Ramon Thiago de Campos Facchin

Um Sistema de Emissão de Atestados Médicos com Controle de Acesso baseado em Certificados de Atributo do Modular PERMIS

Santa Catarina, Brasil

Florianópolis, 1 de Novembro de 2013

Ramon Thiago de Campos Facchin

Um Sistema de Emissão de Atestados Médicos com Controle de Acesso baseado em Certificados de Atributo do Modular PERMIS

Trabalho da área de Segurança da Informação que propõe um protótipo de sistema para emissão de atestados médicos eletrônicos, como prova de conceito para o controle de acesso baseado no uso de Certificados de Atributo.

Universidade Federal de Santa Catarina – UFSC

Centro Tecnológico

Departamento de Informática e Estatística

Bacharelado em Sistemas de Informação

Orientador: João Bosco Mangueira Sobral

Santa Catarina, Brasil

Florianópolis, 1 de Novembro de 2013

Ramon Thiago de Campos Facchin
Um Sistema de Emissão de Atestados Médicos com Controle de Acesso baseado em Certificados de Atributo do Modular PERMIS/ Ramon Thiago de Campos Facchin. – Santa Catarina, Brasil, Florianópolis, 1 de Novembro de 2013-
117 p. : il. (algumas color.) ; 30 cm.

Orientador: João Bosco Mangueira Sobral

Trabalho de Conclusão de Curso – Universidade Federal de Santa Catarina – UFSC
Centro Tecnológico
Departamento de Informática e Estatística
Bacharelado em Sistemas de Informação, Florianópolis, 1 de Novembro de 2013.

1. certificado. 2. atributo. 3. segurança. 4. sistema. 5. atestado. 6. médico. I. João Bosco Mangueira Sobral. II. Universidade Federal de Santa Catarina. III. Bacharelado em Sistemas de Informação. IV. Um Sistema de Emissão de Atestados Médicos com Controle de Acesso baseado em Certificados de Atributo do Modular PERMIS

CDU 02:141:005.7

Ramon Thiago de Campos Facchin

Um Sistema de Emissão de Atestados Médicos com Controle de Acesso baseado em Certificados de Atributo do Modular PERMIS

Trabalho da área de Segurança da Informação que propõe um protótipo de sistema para emissão de atestados médicos eletrônicos, como prova de conceito para o controle de acesso baseado no uso de Certificados de Atributo.

Trabalho aprovado. Santa Catarina, Brasil, 24 de novembro de 2012:

João Bosco Mangueira Sobral
Orientador

Frank Augusto Siqueira
Universidade Federal de Santa Catarina

Darlan Vivian
BRy Tecnologia

Luiz Carlos Zancanella
Safeweb

Santa Catarina, Brasil
Florianópolis, 1 de Novembro de 2013

Este trabalho é dedicado aos amigos que surgiram (e aos que sumiram também), durante a minha trajetória acadêmica e que proporcionaram os momentos de distração para desopilar das responsabilidades da universidade e do trabalho.

À minha família, que sempre me ofereceu um lar, e segurança, não apenas um teto.

À minha namorada, que me deu um estímulo fundamental nas etapas finais da minha graduação.

Dedico também ao Heavy Metal, pois sem trilha sonora a vida não tem graça e o trabalho não prossegue. Raise your fist to the sky!!!

“A frase mais emocionante de se ouvir na ciência, a que apresenta novas descobertas, não é "Eureka!", e sim "Isso é divertido..."(Isaac Asimov)

Resumo

A crescente difusão do modelo de distribuição e comercialização de Software conhecido como SaaS (ou "Software como um Serviço") trouxe, consigo, uma necessidade de adaptação dos mecanismos de segurança de sistemas para o uso de modelos que proporcionam interoperabilidade e maior nível de abstração na implementação do controle de acesso. Neste contexto, o presente trabalho apresenta um estudo dos Certificados de Atributo X.509 e da Infraestrutura de Gerenciamento de Privilégios como uma solução para o controle de acesso de forma interoperável, através do desenvolvimento de um protótipo de aplicação de emissão eletrônica de Atestados Médicos, oferecendo uma conveniência para usuários dos sistemas de saúde apresentarem atestados médicos sem a necessidade de possuir, em mãos, a via original, ou de conhecimento prévio da quantidade de vias necessárias do documento para seu uso.

Palavras-chaves: certificado de atributo. infraestrutura de gerenciamento de privilégios. autorização. controle de acesso. atestado médico.

Abstract

The increasing spread of the software business model known as SaaS (Software as a Service) brought the need to change systems' security engines for using more interoperable models and providing increased abstraction levels for implementing access control. In this scenario, this work presents a study of X.509 Attribute Certificates and of Privilege Management Infrastructure as solutions for enabling interoperable access control, through the development of an application prototype for issuing electronical medical certificates, offering a convenience for healthcare system users to show their medical certificates having the original version of the document, and without prior knowledge of the amount of copies they would need.

Palavras-chaves: attribute certificate. privilege management infrastructure. authorization. access control. medical certificate.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SaaS	Software como Serviço (*Software as a Service*)
IGP	Infraestrutura de Gerenciamento de Privilégios
ICP	Infraestrutura de Chave Pública
AC	Autoridade Certificadora
AR	Autoridade Certificadora Raiz
CA	Certificado de Atributo Digital
CCP	Certificado Digital de Chave Pública
CABP	Controle de Acesso Baseado em Papéis
SoA	*Source of Authority*
AA	Autoridade de Atributo
Java EE6	*Java Enterprise Edition 6*
CDI	*Contexts and Dependency Injection*
EJB	*Enterprise Java Bean*
JPA	*Java Persistence API*
LCR	Lista de Certificados Revogados
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
JSR	*Java Specification Recommendation*
JCP	*Java Community Process*
API	*Application Programming Interface*
DAO	*Data Access Object* (padrão de projeto)
CoC	*Convention over Configuration*
POJO	*Plain Old Java Object*

DTO	*Data Transfer Object*
DAO	*Data Access Object*
JNDI	*Java Naming and Directory Interface*
LDAP	*Lightweight Directory Access Protocol*
XML	*eXtensible Markup Language*
URL	*Uniform Resource Locator*

Sumário

1 Introdução

1.1 Contextualização

Com a popularização do acesso à Internet houve a necessidade de modernização dos Sistemas de Informação. Da distribuição e comercialização de softwares aplicativos que executavam sobre um sistema operacional, passou-se ao uso de softwares através de navegadores de internet, consolidando a Web 2.0. Hoje, um modelo de distribuição e comercialização de software cada vez mais difundido é o modelo SaaS, que traz, ao usuário final, a vantagem de não ter que se preocupar com os requisitos de hardware ou de software necessários para a execução de uma aplicação, exceto pelo fato de que deve-se possuir um navegador e conexão com a internet.

No entanto, as aplicações para Web também trouxeram um acréscimo à complexidade na implementação de mecanismos de segurança em sistemas, devido ao aumento do universo de usuários ao qual uma instância da aplicação está disponível.

A abordagem mais comum para providenciar maior segurança aos sistemas é a de desenvolver-se um mecanismo de segurança específico para cada aplicação. No entanto, essa abordagem apresenta diversas limitações, como a sobrecarga de confiança e de atribuições do Administrador do Sistema, além de, em determinados sistemas, ser questionável se o Administrador do Sistema é o agente mais adequado para atribuir determinadas permissões aos demais usuários. Outro aspecto relevante é a baixa interoperabilidade de se implementar um mecanismo de segurança específico para cada aplicação, como nas vezes em que o domínio de segurança de duas aplicações Web é o mesmo, por exemplo. Nesse caso, ou haverá retrabalho, ou haverá um esforço equivalente na compreensão do domínio de segurança de uma das aplicações com a finalidade de aplicá-lo à outra aplicação.

É nesse sentido que foi desenvolvido o Certificado de Atributo Digital. Com a finalidade de resolver problemas relativos à interoperabilidade dos mecanismos de segurança de sistemas, levando em conta o aspecto da Autorização, com a vantagem de fazê-lo através de uma cadeia de confiança.

O problema a ser tratado no trabalho é o da emissão de atestados médicos através de um sistema de informação, mas com um controle de acesso feito de forma que os papéis de cada usuário possam ser emitidos pelas entidades que possuem a prerrogativa de fazer a atribuição dos papéis. Por exemplo: um usuário só pode emitir um atestado médico, se este usuário for, de fato, um médico - ou seja, se tiver, entre seus papéis, o de médico. Para auxiliar o controle de acesso, devem ser utilizados certificados de atributo que definam os papéis de cada usuário dentro do sistema. A conformidade do mecanismo de controle

de acesso com as restrições que serão propostas deve ser verificada para as diferentes páginas do sistema, autorizando apenas os usuários que possuem os papéis necessários para acessar tais páginas.

1.2 Objetivo Geral

Com base no cenário descrito anteriormente, configura-se o objetivo geral deste trabalho, que é o de estudar a tecnologia de Certificados de Atributo e aplicá-la na implementação de um protótipo de sistema de emissão eletrônica de atestados médicos, como uma prova de conceito do uso de Certificado de Atributo Digital como instrumento de controle de acesso a recursos e funcionalidades de uma aplicação Web.

1.3 Objetivos Específicos

- Compreender o funcionamento dos Certificados de Atributo e o conceito de Infraestrutura de Gerenciamento de Privilégios.
- Verificar a aplicabilidade do Certificado de Atributo como instrumento central num mecanismo de Autorização.
- Especificar um mecanismo de Autorização baseado em uma Infraestrutura de Gerenciamento de Privilégios.
- Propor um protótipo de aplicação para possibilitar a emissão eletrônica de atestados médicos que aplique o mecanismo de autorização proposto.
- Disponibilizar uma implementação de referência para desenvolvedores interessados em utilizar CAs como solução para mecanismos de autorização em suas aplicações.
- Estudar as tecnologias que auxiliarão no processo de desenvolvimento da aplicação proposta.
- Implementar a aplicação.

1.4 Metodologia

O presente trabalho começa pelo estudo de conceitos básicos de segurança da informação relacionados diretamente com controle de acesso e com certificados de atributo.

Em posse do conhecimento adquirido, foram selecionadas tecnologias e bibliotecas que pudessem auxiliar no processo de desenvolvimento de uma aplicação que servisse como prova de conceito do uso de certificados de atributo como instrumentos de controle

de acesso, bem como tecnologias e bibliotecas que ajudassem a diminuir o tempo de desenvolvimento da aplicação, levando em conta, principalmente, a familiaridade do autor com tais ferramentas.

Após feita a apresentação das ferramentas utilizadas, passou-se por uma etapa de especificação da aplicação a ser desenvolvida para que se tenha clareza do cenário no qual será aplicada a prova de conceito proposta. Posteriormente, apresentando a aplicação desenvolvida e explicando como se deu o uso de IGP dentro da aplicação. Propondo, por fim, uma implementação de referência para desenvolvedores interessados no uso de Certificados de Atributo como ferramentas centrais em mecanismos de autorização, apresentando os resultados obtidos.

1.5 Estrutura do Documento

O Documento está estruturado da seguinte forma.

No Capitulo 2 serão apresentados conceitos fundamentais relacionados a Autenticação e Autorização para a aplicação proposta, terminando por detalhar o conceito de Certificado de Atributo, bem como o de Infraestrutura de Gerenciamento de Privilégios.

O Capitulo 3 discorre sobre as tecnologias escolhidas para o desenvolvimento da aplicação proposta, incluindo a ferramenta escolhida para o controle de acesso com base em certificados de atributo.

O Capitulo 4 aborda a especificação da aplicação proposta, delimitando o escopo da aplicação, levantando brevemente os requisitos e apresentando uma visão arquitetural da aplicação e alguns diagramas.

O Capítulo 5 apresenta a aplicação desenvolvida e como se dá seu uso dentro do contexto proposto para que, no Capítulo 6 sejam apresentados os resultados obtidos. No capítulo 7 são feitas as considerações finais do trabalho, bem como sugestões de trabalhos futuros.

2 Conceitos de Segurança de Aplicações

Ao se tratar de segurança da informação, trata-se de proteção de recursos, ou seja, a proteção do valor da informação, presente em um sistema, para uma ou mais pessoas ou organizações. Dentro desse contexto, tratar do Controle de Acesso é imprescindível.

Entenda-se, por "Controle de Acesso", um processo através do qual o acesso a recursos de sistemas é regido por uma política de segurança e é concedido apenas a entidades autorizadas (usuários, programas, processos ou outros sistemas) de acordo com tal política. Ou ainda, as limitações nas interações entre sujeitos e objetos em um sistema de informação (SHIREY, 2007).

Podemos dar três problemas diferentes para a implementação de Controle de Acesso em sistemas: Autenticação, Autorização (embora Sandhu e Samarati (1996) utilizem, em seu trabalho, o termo "Controle de Acesso"referindo-se a Autorização) e Auditoria.

2.1 Autenticação

A autenticação, pode-se dizer, é a atribuição de uma identidade a um agente (usuário/sistema/computador que executa operações) perante um determinado sistema (SANDHU; SAMARATI, 1996). Ou ainda, diz se os agentes envolvidos na interação com um sistema são quem dizem ser. Segundo o Federal Financial Institutions Examination Council (2008), a autenticação deve levar em consideração 3 Fatores:

- Fator de propriedade (ou o que o usuário possui): telefone, documento de identidade, Cadastro de Pessoa Física, pulseira de identificação etc.
- Fator de conhecimento (ou o que o usuário sabe): senha, chave, número de identificação pessoal (PIN), resposta de desafio etc.
- Fator de identidade (ou o que o usuário é): padrão de retina, impressão digital, assinatura, voz, sequência de DNA, outros identificadores biométricos etc.

Com frequência são utilizados no minimo dois, preferencialmente três, dos fatores supracitados para autenticar usuários. Alguns exemplos comuns de procedimentos de autenticação para se identificar em um sistema computacional são o uso de nome de usuário e senha, dispositivos biométricos, smartcards e certificados digitais. Este trabalho se utilizará do método de autenticação baseado em nome de usuário e senha por conta de sua ampla difusão e grande disponibilidade de bibliotecas auxiliares conhecidas e difundidas

no mercado. O método escolhido contempla os fatores de propriedade (nome de usuário) e de conhecimento (senha).

2.2 Auditoria

Auditoria é o processo de registrar as ações de um usuário em um sistema, fornecendo informações sobre o contexto de acesso a um recurso ou funcionalidade daquele sistema, incluindo informações do próprio usuário.

Usuários são os responsáveis pelo seu uso de um sistema de informação e podem ser autorizados a acessar um recusro. Se o acesso for concedido, deve ser criada uma trilha de auditoria que fornece dados históricos de quando e como o usuário acessou tal recurso. Ou mesmo que não tenha sido autorizado, uma trilha de auditoria é recomendável para que se tenha o registro da tentativa de violação da autorização do sistema (TODOROV, 2008).

A interpretação das informações de auditoria é útil na tomada de medidas a respeito de ações já realizadas no sistema, ou mesmo na evolução do próprio sistema, melhorando seu mecanismo de detecção de intrusão ou identificando possíveis falhas na política de segurança.

2.3 Autorização

A autorização, por outro lado, define a capacidade de um sistema permitir o acesso a objetos, comportamentos e funcionalidades de um sistema para um agente já autenticado (SANDHU; SAMARATI, 1994). Isto é, trata-se do mecanismo pelo qual se verifica se um agente possui privilégios para realizar o que deseja no momento da requisição.

O principal objetivo, ao se implementar mecanismos de autorização em um sistema de informação, é o de prevenir ações de usuários que possam levar a uma violação de segurança. É, por definição, uma abordagem preventiva de segurança (em contraposição à Auditoria, por exemplo, que é uma abordagem reativa).

Em se tratando de sistemas de saúde, por exemplo, deve-se garantir que cada agente possa realizar apenas as operações, no sistema, que cabe a sua categoria de usuário. No contexto proposto por este trabalho, o de emissão de atestados médicos, isso significa permitir apenas que os profissionais de saúde com permissão legal para emitir esse tipo de documento possam emiti-los através do sistema. Significa, também, que apenas o dono do recurso (por definição, é a pessoa a quem aquele recurso pertence), o paciente, terá acesso àquele recurso e lhe serão dados poderes para estender esse acesso a outros agentes.

Existem diversas abordagens para tratar da autorização em sistemas computaci-

onais. Para o presente trabalho, será utilizado o Controle de Acesso Baseado em Papéis (CABP).

2.3.1 Controle de Acesso Baseado em Papéis

A primeira idéia de um Controle de Acesso Baseado em Papéis foi apresentada por Ferraiolo e Kuhn (1992). Consiste na definição de papéis (roles) para categorizar os diferentes perfis de usuários que utilizam uma aplicação. O papel, em si, é um rótulo, ou termo "guarda-chuva", para abstrair o conjunto de funções que um determinado sujeito desempenha dentro de uma organização, seja no mundo real ou em um sistema de informação (SANDHU; FERRAIOLO; KUHN, 2000).

Trata-se, portanto, de uma abstração do funcionamento de organizações no mundo real com o objetivo de simplificar o gerenciamento de permissões dentro de um sistema, de forma que o alvo das definições de permissões deixe de ser o usuário em si, e passe a ser o papel para ele definido no sistema, ajudando a impessoalizar a política de segurança.

Custódio (2010) explica, em seu trabalho, que o CABP é uma abordagem que atende aos requisitos de controle de acesso a sistemas de saúde sendo, inclusive, recomendado pelo Health Insurance Portability and Accountability Act of 1996 (DEPARTMENT OF HEALTH AND HUMAN SERVICES, 1998). Sendo, portanto, adequada sua adoção para o desenvolvimento deste trabalho.

Para a implementação do CABP dentro do protótipo de aplicação a ser desenvolvido, foi escolhida a ferramenta PERMIS, que será apresentada nos capítulos 2 e 3. O PERMIS oferece uma infraestrutura completa para gerenciamento de privilégios e mecanismos de autorização e, entre suas funcionalidades, está a de auxiliar o controle de acesso utilizando-se de Certificados de Atributo (CA). CAs são uma categoria de Certificado Digital de estrutura similar à do Certificado de Chave Pública (CCP), mas com um propósito completamente diferente: o de atribuir uma permissão ou qualidade, com temporalidade variável, a um sujeito, enquanto que CCPs buscam associar um sujeito (uma identidade) a uma Chave Pública. Os CAs também seguem o padrão *X.509* e sua notação é *ASN.1*, assim como os CCP.

que se assemelham à idéia de Certificado Digital de Chave Pública, mas com um propósito diferente, conforme será visto adiante.

2.4 Criptografia

A Criptografia é a ciência (e arte) de manter mensagens seguras (SCHNEIER, 1996). Isto é, garantir a segurança de mensagens entre o emissor e o receptor, geralmente no sentido de manter a confidencialidade da mensagem.

No contexto da Computação e da Segurança da Informação, entende-se a Criptografia como uma técnica de processamento de um conjunto qualquer de dados, através de um algoritmo de cifragem, para resultar em um outro conjunto de dados, ininteligível enquanto não for executado o seu algoritmo inverso (de decifragem).

Criptografia é, portanto, um conceito essencial para a proteção de dados. Ao longo da história da humanidade há exemplos de uso de criptografia e, por muito tempo, o segredo do algoritmo de cifragem foi o modo encontrado de garantir a segurança de mensagens.

Para a computação, no entanto, não é um bom negócio se apoiar no sigilo do algoritmo para garantir a segurança de mensagens, já que, uma vez descoberto, quaisquer mensagens cifradas por aquele algoritmo que forem trocadas estarão vulneráveis. Por isso, um nível de segurança adicional é conferido aos algoritmos de cifragem se for feito o uso de uma chave. Dessa forma, para a cifragem de um conjunto de dados, não mais apenas os dados são suficientes, mas o fornecimento de uma chave, essa sim, mantida em segredo, para realizar a cifragem. Assim como na decifragem, também são necessários os dados cifrados e uma chave.

O uso de chaves trouxe uma divisão no conceito de criptografia. Em termos gerais, quando se utiliza uma mesma chave para a cifragem e para a decifragem, trata-se de *Criptografia Simétrica*, enquanto que, quando a chave utilizada para a cifragem e uma chave diferente é utilizada para a decifragem, trata-se de *Criptografia Assimétrica*.

2.4.1 Criptografia Assimétrica

Conforme citado anteriormente, a Criptografia Assimétrica consiste no uso de um par de chaves para cifragem e decifragem. Enquanto uma chave é utilizada para realizar a função de cifragem, a outra é utilizada para a decifragem.

Diffie e Hellman (1976) propuseram, pela primeira vez, o uso de um par de chaves para a troca de dados, com uma relação que permite que, dado um par de chaves, A e B, a cifragem de dados com A possa ser decifrada com B, e a cifragem de dados com B possa ser decifrada com A, desde que esse par de chaves tenha sido gerado de maneira adequada. Isso permitiu a troca de mensagens de maneira segura de uma forma que os problemas com gerenciamento de chaves fosse minimizado, desde que uma dessas chaves fosse a de conhecimento privado de um agente (chave privada), enquanto que a outra fosse de conhecimento público (chave pública).

Respeitando o conceito de chave pública e de chave privada, quando uma mensagem é cifrada com a chave pública de um agente X, somente a chave privada daquele agente é capaz de decifrar a mensagem, garantindo a confidencialidade da mensagem, uma vez que somente quem tem o conhecimento da chave privada, necessária para a de-

cifragem, é o próprio agente *X*. No outro sentido, cifrando uma mensagem com a chave privada do agente *X*, garante-se a autenticidade, uma vez que a mensagem só poderá ser decifrada com a chave pública do agente *X*. Esse segundo caso representa a base da Assinatura Digital.

2.5 Assinatura Digital

A Assinatura Digital é, resumidamente, um conjunto de técnicas empregado para demonstrar a autenticidade de uma mensagem ou documento eletrônico. Uma assinatura digital válida fornece, ao receptor de uma mensagem, razão suficiente para que ele confie que tal mensagem foi enviada por um remetente conhecido, sendo que aquele remetente não pode negar que a mensagem foi enviada (não-repúdio) nem afirmar que a mensagem foi alterada antes do recebimento. A criptografia assimétrica é uma das técnicas empregadas no processo de Assinatura Digital utilizando-se de uma Infraestrutura de Chave Pública (ICP).

Um outro problema em se tratando de Assinatura Digital com ICP está relacionado ao tamanho das mensagens assinadas. Embora esse procedimento funcione bem para mensagens curtas, por vezes é necessária a cifragem de dados muito grandes, da casa de Gigabytes, fazendo com que o processo de cifragem seja demorado e oneroso. Para isso, em vez de se assinar os dados em sua forma original, a assinatura é realizada sobre um resumo daqueles dados. Esse resumo é obtido aplicando-se, sobre os dados, uma função de resumo, denominada *Hash*.

Um algoritmo de Hash consiste em uma função que, toda vez que aplicada a uma mensagem *m*, resultará em um mesmo resumo *h(m)* de tamanho fixo. Algumas características da função Hash:

- Um algoritmo de Hash, quando aplicado a qualquer mensagem, resultará sempre num conjunto de bytes de tamanho fixo.
- Um algoritmo de Hash apresentará sempre o mesmo resultado do resumo de uma mensagem, se essa mensagem não for modificada. Ou seja, para uma mesma mensagem *m* sempre haverá apenas um resultado *m'* da execução da função de hash *h(m) = m'*.
- Diferentes mensagens podem, eventualmente, resultar em um mesmo Hash. Esse fenômeno é chamado de colisão, e a qualidade do algoritmo de Hash é inversamente proporcional ao número de colisões que podem ser ocasionadas do seu uso.
- No geral, é praticamente nula a probabilidade de uma mensagem M1 e uma mensagem M2 gerarem um mesmo Hash e ambas fazerem sentido.

Dessa forma, com as características observadas, o processo de Assinatura Digital ocorre sempre sobre um resumo das mensagens a serem assinadas, e não sobre as mensagens em si, desonerando o processo de cifragem e garantindo o conteúdo. Portanto, uma Assinatura Digital consiste em uma estrutura de dados que contém, entre outras informações, o resultado da cifragem do resumo da mensagem "assinada". E, ainda, pode-se dizer que toda verificação de assinatura digital consiste na comparação do resumo da mensagem recebida com o resumo assinado digitalmente, que representa a mensagem original.

