UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

UMA EXTENSÃO DO XACML PARA FEDERAÇÃO DE IDENTIDADES EM NUVEM COMPUTACIONAL

ALEX FELIPE RAULINO

BLUMENAU
2013

2013/1-01

ALEX FELIPE RAULINO

UMA EXTENSÃO DO XACML PARA FEDERAÇÃO DE IDENTIDADES EM NUVEM COMPUTACIONAL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Paulo Fernando da Silva, Mestre – Orientador

BLUMENAU
2013

2013/1-XX

UMA EXTENSÃO DO XACML PARA FEDERAÇÃO DE IDENTIDADES EM NUVEM COMPUTACIONAL

Por

ALEX FELIPE RAULINO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo Fernando da Silva, Mestre – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Dr. – FURB
Membro:	Prof. Jhony Alceu Pereira, Especialista – FURB

Blumenau, 08 de julho de 2013

Dedico este trabalho a minha família e todos os meus amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que me apoio e sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, por ter acreditado na realização e conclusão deste trabalho.

RESUMO

O *eXtensible Access Control Markup Language* (XACML) descreve uma linguagem para políticas de controle de acesso e também um formato para mensagens de pedido e resposta, porém tem a limitação de não proporcionar a gerência de identidade federadas e não tem especificado uma forma para identificar a organização da federação. Portanto este trabalho apresenta a implementação de uma extensão XACML que aplica o conceito de federação de identidade ao controle de acesso. Também são apresentadas três aplicações de testes que demonstram a utilização da extensão XACML. A partir dos testes efetuados, demonstrou-se que com a extensão XACML é possível prover federação de identidades. A extensão XACML foi desenvolvida tendo como base a API Sun's XACML 1.2 que já disponibiliza uma implementação básica dos elementos que compõem a linguagem XACML. No trabalho é apresentado um caso de aplicação da extensão XACML na criação da federação de identidades entre duas nuvens. Por fim é apresentada uma conclusão descrevendo se as ferramentas utilizadas foram adequadas e quais as principais contribuições do trabalho para o avanço do padrão XACML.

Palavras-chave: XACML. Controle de acesso. Federação de identidades.

ABSTRACT

The eXtensible Access Control Markup Language (XACML) describes a language for access control policies and also a format for request and response messages, but has the limitation of not providing federated identity management and has not specified a way to identify the organization of the federation. Therefore this paper presents the implementation of an extension to XACML applies the concept of identity federation to access control. Also shown are three applications of tests that demonstrate the use of the extension XACML. From the tests conducted, it was demonstrated that it is possible to provide XACML extension identity federation. Extending XACML was developed based on the API Sun's XACML 1.2 already provides a basic implementation of the elements of the XACML language. In the paper we present a case for the extension of the XACML in creating identity federation between two clouds. Finally a conclusion is presented describing the tools used were appropriate and what the main contributions to the advancement of the XACML standard.

Key-words: XACML. Access control. Identity federation.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

EC2 – amazon *Elastic Compute cloud*

FURB – Fundação Universidade Regional de Blumenau

IdP – *Identity Provider*

JEE – Java *Enterprise Edition*

JSON – *JavaScript Object Notation*

MEC – Ministério da Educação

PAP – *Policy Administration Point*

PDP – *Policy Decision Point*

PEP – *Policy Enforcement Point*

PIP – *Policy Information Point*

QoS – *Quality of Service*

RSVP – *Resource reSerVation Protocol*

SP – *Service Provider*

Tspec – *Traffic SPECification*

UFRJ – Universidade Federal do Rio de Janeiro

UML – *Unified Modeling Language*

WSDL – *Web Services Description Language*

XACML – *eXtensible Access Control Markup Language*

XML – *eXtensible Markup Language*

XSD – XML *Schema Definition*

SUMÁRIO

1 INTRODUÇÃO

Os serviços disponibilizados na nuvem podem alavancar a eficiência e eficácia nas operações empresariais, principalmente melhorando o custo-benefício, alta disponibilidade, escalabilidade, grande capacidade de tolerância a falhas, entre outros benefícios (LEANDRO, 2012, p. 23). Porém muitas das vezes existe a necessidade de realizar integrações, ou até mesmo fusões ou criação de alianças, onde os serviços, dados de clientes e recursos precisam estar disponíveis de forma imediata e transparente, sem ter que manter várias senhas e utilizar diferentes meios de acesso aos recursos (SILVA, 2009). Com o aumento deste tipo de prática, há uma grande necessidade de gerência de identidade, no que diz respeito a facilitar a inserção e remoção de usuários, garantir a segurança das informações, gerenciar as políticas de acesso, para que estes serviços sejam utilizados efetivamente por indivíduos autorizados.

Conforme Leandro (2012, p. 24), uma federação é uma forma de associação de parceiros de uma rede colaborativa, onde o compartilhamento de identidades é a chave para alcançar harmonia entre competência de negócio e eficiência tecnológica, permitindo que as organizações desta federação interajam com base na gestão da identidade e políticas de acesso.

Geralmente cada sistema implementa sua linguagem própria para definição de políticas de controle de acesso e gerenciamento de identidade, dando assim um fator limitante para a concepção de sistemas distribuídos e abertos. Visando garantir a interoperabilidade entre os diversos sistemas, o órgão *Organization for the Advancement of Structured Information Standards* (OASIS, 2005a) lançou o XACML, um sistema de políticas de propósito geral, baseado em *eXtensible Markup Language* (XML).

O XACML descreve uma linguagem para políticas de controle de acesso e também um formato para mensagens de pedido e resposta, porém tem a limitação de não proporcionar a gerência de identidade federadas, não tem especificado uma forma para identificar a organização da federação, ou então representar na política um recurso especifico de uma organização. O trabalho criará uma extensão XACML para federação de identidades em nuvem, onde será especificada uma forma de tratamento de federação de identidades, proporcionando um avanço para o padrão.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é a criação de uma extensão da linguagem XACML para federação de identidades em nuvem.

Os objetivos específicos do trabalho são:

a) gerenciar políticas de acesso no conceito de federação de identidades;
b) interpretar *Request* e *Response* do padrão XACML no conceito de federação de identidades;
c) identificar a política a ser aplicada no conceito de federação de identidades.

1.2 ESTRUTURA DO TRABALHO

O trabalho está desenvolvido em quatro capítulos. O segundo capítulo do presente trabalho refere-se à fundamentação teórica necessária para o seu entendimento.

O terceiro capítulo contém a descrição de como ocorreu o desenvolvimento da extensão XACML e das aplicações de testes, os casos de uso envolvidos, os diagramas de classe e documentações necessárias. Ainda, são apresentados os resultados e discussões ocorridas durante o desenvolvimento.

Ao final, o quarto capítulo apresenta as conclusões do presente trabalhando, apresentando também sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está organizado em cinco seções: a primeira trata segurança e autorização, a segunda descreve nuvem computacional, a terceira descreve federação de identidades, a quarta apresenta conceitos de XACML e a última seção traz trabalhos correlatos ao trabalho proposto.

2.1 SEGURANÇA E AUTORIZAÇÃO

A segurança da informação é a proteção dos sistemas de informação contra a negação de serviço a usuários autorizados, assim como contra a intrusão e a modificação não autorizada de dados ou informações, armazenados em processamento ou em trânsito (LAUREANO, 2005).

Portando a autorização em segurança da informação é o mecanismo responsável por garantir, que apenas usuários devidamente autorizados consumam os recursos de um sistema computacional. Os recursos a serem considerados podem ser arquivos, servidores, softwares, funcionalidade, módulos de um sistema, periféricos e dispositivos de hardware, Já os consumidores podem ser programas, usuários através de um sistema ou hardware.

Para garantir a segurança e um controle de autorização adequado é necessário realizar uma autenticação, de modo a garantir de que o usuário ou o objeto remoto é autentico e confirmar a sua procedência. É um serviço essencial de segurança, pois uma autenticação confiável assegura o controle de acesso, determina quem está autorizado a ter acesso à informação, permite trilhas de auditoria e assegura a legitimidade do acesso (LAUREANO, 2005).

Conforme descreve Laureano (2005), segue alguns princípios básicos para garantir a segurança da informação são:

a) confidencialidade: a informação somente pode ser acessada por pessoas explicitamente autorizadas. É a proteção de sistemas de informação para impedir que pessoas não autorizadas tenham acesso ao mesmo. O aspecto mais importante deste item é garantir a identificação e autenticação das partes envolvidas;
b) disponibilidade: a informação ou sistema de computador deve estar disponível no momento em que a mesma for necessária;
c) integridade: a informação deve ser retornada em sua forma original no momento em que foi armazenada. É a proteção dos dados ou informações contra modificações intencionais ou acidentais não autorizadas.

O sistema que administra as informações ainda deve respeitar os seguintes itens (SANDHU; SAMARATI, 1996):

a) autenticidade: garante que a informação ou o usuário da mesma é autêntico. Atesta com exatidão a origem do dado ou informação;
b) não repúdio: não é possível negar (no sentido de dizer que não foi feito) uma operação ou serviço que modificou ou criou uma informação. Não é possível negar o envio ou recepção de uma informação ou dado;
c) legalidade: garante a legalidade (jurídica) da informação e aderência de um sistema à legislação. Característica das informações que possuem valor legal dentro de um processo de comunicação, onde todos os ativos estão de acordo com as cláusulas contratuais pactuadas ou a legislação política institucional, nacional ou internacional vigentes;
d) privacidade: foge do aspecto de confidencialidade, pois uma informação pode ser considerada confidencial, mas não privada. Uma informação privada deve ser vista / lida / alterada somente pelo seu dono. Garante ainda que a informação não será disponibilizada para outras pessoas (nesse caso é atribuído o caráter de confidencialidade a informação). É a capacidade de um usuário realizar ações em um sistema sem que seja identificado;
e) auditoria: rastreabilidade dos diversos passos que um negócio ou processo realizou ou que uma informação foi submetida, identificando os participantes, os locais e horários de cada etapa. Auditoria em software significa uma parte da aplicação, ou conjunto de funções do sistema, que viabiliza uma auditoria. Consiste no exame do histórico dos eventos dentro de um sistema para determinar quando e onde ocorreu uma violação de segurança.

2.2 NUVEM COMPUTACIONAL

Computação em nuvem é um termo em evolução que descreve o desenvolvimento de muitas das tecnologias e abordagens existentes em computação. A nuvem separa as aplicações e os recursos de informação de sua infraestrutura básica, e os mecanismos utilizados para entregá-los (LEANDRO, 2012, p. 29).

A computação em nuvem é um modelo que permite o acesso conveniente à rede sob demanda para um conjunto compartilhado de recursos de computação configuráveis que podem ser rapidamente disponibilizados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviços.

Leandro (2012, p. 29) descreve que a computação em nuvem tem cincos características essenciais, as quais são:

a) autoatendimento sob demanda: um consumidor pode configurar sua capacidades de computação, como tempo de servidor e armazenamento em rede, conforme necessário, automaticamente, sem a necessidade de interação humana com cada prestador de serviço;
b) amplo acesso à rede: recursos são disponibilizados através da rede e acessados por meio de mecanismos-padrão que promovam o uso por plataformas-cliente heterogêneas com qualquer capacidade de processamento;
c) *pool* de recursos: os recursos de computação do provedor são agrupados para atender múltiplos consumidores através de um modelo multi-inquilino, com diferentes recursos físicos e virtuais atribuídos dinamicamente e redesignados novamente de acordo com a demanda do consumidor;
d) elasticidade rápida: os limites computacionais da nuvem podem ser elasticamente provisionadas e liberadas, em alguns casos automaticamente, para se ajustar à escala, crescente ou decrescente, compatível com a demanda;
e) serviço medido: sistemas em nuvem controlam e otimizam automaticamente o uso dos recursos, aproveitando uma capacidade de medição em algum nível de abstração apropriado para o tipo de serviço.

Com as cinco características relacionadas é possível dividir a entrega de serviços de nuvem em três modelos de arquiteturas.

a) software como um serviço (*Software as a Service* - SaaS): o fornecedor do software se responsabiliza por toda a estrutura necessária para a disponibilização do sistema (servidores, conectividade, cuidados com segurança da informação) e o cliente utiliza o software via internet;
b) plataforma como um serviço (*Platform as a Service* - PaaS): serviço propriamente dito de hospedagem e implementação de hardware e software para o uso e acesso de aplicativos por meio da interne;
c) infraestrutura como um serviço (*Infrastructure as a Service* - IaaS): fornecimento de infraestrutura computacional como um serviço.

Os controles de segurança e acesso para computação em nuvem, em geral, não são diferentes dos controles de segurança para outros ambientes de TI. No entanto, a computação em nuvem pode apresentar riscos diferentes para uma organização quando comparada com as soluções tradicionais de tecnologia da informação.

2.3 FEDERAÇÃO DE IDENTIDADES

A demanda pelo compartilhamento de informações de forma transparente e segura é uma exigência cada vez mais atual e com requisitos rigorosos, principalmente no contexto da internet onde as mudanças são rápidas, podendo gerar situações extremamente drásticas, caso não for usado uma tecnologia adequada que proporciona esta flexibilidade (MELLO, 2006).

A federação de identidades é um meio que permite as organizações gerenciar de forma unificada os perfis dos utilizadores, seus atributos e regras de acesso, utilizando um conjunto de regras e tecnologias, de modo que seja possível compartilhar de forma segura os recursos disponíveis. Então por meio da federação de identidade, usuários que utilizem o mesmo nome de identificador e *password* podem acessar recursos de mais do que uma organização. Isso permite que organizações partilhem aplicações sem a necessidade de adaptar as mesmas tecnologias para serviços de diretório, segurança e autenticação. A tecnologia de federação de identidade é assim utilizada para criar uma identidade global, interoperável, conduzindo relacionamentos ou modelos de negócio com afinidades (FAÍSCA, 2009, p. 3).

Segundo Silva (2009) os objetivos da federação de identidades são:

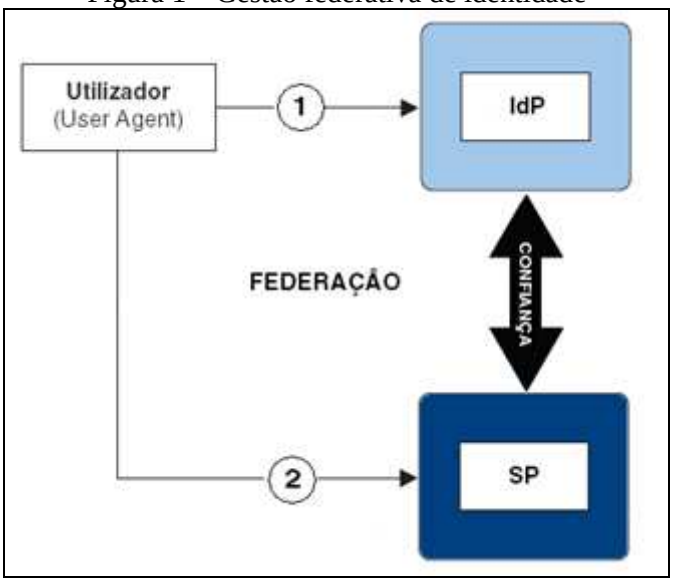
a) reduzir os custos de gestão de identidades;
b) melhorar a experiência do usuário;
c) prover segurança e confiança *end-to-end* na integração de aplicações inter-organizacionais.

Silva (2009) também descreve as formas de federação de identidades, as quais são:

a) *Single Sign-On* (SSO) com conta vinculada: o ID do usuário pode diferir entre sites;
b) *attribute federation*: atributos do usuário são usados para fazer o *link* da conta;
c) *persistent federation*: um *name identifier* constante é usado para identificar cada utilizador;
d) *transient federation*: suporta *role-based* e mapeamento de identidade anônima.

A gestão federativa de identidade implica na existência de *Identity Provider* (IdP) e *Service Provider* (SP), o IdP é responsável por assegurar a identidade do utilizador na federação. Já o SP é o parceiro de validação na transação federada, dessa forma o IdP e SP criam entre si uma infraestrutura de confiança. O utilizador é autenticado pelo IdP em seguida visita o SP utilizando a informação de segurança produzida pelo IdP, conforme representado na Figura 1.

Figura 1 – Gestão federativa de identidade



Fonte: Faísca (2009).

2.4 XACML

O XACML é um padrão proposto pela OASIS (2005a) que define uma linguagem para modelar, armazenar e distribuir políticas, sendo um modelo voltado para representação de políticas descritivas e disponibilizando uma forma para realizar consultas com resposta para identificar se determinada ação pode ser realizada.

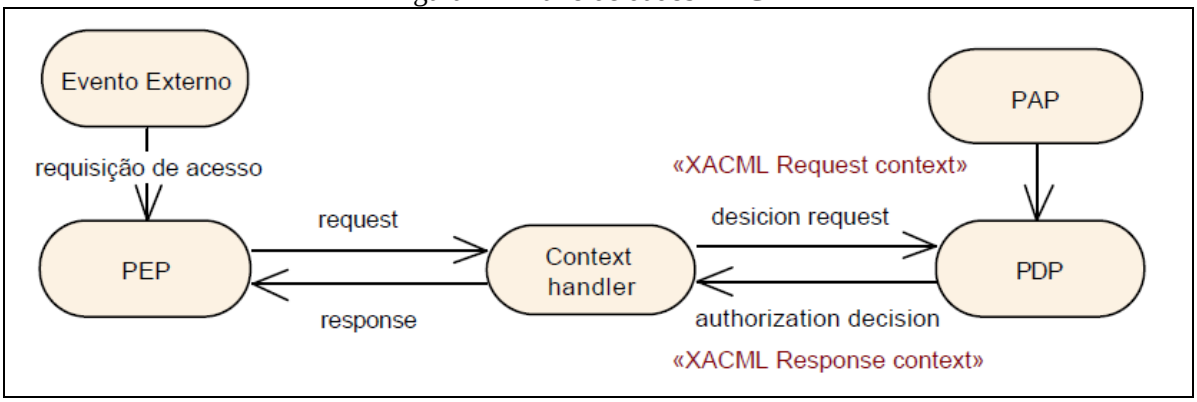
A representação da política com XACML é utilizada para descrever os requisitos gerais de controle de acesso e têm pontos de extensão padrão para definição de novas funções, tipos de dados, tornando este padrão bem extensível (OASIS, 2005a).

O formato de pedido e resposta descreve como as consultas sobre o sistema de políticas deverão ser realizadas (pedido) e como deverão ser as respostas.

O XACML define o fluxo de dados envolvendo duas entidades principais, o *Policy Enforcement Point* (PEP) e o *Policy Decision Point* (PDP), conforme pode ser visto na Figura 2. As políticas ou conjunto delas são descritas por um administrador chamado *Policy Administration Point* (PAP) que serão utilizadas pelo PDP. Um evento externo invoca um PEP gerando uma requisição de acesso e envia ao *context handler*, contendo dados necessários para uma avaliação de política. O *context handler* gera uma requisição no formato XACML *Request context*, chamada *decision request* e disponibiliza ao PDP para avaliação. O *context handler* é uma entidade do sistema responsável por converter os dados requisitados pelo PEP em forma nativa para o modelo XACML *Request context*, possibilitando o processamento do PDP. A avaliação da decisão do PDP, chamada de *authorization decision*, é gerada no modelo XACML *Response context* e encaminhada ao *context handler* que converte

para o formato nativo encaminhando a resposta ao PEP. O *context handler* sempre intermediará qualquer transação entre o PEP e o PDP, garantindo a conversão de informações entre o formato XACML *context* e dados nativos, recebidos ou enviados ao PEP (Toktar, 2003, p. 57).

Figura 2 – Fluxo de dados XACML



Fonte: Faísca (2009).

2.4.1 *Policy* e *PolicySet*

Um repositório de políticas XACML é composto por *Policys* e *PolicySets*. A *PolicySet* é um recipiente que pode conter outras *Policys* ou *PolicySets*, bem como referências a políticas encontradas em locais remotos. A *Policy* representa uma única política de controle de acesso, composta por um conjunto de *Rules* (OASIS, 2005a).

Uma *Policy* ou *PolicySet* pode conter várias *Policys* ou *Rules*, pois cada um pode avaliar as diferentes decisões de controle de acesso e por isso o padrão XACML tem um mecanismo para avaliar e de alguma forma retornar no *Resquest* uma das decisões encontrada de cada um. Isto é feito através de um conjunto de algoritmos de combinação. Cada algoritmo representa uma maneira diferente de combinar várias decisões em uma única decisão. Há dois tipos de algoritmo de combinação, algoritmos de combinação de *Policys* usado por *PolicySets* e algoritmos de combinação de *Rules* usado por *Policys*. Um exemplo deles é o *Deny Overrides Algorithm*, que diz que não importa o que, se qualquer decisão for *deny*, ou *permit*, o resultado final sempre será *deny*. Os algoritmos de combinação também são usados para construir políticas cada vez mais complexas, pois embora existam vários algoritmos padrão, é possível construir um algoritmo de combinação personalizado para outras necessidades (OASIS, 2005a).

