

A figura 2.1 mostra as colaborações em uma Arquitetura Orientada a Serviços. As colaborações seguem o paradigma “busque, ligue e invoque” (*find, bind and invoke*), onde um “serviço consumidor” localiza dinamicamente um “serviço de registro” em busca de algum serviço que satisfaça determinados critérios. Se o serviço procurado existe, então o “serviço de registro” fornece ao consumidor a descrição daquele serviço, o qual contém informações sobre sua interface e localização, Erl (2004).

Figura 2.1: Colaborações entre os serviços numa SOA.

Os papéis em uma Arquitetura Orientada a Serviços são:

- **Serviço Consumidor:** o serviço consumidor pode ser uma aplicação, um módulo de software ou outro serviço. Ele inicia pela busca em um serviço de registro, logo após a localização, ele se liga ao serviço fornecedor encontrado através de um protocolo de transporte e executa operações desejadas. O serviço consumidor executa operações de acordo com a interface estabelecida pela descrição do serviço fornecedor.

- **Serviço Fornecedor:** o serviço fornecedor é uma entidade que possui um endereço acessível pela rede, aceitando e executando requisições feitas por consumidores. Publica seu serviço e o contrato de sua interface para o serviço de registro, permitindo que consumidores possam facilmente descobri-lo e acessá-lo.

- **Serviço de Registro:** o serviço de registro é quem possibilita a descoberta de serviços. Contém um repositório dos serviços disponíveis e permite a busca destes, feita por serviços consumidores.

2.1.1 Características de uma SOA

As características básicas de uma Arquitetura Orientada a Serviços são:

- **Reusabilidade:** a parte lógica das aplicações são divididas em serviços com a intenção de promover o reuso;

- **Baixo Acoplamento:** os serviços mantêm uma relação que minimiza dependências. Esta característica é alcançada através do contrato de serviço estabelecido,

permitindo que eles interajam através da troca de mensagens pré-definidas.

- **Autonomia:** os serviços tem total controle sobre a lógica que eles encapsulam, ou seja, esta lógica tem limites bem definidos. Dessa forma, consegue-se autonomia e também evita a dependência de outros serviços o que prejudicaria sua evolução.

- **Componentização:** coleções de serviços podem ser coordenadas e montadas formando o chamado *Serviço Composto*. Dessa forma, um processo de negócio pode ser dividido em uma série de serviços, cada um responsável pela execução de parte do processo, (KRAFZIG, 2004).

- **Serviços não possuem estado:** serviços minimizam a retenção de informações específicas a determinada atividade, já que informação de estado é um dado específico a uma atividade corrente. Serviços sem estados são muito importantes também para promover a reusabilidade e escalabilidade dentro de uma SOA.

- **Serviços são fáceis de descobrir:** serviços são descritos externamente de modo que possam ser encontrados e avaliados através dos mecanismos disponíveis. Para prevenir a criação acidental de um serviço redundante, a existência de um registro central de serviços é encorajada.

2.2 Web Services

De acordo com o Grupo de Trabalho do *World Wide Web Consortium* (W3C) responsável pela arquitetura dos *Web Services*, surgiu a seguinte definição sobre esta tecnologia:

> Um *Web Service* é uma aplicação de *software* identificada através de uma Identificação Uniforme de Recursos (URI - *Uniform Resource Identifier*) e contém interfaces e ligações capazes de serem definidas, descritas e descobertas através do uso de uma Linguagem de Marcação eXtendida (XML - *eXtensible Markup Language*). Um *Web Service* suporta interações diretas com outros agentes de softwares através da troca de mensagens baseada em XML via protocolos de comunicações da Internet, (WEB, 2004).

Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam inte-

ragir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

Um dos motivos que tornam *Web Services* atrativos é o fato deste modelo ser baseado em tecnologias padrões, em particular XML e HTTP (*HyperText Transfer Protocol*). *Web Services* são usados para disponibilizar serviços interativos na *web*, podendo ser acessados por outras aplicações. O protocolo SOAP, melhor detalhado na seção 2.2.3, está se tornando padrão para a troca de mensagens entre aplicações e *Web Services*, já que é uma tecnologia construída com base em XML e HTTP.

2.2.1 Características dos Web Services

A seguir mostra-se algumas características dos *Web Services*:

- Um *Web Service* é um recurso *web*. É acessado usando protocolos independentes de plataforma e de linguagem como HTTP por exemplo. Estes protocolos propiciam fácil integração de ambientes heterogêneos.

- Um *Web Service* provê uma interface, chamada de API Web (*Application Programming Interface*), que pode ser acessada por outro programa. Esta interface aplicação-aplicação pode ser invocada de qualquer tipo de aplicação. A API Web habilita o acesso a lógica da aplicação que implementa o serviço.

- Um *Web Service* é registrado e pode ser localizado através de um registro de *Web Services*. O registro permite aos consumidores encontrar os serviços que satisfazem suas necessidades. Os consumidores dos serviços normalmente são outras aplicações mas podem também ser humanos.

- Os sistemas conectados aos *Web Services* devem ser fracamente acoplados. Os *Web Services* se comunicam passando mensagens XML entre si pela API Web. Isto adiciona uma camada de abstração ao ambiente fazendo que as conexões sejam flexíveis e fáceis de se adaptar.

2.2.2 A linguagem WSDL

A vasta aceitação do *Web Service* resultou na criação de novas tecnologias que se tornaram padrões de fato, dentre elas uma das mais importantes é a Linguagem de Descrição de Web Service (WSDL - *Web Services Description Language*). Esta linguagem é usada

para descrever e também para localizar *Web Services*, representando um contrato entre o cliente que requisita os serviços e o provedor dos serviços.

WSDL não é uma linguagem simples, pois a intenção dos seus criadores foi criar para os *Web Services* uma Linguagem de Definição de Interface (IDL - *Interface Definition Language*) não amarrada a nenhum protocolo, nenhuma linguagem de programação e nenhum sistema operacional. Também porque havia uma grande preocupação com sua modularidade de modo a permitir reusar seus artefatos.

2.2.3 O protocolo SOAP

Devido à comunicação entre os serviços ser baseada na troca de mensagens, a especificação do protocolo SOAP (*Simple Object Access Protocol*) tem como principal objetivo definir um formato de mensagem padronizado, o qual consiste em um documento XML capaz de armazenar tanto dados de um documento como os de uma Chamada de Procedimento Remoto (RPC - *Remote Procedure Call*). A estrutura deste formato é muito simples, mas a habilidade de estendê-lo e personalizá-lo tem possibilitado a criação de novas especificações, relacionadas a *Web Services* (ERL, 2004).

Uma importante característica do SOAP é que ele usa o protocolo HTTP para o transporte do seu conteúdo, permitindo que documentos XML passem através de *firewalls* sem problemas, pois, por motivos de segurança, os administradores de sistemas geralmente bloqueiam todas as portas de comunicação dos seus servidores, exceto a porta 80, usada pelo HTTP.

Toda mensagem SOAP é empacotada dentro de um *container* conhecido como envelope, responsável por conter todas as partes da mensagem. Cada mensagem pode conter um cabeçalho, área dedicada a armazenar meta informações. Embora seja opcional, o cabeçalho é usado na implementação de várias extensões. O real conteúdo é armazenado no corpo da mensagem, o qual consiste tipicamente de dados formatados em XML (ERL, 2005). A listagem 2.1 mostra um exemplo de requisição SOAP, realizada por um serviço implementado no CIBAC.

O protocolo SOAP também define meios para enviar arquivos anexos, facilitando a entrega de dados que não são facilmente representadas através de um documento XML. Anexos SOAP são comumente usados para transportar arquivos binários, como imagens.

Finalmente, as mensagens SOAP oferecem a habilidade de adicionar uma lógica para

tratamento de exceções, através do uso de uma seção opcional chamada *fault*, que reside dentro do corpo da mensagem. O uso típico dessa seção é para armazenar uma simples mensagem para informar o serviço requisitante que alguma exceção ocorreu.