2.6 Certificado Digital de Chave Pública

No mesmo ano de proposição do algoritmo RSA, Kohnfelder (1978) elaborou, em seu trabalho, a primeira proposta de associação da identidade de um par de chaves a um usuário de uma maneira formal. Surgia o conceito de Certificado Digital de Chave Pública, assim como uma série de conceitos relacionados.

Burnett e Payne (2002) definem os certificados digitais como um meio seguro de distribuição de chaves públicas, dentro de uma rede, para partes verificadoras - de forma análoga a um documento de identidade.

O CCP é um documento digital de formato padronizado e que contém informações que são suficientes para associar uma chave pública a uma identidade. Além da própria chave pública, num CCP constam informações como a data de validade do certificado, data de emissão, nome do portador, assinatura, número de série, dados da entidade emissora do certificado etc. Para que a validade de um certificado esteja assegurada, é necessário que o mesmo esteja assinado digitalmente, isto é, há a necessidade de que alguém garanta as informações que ali constam. Para o uso e aceitação do certificado, é necessário que se confie no agente que assinou aquele certificado que, dependendo do caso, pode ser auto-assinado. Para regulamentar essas situações e definir precisamente cada agente envolvido, foi proposto o conceito de Infraestrutura de Chave Pública (ICP).

2.6.1 Infraestrutura de Chave Pública

A ICP foi proposta com o objetivo de simplificar o gerenciamento de CCPs com base em uma Cadeia de Confiança. Ou seja, se é certo que, para associar uma chave pública a um indivíduo é necessário um CCP, e para que esse CCP seja aceito é necessário confiar em quem o assinou digitalmente, como definir quem são os agentes confiáveis e quem não são, nesse processo?

Para isso, uma ICP define dois níveis diferentes de Autoridades que podem emitir CCPs. Em primeiro lugar, Autoridades Certificadoras (ACs), que são as responsáveis por emitir, distribuir e gerenciar o estado de CCPs de usuários finais. Isto é, uma AC possui seu próprio CCP, com o qual realiza a assinatura de CCPs emitidos para usuários finais.

Portanto, dentro de uma ICP, confia-se em um CCP se também há a confiança na AC que o emitiu. Da mesma forma, confiar na AC envolve confiar na Autoridade que emitiu o CCP daquela AC - e esse processo pode se propagar por diversos níveis, seguindo uma estrutura em árvore. Se pensarmos em um CCP como análogo a uma Carteira de Identidade, a AC seria o equivalente ao órgão expedidor.

No momento em que se atinge a raiz dessa árvore, o nó raiz chama-se Autoridade Certificadora Raiz (AR), cujo certificado é autoassinado. Confiar em uma AR implica confiar em todas as ACs cujos certificados foram emitidos por aquela AR, dessa forma, estabelecendo-se a Cadeia de Confiança, resumida na Figura 1.

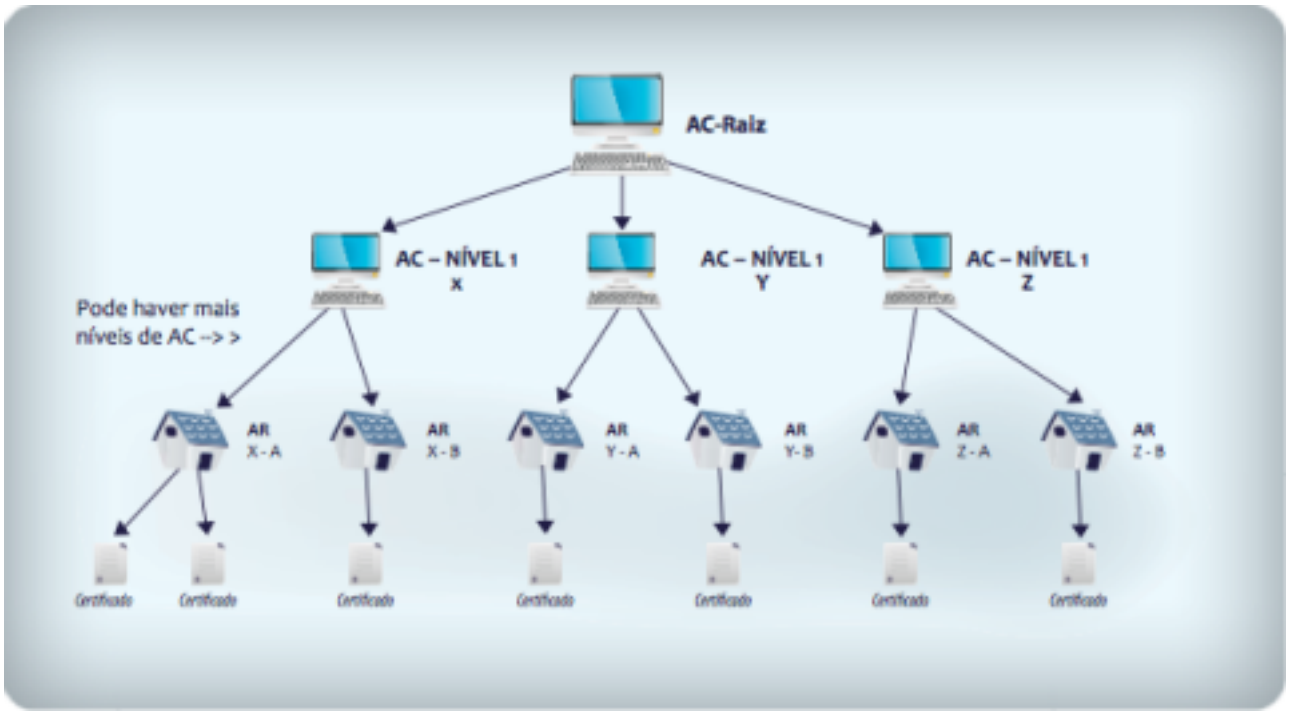

Figura 1 – Modelo de Infraestrutura de Chave Pública (Retirado de Macêdo (2012))

A existência de uma ICP e a associação de uma chave pública a uma identidade tornam o CCP um instrumento excelente para a autenticação de usuários de uma forma segura e interoperável, uma vez que dados relativos à identidade de um usuário são, em geral, imutáveis.

2.7 Certificado de Atributo

No entanto, para os mecanismos de autorização, um CCP não é uma boa solução. Pensando em um contexto em que se utilize CABP, uma solução possível, seria a de criar um campo adicional no CCP contendo os papéis associados àquele indivíduo (o portador do certificado). Por que essa não é uma solução interessante? A temporalidade de um atributo (no caso, um papel) de um indivíduo e a temporalidade de sua identidade

possuem naturezas diferentes. Enquanto um certificado de chave pública costuma ter um período de validade de 1 ano ou mais, pode ser que um atributo tenha sua validade programada para um único uso, ou para um período muito mais curto, por exemplo (BEZERRA, 2011).

Para resolver esse problema, Farrel e Housley (2002) propuseram o uso de Certificados de Atributo para guardar informações relativas à Autorização de um indivíduo. Enquanto um Certificado de Chave Pública procura vincular uma Chave Pública a um indivíduo ou uma identidade, um Certificado de Atributo, embora possua uma estrutura semelhante, conta com uma diferença essencial: a não existência de um campo de Chave Pública. Em vez disso, é definida uma lista de atributos que podem especificar uma relação de pertinência a um grupo, um papel, uma definição de permissão ou outras informações de autorização associadas ao indivíduo para quem aquele Certificado de Atributo foi emitido (FARREL; HOUSLEY; TURNER, 2010). Ou ainda, pode-se dizer que um Certificado de Atributo é um documento eletrônico, assinado digitalmente, que apresenta qualidades associadas a uma determinada entidade (CERTFORUM, 2007). Farrel e Housley (2002) exemplificam bem isso, comparando um passaporte com um visto de entrada: o passaporte seria a representação de um CCP, pois identifica a pessoa portadora do documento, que possui uma validade muito grande. Já o visto de entrada seria a representação de um CA, por estar referenciando o passaporte e por possuir um período de validade bem mais curto, o que reforça a questão da temporalidade.

Em um primeiro momento, pode haver confusão entre CA e CCP pelo fato de ambos seguirem o padrão X.509, tornando o termo *Certificado X.509* ambíguo.

Para tornar a diferença mais evidente, então, é possível fazer uma analogia: enquanto o CCP representaria um documento como uma carteira de identidade ou CPF, um CA funciona como uma carteira de motorista. A carteira de motorista está sempre vinculada à identidade do seu proprietário (consta, por exemplo, entre suas informações, o CPF para o qual ela é aplicada), e define uma permissão - no caso, para conduzir veículos automotores. Enumerando algumas semelhanças entre CCPs e CAs, encontramos:

1. O fato de ambos serem assinados digitalmente.

2. Possuem um período de validade.

3. São verificáveis através de Listas de Certificados Revogados.

4. Aderem ao padrão X.509.

A Figura 2 apresenta uma visão geral dos campos existentes em um Certificado de Atributo.

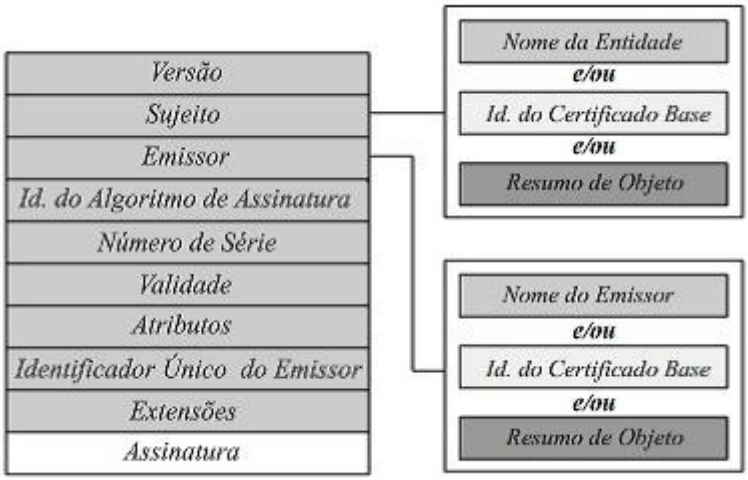

Figura 2 – Visão Geral de um CA (Retirado de Macêdo (2012))

Outro argumento forte contra o uso de campos adicionais em CCPs para definir informações de autorização é o de que, dificilmente, a AC é a melhor entidade para definir informações de autorização de um usuário. Se a fosse, isso demandaria um aumento na burocracia do processo de emissão de certificados, uma vez que, dependendo da natureza do atributo certificado, tornaria necessários procedimentos adicionais para validar o atributo, como consultar entidades de classe, como conselhos regionais, para validar a permissão para exercer determinada profissão, por exemplo.

2.7.1 Aplicações de um CA

As aplicações de um CA podem ser as mais variadas, mas todas envolvendo a definição de situação ou de qualidades do portador do certificado, com a finalidade de conceder ou não acesso a um recurso por parte do portador. Dentro do contexto de sistemas de saúde, por exemplo, o certificado de atributo poderia representar o CRM de um Médico, atributo que lhe concede a permissão para exercer a profissão de médico. Ou ainda, um dos atributos definidos por um CA do médico, seriam as suas especialidades - o que restringiria seus campos de atuação dentro da medicina. Do ponto de vista do paciente, o atestado médico do paciente também poderia ser representado por um certificado de atributo, que definiria sua condição de atestado por um determinado período.

Os CAs podem ser utilizados nos mais variados contextos e aplicações e, inclusive, podem fornecer ou restringir acesso a ambientes físicos (desde que existente todo o aparato eletrônico para fazer a checagem desses certificados, evidentemente).

Portanto, embora o trabalho apresentado se restrinja o uso de certificado de atributo como um documento para definir papéis dentro de um sistema, há inúmeras outras aplicações para os CAs. Uma coisa é certa: em todas, o propósito do CA será o de definir

um estado ou característica do seu portador, com a finalidade de conceder-lhe um acesso a recurso ou um benefício, verificáveis eletronicamente.

2.7.2 Estrutura de um CA

A seguir, são descritos os campos mais importantes de um CA, fornecendo mais informações a respeito do conteúdo de um CA do que consta na Figura 2, dessa vez, dentro do contexto da ICP-Brasil:

- **Emissor (*issuer*)**: é a entidade responsável pelas informações de autorização descritas pelo certificado de atributo. Seu conteúdo deve ter o nome da Entidade Emissora do Atributo e o CNPJ associado.

- **Titular do CA (*holder*)**: é a entidade à qual o atributo, ou conjunto de atributos, se refere. Deve conter informação suficiente para identificar univocamente a entidade titular do certificado. Exemplos de valores para este campo são CPF/CNPJ ou matrícula da própria entidade emissora que permita identificar aquele titular.

- **Período de Validade (*attCertValidityPeriod*)**: o intervalo de tempo para o qual o CA é válido.

- **Número de Série (*serialNumber*)**: todo certificado deve ter um número de série que permita, à entidade responsável pela sua emissão, a identificação do certificado através do fornecimento de um único campo de identificação.

- **Versão (*version*)**: conforme a RFC5755 (FARREL; HOUSLEY; TURNER, 2010), este campo deve ser preenchido com o valor "v2".

- **Algoritmo de Assinatura (*signature*)**: define qual o algoritmo utilizado pela autoridade de atributo para assinar o presente CA.

- **Assinatura da Autoridade de Atributo (*SignatureValue*)**: conteúdo da assinatura digital deste certificado, feita com o certificado de chave pública da Autoridade de Atributo reponsável pela emissão do certificado e, assim, conferindo-lhe a autenticidade e validade jurídica.

- **Atributos (*attributes*)**: campo "razão de ser"do CA. Nele são descritas as qualificações (estados / características) do titular do certificado. Este campo pode servir a vários propósitos, como identificação de acesso, incumbência, níveis de acesso, funções, grupos do titular etc.

- **Extensões (*extensions*)**: campo que possui informações complementares que permitem a verificação do certificado de atributo como, por exemplo, a lista de CAs

revogados e condições de revogação do certificado antes do término do seu intervalo de validade.

- **Tipo de Certificado**: este campo é uma especificação da própria ICP-Brasil define se o certificado de atributo é Autônomo ou Vinculado. Um certificado de atributo autônomo, é um CA que não depende da existência de um certificado de chave pública correspondente para definir um atributo para uma entidade, enquanto que um certificado de atributo vinculado é um certificado de atributo que referencia um CCP para que seja possível identificar o titular.

A notação ASN.1 que representa a estrutura descrita anteriormente pode ser verificada no Anexo A com maiores detalhes. Mais detalhes sobre o perfil de emissão de certificados de atributos podem ser obtidos em ICP-Brasil (2012b), ICP-Brasil (2012a), documentos que regulamentam, no Brasil, o uso e emissão de Certificados de Atributo.

2.7.3 Infraestrutura de Gerenciamento de Privilégios

Da mesma forma que existe uma ICP para regulamentar a emissão, distribuição e controle de CCPs, existe a Infraestrutura de Gerenciamento de Privilégios para regulamentar emissão, distribuição e controle de CAs.

Custódio (2010) define uma Infraestrutura de Gerenciamento de Privilégios (IGP), ou em inglês *Privilege Management Infrastructure*, como um sistema capaz de fornecer, para sistemas que necessitem de prover controle de acesso a dados importantes e prover mecanismos de delegação de acesso a esses mesmos dados, o gerenciamento de privilégios para usuários.

Diferentemente de CCPs, os CAs são emitidos por Autoridades de Atributo (AA). AAs são responsáveis pela manutenção, revogação, distribuição e emissão de CAs e possuem um CCP próprio cujo propósito é o de realizar a assinatura digital de CAs.

A Autoridade de Atributo tem o papel de regulador de um atributo que gerencia. O fato de um CCP ser o certificado da Autoridade de Atributo faz com que qualquer órgão responsável por regulamentar determinado atributo seja capaz de emitir certificados sem precisar de toda a complexidade envolvida em uma ICP.

Nesse contexto, podemos comparar uma Autoridade de Atributo a um órgão regulador de entidade (ou classe) como, por exemplo, um Conselho Regional de Medicina, responsável por manter os registros de médicos autorizados a exercer a profissão. Nesse caso, o atributo gerenciado seria a profissão de médico, assim como poderiam ser outros exemplos de atributos o número de registro no CRM do médico, ou sua especialidade. Dessa forma, possibilitando-se que o administrador do atributo do usuário não mais seja

o emissor do seu CCP ou de documento equivalente de identidade, mas um órgão com competência para definir qualidades específicas de um indivíduo.

A base de funcionamento de uma IGP se dá através de 3 modelos sugeridos em um *Proposed Draft Amendment (PDAM)* que introduz uma emenda ao padrão ISO/IEC 9594-8 (LINN; NYSTRÖM, 1999):

1. O de gerenciamento, envolvendo o objeto solicitado, o solicitante, o verificador e definindo como deve ser feito o controle de acesso a objetos protegidos.

2. O de controle, envolvendo solicitante, verificador, método do objeto requisitado, política de privilégios e variáveis de ambiente, e definindo como o controle será exercido através dos métodos sensíveis dos objetos - isto é, a avaliação do contexto envolvido na solicitação de acesso a um objeto; e

3. O de delegação, envolvendo verificador, solicitante e a origem de autorização, que define como se dá a concessão de autorização de um agente para o outro, validando se o solicitante possui a permissão que procura delegar.

A Figura 3 dá uma visão geral da divisão de papéis dentro de uma Infraestrutura de Gerenciamento de Privilégios:

Figura 3 – Divisão de papéis numa IGP (Retirado de Macêdo (2012))

Sendo assim, as partes envolvidas em uma IGP são:

- Entidade final: representada pelos usuários finais, titulares de CAs.

- Autoridade de atributo.

- Fonte de autoridade: Dada uma hierarquia de Autoridades de Atributo, a Fonte de Autoridade (em inglês *Source of Authority*, ou SoA) seria a Autoridade Raiz da cadeia.

- Verificador de privilégios: no momento da solicitação de um recurso, é o responsável por validar o certificado de atributo que, uma vez validado, resulta na liberação de acesso ao recurso protegido.
- Recurso: é o objeto a ser acessado e obtido/invocado, possuindo propriedades bem definidas controladas pelo Verificador de Privilégios.
- Repositório: local de armazenamento dos certificados e das listas de certificados revogados.

Além disso, há a questão do modelo de distribuição de certificados de atributo dentro de uma IGP. Os dois modelos existentes para a publicação de CAs pelas AAs são:

- *Pull*: modelo que torna mais complexo o serviço para o verificador de privilégios, que será o responsável por buscar os certificados de atributo necessários em repositórios - a adoção deste modelo implica a necessidade de a AA manter disponível um repositório para a consulta dos CAs por ela emitidos.
- *Push*: neste modelo, a complexidade aumenta do lado do cliente, que passa a ser o responsável pelo fornecimento do certificado de atributo para o verificador. Este modelo normalmente é adotado em contextos nos quais há grande diferença entre o domínio do verificador de privilégios e das autoridades de atributo. Embora este modelo torne mais simples a verificação de privilégios, deve-se tomar maiores cuidados na validação dos certificados fornecidos, já que não necessariamente serão recuperados de um repositório confiável.

3 Tecnologias e Bibliotecas Selecionadas

Este capítulo discorre de maneira breve sobre as tecnologias e bibliotecas escolhidas para o desenvolvimento da aplicação proposta por este trabalho. Serão apresentadas somente as bibliotecas e tecnologias mais relevantes e de uso direto.

3.1 A Linguagem Java

A linguagem Java é uma linguagem de propósito geral do paradigma Imperativo, com suporte a concorrência, Orientação a Objetos e baseada em Classes, projetada para ser didática (ORACLE, 2011). Seu amplo uso (sempre entre primeiro e segundo lugar nas linguagens mais utilizadas em projetos atualmente (TIOBE, 2013)), suporte, documentação e destaque em fóruns de discussão justificou a sua escolha para o protótipo de sistema proposto, além de uma grande disponibilidade de bibliotecas públicas que abstraem a realização de muitas tarefas de programação intensivas. Entre outras características da linguagem, pode-se citar a Tipagem Forte e Estática, seu alto nível (omitindo a maior parte dos detalhes do hardware sobre o qual seus programas executam), endereçamento e gerenciamento automático de memória (garbage collection) e seu modo de execução (o código é traduzido para um bytecode compatível com a especificação de uma Máquina Virtual - a JVM - instalada no Sistema Operacional, o que torna portáveis os programas escritos em Java para a maioria dos sistemas operacionais de amplo uso existentes).

Segundo Sebesta (2010), dizer que uma linguagem tem Tipagem Forte, significa que diferentes tipos de identificadores e constantes, quando não são compatíveis, não podem ser utilizados numa mesma expressão. Isto é, uma operação ou expressão que espera operandos de um tipo não será compilável se o código fornecer ao menos um operando de outro tipo, a menos que seja feito algum tipo de conversão ou typecast (algumas conversões são realizadas implicitamente na linguagem, como na operação de concatenação de String, quando apresentado a uma String um operando do tipo inteiro para ser concatenado). Outra característica da tipagem forte é que a detecção de erros de tipagem é realizada sempre - tanto em tempo de compilação quanto em tempo de execução. Tipagem estática significa que identificadores, quando declarados, devem ter um tipo associado na declaração, indicando que aquele identificador só comportará valores do tipo declarado. Orientação a Objetos é um Paradigma de Programação que pode ser resumido às suas três características mais notáveis: (1) Tipos de dados abstratos; (2) Herança e (3) Polimorfismo.

A escolha da linguagem Java para o desenvolvimento do trabalho se deu por vários motivos. Entre eles:

- É, atualmente, a segunda linguagem de programação mais popular que existe (no que diz respeito à quantidade de projetos e referências existentes) segundo o índice TIOBE (2013).
- Uma consequência de sua popularidade, é a ampla documentação existente, mantida por uma comunidade bastante ativa.
- Ainda como consequência da popularidade, há o grande número de códigos escritos por terceiros que podem ser utilizados como referência.
- Possui um ferramental muito extenso de bibliotecas que auxiliam no desenvolvimento, e são mantidos por grandes organizações e/ou comunidades, das quais um grande exemplo é a fundação Apache.
- O fato de os programas escritos em Java executarem sobre uma máquina virtual ajuda a reduzir as preocupações relativas ao ambiente de execução da linguagem, tornando os programas facilmente portáveis para várias plataformas.

3.1.1 O Java Community Process (JCP)

Outro atrativo que levou à escolha do Java como linguagem para implementação da aplicação é a existência de um mecanismo para desenvolver especificações técnicas padrão para tecnologias da linguagem. Esse mecanismo é o Java Community Process (JAVA COMMUNITY, 2003), através do qual qualquer pessoa interessada nas tecnologias que envolvem o uso de Java podem contribuir com opinião e se inscrever para ser membro, auxiliando no processo de escolha de Java Specification Requests (JSRs).

As JSRs são os documentos com especificações formais de tecnologias candidatas para a plataforma Java, que são submetidos a revisões formais públicas antes de se tornarem finais e serem votados pelo comitê executivo do JCP. Existe uma JSR que descreve o próprio JCP (JAVA COMMUNITY, 2009a). As JSRs funcionam de forma análoga às Requests For Comments (RFCs) (IETF, 2004), mas dentro do escopo da plataforma Java.