2.4.1.1 *Target* e *Rules*

Uma das responsabilidades do PDP é encontrar *Policys* que se aplicam a um determinado *Request*. Para fazer isso, o padrão XACML fornece outro recurso chamado *Target*. Um *Target* é basicamente um conjunto de condições simplificadas que são *Subject*, *Resource* e *Action* e que devem ser verificadas em uma *PolicySet*, *Policy* ou *Rule* para que se aplique a um determinado pedido. Se todas as condições do *Target* forem satisfeitas, então o *PolicySet*, *Policy* ou *Rule* se aplica ao pedido. Além de ser uma maneira de verificar a aplicabilidade, as informações do *Target* também fornece uma maneira simples de criar índices de *Policy*s, o que é útil se você precisa armazenar muitas *Policys* e analisa-las rapidamente. Por exemplo, uma *Policy* pode conter um *Target* que só se aplica aos *Resquest*s de um serviço específico. Quando uma solicitação para acessar esse serviço chega, o PDP vai saber onde procurar as *Policys* que podem ser aplicadas a este pedido porque as *Policys* vão estar indexadas com base nas condições dos *Targets* (OASIS, 2005a).

Uma vez que a *Policy* é encontrada e verificada como aplicável para um *Request*, as *Rules* são avaliadas. Uma *Policy* pode ter inúmeras *Rules*. O coração da maioria das regras é uma condição *booleana*. Se a condição avaliada for verdadeira, então o *effect* da *Rule*, *permit* ou *deny* é retornado. Se a condição avaliada for falsa, é retornado *not applicable*. A avaliação de uma condição pode também resultar em erro e neste caso é retornado *indeterminate*. É importante resaltar que a criação de um algoritmo de combinação deve tratar todos os retornos possíveis de uma avaliação, assim garantindo que a escolha final seja correta (OASIS, 2005a).

2.4.2 *Attributes* e *Attribute Value*

Dentro do padrão XACML as informações ou valores de uma *Policy* ou de um *Resquest* são representados por *Attributes.* Um *Attribute* é composto por um *Attribute Value*. Todos os *Attributes* são definidos por um tipo conhecido, que pode ser uma data, texto, número, hora ou até um tipo customizado. O *Attribute* também contém um *id* para que seja possível relacionar os *Attributes* da *Policys* e *Requests*. Especificamente nas *Policys*, os *Attributes* vão representar as informações dos *Targets*, que são *Subjects*, *Resources*, *Actions*.

Quando um pedido é enviado a partir do PEP para o PDP, esse *Request* é formado exclusivamente de *Attributes* e eles são comparados com os *Attributes* do *Target* das *Policys* para tomar as decisões de acesso, essa comparação é realizada comparando os A*ttribute Values* dos *Attributes* de *ids* iguais. Caso a *Policy* não conseguir encontrar um *Attribute* no Request, o PDP solicita ao PIP o *Attribute Value* desse *Attribute*, porém o PIP pode retornar vários *Attribute Value*, por isso o padrão XACML fornece um tipo de *Attribute* especial

chamado *bag*. As *bags* são conjuntos desordenados que permitem desde *Attribute Value* duplicados a *bags* vazias. Quando o PIP retorna uma *bag* de *Atribute Value*, os *Attributes Value* são comparados com os *Attribute Value* do *Attribute* procurado e dessa forma avalia a *Policy* (SUN, 2004a).

2.4.3 *Request* e *Result*

Além de definir um formato padrão para *Policy*, o XACML define uma forma padrão de expressar *Requests* e *Results*. O *Request* pode conter vários *Attributes* para representar *Subjects*, *Resources* e *Actions*. E cada *Attribute* deverá conter um *id* para identificá-lo nas avaliações como pode ser visto no Quadro 1.

O *Result* consiste em um ou mais resultados, cada uma das quais representa o resultado de uma avaliação. Vários resultados só podem ser causados por uma avaliação de um recurso hierárquico, mas normalmente haverá apenas um resultado numa *Result*. Cada *Result* contém uma decisão que são p*ermit*, d*eny*, *not applicable*, ou *indeterminate*. A principal funcionalidade do *Request* e *Result* é fornecer um formato padrão para interagir com um PDP.