Listagem 2.1: Requisição SOAP no estilo RPC

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
    envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/
        encoding/"/>
      <n:buscaAcoesDelegadas>
        <arg0 xsi:type="xsd:string">Leandro</arg0>
        <arg1 xsi:type="xsd:string">Listar Prontuarios</arg0>
        <arg2 xsi:type="xsd:string">Servicos</arg0>
      </n:buscaAcoesDelegadas>
    </soap:Body>
</soapenv:Envelope>
```

3 MODELOS DE CONTROLE DE ACESSO

O controle de acesso a determinadas informações, de forma alguma, deve prejudicar o usuário do sistema por negar o acesso legítimo às informações e aos serviços requisitados. No entanto, algumas informações devem permanecer confidenciais, exceto quando em atendimento à vontade do usuário ou a determinações legais.

Este capítulo faz algumas considerações sobre modelos de controle de acesso que possam vir a atender esta premissa. O controle de acesso nos sistemas computacionais normalmente segue as seguintes diretrizes (RAVI SANDHU, 1994):

- **Modelo Discricionário (DAC)**;
- **Modelo Obrigatório (MAC)**;
- **Modelo Baseado em Papéis (RBAC)**;

Além destes modelos de controle de acesso, também são feitas neste capítulo algumas considerações em relação aos modelos de controle de acesso relacionados especificamente à utilização de contextos, incluindo o CIBAC. Também é comentado sobre a descrição de políticas de acesso, mais precisamente sobre a linguagem XACML (seção 3.6).

3.1 Modelo Discricionário

As políticas de controle do modelo discricionário (DAC - *Discretionary Access Control*) são baseadas na identidade dos usuários. As autorizações especificam para cada usuário (ou grupo de usuários) e cada objeto do sistema os modos de acesso permitidos (RAVI SANDHU, 1994). Quando um sujeito tenta efetuar uma operação num objeto, o monitor de referência verifica na lista de controle de acesso do objeto se existe alguma autorização que concede o privilégio. Existindo a autorização, o acesso é concedido, caso contrário, ele é negado.

O que faz o DAC ser amplamente utilizado é sua flexibilidade, o proprietário de um objeto é livre para conceder ou revogar privilégios de acesso a outros usuários (JOSHI, 2001). Entretanto, possibilita que um usuário autorizado a acessar um objeto faça uma cópia do mesmo e o distribua a outros usuários sem a permissão explícita do proprietário (RAVI SANDHU, 1994).

Por ser um modelo assimétrico e descentralizado, o DAC dificulta a administração da política de acesso. Por exemplo, para saber todos os privilégios de um usuário é necessário examinar as listas de controle de acesso de cada objeto. Outro problema ocorre quando precisamos remover privilégios. Além de ser necessário percorrer todas as listas, o proprietário do objeto deve dar o privilégio de alteração da lista para quem administra a política de controle de acesso.

3.2 Modelo Obrigatório

As políticas de controle do modelo obrigatório (MAC - *Mandatory Access Control*) são baseadas na classificação dos *subjects* e *objects* do sistema. Para cada objeto e usuário do sistema é associado um nível de segurança.

O nível de segurança associado ao objeto reflete o nível de importância da informação contida no mesmo, classificando o objeto quanto ao dano potencial que um acesso não autorizado traria. O nível de segurança associado ao usuário, também chamado de *clearance*, reflete o nível de confiabilidade do usuário em não expor a informação a um usuário que não esteja autorizado para tal (RAVI SANDHU, 1994). De maneira geral, o acesso de um usuário a um objeto é verificado em função do relacionamento entre os níveis de segurança deles e levando-se em conta o tipo de acesso solicitado.

3.3 Modelo baseado em papéis

O modelo de controle de acesso baseado em papéis (RBAC - *Role Based Access Control*) é caracterizado pelas atividades que os usuários desempenham no sistema (perfis). Um perfil pode ser visto como um conjunto de ações e responsabilidades associadas a uma atividade de trabalho (RAVI SANDHU, 1994). Existem quatro modelos relacionados com o RBAC, mas o modelo base pode ser observado na figura 3.1.

Esse modelo possui quatro entidades principais: U (Usuários), P (Papéis), A (Autorizações) e S (Sessões). O relacionamento UP caracteriza uma relação muitos para muitos

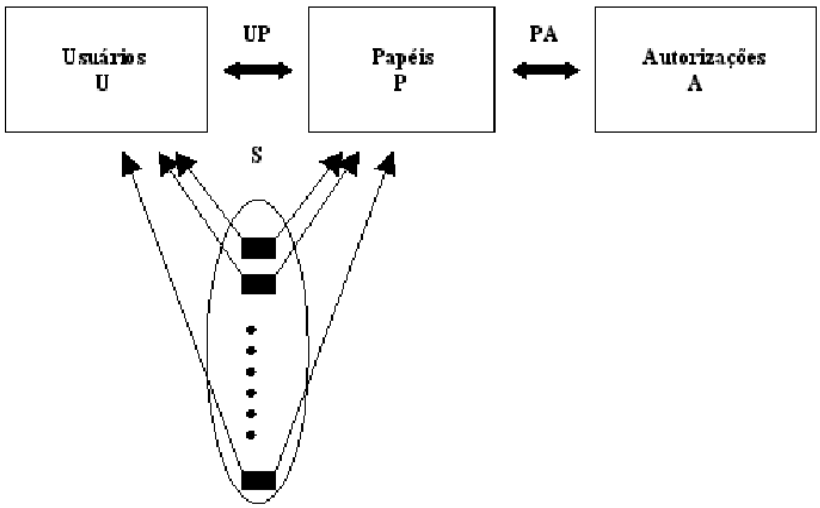

Figura 3.1: Estrutura Esquemática do Modelo RBAC.

entre usuários e papéis, o que mostra que um usuário pode possuir simultaneamente mais de um papel. O relacionamento PA, também muitos para muitos, é que configura as autorizações que um determinado papel deve possuir. As sessões se relacionam com apenas um usuário por vez, mas permite a ativação de qualquer papel que estiver na relação UP com aquele usuário (FERRAIOLO, 2001).

A grande vantagem deste modelo é que em vez da atribuição de direitos de acesso de forma individualizada a cada usuário, esta atribuição é feita aos perfis. Em uma segunda etapa, os usuários são associados a perfis em função das atividades desempenhadas no sistema.

Um importante conceito também empregado pelo RBAC é a hierarquia de perfis. O modelo define uma relação de ordem entre os papéis, de forma a explicitar melhor as linhas de autoridade e responsabilidade de uma organização. Em um cenário típico de uma organização, indivíduos que desempenham diferentes funções podem executar tarefas em comum. Podemos ver que a hierarquia entre perfis simplifica bastante a descrição de direitos de usuários, pois evita a necessidade de especificar de forma repetida operações em comum. Ainda, este relacionamento hierárquico pode ser destinado à estrutura hierárquica das próprias organizações, permitindo que a política de segurança seja descrita de maneira extremamente natural. Em uma hierarquia de perfis, um perfil pode "conter"outros perfis. No RBAC, um perfil hierarquicamente superior pode conter um ou mais perfis hierarquicamente inferiores. Isto significa que um perfil superior tem todos os privilégios e pode executar todas as operações que os perfis de níveis inferiores contêm.

3.4 Controle de Acesso Sensível ao Contexto

Contexto é um fator muito importante no controle de acesso. Ele denota informações ambientais existentes no momento de uma solicitação de acesso. Alguns exemplos típicos de variáveis contextuais são:

- **Informações sobre o usuário corrente:** nome de login, matrícula, unidade de lotação, papéis associados, etc;
- **Informações sobre o momento do acesso:** data, hora, dia da semana, etc;
- **Informações sobre o local da solicitação de acesso:** endereço IP (*Internet Protocol*), DNS (*Domain Name System*), porta do cliente, etc;
- **Informações de segurança:** domínios DNS confiáveis, indicação se a conexão é segura, etc;

Contextos ainda podem indicar informações mais específicas, relacionadas ao recurso que se pretende acessar via aplicação utilizada pelo usuário. Por exemplo, um contexto poderá representar o conjunto dos pacientes internados num hospital, ou indicar se um paciente está sendo acompanhado por um determinado médico ou equipe médica.

Um modelo CSAC (*Context Sensitive Access Control* - Controle de Acesso Sensível ao Contexto) deve possuir uma definição formal de contexto. Ou seja, deve dizer quantos fatores no contexto devem ser considerados, e como eles devem ser considerados. Também deve possuir uma estrutura extensível para que terceiros possam considerar outros fatores no contexto.

Uma autorização contextual poderá ser positiva, concedendo o acesso, ou negativa, proibindo o acesso, com base na avaliação de uma expressão lógica, denominada *Regra de Autorização*. Essa expressão é definida através do contexto que, quando avaliada, resulta em informações sobre o usuário corrente.

3.5 Modelos Contextuais

Nesta seção são detalhados alguns modelos de controle de acesso baseados em informações contextuais. Para um melhor entendimento destes modelos, será abordado primeiramente o modelo de controle de acesso utilizado neste estudo, o CIBAC.

3.5.1 Modelo CIBAC

Nesta subseção inicialmente é apresentado a terminologia a cerca do termo contexto, para então apresentar a especificação formal dos termos utilizados no modelo CIBAC. Salienta-se que embora a exemplificação utilizada nesta subseção seja direcionada para a área da saúde, as proposições podem ser utilizadas também em outros domínios.

3.5.1.1 Conceitos Básicos

O modelo CIBAC foi proposto por (SOARES, 2006) e tem como principal característica o uso de informações contextuais para prover o controle de acesso a dados médicos.

De acordo com (DEY, 1999), contexto é qualquer informação relevante que possa ser utilizada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar, ou um objeto, relevantes para a interação entre o usuário e a aplicação. No CIBAC, estas informações são modeladas de acordo com a política adotada e, por este motivo, são definidas como propriedades, pois fornecem os dados necessários a caracterização de um contexto. Cada propriedade distinta pode, por sua vez, ser agrupada diretamente nos contextos a que se referem, isto é, um contexto pode ser definido a partir de um conjunto de informações sobre determinadas entidades, como um usuário ou mesmo um objeto, (SOARES, 2006).

O agrupamento de informações de contexto numa classe de informações sobre uma dada entidade, por exemplo, um usuário, caracteriza-se como um tipo de contexto. Na área da saúde o conceito de tipo de contexto auxilia a especificação de regras de acesso cientes de contexto e afinadas com a política de acesso utilizada. Por exemplo, médicos plantonistas na emergência devem ser autorizados a acessar dados sobre pacientes em atendimento na emergência, mas não sobre pacientes internados em outras unidades do hospital, a menos que ele seja também o médico de um determinado paciente.

Como toda regra de acesso baseia-se em informações e, como no CIBAC as informações são modeladas como propriedades, um exemplo de contexto para um médico plantonista na emergência pode ser modelado como:

- Tipo de Contexto: Usuário;
- Elemento: Médico;
- Propriedade: Função, com valor de propriedade = plantonista;

- Propriedade: Local, com valor de propriedade = emergência.

Note que a existência de algumas propriedades não se restringe a um único elemento, uma vez que as mesmas podem estar relacionadas a mais de um tipo de contexto.

Outro ponto interessante abordado pelo CIBAC é que, em um ambiente hospitalar pode ser comum a delegação de atribuições a médicos ou especialistas de outras áreas da saúde, a fim de prover assistência a um determinado paciente. Quando se trata a questão da delegação como sendo uma informação contextual e gera-se regras para que esta delegação seja coerente, fica facilitada a modelagem de dados temporais, locais, e relacionados a uma dada assistência, o que facilita a adequação do controle de acesso à legislação pertinente.

3.5.1.2 Definições

Nesta subseção apresentam-se as definições formais que norteiam a utilização do termo contexto. De acordo com (SOARES, 2006) são definidas:

- **Propriedade de Contexto:** Uma Propriedade de Contexto é um par (P, V), onde P é o nome da propriedade e V é o valor da propriedade P. Observa-se que como uma propriedade de contexto é aplicável a um dado elemento (usuário, dispositivo ou recurso), pode-se dizer que todo par (P, V) pertence a D, onde D é um domínio.

- **Contexto:** Um contexto CTX é um conjunto de propriedades de contexto (P, V). Um contexto CTX pode também ser formado por outros diferentes contextos. Por exemplo, CTX1 = (P11,V11), (P12,V12) e CTX2 = (P21,V21), (P22,V22), mas CTX3 = (CTX1,CTX2).

- **Condição de Contexto:** Uma Condição de Contexto ContextCond é uma fórmula booleana em forma de tupla, tal como (CT, P, OP, V), onde CT é o tipo de contexto, P é o nome da propriedade, OP é um operador (por exemplo, >, <, =) e V é o valor que a propriedade assume.

- **Tipo de Contexto:** Um Tipo de Contexto é uma classificação dada de acordo com as características das informações de cada contexto, ou seja, possibilita agrupar propriedades contextuais de acordo com a natureza de cada informação (por exemplo, sujeito ou objeto).

- **Autorização:** Uma Autorização é uma tupla (CE, O, AM, CC), onde CE representa as expressões credenciais do usuário, O é o objeto, ou objetos, a ser acessado, AM é o modo de acesso e CC é o resultado da avaliação sobre uma condição de contexto.

De acordo com (SOARES, 2006), a contribuição do CIBAC caracteriza-se pela inserção da condição de contexto (CC) como regra de autorização e pela manipulação das informações de contexto (CI) na forma de propriedades de contexto. Os demais termos são originários da definição de autorização do modelo de referência RBAC, comentado anteriormente.

3.5.1.3 Condição de Contexto

De acordo com os modelos de controle de acesso existentes, para descrever uma política eficiente, costuma-se utilizar regras que consideram objetos e modos de acesso. Entretanto, no CIBAC as regras consideram também expressões credenciais dos usuários, resultando em uma avaliação sobre uma determinada condição de contexto. Por exemplo, suponha que um determinado arquivo de paciente só pode ser acessado, ou tenha seu acesso liberado, se uma das duas condições seguintes for satisfeita: 1) um usuário com função de enfermeira acessa os dados a partir das 10:00h; ou 2) o objeto em questão deve estar no setor de emergência com um contador de acessos com valor inferior a 20 ocorrências. Esta regra ou condição de contexto pode ser codificada em XML conforme a listagem 3.1.

Listagem 3.1: Exemplo Elemento XML - ContextCond

```
<ContextCond>
   <Clause>
      <Context Type="Sujeito">
         <Property Name="Tempo"/>
         <Operator OP=">"/>
         <Value V="10:00"/>
      </Context>
      <Context Type="Sujeito">
         <Property Name="Função"/>
         <Operator OP="="/>
         <Value V="Enfermeira"/>
      </Context>
   </Clause>
   <Clause>
      <Context Type="Objeto">
         <Property Name="Contador"/>
         <Operator OP="<"/>
         <Value V="20"/>
      </Context>
      <Context Type="Objeto">
```