Assim sendo, as JSRs delimitam que Interfaces de Programação de Aplicativos (APIs) são suportadas pelo Java em suas plataformas *Standard Edition* (SE), *Micro Edition* (ME) e *Enterprise Edition* (EE). Como o foco da aplicação proposta é a plataforma Java EE (na sua versão 6, descrita pela JSR 316 (JAVA COMMUNITY, 2009c)). Embora dezenas de JSRs estejam envolvidas no uso do Java EE 6, algumas JSRs merecem destaque por serem fundamentais para a compreensão do código desenvolvido: *Enterprise Java Beans* (EJB), cuja versão 3.1 é descrita pela JSR 318 (SAKS, 2009a); *Java Persistence API* (JPA), cuja versão 2.0 é descrita pela JSR 317 (DEMICHIEL, 2009a); *Contexts and Dependency Injection* (CDI), contemplada pela JSR 299 (KING, 2009a); *Java Server Faces* (JSF), na versão 2.2, descrita na JSR 344 (JAVA COMMUNITY, 2013a). Essas JSRs

serão introduzidas de uma maneira breve nesta seção, sem adentrar em detalhes de uso e implementação.

3.1.2 Java Enterprise Edition (Java EE) 6

Sistemas corporativos são, geralmente, implementados como aplicações multicamadas, sendo que, geralmente, há camadas intermediárias nesses sistemas que se encarregam de integrar os sistemas de informação corporativos com os dados e funções de negócio. O objetivo do Java EE é reduzir o custo e a complexidade no desenvolvimento dessas aplicações multicamadas (JAVA COMMUNITY, 2009b).

De uma forma breve, pode-se dizer que o Java EE é uma seleção de APIs que, em conjunto, oferecem produtividade no desenvolvimento de sistemas corporativos, permitindo maior foco nos negócios a partir da abstração de obstáculos técnicos no desenvolvimento, como invocação remota de procedimentos, mapeamento objeto-relacional, interface e controle visual de aplicações Web etc.

A Figura 4 apresenta a relação lógica entre os elementos da arquitetura da plataforma Java EE 6. Sendo os retângulos as representações dos contêineres, na parte superior dos retângulos estão os componentes de aplicação envolvidos e na parte inferior são evidenciados os serviços suportados por esses contêineres (os serviços destacados com o cinza escuro são as novidades da versão 6 do Java EE em relação às versões anteriores). As setas representam o acesso requisitado às demais partes da plataforma.

3.1.2.1 Enterprise JavaBeans (EJB)

A especificação de EJBs tem como objetivo simplificar o desenvolvimento de aplicações modulares com Java, permitindo o uso de objetos distribuídos associados a um contexto de execução de forma que o desenvolvedor não precise entender detalhes de gerenciamento de estado e de transações, pooling de conexões e multi-threading (SAKS, 2009b).

Além disso, os EJBs fornecem facilidade para transformar o código desenvolvido em *Web Service* e promover uma maior interoperabilidade de sistemas. São geralmente utilizados na implementação da camada de modelo de aplicações corporativas, executando a lógica de negócio dessas aplicações.

3.1.2.2 Java Persistence API (JPA)

A JPA é a API que torna possível a representação, com objetos, de um modelo de dados relacional (mapeamento objeto-relacional) (DEMICHIEL, 2009b), abstraindo a complexidade de interagir com bancos de dados e diminuindo o esforço com refatoração de código mediante mudanças no modelo relacional. Através de anotações de classes e

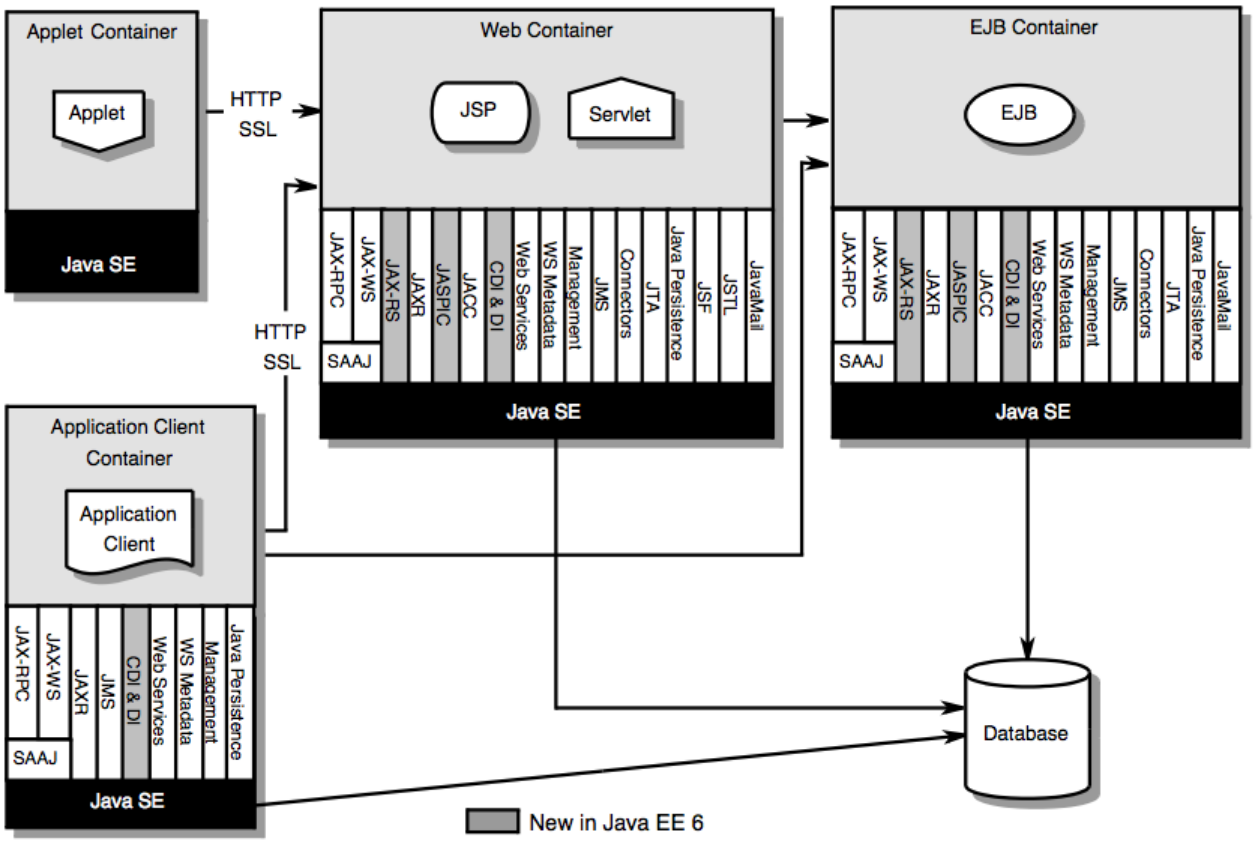

Figura 4 – Diagrama Arquitetural do Java EE 6 (Retirado de Java Community (2009b))

atributos cria-se o mapeamento e é oferecida uma linguagem declarativa de consulta (JPQL, ou *Java Persistence Query Language*) similar ao SQL, mas mais próxima da orientação a objetos.

3.1.2.3 Java Server Faces (JSF)

JSF é um framework de desenvolvimento de interface gráfica para aplicações Web projetado para diminuir a complexidade de criar e manter aplicações que requerem regras de renderização, por vezes complexas, em alguns clientes Web (JAVA COMMUNITY, 2013b).

JSF oferece uma paleta de componentes reutilizáveis que diminuem o volume de código escrito, além de permitir a construção de componentes próprios. Também é facilitado o transporte de dados da Interface Gráfica para o servidor e vice-versa, além da simplicidade em atribuir eventos gerados no cliente com código do lado do servidor.

3.1.2.4 Contexts and Dependency Injection (CDI)

CDI trata-se de um conjunto de serviços complementares à pltaforma Java EE que permitem uma melhor estrutura do código da aplicação. A API do CDI permite que,

com uma série de anotações, seja possível definir estado, ciclo de vida, pontos de injeção de dependências e nomes para objetos Java cujo ciclo de vida deve ser gerenciado pelo contêiner Java EE.

Em outras palavras, o CDI permite a declaração e uso facilitados de Java Beans (tanto em código como via *Expression Language*), de forma desacoplada (KING, 2009b). Entende-se por "*Java Beans*", os objetos Java que pertencem a algum contexto de execução dentro de um contêiner Java EE - EJBs *Stateful* e *Managed Beans* do JSF, por exemplo.

A principal função do CDI é de integrar os EJBs, tipicamente utilizados para implementação da camada de modelo de uma aplicação, do JSF, utilizado na camada de visão (considerando uma arquitetura *Model-View-Controller*, ou MVC), permitindo a interação entre EJBs e *Managed Beans*.

3.2 O Servidor de Aplicação JBoss (JBoss AS)

Um Servidor de Aplicação é um programa executado em um computador conectado em rede, sendo que esse programa provê um ambiente de execução completo para aplicativos que seguem determinadas especificações pré-estabelecidas (ROUSE; ROURKE, 2005). No caso da linguagem Java, considera-se como um servidor de aplicação um programa que provê todas as bibliotecas e funcionalidades especificadas pela plataforma Java EE.

O Servidor de Aplicação JBoss AS encaixa-se nessa categoria de programas e é uma das alternativas mais utilizadas atualmente (KALALI, 2009). Entre os demais motivos para a escolha do JBoss AS como servidor para executar a aplicação proposta, estão:

1. A versão escolhida (7.1.1.Final) apresenta uma inicialização rápida, adequada para o tempo de desenvolvimento.

2. Segue a especificação do Java EE versão 6 (Java EE 6 *Compliant*).

3. Configuração facilitada, através de um único arquivo XML existente no diretório da instância do servidor que será utilizada.

4. Amplo suporte e documentação.

5. Projeto modular, permitindo que sejam desenvolvidos módulos acessórios para o servidor.

6. Possui um sistema de gerenciamento de banco de dados (SGBD) embarcado, o H2, que atende às necessidades de um protótipo de aplicação como o proposto nesse trabalho. Para realizar uma migração para um SGBD, se torna necessário, apenas,

realizar as alterações nos *Data Sources* da aplicação (geralmente configurados em arquivos XML.

Mais informações sobre a versão 7.1.1.Final do servidor de aplicação JBoss podem ser encontradas na documentação oficial (KHAN et al., 2012), que é escrita tanto para desenvolvedores que pretendem utilizar o JBoss AS, quanto para administradores de sistemas que irão gerenciar o servidor.

3.3 PERMIS

Guilhen (2008), em seu trabalho, realizou uma análise comparativa de diferentes ferramentas de controle de acesso baseado em CAs para propor sua própria solução de autorização. Em seu trabalho, ainda é desenvolvido um *framework* de autorização baseado no uso de certificados de atributo. No entanto, o código fonte dessa solução não foi disponibilizado e, ainda assim, operava sobre uma versão muito antiga do servidor JBoss (a versão 4.2.3), impossibilitando sua adoção. Uma das ferramentas apresentadas é o PERMIS, escolhido para o desenvolvimento do presente trabalho por dois motivos principais: (1) aderência do padrão dos certificados emitidos com as recomendações de Farrel, Housley e Turner (2010), e (2) uma API escrita em Java para realizar as chamadas aos serviços de autorização, o que torna uma escolha imediata, tendo em vista a linguagem de programação escolhida para o desenvolvimento do presente trabalho.

O PERMIS, acrônimo para *PrivilEge and Role Management Infrastructure Standards* é um *framework* de autorização baseado em CABP e que oferece suporte ao uso de certificados de atributo X.509, que foi desenvolvido no Instituto de Segurança da Informação da Universidade de Salford.

O PERMIS se apóia na definição de uma política de segurança descrita em XML que é armazenada em um serviço de diretório LDAP, bem como também o são os certificados de atributo emitidos pela ferramenta, e se divide, basicamente, em dois subsistemas: (1) o de Alocação de Privilégios, ilustrado pela Figura 5 e (2) o de Verificação de Privilégios, ilustrado pela Figura 6.

O Alocador de Privilégios é o subsistema que provê as ferramentas necessárias para geração e armazenamento da política de segurança - um arquivo XML, assinado digitalmente com o certificado da Fonte de Autoridade, que depois é persistido em um serviço de diretórios LDAP que deve estar acessível publicamente para leitura. Uma vez que a política é assinada digitalmente, não há como alterar seu conteúdo sem comprometer a assinatura, portanto, não há uma necessidade de tornar seguro o canal de comunicação com o serviço LDAP que provê a política. Quanto aos CAs de usuários finais, devem ser persistidos dentro do contexto de cada usuário dentro da base LDAP.

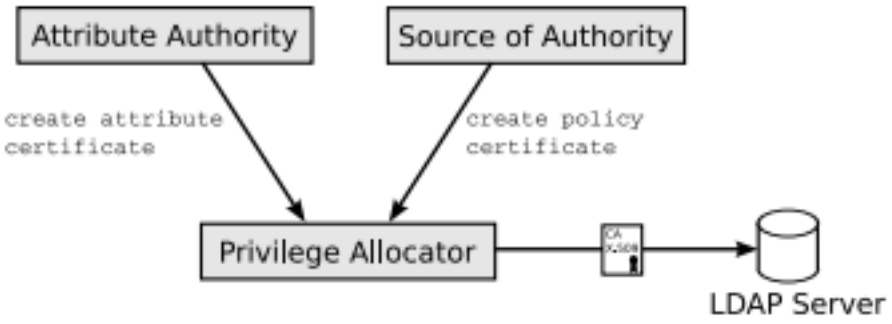

Figura 5 – Subsistema de Alocação de Privilégios do PERMIS (Retirado de Guilhen (2008))

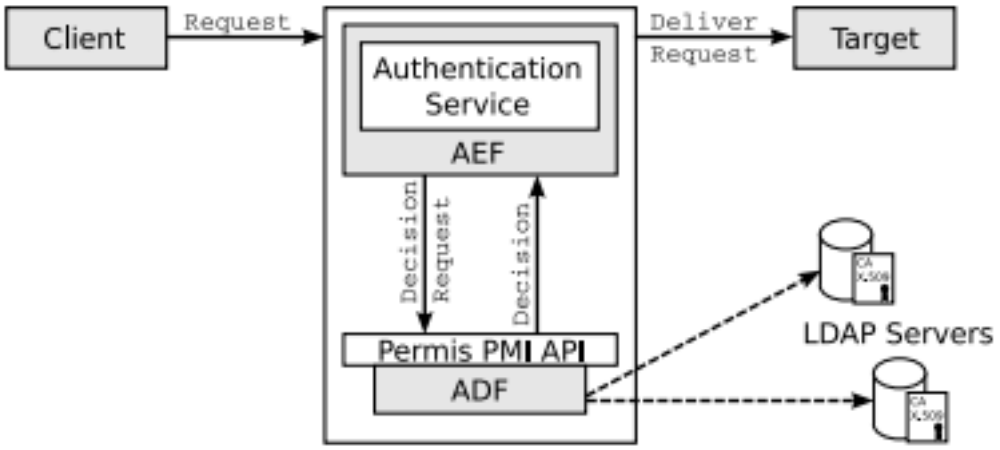

Figura 6 – Subsistema de Verificação de Privilégios do PERMIS (Retirado de Guilhen (2008))

Já ao subsistema de Verificação de Privilégios cabe a parte de autorizar o acesso de usuários, realizando a verificação de suas permissões. É dividido essencialmente entre duas partes, a AEF (*Access-Control Enforcement Function*) e a ADF (*Access-Control Decision Function*), sendo que a primeira é responsável por verificar o estado de autenticação dos usuários e a segunda pela aplicação da política de autorização. Tais funções são acessíveis através de uma API que também realiza a comunicação entre as partes.

Entre as limitações do PERMIS estão o fato de ser obrigatório o armazenamento de certificados e políticas de segurança em LDAP (impossibilitando o fornecimento direto do certificado pelo cliente) e pelo fato de o atributo de autorização ser proprietário e imutável (o atributo *permisRole*) - o que restringe em muito as possibilidades de adoção do PERMIS em um contexto real sem que sejam feitas alterações no código fonte do sistema. No entanto, seu baixo acoplamento o torna uma boa solução a ser adotada, por exigir código mínimo e configuração mínima das aplicações clientes, como a proposta por esse trabalho.

3.4 Apache Shiro

O Shiro é um *framework* de segurança de código aberto, desenvolvido em Java, que auxilia no desenvolvimento de mecanismos de autenticação, autorização, criptografia e gerenciamento de sessão. Seu principal objetivo é o de ser fácil de utilizar e de se compreender, uma vez que segurança de aplicações costuma ser algo complexo e difícil de se implementar (APACHE FOUNDATION, 2013).

Sua escolha se deu pelo fato de ser um *framework* para implementação de mecanismos de segurança baseados na aplicação, enquanto que a maioria das ferramentas com o mesmo propósito realizam autenticação e autorização com base no contêiner (servidor de aplicação), como o PicketLink (projeto que hoje incorpora o PicketBox), mantido pela comunidade JBoss, ou o Seam Security, feito especificamente para o *framework* JBoss Seam. Além disso, o Apache Shiro segue o princípio de CoC, o que minimiza os esforços com configuração da biblioteca, como é o caso do JAAS (*Java Authentication and Authorization Service*), que é bastante flexível e robusto, mas a dificuldade com configuração torna seu uso pouco intuitivo.

Entre algumas das funcionalidades providas pelo Shiro estão:

- Autenticar um usuário para verificar sua identidade.
- Prover controle de acesso.
- Prover uma API de gerenciamento de sessão independente da existência de containers EJB ou de Servlet no ambiente de execução.
- Responder a eventos durante os processos de autenticação, controle de acesso, ou durante o tempo de vida de uma sessão.
- Permitir que se coloque um mecanismo de segurança em uso de uma maneira pouco acoplada através da implementação de poucas interfaces e de configuração mínima.

A Figura 7 apresenta uma visão arquitetural que abstrai uma série de detalhes de funcionamento do framework, mas que torna claros os pontos de interação entre a aplicação e o código específico de segurança. Destaque para três elementos:

- *Subject*: É o usuário que está utilizando o sistema, do ponto de vista de segurança, e é o elemento que intermedia a interação da aplicação com o *Security Manager*.
- *Security Manager*: É o elemento que realiza a ligação entre todos os componentes de segurança por trás do uso do *Subject*. Trata-se de um elemento "guarda-chuva", que abstrai os detalhes de funcionamento dos mecanismos de segurança do Shiro.

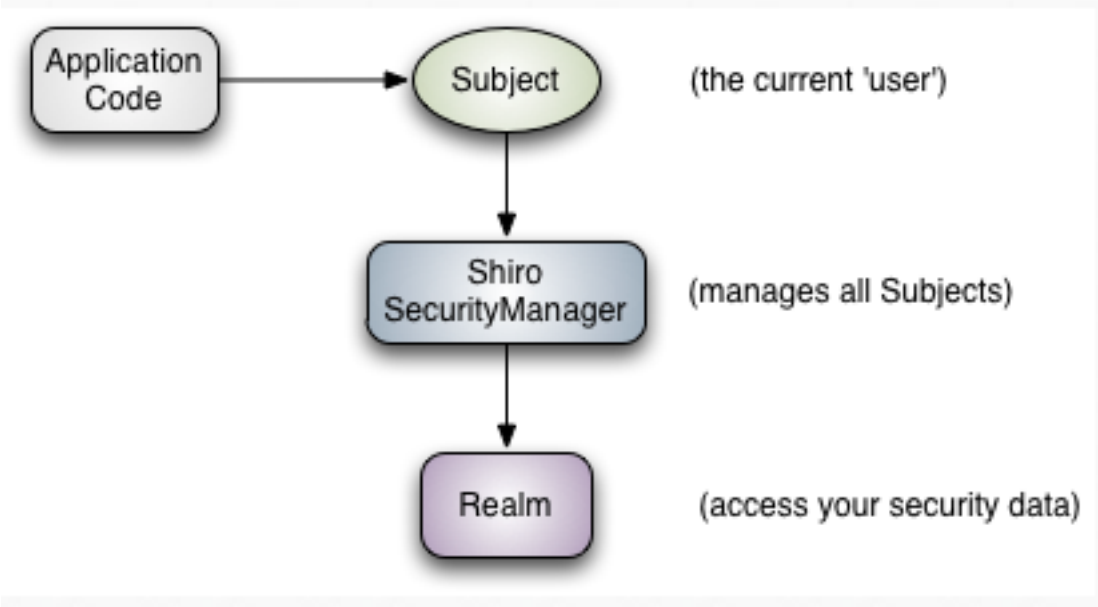

Figura 7 – Shiro: visão arquitetural de alto-nível (Retirado de Apache Foundation (2013))

- *Realm*: É o conector entre a aplicação e seus dados de segurança. Na sua essência, um *Realm* é um DAO específico para segurança, que tratará de encapsular detalhes de conexão com fontes de dados e de verificações de credenciais de autenticação.

O *Security Manager* do Shiro está dividido em diversos subelementos. É importante destacar que, para o uso comum, o Shiro já oferece implementações padrão para praticamente todos os elementos e subelementos de sua arquitetura, respeitando o conceito de CoC (convenção sobre a configuração), bastando diretivas de configuração para realizar personalizações dos elementos que não atenderem às necessidades de uma aplicação que venha a utilizar o *framework*. Alguns outros elementos presentes são:

- Autenticador: Componente que responde às tentativas de autenticação e executa a lógica que coordena o uso dos *Realms* para verificar credenciais. Um subelemento do Autenticador, que é a Estratégia de Autenticação, é que chaveia o uso de um *Realm* ou de outro para autenticação, quando da existência de mais de um *Realm*.

- Autorizador: É o componente responsável por dizer se um usuário tem acesso a um recurso ou não, e funciona de forma similar ao Autenticador quando da existência de múltiplas fontes de acesso a dados de permissões.

- Gerenciador de Sessão: Elemento responsável pela criação, gerenciamento e ciclo de vida de uma sessão de usuário e é um dos pontos notáveis do Shiro, por permitir que exista gerenciamento de sessão mesmo fora de ambientes Web (*servlet containers*), possibilitando a um usuário criar sessões nativas.

- DAO de Sessão: Define como são persistidos os dados de sessão. Em uma aplicação Java Web tradicional, o único meio de persistir os dados de sessão segundo a especificação, é no contexto do servlet, a partir de um *Hash Map* vinculado à sessão do usuário. O Shiro permite que os dados de sessão sejam persistidos em qualquer estrutura de dados definida pelo usuário, desde que implementadas as interfaces corretas para atuar como um *Session DAO*.
- Gerenciador de *Cache*: elemento necessário para melhorar o desempenho de acesso a dados, uma vez que diversas funcionalidades do Shiro preveem a possibilidade de se disponibilizar diferentes fontes de dados para se trabalhar.
- Criptografia: Conjunto de classes que oferecem um nível de abstração adicional para o uso de *codecs*, algoritmos de *Hash* e de cifragem.

A Figura 8 representa, visualmente, o modo como se dá a interação entre os elementos apresentados dentro da arquitetura do Shiro.