Quadro 1 – Exemplo de *Request*

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Subject>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                   DataType="http://www.w3.org/2001/XMLSchema#integer">
            <AttributeValue>100</AttributeValue>
        </Attribute>
    </Subject>
    <Resource>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                   DataType="http://www.w3.org/2001/XMLSchema#anyURI">
            <AttributeValue>http://server.example.com/</AttributeValue>
        </Attribute>
    </Resource>
    <Action>
    </Action>
    <Federation>
    </Federation>
</Request>
```

2.5 TRABALHOS CORRELATOS

A seguir são apresentados dois trabalhos vinculados aos assuntos de Federação de Identidades e Extensão de XACML, os quais são: “Federação de Identidades e Computação em Nuvem: Estudo de Caso Usando Shibboleth” (LEANDRO, 2012) e “Controle de Admissão de *Resource reSerVation Protocol* (RSVP) utilizando XACML” (TOKTAR, 2003).

2.5.1 Federação de Identidades e Computação em Nuvem: Estudo de Caso Usando Shibboleth

De acordo com Leandro (2012, p. 25), “O objetivo geral deste trabalho é apresentar um estudo de caso do uso da ferramenta Shibboleth, destacando o conceito de gerenciamento de identidades federadas em um ambiente de nuvem computacional.”.

No trabalho são demonstrados vários conceitos de Gerenciamento de Identidades, dentre eles, descreveu as características de computação em nuvem, os requisitos de um sistema de gerenciamento de identidades e modelos de gerenciamentos de identidades.

Sobre a ferramenta Shibboleth, no trabalho foi visível sua importância ao dar apoio às tarefas de gerenciamento de identidades e grande facilidade no que diz respeito à configuração do controle de acesso, seja num ambiente comum, ou num ambiente de nuvem. Mas por outro lado, considerou inicialmente uma ferramenta complexa, consequentemente necessitando de uma análise detalhada sobre seu funcionamento e estrutura.

No desenvolvimento do trabalho foi abortada a utilização do serviço de computação em nuvem Amazon *Elastic Compute Cloud* (EC2), na qual foi caracterizado com grande escalabilidade, disponibilidade, elasticidade e alto desempenho. Os recursos de infraestrutura que o EC2 disponibiliza podem ser acessados por meio de APIs, ou ferramentas da web e utilitários, tornando-a uma infraestrutura completa para computação em diversos níveis de processamento, desde tarefas simples até de alto desempenho e possui uma gerência eficaz dos recursos.

Em sua conclusão descreve que o uso de federação de identidades tem um papel vital ao permitir que organizações autentiquem seus usuários de serviços de nuvem. Também ressalta a preocupação em prover controle de acesso, para que esses serviços sejam efetivamente usados pelas organizações.

2.5.2 Controle de Admissão de RSVP Utilizando XACML

O principal objetivo do trabalho de Toktar (2003) é a criação de uma extensão da estrutura XACML, utilizando o protocolo RSVP para controlar o acesso aos recursos da rede e determinar a quantidade de recursos que será disponibilizada ao usuário.

Demonstrou que o padrão XACML é flexível e extensível, suportando novas exigências no modelo de políticas. No trabalho foi utilizado a arquitetura de implementação da SUN XACML em Java para suportar XACML, que segue a definição da OASIS, porém não descreveu se as implementações necessárias foram complexas.

A principal modificação para a nova extensão XACML, foi a criação da *tag ResourceRsvp* que foi estendida da *tag Resource*. A nova *tag* foi desenvolvida para suportar a *classes* de aplicações QoS, tipo de serviço, estilo e especificação para o tráfego de dados *Traffic Specification* (*Tspec)*, necessários à configuração em servidores *Quality of Service* (QoS) *Resource reSerVation Protocol* (RSVP). Também demonstrou a importância de especificar a nova *tag* no *policy schema* que define a estrutura para validação políticas, *schema* chamado “cs-xacml-schema-policy-01.xsd”.

De forma organizada, mostrou um comparativo entre a estrutura do XACML modificado e a original, dando ênfase em manter suas funcionalidades originais.

3 DESENVOLVIMENTO

Neste capítulo são apresentadas as etapas do desenvolvimento da extensão XACML, assim como das aplicações de testes e demonstração. São abordados os principais requisitos, a especificação e implementação.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A extensão XACML e as aplicações de testes deverão atender os seguintes requisitos:

a) ser desenvolvida utilizando a linguagem Java (Requisito Não-Funcional - RNF);
b) utilizar a IDE eclipse (RNF);
c) ter como base a versão 1.0 ou superior da especificação XACML (RNF);
d) deverá permitir gerenciar políticas de acesso no conceito de Federação de Identidades (Requisito Funcional - RF);
e) deverá permitir interpretar *request* e *response* do padrão XACML no conceito de federação de identidades (RF);
f) deverá permitir identificar a política a ser aplicada no conceito de federação de identidades (RF).

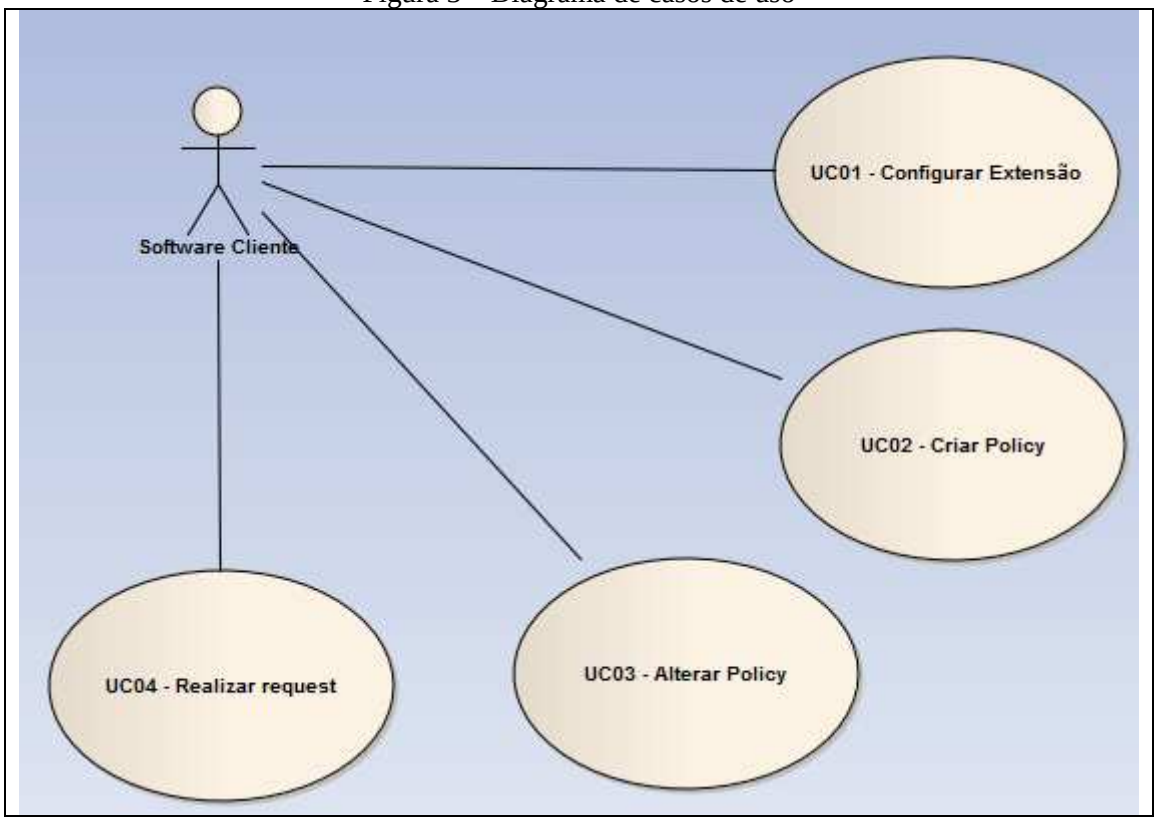
3.2 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando a ferramenta Enterprise Architect, utilizando os conceitos de *Schema* XML e diagramas da *Unified Modeling Language* (UML). Assim, nas próximas seções são apresentados os diagramas de caso de uso, diagramas de classes e *Schema* XML.

3.2.1 Casos de uso

Nesta seção são descritos os casos de uso de todas as funcionalidades da extensão XACML. Foi identificado um ator que utilizará a extensão XACML, o ator `Software Cliente`. Este ator irá utilizar a extensão XACML para aplicar o conceito de controle de acesso com federação de identidades. Na Figura 3 é apresentado o diagrama de casos de uso da aplicação.

Figura 3 – Diagrama de casos de uso



3.2.1.1 Configurar Extensão

Este caso de uso descreve como o `Software Cliente` deverá configurar a extensão XACML. Detalhes são apresentados no Quadro 2.

Quadro 2 – Caso de uso UC01

UC01 – Configurar extensão	
Descrição	Para utilizar a extensão XACML, é necessário informar para a extensão o arquivo de configuração na qual será utilizado para identificar as classes responsáveis em tratar os atributos, regras de combinações, funções de comparações, configurações do PDP e arquivos *schema* para validação das *Policys* e *Requests*.
Cenário Principal	1. O `Software Cliente` atribui a uma propriedade do sistema o caminho do arquivo de configuração. 2. O `Software Cliente` requisita para a extensão uma instância do PDP para carrega-lo. 3. O `Software Cliente` requisita para a extensão uma instância do PAP para carrega-lo.

3.2.1.2 Criar *Policy*

Este caso de uso descreve como o `Software Cliente` deverá utilizar a extensão XACML para criar as *Policys*. Detalhes são apresentados no Quadro 3.

Quadro 3 – Caso de uso UC02

UC02 - Criar *Policy*	
Descrição	Para criar uma *Policy* é necessário seguir uma sequência lógica de chamada de métodos do administrador de P*olicys* da extensão XACML. Essas chamadas de método irá criar os elementos que compõem uma *Policy.*
Pré-condição	É necessário que o `Software Cliente` já tenha extensão XACML com o arquivo de configuração carregado.
Cenário Principal	1. O `Software Cliente` deve sinalizar para o administrador de *Policy* da extensão XACML qual repositório ira utilizar. 2. O `Software Cliente` deve utilizar o método estático `addTargetPolicy` da classe `FacadeAccessControl` para criar os *Targets* de *Subjects*, *Actions*, *Resources* e *Federations* da nova *Policy*. 3. O `Software Cliente` deve utilizar o método estático `addTargetRule` da classe `FacadeAccessControl` para criar os *Targets* de *Subjects*, *Actions*, *Resources* e *Federations* que serão utilizados para cria uma *Rule* na nova *Policy*. 4. O `Software Cliente` deve utilizar o método estático `createRuleList` da classe `FacadeAccessControl` para criar uma nova Rule na nova *Policy*. Neste momento deve ser passado como parâmetro para o método o *Effect* da *Rule.* *5.* O `Software Cliente` deve utilizar o método estático createPolicy da classe `FacadeAccessControl` para criar a nova *Policy*. Neste momento deve ser passado como parâmetro o *id*, descrição e algoritmo de combinação da nova *Policy*. 6. Se ainda houver *Policys* para serem criadas, voltar ao passo 1 do cenário principal.

3.2.1.3 Alterar *Policy*

Este caso de uso descreve como o `Software Cliente` deverá utilizar a extensão XACML para alterar as *Policys*. Detalhes são apresentados no Quadro 4.

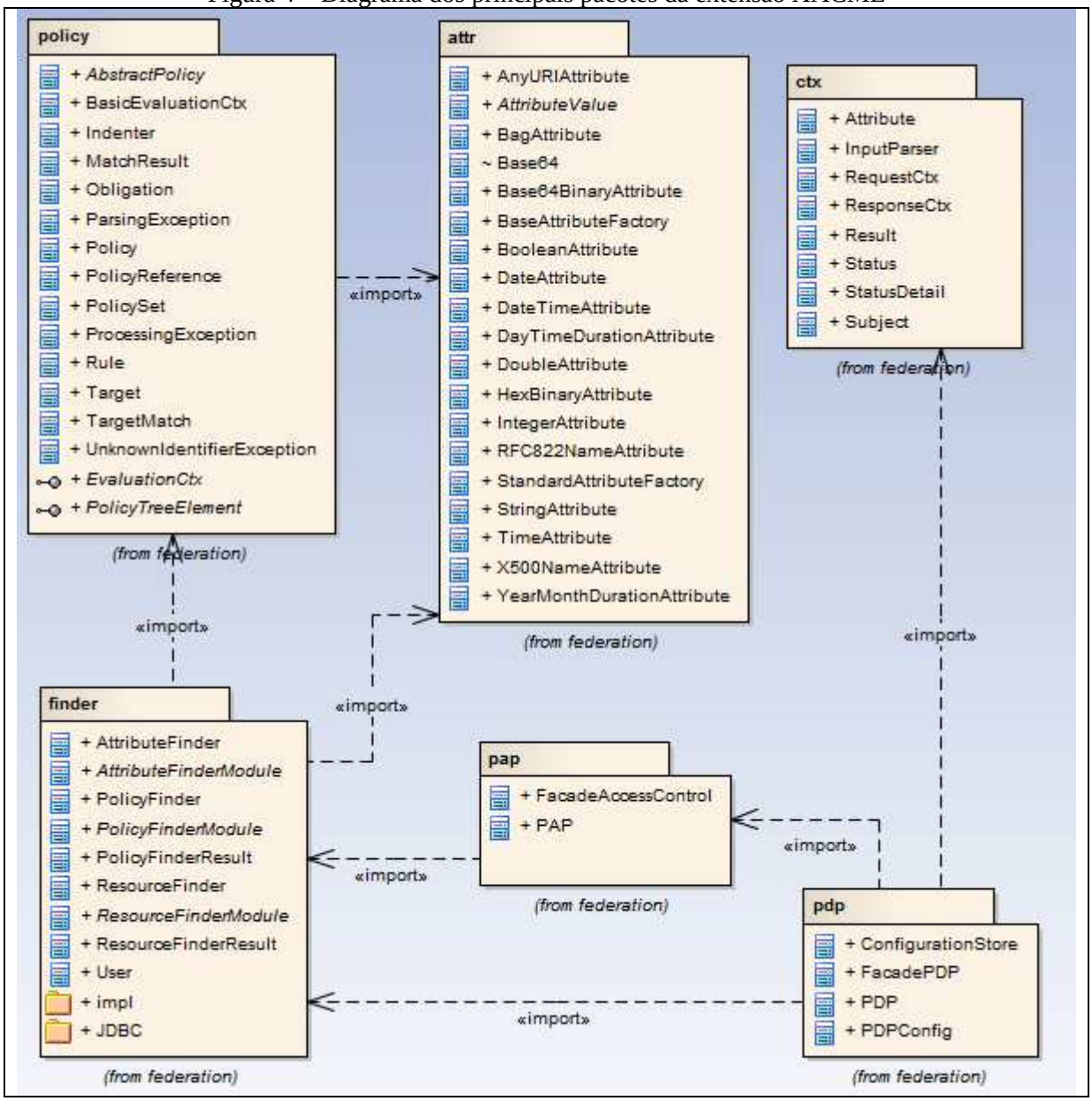
Quadro 4 – Caso de uso UC03

UC03 - Alterar *Policy*	
Descrição	Para alterar uma *Policy* é necessário seguir uma sequência lógica de chamada de métodos do administrador de P*olicys* da extensão XACML.
Pré-condição	É necessário que o `Software Cliente` já tenha a extensão XACML configurada.
Cenário Principal	1. O `Software Cliente` deve sinalizar para o administrador de *Policy* da extensão XACML qual repositório ira utilizar. 2. O `Software Cliente` deve utilizar o método `getPolicy` do administrador *Policy*s para guardar a referência da *Policy* a ser alterar. 3. O `Software Cliente` deve realizar as alterações na referência da *Policy*. 4. O `Software Cliente` deve salvar a *Policy* alterada chamando o método `savePolicyXML` do administrador de *Policy*s. 5. Se ainda houver *Policys* para serem alteradas, voltar ao passo 1 do cenário principal.

3.2.2 Diagramas de classes

Nesta seção são apresentadas as principais classes envolvidas na extensão XACML, bem como seus relacionamentos e estrutura. Na Figura 4 são demonstradas as dependências entre os principais pacotes definidos para a extensão XACML juntamente com as classes que os compõem.

Figura 4 – Diagrama dos principais pacotes da extensão XACML

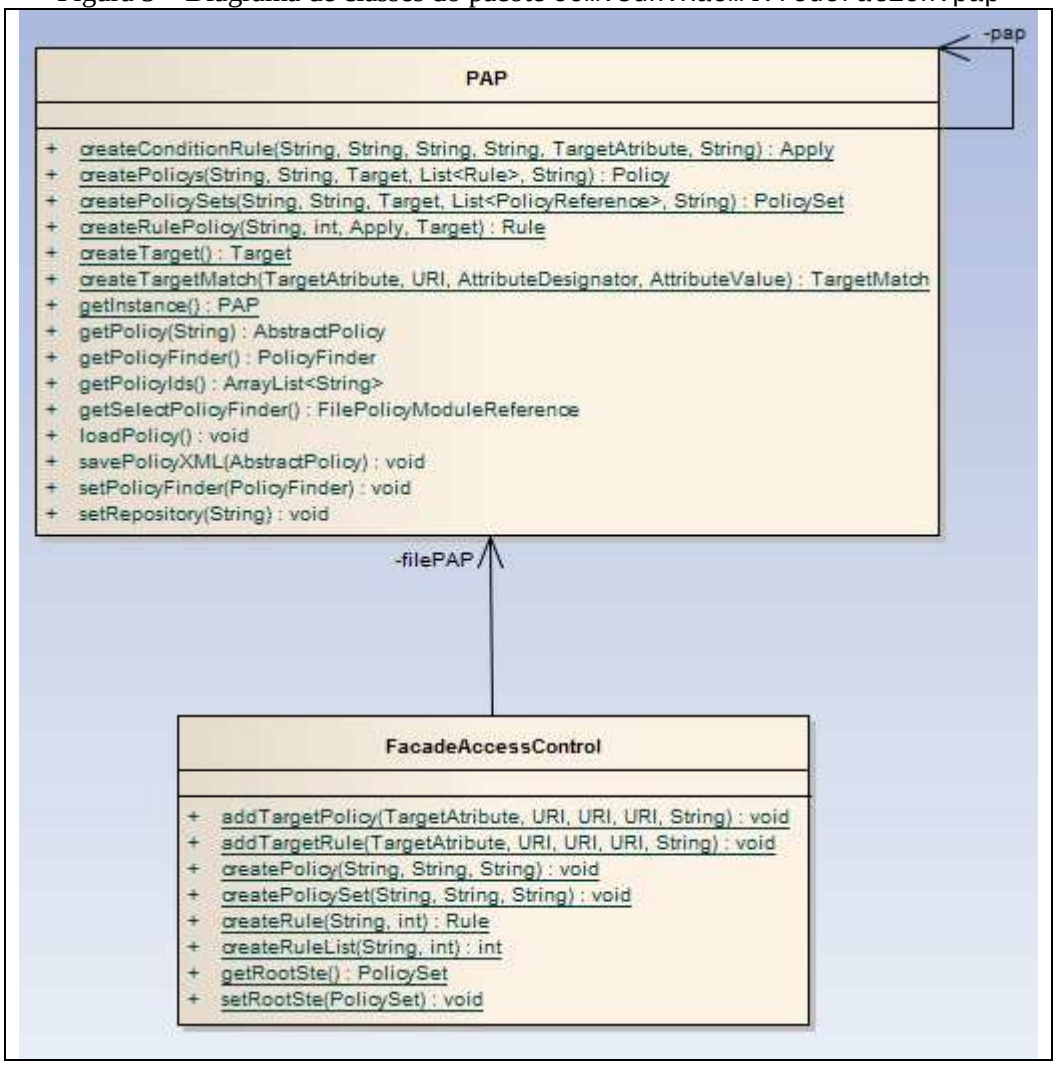


3.2.2.1 Pacote `com.sun.xacml.federation.pap`

As classes desse pacote são classes para administração das *Policys*. São responsáveis em criar, alterar e prover ao PDP as *Policys*. A Figura 5[1] representa o diagrama de classes desse pacote.

[1] Todos os atributos e os métodos privados foram removidos para uma melhor visualização.

Figura 5 – Diagrama de classes do pacote `com.sun.xacml.federation.pap`



A classe PAP é responsável em administrar as *Policys*, nela estão disponíveis os métodos para criar os elementos que compõem uma *Policy*. Também possui dois métodos para retornar uma *Policy* específica ou uma lista de todas as *Policys*. A classe PAP deve ser utilizada sempre quando se for criar ou alterar uma *Policy*.

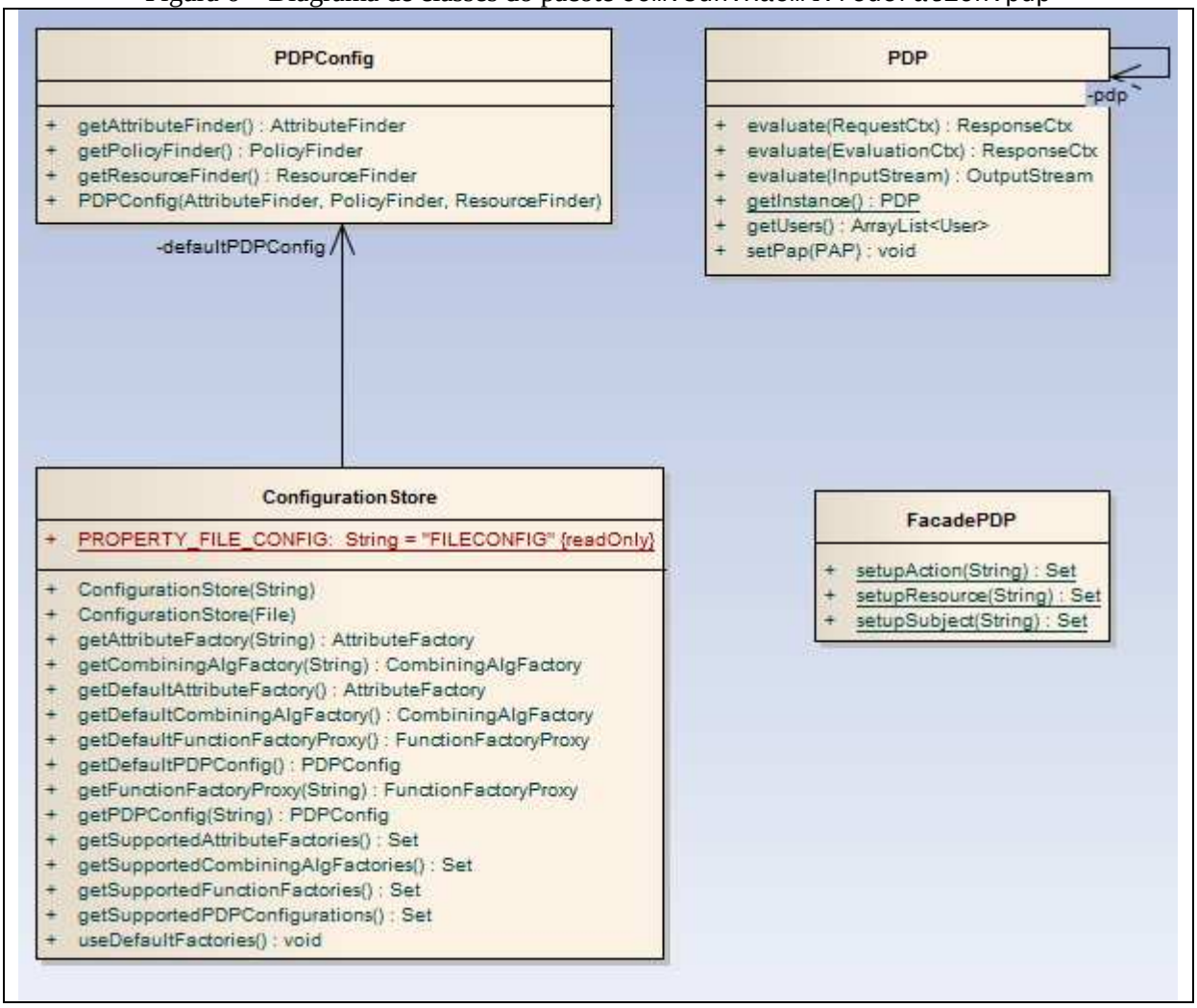
O pacote `com.sun.xacml.federation.pap` também contém a classe `FacadeAcesssControl` que é um façade com alguns métodos que auxiliam a criação de uma nova *Policy*.

3.2.2.2 Pacote `com.sun.xacml.federation.pdp`

É neste pacote onde está a implementação do PDP que centraliza e gerencia os *requests*. A Figura 6[2] representa o diagrama de classes desse pacote.

[2] Todos os atributos e os métodos privados foram removidos para uma melhor visualização.

Figura 6 – Diagrama de classes do pacote `com.sun.xacml.federation.pdp`



A classe `ConfigurationStore` é responsável por carregar as configurações do arquivo de configurações para o PDP, nela também estão definidas as configurações *default* caso não seja informado um arquivo de configurações. Ao carregar as configurações do PDP, a classe `ConfigurationStore` deve receber um arquivo de configurações conforme a especificação do *Schema* XML do arquivo de configurações.

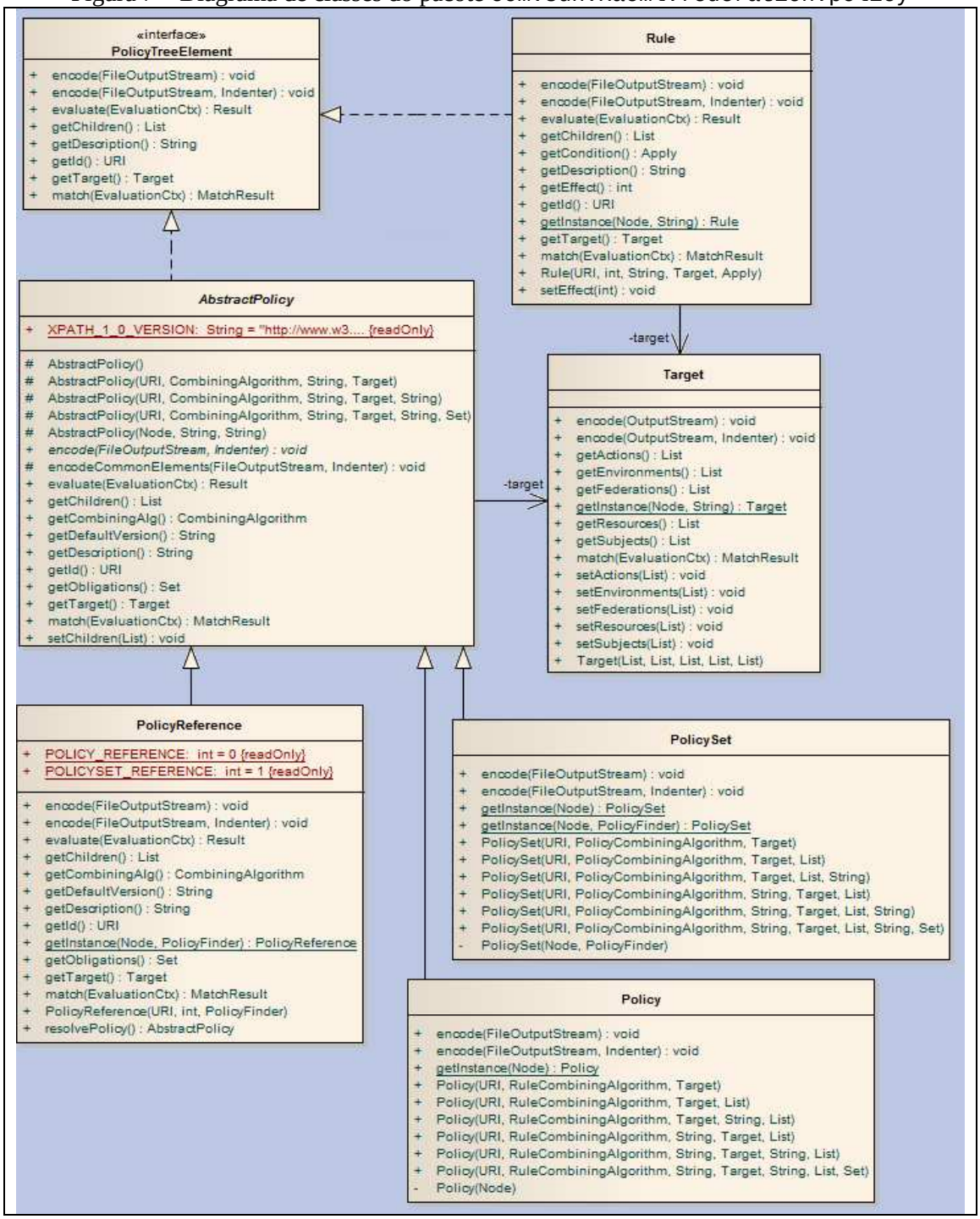
A classe `PDP` deve ser utiliza sempre que for necessário realizar um *Request* para determinar a consulta de uma *Policy*. Um método importante da classe `PDP` é o `getUsers` o qual é responsável em retornar os usuários de todas as federações.

O pacote também contém o façade `FacadaPDP` que facilita a criação de *Requests* e deve ser utilizado sempre que for necessário criar um novo *Request*. O façade encapsula a necessidade de criar classes `URI` para a criação dos *Request*.

3.2.2.3 Pacote `com.sun.xacml.federation.policy`

As classes desse pacote representam os elementos que compõem uma *Policy* e os relacionamentos e estrutura do repositório de *Policys*. A Figura 7[3] representa o diagrama de classes desse pacote.

Figura 7 – Diagrama de classes do pacote `com.sun.xacml.federation.policy`



[3] Todos os atributos e os métodos privados foram removidos para uma melhor visualização.

A classe `AbstractPolicy` é a principal do pacote, ela deve ser utilizada sempre que for necessário representar uma *Policy* ou uma *Policyset*. A classe `AbstractPolicy` é uma classe abstrata que já tem estruturado os elementos *Id*, descrição, regra de combinação, *Targets*, *Rules* que são comuns entre *Policy* ou uma *Policyset*.

Nas classes `Policy` e `PolicySet` estão implementados os métodos e elementos que são diferentes entre si, um deles é o método `encode` que é responsável em gravar a *Policy* ou *Policyset* em arquivo XML no repositório. Essas classes também são responsáveis por interpretar um *node* XML com as definições de uma *Policy* ou *Policyset*.

A classe `Target` encapsula *Subjects, Resources, Actions e Federations* que serão controlados pela *Policy*. Esta classe deve ser utiliza sempre que for necessário alterar o *Target* de uma *Rule, Policy* ou *Policyset.*

A classe `Rule` é responsável em representar as regras de uma *Policy*, internamente ela contem um `Target` para determinar se o *effect* desta *Rule* é aplicado ao *Request*. Esta classe deve ser utiliza sempre que for necessário alterar uma *Rule* de uma *Policy* ou *Policyset*.

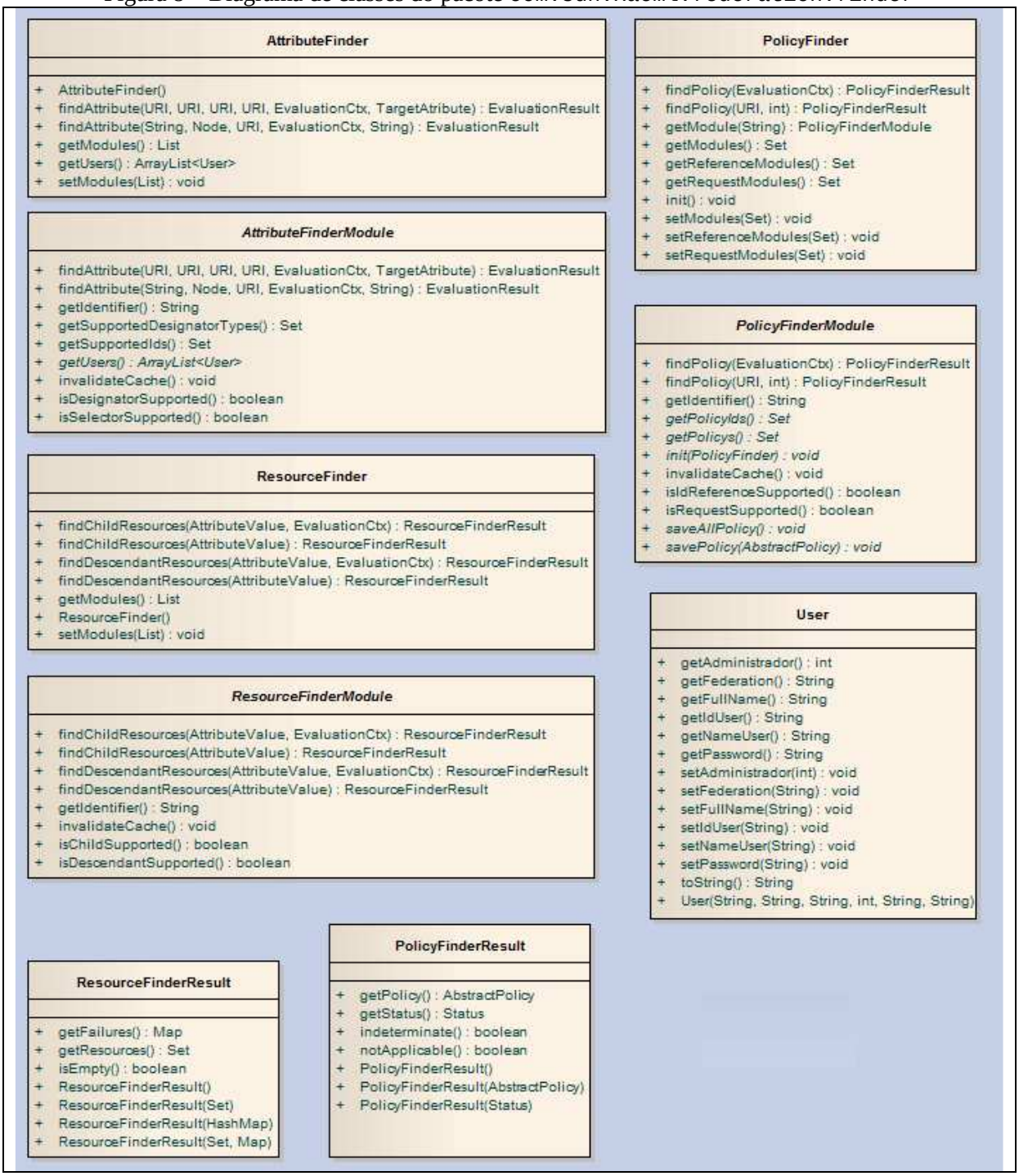
A classe `PolicyReference` é utilizada para controle das *Policy*s dentro da *Policyset*, essa classe deve ser utilizada sempre que for necessário obter a referência de uma *Policy* que está em uma *Policyset*, isso é possível pelo método `resolvePolicy`.

3.2.2.4 Pacote `com.sun.xacml.federation.finder`

As classes desse pacote têm a funcionalidade de encontrar os recursos, atributos, usuários e políticas que a extensão XACML administrará. A Figura 8[4] representa o diagrama de classes desse pacote.

[4] Todos os atributos e os métodos privados foram removidos para uma melhor visualização.

Figura 8 – Diagrama de classes do pacote `com.sun.xacml.federation.finder`



As classes `AttributeFinder` e `AttributeFinderModule` são responsáveis em encontrar e disponibilizar ao PDP os atributos não contidos nos *Requests*.

A classe `AttributeFinderModule` é uma classe base que dever ser estendida pelo `Software Cliente` ao criar os PIPs no PDP, o método `finderAttribute` é o método mais importante da classe, é ele que será chamado para encontrar o valor de um atributo. Esta classe representará a federação de identidades, é ela que retornará as lista de usuários através do método `getUsers` utilizando a classe `User` para representar um usuário e seus atributos.

No PDP a classe `AttributeFinder` é responsável por agrupar todos os `AttributeFinderModule` que representam os PIPs. O método `finderAttribute` busca em todos `AttributeFinderModule` o valor do atributo procurado.

As classes `PolicyFinderModule` e `PolicyFinder` são responsáveis em encontrar e disponibilizar ao PAP as *policys*. A classe `PolicyFinderModule` é a responsável em armazenar as *Policys* e salvá-las utilizando o método `savePolicy`. Os métodos `getPolicyIds`, `getPolicys` e `findPolicy` são utilizados pelo PAP para procurar e interagir com as *Policys*.

A classe `PolicyFinder` tem o papel de agrupar os `PolicyFinderModule` e dessa forma é utilizado no PDP para armazenar vários `PolicyFinderModule`. Também disponibiliza o método `findPolicy` que se encarrega de chamar os `findPolicy` de todos os `PolicyFinderModule` retornando um objeto da classe `PolicyFinderResult` que representa o resultado da busca.

As classes `ResourceFinder` e `ResourceFinderModule` são responsáveis em retornar ao PDP uma lista de *resources* administrados. Essas classes são opcionais na implementação de uma aplicação, elas deverão ser utilizadas caso a aplicação queira garantir que *Policys* sejam criadas somente para *resources* existentes.

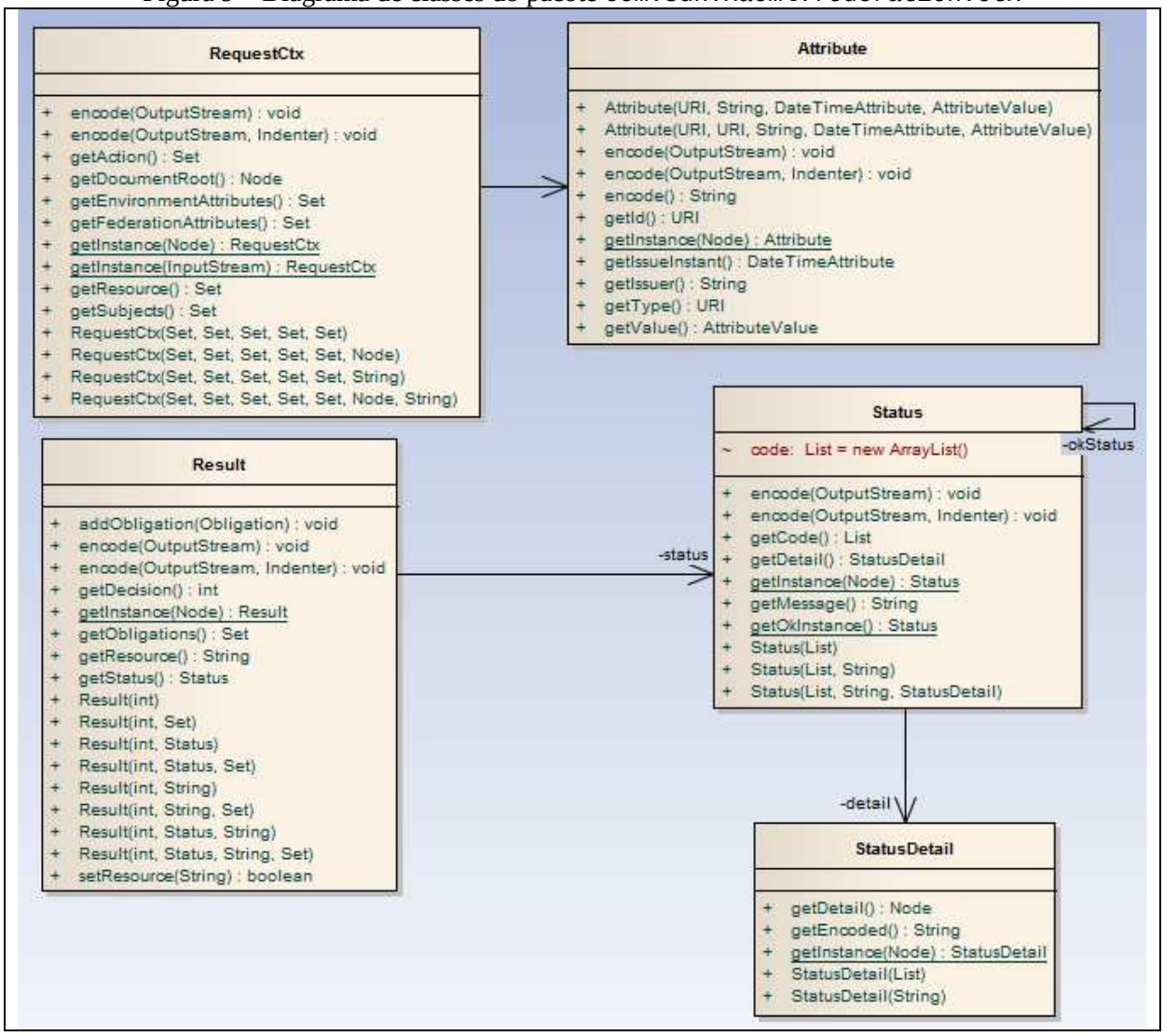
A classe `ResourceFinderModule` é a responsável em retornar e pesquisar os *resources*. A classe `ResourceFinder` tem o papel de agrupar os `ResourceFinderModule` e dessa forma é utilizado no PDP para armazenar vários `ResourceFinderModule`. Na classe é disponibilizando o método `findChildResource` que se encarrega de chamar os `findChildResource` de todos os `ResourceFinderModule` retornando um objeto da classe `ResourceFinderResult` que representa o resultado da busca.

3.2.2.5 Pacote `com.sun.xacml.federation.ctx`

As classes desse pacote têm a funcionalidade de representar os *Requests* e *Result* e seus respectivos *status*. A Figura 9[5] representa o diagrama de classes desse pacote.

[5] Todos os atributos e os métodos privados foram removidos para uma melhor visualização.

Figura 9 – Diagrama de classes do pacote `com.sun.xacml.federation.ctx`



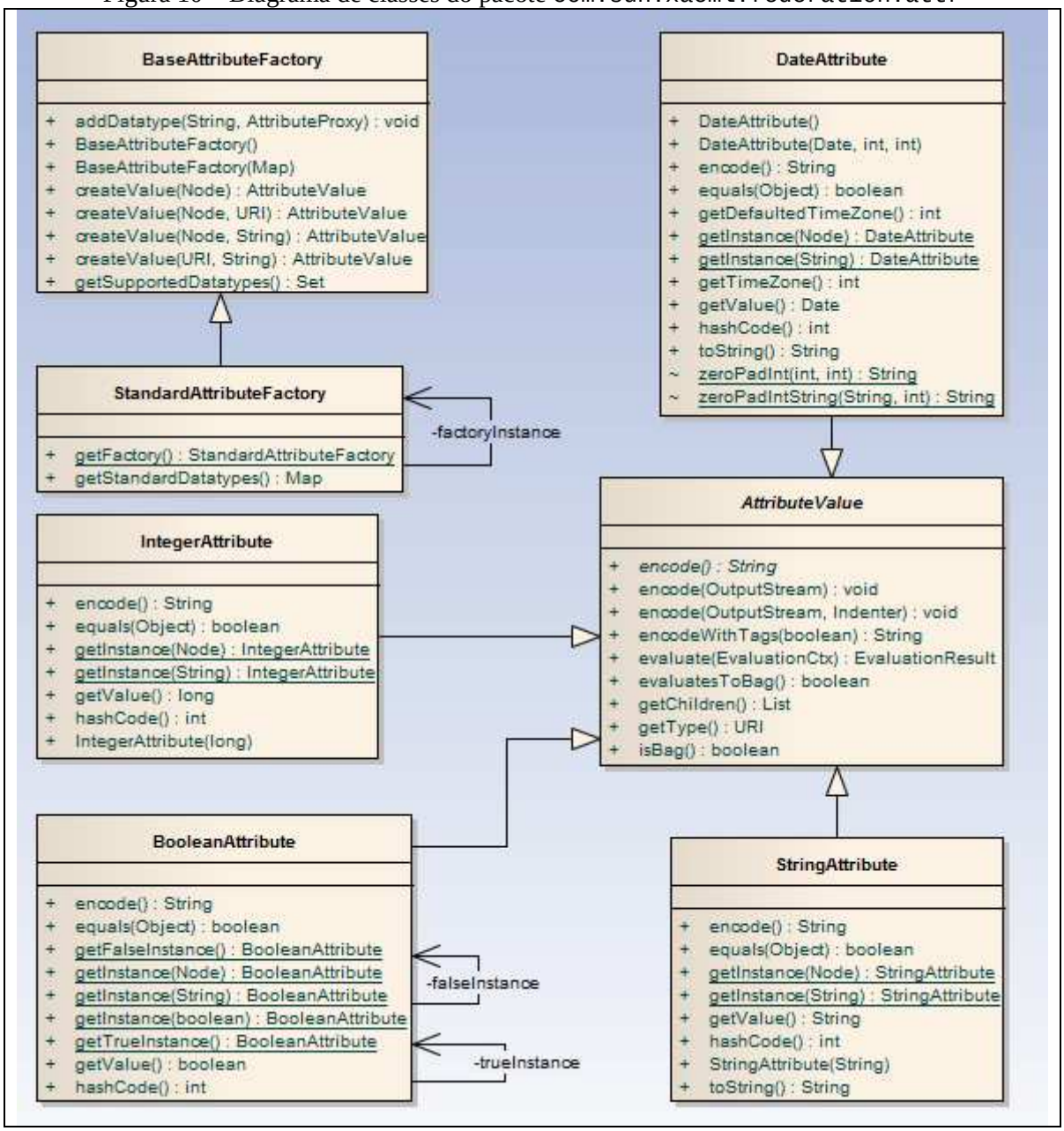
A classe `RequestCtx` representa o *Request*, ela é responsável em agrupar os atributos do *Request*, que são *Subjects*, *Actions*, *Federations* e *Resources*, que são representados pela classe `Attribute` e deveram ser passados em uma `List` como parâmetro no construtor do objeto.

A classe *Result* representa o resultado de um *Request*. O método `getDecision` deve ser utilizado para verificar o resultado do *Request* e caso for necessário verificar mais detalhadamente o resultado, o método `getStatus` retorna um objeto da classe `Status` onde é possível obter uma mensagem. A classe `Status` pode conter também vários `StatusDetail` que possibilita detalhar ainda mais o resultado do *Request*.

3.2.2.6 Pacote `com.sun.xacml.federation.attr`

As classes desse pacote têm a funcionalidade de representar os tipos de *Attribute Value* dentro da extensão XACML. A Figura 10[6] representa o diagrama de classes desse pacote.

Figura 10 – Diagrama de classes do pacote `com.sun.xacml.federation.attr`



3.2.3 Especificação *Schema* XML da extensão XACML

Nesta seção são apresentados os principais elementos da especificação *Schema* XML da extensão XACML.

[6] Todos os atributos e os métodos privados foram removidos para uma melhor visualização.

3.2.3.1 Especificação *Schema* XML da *Policy* e *PolicySet*

A seguir são apresentados os principais elementos da especificação *Schema* XML da *Policy* que pode ser encontrado por completo no apêndice A.

Na Figura 11 está especificada a estrutura de uma *Policy* e seus elementos, esta especificação deve ser seguida sempre que se for salvar ou ler uma *Policy* em XML.

Figura 11 – Especificação *Schema* XML da estrutura de uma *Policy*