```
21          <Property Name="Local"/>
22          <Operator OP="="/>
23          <Value V="Emergência"/>
24       </Context>
25    </Clause>
26 </ContextCond>
```

Podemos observar que um elemento *ContextCond* é dividido em cláusulas (*Clause*), que contém as possíveis regras de liberação de acesso baseadas em informações contextuais. Estas cláusulas contêm os tipos de contextos utilizados, neste caso *Sujeito* e *Objeto* com suas respectivas propriedades e valores, e um operador que determina a condição efetiva da regra.

Consideram-se as cláusulas da Condição de Contexto como uma operação "OR", determinando que: uma ou outra condição é suficiente para liberar o acesso de determinado sujeito à determinado recurso; e um conjunto de expressões em cada cláusula é considerado como uma operação "AND", ou seja, a cláusula só é verdadeira se todas as expressões em seu interior forem verdadeiras.

3.5.1.4 Tipo de Contexto

De acordo com (SOARES, 2006), no CIBAC uma informação contextual é definida por um conjunto de propriedades pertencentes a um dado tipo de contexto, ou seja, considera-se o tipo de contexto, suas propriedades e seus respectivos valores. Cada tipo de contexto possui seus respectivos *target*. É o identificador de um objeto específico ou de um conjunto de objetos, e é utilizado na codificação XML para se efetuar uma separação entre os elementos inseridos em cada tipo de contexto.

O código da listagem 3.2 exemplifica a representação XML de um tipo de contexto *Objeto*. O elemento específico é o arquivo *OrdemMédica.doc* e suas informações são descritas pelas propriedades de contexto: *Tempo* e *Contador*.

Listagem 3.2: Exemplo Elemento XML - Context Type