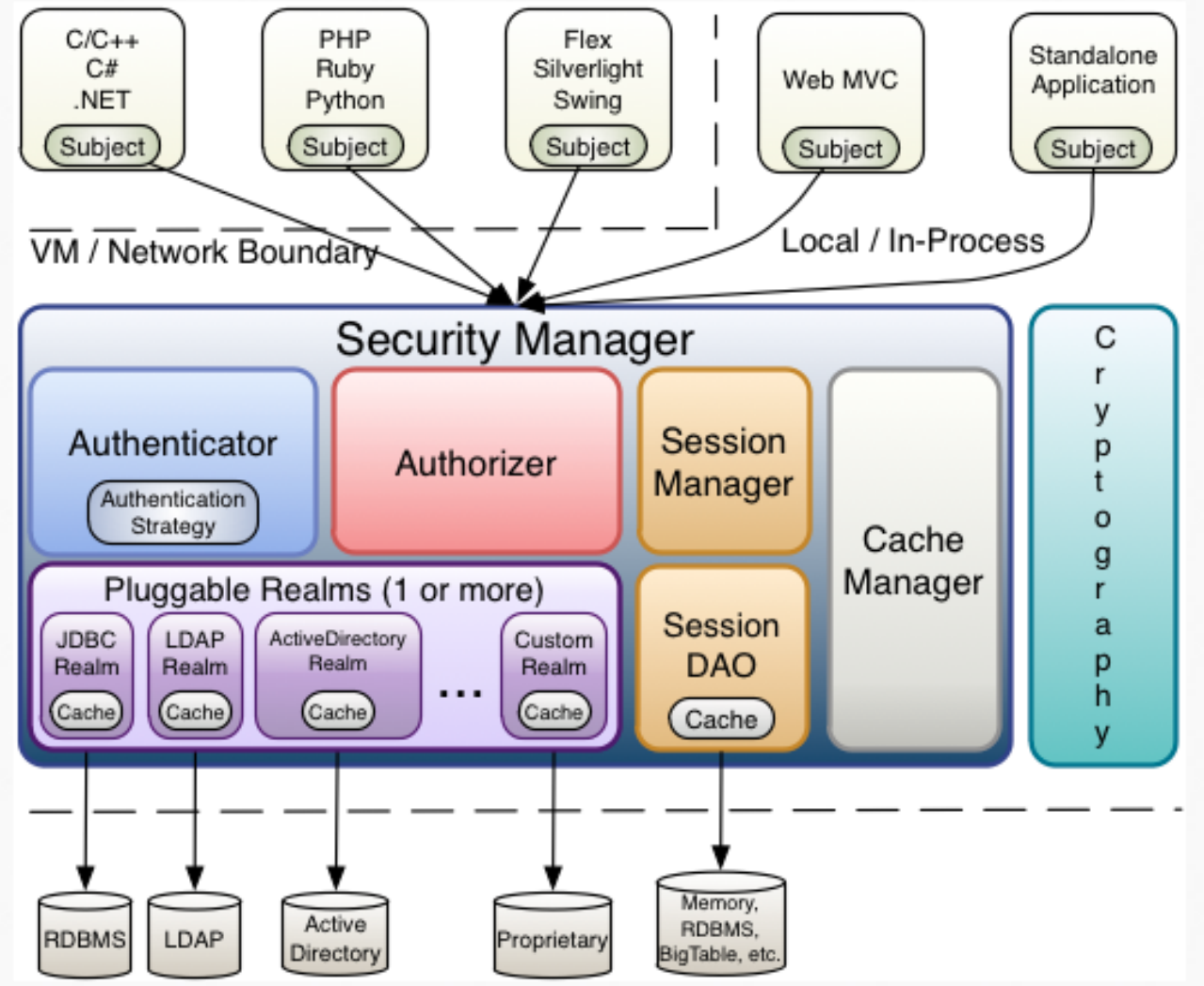

Figura 8 – Shiro: arquitetura detalhada (Retirado de Apache Foundation (2013))

4 Especificação da Aplicação

O presente capítulo tem por objetivo especificar a aplicação desenvolvida no trabalho, de forma a delimitar o problema a ser tratado e a forma como será tratado, de um modo breve e compreensível.

4.1 Justificativa

A automatização de processos, antes exclusivamente manuais trouxe benefícios inquestionáveis para as mais variadas áreas. Em se tratando de processos que envolvem a emissão de documentos, especialmente, a automatização se torna ainda mais interessante.

Estabelecimento de contratos, procedimentos de compra e venda e diversas outras interações que geram documentos como prova de sua existência só tem valor a agregar com a automatização, já que documentos exclusivamente físicos estão sujeitos a diversas condições que não afetam documentos eletrônicos. Algumas delas são:

- Espaço físico necessário para sua armazenagem.
- Ação do tempo sobre o material.
- Cuidados com condições de armazenamento.
- Cuidados com preservação dos documentos originais.
- Extravio de documentos de via única.
- Disponibilidade e meios de distribuição.

Embora o uso de documentos eletrônicos crie novos riscos para o processo, de uma forma geral as vantagens se sobressaem. É o exemplo da possibilidade de melhor fiscalização na sua emissão e uso, não sofrem com a ação do tempo e intempéries sobre o material, e já existem técnicas que visam assegurar a sua autenticidade.

A emissão de atestados médicos é um processo ainda predominantemente manual no Brasil e se dá, quase que exclusivamente, no momento da interação médico-paciente e exigindo conhecimento prévio por parte do paciente do número de vias que serão necessárias bem como condições específicas para sua apresentação. Há iniciativas dentro do próprio Sistema Único de Saúde de automatização, mas apenas para os processos internos, excluindo o cidadão comum do benefício dessa automatização, assim como a Associação

Paulista de Medicina também possui um sistema de emissão eletrônica de atestados médicos, mas que possui um modelo de cobrança que onera o médico, dificultando sua adoção e descaracterizando-o como serviço de utilidade pública.

Dado esse contexto, um sistema de emissão eletrônica de Atestados Médicos se faz desejável tanto pelo benefício ao cidadão, que terá acesso a quantas cópias forem necessárias do seu atestado, como beneficiará ao médico emissor, que poderá controlar os atestados por ele emitidos e permitirá uma melhor fiscalização desses documentos, auxiliando, portanto, na detecção de fraudes.

4.2 Visão Geral

Este trabalho propõe uma aplicação que permita a um médico emitir atestados de forma eletrônica para seus pacientes. O registro do atestado emitido permanecerá no sistema para controle do próprio médico e para emissão de vias adicionais para o paciente. Será disponibilizada, também, uma verificação de autenticidade baseada em Hash, similar à funcionalidade de emissão de atestado de matrícula da UFSC, para controlar a falsificação de atestados médicos. Os médicos deverão ter seu registro no CRM cadastrado no sistema, sujeito à aprovação de um administrador, para que possam realizar a emissão e o controle desses atestados.

4.3 Escopo da Aplicação

A aplicação proposta trata-se de um protótipo com as principais funcionalidades de um sistema de emissão de atestado médico implementadas. Entre as funcionalidades atendidas estão o cadastro de usuários, autenticação e desautenticação, cadastro de registro no CRM, verificação de autenticidade de atestado médico, emissão de segunda via de atestado médico, listagem de atestados médicos tanto por parte de usuários comuns quanto de médicos, emissão de atestado médico, autorização de registro no CRM e ajuste de configurações do sistema.

O "Registro no CRM"utilizado na aplicação trata-se apenas de uma referência ao número do CRM do médico, para que essa informação conste no Atestado Emitido. Não é, de forma alguma, um atributo gerenciado pela entidade mais adequada (o Conselho Regional ou Federal de Medicina). Sendo essa uma das limitações do trabalho.

O uso de certificados de atributo se restringirá ao de certificados que armazenam o papel (*Role*) do usuário dentro do sistema. E a emissão desses certificados será gerenciada por uma entidade externa à aplicação (no caso deste trabalho, essa entidade é representada pelo subsistema de emissão de CAs do PERMIS). Essa limitação é necessária, uma vez

que não há total liberdade, dada pela ferramenta, para se criar um atributo. Em vez disso, deve-se trabalhar com um conjunto de atributos limitado pelo próprio PERMIS.

A Resolução nº 1.658/2002 do Conselho Federal de Medicina (2002) define os seguintes campos para um atestado médico:

- Diagnóstico.
- Prognóstico.
- Resultados de exames complementares.
- Conduta terapêutica.
- Consequências à saúde do paciente.
- Data de emissão.
- Data de validade do atestado (definida pelo tempo de repouso necessário para o paciente).
- Identificação do médico.
- Identificação do paciente.
- CID (Código Internacional de Doenças).

São campos adicionais disponibilizados pelo sistema para preenchimento:

- Propósito da emissão do atestado.
- Comentários adicionais.

São itens fora de escopo desta aplicação:

1. Assinatura digital do atestado médico (e, portanto, sua "validade jurídica").
2. Normalização de endereços e verificações de integridade.
3. Validação de formato de CRM.
4. Obrigatoriedade de todos os campos exigidos pela Resolução nº 1.658/2002 do Conselho Federal de Medicina (2002), embora todo os campos exigidos estejam disponíveis para preenchimento.
5. Exigência de assinatura para expressar concordância do paciente com a publicação do CID.

6. Implementação de funcionalidades administrativas, como manutenção de usuários, de médicos, bloqueio de contas de usuários etc.
7. A administração de perfis de usuários e atribuição dos mesmos.
8. Alteração de dados pessoais de usuários.
9. Exigência de confirmação de cadastro de usuário, antes do primeiro uso do sistema.
10. A aplicação proposta não contempla o uso por cidadãos que não possuem CPF, por questões de complexidade do tratamento de duplicidade de nomes e datas de nascimento - procurou-se trabalhar com um conjunto limitado de informações do usuário, já que a finalidade da aplicação é a de servir como uma prova de conceito, e não de prover uma solução completa de emissão de atestados médicos.
11. Requisitos de usabilidade do sistemas, como densidade informacional, navegabilidade e disposição de campos de formulário e mensagens de retorno da interface com o usuário.
12. Uso de SGBD externo ao servidor de aplicação - reinicializações da aplicação farão com que o estado da base de dados volte a ser o da primeira instalação.
13. Paginação de telas de listagem.
14. Requisitos de desempenho e de disponibilidade da aplicação.

4.4 Atores Envolvidos

A presente seção descreve os atores envolvidos na interação com o sistema. Um ator não necessariamente implica a existência de um papel (Role) correspondente, mas sim uma condição do agente que está interagindo com o sistema em determinado momento.

4.4.1 Visitante

Visitante não é um papel.

Visitante é a condição de um usuário que está acessando o sistema e que ainda não realizou autenticação.

Ao Visitante estão disponíveis as seguintes funcionalidades:

- Cadastrar usuário.
- Autenticar-se no sistema.
- Verificar autenticidade de atestado médico, desde que em posse do seu código de autenticação.

4.4.2 Usuário Comum

Usuário Comum não é um papel.

Usuário Comum é a condição de um usuário que está acessando o sistema e que já realizou a autenticação.

Um Usuário Comum possui acesso a todas as funcionalidades que o Visitante possui, à exceção da possibilidade de autenticar-se no sistema, uma vez que o processo de autenticação já foi feito.

Além das funcionalidades de Visitante, um Usuário Comum possui acesso a uma listagem dos atestados médicos que já foram emitidos em seu nome, tanto válidos como já vencidos, bem como a possibilidade de gerar arquivos PDF desses atestados.

4.4.3 Médico

Médico é um papel (*Role*) do sistema.

Médico é uma extensão do ator Usuário Comum, portanto, para se possuir a condição de Médico, deve-se estar autenticado no sistema.

Um Médico possui acesso a todas as funcionalidades que um Usuário Comum possui.

Um médico também pode:

- Solicitar a um Administrador do Sistema que autorize seu registro no CRM.
- Emitir atestados médicos.
- Listar atestados médicos por ele emitidos e gerar arquivos PDF daqueles certificados.

4.4.4 Administrador do Sistema

Administrador do Sistema é um papel (*Role*) do sistema.

Administrador do Sistema é uma extensão do ator Usuário Comum, portanto, para se possuir a condição de Administrador do Sistema, é necessário estar autenticado.

Um Administrador do Sistema possui acesso a todas as funcionalidades que um Usuário Comum possui.

Um Administrador do Sistema também pode:

- Autorizar inclusão de registros no CRM solicitadas por Médicos.
- Alterar configurações do sistema.

4.5 Requisitos

4.5.1 Requisitos Funcionais

RF01 - O sistema deve permitir que um usuário se autentique.

Informações: Nome de usuário, senha.

Regras: O sistema deve permitir que um usuário, a partir de um link do sistema, ou ao tentar acessar uma página restrita, seja redirecionado para a tela de autenticação. Na tela de autenticação, devem ser apresentados campos para informar nome de usuário e senha e a opção de envio do formulário para autenticação. Na ocorrência de algum erro de autenticação, o usuário deve ser informado.

- O nome de usuário é de preenchimento obrigatório.
- A senha é de preenchimento obrigatório.
- Informar um nome de usuário que não possui nenhum usuário correspondente dispara um erro.
- Informar nome de usuário e senha que não possuem correspondência dispara um erro.

RF02 - O sistema deve possibilitar a verificação de autenticidade de atestados.

Informações: Código de verificação (de autenticidade do atestado médico).

Regras: O sistema deve disponibilizar acesso público a uma página que permita verificar a autenticidade de um atestado médico emitido através do sistema. A verificação se dá através do envio do formulário com o Código de verificação de um atestado. Após o envio do formulário, será oferecido o download de um arquivo no formato PDF no qual constam os dados do atestado original para que possam ser comparados a alguma via apresentada. Caso os dados do PDF oferecido sejam diferentes dos dados do atestado sendo verificado, isso significa que o atestado verificado é falso.

- O Código de verificação é de preenchimento obrigatório.

RF03 - O sistema deve permitir que um usuário se cadastre.

Informações: Nome de usuário, senha, confirmação de senha, e-mail, CPF, nome, data de nascimento e gênero.

Regras: O sistema deve disponibilizar acesso *[público* a uma página que permita que um visitante se cadastre no sistema. Todos os campos do formulário de cadastro são de preenchimento obrigatório.

- Não são permitidos dois usuários com o mesmo nome de usuário ou com mesmo endereço de e-mail no sistema.
- A senha deve ser digitada duas vezes.
- Somente CPFs válidos serão aceitos (segundo a fórmula de verificação do CPF).

RF04 - O sistema deve permitir que um usuário se desautentique.

Informações: Não aplicável.

Regras: Uma vez autenticado no sistema, o sistema deve disponibilizar um link para que o usuário se desautentique (realize *logout* do sistema). Também deve ocorrer a desautenticação no caso de o usuário fechar o navegador.

RF05 - O sistema deve listar atestados médicos emitidos para o usuário.

Informações: Médico emissor, nome do atestado, data de emissão, validade.

Regras: O sistema deve disponibilizar um link para que o usuário, autenticado, veja a lista de atestados que já foram para ele emitidos. Deve ser possível filtrar para que a lista exiba apenas os atestados válidos. Também deve ser disponibilizado, para cada atestado apresentado, um link para visualizar detalhes do atestado e que permita o download no formato PDF.

RF06 - O sistema deve listar atestados médicos emitidos pelo médico.

Informações: Médico emissor, nome do atestado, data de emissão, validade.

Regras: O sistema deve disponibilizar um link para que o médico, autenticado, veja a lista de atestados que já foram emitidos por ele. Deve ser possível filtrar para que a lista exiba apenas os atestados válidos. Também deve ser disponibilizado, para cada atestado apresentado, um link para visualizar detalhes do atestado e que permita o download no formato PDF.

RF07 - O sistema deve permitir a um médico cadastrar seu registro no CRM.

Informações: Número do registro no CRM, país, estado, cidade e localidade.

Regras: O sistema deve disponibilizar um link para que o médico, autenticado, cadastre seus registros no CRM. Os registros cadastrados pelo médico devem passar pela aprovação de um administrador do sistema antes que possam ser utilizados. O sistema não permitirá números de registro no CRM duplicados.

- O número do registro no CRM é de preenchimento obrigatório.

- O país é de preenchimento obrigatório.
- O estado é de preenchimento obrigatório.

RF08 - O sistema deve permitir a um médico, emitir um atestado médico.

Informações: Número do registro no CRM, CPF do paciente, nome do paciente, data de nascimento do paciente, gênero do paciente, propósito do atestado, validade do atestado, comentários, código da classificação internacional de doenças (CID), diagnóstico, prognóstico, conduta terapêutica, consequências à saúde do paciente, resultados de exames complementares.

Regras: O sistema deve disponibilizar um link para que o médico, autenticado, emita atestados médicos. No campo de seleção do número de registro no CRM, devem ser exibidos apenas os registros aprovados por administradores do sistema. Ao preencher o CPF do paciente, deve ser realizada uma busca no sistema para ver se os dados daquele paciente já constam no sistema, preenchendo automaticamente os demais campos de identificação do paciente caso aquele paciente já exista. Após a emissão do atestado, deve ser gerado um código de verificação de autenticidade e os dados do atestado emitido não podem mais ser editados.

- O número do registro no CRM é de preenchimento obrigatório.
- CPF, nome, data de nascimento e gênero do paciente são de preenchimento obrigatório.
- O propósito do atestado é de preenchimento obrigatório.
- A data de validade do atestado é de preenchimento obrigatório.

RF09 - O sistema deve permitir que um administrador autorize registros no CRM.

Informações: Nome do requerente, número do registro no CRM, país, estado, cidade e localidade.

Regras: O sistema deve disponibilizar um link para que o administrador do sistema, autenticado, autorize ou bloqueie registros no CRM cadastrados por médicos. Deve ser exibida uma listagem com todas as solicitações pendentes de aprovação e o administrador deve ter a opção de autorizar ou de não autorizar (bloquear). A não autorização de um registro implica a sua remoção física da base de dados do sistema, tornando aquele registro disponível novamente para solicitação.

RF10 - O sistema deve permitir que um administrador edite configurações do sistema.

Informações: Chave, valor.

Regras: O sistema deve disponibilizar um link para que o administrador do sistema, autenticado, visualize uma listagem de configurações do sistema apresentadas na forma de um mapa "chave/valor". Somente é possível realizar a inclusão de novas configurações ou edição de configurações já existentes. Não é possível inserir duas configurações com a mesma chave. A edição deve ser disponibilizada de forma tabular (no mesmo contexto em que é feita a listagem - em lote).

- A chave da configuração é de preenchimento obrigatório.
- São configurações básicas do sistema:

  Duração de conversação: tempo de uma "transação"dentro do sistema.

  Algoritmo de Hash: para geração do código de verificação de autenticidade.

4.5.2 Requisitos Não-Funcionais

4.5.2.1 Implementação

Nesta seção estão detalhados os Requisitos Não-Funcionais que esclarecem questões relativas à Implementação da Aplicação, como linguagem utilizada, tecnologias, bibliotecas e frameworks envolvidos e ambiente, durante o processo de desenvolvimento da aplicação proposta.

- **RNF01:** O sistema deve ser implementado em Java.
- **RNF02:** O gerenciamento de dependências deve ser feito com Apache Maven.
- **RNF03:** A segurança da aplicação Web deve ser implementada com Apache Shiro e PERMIS.
- **RNF04:** A implementação da camada de negócio deve utilizar JPA.
- **RNF05:** A implementação da camada de apresentação deve utilizar JSF.

4.5.2.2 Configurabilidade

Nesta seção, são apresentados os artefatos configuráveis do sistema. Isto é, que comportamentos que o sistema oferece que serão alteráveis através de configuração por parte do usuário final.

- **RNF06:** O tempo de duração das conversações do CDI (transações) poderá ser personalizado por um administrador.
- **RNF07:** O algoritmo de hash para geração de código de verificação de autenticidade poderá ser personalizado por um administrador.

4.5.2.3 Segurança

Nesta seção são apresentados os Requisitos de segurança da aplicação, como regras gerais de controle de acesso, ferramentas de segurança utilizadas e parâmetros de segurança para a aplicação.

- **RNF08:** O mecanismo de autenticação do sistema deverá ser de usuário e senha.
- **RNF09:** O mecanismo de autorização do sistema deverá utilizar certificados de atributo.
- **RNF10:** O sistema não disponibilizará distribuição de papéis (*roles*) de usuários. Isso deverá ser feito externamente, através do PERMIS.

4.5.2.4 Legalidade

Nesta seção são apresentados aspectos legais que serão tratados pela aplicação proposta.

- **RNF11:** Os campos previstos para um atestado médico devem ser, no mínimo, o mesmo conjunto do apresentado em Conselho Federal de Medicina (2002).
- **RNF12:** Embora alguns dos campos apresentados na Conselho Federal de Medicina (2002), essa obrigatoriedade não será considerada no sistema.
- **RNF13:** Não é responsabilidade do sistema manter os registros de CRMs sincronizados.

4.6 Casos de Uso

4.6.1 UC01 - Cadastro de Usuário

Sumário

O objetivo deste caso de uso é permitir que um visitante registre-se no sistema para acessar a área restrita do sistema, que oferece mais funcionalidades.

Pré-condições

São pré-condições do Cadastro de Usuário:

- O usuário não pode estar autenticado.

Figura 9 – Cadastro de Usuário.

Fluxo

1. O usuário preenche os campos do formulário.

   a) Se o CPF informado já existir no sistema, os campos Nome, Gênero e Data de Nascimento serão preenchidos com os valores já existentes.

2. O usuário envia o formulário.

3. O sistema valida os dados informados (desviando para o Fluxo Alternativo, na ocorrência de erros).

   a) O sistema exibe uma mensagem de erro informando que o nome de usuário, ou o e-mail já estão em uso.

4. O sistema cria um novo usuário na base de dados.

4.6.2 UC02 - Visualizar Meus Atestados

Sumário

O objetivo deste caso de uso é o de listar, para o usuário autenticado no sistema, os atestados que já foram emitidos para ele, além de disponibilizar um meio para visualização de detalhes e aquisição de segundas vias dos atestados.

É disponibilizada a opção de visualizar apenas os atestados que ainda estão válidos (dentro do período de vigência).

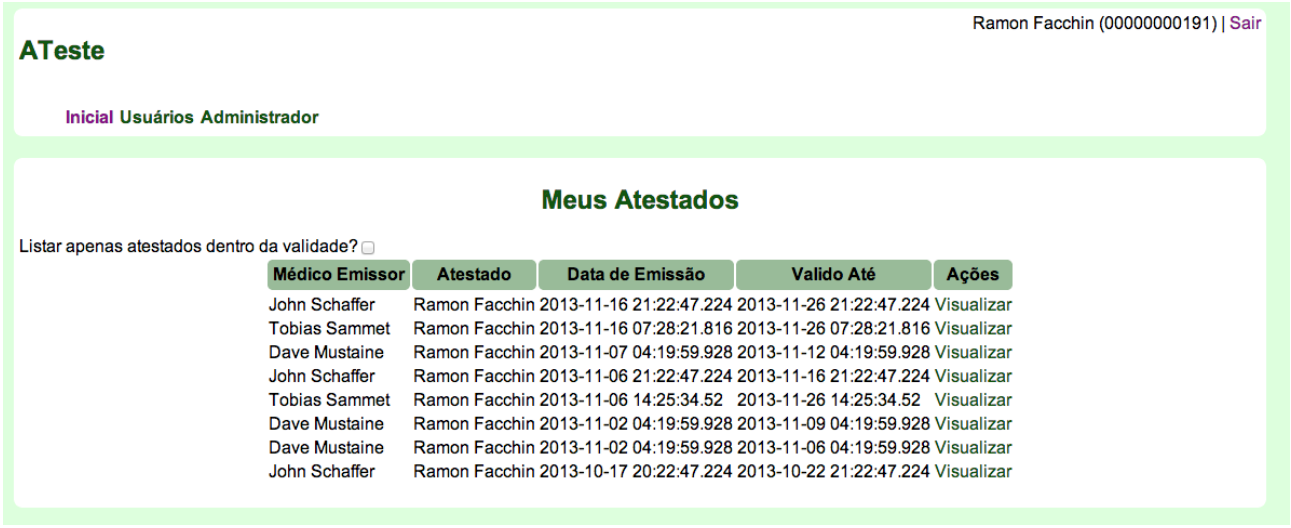

Figura 10 – Visualizar Meus Atestados.