```
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
    <xs:sequence>
        <xs:element ref="xacml:Description" minOccurs="0"/>
        <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
        <xs:element ref="xacml:Target"/>
        <xs:element ref="xacml:Rule" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="xacml:Obligations" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
    <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

Toda *Policy* da extensão XACML representada em XML deve apresentar seus elementos conforme sequencia definida na especificação, `Description`, `PolicyDefaults`, `Target`, `Rule`, `Obligations`.

Na Figura 12 está especificada a estrutura de uma *PolicySet* e seus elementos, está especificação deve ser seguida sempre que for salvar ou ler uma *PolicySet* em XML.

Figura 12 – Especificação *Schema* XML da estrutura de uma *PolicySet*

```
<xs:element name="PolicySet" type="xacml:PolicySetType"/>
<xs:complexType name="PolicySetType">
    <xs:sequence>
        <xs:element ref="xacml:Description" minOccurs="0"/>
        <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
        <xs:element ref="xacml:Target"/>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xacml:PolicySet"/>
            <xs:element ref="xacml:Policy"/>
            <xs:element ref="xacml:PolicySetIdReference"/>
            <xs:element ref="xacml:PolicyIdReference"/>
        </xs:choice>
        <xs:element ref="xacml:Obligations" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
    <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

Toda *PolicySet* da extensão XACML representada em XML deve apresentar seus elementos conforme sequencia definida na especificação, `Description`, `PolicyDefaults`,

`Target`, `Lista`, `Rule`, `Obligations`. Conforme especificação, uma *PolicySet* conte uma lista de *Policys* ou *PolicySets*, podendo ou não ser uma referencia por *Id*.

Na Figura 13 está especificada a estrutura de um *Target* e seus elementos.

Figura 13 – Especificação *Schema* XML da estrutura de um *Target*

```
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
    <xs:sequence>
        <xs:element ref="xacml:Subjects"/>
        <xs:element ref="xacml:Resources"/>
        <xs:element ref="xacml:Actions"/>
        <xs:element ref="xacml:Federations"/>
    </xs:sequence>
</xs:complexType>
```

Todo *Target* de uma *Policy* ou *Rule* da extensão XACML representado em XML deve apresentar seus elementos conforme sequencia definida na especificação, `Subjects`, `Resources`, `Actions` e `Federations`. A especificação do padrão XACML foi alterada para incluir no `Target` o elemento `Federations`.

3.2.3.2 Especificação *Schema* XML do *Request*

A seguir são apresentados os principais elementos da especificação *Schema* XML do *Request* que pode ser encontrado por completo no apêndice B.

Na Figura 14 está especificada a estrutura de um *Request* e seus elementos.

Figura 14 – Especificação *Schema* XML da estrutura de um *Request*

```
<xs:element name="Request" type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
    <xs:sequence>
        <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
        <xs:element ref="xacml-context:Resource"/>
        <xs:element ref="xacml-context:Action"/>
        <xs:element ref="xacml-context:Federation" />
    </xs:sequence>
</xs:complexType>
```

Todo *Request* da extensão XACML representado em XML deve apresentar seus elementos conforme sequencia definida na especificação, `Subject`, `Resource`, `Action` e `Federation`. A especificação do padrão XACML foi alterada para incluir no `RequestType` o elemento `Federation`.

3.2.3.3 Especificação *Schema* XML do arquivo de configuração do PDP

A seguir são apresentados os principais elementos da especificação *Schema* XML do arquivo de configuração do PDP que pode ser encontrado por completo no apêndice C.

Na Figura 15 está especificada a estrutura do arquivo de configuração e seus elementos do arquivo de configuração do PDP.

Figura 15 – Especificação *Schema* XML da estrutura do arquivo de configuração do PDP

```
<xs:element name="config">
    <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="confgfiles" type="config:ConfgFilesType"/>
            <xs:element name="pdp" type="config:PDPType"/>
            <xs:element name="attributeFactory" type="config:AttributeFactoryType"/>
            <xs:element name="combiningAlgFactory" type="config:CombiningFactoryType"/>
            <xs:element name="functionFactory" type="config:FunctionFactoryType"/>
        </xs:choice>
    </xs:complexType>
    <xs:attribute name="defaultPDP" type="xs:string" use="required"/>
    <xs:attribute name="defaultAttributeFactory" type="xs:string"
                  use="required"/>
    <xs:attribute name="defaultCombiningAlgFactory" type="xs:string"
                  use="required"/>
    <xs:attribute name="defaultFunctionFactory" type="xs:string"
                  use="required"/>
    <xs:attribute name="defaultFiles" type="xs:string" use="required"/>
</xs:element>
```

Os elementos `attributeFactory`, `combiningAlgFactory` e `functionFactory` são os elementos customizados e responsáveis na extensão XACML em prover as classes de atributos, algoritmos de combinação e funções de comparação.

A especificação *Schema* XML do arquivo de configuração do PDP foi alterada para criar os elementos `Confgfiles` e `PDPType`. O elemento `Confgfiles` é utilizado para informar o caminho dos arquivos XML *Schema Definition* (XSD) da especificação *Schema* XML da extensão XACML. Na Figura 16 é apresentada a especificação *Schema* XML do *type* `ConfgFilesType` que é utilizado pelo elemento `Confgfiles`.

Figura 16 – Especificação *Schema* XML do elemento `ConfgFilesType`

```
<xs:complexType name="ConfgFilesType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="policySchema" type="config:ClassType"/>
        <xs:element name="contextSchema" type="config:ClassType"/>
    </xs:choice>
    <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

O elemento `pdp` é utilizado para carregar ao PDP as classes responsáveis em encontrar os atributos de usuário, os recursos controlados e os repositórios de políticas. Na Figura 17 é

apresentada a especificação *Schema* XML do *type* `PDPType` que é utilizado pelo elemento `pdp`.

Figura 17 – Especificação *Schema* XML do elemento `PDPType`

```
<xs:complexType name="PDPType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="attributeFinderModule" type="config:ClassType"/>
        <xs:element name="resourceFinderModule" type="config:ClassType"/>
        <xs:element name="policyFinderModule" type="config:ClassType"/>
    </xs:choice>
    <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas informações sobre técnicas e ferramentas utilizadas para a implementação, bem como a própria implementação e cada etapa de seu desenvolvimento.

Ao final, será descrita a operacionalidade do protótipo desenvolvido.

3.3.1 Técnicas e ferramentas utilizadas

O desenvolvimento da extensão XACML, bem como a aplicação de testes que demonstra o uso da extensão, foi feito na linguagem de programação Java, seguindo o paradigma da orientação a objetos.

Para a codificação foram utilizados dois ambientes de desenvolvimento Eclipse e NetBeans. O Eclipse (versão Juno *Service Release* 2) foi utilizado para a implementação de toda a extensão XACML e a regra de negocio das aplicações de testes. Já o NetBeans (versão 7.3) foi utilizado na implementação das telas das aplicações de testes, por ser mais ágil nesta área.

3.3.2 Implementação extensão XACML

Para o desenvolvimento da extensão XACML foi utilizado a API Sun's XACML por ser mais popular. A API Sun's XACML proporciona as implementações básicas dos elementos do XACML, como *Policys*, *PolicySets*, *Targets*, *Rules*, *Attributes* e um arquivo de configuração básico do PDP (SUN, 2004c).

Os pacotes da API Sun's XACML foram reestruturados e criados três novos pacote que são eles `pap`, `pdp`, `policy`, para facilitar o entendimento da extensão.

No pacote `com.sum.xacml.federation` foram desenvolvidas as classes de configuração e administração do PDP e sua principal classe é a classe `PDP`. A classe `PDP` foi

desenvolvida para ser um *singleton*, portanto não pode se instanciada, para utiliza-la deve ser clamado o seu método estático `getInstance` que pode ser visto no Quadro 5.

Quadro 5 – Método `getInstance` da classe `PDP`

```
01  public static PDP getInstance() {
02          if (pdp == null) {
03                  try {
04                          ConfigurationStore store = new ConfigurationStore(
05                          System.getProperty(ConfigurationStore.PROPERTY_FILE_CONFIG));
06
07                          store.useDefaultFactories();
08                          pdp = new PDP(store.getDefaultPDPConfig());
09                  } catch (ParsingException e) {
10                          e.printStackTrace();
11                  } catch (UnknownIdentifierException e) {
12                          e.printStackTrace();
13                  }
14          }
15          return pdp
16  }
```

No método `getInstance` é utilizada a classe `ConfigurationStore` que é responsável em ler o arquivo de configuração do PDP e passa-las como parâmetro no construtor do novo `PDP`.

No Quadro 6 é apresentado o método `setupConfig` da classe `ConfigurationStore`, onde é possível visualizar a sequencia lógica de leitura dos *node* do arquivo de configuração do PDP. O método `setupConfig` também é responsável em chamar os métodos que iram instanciar as classes que compõem o PDP.

Quadro 6 – Método `setupConfig` da classe `ConfigurationStore`