```
1  <Context Type="Objeto">
2     <Objeto target="OrdemMedica.doc">
3        <Property name="Tempo">
4          10:00
5        </Property>
6        <Property name="Contador">
7          12
8        </Property>
9     </Objeto>
10 </Context>
```

Como o Contexto é do tipo *Objeto*, as informações referentes ao arquivo OrdemMédica.doc são agrupadas de acordo com seu *target* e uma vez que uma, ou várias propriedades sejam consultadas, ou requeridas por uma regra em uma Condição de Contexto, os valores das mesmas podem ser facilmente localizados.

3.5.1.5 Delegação de Atribuições

Através de uma delegação, via condições de contexto, um usuário pode estabelecer uma delegação a outro usuário apenas para uma ação específica sobre algum recurso. Esta nova atribuição é então verificada em uma tabela de delegações, onde há a descrição do acesso e também características relativas à revogação das delegações.

Supondo que um usuário X deseja delegar uma dada ação sobre o arquivo OrdemMédica.doc ao usuário Y, é então de responsabilidade do CIBAC determinar se esta delegação é viável. Isto é feito através da consulta às condições de contexto para a ação de delegar. Uma vez que o usuário X possua todas as atribuições necessárias para que o acesso à ação de delegar seja concedido ele pode efetuar a delegação, e o usuário Y passa a fazer parte da tabela de delegações com seus respectivos direitos bem como a validade dos mesmos sobre o referido arquivo. Caso a ação seja negada, o usuário Y não é inserido na tabela de delegações e a delegação então não existe.

3.5.2 Modelo C-TMAC

O modelo C-TMAC (*Contextual Team-Based Access Control*) foi desenvolvido com o objetivo de controlar o acesso em ambientes coorporativos. Segundo (MOTTA, 2003) o C-TMAC estende o RBAC básico com a introdução de equipes, contextos e relacionamentos usuário-equipe e equipe-contexto. Usuários são membros de equipes que têm contextos associados, ou seja, o conceito de equipe é usado para relacionar usuários com contextos. Um contexto contém uma regra lógica relacionando os objetos usados na execução de uma tarefa com variáveis de ambiente, como horário de acesso e localização do usuário.

Durante uma sessão, um usuário ativa papéis e equipes e as autorizações disponíveis para ele correspondem a combinação das autorizações associadas a todos os papéis correntes ativados pelos membros das equipes ativadas, filtradas pelas regras contextuais vinculadas as equipes. Por exemplo, suponha-se os papéis Enfermeiro e Médico e a equipe Emergência com uma regra contextual estabelecendo que os acessos somente são

permitidos das salas de emergência. Quando um enfermeiro inicia uma sessão, ele ativa o papel Enfermeiro e a equipe Emergência e tem as autorizações disponíveis para o seu papel, desde que efetue o acesso de alguma sala da emergência. Quando um médico inicia uma sessão e ativa o papel Médico e a equipe Emergência, então, tanto ele, quanto o enfermeiro, vão compartilhar as informações de ambos os papéis porque são membros da mesma equipe. Quando o médico encerra a sessão, desativando seu papel, o enfermeiro deixa de ter as autorizações do papel Médico. Ou seja, os membros de uma equipe compartilham as autorizações enquanto realizam uma tarefa em comum.

O modelo foi especificado sem suporte a hierarquia de perfis. As regras contextuais são suportadas pelo C-TMAC associadas a equipes e utilizadas como critério de filtragem de autorizações possíveis para os perfis ativos de membros de cada equipe.

3.5.3 Modelo CS-RBAC

O CS-RBAC (*Context-Sensitive Role-Based Access Control*) foi desenvolvido por (KUMAR, 2002) e estende o RBAC básico através da introdução dos conceitos de contextos de perfis e de filtros contextuais. O contexto de um papel é formado pela combinação do contexto do usuário que iniciou uma sessão com o contexto do objeto associado à autorização de acesso solicitada. Contextos são formados por conjuntos de pares de nomes e valores e denotam informações relevantes para segurança, sendo dependentes de aplicações específicas. Filtros contextuais são definidos em papéis e estabelecem uma regra lógica relacionando os contextos de usuário e objeto. Autorizações de acesso a um objeto só são permitidas quando a avaliação da regra do filtro contextual é verdadeira. Assim, a concessão ou a proibição de uma autorização dependerá do contexto do papel existente durante a solicitação da autorização de acesso. Podemos tomar como exemplo uma autorização (*cliente, alterar, pedido*) estabelecendo que um usuário com papel cliente pode alterar um objeto do tipo pedido. Sem a definição de um filtro contextual, não há, no modelo, como impedir que um usuário altere pedidos de uma empresa diferente daquele em que ele trabalha. Neste caso, a inclusão do filtro contextual *ObjectContext.empresaID = UserContext.empresaID* no papel cliente, recupera a limitação citada anteriormente, ou seja, impede que um usuário altere pedidos de uma empresa diferente da qual ele trabalha.

O modelo foi especificado sem hierarquias de perfis. Segundo (MOTTA, 2003) existem apenas dois contextos (de usuário e de objeto) e que estes estão fixos na linguagem

usada para expressar os filtros contextuais.

3.6 Descrição de políticas de acesso

XACML (*eXtensible Access Control Markup Language*) é uma linguagem de marcação, baseada no XML, usada na descrição de políticas de controle de acesso. Esta linguagem define o formato de uma requisição que contém informações sobre o usuário, recurso, ação e ambiente, afim de achar políticas que se aplicam a ela e, então, seja tomada uma decisão, gerando uma resposta XACML (GRIFFIN, 2004). Esta linguagem é muito utilizada porque ela fornece padronização para os algoritmos de decisão.

Ao se criar um arquivo contendo uma política, defini-se, através do elemento *Target* da linguagem XACML, que é uma marcação contendo informações relativas a uma requisição, quais os atributos da requisição que devem ser analisados para verificar se uma política é aplicável. Se uma política é aplicável, então avaliam-se as regras existentes nela.

Como muitas políticas podem ser aplicáveis a uma determinada requisição e uma política pode conter várias regras, especifica-se algoritmos para se gerar uma decisão final. Esta decisão pode assumir os seguintes estados: *permitido*, *negado*, *não aplicável* ou *indeterminado*.

Com a utilização de XACML, cria-se um Ponto de Decisão de Políticas (PDP). O PDP é o responsável por receber a requisição XACML, procurar, através do *Target*, todas as políticas aplicáveis, avaliar as regras e usando um algoritmo de decisão retornar uma resposta XACML. Esta resposta também pode conter informações adicionais, como o tempo de duração da permissão.

4 PORTAL DO HUSM

Este capítulo apresenta inicialmente uma visão geral do portal e também comenta os requisitos necessários para atender as necessidades do mesmo. Logo após é apresentada a arquitetura do portal, mostrando passo-a-passo como o mesmo se integra a aplicações do SIE-Saúde. Depois, é explicado a implementação do portal assim como as tecnologias e políticas utilizadas. Por fim, há uma breve descrição de como foi feito o registro de auditoria no processo de autorização entre o portal e o CIBAC e, também é apresentado a aplicação alvo do SIE a ser migrada para o ambiente *web*.

4.1 Visão Geral

É inegável que disponibizar dados clínicos na Internet facilita a vida de muitas pessoas. Porém, isso exige muita responsabilidade por parte do sistema, devendo manter a confidencialidade e integridade das informações.

O portal do hospital é o ponto comum de entrada para os usuários acessarem os dados hospitalares, desde médicos, pacientes, equipe de funcionários entre outros. A autenticação é flexível e deve avançar conforme a tecnologia, sendo empregados os mecanismos *passwords*, certificado digital, reconhecimento de impressão digital. Também é sabido que diferentes tecnologias de autenticação tem diferentes níveis de confiabilidade, devido à isso, será introduzido o conceito de nível de confiança para cada tecnologia de autenticação utilizada[1] (MüLLER, 2007).

Para a autenticação sob-demanda o usuário se autentica utilizando alguma tecnologia disponível em seu computador. O acesso ao portal prossegue até o momento em que o usuário tentar acessar uma informação cuja regra requer uma autenticação com nível maior daquela provida correntemente pelo usuário. Neste momento é dado ao usuário a

[1]Detalhes relacionados à autenticação não serão abordados por este trabalho, já que foge do escopo do mesmo.

oportunidade de elevar seu nível utilizando uma tecnologia de maior confiabilidade (MüLLER, 2007).

Toda parte de autorização do portal é dirigida a políticas, ou seja, baseada em regras expressas em documentos XACML (detalhadas no capítulo 5). Para esta finalidade, foi utilizado o CIBAC, modelo de controle de acesso que representa políticas de autorização através de documentos XACML.

4.2 Requisitos necessários ao Portal

Para atender as necessidades do portal, alguns requisitos são necessários:

- **Autenticação Individual de Usuários:** todos os profissionais da instituição devem ter um identificador único (MüLLER, 2007).
- **Controle de Acesso:** procedimentos devem garantir que os usuários tenham acesso somente àquelas informações que realmente eles tenham direito a conhecer.
- **Trilhas de Auditoria:** deve-se manter o registro de todos os acessos às informações clínicas, numa forma que permita a recuperação para fins de auditoria.
- **Aplicações SIE-Saúde:** deve-se listar as aplicações disponíveis através do sistema de acordo com o usuário corrente.

4.3 Arquitetura do Portal

Neste seção é apresentada a arquitetura do portal. Maiores detalhes de segurança são explicados na seção 4.4. A figura 4.1 ilustra o protótipo inicial do projeto.

De acordo com a figura 4.1, podemos dividir a parte de autorização do portal nos seguintes passos:

- **Passo 1:** neste momento o usuário já está autenticado. Deve ser exibida uma página com todas aplicações que este usuário tem autorização de acesso. Para acessar estas aplicações será considerado, além do usuário que está logado, o seu nível de confiança no sistema[2].
- **Passos 2 e 3:** antes da página com a listagem da aplicações ser exibida, deve-se descobrir quais as aplicações que o usuário tem autorização de acesso. Para isso,

[2]Níveis de Confiança: Login/Senha (nível 1), Impressão Digital (nível 2) e Certificado Digital (nível 3) (MüLLER, 2007)

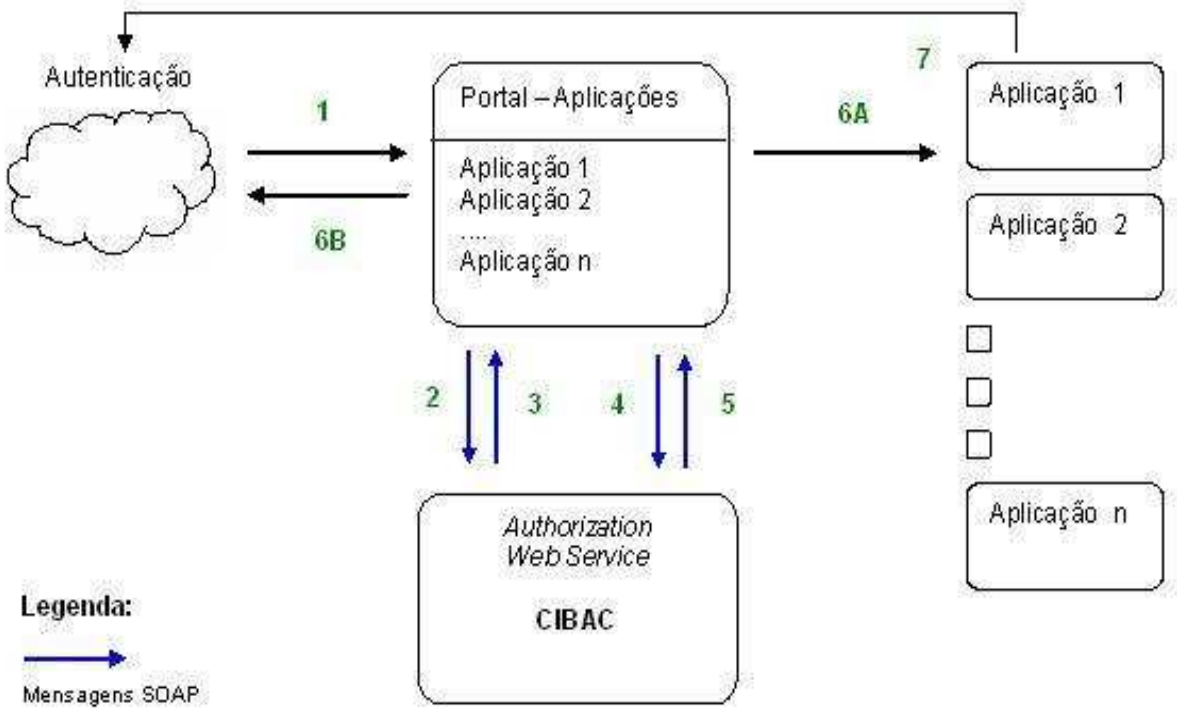

Figura 4.1: Arquitetura do Portal

o portal se comunica com o CIBAC através de mensagens SOAP[3] obtendo assim as aplicações. As aplicações retornadas pelo CIBAC são independentes do nível de autenticação que o usuário possui atualmente no sistema, ou seja, de acordo com a aplicação que este usuário quiser acessar posteriormente, pode ser necessário se autenticar com um nível maior de confiança[4].

- **Passo 4 e 5:** quando o usuário tentar acessar uma das aplicações que foram listadas, o portal se comunica novamente com o CIBAC afim de saber se o usuário tem acesso liberado. Deve-se notar que, agora é considerado o nível de confiança do usuário no sistema. Caso o usuário tenha autorização, o CIBAC recupera todas as ações que o mesmo pode executar nesta aplicação.

- **Passo 6A:** este passo significa que o usuário tem autorização de acesso a aplicação. Pode-se concluir então que o CIBAC além de dar permissão de acesso, retornou também todas as ações que este usuário pode executar na aplicação.

- **Passo 6B:** se o usuário for redirecionado para a página de login (*Autenticação*),

[3]As mensagens SOAP não estão sendo criptografadas nesta comunicação, devido a limitação de tempo na implementação das tarefas propostas neste trabalho .

[4]Autenticação sob-demanda: esta característica não será detalhada neste estudo.

significa que ele não teve acesso a determinada aplicação, ou seja, o CIBAC não deu autorização. Há a possibilidade desta restrição ter ocorrido devido ao usuário ter se autenticado com um nível de confiança abaixo do esperado pela aplicação. Portanto, o usuário deverá elevar seu nível de confiabilidade para poder acessar determinadas aplicações.

- **Passo 7:** quando este passo ocorre, significa que o usuário não pode executar determinada ação (consulta, alteração, remoção, inserção, impressão, etc) na aplicação, ou seja, o CIBAC deu autorização de acesso a aplicação mas não permitiu tal ação na mesma. Esta restrição aconteceu devido ao usuário ter se autenticado com um nível de confiança abaixo do esperado ou porque este usuário, considerando suas informações contextuais correntes, não deve desempenhar tal ação nesta aplicação.

4.3.1 Serviços do CIBAC

Nesta seção é apresentado a interação entre os serviços que compõe o módulo de controle de acesso do CIBAC. Este nível de detalhes é necessário para um melhor entendimento do funcionamento interno do modelo de autorização utilizado pelo portal[5].

4.3.1.1 Arquitetura de Controle de Acesso do CIBAC

A arquitetura do portal comunicando-se com a implementação do CIBAC é ilustrada através da figura 4.2.

O CIBAC foi criado em cima de uma Arquitetura Orientada a Serviços. Esta arquitetura é formada pela implementação de três *Web Services* com responsabilidades bem distintas, são eles: *Serviço de Administração*, *Serviço de Decisão* e *Serviço de Autorização*.

4.3.1.2 Serviço de Administração

No CIBAC, as informações contextuais são modeladas como propriedades e armazenadas num banco de dados relacional. O Serviço de Administração contempla o acesso ao banco de dados e provê diversas operações para a manipulação destes dados (SOARES, 2006). Entre elas podemos citar:

- gravação (inserção ou atualização) de uma delegação, objeto, sujeito;

[5]Deve-se deixar claro que estes serviços não foram implementados neste trabalho. Eles já haviam sido implementados anteriormente (SOUZA, 2006).

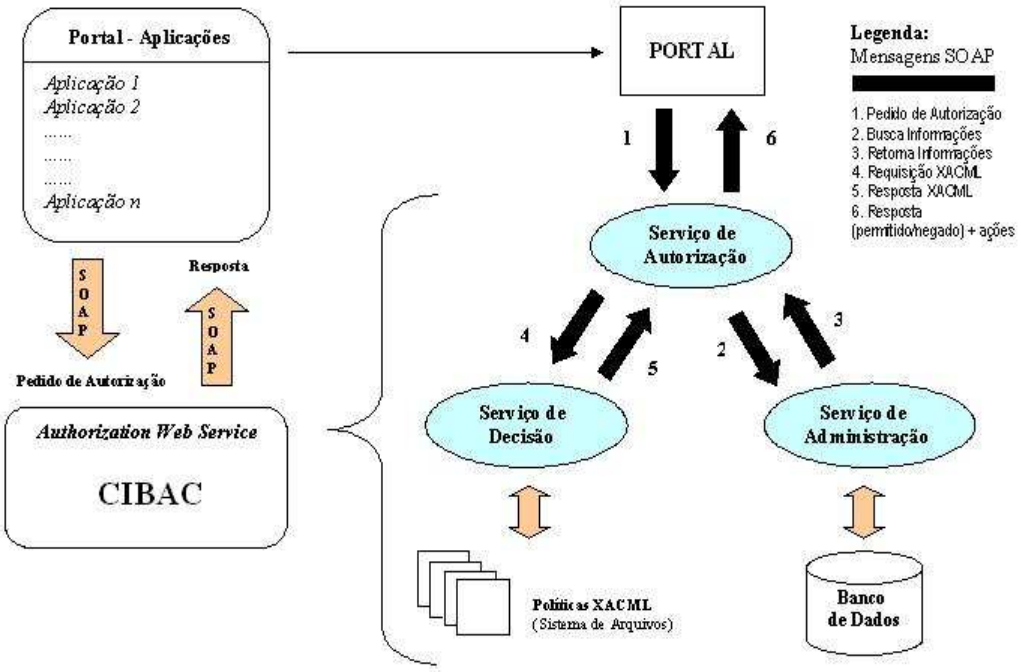

Figura 4.2: Arquitetura do módulo de controle de acesso que compõe o CIBAC.

- busca de todas as delegações, objetos, sujeitos;
- exclusão de uma delegação, objeto, sujeito;

Do ponto de vista da arquitetura do CIBAC, o resultado de uma consulta a este serviço é a recuperação das informações contextuais de uma dada requisição, as quais irão subsidiar o Serviço de Decisão (SOARES, 2006).

4.3.1.3 Serviço de Decisão

O Serviço de Decisão é responsável por gerenciar os arquivos XACML contendo as políticas de acesso, as quais são essenciais ao CIBAC. Estas políticas são armazenadas em um diretório do servidor onde o serviço é executado. A localização de tal diretório é informada ao serviço via parâmetro de inicialização (SOARES, 2006).

Outra responsabilidade deste serviço é receber mensagens SOAP contendo requisições XACML, devidamente instruídas com informações contextuais, e confrontá-las com as diversas políticas previamente cadastradas, seguindo a especificação XACML. Após isso, é gerada uma resposta XACML que é retornada ao requisitante (SOARES, 2006).

Para avaliar se existe alguma política que se aplica a uma dada requisição XACML, este serviço executa um algoritmo de decisão que inicialmente avalia as políticas em XACML para verificar se existe uma ou mais políticas aplicáveis a uma dada requisi-

ção. Se existe pelo menos uma política aplicável, o algoritmo avalia se existe pelo menos uma cláusula verdadeira que libere o acesso. Uma cláusula só é verdadeira se todas as expressões credenciais (regras) em seu interior forem verdadeiras.

4.3.1.4 Serviço de Autorização

O Serviço de Autorização desempenha a função de gerente do CIBAC, ou seja, coordena todo o processo de autorização para autorizar ou não um determinado sujeito a executar uma ação que envolve algum objeto (recurso). Para que este serviço retorne uma resposta apropriada, ele utiliza-se dos dois serviços descritos anteriormente (SOARES, 2006).

A função principal do Serviço de Autorização é buscar as informações contextuais (CI) das entidades envolvidas na requisição original (sujeito e objeto), que foram armazenadas através do Serviço de Administração, e montar uma requisição XACML para que esta possa ser utilizada pelo Serviço de Decisão. Logo após, converter a resposta XACML, recebida através do Serviço de Decisão, em uma resposta utilizável pelo sistema usuário (SOARES, 2006).

O portal, para tomar uma decisão, ao invés de consular uma base de dados, ele comunica-se com o CIBAC, através do Serviço de Autorização. A integração entre o Portal e o CIBAC demonstra-se de fácil implementação, em virtude da grande aceitação e suporte à tecnologia *Web Services* pela maioria das plataformas de desenvolvimento.

4.4 Implementação do Portal

Nesta seção será apresentada a implementação do portal do HUSM. Antes de detalhar os componentes que controlam a segurança do portal, serão destacadas algumas das tecnologias utilizadas nesta implementação.

4.4.1 Tecnologias Utilizadas na Implementação do Portal

O portal foi implementado utilizando os recursos da tecnologia JEE 5(*Java Enterprise Edition*)[6] e também o Servidor *Web Apache Tomcat*[7] (v. 5.5.20). Outras tecnologias foram necessárias à implementação do portal, sendo que podemos dividí-las em: *Tecnologias para a Interface* e *Tecnologias para o processamento no servidor*.

[6]http://java.sun.com/javaee/
[7]http://tomcat.apache.org/

4.4.1.1 Tecnologias para a Interface

Uma aplicação *web* utiliza-se de páginas em HTML, interpretada pelo navegador, para interagir com o usuário, formando a camada de apresentação. Outras tecnologias podem ser misturadas ao HTML para a construção de uma interface mais poderosa, com um visual mais adequado, além de proporcionar recursos que o HTML isoladamente não é capaz. A seguir, um breve resumo sobre essas tecnologias, as quais são responsáveis pela construção da interface com o usuário:

- **HTML:** O *HyperText Markup Language* utiliza os conceitos do *HyperTexto* e da *Hipermídia* para apresentar, num mesmo ambiente: dados, imagens e outros tipos de mídia, como vídeos, sons e gráficos. O HTML é um subconjunto do *Standard Generalized Markup Language* (SGML) e utiliza rótulos (*tags*) que definem a aparência e o formato dos dados, sendo padronizado pelo *Object Management Group* (OMG). É interpretado por qualquer navegador, em qualquer plataforma.

- **CSS:** *Cascading Style Sheet* permite que os estilos dos elementos da página (espaçamento, cores, fontes, margens, etc.) sejam especificados separadamente da estrutura do documento, facilitando dessa forma, uma futura modificação no estilo da página.

4.4.1.2 Tecnologias para o processamento no servidor

Na camada *middleware* (software intermediário), ocorre realmente o trabalho de programação do aplicativo *web*, sendo esta camada a responsável por processar a informação enviada pelo cliente, processar a regra de negócio (que pode estar em outra camada), interagir com o banco de dados, preparar a resposta (quase sempre na forma de uma página HTML) e enviá-la ao cliente. Os componentes dessa camada estão no Servidor *Web* e são capazes de utilizar os recursos desses servidores e dos demais recursos conectados para realizar o processamento. É importante perceber que a forma com que todas essas tecnologias trabalham é similar: recebem uma solicitação do cliente, processam essa solicitação e respondem na forma de uma página HTML. As principais tecnologias utilizadas nesta camada são:

- **Servlets (v. 2.4):** a tecnologia *Java Servlet* fornece aos desenvolvedores *web* um simples e consistente mecanismo de extensão das funcionalidades do Servidor *Web*,

permitindo também o acesso a camada de negócio da aplicação. Possibilitam que aplicações *web* sejam construídas independentes de plataforma e de servidor de aplicação (MICROSYSTEMS, 2006a).

- **JSP (v. 2.0):** *JavaServer Pages* é uma tecnologia que fornece uma rápida maneira de criar conteúdos *web* dinâmicos. Possibilitam também que aplicações *web* sejam construídas independentes de plataforma e de servidor de aplicação. Na verdade, JSP é uma estensão da tecnologia Java Servlet e, por isso, herda todas as caracterís-tivas desta tecnologia (MICROSYSTEMS, 2006b).

- **Jakarta Struts (v. 1.2.9):** o *Apache Struts* foi criado por Craig R. McClanahan e doado para a ASF (*Apache Software Fundation*). Se trata de um *framework* de código aberto que permite a criação de aplicações *web* Java. Este framework usa a arquitetura *Model View Controller* (MVC), que representa, respectivamente, a camada de negócio, a camada de apresentação e a camada de controle da navega-ção (FOUNDATION, 2006).

- **Hibernate (v. 3.1.3):** é um poderoso *framework* de persistência objeto/relacional. Ele transforma os dados tabulares de um banco de dados relacional em um grafo de objetos definido pelo desenvolvedor. Com o Hibernate, o desenvolvedor pratica-mente não se preocupa com o código de acesso a banco de dados e de SQL, o que acelera o desenvolvimento das aplicações (HIBERNATE, 2006).

- **JSTL (v. 1.1.2):** *JavaServer Pages Standard Tag Library* é uma coleção de bibli-otecas de tags customizadas que implementam funcionalidades gerais, comuns a aplicações *web*, incluindo iteração e condição, formatação de dados, manipulação de XML e acesso a base de dados (MICROSYSTEMS, 2006c). Com o consenso praticamente global da utilização do padrão MVC para a criação de aplicativos, principalmente *web*, a utilização de JSP juntamente com a JSTL se mostra uma al-ternativa muito interessante para criação da camada de apresentação, uma vez que com JSTL é possível eliminar praticamente todo o código Java das páginas dinâmi-cas.

4.4.2 Componentes de Segurança do Portal

A figura 4.3 representa a página inicial do portal, na qual é efetuada a autenticação do usuário. Como já foi comentado anteriormente, a primeira ação de determinado usuário após acessar a página inicial do sistema é escolher dentre os tipos de mecanismos disponíveis para efetuar sua autenticação. Após o mesmo ter sido autenticado, será direcionado para a página com todas as aplicações acessíveis por ele. Porém antes desta página ser exibida, o usuário passa por dois filtros[8] conforme mostra a figura 4.4.

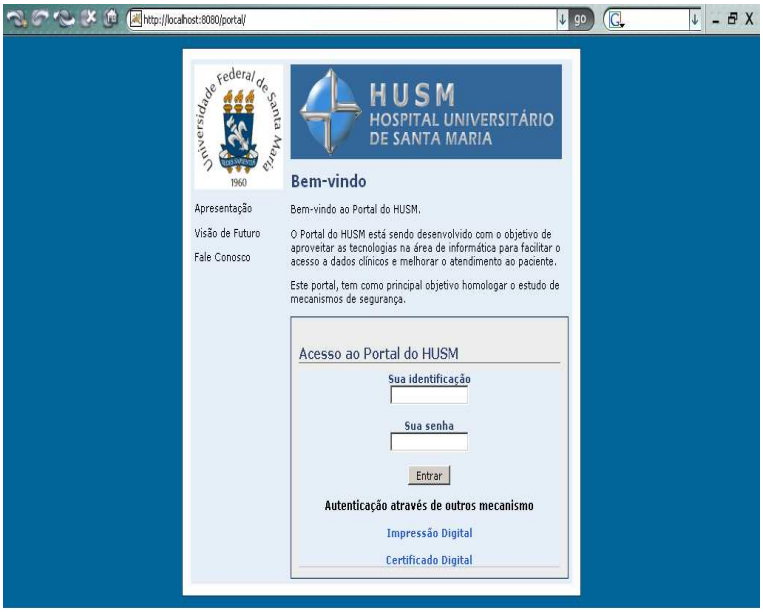

Figura 4.3: Página inicial do portal.

Filtro de Login: após o usuário ter sido autenticado, ele terá que passar pelo *Filtro de Login*. Este filtro apenas verifica se o usuário está logado no sistema para impedir acesso direto à página de listagem das aplicações[9].

4.4.2.1 Filtro de Aplicações

Após o Filtro de Login verificar a autenticidade do usuário, surge o papel do *Filtro de Aplicações*. Tal filtro é responsável por descobrir quais aplicações determinado usuário tem autorização de acesso (passos 2 e 3 da figura 4.1). Três tarefas são executas neste processo:

- **Chamada ao CIBAC:** a primeira tarefa é fazer um pedido ao CIBAC por todas

[8]Filtro: é um objeto capaz de filtrar requisições e/ou respostas a determinados recursos, sendo que um recurso pode ser um *servlet* ou qualquer conteúdo estático.

[9]Este filtro não será detalhado já que foge do escopo deste estudo.

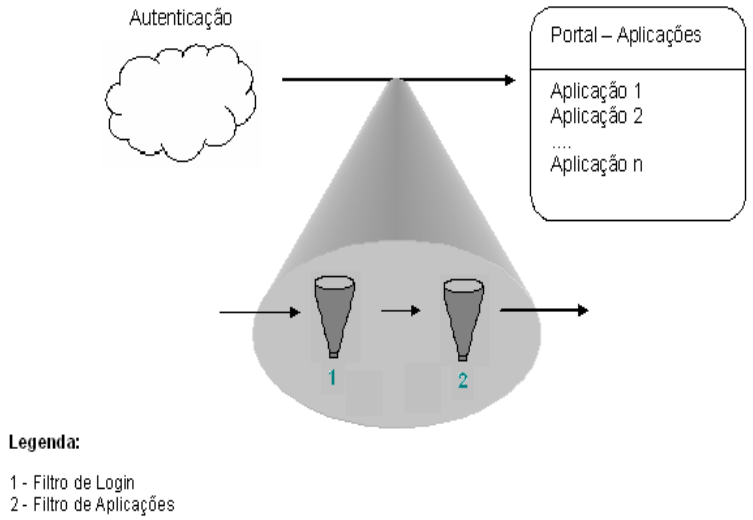

Figura 4.4: Filtros de controle da página de listagem das aplicações.

aplicações que poderão ser acessadas pelo usuário corrente. Para isso, este filtro recupera o identificador do usuário na sessão do mesmo, mantida pelo Servidor Web. De posse deste identificador, o filtro chama o CIBAC passando o identificador (ID) do usuário em uma mensagem SOAP. O CIBAC então, avalia suas políticas e retorna os identificadores das aplicações que este usuário tem autorização de acesso. Estes IDs retornados pelo CIBAC são independentes do nível de autenticação que o usuário possui atualmente no sistema, ou seja, de acordo com a aplicação que este usuário quiser acessar após a lista ser exibida, pode ser necessário se autenticar com um nível maior de confiança.

- **Consulta ao Banco de Dados:** de posse dos IDs das aplicações, o filtro se conecta a um banco de dados que contém uma tabela chamada APLICACOES, com o objetivo de recuperar o nome e a URL (*Uniform Resource Locator*) das mesmas. Esta tabela possui três colunas: ID, NOME e URL. O nome das aplicações é necessário quando a página de listagem das aplicações for renderizada. Já as URLs são necessárias posteriormente, quando o usuário quiser acessá-las.

- **Colocar na sessão do usuário as aplicações:** a última tarefa deste filtro é atualizar a sessão com todas as informações sobre as aplicações do usuário, para posteriormente utilizá-las. O tipo do objeto que contém estas informações é um mapa, como o mostrado, por exemplo, na figura 4.5.

  Pode-se notar que o identificador do mapa é a URL da aplicação e o seu valor é

Chave do Mapa	Valor do Mapa (objetos do tipo *Aplicacao*)				
	id	nome	url	autorizado	acoes
realizacaoExames/	1	Realização De Exames	realizacaoExames/	false	null
consultaLaudos/	2	Consulta Laudos Liberados	consultaLaudos/	false	null
resultadoExames/	3	Resultado dos Exames	resultadoExames/	false	null
...	...	...	...	...	...
...	...	...	...	...	...

Figura 4.5: Mapa com as informações das aplicações.

um objeto do tipo *Aplicacao*. Este objeto possui os seguintes atributos: *id*, *nome*, *url*, *autorizado* e *acoes*. Os atributos *id*, *nome* e *url* representam respectivamente o identificador, o nome e a URL da aplicação. O atributo *autorizado* informa se o usuário tem autorização de acesso a tal aplicação e será explicado depois, junto ao *Servlet de Ações* (seção 4.4.2.2). Já o atributo *acoes* representa as ações que o usuário poderá, posteriormente, executar na aplicação. Após o Filtro de Aplicações ser executado, a página de listagem das aplicações é chamada, exibindo somente as aplicações que o usuário corrente tem autorização de acesso (figura 4.6). Lembrando que o acesso as aplicações depende do nível de confiança que o usuário possui atualmente no sistema.

A partir da página da figura 4.6, o usuário pode acessar as aplicações integradas ao portal. Porém, nesse momento, deve ser verificado junto ao CIBAC, se o usuário tem autorização de acesso a tal aplicação, considerando agora, o nível de confiança que o usuário possui correntemente no sistema (passos 4 e 5 da figura 4.1). O componente que se comunica com o CIBAC é o *Servlet de Ações*.

4.4.2.2 Servlet de Ações

A figura 4.7 representa o papel que este servlet desempenha no portal. Tal *servlet* é responsável por verificar se o usuário tem autorização de acesso a determinada aplicação. Se tiver, deve recuperar quais ações o mesmo pode executar em tal aplicação. Duas tarefas são executas neste processo:

Figura 4.6: Página com a listagem das aplicações.

- **Chamada ao CIBAC :** para chamar o CIBAC, este *servlet* necessita do identificador da aplicação e também do tipo de autenticação que o usuário efetuou anteriormente.

  – **Identificador da Aplicação:** para descobrir o identificador da aplicação, o Servlet de Ações primeiramente resgata a URL da aplicação que o usuário está tentando acessar. De posse desta URL, o servlet busca no mapa (figura 4.5) criado pelo Filtro de Aplicações, todas informações daquela aplicação, inclusive o seu identificador.

  – **Tipo de Autenticação:** o Servlet de Ações descobre o tipo de autenticação efetuada pelo usuário corrente, através de um atributo na sessão do mesmo. Tal atributo deve ser controlado pelo serviço de autenticação.

  De posse destas informações o *Servlet de Ações* pode chamar o CIBAC para verificar se o usuário tem autorização de acesso aquela aplicação (passo 4).

- **Resposta do CIBAC:** O CIBAC avalia suas políticas e descobre se o usuário tem autorização de acesso à determinada aplicação.

  – **Caso o usuário tenha autorização:** no caso do usuário ter autorização, o CIBAC retorna também as ações que o mesmo pode executar nesta aplicação. A partir deste momento, o *servlet* atualiza o mapa com as informações da aplicação (atualiza o atributo *autorizado* para *true* e também coloca as ações

que este usuário pode executar em tal aplicação). Ou seja, a partir deste momento o usuário acessa a aplicação solicitada (passo 6A), sendo que, todas suas ações na aplicação podem ser contraladas de forma segura pelo *Filtro de Ações* (seção 4.4.2.4).

– **Caso o usuário não tenha autorização:** se a resposta do CIBAC for negada, o *Servlet de Ações* volta para página com a listagem das aplicações informando ao usuário da não autorização (passo 5). A partir deste momento o usuário pode elevar seu nível de confiança, autenticando-se através de outro mecanismo (passo 6B).

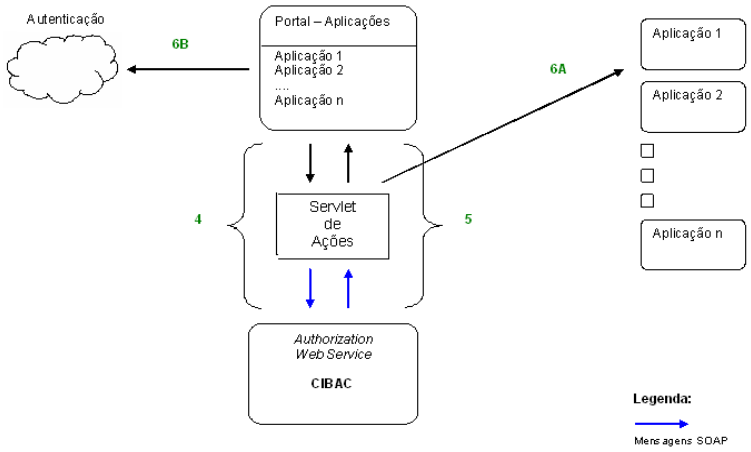

Figura 4.7: Servlet de Ações.

4.4.2.3 Filtro de Autorização

Antes do usuário acessar a aplicação, ele deve passar por dois filtros (passo 6A da figura 4.1). Um deles é o *Filtro de Login* (já comentado anteriormente) e o outro é o *Filtro de Autorização*, como pode ser visualizado na figura 4.8.

Este filtro é responsável por verificar se o usuário corrente está ou não autorizado a acessar as páginas de determinada aplicação. Cada aplicação possui sua variável de controle (atributo *autorizado*), ou seja, o usuário não consegue acessar outra aplicação diretamente sem passar pelo CIBAC. Assim como o *Filtro de Login*, este filtro é mapeado para todas as páginas das aplicações que compõe o portal.

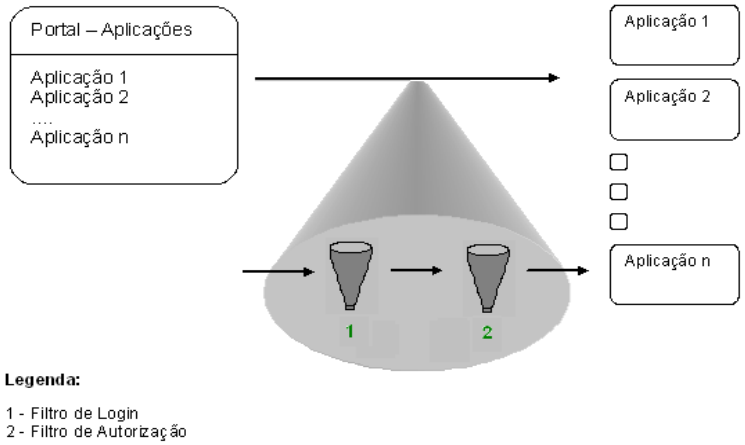

Figura 4.8: Filtros de controle das aplicações.

4.4.2.4 Filtro de Ações

Este filtro é responsável por controlar todas as ações que o usuário executa dentro de uma aplicação. No momento que o usuário acessa determinada aplicação, são colocadas todas as ações permitidas em sua sessão. A descoberta das ações de determinado usuário é responsabilidade do *Servlet de Ações*, explicado na seção 4.4.2.2. Portanto de posse de tais ações, o filtro verifica se aquela ação que o usuário está tentando executar é permitida. Se não for, é informado ao usuário a não autorização. Deve-se observar que, assim como o de *Filtro de Login* e de *Autorização*, este filtro está mapeado para todas ações que o usuário executar nas aplicações.

4.4.3 Políticas utilizadas no Portal

O portal foi modelado para suportar que várias aplicações sejam incorporadas. Dessa forma, cada aplicação deve possuir suas próprias políticas de controle de acesso.

Como exemplo, é analisada algumas partes da política de controle de acesso usada na aplicação *Realização de Exames*. A listagem 4.1 mostra o campo *Target*, responsável por determinar se uma política se aplica a uma determinada requisição de acesso (requisição também no formato XACML). Pode-se notar através do conteúdo encontrado entre o campo *Target* que esta política se aplica ao sujeito que tem a função de *médico* (linha 8) e quer acessar uma *aplicacao* (linha 19) que possui o identificador *1* (linha 26). Se esta política for aplicável, então deve-se avaliar as regras que a compõe. As regras serão discutidas em mais detalhes no capítulo 5.

Listagem 4.1: Campo *Target* de uma política XACML.