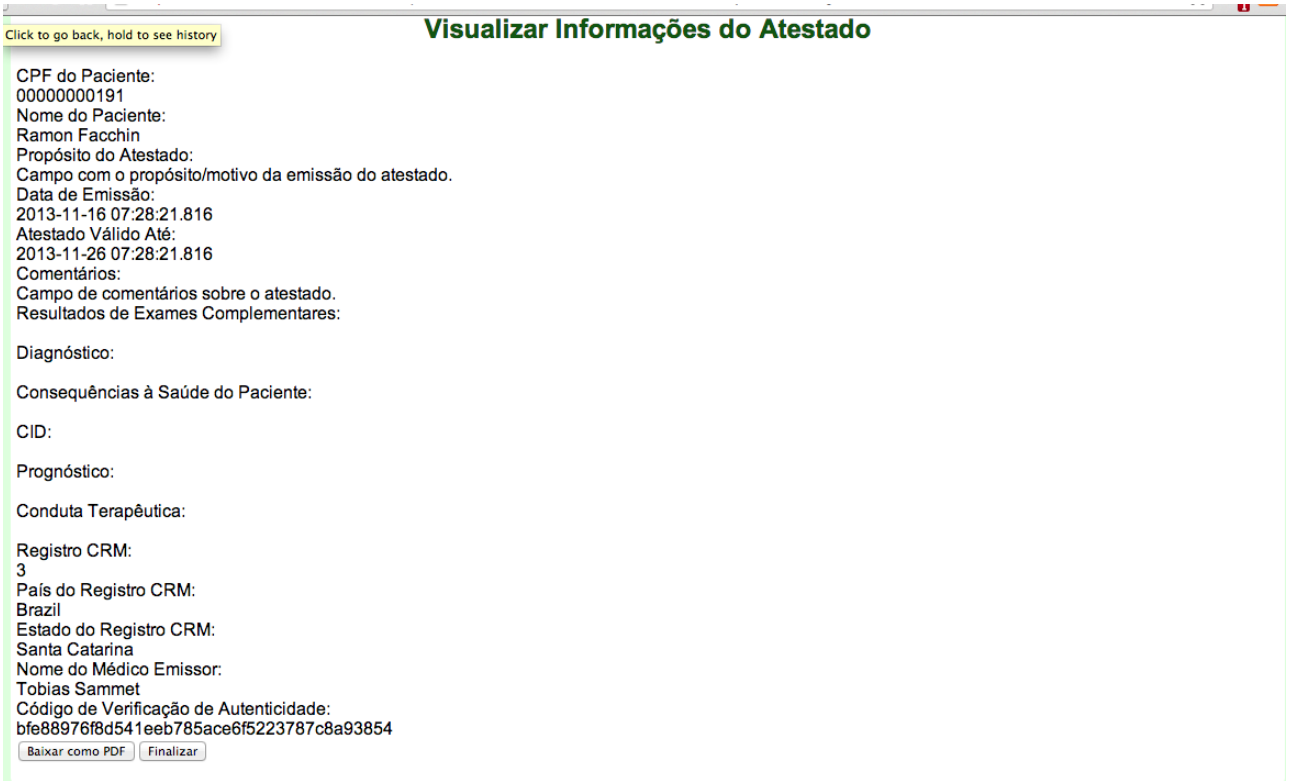

Figura 11 – Visualização de Atestado.

Pré-condições

São pré-condições do Cadastro de Usuário:

- O usuário deve estar autenticado.

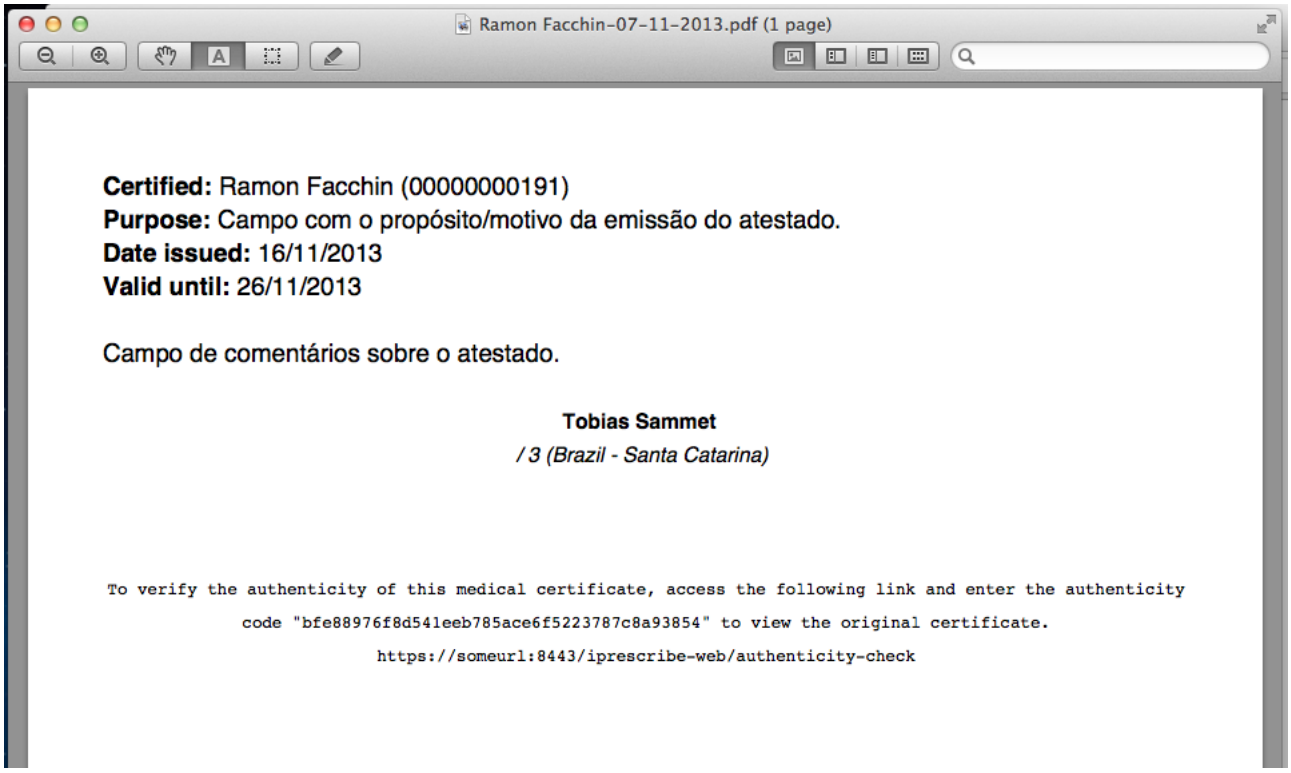

Figura 12 – Visualização de Atestado em PDF.

Fluxo

1. O usuário acessa a tela de listagem.
2. O usuário clica no link "Visualizar"de um dos atestados listados.
3. O sistema exibe os detalhes do atestado selecionado.
4. Vistos os detalhes do atestado, o usuário possui duas opções:
   a) Finalizar a visualização detalhada.
   b) Solicitar o download do atestado em formato PDF.

4.6.3 UC03 - Adicionar Registro do CRM

Sumário

O objetivo deste caso de uso é o de permitir, a um médico, solicitar o cadastro do seu registro num Conselho Regional de Medicina para que possa utilizar seu registro para a emissão de atestados médicos.

Pré-condições

São pré-condições do Cadastro de Usuário:

Figura 13 – Adicionar Registro do CRM.

- O usuário deve estar autenticado.
- O usuário deve ter o perfil de médico.

Fluxo

1. O usuário preenche os dados do formulário.
2. O usuário envia o formulário.
3. O sistema valida os dados informados (desviando para o Fluxo Alternativo, no caso de erros).
    a) O sistema exibe uma mensagem de erro, notificando o usuário de que aquele registro do CRM informado já está em uso.
4. O sistema cadastra um novo registro CRM, que ficará pendente de aprovação por um administrador.

4.6.4 UC04 - Emitir Atestado

Sumário

O objetivo deste caso de uso é o de permitir, a um médico, emitir um atestado médico para um paciente.

Pré-condições

São pré-condições do Cadastro de Usuário:

- O usuário deve estar autenticado.

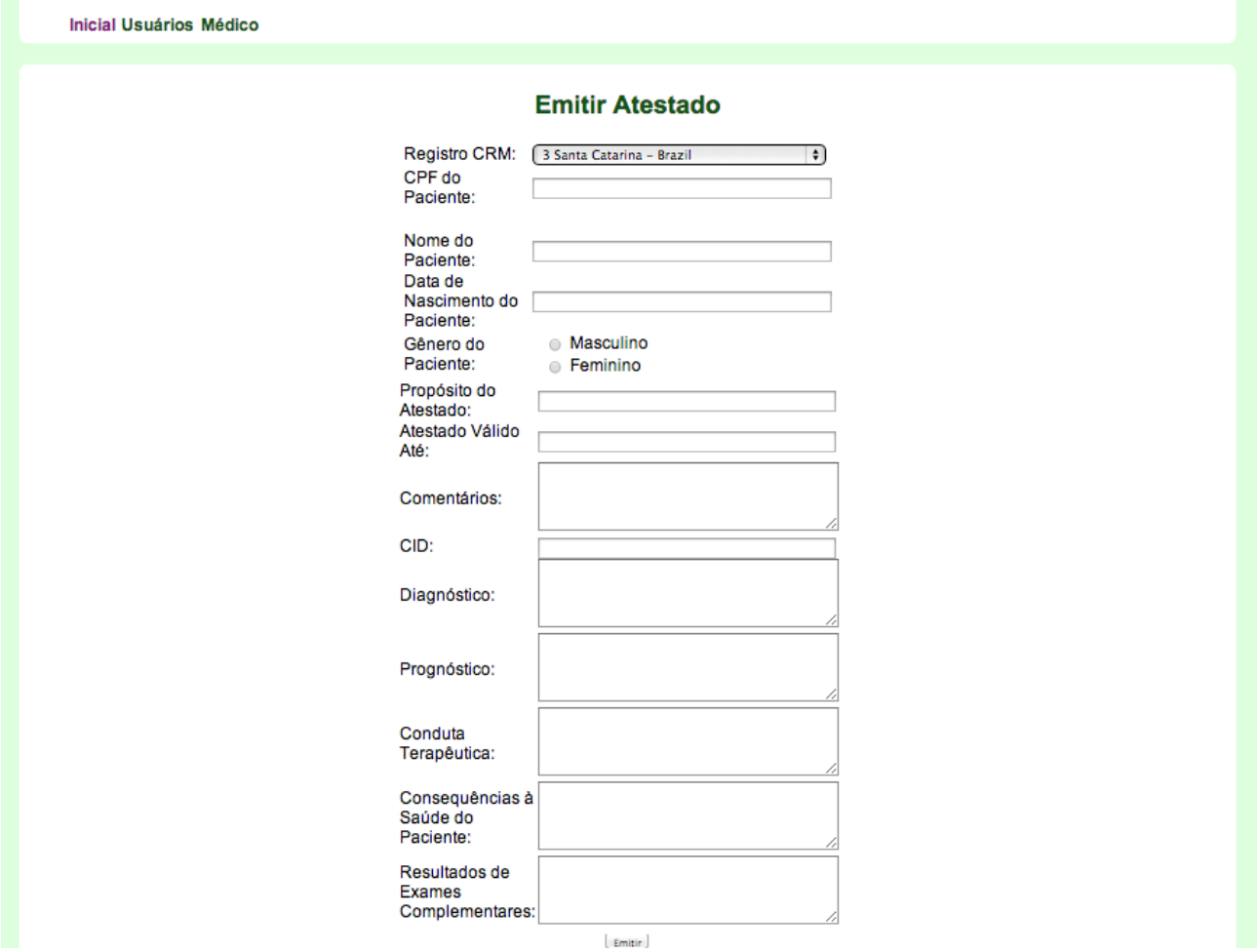

Figura 14 – Emitir Atestado.

- O usuário deve ter o perfil de médico.
- O usuário deve possuir ao menos um registro no CRM já aprovado por um administrador.

Fluxo

1. O usuário seleciona um dos seus registros CRM.
2. O usuário preenche os campos de formulário.
3. O usuário envia o formulário.
4. O sistema valida o formulário enviado.
    a) Se algum campo obrigatório não foi preenchido, ou algum campo foi preenchido de forma incorreta, interrompe-se o fluxo e o sistema exibe uma mensagem de erro.

5. O sistema cria um novo atestado e gera um código de verificação de autenticidade para aquele atestado gerado.
6. O usuário é redirecionado para uma tela de visualização de detalhes do atestado emitido
7. Visualizados os detalhes do atestado, há duas opções:
   a) Realizar download do atestado em formato PDF.
   b) Finalizar a interação.

4.6.5 UC05 - Atestados Emitidos por Mim

Figura 15 – Atestados Emitidos por Mim.

Sumário

O objetivo deste caso de uso é o de disponibilizar, ao médico, um meio de controle para saber quantos atestados já emitiu, quais estão em vigência e possibilitar a emissão de segundas vias.

Pré-condições

São pré-condições do Cadastro de Usuário:

- O usuário deve estar autenticado.
- O usuário deve ter o perfil de médico.

Fluxo

1. O usuário acessa a tela de listagem.
2. O usuário clica no link "Visualizar"de um dos atestados listados.

3. O sistema exibe os detalhes do atestado selecionado.

4. Vistos os detalhes do atestado, o usuário possui duas opções:

   a) Finalizar a visualização detalhada.

   b) Solicitar o download do atestado em formato PDF.

4.6.6 UC06 - Configurações do Sistema

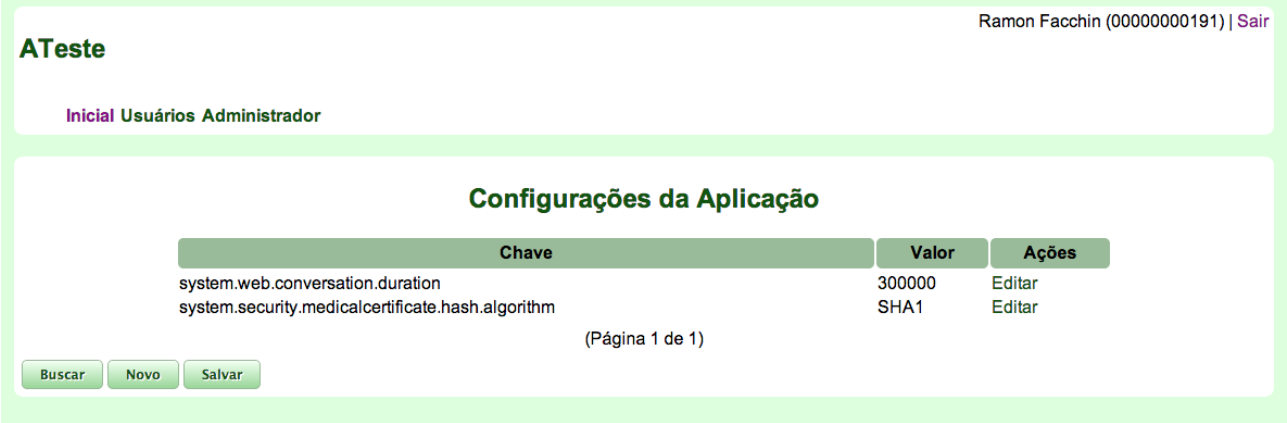

Figura 16 – Configurações do Sistema.

Sumário

O objetivo deste caso de uso é o de permitir, a um administrador do sistema, customizar alguns comportamentos da aplicação em tempo de execução, como a duração de transações e o algoritmo hash utilizado na geração de códigos de verificação de autenticidade.

Pré-condições

São pré-condições do Cadastro de Usuário:

- O usuário deve estar autenticado.
- O usuário deve ter o perfil de administrador do sistema.

Fluxo

1. O usuário acessa a tela de configurações.

2. Na tela de configurações, há 2 opções:

   a) Clicar em "Novo", para adicionar uma nova configuração.

i. O sistema exibe campos editáveis para acrescentar uma nova configuração.
ii. O usuário preenche os campos.
iii. O usuário envia o formulário.
iv. O sistema valida os dados entrados. Se houver erros, o fluxo é interrompido e uma mensagem de erro é exibida.
v. A nova configuração é persistida.

b) Clicar na ação "Editar"de uma das linhas listadas na tabela de configurações.

i. O sistema transforma as células da linha em edição em campos editáveis com os valores atuais.
ii. O usuário altera os valores existentes.
iii. O usuário envia o formulário.
iv. O sistema valida os dados entrados. Se houver erros, o fluxo é interrompido e uma mensagem de erro é exibida.
v. Os valores da configuração são atualizados.

4.6.7 UC07 - Autorizar Registro do CRM

Figura 17 – Autorizar Registro do CRM.

Sumário

O objetivo deste caso de uso é o de permitir, a um administrador do sistema, autorizar ou não autorizar solicitações de inclusão de registro no CRM, feitas por médicos. Permite um nível de controle adicional de acesso por parte do administrador de sistema, já que a existência de um registro no CRM é pré-requisito para realizar a emissão de atestados.

Pré-condições

São pré-condições do Cadastro de Usuário:

- O usuário deve estar autenticado.
- O usuário deve ter o perfil de administrador do sistema.

Fluxo

1. O usuário acessa a tela de autorização.
2. O sistema lista as solicitações de autorização de registro no CRM em aberto.
3. Para cada item listado há duas opções:
   a) O usuário clica em "Autorizar".
      i. O sistema altera o status do registro CRM em questão para autorizado.
      ii. O registro CRM fica disponível para uso do médico solicitante em emissões de atestados.
   b) O usuário clica em "Não Autorizar".
      i. O sistema exclui (fisicamente) o registro CRM que não foi autorizado.
      ii. O registro fica novamente disponível para solicitação por outros médicos.

4.6.8 UC08 - Autenticação no Sistema

Figura 18 – Autenticação no Sistema.

Sumário

O objetivo deste caso de uso é o de permitir, a um visitante, ter uma identidade associada durante o uso do sistema, conferindo-lhe acesso à área restrita.

Pré-condições

São pré-condições da Autenticação no Sistema:

- O usuário não pode estar autenticado.
- O usuário já deve estar cadastrado no sistema.

Fluxo

1. O usuário acessa a tela de login (ou tenta acessar alguma área restrita da aplicação).
2. O usuário preenche seu nome de usuário e senha.
3. O sistema valida o preenchimento dos dados entrados, interrompendo o fluxo se algum campo estiver em branco e exibindo uma mensagem de erro.
4. O sistema valida se o usuário existe, interrompendo o fluxo caso o usuário não exista e exibindo uma mensagem de erro.
5. O sistema valida se a senha daquele usuário está correta, interrompendo o fluxo caso a senha esteja incorreta e exibindo uma mensagem de erro.
6. O sistema autentica o usuário, atribuindo uma identidade a ele.
7. O sistema torna visíveis os itens de menu da área restrita.

4.6.9 UC09 - Desautenticação no Sistema

Sumário

O objetivo deste caso de uso é o de permitir, a um usuário, deixar de ter sua identidade atribuída à sua sessão dentro da aplicação.

Pré-condições

São pré-condições da Desautenticação no Sistema:

- O usuário deve estar autenticado.

Fluxo

1. O usuário clica em "Sair", no canto superior direito da tela.
2. O sistema remove as informações da sessão do usuário.
3. O sistema desvincula o usuário à sua identidade, fazendo com que o usuário volte a ser considerado um visitante.
4. O sistema oculta itens de menu da área restrita.

4.6.10 UC10 - Verificação de Autenticidade de Atestado

Figura 19 – Verificação de Autenticidade de Atestado.

Sumário

O objetivo deste caso de uso é o de permitir, a um usuário ou visitante, verificar se um atestado, emitido pelo sistema, que lhe foi apresentado, é autêntico ou falso.

Não há pré-condições para este caso de uso.

Fluxo

1. O usuário acessa a tela de verificação.
2. O usuário informa o código de verificação de autenticidade do atestado.
3. O usuário envia o formulário.
4. O sistema busca por um registro de atestado médico que possua aquele código de verificação.
    a) Se não for encontrado um registro de atestado médico para o código informado, o fluxo é interrompido e é exibida uma mensagem de erro para o usuário.
    b) Se o registro for encontrado, o sistema oferece download de um atestado, no formato PDF, idêntico ao original gerado, para que o usuário verifique as informações.

5 Apresentação da Aplicação

5.1 Arquitetura da Aplicação

A aplicação proposta contará com 8 módulos diferentes, cada qual com um propósito específico. Juntamente com os módulos serão fornecidos diagramas arquiteturais para melhor compreensão da aplicação.

5.1.1 Módulo *Common*

Este módulo possui classes que não possuem dependências com outros módulos e que são utilitárias para todos os demais módulos do sistema. Entre os estereótipos de classes presentes no módulo *Common*, estão:

- **Constantes**: Classes que definem constantes do sistema. Ou seja, definem valores que não mudam durante a execução do sistema e que serão de refatoração facilitada se forem utilizadas referências para os valores, em vez de utilizar tais valores de forma explícita no código da aplicação.

- **Anotações**: Interfaces que servem como recursos de meta programação, seja para implementação facilitada de padrões de projeto, seja para seguir especificações do Java EE 6 como, por exemplo, na API de validadores.

- **Enumerações**: Java *enums*, que serão de uso geral no sistema.

- **Exceções**: Classes de exceção que poderão ser capturadas em diversos módulos do sistema quando da ocorrência de falhas previstas.

- **Interceptadores**: Classes que implementam um padrão de projeto *Interceptor*, mas de acordo com as recomendações do Java EE 6, possibilitando estender o ciclo de vida de execução de alguns componentes Java EE (como EJBs, por exemplo).

- **Validadores**: Classes cujo propósito de existência é realizar a validação de entradas e que seguem a especificação da *Validation* API do Java EE 6. Um exemplo é o validador de CPF, que possui também uma anotação correspondente.

- **Úteis**: Classes que são compostas unicamente de métodos estáticos cujo objetivo é apenas auxiliar em tarefas simples como conversão e formatação de *Strings* e Datas.

5.1.2 Módulo *Entity*

Este módulo contém as classes resultantes do mapeamento objeto-relacional da aplicação e representa o Modelo de Dados do sistema. Também são definidas enumerações que fazem parte do modelo de dados. O mapeamento objeto-relacional fica por conta das anotações fornecidas pela JPA, cujo *provider* escolhido foi o Hibernate, que já vem embarcado no JBoss 7.

5.1.3 Módulo *Client*

Este módulo tem, por finalidade, disponibilizar classes para comunicação com a aplicação (remota, inclusive), classificadas em:

- **Service Locator**: Implementação de um padrão de projeto. Tem o objetivo de localizar EJBs quando o código cliente for executado de fora de um contexto que possibilite a injeção dos EJBs via contêiner.

- **Value Objects**: POJOs e DTOs necessários durante a execução de serviços da camada de modelo.

- **Interfaces de EJBs (Services)**: São as interfaces Java que definem o contrato de EJBs (operações que obrigatoriamente serão suportadas). São separadas interfaces locais e remotas para cada EJB desenvolvido na aplicação, sendo que as interfaces locais são especializações das remotas. Essa abordagem foi escolhida para facilitar a manutenção e para possibilitar a adoção de um modelo restritivo no qual as operações críticas (e que não há interesse em disponibilização para módulos externos à aplicação) são invisíveis a clientes remotos. Por uma convenção adotada dentro do sistema, foi criada a anotação *DefaultBeanName* para dizer, ao *Service Locator*, qual o radical do nome JNDI a ser buscado no momento do *lookup* de um EJB, caso nenhuma outra informação sobre o JNDI seja fornecida. Toda interface *remota* de EJB deve ser anotada com *@DefaultBeanName*.

5.1.4 Módulo EJB

Este módulo comporta as classes que executam a lógica de negócio da aplicação. São elas:

- **DAOs (interfaces)**: Interfaces java que definem o contrato dos DAOs, de forma análoga às interfaces de EJBs do módulo *Client*, mas são internas ao módulo EJB, uma vez que não é interessante expôr os DAOs da aplicação.

- **DAOs**: Classes que executam os métodos que requerem interação com repositórios de dados, como SGBDs, por exemplo. Intermediam a lógica de persistência da aplicação e executam consultas à base de dados.

- **Úteis**: Classes utilitárias que são utilizadas apenas dentro do contexto do módulo EJB, com a finalidade de reduzir replicação de código.

- **Services**: São as implementações dos EJBs definidos pelas interfaces do módulo *Client*. São os brokers entre a camada de controle e a camada de modelo e gerenciam o acesso aos DAOs, executando regras de negócio nesse processo. Uma classe *Service* está mais ligada a uma funcionalidade da aplicação do que a uma Entidade, diferentemente dos DAOs, e agrega, dentro de si, referências para DAOs e, inclusive, para outros *Services*.

A Figura 20 ilustra a forma como se dá a implementação das classes que implementam a lógica de negócio da aplicação, bem como sua interação com o módulo *Client*.