```
01  private void setupConfig(File configFile) throws ParsingException {
02          Node root = getRootNode(configFile);
03          pdpConfigMap = new HashMap();
04          attributeMap = new HashMap();
05          combiningMap = new HashMap();
06          functionMap = new HashMap();
07
08          NamedNodeMap attrs = root.getAttributes();
09          String defaultPDP = attrs.getNamedItem("defaultPDP").getNodeValue();
10          String defaultAF = attrs.getNamedItem("defaultAttributeFactory")
11                          .getNodeValue();
12          String defaultCAF = attrs.getNamedItem("defaultCombiningAlgFactory")
13                          .getNodeValue();
14          String defaultFF = attrs.getNamedItem("defaultFunctionFactory")
15                          .getNodeValue();
16
17          NodeList children = root.getChildNodes();
18          for (int i = 0; i < children.getLength(); i++) {
19                  Node child = children.item(i);
20                  String childName = child.getNodeName();
21                  String elementName = null;
22
23                  if (child.getNodeType() == Node.ELEMENT_NODE)
24                          elementName = child.getAttributes().getNamedItem("name")
25                                          .getNodeValue();
26
27                  if (childName.equals("pdp")) {
28                          if (logger.isLoggable(Level.CONFIG))
29                                  logger.config("Loading PDP: " + elementName);
30                          if (pdpConfigMap.containsKey(elementName))
31                                  throw new ParsingException("more that one pdp with "
32                                                  + "name \"" + elementName + "\"");
33                          pdpConfigMap.put(elementName, parsePDPConfig(child));
```

Quadro 6 – Continuação método `setupConfig` da classe `ConfigurationStore`

```
38                  }
39                  if (childName.equals("confgfiles")) {
40                          if (logger.isLoggable(Level.CONFIG))
41                                  logger.config("Loading confgfiles: " + elementName);
42                          if (pdpConfigMap.containsKey(elementName))
43                                  throw new ParsingException("more that one pdp with "
44                                          + "name \"" + elementName + "\"");
45                          parseFilesConfig(child);
46                  } else if (childName.equals("attributeFactory")) {
47                          if (logger.isLoggable(Level.CONFIG))
48                                  logger.config("Loading AttributeFactory: " + elementName);
49                          if (attributeMap.containsKey(elementName))
50                                  throw new ParsingException("more that one "
51                                                  + "attributeFactory with name " + elementName
52                                                  + "\"");
53                          attributeMap.put(elementName, parseAttributeFactory(child));
54                  } else if (childName.equals("combiningAlgFactory")) {
55                          if (logger.isLoggable(Level.CONFIG))
56                                  logger.config("Loading CombiningAlgFactory: " + elementName);
57                          if (combiningMap.containsKey(elementName))
58                                  throw new ParsingException("more that one "
59                                                  + "combiningAlgFactory with " + "name \""
60                                                  + elementName + "\"");
61                          combiningMap.put(elementName, parseCombiningAlgFactory(child));
62                  } else if (childName.equals("functionFactory")) {
63                          if (logger.isLoggable(Level.CONFIG))
64                                  logger.config("Loading FunctionFactory: " + elementName);
65                          if (functionMap.containsKey(elementName))
66                                  throw new ParsingException("more that one functionFactory"
67                                                  + " with name \"" + elementName + "\"");
68                          functionMap.put(elementName, parseFunctionFactory(child));
69                  }
70          }
71
72          defaultPDPConfig = (PDPConfig) (pdpConfigMap.get(defaultPDP));
73          defaultAttributeFactory = (AttributeFactory) (attributeMap
74                          .get(defaultAF));
75          defaultCombiningFactory = (CombiningAlgFactory) (combiningMap
76                          .get(defaultCAF));
77          defaultFunctionFactoryProxy = (FunctionFactoryProxy) (functionMap
78                          .get(defaultFF));
79  }
```

Da mesma forma que a classe `PDP` a classe `pap` é um *singleton* e é carregada a partir do arquivo de configuração como é demonstrado no Quadro 7.

Quadro 7 – Método `getInstance` da classe `PAP`

```
01  public static PAP getInstance() {
02          if (pap == null) {
03                  try {
04                          ConfigurationStore store;
05                          store = new ConfigurationStore(
06          System.getProperty(ConfigurationStore.PROPERTY_FILE_CONFIG));
07
08                          store.useDefaultFactories();
09
10                          pap = new PAP(store.getDefaultPDPConfig().getPolicyFinder());
11
12                  } catch (ParsingException e) {
13                          // TODO Auto-generated catch block
14                          e.printStackTrace();
15                  } catch (UnknownIdentifierException e) {
16                          // TODO Auto-generated catch block
17                          e.printStackTrace();
18                  }
19          }
20          return pap;
21  }
```

No Quadro 8 é demonstrado o trecho de código onde o PDP recebe um *Request* para ser avaliado.

Quadro 8 – Método `evaluateContext` da classe `PDP`

```
01  public Result evaluateContext(EvaluationCtx context) {
02          PolicyFinderResult finderResult = pap.getPolicyFinder().findPolicy(
03                          context);
04          if (finderResult.notApplicable())
05                  return new Result(Result.DECISION_NOT_APPLICABLE, context
06                                  .getResourceId().encode());
07          if (finderResult.indeterminate())
08                  return new Result(Result.DECISION_INDETERMINATE,
09                                  finderResult.getStatus(), context.getResourceId().encode());
10          return finderResult.getPolicy().evaluate(context);
11  }
```

O PDP quando recebe um *Request*, solicita ao PAP que verifique entre as políticas, quais se aplicam. O PAP percorre todas as políticas e chama o método `match` dos *Target* que realiza a avaliação. No Quadro 9 é demonstrado o trecho de código do método `match`.

Quadro 9 – Método `match` da classe `Target`

```
01  public MatchResult match(EvaluationCtx context) {
02                  if (!subjects.isEmpty()) {
03                  MatchResult result = checkSet(subjects, context);
04                  if (result.getResult() != MatchResult.MATCH) {
05                          logger.finer("failed to match Subjects section of Target");
06                          return result;
07                  }
08          }
09          if (!resources.isEmpty()) {
10                  MatchResult result = checkSet(resources, context);
11                  if (result.getResult() != MatchResult.MATCH) {
12                          logger.finer("failed to match Resources section of Target");
13                          return result;
14                  }
15          }
16          if (!actions.isEmpty()) {
17                  MatchResult result = checkSet(actions, context);
18                  if (result.getResult() != MatchResult.MATCH) {
19                          logger.finer("failed to match Actions section of Target");
20                          return result;
21                  }
22          }
23          if (!federations.isEmpty()) {
24                  MatchResult result = checkSet(federations, context);
25                  if (result.getResult() != MatchResult.MATCH) {
26                          logger.finer("failed to match Federations section of Target");
27                          return result;
28                  }
29          }
30          if (!environments.isEmpty()) {
31                  MatchResult result = checkSet(environments, context);
32                  if (result.getResult() != MatchResult.MATCH) {
33                          logger.finer("failed to match Environments section of Target");
34                          return result;
35                  }
36          }
37          return new MatchResult(MatchResult.MATCH);
38  }
```

É neste momento, ao fazer a avaliação de um *Target*, caso não seja encontrado o valor de um atributo no *Request* é chamado o método `getAttribute` que percorrer todos os PIP e retornar o valor do atributo.

No Quadro 10 é demonstrado o trecho de código do método `getAttribute` da classe `BasicEvaluationCtx`.

Quadro 10 – Método `getAttribute` da classe `BasicEvaluationCtx`

```
01  public EvaluationResult getAttribute(String contextPath,
02                  Node namespaceNode, URI type, String xpathVersion) {
03          if (finder != null) {
04                  return finder.findAttribute(contextPath, namespaceNode, type, this,
05                                  xpathVersion);
06          } else {
07                  logger.warning("Context tried to invoke AttributeFinder but was "
08                                  + "not configured with one");
09                  return new EvaluationResult(BagAttribute.createEmptyBag(type));
10          }
11  }
```

Para a implementação da federação de identidades foi alterada a classe `Target` da API base para implementar uma nova *tag* que representa a federações. Um dos métodos alterados da classe `Target` foi o `getInstance`. No Quadro 11 é demonstrado o método `getInstance` que interpreta o XML de um *Target* com a *tag* de federação. Todas as classes base da API foram alteradas para considerar a nova *tag* que representa a federação de identidades.

Quadro 11 – Método `getInstance` da classe `Target`

```
01  public static Target getInstance(Node root, String xpathVersion)
02                          throws ParsingException {
03          List subjects = null;
04          List resources = null;
05          List actions = null;
06          List federations = null;
07          List environments = null;
08          NodeList children = root.getChildNodes();
09          for (int i = 0; i < children.getLength(); i++) {
10                  Node child = children.item(i);
11                  String name = child.getNodeName();
12                  if (name.equals("Subjects")) {
13                          subjects = getAttributes(child, "Subject", xpathVersion);
14                  } else if (name.equals("Resources")) {
15                          resources = getAttributes(child, "Resource", xpathVersion);
16                  } else if (name.equals("Actions")) {
17                          actions = getAttributes(child, "Action", xpathVersion);
18                  } else if (name.equals("Federations")) {
19                          federations = getAttributes(child, "Federation", xpathVersion);
20                  } else if (name.equals("Environments")) {
21                          federations = getAttributes(child, "Environment", xpathVersion);
22                  }
23          }
24          return new Target(subjects, resources, actions, federations,
25                  environments);
26  }
```

A classe `PAP` é a classe responsável por realizar à administração das *Policys*. No Quadro 12 é apresentado o trecho de código do método `loadPolicy` responsável em iniciar os `PolicyFinderModule` que carregaram as *Policys* e o método `savePolicyXML` responsável em salvar uma *policy*.

Quadro 12 – Métodos `loadPolicy` e `savePolicyXML` da classe `PolicyFinderModule`

```
01  public void loadPolicy() {
02          Iterator it = policyFinder.getModules().iterator();
03          while (it.hasNext()) {
04                  PolicyFinderModule module = (PolicyFinderModule) (it.next());
05                  module.init(policyFinder);
```

Quadro 12 – Continuação métodos `loadPolicy` e `savePolicyXML` da classe `PolicyFinderModule`

```
06          }
07  }
08  public void savePolicyXML(AbstractPolicy policy) {
09                  selectPolicyFinder.savePolicy(policy);
10  }
```

No Quadro 13 é apresentado o trecho de código responsável em criar uma *Policy*. Conforme parâmetros passados, o método instância os elementos que compõem uma *Policy* e em seguida cria uma *Policy* e passa seus elementos como parâmetro no construtor.

Quadro 13 – Método `createPolicys` da classe `PAP`

```
01  public static Policy createPolicys(String aPolicyId,
02                  String aPolicyDescription, Target aPolicyTarget,
03                  List<Rule> aPolicyRules, String aAlgCombId)
04                  throws URISyntaxException, UnknownIdentifierException,
05                  FileNotFoundException, FunctionTypeException {
06
07          URI policyId = new URI(aPolicyId);
08
09          URI combiningAlgId = new URI(aAlgCombId);
10          CombiningAlgFactory factory = CombiningAlgFactory.getInstance();
11
12          RuleCombiningAlgorithm combiningAlg = (RuleCombiningAlgorithm) (factory
13                          .createAlgorithm(combiningAlgId));
14
15          String description = aPolicyDescription;
16          Policy policy = new Policy(policyId, combiningAlg, description,
17                          aPolicyTarget, aPolicyRules);
18
19          return policy;
20
21  }
```

A classe pap utiliza a classe `FilePolicyModule` para armazenar e controlar as *Policys*. No Quadro 14 é apresentado trecho de código do método `init` da classe `FilePolicyModule`.

Quadro 14 – Método `init` da classe `FilePolicyModule`

```
01  public void init(PolicyFinder finder) {
02          this.finder = finder;
03          if (fileNames.size() == 1) {
04                  String dir = fileNames.iterator().next().toString();
05                  if (!dir.contains(".xml")) {
06                          repository = dir;
07                          File directory = new File(dir);
08                          File[] files = directory.listFiles();
09                          fileNames.clear();
10                          for (File aux : files) {
11                                  if ((aux.isFile())
12                                                  && (aux.getAbsolutePath().contains(".xml")))
13                                          fileNames.add(aux.getAbsolutePath());
14                          }
15                  } else {
16                          repository = dir.substring(0, dir.lastIndexOf("\\"));
17                  }
18          }
19          Iterator it = fileNames.iterator();
20          policies.clear();
21          while (it.hasNext()) {
22                  String fname = (String) (it.next());
23                  File fl = new File(fname);
24                  if (!fl.exists()) continue;
25                  AbstractPolicy policy = loadPolicy(fname, finder, schemaFile, this);
26                  if (policy != null)
27                          policies.add(policy);
28          }
29  }
```

O método `init` é chamado pela classe `pap` e é responsável por pesquisar no repositório de *Policys* e carregar uma lista com o nome dos arquivos. O método `loadPolicy` é chamado dentro do método `init` para carregar as *Policys* encontradas. No Quadro 15 é demonstrado o trecho de código do método `loadPolicy`. No método `loadPolicy` antes de carregar uma *Policy* é realizado a validação do XML com a especificação *Schema* XML da *Policy*.

Quadro 15 – Método `loadPolicy` da classe `FilePolicyModule`

```
01  public static AbstractPolicy loadPolicy(String filename,
02                  PolicyFinder finder, File schemaFile, ErrorHandler handler) {
03          try {
04                  DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
05                  factory.setIgnoringComments(true);
06
07                  DocumentBuilder db = null;
08
09                  factory.setNamespaceAware(true);
10
11                  if (schemaFile == null) {
12                          factory.setValidating(false);
13
14                          db = factory.newDocumentBuilder();
15                  } else {
16                          factory.setValidating(true);
17
18                          factory.setAttribute(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
19                          factory.setAttribute(JAXP_SCHEMA_SOURCE, schemaFile);
20
21                          db = factory.newDocumentBuilder();
22                          db.setErrorHandler(handler);
23                  }
24
25                  Document doc = db.parse(new FileInputStream(filename));
26
27                  Element root = doc.getDocumentElement();
28                  String name = root.getTagName();
29
30                  if (name.equals("Policy")) {
31                          Policy po = Policy.getInstance(root);
32                          return po;
33                  } else if (name.equals("PolicySet")) {
34                          PolicySet po = PolicySet.getInstance(root, finder);
35                          return po;
36                  } else {
37                          throw new Exception("Unknown root document type: " + name);
38                  }
39          } catch (Exception e) {
40                  if (logger.isLoggable(Level.WARNING))
41                          logger.log(Level.WARNING, "Error reading policy from file "
42                                          + filename, e);
43          }
44          return null;
45  }
```

3.3.3 Operacionalidade da implementação

Esta seção apresenta a operacionalidade da extensão XACML desenvolvida por este trabalho.

Para realizar a demonstração da extensão foram desenvolvidas três aplicações de testes onde é possível demonstrar o funcionamento da extensão e elencar os passos que devem ser seguidos para utilizá-la.

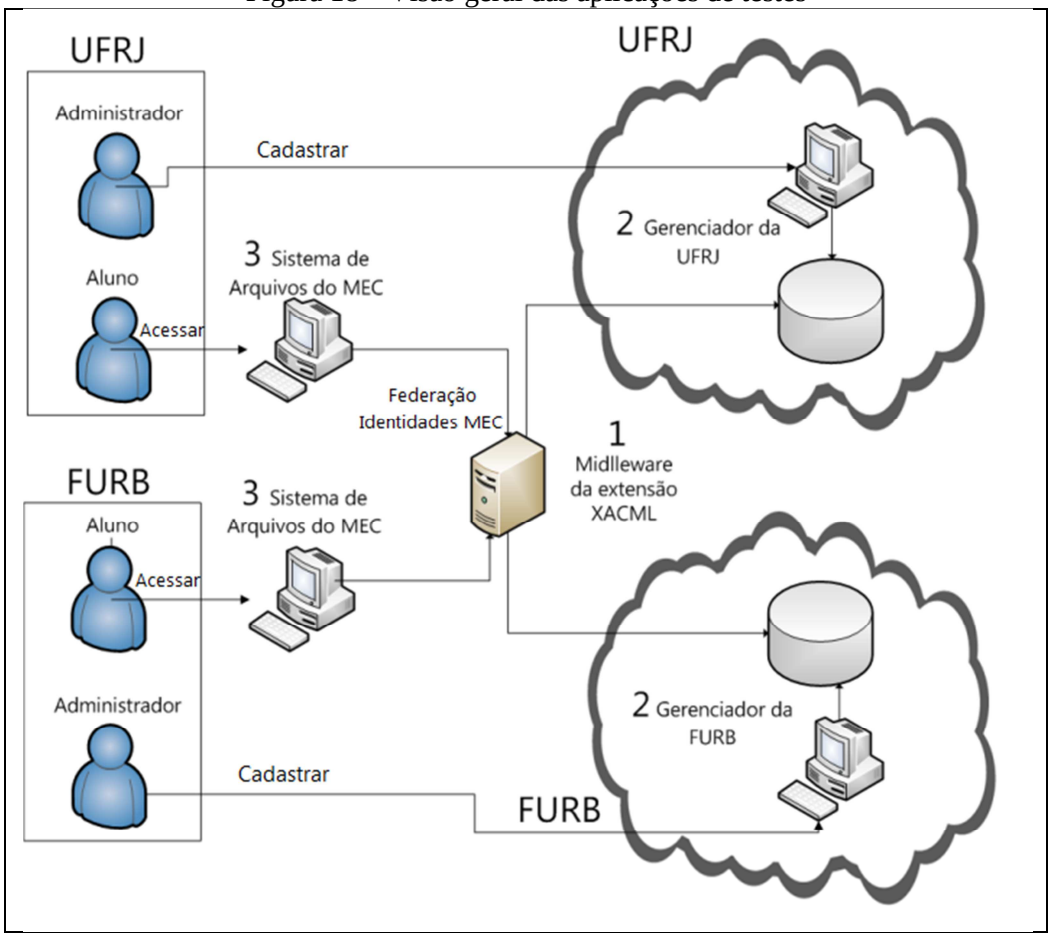
Considerando um cenário hipotético onde o Ministério da Educação (MEC) necessite que alunos das universidades Fundação Universidade Regional de Blumenau (FURB) e Universidade Federal do Rio de Janeiro (UFRJ), utilizem de forma segura e controlada um sistema para compartilhamento e pesquisa de arquivos, foram desenvolvidas três aplicações de testes que irão representar um exemplo de federação das identidades dos alunos das universidades com a aplicação do MEC.

Aplicações de testes desenvolvidas foram:

a) *middleware* de teste da extensão XACML (Figura 18, item 1): é o *Middleware* de testes construído que faz uso da extensão XACML desenvolvida;
b) gerenciador da Furb e gerenciador UFRJ (Figura 18, item 2): é a aplicação de testes que gerencia os usuários. A FURB e UFRJ têm aplicações distintas;
c) sistema de arquivos do MEC (Figura 18, item 3): é a aplicação de teste que faz uso do *middleware* de teste da extensão XACML para controlar e prover segurança dos arquivos compartilhados pelo MEC.

Na Figura 18 é apresentada uma representação geral das aplicações de testes e seus relacionamentos.

Figura 18 – Visão geral das aplicações de testes

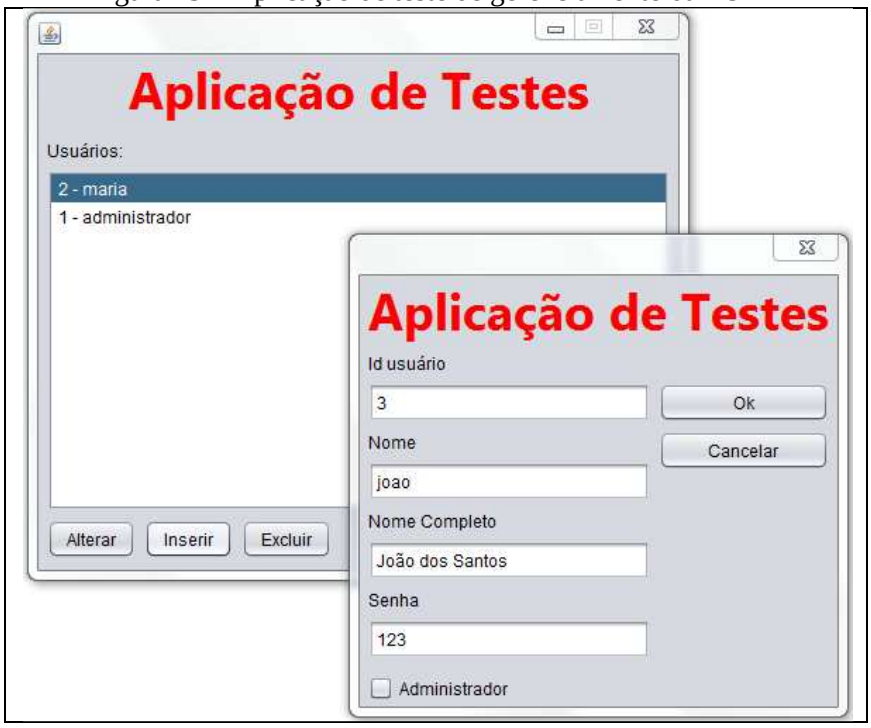


3.3.3.1 Gerenciador

O aplicativo de testes gerenciador tem a responsabilidade de realizar a administração dos usuários compartilhados. A FURB e URFJ têm gerenciadores distintos, eles se conectam no banco de dados que armazena os atributos dos usuários e permite a administração das informações.

Na Figura 19 é demonstrado utilização da aplicação de teste de gerenciamento da FURB para cadastrar o usuário João dos Santos. O usuário Maria também foi cadastrado da mesma forma.

Figura 19 – Aplicação de teste de gerenciamento da FURB



3.3.3.2 *Middleware* de teste da extensão XACML

O *middleware* de teste da extensão XACML utiliza da extensão XACML os recursos para armazenar as políticas aplicadas no sistema de arquivos do MEC bem como permitir controlar o acesso dos alunos da FURB e UFRJ utilizando o conceito de federação.

O *middleware* de teste da extensão XACML foi desenvolvido para rodar no servidor de aplicações Java *Enterprise Edition* (JEE) Tomcat 7.0 e disponibilizar suas funcionalidades por *web services* utilizando a biblioteca Axis2. Os PIPs que representam o relacionamento com a UFRJ e FURB foram implementado para se conectarem em um banco de dados MySQL utilizando a biblioteca MySql Connector Java 5.1.24.

Na Figura 20 é demonstrado o diagrama de classes do *middleware* de teste da extensão XACML.

Figura 20 – Diagrama de classes do *middleware* de teste da extensão XACML

As classes `AttributeFinderFederationFurb` e `AttributeFinderFederationUFRJ` foram implementas para prover ao *middleware* os atributos para federação. No Quadro 16 é apresentado o trecho de código do método `findAttribute` da classe `AttributeFinderFederationUFRJ` que é utilizado no *middleware* para buscar os atributos.

Quadro 16 – Método `findAttribute` da classe `AttributeFinderFederationUFRJ`