```
<Target>
   <Subjects>
      <Subject>
         <SubjectMatch MatchId="..function:regexp-string-match">
            <SubjectAttributeDesignator DataType="../XMLSchema#string"
                    AttributeId="_funcoes"/>
            <AttributeValue DataType="../XMLSchema#string">
                .*# medico #.*
            </AttributeValue>
         </SubjectMatch>
      </Subject>
   </Subjects>
   <Resources>
      <Resource>
         <ResourceMatch MatchId="..function:string-equal">
            <ResourceAttributeDesignator DataType="../XMLSchema#string"
                    AttributeId="..resource:resource-id"/>
            <AttributeValue DataType="../XMLSchema#string">
                aplicacao
            </AttributeValue>
         </ResourceMatch>
         <ResourceMatch MatchId="..function:string-equal">
            <ResourceAttributeDesignator DataType="../XMLSchema#string"
                    AttributeId="_identificador"/>
            <AttributeValue DataType="../XMLSchema#string">
                1
            </AttributeValue>
          </ResourceMatch>
       </Resource>
   </Resources>
</Target>
```

Juntamente ao campo *Target* da política da listagem 4.1, deve-se colocar as ações disponíveis ao sujeito. A listagem 4.2 representa tais ações (linha 5) que devem ser informadas ao portal caso a autorização seja permitida. Além da hora máxima (linha 9) que esta autorização é válida.

Listagem 4.2: Campo *Obligations* de uma política XACML.