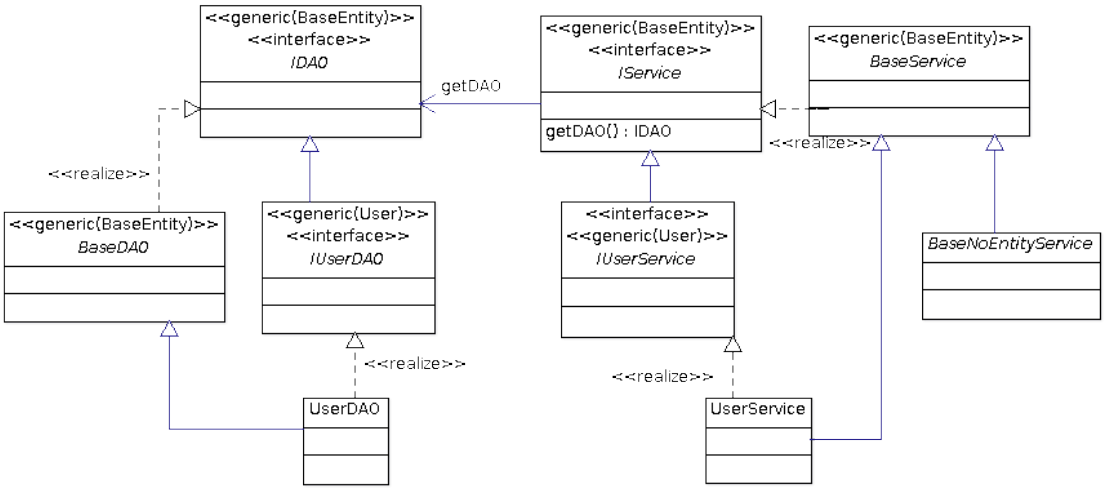

Figura 20 – Diagrama de classes (resumido) da Arquitetura de Serviços.

Observe que os DAOs sempre implementam a interface *IDAO<T extends BaseEntity>*, garantindo, no mínimo a implementação dos seguintes métodos:

- *findById*: Realiza uma busca por chave primária para a entidade definida pela classe *T*.

- *list*: Realiza uma listagem de todas as instâncias da classe *T* na base de dados. Podem ser informados: o deslocamento da listagem (índice a partir do qual a listagem

será feita, desconsiderando os resultados anteriores), o número máximo de resultados desejados, e se apenas as instâncias ativas (que não sofreram exclusão lógica do sistema) devem ser listadas, apenas inativas, ou ambas.

- *listCount*: Informa quantas instâncias existem da classe T, podendo-se optar pela contagem de ativas, inativas, ou ambas.
- *listAll*: Equivale à chamada de *list*, mas sem especificar deslocamento, nem número máximo de resultados e listando tanto instâncias ativas como inativas.
- *listAcative*: Equivale à chamada de *list*, mas sem especificar deslocamento, nem número máximo de resultados e listando apenas instâncias ativas.
- *listInactive*: Equivale à chamada de *list*, mas sem especificar deslocamento, nem número máximo de resultados e listando apenas instâncias inativas.
- *persist*: Salva uma nova instância da classe T na base de dados.
- *update*: Atualiza uma instância da classe T na base de dados.
- *remove*: Remove uma instância da classe T da base de dados.
- *getTableName*: Retorna o nome da entidade da classe T para ser utilizado em consultas JPQL.
- *getSchemaName*: Retorna o nome do esquema da entidade da classe T para ser utilizado em consultas JPQL.

Para a interface obrigatória dos DAOs é oferecida uma implementação padrão com a classe *BaseDAO<T extends BaseEntity>*. Essa implementação se baseia nos parâmetros genéricos passados na assinatura da classe e nos nomes de classe para as implementações padrão oferecidas.

As classes de serviço seguem um padrão similar ao dos DAOs, mas com duas diferenças essenciais: (1) As interfaces estão definidas no módulo *Client* e (2) há, pelo menos, duas interfaces para cada implementação de serviço. Inclusive, os métodos que devem ser suportados por uma implementação de serviço são quase os mesmos que os métodos de DAOs, à exceção de:

- *getDAO*: Deve retornar uma instância do DAO que trata da principal entidade envolvida naquele serviço.
- *save*: Uma junção dos métodos *persist* e *save* do DAO, que realiza a verificação se uma entidade deve ser criada ou apenas atualizada por já existir na base de dados, evitando confusões nas camadas superiores da aplicação e replicação de código.

Para os serviços são oferecidas duas implementações padrão.

BaseService define, de forma similar aos DAOs, implementações padrão para os métodos da interface, mantendo o método *getDAO* abstrato, pois a implementação dele dependerá de injeções de dependência que só podem ser feitas pela classe final, uma vez que o CDI não trabalha bem com suporte a tipos genéricos.

BaseNoEntityService funciona de forma similar à *BaseService*, porém, é indicada para serviços em que não se pode definir uma entidade principal envolvida, já que interage com diversas classes diferentes.

5.1.5 Módulo Web

É o módulo que agrega as camadas de visão e de controle da aplicação. Por conta da maior produtividade e reutilização de código possibilitada (com o uso da linguagem de expressão JSF aliada aos *Managed Beans*), foi optado por utilizar o framework JSF na sua implementação.

Estabeleceu-se, como padrão, utilizar um *Managed Bean* por funcionalidade do sistema, uma vez que o controle de fluxo e a lógica de interface, se tratados de forma muito genérica com JSF oferecem mais dificuldades no *refactoring* do que quando criados cada um para tratar um caso específico. Para realizar tarefas que são explicitamente comuns a todas as funcionalidades do sistema, foi criada uma superclasse, *BaseMB* (e algumas especializações, como *BaseCrudMB e BaseConversationMB*), além de outros *Managed Beans* de função utilitária (estes, não vinculados diretamente a uma funcionalidade), para servirem de forma análoga às classes utilitárias do módulo *Common*, mas que possam ser invocados com linguagem de expressão, nas páginas.

Embora o Java EE 6 tenha acabado com a necessidade de se definir um arquivo de configuração de aplicação Web (o *web.xml*), para a implementação da aplicação proposta foi necessário criá-lo por conta dos parâmetros de inicialização do *framework* de segurança selecionado, o Apache Shiro. Embora fosse possível, também, uma configuração programática do Shiro, a configuração declarativa oferece uma melhor integração com a Servlet API. Cabe detalhar um pouco mais o propósito dos *Managed Beans* utilitários:

- *AuthenticationMB*: Responsável por armazenar as informações de sessão do usuário, uma vez que ele for autenticado. Para que isso seja possível, o escopo definido foi o *SessionScoped*.

- *EnumSelects*: Definido como *ApplicationScoped*, possui apenas o propósito de prover listas de enumerações, sempre constantes e idênticas em todos os contextos da aplicação e independente de sessão do usuário.

- *InternationalizationMB*: Também definido como *SessionScoped*, armazena informações do locale do usuário e recupera mensagens internacionalizadas. Embora esteja fora do escopo a internacionalização da aplicação desenvolvida, o suporte à internacionalização foi implementado.
- *UrlMB*: Funciona de maneira similar a uma classe de constantes, mas permitindo invocação via linguagem de expressão. Além disso, as invocações de URL feitas através deste *Managed Bean* podem acionar diretivas para inicializar uma conversação (escopo do CDI que possui ciclo de vida maior que o de requisição e menor que o de sessão), bem como finalizá-la.

No módulo Web estão implementadas as regras de segurança da aplicação, mas as especificidades dessa implementação serão descritas apenas na seção *Arquitetura de Segurança*.

5.1.6 Módulo EAR

É o módulo que define a estrutura de empacotamento de todos os outros módulos da aplicação, e que comporta as dependências (bibliotecas) que não são fornecidas pelo Servidor de Aplicação.

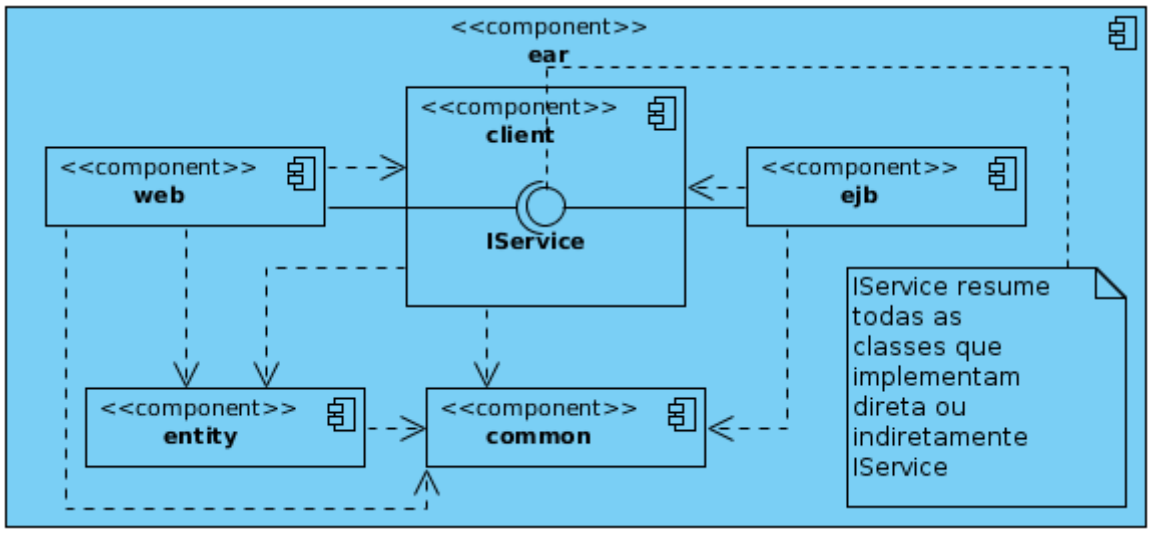

Figura 21 – Diagrama de componentes da aplicação.

Como pode ser observado pela Figura 21, o pacote EAR é composto de três módulos básicos (*common, client e entity*, referenciados por todos os demais módulos da aplicação), dois módulos de testes (*service-test* para teste de EJBs e *samplews* como um exemplo de cliente de Web Service), e dois de integração (sendo um, *web* o agregador das camadas de visão e de controle, e o outro, *ejb*, o elo principal da camada de modelo). Os módulos de integração se comunicam através das interfaces disponibilizadas pelo módulo *client*, nunca diretamente.

5.2 Segurança

A implementação dos mecanismos de segurança da aplicação se deu através da adaptação da aplicação Web para utilizar o Apache Shiro e, posteriormente, foram acrescentados os mecanismos de autorização, também utilizando o Shiro, mas comunicando-se com a API do PERMIS, que é a encarregada de realizar as decisões de autorização.

5.2.1 Configuração do Apache Shiro e Autenticação

Para habilitar o Apache Shiro na aplicação Web, foi necessário acrescentar algumas diretivas no descritor da aplicação web (o arquivo *web.xml*), criar um arquivo de configuração *shiro.ini* no *classpath* da aplicação web e acrescentar as dependências (bibliotecas) do Apache Shiro na aplicação. O arquivo *web.xml* resultante pode ser visualizado no Apêndice C, na seção C.2.

Com isso, está configurado o *listener* que se encarregará de inicializar o Apache Shiro quando da inicialização da aplicação Web e estará habilitado um *ServletFilter* que interceptará todas as requisições feitas à aplicação, para que sejam tratadas pelo Shiro.

Outro arquivo de configuração necessário é o *shiro.ini*. Inicialmente, foram realizadas apenas as configurações necessárias para autenticação, deixando questões relativas à autorização para um momento posterior. O arquivo *shiro.ini* que contém tais configurações pode ser visualizado no Apêndice C, na seção C.1.

A partir da configuração obtida, cabem as seguintes observações:

- Foi criada uma classe *IPrescribeRealm*, que implementa a interface *Realm* do Apache Shiro. Essa classe é a encarregada de conduzir o processo de autenticação, e foi a única classe da API do Shiro que precisou ser implementada para possibilitar a autenticação (todas as demais funcionalidades foram implementadas apenas utilizando as classes já disponibilizadas pela API).
- *ssl, logout, anon e authc* são filtros de URL disponibilizados pelo Shiro para declarar regras de navegação.
- *logout* é o filtro que indica uma URL que deve desautenticar o usuário.
- *anon* é o filtro que indica que uma URL (ou padrão de URL) deve ser acessível a usuários não autenticados (visitantes).
- *authc* é o filtro que indica que uma URL (ou padrão de URL) deve ser acessível apenas a usuário autenticados.
- *ssl* é o filtro que faz com que seja exigida a comunicação HTTPS para aquele padrão de URL.

- As diretivas declaradas no arquivo de configuração do Shiro seguem uma regra de priorização das diretivas declaradas antes. Portanto, a regra "*/restricted/** = authc*"é autoritária em relação à regra "*/** = anon*", por exemplo.

Com essa configuração, a aplicação está habilitada a restringir o acesso ao contexto "*/restricted*"apenas a usuários autenticados, fazendo com que qualquer tentativa de acesso a esse contexto gere um redirecionamento para a página de autenticação do usuário.

5.2.2 Autorização

No entanto, apenas a autenticação não é suficiente para fazer, de forma ideal, o controle de acesso da aplicação especificada. Ainda devem ser levadas em consideração as regras de autorização, isto é, exigir os perfis de Médico e de Administrador do Sistema para permitir o acesso a algumas das páginas da aplicação.

A abordagem escolhida para implementar os mecanismos de autorização foi a do CABP, estudado no primeiro capítulo do presente trabalho. O primeiro passo para implementar o CABP é ter, com clareza, quais são os papéis que deverão existir no sistema. Para isso, os perfis de Administrador do Sistema e Médico (definidos na especificação) foram mapeados nos papéis (Roles) *admin* e *md*, respectivamente.

Para que os requisitos de segurança especificados, sejam atendidos, os contextos "*/restricted/administrator*"e "*/restricted/medicaldoctor*"devem exigir, do usuário autenticado, os papéis *admin* e *md*, respectivamente. Para isso, o arquivo de configuração *shiro.ini* teve de ser atualizado, assumindo a configuração apresentada na seção C.3, no Apêndice C.

Observa-se que, em relação à configuração anterior, foram acrescentadas apenas duas linhas no mapeamento de URLs, para que os contextos citados anteriormente exigissem determinados papéis para que o acesso seja liberado.

Com essa configuração, também foi necessária uma alteração na classe *IPrescribeRealm*, para que passasse a estender a classe *AuthorizingRealm*, do Apache Shiro e, por consequência, implementasse o método *doGetAuthorizationInfo*, o qual realiza a busca das permissões do usuário que está autenticado. É dentro desse método que é feita a invocação do PERMIS.

5.2.2.1 PERMIS

O PERMIS oferece um conjunto de ferramentas com a finalidade de implementar o controle de acesso em aplicações. Para que fosse possível instalar o mecanismo de decisão de controle de acesso do PERMIS, foi necessário, antes de tudo, instalar um serviço de diretórios LDAP. A finalidade do LDAP, no contexto do PERMIS, é a de armazenar

as políticas de controle de acesso geradas e armazenar, também, os certificados de atributo emitidos. Para esta aplicação, foi instalado o serviço na mesma máquina em que a aplicação executa, embora isso não seja necessário.

O segundo passo para a configuração, envolve a criação de uma política de controle de acesso. Para isso, é disponibilizada a ferramenta *Policy Editor*, que oferece um *wizard* para criação da política, que resultará em um arquivo XML, como o da aplicação proposta, que pode ser visualizado no Apêndice A deste trabalho.

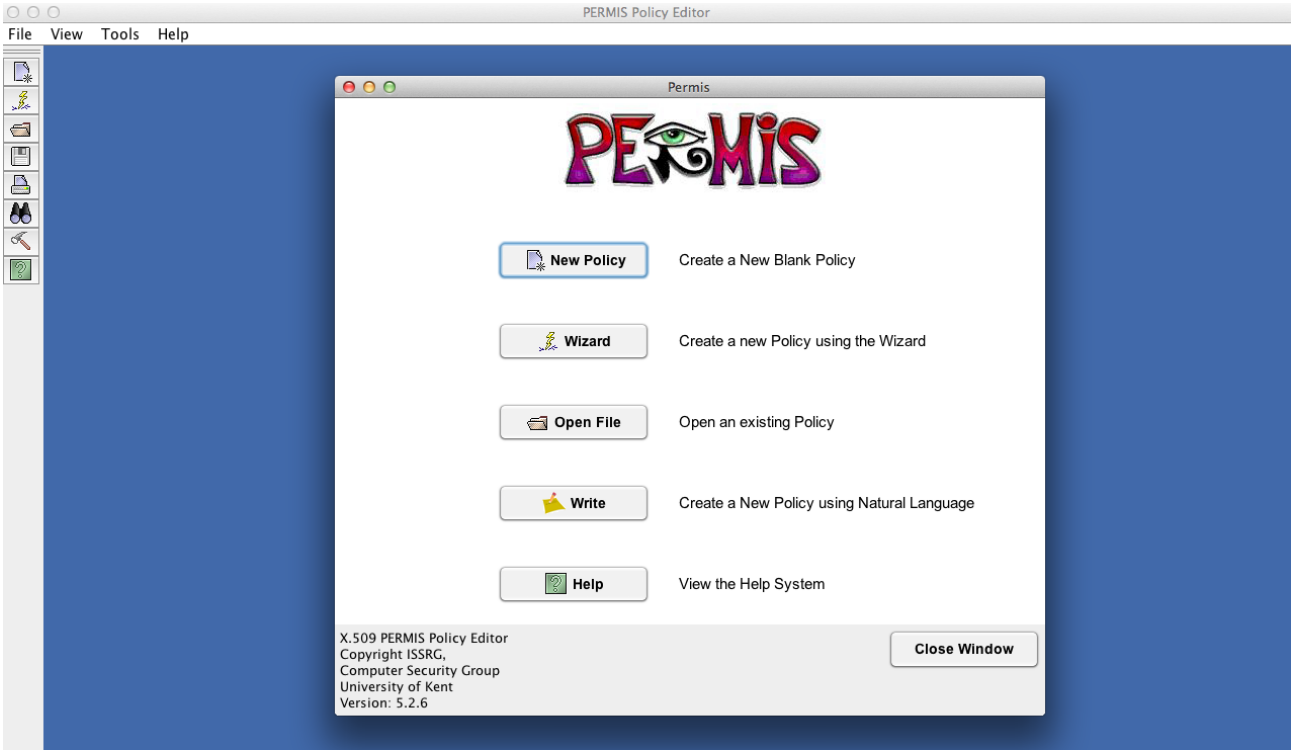

Figura 22 – Editor de Política de Controle de Acesso do PERMIS.

O terceiro passo envolve a emissão de certificados de atributo para os usuários do sistema. A aplicação possui alguns usuários pré-configurados, cujas informações principais constam na Tabela 1, que foi utilizada como base para a geração dos CAs. O próprio PERMIS já fornece um certificado de chave pública para realizar a assinatura dos CAs a serem emitidos. Para que não seja necessário criar uma ICP nova apenas para a prova de conceito, foi optado por utilizar o certificado fornecido. O atributo definido na política (e padrão do PERMIS) é o *permisRole*, cujo valor será o *alias* dos papéis definidos para esta aplicação.

Terminados os passos de criação e armazenamento da política e dos certificados de atributo, o passo seguinte foi o uso da API Java do PERMIS dentro do código da aplicação proposta. Para isso, devem ser informados os parâmetros de conexão ao serviço LDAP, identificador único da política de segurança a ser aplicada e as questões relativas ao contexto de verificação do atributo (dados do usuário - *Principal* - e nome do recurso

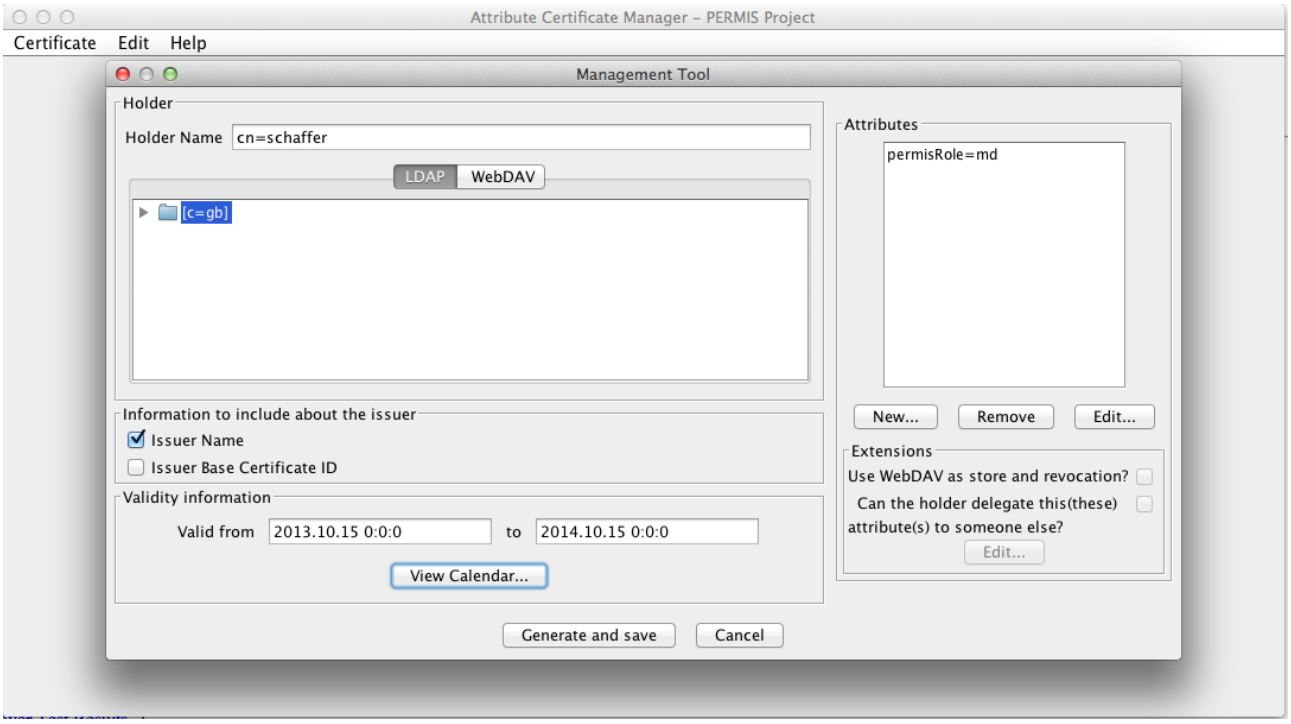

Figura 23 – Gerenciador de Certificados de Atributo do PERMIS.

Tabela 1 – Usuários da aplicação

Nome Completo	Nome de Usuário	Papéis (permisRole)
Ramon Facchin	ramonfacchin	admin
John Schaffer	schaffer	md
Tobias Sammet	sammet	md
Dave Mustaine	mustaine	md
James Hetfield	hetfield	-
Anders Friden	friden	-
Mariane Flôr	marianeflor	-

sendo requisitado - que, no caso em estudo, é a URL restrita). Para mais detalhes, o guia de instalação e testes do PERMIS (INFORMATION SYSTEMS SECURITY RESEARCH GROUP (ISSRG), 2008) auxilia de forma incremental.

Outro detalhe importante é o de que as regras de controle de acesso na aplicação já estão descritas na configuração do Apache Shiro e, por isso, optou-se por invocar o PERMIS apenas para descobrir se um usuário possui um papel declarado ou não. Isso implica que só será testada uma restrição de cada papel (apesar de terem sido definidas duas restrições para administrador e quatro para médico), com a finalidade de identificar se o papel existe ou não para o usuário sendo testado. Se o papel existir para o usuário, ele é adicionado ao conjunto de *Roles* da classe *SimpleAuthorizationInfo* que, por sua vez, é invocada pela API do Apache Shiro para a verificação do papel associado. Uma simplificação da troca de mensagens ocorrida para a interação do Apache Shiro com o Permis é ilustrada na Figura 24.