```
01  public EvaluationResult findAttribute(URI attributeType, URI attributeId,
02                  URI issuer, URI subjectCategory, EvaluationCtx context,
03                  TargetAtribute designatorType) {
04          Attribute idUsuario = getAttributeIdUsuario(context);
05
06          String sqlStmt = "select " + getNameAtt(attributeId)
07                          + " from usuarios where idUsuario='"
08                          + idUsuario.getValue().encode() + "';";
09          Set<AttributeValue> values = new HashSet<AttributeValue>();
10          PreparedStatement prepStmt = null;
11          ResultSet resultSet = null;
12          Connection connection = null;
13          try {
14                  connection = (Connection) dataSource.getConnection();
15                  if (connection != null) {
16                          prepStmt = (PreparedStatement) connection
17                                          .prepareStatement(sqlStmt);
18                          resultSet = prepStmt.executeQuery();
19
20                          while (resultSet.next()) {
21                                  if (attributeId.toString()
22                                  .equals("urn:oasis:names:tc:xacml:1.0:federation:federation-id"))
23                                  values.add(AttributeFactory.createAttribute(attributeType,
24  ID_FEDERATION));
25                                  else
26                                          values.add(AttributeFactory.createAttribute(
27                                                  attributeType, resultSet.getString(1)));
28                                  }
29                                  if (!values.isEmpty()) {
30                                          return new EvaluationResult(new
31  BagAttribute(attributeType,
32                                                          values));
33                                  }
34                          }
35                  }
```

Quadro 16 – Continuação método `findAttribute` da classe `AttributeFinderFederationUFRJ`

```
38          } catch (SQLException e) {
39                          try {
40                                  throw new Exception("Error while retrieving attribute values", e);
41                          } catch (Exception e1) {
42                          // TODO Auto-generated catch block
43                                  e1.printStackTrace();
44                          }
45                  } catch (UnknownIdentifierException e) {
46                          // TODO Auto-generated catch block
47                          e.printStackTrace();
48                  } catch (ParsingException e) {
49                          // TODO Auto-generated catch block
50                          e.printStackTrace();
51                  } finally {
52                          if (resultSet != null) {
53                                  try {
54                                          resultSet.close();
55                                  } catch (SQLException ignore) {
56
57                                  }
58                          }
59
60                          if (prepStmt != null) {
61                                  try {
62                                          prepStmt.close();
63                                  } catch (SQLException ignore) {
64
65                                  }
66                          }
67
68                          if (connection != null) {
69                                  try {
70                                          connection.close();
71                                  } catch (SQLException ignore) {
72
73                                  }
74                          }
75                  }
76
77                  return new EvaluationResult(BagAttribute.createEmptyBag(attributeType));
78          }
```

Na classe `Pdpmdw` foi implementado os métodos que chamaram os recursos da extensão XACML. No Quadro 17 é demonstrado o trecho de código da inicialização da extensão XACML, onde é setado como *property* o caminho do arquivo de configuração do PDP e retornado a instancia do PDP e PAP.

Quadro 17 – Método construtor da classe `Pdpmdw`

```
01  public Pdpmdw() {
02
03          System.setProperty(ConfigurationStore.PROPERTY_FILE_CONFIG,
04                                          "C:\\Teste_XACML\\config\\config.xml");
05          pdp = PDP.getInstance();
06          pap = PAP.getInstance();
07  }
```

No arquivo de configuração foi informado os elementos responsáveis em estabelecer o relacionamento das federações para prover federação de identidades, foram informadas as classes `AttributeFinderFederationFurb` e `AttributeFinderFederationUFRJ`, as classes `FilePolicyModule` e `FilePolicyModuleReference` juntamente com seus repositórios de políticas, os arquivos com as especificações *Schema* XML das *Policys* e *Requests*. Nos repositórios podem ser utilizado caminhos relativos.

Na Figura 21 é demonstrado parte do arquivo de configuração do *middleware* de teste da extensão XACML.

Figura 21 – Parte do arquivo de configuração do *middleware* de teste da extensão XACML

```
<config xmlns="http://sunxacml.sourceforge.net/schema/config-0.3"
        defaultPDP="pdpSample" defaultAttributeFactory="attr"
        defaultCombiningAlgFactory="comb" defaultFunctionFactory="func"
        defaultFiles="filesConfig">
    <confgfiles name="filesConfig">
        <policySchema class="">
            <string>C:\\Teste_XACML\\policy\\cs-xacml-schema-policy-01_FED.xsd</string>
        </policySchema>
        <contextSchema class="">
            <string>C:\\Teste_XACML\\request\\cs-xacml-schema-context-01_FED.xsd</string>
        </contextSchema>
    </confgfiles>
    <pdp name="pdpSample">
        <attributeFinderModule class="com.sun.xacml.federation.finder.impl.SelectorModule"/>
        <attributeFinderModule class="xacmlmdw.pip.AttributeFinderFederationFurb"/>
        <attributeFinderModule class="xacmlmdw.pip.AttributeFinderFederationUFRJ"/>
        <policyFinderModule class="com.sun.xacml.federation.finder.impl.FilePolicyModule">
            <list>
                <string>C:\\Teste_XACML\\policy\\SetRoot.xml</string>
            </list>
        </policyFinderModule>
        <policyFinderModule class="com.sun.xacml.federation.finder.impl.FilePolicyModuleReference">
            <list>
                <string>C:\\Teste_XACML\\policy\\</string>
            </list>
        </policyFinderModule>
    </pdp>
```

No Quadro 18 é demonstrado trecho de código onde o pedido de *Request* é montado e passado para a extensão. Pode ser observado que é instanciado um objeto da classe `RequestCtx` onde em seu construtor é utilizado a classe `FacadePDP` que auxilia na criação dos atributos que devem ser passados como parâmetro.

Neste momento também é utilizado a classe `ResponseCtx` que armazenara o retorno do *Request*, podendo conter vários resultados caso as políticas forem estruturadas hierarquicamente.

Quadro 18 – Trecho de código do método `verificaPolicy` da classe `PDP`

```
01  public int verificaPolicy(String subject, String resource, String action) {
02          RequestCtx request;
03          try {
04                  request = new RequestCtx(FacadePDP.setupSubject(subject),
05                                  FacadePDP.setupResource(resource),
06                                  FacadePDP.setupAction(action), new HashSet(), new HashSet());
07                  ResponseCtx resp = evaluate(request);
08                  Result result = (Result) resp.getResults().iterator().next();
09                  return result.getDecision();
10
11          } catch (URISyntaxException e) {
12                  // TODO Auto-generated catch block
13                  e.printStackTrace();
14          }
15          return -1;
16  }
17  public ResponseCtx evaluate(RequestCtx evaluate) {
18          // evaluate the request
19          return pdp.evaluate(evaluate);
20  }
```

Na classe `Pdpmdw` também foi implementado o método responsável em chamar o PAP para criar uma *Policy*. No Quadro 19 é demonstrada a montagem de uma *Policy* utilizando as classes da extensão XACML.

Quadro 19 – Trecho de código do método `criarPolicy` da classe `PDP`

```
01  public String criarPolicy(String subject, String resource, String action,
02                          String federation, int effect, String repository)
03                          throws URISyntaxException, FileNotFoundException,
04                          UnknownIdentifierException {
05          String sujeito = "";
06          if (subject.isEmpty())
07                  sujeito = federation;
08          else
09                  sujeito = subject;
10
11          pap.setRepository(repository);
12          PolicySet rootSte = null;
13          if (!pap.getPolicyIds().contains(new URI(SET_ROOT_POLICY))) {
14                  FacadeAccessControl.createPolicySet(SET_ROOT_POLICY, "Policy set raiz",
15               "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:ordered-deny-overrides");
16                  rootSte = FacadeAccessControl.getRootSte();
17          } else {
18                  rootSte = (PolicySet) pap.getPolicy(SET_ROOT_POLICY);
19                  FacadeAccessControl.setRootSte(rootSte);
20          }
21
22          Policy poli = (Policy) pap.getPolicy(resource + sujeito);
23          if (poli == null) {
24                  FacadeAccessControl.addTargetPolicy(TargetAtribute.RESOURCE_TARGET,
25                          new URI("urn:oasis:names:tc:xacml:1.0:function:string-equal"),
26                          new URI("http://www.w3.org/2001/XMLSchema#string"),
27                          new URI("urn:oasis:names:tc:xacml:1.0:resource:resource-id"),
28                          resource);
29
30                  if (!subject.isEmpty())
31                          FacadeAccessControl.addTargetPolicy(TargetAtribute.SUBJECT_TARGET,
32  new URI("urn:oasis:names:tc:xacml:1.0:function:string-equal"),
33                                  new URI("http://www.w3.org/2001/XMLSchema#string"),
34                                  new URI("urn:oasis:names:tc:xacml:1.0:subject:subject-id"),
35                                  subject);
36                          FacadeAccessControl.addTargetRule(TargetAtribute.ACTION_TARGET,
37                                  new URI("urn:oasis:names:tc:xacml:1.0:function:string-equal"),
38                                  new URI("http://www.w3.org/2001/XMLSchema#string"),
39                                  new URI("urn:oasis:names:tc:xacml:1.0:action:action-id"),
40                                  action);
41                  if (!federation.isEmpty())
42                          FacadeAccessControl.addTargetRule(TargetAtribute.FEDERATION_TARGET,
43                                  new URI("urn:oasis:names:tc:xacml:1.0:function:string-equal"),
44                                  new URI("http://www.w3.org/2001/XMLSchema#string"),
45                                  new URI("urn:oasis:names:tc:xacml:1.0:federation:federation-id"),
46                                          federation);
47                  FacadeAccessControl.createRuleList(action, effect);
48                  FacadeAccessControl.createRuleList("default", 1);
49                  FacadeAccessControl.createPolicy(resource + sujeito, resource,
50                       "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-
51  overrides");
52          } else {
53                  Iterator it = poli.getChildren().iterator();
54                  Boolean achou = false;
55                  while (it.hasNext()) {
56                          Rule rl = (Rule) (it.next());
57                          if (rl.getId().equals(new URI(action))) {
58                                  rl.setEffect(effect);
59                                  achou = true;
60                                  break;
61                          }
62                  }
63                  if (!achou) {
64                          FacadeAccessControl
```

Quadro 19 – Continuação trecho de código do método `criarPolicy` da classe `PDP`

```
65                                          .addTargetRule(
66                                                  TargetAtribute.ACTION_TARGET,
67                                                  new URI(
68
69          "urn:oasis:names:tc:xacml:1.0:function:string-equal"),
70                                                  new URI(
71
72          "http://www.w3.org/2001/XMLSchema#string"),
73                                                  new URI(
74
75          "urn:oasis:names:tc:xacml:1.0:action:action-id"),
76                                                  action);
77                          poli.getChildren().add(facAcc.createRule(action, effect));
78                  }
79                  pap.savePolicyXML(poli);
80                  pap.loadPolicy();
81          }
82          return "ok";
83  }
```

Na classe `Mdw` foi implementado os métodos estáticos `verificarPermisao`, `adicionarPermisao`, `getUsers` chamados nos *web services*. No Quadro 20 é demonstrada a classe `Mdw`.

Quadro 20 – Classe `Mdw`

```
01  package xacmlmdw;
02
03  import java.io.FileNotFoundException;
04  import java.net.URISyntaxException;
05  import java.util.ArrayList;
06
07  import com.sun.xacml.federation.cond.FunctionTypeException;
08  import com.sun.xacml.federation.finder.User;
09  import com.sun.xacml.federation.policy.UnknownIdentifierException;
10
11  public class Mdw {
12
13          private static Pdpmdw pdpmdw = new Pdpmdw();
14
15          public static int verificarPermisao(String subject, String resource,
16                          String action) {
17
18                  return pdpmdw.verificaPolicy(subject, resource, action);
19          }
20
21          public static String adicionarPermisao(String subject, String resource,
22                          String action, String federation, int effect, String repository)
23                          throws FileNotFoundException, URISyntaxException,
24                          UnknownIdentifierException, FunctionTypeException {
25
26                  return pdpmdw.criarPolicy(subject, resource, action, federation,
27                                  effect, repository);
28          }
29
30          public static ArrayList<User> getUsers() {
31                  return pdpmdw.getPdp().getUsers();
32          }
33
34          public static ArrayList<String> getPolicyIds() {
35                  return pdpmdw.getPap().getPolicyIds();
36          }
37  }
```

Utilizando a biblioteca Axis2, foram criados *web services* dos métodos estáticos e realizado *deploy* no servidor de aplicação JEE Tomcat.

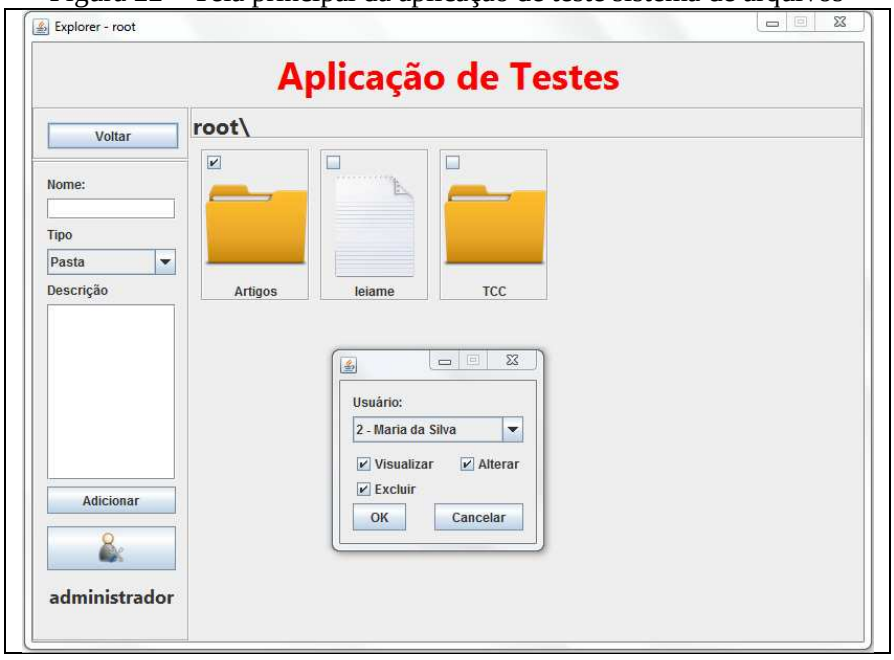
3.3.3.3 Sistema de arquivos do MEC

O sistema de arquivos do MEC é uma aplicação de teste que faz uso do *middleware* de teste da extensão XACML para controlar e prover segurança dos arquivos compartilhados pelo MEC, onde alunos da FURB e UFRJ usaram os usuários e senha já utilizados nas suas instituições pertencentes, resumindo a aplicação de teste tem dois objetivos, o primeiro é simular as ações de visualizar, alterar e excluir pastas ou arquivos, o segundo é simular a administração e controle das ações dos usuários sobre os arquivos e pastas.

Primeiramente foi desenvolvida uma tela de *login* para identificar os usuários. A tela de *login* verifica dentre os usuários retornados do *middleware* qual corresponde ao usuário e senha.

A parte central da tela é destinada a apresentação dos arquivos fictícios. Na parte superior é apresentado o botão voltar e o caminho de diretórios que está sendo visualizado. Na pare inferior esquerda, estão localizados os controles para criação de um novo arquivo ou pasta. Para aplicar permissões, o usuário administrador seleciona as pastas e arquivos e em seguida clica no botão de administração, na qual é apresentada a tela onde é possível selecionar um usuário ou federação para aplicar as permissões. Na Figura 22 é demonstrada a aplicação de teste do sistema de arquivos, onde o usuário administrador aplica permissão na pasta Artigos para que o usuário Maria da Silva consiga visualizar, excluir e alterar. Da mesma forma, foi aplicado permissão para que o usuário João dos Santos não consiga excluir e alterar, permitindo apenas visualizar.

Figura 22 – Tela principal da aplicação de teste sistema de arquivos



Na tela de permissões também é possível aplicar uma permissão para todos os alunos da FURB ou UFRJ, permitindo assim aplicar permissões por federações.

A tela de permissões utiliza a classe `FacadeMDW` chamando o método `adicionarPermisao` para que *middleware* crie a política.

A comunicação da aplicação com o *middleware* é feita pela classe façada `FacadeMDW`, na qual tem a implementação das chamadas de *web service* do *middleware*. As chamadas *web service* foram desenvolvidas utilizando a biblioteca Axis2 para gerar a classe *client* do *web service* utilizando o arquivo *Web Services Description Language* (WSDL).

No Quadro 21 é demonstrado trecho de código da tela de permissões que aplica permissões.

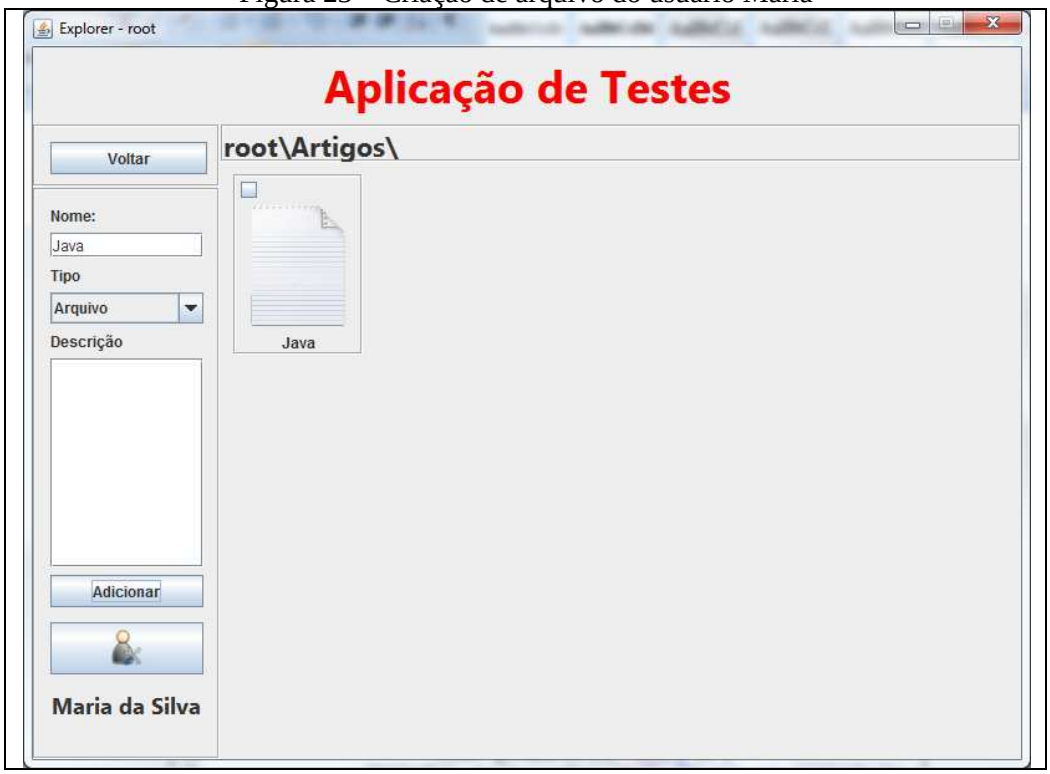
Quadro 21 – Trecho de código da tela de permissões