```
<Obligations>
   <Obligation ObligationId="Resposta" FulfillOn="Permit">
       <AttributeAssignment AttributeId="acoes"
                            DataType=".../XMLSchema#string">
           criar alterar excluir visualizar
       </AttributeAssignment>
       <AttributeAssignment AttributeId="hora"
                            DataType=".../XMLSchema#time">
           20:00:00
       </AttributeAssignment>
   </Obligation>
</Obligations>
```

A partir da listagem 4.1 pode-se notar que as aplicações são identificadas através de um número (linha 26). Cada aplicação que for incorporada ao portal deve ter sua própria

URL e seu próprio identificador.

4.5 Trilhas de Auditoria

Em decorrência da evolução das fraudes, seja no aspecto quantitativo, seja no aspecto qualitativo, atingindo mundialmente empresas públicas e privadas, a auditoria se torna uma necessidade. Por isso, a cada pedido de autorização feito ao CIBAC, é registrado em um arquivo de *log* as seguintes informações: *identificador do usuário*, *data e hora do pedido de autorização*, *tipo e identificador do objeto de acesso* e *endereço de IP do usuário*. Desta forma, tem-se todas as informações necessárias à depuração.

4.6 Estudo de Caso

Esta seção apresenta a aplicação do SIE-Saúde que foi parcialmente migrada para o ambiente *web*. Como o principal objetivo desta aplicação foi testar alguns componentes da arquitetura de segurança do portal, detalhes de sua implementação não foram considerados.

A aplicação implementada é chamada de *Realização de Exames*. Ela representa apenas uma aplicação que pode ser inicialmente acessada pelo portal, sendo que posteriormente poderão ser incorporadas novas aplicações.

4.6.1 Realização de Exames

A figura 4.9 é usada para registrar a realização de um exame ambulatorial. Ela pode ser utilizada pelos técnicos ou pelos médicos que realizam os seus próprios exames.

De acordo com a figura 4.9, para registrar a realização de um exame, deve-se primeiro localizar os exames agendados (seção 4.6.1.1 - Localizar Exames Agendados). Após localizar um exame que deverá ser realizado, o usuário terá que informar o *Técnico Responsável* e também o *Médico Responsável* (seção 4.6.1.1 - Localizar Profissional Prestador), que são os profissionais responsáveis pela realização do exame.

Após informar todos os dados necessários, podemos salvar estas informações no banco de dados. Caso o paciente tenha passado mal durante a realização do exame há a opção *Exame não realizado* para que seja informado o motivo da não realização. Algumas outras informações podem ser adicionadas através do campo *Observação*.

De acordo com a aplicação que o usuário acessar, apenas determinadas ações estarão

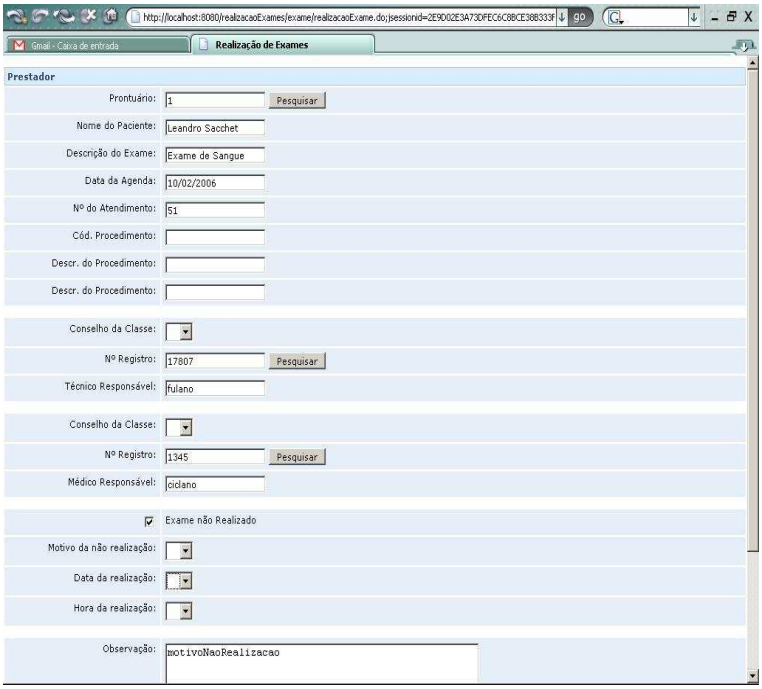

Figura 4.9: Página com o formulário para realização de um exame.

disponíveis para o mesmo. Dessa forma foi possível testar o *Servlet de Ações*, o qual busca junto ao CIBAC quais as ações determinado usuário pode executar dentro de uma aplicação, e também o *Filtro de Ações*, o qual controla as ações que o usuário executa na aplicação.

4.6.1.1 Localizadores Utilizados na Aplicação

Na aplicação de *Realização de Exames* são utilizados alguns campos de localização, conforme ilustra a figura 4.10.

A seguir são destacados alguns destes localizadores:

- **Localizar Exames Agendados:** este localizar permite realizar a busca de exames agendados utilizando diversos filtros de pesquisa.
- **Localizar Exames Realizados:** semelhante ao *Localizar Exames Agendados*, mudando apenas os filtros a serem utilizados e os dados retornados.
- **Localizar Profissional Prestador:** este localizar permite realizar a busca do profissional prestador, ou seja, o profissinal responsável pela realização do exame.

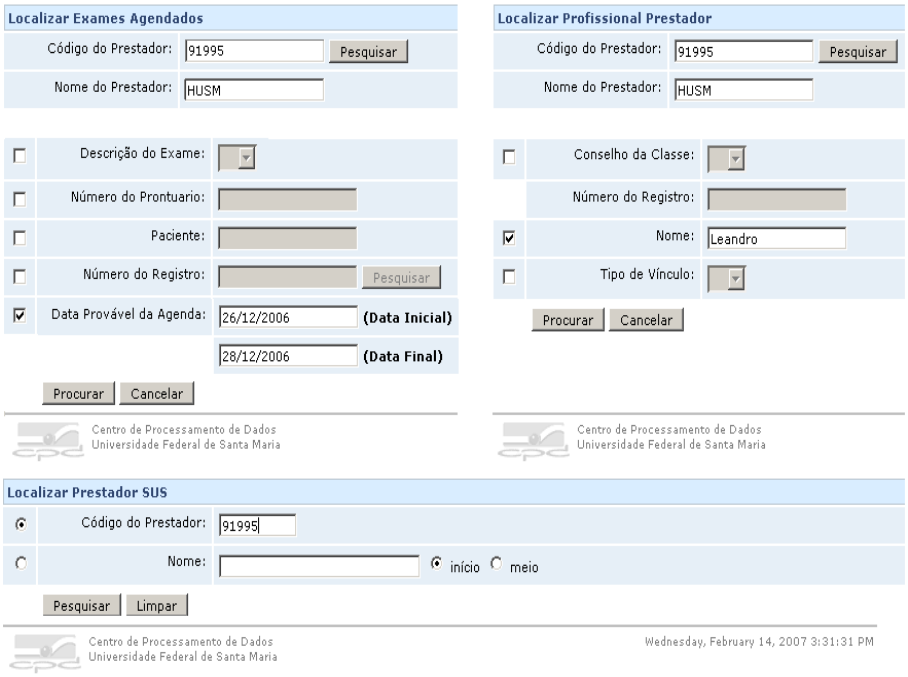

Figura 4.10: Localizadores da aplicação de Realização de Exames.

- **Localizar Prestador:** este localizar permite realizar a busca do prestador, ou seja, a instituição responsável pela realização do exame.

5 REGRAS NEGATIVAS NO CIBAC

Este capítulo detalha como o CIBAC está atualmente, ou seja, explica como são representadas as regras positivas e também descreve o algoritmo utilizado na avaliação de tais regras. Devido ao CIBAC suportar apenas regras positivas, é explicado o que foi incluído e alterado para que o mesmo passasse a suportar regras negativas, mostrando as novas representações e os algoritmos utilizados.

5.1 Importância das Regras Negativas

Regras negativas são muito importantes na descrição de regras de controle de acesso. Tal importância se torna ainda maior quando não se tem uma hierarquia pré-definida na organização que usará tais regras.

Um sistema de controle de acesso sem suporte a regras negativas necessita de uma definição mais abrangente no momento de cadastrar as políticas. Por exemplo, os pacientes do HUSM têm acesso aos seus próprios dados cadastrais, mas não podem criar ou excluir cadastros, incluindo o seu. Como o CIBAC suportava apenas regras positivas, era necessário incluir uma ou mais regras dizendo quais os perfis que tinham autorização para tais ações. Com o suporte a regras negativas, isso se torna uma tarefa simples, já que se escreve apenas uma regra impedindo pacientes de criar e excluir cadastros.

5.2 CIBAC atualmente (antes das regras negativas)

Como foi comentado anteriormente, o CIBAC estabelece o acesso aos recursos através da análise de restrições pré-definidas na forma de políticas e regras de autorização. Cada regra pode ser estruturada como expressões contextuais e estas expressões podem ser agrupadas na forma de cláusulas. A listagem 5.1 mostra a representação comprimida de uma regra positiva no CIBAC.

Listagem 5.1: Representação XML de uma regra positiva no CIBAC