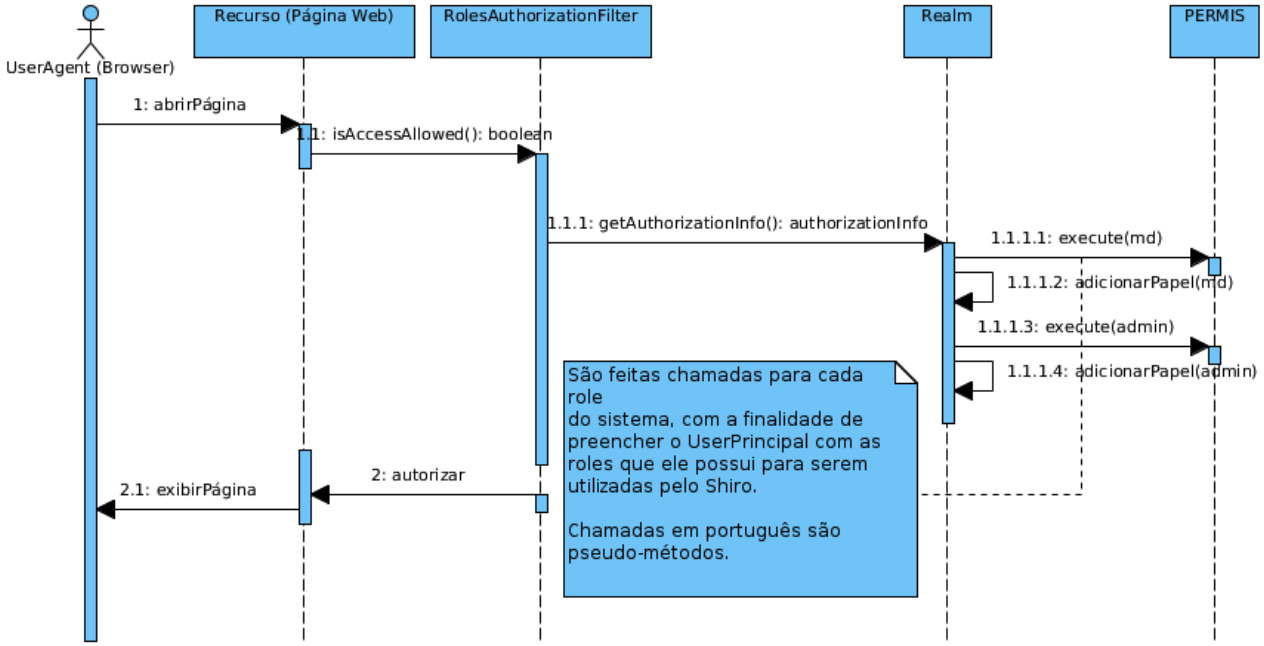

Figura 24 – Diagrama de sequência que resume a interação entre o usuário, a aplicação, o Shiro e o PERMIS (omitindo vários detalhes, apenas para se ter uma visão geral do fluxo).

Optou-se por essa abordagem pelo fato de a aplicação ser simples e pequena, além de estar sendo testada em um cenário no qual a fonte de autoridade e a aplicação estão no mesmo domínio. No entanto, se a fonte de autoridade fosse externa, seria necessário um maior detalhamento dos recursos protegidos pela política de controle de acesso.

6 Resultados

Especificada a aplicação e apresentada sua arquitetura, bem como as opções de implementação dos mecanismos de segurança, devem ser apresentados os resultados de execução. Os seguintes critérios foram considerados:

1. Na condição de visitante, a tentativa de acessar uma página que requer autenticação deve redirecioná-lo para a página de autenticação.
2. Na condição de usuário autenticado, ao tentar acessar uma página que requer um determinado papel, caso o usuário não o possua, deve gerar um erro HTTP 401 (se a autenticação fosse gerenciada pelo contêiner, o erro gerado seria o 403, mas como a autenticação é feita pela aplicação, o contêiner assume que o usuário nem mesmo está autenticado).

Nas Tabelas 2 e 3 estão relacionadas as páginas da aplicação e os perfis de usuários que devem possuir acesso. Há uma coluna para cada perfil previsto na aplicação. As colunas de perfis estão preenchidas com *OK* nos casos em que houve sucesso no acesso, *NOK* nos casos em que o usuário foi redirecionado para a página de autenticação, *Erro 401* nos casos em que o usuário não possui permissões para acessar a página e *N/A* nos casos em que o teste não faz sentido.

Um código de *Erro 401* significa que a requisição para o servidor requer credenciais de autenticação do usuário para conceder o acesso. Se o cliente já está fornecendo as credenciais de autenticação, significa que o servidor não reconhece aquelas credenciais e, portanto, está recusando o acesso ao recurso. No entanto, no cenário proposto, o usuário já está autenticado, mas não possui as permissões necessárias para acessar o recurso, o que caracteriza um *Erro 403*. Então, por que não é apresentado o *Erro 403*, em vez de o *Erro 401*? Esse comportamento se deve ao fato de o Apache Shiro persistir as informações de sessão nativamente. Ou seja, o contexto do *servlet* não está ciente de que o usuário está autenticado, já que a autenticação não é feita através do servlet container, mas de uma aplicação que tem seus próprios mecanismos para persistir os dados da sessão do usuário. Portanto, é como se o usuário não estivesse autenticado, para o *servlet*, embora para o Apache Shiro ele já esteja autenticado.

A Tabela 4 evidencia o sucesso na execução para todos os usuários testados. Para evitar redundância na exibição dos resultados, foram testados apenas o visitante e mais 3 usuários (Ramon Facchin, John Schaffer e Anders Friden) - conjunto já considerado contemplativo, por abordar todos os perfis levantados. A associação dos usuários testados com seus respectivos perfis pode ser verificada na Tabela 1.

Tabela 2 – Permissões por páginas da aplicação (Visitante e Usuário Comum).

Caso	**Página**	**Visitante**	**Usuário Comum**
1	/index.jsf	OK	OK
2	/login.jsf	OK	N/A
3	/public/user/authcheck.jsf	OK	OK
4	/public/user/register.jsf	OK	OK
5	/restricted/user/mycertificates.jsf	NOK	OK
6	/restricted/certificate/view-certificate.jsf	NOK	OK
7	/restricted/administrator/configurations.jsf	NOK	Erro 401
8	/restricted/administrator/license-request.jsf	NOK	Erro 401
9	/restricted/medicaldoctor/add-license.jsf	NOK	Erro 401
10	/restricted/medicaldoctor/certify.jsf	NOK	Erro 401
11	/restricted/medicaldoctor/mycertificates.jsf	NOK	Erro 401
12	/restricted/medicaldoctor/view-certificate.jsf	NOK	Erro 401

Tabela 3 – Permissões por páginas da aplicação (Administrador e Médico).

Caso	**Página**	**Administrador**	**Médico**
1	/index.jsf	OK	OK
2	/login.jsf	N/A	N/A
3	/public/user/authcheck.jsf	OK	OK
4	/public/user/register.jsf	OK	OK
5	/restricted/user/mycertificates.jsf	OK	OK
6	/restricted/certificate/view-certificate.jsf	OK	OK
7	/restricted/administrator/configurations.jsf	OK	Erro 401
8	/restricted/administrator/license-request.jsf	OK	Erro 401
9	/restricted/medicaldoctor/add-license.jsf	Erro 401	OK
10	/restricted/medicaldoctor/certify.jsf	Erro 401	OK
11	/restricted/medicaldoctor/mycertificates.jsf	Erro 401	OK
12	/restricted/medicaldoctor/view-certificate.jsf	Erro 401	OK

Tabela 4 – Resultados da execução.

Caso	**Usuário**	**Resultado**
1	Visitante	OK
2	Visitante	OK
3	Visitante	OK
4	Visitante	OK
5	Visitante	NOK
6	Visitante	NOK
7	Visitante	NOK
8	Visitante	NOK
9	Visitante	NOK
10	Visitante	NOK
11	Visitante	NOK
12	Visitante	NOK
1	Ramon Facchin	OK
2	Ramon Facchin	N/A
3	Ramon Facchin	OK
4	Ramon Facchin	OK
5	Ramon Facchin	OK
6	Ramon Facchin	OK
7	Ramon Facchin	OK
8	Ramon Facchin	OK
9	Ramon Facchin	Erro 401
10	Ramon Facchin	Erro 401
11	Ramon Facchin	Erro 401
12	Ramon Facchin	Erro 401
1	Anders Friden	OK
2	Anders Friden	N/A
3	Anders Friden	OK
4	Anders Friden	OK
5	Anders Friden	OK
6	Anders Friden	OK
7	Anders Friden	Erro 401
8	Anders Friden	Erro 401
9	Anders Friden	Erro 401
10	Anders Friden	Erro 401
11	Anders Friden	Erro 401
12	Anders Friden	Erro 401
1	John Schaffer	OK
2	John Schaffer	N/A
3	John Schaffer	OK
4	John Schaffer	OK
5	John Schaffer	OK
6	John Schaffer	OK
7	John Schaffer	Erro 401
8	John Schaffer	Erro 401
9	John Schaffer	OK
10	John Schaffer	OK
11	John Schaffer	OK
12	John Schaffer	OK

7 Conclusão

7.1 Considerações Finais

A reutilização de mecanismos de autenticação em aplicações Web já é uma realidade, observável pelos inúmeros exemplos de aplicações que não mais exigem um cadastro para que sejam utilizadas, mas optam por integrar-se a ferramentas que já oferecem uma API de autenticação, como *Google Accounts* ou o próprio *Facebook* (seja via Oauth ou OpenID, por exemplo).

Com o uso de Certificados de Atributo é possível chegar-se em um cenário parecido para a Autorização, uma vez que o objetivo de uma Infraestrutura de Gerenciamento de Privilégios é o de desonerar as aplicações com questões relativas à Autorização, pois nem sempre o administrador de uma aplicação é o melhor agente para definir que permissões os usuários devem ter, conforme citado e justificado anteriormente no trabalho. Busca-se, portanto, maior interoperabilidade na implementação de segurança nos sistemas.

Este trabalho contribui para a compreensão do funcionamento de uma Infraestrutura de Gerenciamento de Privilégios e de como se dá o uso de Certificados de Atributo para controlar o acesso a recursos de aplicações.

Bastando confiar no assinante dos certificados de atributo armazenados no serviço de diretório LDAP controlado pelo PERMIS, tem-se uma maneira de tornar a gestão de permissões de usuários externa à aplicação, permitindo que diferentes entidades possam emitir certificados de atributo que determinam a que recursos quais usuários possuem acesso.

Com o estudo das bibliotecas e ferramentas para auxiliar o desenvolvimento da aplicação "Prova de Conceito"proposta, ficou evidente que implementar os mecanismos de controle de acesso em uma aplicação já não é um desafio crítico como outrora, sendo possível uma configuração quase que exclusivamente declarativa para se ter Autenticação e Autorização operando de maneira funcional dentro de um sistema. Basta observar que tudo que a aplicação fez foi conectar-se com o serviço de Verificação de Privilégios do PERMIS, cabendo o resto do trabalho a uma infraestrutura externa à aplicação. Em um cenário mais próximo do real, por exemplo, poderiam existir duas entidades diferentes, com seu próprio certificado de chave pública, assinando os certificados de atributo utilizados pela aplicação - uma entidade para controlar o papel de administrador, enquanto que a outra controlaria a emissão de atributos para o papel de médico, sem que fossem necessárias grandes alterações (seria apenas necessário que a aplicação aceitasse os certificados de ambas as entidades utilizados para assinar os certificados de atributo).

No entanto, o uso de Certificados de Atributo, apesar de desonerar a aplicação final de esforços com a implementação de mecanismos de segurança, no que depender dos frameworks existentes, como demonstrado por Guilhen (2008), ainda estão muito dependentes de configuração, e muitos desses frameworks nem mesmo cumprem todas as recomendações de Farrel, Housley e Turner (2010). No caso do PERMIS, por exemplo, não há a liberdade de escolha do atributo que será definido nos certificados - apenas é oferecido um conjunto pré-existente de atributos, dentre os quais devem ser selecionados os que serão utilizados. Da mesma forma, a dependência de serviços de diretório LDAP tornam o uso desse tipo de solução menos intuitivo e bastante restritivo para sua adoção, seja pelo risco da exposição de um serviço de diretórios LDAP ao grande público ou pelo fato de o usuário final não poder definir um atributo.

Com isso, fica evidente que o desafio para oferecer maior interoperabilidade nos mecanismos de controle de acesso está na implementação de uma solução flexível de gerenciamento de privilégios, que permita atender cenários tanto de acesso público quanto de acesso privado, oferecendo a possibilidade de definição dos atributos e de armazenamento de políticas e de certificados de diferentes formas. Além disso, a configuração de um ambiente propício a se comunicar com uma IGP deve ser tão simples quanto possível, para aumentar o interesse na adoção dessa solução.

Por fim, a aplicação desenvolvida atendeu ao desafio de oferecer uma implementação de referência para outros desenvolvedores interessados em externalizar os mecanismos de decisão de controle de acesso utilizando-se de certificados de atributo, além de disponibilizar um protótipo funcional de aplicativo de emissão de atestados médicos eletrônicos que, se evoluído, pode tornar-se um serviço de utilidade pública.

7.2 Trabalhos Futuros

Ficam abertas, portanto, algumas possibilidades de se prosseguir com o trabalho iniciado.

Mantém-se necessário o estudo de outras ferramentas de implementação de controle de acesso baseado em Certificados de Atributo. O presente trabalho explorou apenas uma, dentre muitas.

Outro ponto de extensão da solução proposta seria o uso de XACML (*eXtensible Access Control Markup Language*) para descrição de políticas de controle de acesso e processamento de requisições - algo em maior conformidade com padrões internacionais.

Também se faz necessário continuar o trabalho da parte em que mais houve carência, que é a extensão do próprio PERMIS, de forma a permitir definição de nomes de atributos, ou ainda, que políticas de segurança e certificados de atributo possam ser

persistidos em sistemas de arquivos, bancos de dados relacionais etc.

Saindo do escopo de Segurança da Informação e passando ao de Desenvolvimento de Software, é possível uma melhoria da arquitetura de sistema proposta e a agregação de novas funcionalidades, transformando o protótipo de aplicação proposto em um sistema completo e em maior conformidade com normas e leis existentes, além de o suporte a prontuários médicos e prescrições médicas ser algo bastante desejável, em se tratando da área médica. Falta tratar, também, a restrição que só permite que usuários possuidores de CPF utilizem a aplicação. Para isso, deve-se ter em vista um modelo de dados que não permita ambiguidades, mas que seja flexível o suficiente para que não se perca a utilidade em função da pouca conveniência do excesso de burocracia. Além disso, é possível ampliar o escopo para comportar a expressa concordância do paciente com a divulgação do CID, entre outras possibilidades.

Referências

APACHE FOUNDATION. *Apache Shiro Reference Documentation*. Apache Foundation, 2013. Disponível em: <http://shiro.apache.org/reference.html>. Acesso em: 12 set. 2013. Citado 4 vezes nas páginas 13, 48, 49 e 50.

BEZERRA, E. L. *Introdução a Certificado de Atributo*. Wordpress, 2011. Disponível em: <http://ernandeslb.files.wordpress.com/2011/07/certificado-de-atributo1.pdf>. Acesso em: 6 jun. 2013. Citado na página 34.

BURNETT, S.; PAYNE, S. *Criptografia e Segurança:*: O guia oficial rsa. 5. ed. Rio de Janeiro: Elsevier, 2002. 367 p. Citado na página 32.

CERTFORUM, 5., 2007, Brasília. *Certificado de Atributo Digital na ICP- Brasil.* Brasília: Serasa, 2007. 1-6 p. Disponível em: <http://www.certificadodigital.com.br-/certforum2007/WhitePaperCertificadoAtributoDigital.p df>. Acesso em: 6 jun. 2013. Citado na página 34.

CONSELHO FEDERAL DE MEDICINA. *Resolução CFM n.º 1.658/2002*. Conselho Federal de Medicina, 2002. Disponível em: <http://www.portalmedico.org.br/resolucoes-/CFM/2002/1658_2002.htm>. Acesso em: 6 jun. 2013. Citado 2 vezes nas páginas 53 e 60.

CUSTÓDIO, I. V. *H-PMI:*: Uma arquitetura de gerenciamento de privilégios para sistemas de informação da área da saúde. 139 f. Monografia (Pós-graduação) — Ciências da Computação, Universidade Federal de São Carlos, São Carlos, SP, 2010. Disponível em: <http://www.bdtd.ufscar.br/htdocs/tedeSimplificado//tde_busca-/arquivo.php?codArquivo=3197>. Acesso em: 6 jun. 2013. Citado 2 vezes nas páginas 29 e 37.

DEMICHIEL, L. *JSR 317:*: Java persistence 2.0. 2009. Disponível em: <http:/-/www.jcp.org/en/jsr/detail?id=317>. Acesso em: 27 set. 2013. Citado na página 42.

DEMICHIEL, L. *JSR 317:*: Java persistence api, version 2.0. 2009. Disponível em: <http://download.oracle.com/otn-pub/jcp/persistence-2.0-fr-eval-oth-JSpec-/persistence-2_0-final-spec.pdf>. Acesso em: 27 set. 2013. Citado na página 43.

DEPARTMENT OF HEALTH AND HUMAN SERVICES. *Security and electronic signature standards*. Services: Federal Register., 1998. Disponível em: <http://aspe-.hhs.gov/admnsimp/nprm/secnprm.pdf>. Acesso em: 10 set. 2013. Citado na página 29.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. *IEEE Communication Magazine*, p. 29–40, 1976. Disponível em: <http://www.cs.berkeley.edu/~christos-/classics/diffiehellman.pdf>. Acesso em: 6 jun. 2013. Citado na página 30.

FARREL, S.; HOUSLEY, R. *An Internet Attribute Certificate Profile for Authorization. RFC3281*. [S.l.], 2002. Disponível em: <http://datatracker.ietf.org/doc/rfc3281/>. Citado na página 34.

FARREL, S.; HOUSLEY, R.; TURNER, S. *An Internet Attribute Certificate Profile for Authorization. RFC5755*. [S.l.], 2010. Disponível em: <http://datatracker.ietf.org/doc-/rfc5755/>. Citado 4 vezes nas páginas 34, 36, 46 e 90.

FEDERAL FINANCIAL INSTITUTIONS EXAMINATION COUNCIL. *Authentication in an Internet Banking Environment*. Federal Financial Institutions Examination Council, 2008. Disponível em: <http://www.ffiec.gov/pdf/authentication_guidance.pdf>. Acesso em: 10 set. 2013. Citado na página 27.

NATIONAL COMPUTER SECURITY CONFERENCE, 15., 1992, Baltimore. *Role-Based Access Controls*. Gaithersburg: National Institute of Standards and Technology, 1992. 554-563 p. Citado na página 29.

GUILHEN, S. N. *Um Serviço de Autorização Java EE Baseado em Certificados de Atributos X.509*. 110 f. Dissertação (Mestrado em Ciências da Computação) — Universidade de São Paulo, São Paulo, SP, 2008. Disponível em: <http://www.ime-.usp.br/~reverbel/students/master_theses/stefan_guilhen.pdf>. Acesso em: 6 out. 2013. Citado 4 vezes nas páginas 13, 46, 47 e 90.

ICP-BRASIL. *PERFIL DE USO GERAL E REQUISITOS PARA GERAÇÃO E VERIFICAÇÃO DE CERTIFICADOS DE ATRIBUTO NA ICP-BRASIL*. [S.l.], 2012. Disponível em: <http://bit.ly/1aR1Y7R>. Citado 2 vezes nas páginas 37 e 117.

ICP-BRASIL. *VISÃO GERAL SOBRE CERTIFICADO DE ATRIBUTO PARA A ICP-BRASIL DOC-ICP-16*. [S.l.], 2012. Disponível em: <http://bit.ly/18yRKMC>. Citado na página 37.

IETF. *Request For Comments (RFC)*. 2004. Disponível em: <http://www.ietf.org/rfc-.html>. Acesso em: 27 set. 2013. Citado na página 42.

INFORMATION SYSTEMS SECURITY RESEARCH GROUP (ISSRG). *Installing and Testing PERMIS*. [S.l.], 2008. Disponível em: <http://sec.cs.kent.ac.uk/permis-/documents/InstallingTestingPERMISv1.1.doc>. Citado na página 82.

JAVA COMMUNITY. *Welcome to the Java Community Process*. 2003. Disponível em: <http://jcp.org/en/home/index>. Acesso em: 27 set. 2013. Citado na página 42.

JAVA COMMUNITY. *JSR 215:*: Java community process version 2.6. 2009. Disponível em: <http://www.jcp.org/en/jsr/detail?id=215>. Acesso em: 27 set. 2013. Citado na página 42.

JAVA COMMUNITY. *JSR 316:*: Jsr-000316 java platform, enterprise edition 6 specification 6.0 final release. 2009. Disponível em: <http://download.oracle.com-/otn-pub/jcp/javaee-6.0-fr-oth-JSpec/javaee-6_0-fr-spec.zip>. Acesso em: 27 set. 2013. Citado 3 vezes nas páginas 13, 43 e 44.

JAVA COMMUNITY. *JSR 316:*: Java platform, enterprise edition 6 (java ee 6) specification. 2009. Disponível em: <http://www.jcp.org/en/jsr/detail?id=316>. Acesso em: 27 set. 2013. Citado na página 42.

JAVA COMMUNITY. *JSR 344::* Javaserver faces 2.2. 2013. Disponível em: <http://www.jcp.org/en/jsr/detail?id=344>. Acesso em: 27 set. 2013. Citado na página 42.

JAVA COMMUNITY. *JSR 344::* Javaservertm faces specification. 2013. Disponível em: <http://download.oracle.com/otn-pub/jcp/jsf-2_2-fr-eval-spec/javax.faces-api-2.2-FINAL.zip>. Acesso em: 27 set. 2013. Citado na página 44.

KALALI, M. *Top 6 Open Source Java EE Application Servers*. DZone, 2009. Disponível em: <http://soa.dzone.com/articles/top-open-source-javaEE-application-servers>. Acesso em: 6 jun. 2013. Citado na página 45.

KHAN, K. et al. *JBoss Application Server 7 Documentation*. JBoss, 2012. Disponível em: <https://docs.jboss.org/author/display/AS71/Documentation>. Acesso em: 6 jun. 2013. Citado na página 46.

KING, G. *JSR 299::* Contexts and dependency injection for the java ee platform. 2009. Disponível em: <http://www.jcp.org/en/jsr/detail?id=299>. Acesso em: 27 set. 2013. Citado na página 42.

KING, G. *JSR 299::* Contexts and dependency injection for the java ee platform. 2009. Disponível em: <http://download.oracle.com/otn-pub/jcp/web_beans-1.0-fr-eval-oth-JSpec/web_beans-1_0-fr-eval-spec.pdf>. Acesso em: 27 set. 2013. Citado na página 45.

KOHNFELDER, L. M. *Towards a practical public-key cryptosystem.* 54 f. Monografia (Pós-graduação) — Bachelor of Science, MIT, Boston, 1978. Disponível em: <http://groups.csail.mit.edu/cis/theses/kohnfelder-bs.pdf>. Acesso em: 6 jun. 2013. Citado na página 32.

FOURTH ACM WORKSHOP ON ROLE-BASED ACCESS CONTROL, 1999, Fairfax. *Attribute certification::* an enabling technology for delegation and role-based controls in distributed environments. Virginia, United States: ACM, 1999. 121-130 p. Citado na página 38.

MACÊDO, D. *Mecanismos de Certificação e a Criptografia*. 2012. Disponível em: <http://www.diegomacedo.com.br/mecanismos-de-certificacao-e-a-criptografia/>. Acesso em: 27 set. 2013. Citado 4 vezes nas páginas 13, 33, 35 e 38.

ORACLE. *Java Language Specification*. 2011. Disponível em: <http://docs.oracle.com/javase/specs/jls/se7/html/jls-1.html>. Acesso em: 27 set. 2013. Citado na página 41.

ROUSE, M.; ROURKE, T. *Application Server*. TechTarget, 2005. Disponível em: <http://searchsqlserver.techtarget.com/definition/application-server>. Acesso em: 6 jun. 2013. Citado na página 45.