```
01  private void jbOkActionPerformed(java.awt.event.ActionEvent evt) {
02
03          MdwStub.User user = (User) jcUsuario.getSelectedItem();
04
05          if (jcVisualizar.isSelected()) {
06                  FacadeMDW.adicionarPermisao(user.getIdUser(), sourcers, "read",
07                                  user.getFederation(), 0);
08          } else {
09                  FacadeMDW.adicionarPermisao(user.getIdUser(), sourcers, "read",
10                                  user.getFederation(), 1);
11          }
12
13          if (jcExcluir.isSelected()) {
14                  FacadeMDW.adicionarPermisao(user.getIdUser(), sourcers, "delete",
15                                  user.getFederation(), 0);
16          } else {
17                  FacadeMDW.adicionarPermisao(user.getIdUser(), sourcers, "delete",
18                                  user.getFederation(), 1);
19          }
20
21          if (jcAlterar.isSelected()) {
22                  FacadeMDW.adicionarPermisao(user.getIdUser(), sourcers, "modify",
23                                  user.getFederation(), 0);
24          } else {
25                  FacadeMDW.adicionarPermisao(user.getIdUser(), sourcers, "modify",
26                                  user.getFederation(), 1);
27          }
28
29          dispose();
30
31  }
```

Quando o usuário tentar visualizar, excluir ou alterar uma pasta ou arquivo, a aplicação valida se o usuário tem permissão para esta ação.

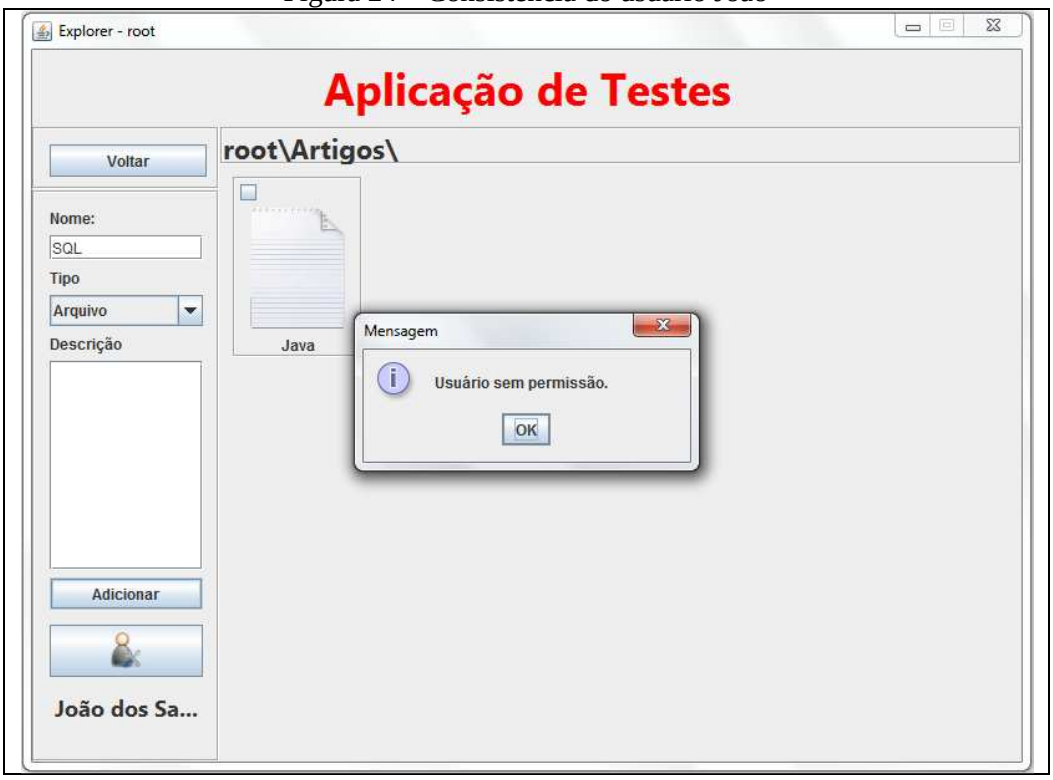
Na Figura 23 é demonstrado que o sistema permitiu o usuário Maria cadastrar um arquivo na pasta Artigo.

Figura 23 – Criação de arquivo do usuário Maria



Na figura 24 é demonstrado que o sistema não permitiu que o usuário João cria-se um novo arquivo chamado SQL dentro da pasta Artigos, pois o usuário não tem permissão para alterar a pasta.

Figura 24 – Consistência do usuário João



No Quadro 22 é demonstrado o trecho de código da validação feita no botão adicionar da aplicação de teste sistema de arquivos. O método `btnAdicionar` utiliza a classe `FacadeMDW` chamando o método `verificaPermissao`.

Quadro 22 – Trecho de código da validação feita no botão adicionar

```
01  private void btnAdicionar(java.awt.event.ActionEvent evt) {
02
03          if (!FacadeMDW.verificaPermissao(user.getIdUser(),
04                          folder.getNamePath(), "modify")) {
05                  JOptionPane.showMessageDialog(null, "Usuário sem permissão.");
06                  return;
07          }
08
09          if (jtName.getText().isEmpty()) {
10                  return;
11          }
12
13          if (folder.getFiles().containsKey(jtName.getText())) {
14                  return;
15          }
16
17          if (jcFileType.getSelectedIndex() == 0) {
18                  folder.add(new dados.Folder(jtName.getText(), jtDesc.getText(),
19                                  folder));
20          } else {
21                  folder.add(new dados.File(jtName.getText(), jtDesc.getText(),
22                                  folder));
23          }
24          refreshTela();
25  }
```

No Quadro 23 é apresentado o trecho de código do método `verificaPermissao` da classe `FacadeMDW`.

Quadro 23 – Método `verificaPermissao` da classe `FacadeMDW`

```
01  public static boolean verificaPermissao(String subject, String resource,
02                  String action) {
03          MdwStub proxy;
04          try {
05                  proxy = new MdwStub();
06                  MdwStub.VerificarPermisao ids = new MdwStub.VerificarPermisao();
07                  MdwStub.VerificarPermisaoResponse resp = new MdwStub.VerificarPermisaoResponse();
08                  ids.setAction(action);
09                  ids.setResource(resource);
10                  ids.setSubject(subject);
11                  resp = proxy.verificarPermisao(ids);
12                  if ((resp.get_return() != DECISION_PERMIT)
13                                  && (resp.get_return() != DECISION_NOT_APPLICABLE)) {
14                          return false;
15                  }
16                  return true;
17          } catch (AxisFault e) {
18                  // TODO Auto-generated catch block
19                  e.printStackTrace();
20          } catch (RemoteException e) {
21                  // TODO Auto-generated catch block
22                  e.printStackTrace();
23          }
24          return false;
25  }
```

Desta forma foi estabelecido um relacionamento de confiança entre as federações criando a federação de identidades, proporcionando maior flexibilização no controle de acesso das aplicações. O usuário que tem sua identidade compartilhada pelas nuvens da federação

que formam a federação de identidades, poderá de forma transparente utilizar as aplicações com apenas uma credencial, sem redundância de cadastramento de usuários.

3.4 RESULTADOS E DISCUSSÃO

Os resultados encontrados com o término do trabalho demonstram um importante avanço para o padrão XACML, pois com a utilização da extensão XACML é possível realizar o controle de acesso utilizando o conceito de federações de identidades, permitindo gerenciar políticas de acesso, interpretar *Request* e *Response* e identificar a política a ser aplicada.

A utilização da extensão juntamente com um modelo de nuvem computacional deve seguir os mesmos passos descritos na operacionalidade. Porém a aplicação da extensão em uma situação real requer cuidados em relação à demanda de infraestrutura, tendo em vista que ao adicionar mais uma federação nas aplicações, estará aumentando muito quantidade de usuários e acesso aos servidores, ou então a retirada de uma federação, pode fazer com que tenha uma infraestrutura desproporcional a sua demanda. Portando a utilização de um modelo de nuvem computacional nas aplicações que utilizaram a extensão, ira anular os possíveis problemas de infraestrutura, pois a aplicação usufruirá do provisionamento dinâmico de recursos sob demanda e escalabilidade proporcionados pela nuvem.

No Quadro 24 é apresentado o comparativo das características da aplicação com os trabalhos correlatos.

Quadro 24 – Características da aplicação e trabalhos correlatos

Características	**Este Trabalho**	**Toktar(2003)**	**Leandro(2012)**
Federação de Identidades	✓	×	✓
XACML	✓	✓	×
Extensão XACML	✓	✓	×
Java	✓	✓	✓
Utilizou outro Protocolo	×	✓	×
Modelo de Nuvem	✓	×	✓

O trabalho de Toktar (2003) é diferente deste trabalho, pois não implementa o conceito de federação de identidades e utiliza o protocolo RSVP para alocação de recursos.

O trabalho de Leandro (2012) assim com este trabalho, utilizou o conceito de federação de identidades, demonstrando o relacionamento entre as federações, porém não utilizou um padrão para controle de acesso.

Pode-se observar que todas as aplicações utilizam Java para as implementações.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma extensão XACML e também demonstrou o uso da extensão em aplicações de testes.

As aplicações de testes entregues por este trabalho atendem os objetivos propostos, sendo útil para demonstrar o uso da extensão para os desenvolvedores interessados em utilizá-la e também demonstrar a sua forma de funcionamento.

A linguagem XACML se demonstrou muito flexível e extensível para diversas situações e necessidades, porém tem a limitação de não prover autenticação, o que demandou uma implementação pela aplicação. Foi percebido que as aplicações desenvolvidas utilizando a linguagem XACML devem abstrair do usuário final a necessidade de entender XACML, pois a linguagem é complexa e pode prejudicar a experiência do usuário com a aplicação.

Como o padrão XACML não prevê a administração de usuários, o conceito de federação de identidades se encaixou bem na extensão, pois descentraliza a administração de usuários e passa essa responsabilidade para as federações.

A API Sun's XACML em JAVA foi utilizada como base para a extensão, se demonstrou bem intuitiva e estrutura. Comparada com outras implementações, a API Sun's XACML é bem mais simples e leve, e por estes motivos foi escolhida como base da extensão.

4.1 EXTENSÕES

Durante o desenvolvimento do trabalho foram identificados alguns pontos de melhora na extensão e também na forma de utilizá-la

A extensão hoje realiza o controle de acesso após o usuário já estar autenticado e identificado com *id* único, portanto uma melhoria para a extensão seria a implementação de autenticação de usuário considerando o conceito de federação de identidades.

O trabalho também apresentou um *middleware* de teste que utilizou os principais recursos da extensão. Esse *middleware* poderia ser evoluído para se tornar um *middleware* genérico, permitindo criar qualquer tipo de controle de acesso, provendo encriptação na transferência de dados e controle de concorrência.

A extensão está especificada utilizando XML, porém poderia evoluída para trabalhar com *JavaScript Object Notation* (JSON).

Outra extensão proposta, seria a evolução do PIP, provendo uma forma mais fácil de adicionar um novo PIP nas aplicações, sem a necessidade de implementar uma nova classe a cada situação. O objetivo seria prever apenas algumas formas de PIP padrão e apenas informar alguns dados no arquivo de configuração do PDP.

A última extensão proposta é em relação aos recursos controlados, a extensão poderia prover o cadastramento dos recursos e suas características.

REFERÊNCIAS BIBLIOGRÁFICAS

CAMARGO, Edson T. et al. **Autenticação e autorização em arquiteturas orientadas a serviço através de identidades federadas**. Florianópolis, São José, 2007. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbrc/2007/006.pdf>. Acesso em: 10 set. 2012.

FAÍSCA, José. **Gestão federativa de identidade:** iniciativas e padrões emergentes. [S.l.], [2009]. Disponível em: <http://conferencias.ulusofona.pt/index.php/sopcom_iberico/sopcom_iberico09/paper/viewFile/310/286>. Acesso em: 17 set. 2012.

LAUREANO. Marcos A. P. **Gestão de segurança da informação**. [S.l.], 2005. Disponível em: <http://pt.scribd.com/doc/100687499/18/Autenticacao-e-autorizacao>. Acesso em: 16 set. 2012.

LEANDRO, Marcos A. P. **Federação de identidades e computação em nuvem:** estudo de caso usando shibboleth. Florianópolis, 2012. Disponível em: <http://www.inf.furb.br/~paulofernando/disciplinas/TCC/UFSC_2012_Disserta%e7%e3oMarcosLeandro.pdf>. Acesso em: 10 set. 2012.

MELLO, Emerson R.; et al. **Segurança em serviços web.** Porto Alegre, 2006. Disponível em: <http://tele.sj.ifsc.edu.br/~mello/artigos/mellomcsbseg06.pdf>. Acesso em: 17 set. 2012.

OASIS. **eXtensible Access Control Markup Language (XACML).** [S.l.], 2005a. Disponível em: <http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf>. Acesso em: 01 ago. 2012.

_____. **eXtensible Access Control Markup Language (XACML).** [S.l.], 2005b. Disponível em: <http://docs.oasis-open.org/xacml>. Acesso em: 01 ago. 2012.

SANDHU, Ravi S.; SAMARATI, Pierangela. **Authentication, access control, and audit**. Milano, 1996. Disponível em: <http://www.profsandhu.com/journals/acm/survey96(org).pdf>. Acesso em: 16 set. 2012.

SILVA, Felipe. P. **Federação de identidades e segurança em SOA –** conceitos e casos. [S.l.], 2009. Disponível em: <ftp://ftp.software.ibm.com/la/documents/imc/br/security_forum/fim_e_soa_penaranda.pdf>. Acesso em: 10 set. 2012.

SUN MICROSYSTEMS. **Sun's XACML implementation programmer's guide**: help. Version 1.2. [S.l.], 2004a. Disponível em: <http://sunxacml.sourceforge.net/guide.html>. Acesso em: 05 abr. 2013.

_____. **Sun's XACML Implementation run-time configuration guide**: help. Version 1.2. [S.l.], 2004b. Disponível em: <http://sunxacml.sourceforge.net/config.html>. Acesso em: 05 abr. 2013.

_____. **Sun's XACML implementation**: help. Version 1.2. [S.l.], 2004c. Disponível em: <http://sunxacml.sourceforge.net/javadoc/>. Acesso em: 05 abr. 2013.

TOKTAR, Emir. **Controle de admissão de RSVP utilizando XACML.** Curitiba, 2003. Disponível em: <http://www.ppgia.pucpr.br/lib/exe/fetch.php?media=dissertacoes:2005:emir_toktar-2003.pdf>. Acesso em: 10 set. 2012.

APÊNDICE A – Especificação *Schema* XML de *Policy*

O Quadro 25 apresenta a especificação *Schema* XML da Policy.

Quadro 25 – Especificação *Schema* XML da *Policy*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<!-- -->
	<xs:element name="PolicySet" type="xacml:PolicySetType"/>
	<xs:complexType name="PolicySetType">
		<xs:sequence>
			<xs:element ref="xacml:Description" minOccurs="0"/>
			<xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
			<xs:element ref="xacml:Target"/>
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="xacml:PolicySet"/>
				<xs:element ref="xacml:Policy"/>
				<xs:element ref="xacml:PolicySetIdReference"/>
				<xs:element ref="xacml:PolicyIdReference"/>
			</xs:choice>
			<xs:element ref="xacml:Obligations" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
		<xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="PolicySetIdReference" type="xs:anyURI"/>
	<xs:element name="PolicyIdReference" type="xs:anyURI"/>
	<!-- -->
	<xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
	<xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
	<xs:complexType name="DefaultsType">
		<xs:sequence>
			<xs:choice>
				<xs:element ref="xacml:XPathVersion"/>
			</xs:choice>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="XPathVersion" type="xs:anyURI"/>
	<!-- -->
	<xs:element name="Policy" type="xacml:PolicyType"/>
	<xs:complexType name="PolicyType">
		<xs:sequence>
			<xs:element ref="xacml:Description" minOccurs="0"/>
			<xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
			<xs:element ref="xacml:Target"/>
			<xs:element ref="xacml:Rule" minOccurs="0" maxOccurs="unbounded"/>
			<xs:element ref="xacml:Obligations" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
		<xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="Description" type="xs:string"/>
	<!-- -->
	<xs:element name="Rule" type="xacml:RuleType"/>
	<xs:complexType name="RuleType">
		<xs:sequence>
			<xs:element ref="xacml:Description" minOccurs="0"/>
			<xs:element ref="xacml:Target" minOccurs="0"/>
			<xs:element ref="xacml:Condition" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="RuleId" type="xs:anyURI" use="required"/>
		<xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
	</xs:complexType>
	<!-- -->
```

Quadro 25 – Continuação especificação *Schema* XML da *Policy*

```
	<xs:simpleType name="EffectType">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Permit"/>
			<xs:enumeration value="Deny"/>
		</xs:restriction>
	</xs:simpleType>
	<!-- -->
	<xs:element name="Target" type="xacml:TargetType"/>
	<xs:complexType name="TargetType">
		<xs:sequence>
			<xs:element ref="xacml:Subjects"/>
			<xs:element ref="xacml:Resources"/>
			<xs:element ref="xacml:Actions"/>
			<xs:element ref="xacml:Federations"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="Subjects" type="xacml:SubjectsType"/>
	<xs:complexType name="SubjectsType">
		<xs:choice>
			<xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
			<xs:element ref="xacml:AnySubject"/>
		</xs:choice>
	</xs:complexType>
	<!-- -->
	<xs:element name="Subject" type="xacml:SubjectType"/>
	<xs:complexType name="SubjectType">
		<xs:sequence>
			<xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="AnySubject"/>
	<!-- -->
	<xs:element name="Resources" type="xacml:ResourcesType"/>
	<xs:complexType name="ResourcesType">
		<xs:choice>
			<xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
			<xs:element ref="xacml:AnyResource"/>
		</xs:choice>
	</xs:complexType>
	<!-- -->
	<xs:element name="AnyResource"/>
	<!-- -->
	<xs:element name="Resource" type="xacml:ResourceType"/>
	<xs:complexType name="ResourceType">
		<xs:sequence>
			<xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="Actions" type="xacml:ActionsType"/>
	<xs:complexType name="ActionsType">
		<xs:choice>
			<xs:element ref="xacml:Action" maxOccurs="unbounded"/>
			<xs:element ref="xacml:AnyAction"/>
		</xs:choice>
	</xs:complexType>
	<!-- -->
	<xs:element name="AnyAction"/>
	<!-- -->
	<xs:element name="Action" type="xacml:ActionType"/>
	<xs:complexType name="ActionType">
		<xs:sequence>
			<xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="Federations" type="xacml:FederationsType"/>
	<xs:complexType name="FederationsType">
		<xs:choice>
			<xs:element ref="xacml:Federation" maxOccurs="unbounded"/>
			<xs:element ref="xacml:AnyFederation"/>
		</xs:choice>
```