```
<ContextCond>
   <Clause>+
      <Context>+
         <Property/>
         <Operator/>
         <Value/>
      </Context>
   </Clause>
</ContextCond>
```

Para a interação entre as condições de contexto e os contextos pré-definidos, é utilizado o algoritmo de decisão de acesso representado na listagem 5.2:

Listagem 5.2: Algoritmo de Decisão de Acesso

```
1  Receber requisição de acesso
2  Selecionar todas as condições de contexto que satisfaçam a requisição
       de acesso: CE, O e AM
3  Selecionar todas as cláusulas nas condições de contexto
4  Determinar flagc = FALSE              // flag de cláusula
5  Percorrer todas as cláusulas até encontrar uma cláusula verdadeira
6     Selecionar as expressões em cada cláusula
7     Determinar flage = TRUE       // flag de expressão
8     Percorrer as expressões de cada cláusula até encontrar uma falsa
9       Para cada expressão comparar Tipo de Contexto, Propriedade e Valor
            com o Contexto
10      Se o valor da propriedade não satisfaz o operador da regra,
            determinar flage = FALSE
11    Se flage == FALSE
12      Passar para cláusula seguinte
13    Senão flagc = TRUE          // cláusula verdadeira
14 Se flagc == TRUE                // existe um cláusula verdadeira
15    Permitir Acesso
16 Senão Negar Acesso
```

No algoritmo apresentado, uma condição de contexto (CC) é avaliada testando cláusulas, sendo uma cláusula um conjunto de expressões (regras de acesso). A lógica consiste em encontrar uma cláusula verdadeira para autorizar o acesso. Logo, percorrem-se as cláusulas com o objetivo de verificar se todas as expressões internas a ela são verdadeiras.

Na linha 2 são separadas todas as regras que satisfazem a expressão credencial do usuário (CE), o objeto a ser acessado (O) e o modo de acesso (AM). Na linha 3 são selecionadas as cláusulas nas condições de contexto, em seguida determina-se uma condição para se testar cada cláusula através de um *flag* de controle denominado *flag de cláusula* (*flagc*, linha 4) que assume o valor FALSE até que se encontre uma cláusula verdadeira. Outra condição é estabelecida para se testar cada expressão nas cláusulas, através de um *flag* denominado *flag de expressão* (*flage*, linha 7), que assume o valor TRUE até que se encontre uma expressão falsa.

As linhas 9 e 10 do algoritmo exemplificam a operação de comparação da regra de acesso (expressão) com as informações apresentadas pelo contexto (CI), a fim de verificar se o contexto satisfaz a cláusula e, consequentemente, todas as regras de acesso relativas a esta. No momento em que isto ocorre, o acesso é concedido (linhas 14 e 15).

Se alguma expressão no interior da cláusula é falsa (linha 11 do algoritmo), então toda a cláusula é considerada falsa e é necessário percorrer as demais cláusulas com a finalidade de se determinar o acesso. Quando todas as regras (cláusulas) inerentes à requisição de acesso são percorridas e nenhuma delas é verdadeira o acesso é negado (linha 16).

5.3 CIBAC com Regras Negativas

Além da implementação de um portal para o HUSM, outra proposta deste trabalho é dar continuidade ao estudo e implementação do CIBAC, com a inclusão de regras negativas. Afim de manter uma boa especificação de como o CIBAC representa suas regras de acesso, deve-se estabelecer uma forma de diferenciar regras positivas de negativas. Para isso, foi decidido incluir um atributo interno a tag <*ContextCond*>, chamado *Access*. Portanto, as regras positivas e negativas são diferenciadas a partir deste atributo. A listagem 5.3 representa como elas devem ser representadas:

Listagem 5.3: Representação XML para regras positivas e negativas respectivamente.

```
1  <ContextCond Access="Permit">
2     <Clause>+
3        <Context>+
4           <Property/>
5           <Operator/>
6           <Value/>
7        </Context>
8     </Clause>
9  </ContextCond>
10
11 <ContextCond Access="Deny">
12    <Clause>+
13       <Context>+
14          <Property/>
15          <Operator/>
16          <Value/>
17       </Context>
18    </Clause>
19 </ContextCond>
```

5.3.1 Novos Algoritmos

Para que se complete o suporte às regras negativas, o CIBAC necessita de algoritmos que consideram, além das regras positivas, também as negativas. Para resolver este problema, foram implementados três algoritmos, que podem ser visualizados nas listagens 5.4, 5.5 e 5.6[1].

5.3.1.1 Algoritmo comum de avaliação de regras

O algoritmo da listagem 5.4, apresenta como deve ser avaliada cada regra do CIBAC. O método recebe como parâmetro a regra a ser avaliada (*regra*, linha 1) e assim como no algoritmo anterior do CIBAC (listagem 5.2), a lógica consiste em encontrar uma cláusula verdadeira para autorizar o acesso. Logo, percorrem-se as cláusulas com o objetivo de verificar se todas as expressões internas a ela são verdadeiras.

Listagem 5.4: Algoritmo comum de avaliação de regras.

```
1  avalia(Regra regra){
2     booleano flagc = false
3     string tipoRegra = efeito(regra)
4     repete até percorrer todas as cláusulas {
5        booleano flage = true
6        repete até percorrer todas as expressões {
7           se comparaContexto(expressao) == false {
8              flage = false
9              break (sai do laço)
10          }
11       }
12       se flage == true {
13          flagc = true
14          break (sai do laço)
15       }
16    }
17    se flagc == false {
18       retorna "NAO_APLICADO"
19    }
20    se tipoRegra == "Deny" {
21       retorna "NEGADO"
22    }
23    se tipoRegra == "Permit"{
24       retorna "PERMITIDO"
25    }
26    se ocorreu alguma exceção na execução {
27       retorna "INDETERMINADO"
28    }
29 }
```

[1]Estes algoritmos consideram o banco de regras consistente, ou seja, nenhum tipo de conflito é suportado.

A diferença deste algoritmo para o anterior é que ele pode retornar quatro tipos de resultados: "NAO_APLICADO", "INDETERMINADO", "PERMITIDO" ou "NEGADO". Se for encontrada uma cláusula verdadeira, de acordo com o efeito da regra (*efeito()*, linha 3), o algoritmo pode retornar "PERMITIDO" ou "NEGADO". Caso nenhuma regra se aplique ao contexto, "NAO_APLICADO" é retornado. Por fim, se alguma exceção ocorrer durante a execução, "INDETERMINADO" é retornado.

5.3.1.2 Algoritmo de precedência positiva

O algoritmo da listagem 5.5 utiliza o método *avalia* comentado anteriormente na listagem 5.4. Este algoritmo considera que um simples "PERMITIDO" tem precedência sobre qualquer número de "NEGADO", "NAO_APLICADO" ou "INDETERMINADO". Ou seja, se *avalia* retornar um "PERMITIDO", o usuário tem acesso liberado a determinado recurso.

Listagem 5.5: Algoritmo de precedência positiva.