SAKS, K. *JSR 318::* Enterprise javabeans 3.1. 2009. Disponível em: <http://www.jcp.org/en/jsr/detail?id=318>. Acesso em: 27 set. 2013. Citado na página 42.

SAKS, K. *JSR 318::* Jsr-000318 enterprise javabeans 3.1 specification. 2009. Disponível em: <http://download.oracle.com/otn-pub/jcp/ejb-3.1-fr-oth-JSpec/ejb-3_1-fr-spec-.pdf>. Acesso em: 27 set. 2013. Citado na página 43.

SANDHU, R.; FERRAIOLO, D.; KUHN, R. The nist model for role-based access control: Towards a unified standard. *ACM*, New York, 2000. Disponível em: <http://-/csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf>. Acesso em: 6 jun. 2013. Citado na página 29.

SANDHU, R.; SAMARATI, P. Access control:: Principle and practice. *IEEE Communication Magazine*, v. 32, p. 40–48, 1994. Disponível em: <http://www.ciss100-.com/wp-content/uploads/mobile-computing/Access%20Control.pdf>. Acesso em: 6 jun. 2013. Citado na página 28.

SANDHU, R.; SAMARATI, P. Authentication, access control and audit. *ACM Computing Surveys*, New York, NY, p. 241–243, Março 1996. Disponível em: <http:-//www.profsandhu.com/journals/acm/survey96(org).pdf>. Acesso em: 6 jun. 2013. Citado na página 27.

SCHNEIER, B. *Applied Cryptography:*: Protocols, algorithms and source code in c. 2. ed. New York, NY: John Wiley & Sons, 1996. Citado na página 29.

SEBESTA, R. W. *Concepts of Programming Languages*. 9. ed. [S.l.]: Pearson, 2010. 765 p. Citado na página 41.

SHIREY, R. *Internet Security Glossary, Version 2. RFC4949*. [S.l.], 2007. Disponível em: <http://datatracker.ietf.org/doc/rfc4949/>. Citado na página 27.

TIOBE. *TIOBE Programming Community Index*. 2013. Disponível em: <http://www-.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 27 set. 2013. Citado 2 vezes nas páginas 41 e 42.

TODOROV, D. *Authentication, Authorization and Accounting*. Auerbach Publications, 2008. Disponível em: <http://www.infosectoday.com/Articles/Authentication.htm>. Acesso em: 27 set. 2013. Citado na página 28.

Apêndices

APÊNDICE A – Política de Controle de Acesso Proposta

O uso do *PERMIS Policy Editor* resultou na criação da seguinte política de controle de acesso:

```
<?xml version="1.0" encoding="UTF-8"?>
<X.509_PMI_RBAC_Policy OID="ateste">
    <SubjectPolicy>
        <SubjectDomainSpec ID="ateste-domain">
            <Include LDAPDN="dc=my-domain,dc=com"/>
        </SubjectDomainSpec>
    </SubjectPolicy>
    <RoleHierarchyPolicy>
        <RoleSpec OID="1.2.826.0.1.3344810.1.1.14" Type="permisRole">
            <SupRole Value="admin"/>
            <SupRole Value="md"/>
        </RoleSpec>
    </RoleHierarchyPolicy>
    <SOAPolicy>
        <SOASpec ID="ramonfacchin" LDAPDN="cn=ramonfacchin"/>
    </SOAPolicy>
    <RoleAssignmentPolicy>
        <RoleAssignment ID="role-assignment">
            <SubjectDomain ID="ateste-domain"/>
            <RoleList>
                <Role Type="permisRole"/>
            </RoleList>
            <Delegate/>
            <SOA ID="ramonfacchin"/>
            <Validity/>
        </RoleAssignment>
    </RoleAssignmentPolicy>
    <TargetPolicy>
        <TargetDomainSpec ID="admin-configurations">
            <Include URL="https://ramonfacchin.ufsc.br:8443/
iprescribe-web/restricted/
administrator/configurations.jsf"/>
```

```
        </TargetDomainSpec>
        <TargetDomainSpec ID="admin-license-request">
            <Include URL="https://ramonfacchin.ufsc.br:8443/
iprescribe-web/restricted/
administrator/license-request.jsf"/>
        </TargetDomainSpec>
        <TargetDomainSpec ID="md-add-license">
            <Include URL="https://ramonfacchin.ufsc.br:8443/
iprescribe-web/restricted/
medicaldoctor/add-license.jsf"/>
        </TargetDomainSpec>
        <TargetDomainSpec ID="md-mycertificates">
            <Include URL="https://ramonfacchin.ufsc.br:8443/
iprescribe-web/restricted/
medicaldoctor/mycertificates.jsf"/>
        </TargetDomainSpec>
        <TargetDomainSpec ID="md-view-certificate">
            <Include URL="https://ramonfacchin.ufsc.br:8443/
iprescribe-web/restricted/
medicaldoctor/view-certificate.jsf"/>
        </TargetDomainSpec>
        <TargetDomainSpec ID="md-certify">
            <Include URL="https://ramonfacchin.ufsc.br:8443/
iprescribe-web/restricted/
medicaldoctor/certify.jsf"/>
        </TargetDomainSpec>
    </TargetPolicy>
    <ActionPolicy>
        <Action ID="open" Name="open"/>
    </ActionPolicy>
    <TargetAccessPolicy>
        <TargetAccess ID="admin-privileges">
            <RoleList>
                <Role Type="permisRole" Value="admin"/>
            </RoleList>
            <TargetList>
                <Target>
                    <TargetDomain ID="admin-configurations"/>
                    <AllowedAction ID="open"/>
                </Target>
                <Target>
                    <TargetDomain ID="admin-license-request"/>
```

```
                    <AllowedAction ID="open"/>
                </Target>
            </TargetList>
        </TargetAccess>
        <TargetAccess ID="md-privileges">
            <RoleList>
                <Role Type="permisRole" Value="md"/>
            </RoleList>
            <TargetList>
                <Target>
                    <TargetDomain ID="md-add-license"/>
                    <AllowedAction ID="open"/>
                </Target>
                <Target>
                    <TargetDomain ID="md-mycertificates"/>
                    <AllowedAction ID="open"/>
                </Target>
                <Target>
                    <TargetDomain ID="md-view-certificate"/>
                    <AllowedAction ID="open"/>
                </Target>
                <Target>
                    <TargetDomain ID="md-certify"/>
                    <AllowedAction ID="open"/>
                </Target>
            </TargetList>
        </TargetAccess>
    </TargetAccessPolicy>
</X.509_PMI_RBAC_Policy>
```

APÊNDICE B – Diagramas Complementares

Foram elaborados, no decorrer do trabalho, alguns diagramas que auxiliam na compreensão do sistema. Cabe explicitar que os diagramas não foram elaborados com o objetivo de auxiliar no processo de desenvolvimento, mas sim de elucidar o funcionamento de algumas partes da aplicação desenvolvida que exigem um processamento mais complexo.

B.1 Classes de Entidade (Domínio) do Sistema

O diagrama da Figura 25 apresenta as classes que compõem o domínio do negócio tratado pela aplicação proposta. Isto é, são apresentadas as classes que resultarão em entidades persistentes do modelo Objeto-Relacional desenvolvido. Há também duas enumerações que não são persistentes, mas tem seus valores utilizados por classes de entidade.

B.2 Refinamento UC04 - Emissão de Atestado

A Figura 26 apresenta como o sistema processa o salvamento do atestado médico.

B.3 Refinamento UC10 - Verificação de Autenticidade

A Figura 27 apresenta como se dá o processamento, por parte do sistema, da verificação de autenticidade de um atestado médico, após informado seu código de verificação de autenticidade.

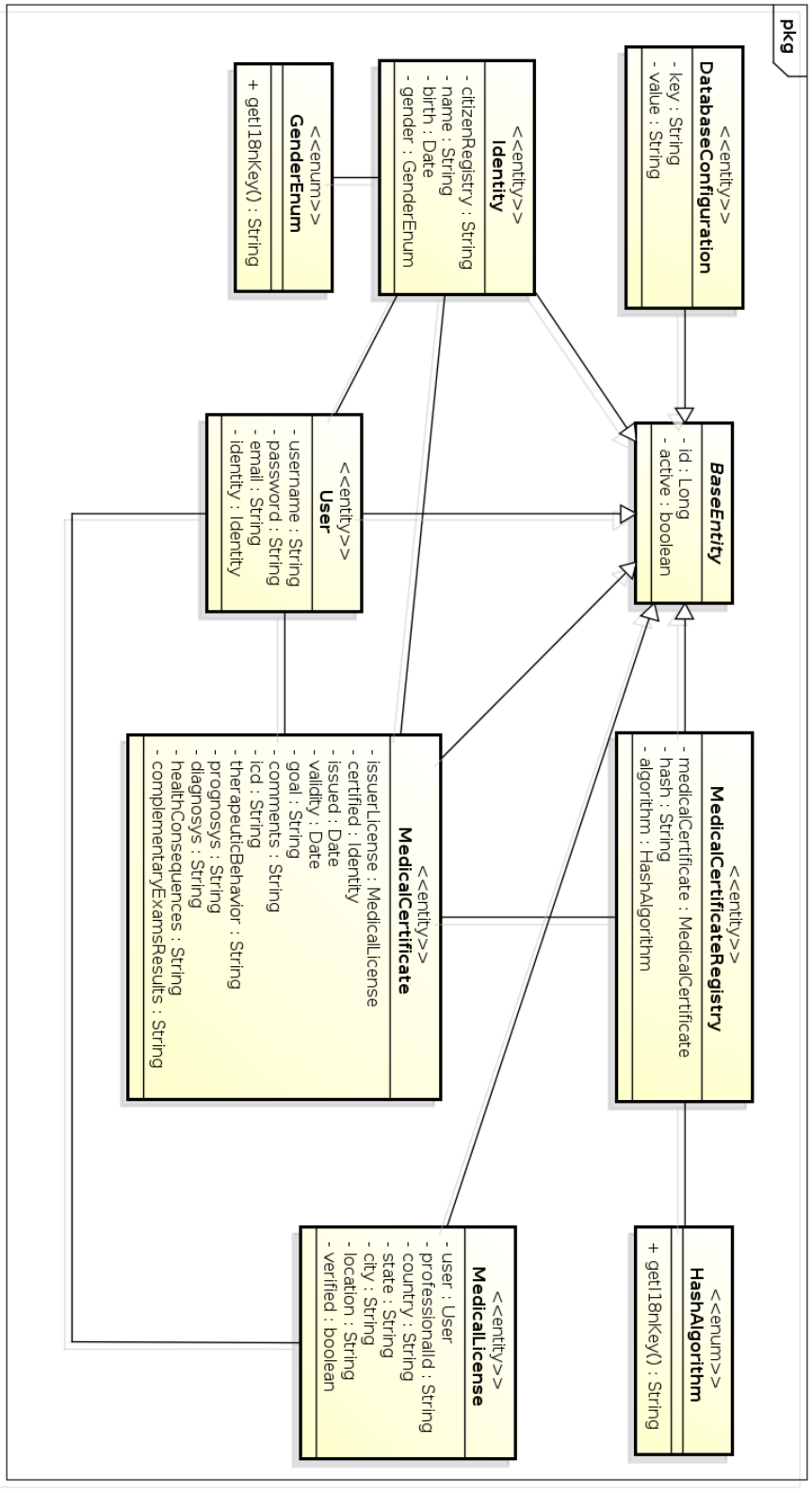

Figura 25 – Classes de Entidades do Sistema.

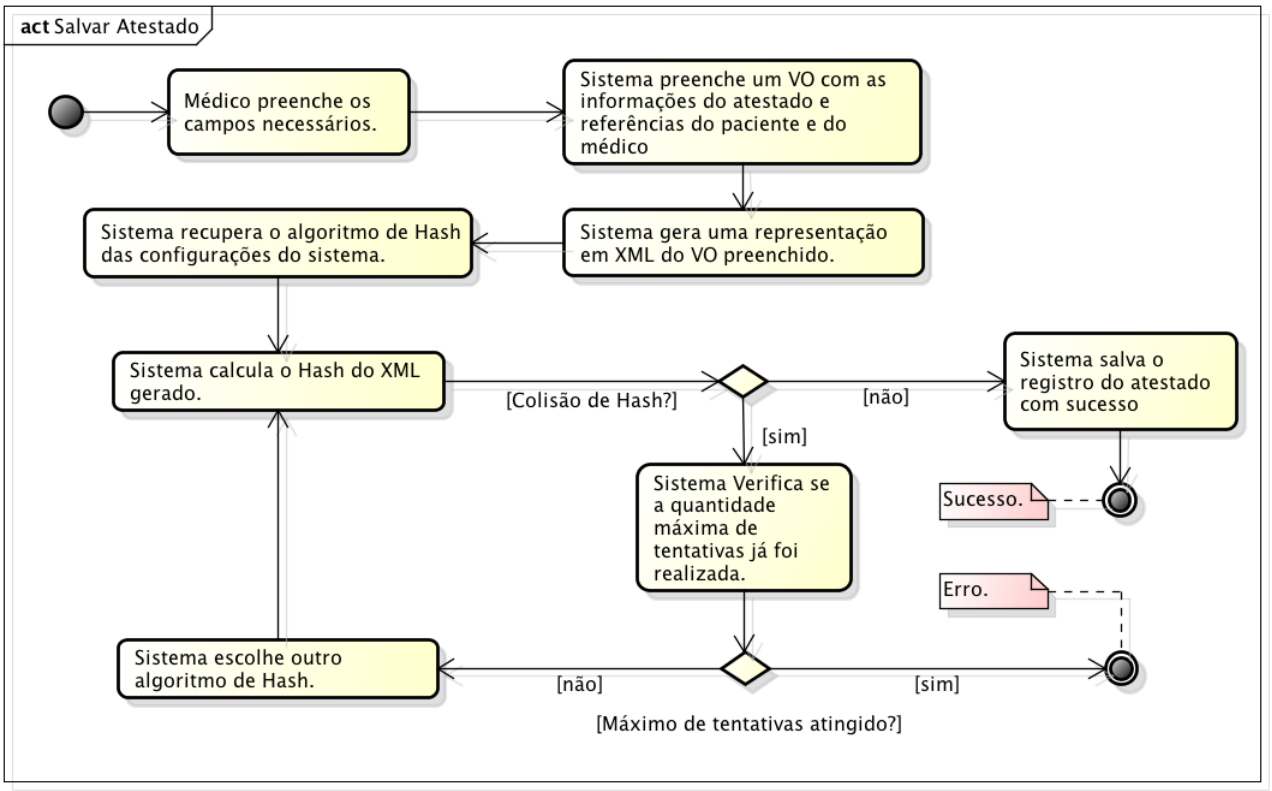

Figura 26 – Procedimento para salvar um atestado.

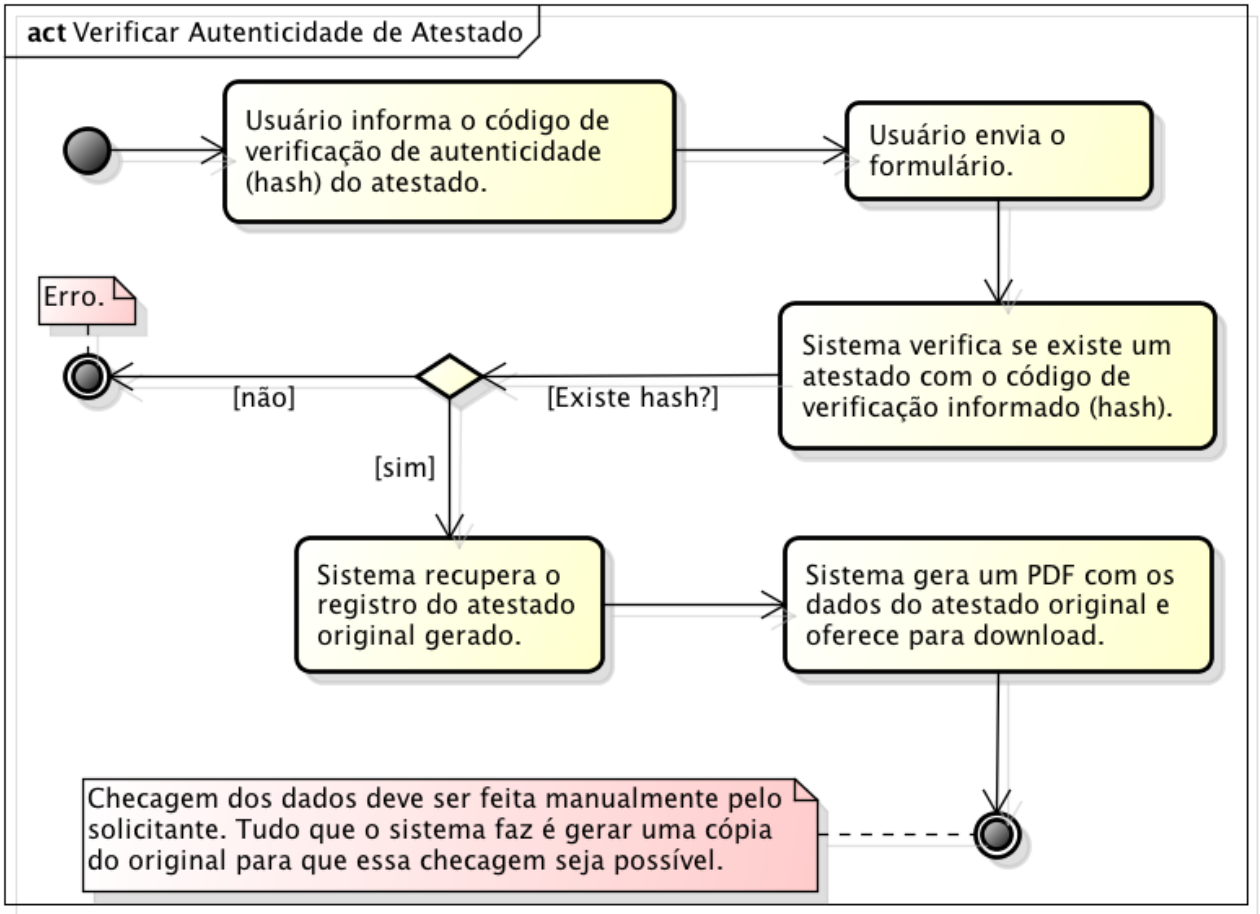

Figura 27 – Procedimento de verificação de autenticidade.

APÊNDICE C – Configuração da Aplicação Web e do Apache Shiro

C.1 Arquivo shiro.ini configurado apenas para autenticação

```
[main]
#sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
iprescribeRealm = br.ufsc.ramonfacchin.security.realm.IPrescribeRealm

#securityManager.sessionManager = $sessionManager
#securityManager.sessionManager.globalSessionTimeout = 3600000
securityManager.realms=$iprescribeRealm

authc = org.apache.shiro.web.filter.authc.PassThruAuthenticationFilter
authc.loginUrl = /login.jsf?op=restricted
ssl = org.apache.shiro.web.filter.authz.SslFilter
ssl.port = 8443

logout.redirectUrl = /login.jsf?op=logout

[urls]
/logout = logout
/public/** = anon
/login.jsf** = ssl
/restricted/** = ssl, authc
/** = anon
```

C.2 Descritor de aplicação Web (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xml="http://www.w3.org/XML/1998/namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd ">
        <listener>
```

```
                <listener-class>
                org.apache.shiro.web.env.EnvironmentLoaderListener
                </listener-class>
        </listener>

        <filter>
                <filter-name>ShiroFilter</filter-name>
                <filter-class>
                org.apache.shiro.web.servlet.ShiroFilter
                </filter-class>
        </filter>

        <filter-mapping>
                <filter-name>ShiroFilter</filter-name>
                <url-pattern>/*</url-pattern>
                <dispatcher>REQUEST</dispatcher>
                <dispatcher>FORWARD</dispatcher>
                <dispatcher>INCLUDE</dispatcher>
                <dispatcher>ERROR</dispatcher>
        </filter-mapping>
</web-app>
```

C.3 Arquivo shiro.ini com diretivas de autorização

```
[main]
iprescribeRealm = br.ufsc.ramonfacchin.security.realm.IPrescribeRealm
securityManager.realms=$iprescribeRealm

authc = org.apache.shiro.web.filter.authc.PassThruAuthenticationFilter
authc.loginUrl = /login.jsf?op=restricted
ssl = org.apache.shiro.web.filter.authz.SslFilter
ssl.port = 8443

logout.redirectUrl = /login.jsf?op=logout

[urls]
/logout = logout
/public/** = anon
/login.jsf** = ssl
/restricted/administrator/** = ssl, authc, roles[admin]
/restricted/medicaldoctor/** = ssl, authc, roles[md]
```

```
/restricted/** = ssl, authc
/** = anon
```

APÊNDICE D – Código Fonte da Aplicação

Para ter acesso ao código fonte da aplicação desenvolvida, basta acessar o endereço https://github.com/ramonfacchin/iprescribe. O repositório aceita acesso público e anônimo para o download dos fontes.

Caso seja de interesse contribuir para o código do projeto, deve ser cadastrada uma conta no GitHub e solicitar permissão para contribuir com o projeto diretamente neste repositório.

Informações sobre como proceder com o *checkout* do projeto para contribuição podem ser encontradas no endereço http://codexico.com.br/blog/linux/tutorial-simples-como-usar-o-git-e-o-github/. O portal do GitHub também disponibiliza diversos wizards e, inclusive, software para facilitar nesses procedimentos.

Anexos

ANEXO A – Padrão de Certificado de Atributo

Para que um certificado de atributo seja válido, ele deve estar em conformidade com o padrão ISO/IEC 9594-8 ou ITU X.509, que define os seguintes campos:

```
AttributeCertificate ::= SEQUENCE {
        acinfo AttributeCertificateInfo,
        signatureAlgorithm AlgorithmIdentifier,
        signatureValue BIT STRING
}

AttributeCertificateInfo ::= SEQUENCE {
        version AttCertVersion,
        -- version is v2
        holder Holder,
        issuer AttCertIssuer,
        signature AlgorithmIdentifier,
        serialNumber CertificateSerialNumber,
        attrCertValidityPeriod AttCertValidityPeriod,
        attributes SEQUENCE OF Attribute,
        extensions Extensions OPTIONAL
}

AttCertVersion ::= INTEGER {
        v2(1)
}

Holder ::= SEQUENCE {
        baseCertificateID [0] IssuerSerial OPTIONAL,
        -- the issuer and serial number of
        -- the holder's Public Key Certificate
        entityName [1] GeneralNames OPTIONAL,
        -- the name of the claimant or role
        objectDigestInfo [2] ObjectDigestInfo OPTIONAL
        -- used to directly authenticate the holder,
        -- for example, an executable
}
```

```
ObjectDigestInfo ::= SEQUENCE {
        digestedObjectType ENUMERATED {
                publicKey (0),
                publicKeyCert (1),
                otherObjectTypes (2)
        },
        -- otherObjectTypes MUST NOT
        -- be used in this profile
        otherObjectTypeID OBJECT IDENTIFIER OPTIONAL,
        digestAlgorithm AlgorithmIdentifier,
        objectDigest BIT STRING
}

AttCertIssuer ::= CHOICE {
        v2Form [0] V2Form
        -- v2 only
}

V2Form ::= SEQUENCE {
        issuerName GeneralNames OPTIONAL,
        baseCertificateID [0] IssuerSerial OPTIONAL,
        objectDigestInfo [1] ObjectDigestInfo OPTIONAL
        -- issuerName MUST be present in this profile
        -- baseCertificateID and objectDigestInfo MUST NOT
        -- be present in this profile
}

IssuerSerial ::= SEQUENCE {
        issuer GeneralNames,
        serial CertificateSerialNumber,
        issuerUID UniqueIdentifier OPTIONAL
}

AttCertValidityPeriod ::= SEQUENCE {
        notBeforeTime GeneralizedTime,
        notAfterTime GeneralizedTime
}

Attribute ::= SEQUENCE {
        type AttributeType,
        values SET OF AttributeValue
        -- at least one value is required
```

```
}

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```

Retirado de ICP-Brasil (2012a).