Quadro 25 – Continuação especificação *Schema* XML da *Policy*

```
	</xs:complexType>
	<!-- -->
	<xs:element name="AnyFederation"/>
	<!-- -->
	<xs:element name="Federation" type="xacml:FederationType"/>
	<xs:complexType name="FederationType">
		<xs:sequence>
			<xs:element ref="xacml:FederationMatch" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="Environments" type="xacml:EnvironmentsType"/>
	<xs:complexType name="EnvironmentsType">
		<xs:choice>
			<xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
			<xs:element ref="xacml:AnyEnvironment"/>
		</xs:choice>
	</xs:complexType>
	<!-- -->
	<xs:element name="AnyEnvironment"/>
	<!-- -->
	<xs:element name="Environment" type="xacml:EnvironmentType"/>
	<xs:complexType name="EnvironmentType">
		<xs:sequence>
			<xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
	<xs:complexType name="SubjectMatchType">
		<xs:sequence>
			<xs:element ref="xacml:AttributeValue"/>
			<xs:choice>
				<xs:element ref="xacml:SubjectAttributeDesignator"/>
				<xs:element ref="xacml:AttributeSelector"/>
			</xs:choice>
		</xs:sequence>
		<xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
	<xs:complexType name="ResourceMatchType">
		<xs:sequence>
			<xs:element ref="xacml:AttributeValue"/>
			<xs:choice>
				<xs:element ref="xacml:ResourceAttributeDesignator"/>
				<xs:element ref="xacml:AttributeSelector"/>
			</xs:choice>
		</xs:sequence>
		<xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
	<xs:complexType name="ActionMatchType">
		<xs:sequence>
			<xs:element ref="xacml:AttributeValue"/>
			<xs:choice>
				<xs:element ref="xacml:ActionAttributeDesignator"/>
				<xs:element ref="xacml:AttributeSelector"/>
			</xs:choice>
		</xs:sequence>
		<xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="EnvironmentMatch" type="xacml:EnvironmentMatchType"/>
	<xs:complexType name="EnvironmentMatchType">
		<xs:sequence>
			<xs:element ref="xacml:AttributeValue"/>
			<xs:choice>
				<xs:element ref="xacml:EnvironmentAttributeDesignator"/>
				<xs:element ref="xacml:AttributeSelector"/>
			</xs:choice>
		</xs:sequence>
		<xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
	</xs:complexType>
```

Quadro 25 – Continuação especificação *Schema* XML da *Policy*

```
	<!-- -->
	<xs:element name="FederationMatch" type="xacml:FederationMatchType"/>
	<xs:complexType name="FederationMatchType">
		<xs:sequence>
			<xs:element ref="xacml:AttributeValue"/>
			<xs:choice>
				<xs:element ref="xacml:FederationAttributeDesignator"/>
				<xs:element ref="xacml:AttributeSelector"/>
			</xs:choice>
		</xs:sequence>
		<xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<!-- -->
	<xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"/>
	<xs:complexType name="AttributeSelectorType">
		<xs:attribute name="RequestContextPath" type="xs:string" use="required"/>
		<xs:attribute name="DataType" type="xs:anyURI" use="required"/>
		<xs:attribute name="MustBePresent" type="xs:boolean" use="optional"
default="false"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="ResourceAttributeDesignator" type="xacml:AttributeDesignatorType"/>
	<xs:element name="ActionAttributeDesignator" type="xacml:AttributeDesignatorType"/>
	<xs:element name="EnvironmentAttributeDesignator"
type="xacml:AttributeDesignatorType"/>
	<xs:element name="FederationAttributeDesignator" type="xacml:AttributeDesignatorType"/>
	<!-- -->
	<xs:complexType name="AttributeDesignatorType">
		<xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
		<xs:attribute name="DataType" type="xs:anyURI" use="required"/>
		<xs:attribute name="Issuer" type="xs:string" use="optional"/>
		<xs:attribute name="MustBePresent" type="xs:boolean" use="optional"
default="false"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="SubjectAttributeDesignator"
type="xacml:SubjectAttributeDesignatorType"/>
	<xs:complexType name="SubjectAttributeDesignatorType">
		<xs:complexContent>
			<xs:extension base="xacml:AttributeDesignatorType">
				<xs:attribute name="SubjectCategory" type="xs:anyURI"
use="optional" default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<!-- -->
	<xs:element name="AttributeValue" type="xacml:AttributeValueType"/>
	<xs:complexType name="AttributeValueType" mixed="true">
		<xs:sequence>
			<xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:attribute name="DataType" type="xs:anyURI" use="required"/>
		<xs:anyAttribute namespace="##any" processContents="lax"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="Function" type="xacml:FunctionType"/>
	<xs:complexType name="FunctionType">
		<xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="Apply" type="xacml:ApplyType"/>
	<xs:element name="Condition" type="xacml:ApplyType"/>
	<!-- -->
	<xs:complexType name="ApplyType">
		<xs:choice minOccurs="0" maxOccurs="unbounded">
			<xs:element ref="xacml:Apply"/>
			<xs:element ref="xacml:Function"/>
			<xs:element ref="xacml:AttributeValue"/>
			<xs:element ref="xacml:SubjectAttributeDesignator"/>
			<xs:element ref="xacml:ResourceAttributeDesignator"/>
			<xs:element ref="xacml:ActionAttributeDesignator"/>
			<xs:element ref="xacml:EnvironmentAttributeDesignator"/>
			<xs:element ref="xacml:FederationAttributeDesignator"/>
```

Quadro 25 – Continuação especificação *Schema* XML da *Policy*

```
                        <xs:element ref="xacml:AttributeSelector"/>
                </xs:choice>
                <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
                <!-- Legal types for the first and subsequent operands are defined in the
accompanying table -->
        </xs:complexType>
        <!-- -->
        <xs:element name="Obligations" type="xacml:ObligationsType"/>
        <xs:complexType name="ObligationsType">
                <xs:sequence>
                        <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <!-- -->
        <xs:element name="Obligation" type="xacml:ObligationType"/>
        <xs:complexType name="ObligationType">
                <xs:sequence>
                        <xs:element ref="xacml:AttributeAssignment" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
                <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
        <xs:complexType name="AttributeAssignmentType" mixed="true">
                <xs:complexContent mixed="true">
                        <xs:extension base="xacml:AttributeValueType">
                                <xs:attribute name="AttributeId" type="xs:anyURI"
use="required"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <!-- -->
</xs:schema>
```

APÊNDICE B – Especificação *Schema* XML de *Request*

O Quadro 26 apresenta a especificação *Schema* XML da *Request*.

Quadro 26 – Especificação *Schema* XML da *Request*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:1.0:context"
           xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy"
                  xmlns:xacml-context="urn:oasis:names:tc:xacml:1.0:context"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  elementFormDefault="qualified"
                  attributeFormDefault="unqualified">
       <xs:import namespace="urn:oasis:names:tc:xacml:1.0:policy" schemaLocation="cs-xacml-
schema-policy-01.xsd"/>
       <!--    -->
       <xs:element name="Request" type="xacml-context:RequestType"/>
       <xs:complexType name="RequestType">
              <xs:sequence>
                     <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
                     <xs:element ref="xacml-context:Resource"/>
                     <xs:element ref="xacml-context:Action"/>
                     <xs:element ref="xacml-context:Environment" minOccurs="0"/>
                     <xs:element ref="xacml-context:Federation" />
              </xs:sequence>
       </xs:complexType>
       <!-- -->
       <xs:element name="Response" type="xacml-context:ResponseType"/>
       <xs:complexType name="ResponseType">
              <xs:sequence>
                     <xs:element ref="xacml-context:Result" maxOccurs="unbounded"/>
              </xs:sequence>
       </xs:complexType>
       <!-- -->
       <xs:element name="Subject" type="xacml-context:SubjectType"/>
       <xs:complexType name="SubjectType">
              <xs:sequence>
                     <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
              <xs:attribute name="SubjectCategory" type="xs:anyURI" use="optional"
default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
       </xs:complexType>
       <!-- -->
       <xs:element name="Resource" type="xacml-context:ResourceType"/>
       <xs:complexType name="ResourceType">
              <xs:sequence>
                     <xs:element ref="xacml-context:ResourceContent" minOccurs="0"/>
                     <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
       </xs:complexType>
       <!-- -->
       <xs:element name="ResourceContent" type="xacml-context:ResourceContentType"/>
       <xs:complexType name="ResourceContentType" mixed="true">
              <xs:sequence>
                     <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
              <xs:anyAttribute namespace="##any" processContents="lax"/>
       </xs:complexType>
       <!-- -->
       <xs:element name="Action" type="xacml-context:ActionType"/>
       <xs:complexType name="ActionType">
              <xs:sequence>
                     <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
       </xs:complexType>
       <!-- -->
```

Quadro 26 – Continuação especificação *Schema* XML do *Request*

```
	<xs:element name="Environment" type="xacml-context:EnvironmentType"/>
	<xs:complexType name="EnvironmentType">
		<xs:sequence>
			<xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="Federation" type="xacml-context:FederationType"/>
	<xs:complexType name="FederationType">
		<xs:sequence>
			<xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="Attribute" type="xacml-context:AttributeType"/>
	<xs:complexType name="AttributeType">
		<xs:sequence>
			<xs:element ref="xacml-context:AttributeValue"/>
		</xs:sequence>
		<xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
		<xs:attribute name="DataType" type="xs:anyURI" use="required"/>
		<xs:attribute name="Issuer" type="xs:string" use="optional"/>
		<xs:attribute name="IssueInstant" type="xs:dateTime" use="optional"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="AttributeValue" type="xacml-context:AttributeValueType"/>
	<xs:complexType name="AttributeValueType" mixed="true">
		<xs:sequence>
			<xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:anyAttribute namespace="##any" processContents="lax"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="Result" type="xacml-context:ResultType"/>
	<xs:complexType name="ResultType">
		<xs:sequence>
			<xs:element ref="xacml-context:Decision"/>
			<xs:element ref="xacml-context:Status"/>
			<xs:element ref="xacml:Obligations" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="ResourceId" type="xs:string" use="optional"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="Decision" type="xacml-context:DecisionType"/>
	<xs:simpleType name="DecisionType">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Permit"/>
			<xs:enumeration value="Deny"/>
			<xs:enumeration value="Indeterminate"/>
			<xs:enumeration value="NotApplicable"/>
		</xs:restriction>
	</xs:simpleType>
	<!-- -->
	<xs:element name="Status" type="xacml-context:StatusType"/>
	<xs:complexType name="StatusType">
		<xs:sequence>
			<xs:element ref="xacml-context:StatusCode"/>
			<xs:element ref="xacml-context:StatusMessage" minOccurs="0"/>
			<xs:element ref="xacml-context:StatusDetail" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="StatusCode" type="xacml-context:StatusCodeType"/>
	<xs:complexType name="StatusCodeType">
		<xs:sequence>
			<xs:element ref="xacml-context:StatusCode" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="Value" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="StatusMessage" type="xs:string"/>
	<!-- -->
```

Quadro 26 – Continuação especificação *Schema* XML do *Request*

```
        <xs:element name="StatusDetail" type="xacml-context:StatusDetailType"/>
        <xs:complexType name="StatusDetailType">
                <xs:sequence>
                        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

APÊNDICE C – Especificação *Schema* XML do arquivo de configuração do PDP

O Quadro 27 apresenta a especificação *Schema* XML do arquivo de configuração do PDP.

Quadro 27 – Especificação *Schema* XML do arquivo de configuração do PDP

```
<xs:schema targetNamespace="http://sunxacml.sourceforge.net/schema/config-0.3"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:config="http://sunxacml.sourceforge.net/schema/config-0.3"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">
       <xs:element name="config">
              <xs:complexType>
                     <xs:choice minOccurs="0" maxOccurs="unbounded">
                            <xs:element name="confgfiles" type="config:ConfgFilesType"/>
                            <xs:element name="pdp" type="config:PDPType"/>
                            <xs:element name="attributeFactory"
type="config:AttributeFactoryType"/>
                            <xs:element name="combiningAlgFactory"
type="config:CombiningFactoryType"/>
                            <xs:element name="functionFactory"
type="config:FunctionFactoryType"/>
                     </xs:choice>
              </xs:complexType>
              <xs:attribute name="defaultPDP" type="xs:string" use="required"/>
              <xs:attribute name="defaultAttributeFactory" type="xs:string"
                  use="required"/>
              <xs:attribute name="defaultCombiningAlgFactory" type="xs:string"
                  use="required"/>
              <xs:attribute name="defaultFunctionFactory" type="xs:string"
                  use="required"/>
              <xs:attribute name="defaultFiles" type="xs:string" use="required"/>
       </xs:element>
       <xs:complexType name="ConfgFilesType">
              <xs:choice minOccurs="0" maxOccurs="unbounded">
                     <xs:element name="policySchema" type="config:ClassType"/>
                     <xs:element name="contextSchema" type="config:ClassType"/>
              </xs:choice>
              <xs:attribute name="name" type="xs:string" use="required"/>
       </xs:complexType>
       <xs:complexType name="PDPType">
              <xs:choice minOccurs="0" maxOccurs="unbounded">
                     <xs:element name="attributeFinderModule" type="config:ClassType"/>
                     <xs:element name="resourceFinderModule" type="config:ClassType"/>
                     <xs:element name="policyFinderModule" type="config:ClassType"/>

              </xs:choice>
              <xs:attribute name="name" type="xs:string" use="required"/>
       </xs:complexType>
       <xs:complexType name="AttributeFactoryType">
              <xs:sequence>
                     <xs:element name="datatype" type="config:IdentifiedType" minOccurs="0"
                  maxOccurs="unbounded"/>
              </xs:sequence>
              <xs:attribute name="name" type="xs:string" use="required"/>
              <xs:attribute name="useStandardDatatypes" type="xs:boolean"
                  use="optional" default="true"/>
       </xs:complexType>
       <xs:complexType name="CombiningFactoryType">
              <xs:sequence>
                     <xs:element name="algorithm" type="config:ClassType" minOccurs="0"
                  maxOccurs="unbounded"/>
              </xs:sequence>
              <xs:attribute name="name" type="xs:string" use="required"/>
              <xs:attribute name="useStandardAlgorithms" type="xs:boolean"
                  use="optional" default="true"/>
       </xs:complexType>
       <xs:complexType name="FunctionFactoryType">
              <xs:sequence>
```

Quadro 27 – Continuação especificação *Schema* XML do arquivo de configuração do PDP

```
                    <xs:element name="target" type="config:FunctionCollectionType"
                minOccurs="0"/>
                    <xs:element name="condition" type="config:FunctionCollectionType"
                minOccurs="0"/>                 <xs:element name="general"
type="config:FunctionCollectionType"
                minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="useStandardFunctions" type="xs:boolean"
                use="optional" default="true"/>
    </xs:complexType>
    <xs:complexType name="FunctionCollectionType">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="function" type="config:ClassType"/>
                    <xs:element name="abstractFunction" type="config:IdentifiedType"/>
                    <xs:element name="functionCluster" type="config:ClassType"/>
            </xs:choice>
    </xs:complexType>
    <xs:complexType name="IdentifiedType" mixed="true">
            <xs:complexContent mixed="true">
                    <xs:extension base="config:ClassType">
                            <xs:attribute name="identifier" type="xs:anyURI"
use="required"/>
                    </xs:extension>
            </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ClassType" mixed="true">
            <xs:complexContent mixed="true">
                    <xs:extension base="config:ClassParametersType">
                            <xs:attribute name="class" type="xs:string" use="required"/>
                    </xs:extension>
            </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ClassParametersType" mixed="true">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="string" type="xs:string"/>
                    <xs:element name="list" type="config:ClassParametersType"/>
            </xs:choice>
    </xs:complexType>
</xs:schema>
```

APÊNDICE D – Arquivo de configuração do PDP do *middleware* de teste da extensão XACML

O Quadro 28 apresenta o arquivo de configuração do PDP do *middleware* de teste da extensão XACML. A biblioteca base Sunxacml 1.2 tambem permite configurar o PDP (SUN, 2004b).

Quadro 28 – Arquivo de configuração do PDP *middleware* de teste da extensão XACML

```
<config xmlns="http://sunxacml.sourceforge.net/schema/config-0.3"
        defaultPDP="pdpSample" defaultAttributeFactory="attr"
        defaultCombiningAlgFactory="comb" defaultFunctionFactory="func"
               defaultFiles="filesConfig">
       <confgfiles name="filesConfig">
               <policySchema class="">
                      <string>C:\\Teste_XACML\\policy\\cs-xacml-schema-policy-
01_FED.xsd</string>
               </policySchema>
               <contextSchema class="">
                      <string>C:\\Teste_XACML\\request\\cs-xacml-schema-context-
01_FED.xsd</string>
               </contextSchema>
       </confgfiles>
       <pdp name="pdpSample">
               <attributeFinderModule
class="com.sun.xacml.federation.finder.impl.SelectorModule"/>
               <attributeFinderModule class="xacmlmdw.pip.AttributeFinderFederationFurb"/>
               <attributeFinderModule class="xacmlmdw.pip.AttributeFinderFederationUFRJ"/>
               <policyFinderModule
class="com.sun.xacml.federation.finder.impl.FilePolicyModule">
                      <list>
                             <string>C:\\Teste_XACML\\policy\\SetRoot.xml</string>
                      </list>
               </policyFinderModule>
               <policyFinderModule
class="com.sun.xacml.federation.finder.impl.FilePolicyModuleReference">
                      <list>
                             <string>C:\\Teste_XACML\\policy\\</string>
                      </list>
               </policyFinderModule>
       </pdp>
       <attributeFactory name="attr" useStandardDatatypes="false">
               <datatype class="com.sun.xacml.federation.attr.proxy.BooleanAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#boolean"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.StringAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#string"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.DateAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#date"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.TimeAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#time"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.DateTimeAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#dateTime"/>
               <datatype
class="com.sun.xacml.federation.attr.proxy.DayTimeDurationAttributeProxy"
              identifier="http://www.w3.org/TR/2002/WD-xquery-operators-
20020816#dayTimeDuration"/>
               <datatype
class="com.sun.xacml.federation.attr.proxy.YearMonthDurationAttributeProxy"
              identifier="http://www.w3.org/TR/2002/WD-xquery-operators-
20020816#yearMonthDuration"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.DoubleAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#double"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.IntegerAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#integer"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.AnyURIAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#anyURI"/>
               <datatype class="com.sun.xacml.federation.attr.proxy.HexBinaryAttributeProxy"
              identifier="http://www.w3.org/2001/XMLSchema#hexBinary"/>
```

Quadro 28 – Continuação arquivo de configuração do PDP do *middleware* de teste da extensão XACML

```
                <datatype
class="com.sun.xacml.federation.attr.proxy.Base64BinaryAttributeProxy"
               identifier="http://www.w3.org/2001/XMLSchema#base64binary"/>
                <datatype class="com.sun.xacml.federation.attr.proxy.X500NameAttributeProxy"
               identifier="urn:oasis:names:tc:xacml:1.0:data-type:x500Name"/>
                <datatype class="com.sun.xacml.federation.attr.proxy.RFC822NameAttributeProxy"
               identifier="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
        </attributeFactory>
        <combiningAlgFactory name="comb" useStandardAlgorithms="false">
                <algorithm class="com.sun.xacml.federation.combine.DenyOverridesPolicyAlg"/>
                <algorithm
class="com.sun.xacml.federation.combine.OrderedDenyOverridesPolicyAlg"/>
                <algorithm class="com.sun.xacml.federation.combine.DenyOverridesRuleAlg"/>
                <algorithm
class="com.sun.xacml.federation.combine.OrderedDenyOverridesRuleAlg"/>
                <algorithm class="com.sun.xacml.federation.combine.PermitOverridesPolicyAlg"/>
         <algorithm class="com.sun.xacml.federation.combine.OrderedPermitOverridesPolicyAlg"/>
                <algorithm class="com.sun.xacml.federation.combine.PermitOverridesRuleAlg"/>
                <algorithm
class="com.sun.xacml.federation.combine.OrderedPermitOverridesRuleAlg"/>
                <algorithm class="com.sun.xacml.federation.combine.FirstApplicablePolicyAlg"/>
                <algorithm class="com.sun.xacml.federation.combine.FirstApplicableRuleAlg"/>
                <algorithm
class="com.sun.xacml.federation.combine.OnlyOneApplicablePolicyAlg"/>
        </combiningAlgFactory>
        <functionFactory name="func" useStandardFunctions="false">
                <target>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.EqualFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.LogicalFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.NOfFunctionCluster"/>

                        <!-- there is a cluster for this function, but a single function -->
                        <!-- is used instead just as an example of how this works -->
                        <function class="com.sun.xacml.federation.cond.NotFunction">
                                <string>urn:oasis:names:tc:xacml:1.0:function:not</string>
                        </function>

                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.ComparisonFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.MatchFunctionCluster"/>
                </target>
                <condition>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.ConditionBagFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.ConditionSetFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.HigherOrderFunctionCluster"/>
                </condition>

                <general>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.AddFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.SubtractFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.MultiplyFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.DivideFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.ModFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.AbsFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.RoundFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.FloorFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.DateMathFunctionCluster"/>
```

Quadro 28 – Continuação arquivo de configuração do PDP *middleware* de teste da extensão XACML

```
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.GeneralBagFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.NumericConvertFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.StringNormalizeFunctionCluster"/>
                        <functionCluster
class="com.sun.xacml.federation.cond.cluster.GeneralSetFunctionCluster"/>
                        <abstractFunction class="com.sun.xacml.federation.cond.MapFunctionProxy"
                          identifier="urn:oasis:names:tc:xacml:1.0:function:map"/>
                </general>
        </functionFactory>
</config>
```