```
1  permitidoPrecedencia(Regra regras[]){
2     booleano peloMenosUmErro = false
3     booleano permissaoPotencial = false
4     booleano peloMenosUmaNegacao = false
5
6     para i de 0 até regras.length {
7        Decisao decisao = avalia(regra[i])
8        se decisao == "NEGADO" {
9           peloMenosUmaNegacao = true
10          vá para a próxima iteração do laço
11       }
12       se decisao == "PERMITIDO" {
13          retorna "PERMITIDO"
14       }
15       se decisao == "NAO_APLICADO" {
16          vá para a próxima iteração do laço
17       }
18       se decisao == "INDETERMINADO" {
19          peloMenosUmErro = true
20          se efeito(regra[i]) == "Permit"
21             permissaoPotencial = true
22          vá para a próxima iteração do laço
23       }
24    }
25
26    se permissaoPotencial == true {
27       retorna "INDETERMINADO"
28    }
29    se peloMenosUmaNegacao == true {
30       retorna "NEGADO"
31    }
32    se peloMenosUmErro == true {
33       retorna "INDETERMINADO"
34    }
```

```
35     retorna "NAO_APLICADO"
36 }
```

5.3.1.3 Algoritmo de precedência negativa

O último dos algoritmos de avaliação é mostrado na listagem 5.6. Ele é semelhante ao da listagem 5.5, só que avalia as regras considerando que um simples "NEGADO" tem precedência sobre qualquer número de "PERMITIDO", "NAO_APLICADO" ou "INDETERMINADO". Ou seja, se o método de avaliação de uma regra (*avalia*), retornar um "NEGADO", este algoritmo considera que o usuário não tem permissão de acesso.

Listagem 5.6: Algoritmo de precedência negativa.

```
1  negadoPrecedencia(Regra regras[]){
2     booleano peloMenosUmErro = false
3     booleano negacaoPotencial = false
4     booleano peloMenosUmaPermissao = false
5
6     para i de 0 até regras.length {
7        Decisao decisao = avalia(regra[i])
8        se decisao == "NEGADO" {
9           retorna "NEGADO"
10       }
11       se decisao == "PERMITIDO" {
12          peloMenosUmaPermissao = true
13          vá para a próxima iteração do laço
14       }
15       se decisao == "NAO_APLICADO" {
16          vá para a próxima iteração do laço
17       }
18       se decisao == "INDETERMINADO" {
19          peloMenosUmErro = true
20          se efeito(regra[i]) == "Deny"
21             negacaoPotencial = true
22          vá para a próxima iteração do laço
23       }
24    }
25
26    se negacaoPotencial == true {
27       retorna "INDETERMINADO"
28    }
29    se peloMenosUmaPermissao == true {
30       retorna "PERMITIDO"
31    }
32    se peloMenosUmErro == true {
33       retorna "INDETERMINADO"
34    }
35    retorna "NAO_APLICADO"
36 }
```

5.3.2 Regras Negativas em Documentos XACML

Para descrição das regras negativas, foi utilizada a linguagem XACML, que já descrevia as regras positivas no CIBAC. A listagem 5.7 mostra um exemplo de como as regras negativas são representadas em documentos XACML. Esta regra usa uma função (linha 2) que lida com um intervalo de tempo, tal intervalo situa-se das 8h às 20h (linha 8 e 11).

Listagem 5.7: Campo *Rule* de uma Política XACML representando uma regra negativa.

```
1  <Rule RuleId="Access" Effect="Deny">
2     <Condition FunctionId="...function#time-in-range">
3        <Apply FunctionId="...function:time-one-and-only">
4           <EnvironmentAttributeDesignator DataType=".../XMLSchema#time"
5                AttributeId="...environment:current-time"/>
6        </Apply>
7        <AttributeValue DataType=".../XMLSchema#time">
8                 08:00:00
9        </AttributeValue>
10       <AttributeValue DataType=".../XMLSchema#time">
11                20:00:00
12       </AttributeValue>
13    </Condition>
14 </Rule>
```

5.4 Casos de Teste

Testes de funcionalidade são de grande importância para um sistema. Em virtude disso é necessário avaliar os algoritmos propostos neste trabalho. Para isso, foram criadas duas situações reais de autorização, descritas nos Casos de Teste 1 e 2[2].

Deve-se lembrar que uma requisição de acesso é uma tupla do tipo (CE, O, AM), onde CE é a expressão credencial do usuário, O é o objeto ou serviço desejado, e AM é o modo de acesso pretendido.

Caso de Teste 1: negação de acesso por existência de cláusula verdadeira. Para a utilização do SIE-Saúde, no âmbito do Hospital Universitário de Santa Maria, os Pacientes têm acesso aos próprios dados cadastrais, mas não podem criar ou excluir cadastros, incluindo o próprio, nem ver cadastros de terceiros; Então, considere a seguinte requisição de acesso: (*paciente*, *cadastro de consulta*, *excluir*). Logo, deve haver uma condição de contexto negativa que suporte a ação de *excluir* para um perfil *paciente* sobre o serviço *cadastro de consulta*, assim existe uma cláusula a ser testada e, portanto o acesso é negado.

[2]Tais testes de funcionalidade foram feitos manualmente, ou seja, não foi implementado nenhum código de teste automático. Esta tarefa poderá ser efetuada em um trabalho futuro, devido a amplitude e a importância da validação da estrutura deste trabalho.

Caso de Teste 2: concessão de acesso por existência de cláusula verdadeira. Suponha a requisição de acesso: (*paciente*, *prescrição*, *visualizar*). Dado que os pacientes podem ver exclusivamente as próprias prescrições, se a prescrição pertencer ao paciente em questão e, havendo uma ou mais cláusulas verdadeiras na condição de contexto que satisfaçam o contexto, o acesso é concedido.

6 CONCLUSÃO

6.1 Conclusões Gerais

O advento e a conseqüente abrangência da Internet nos dias atuais, permitiram a facilidade ao acesso à informações e documentos por qualquer pessoa em qualquer parte do mundo. Num primeiro instante, pode-se avaliar essa situação como amplamente vantajosa, mas, num segundo instante, constatamos que existem determinadas informações que são de caráter confidencial ou sigiloso, pois são capazes de ameaçar, por exemplo, o direito de privacidade de uma pessoa ou instituição.

Este trabalho apresentou um estudo e implementação de autorização no acesso ao portal do HUSM visando principalmente facilitar o acesso a dados clínicos. Para que tais informações pudessem ser disponibilizadas em um ambiente *web*, foram estudados alguns modelos de controle de acesso, dentre eles o CIBAC.

O CIBAC foi o modelo utilizado e, portanto, aprofundado neste trabalho. Para um completo entendimento deste modelo, foi necessário estudar os conceitos envolvidos em uma *Arquitetura Orientada a Serviços*, utilizada na implementação de tal modelo. Além da SOA, foi estudado os principais conceitos na aplicação de *Web Services*, tecnologia fundamental na integração com o CIBAC.

Após um estudo aprofundado do CIBAC, foi possível estendê-lo com a inclusão de regras negativas. Tais regras são muito importantes, pois facilitam a descrição de regras de controle de acesso.

Com a migração parcial da aplicação de *Realização de Exames* para o ambiente *web*, foi possível testar o sistema de autorização que compõe o portal do HUSM. Foi criado uma forma de integração dinâmica de aplicações ao portal, através de um registro de aplicações.

6.2 Trabalhos Futuros

Como propostas para trabalhos futuros, a criação de políticas merece atenção especial. É necessário construir uma interface mais amigável com o usuário e, dessa forma, facilitar o processo de criação. Também é necessário fazer um levantamento sobre que dados devem ser inseridos no CIBAC e como devem ser as políticas de acesso, de forma que melhor atendam os requisitos dos profissionais da área médica.

Além deste trabalho, também poderia ser feito um aplicativo *web* para registro de aplicações. Atualmente todas as aplicações integradas ao portal devem ser registradas diretamente no banco de dados, ou seja, através de comandos SQL. Este aplicativo forneceria uma interface de fácil utilização, tornando este serviço transparente.

6.3 Dificuldades Encontradas

A primeira dificuldade encontrada neste trabalho foi entender o modelo CIBAC. Este modelo exigiu o estudo prévio de diversos conceitos como: SOA, *Web Services*, Contexto, Regras de Controle de Acesso, etc. No entanto, a principal dificuldade encontrada na realização deste trabalho foi a migração de algumas funcionalidades da aplicação do SIE-Saúde, devido a esta aplicação possuir um modelo relacional totalmente desconhecido pelos desenvolvedores.

REFERÊNCIAS

DEY, A. **Towards a Better Understanding of Context and Context-Awareness.** Disponível em: <http://www.cc.gatech.edu/fce/ctk/pubs/HUC99-panel.pdf/>. Acesso em: dez. 2006.

ERL, T. **Service-Oriented Architecture**: A Field Guide to Integrating XML and Web Services. [S.l.]: Prentice Hall, 2004.

ERL, T. **Service-Oriented Architecture**: Concepts, Technology, and Design. [S.l.]: Prentice Hall, 2005.

FERRAIOLO, D. **Proposed NIST Standard for Role-Based Access Control.** [S.l.: s.n.], 2001. Disponível em: <http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf/>. Acesso em Nov/2006.

FOUNDATION, A. S. **Struts**. Disponível em: <http://struts.apache.org/>. Acesso em: dez. 2006.

GRIFFIN, P. **Introduction to XACML**. Disponível em: <http://dev2dev.bea.com/pub/a/2004/02/xacml.html/>. Acesso em: dez. 2006.

HIBERNATE. **Hibernate Overview**. Disponível em: <http://www.hibernate.org/>. Acesso em: dez. 2006.

IEEE. **Recommended Practice for Architectural Description of Software-Intensive Systems.** Disponível em: <http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-2000.pdf/>. Acesso em Nov/2006.

JOSHI, J. **Security Models for Web-Based Applications.** [S.l.: s.n.], 2001. Disponível em: <http://citeseer.ist.psu.edu/579962.html/>. Acesso em Dez/2006.

KRAFZIG, D. **Enterprise SOA**: Service-Oriented Architecture Best Practices. [S.l.]: Prentice Hall, 2004.

KUMAR, A. **Context Sensitivity in Role-Based Access Control.** [S.l.: s.n.], 2002. Disponível em: <http://portal.acm.org//>. Acesso em Dez/2006.

MICROSYSTEMS, S. **Java Servlet Technology**. Disponível em: <http://java.sun.com/products/servlet/overview.html/>. Acesso em: dez. 2006.

MICROSYSTEMS, S. **JavaServer Pages**. Disponível em: <http://java.sun.com/products/jsp/overview.html/>. Acesso em: dez. 2006.

MICROSYSTEMS, S. **JavaServer Pages Standard Tag Library**. Disponível em: <http://java.sun.com/products/jsp/jstl/>. Acesso em: dez. 2006.

MOTTA, G. **Um Modelo de Autorização Contextual para o Controle de Acesso ao Prontuário Eletrônico do Paciente em Ambientes Abertos e Distribuídos.** Disponível em: <http://www.ppgia.pucpr.br/ maziero/pesquisa/ceseg/wseg02/18.pdf/>. Acesso em: dez. 2006.

MüLLER, E. M. **Estudo e Implementação de Autenticação no Acesso para o Portal do HUSM.** Monografia (Bacharel em Ciência da Computação) - Universidade Federal de Santa Maria, Santa Maria, 2007.

PAPAZOGLOU, M. **Service-Oriented Computing**: concepts, Characteristics and Directions. Disponível em: <http://infolab.uvt.nl/pub/papazogloump-2003-51.pdf/>. Acesso em: out. 2006.

RAVI SANDHU, P. S. **Access Control**: Principles and Practice. Disponível em: <http://www.list.gmu.edu/journals/commun/i94ac(org).pdf/>. Acesso em Nov/2006.

SOARES, G. A. **Utilização de Informações Contextuais em um Modelo de Controle de Acesso a Informações Médicas**. XXXII Conferência Latinoamericana de Informática CLEI, Santiago, Chile.

SOUZA, C. A. G. de. **Implementação do CIBAC no SIE usando SOA**. Monografia (Bacharel em Ciência da Computação) - Universidade Federal de Santa Maria, Santa Maria, 2006.

WEB. **Web Service Architecture Requirements.** Disponível em: <http://www.w3.org/TR/wsa-reqs/>. Acesso em Nov/2006.