Uma Solução Unificada para Controle de Acesso Multiplataforma de Aplicações

Cleverson Sacramento de Oliveira
Serge Normando Rehem

Tema: Engenharia de Software (processo de desenvolvimento de sistemas)

Total de páginas: 22

2010

Título RASEA: uma solução unificada para controle de acesso multiplataforma de aplicações

Autores Cleverson Sacramento de Oliveira
Serge Normando Rehem

Tema Engenharia de Software
(processo de desenvolvimento de sistemas)

Resumo A falta de padronização do controle de acesso às aplicações corporativas faz parte da realidade de muitas empresas. É comum observar neste cenário que diversas aplicações implementam soluções próprias. Os efeitos negativos são percebidos pelos usuários dos sistemas, que são obrigados a manter diversas senhas, e pela própria empresa, que precisa garantir o controle por meio de processos e auditorias, onerando os custos operacionais. Este trabalho tem como objetivo prover uma solução orientada a serviços que suporte o controle de acesso unificado de usuários às aplicações, fundamentando-se em conceitos e tecnologias existentes para garantir a interoperabilidade em ambientes heterogêneos. O projeto RASEA foi concebido como uma solução de Software Livre que implementa conceitos como RBAC (*Role-based Access Control*) e SOA (*Service-oriented Architecture*). Disponível publicamente no repositório SourceForge, o projeto é regido pela licença GPLv3. Uma das principais contribuições, e característica marcante do RASEA, é a simplicidade do uso, alcançada através de experiências adquiridas em mais de 30 (trinta) meses de pesquisas e implementações. O elemento 'agente' da arquitetura proposta representa um grande diferencial em relação às soluções de mercado, possibilitando a extensibilidade e integração com diversas tecnologias e plataformas ao invés de substituí-las. O RASEA, que está sendo utilizado em ambiente de produção por algumas empresas, pode ser implantado também no SERPRO, promovendo uma economia nos custos de desenvolvimento de aplicações e a simplificação na administração do controle de acesso dos sistemas corporativos.

Palavras-chave Controle de Acesso. Autorização. RBAC. *Web Services*.

Total de páginas 22

CURRÍCULOS

Cleverson Sacramento de Oliveira é Mestre em Sistemas e Computação, MBA Executivo em Sistemas de Informação e Bacharel em Informática. Fez parte do quadro docente da UNIJORGE. Atua no desenvolvimento de software há mais de 10 anos, participando de projetos críticos em diversas plataformas e tecnologias. Atuou como Gerente de Inovação, Consultor Técnico e Arquiteto de Software. No SERPRO, faz parte da equipe da Coordenação Estratégica de Tecnologia (CETEC) na regional Salvador, dedicado ao projeto *framework* Demoiselle.

Serge Normando Rehem é PMP, Especialista em Sistemas Distribuídos pela UFBA e MBA em Administração pela UNIFACS. Analista do SERPRO há 12 anos, atualmente lidera a equipe técnica do *framework* Demoiselle, na projeção da Coordenação Estratégica de Tecnologia (CETEC) na regional Salvador. É líder do grupo de usuários JavaBahia, colunista da revista Java Magazine e autor do *blog* http://bazedral.blogspot.com sobre trabalho colaborativo.

SUMÁRIO

1. Introdução

A falta de padronização do controle de acesso às aplicações corporativas faz parte da realidade de muitas empresas. É comum observar neste cenário que diversas aplicações implementam soluções próprias. Os efeitos negativos são percebidos pelos usuários dos sistemas e pela própria empresa. Este capítulo delimita o escopo do objeto de estudo, apresenta a motivação e o objetivo do trabalho.

1.1. Motivação

Por manipular informações corporativas, o acesso às aplicações geralmente é restrito a usuários autorizados. O controle de acesso baseia-se em regras incorporadas ao sistema durante o processo de desenvolvimento do *software*, utilizando padrões e boas práticas (BUEGE; LAYMAN; TAYLOR, 2003). De acordo com Kern (2004, p. 87), existem poucas publicações a respeito de segurança de aplicações, se comparadas à vasta literatura sobre segurança de redes.

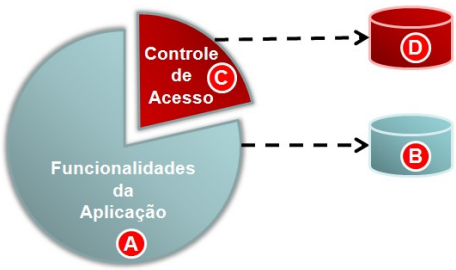

Figura 1.1 – Divisão lógica da aplicação

Sob a ótica deste trabalho, uma aplicação pode ser logicamente subdividida em dois blocos, vide Figura 1.1. Na primeira parte, A e B representam as funcionalidades da aplicação (telas, regras de negócio, relatórios e integrações) e sua a base de dados. Na segunda parte, C e D simbolizam o módulo de segurança, responsável pelas operações de controle de acesso de acordo com a NBR ISO/IEC 27001 (ABNT, 2006, p. 80).

Os autores Gaedke, Meinecke e Nussbaumer (2005, p. 1156) afirmam que diversas empresas costumam projetar diferentes módulos de controle de acesso para cada sistema implementado, gerando gastos desnecessários decorrentes do retrabalho e da dificuldade em gerenciar as políticas de segurança. Este cenário está ilustrado no lado esquerdo da Figura 1.2, apresentando os seguintes pontos fracos do ponto de vista do controle de acesso: redundâncias desnecessárias do módulo de controle (dificultando o gerenciamento) e possí-

vel falta de padronização da estrutura dos dados.

Uma possível solução para este problema é a unificação do controle de acesso, conforme apresentado ao lado direito da Figura 1.2. Cada aplicação possui um ponto de acesso único (ALUR; CRUPI; MALKS, 2003, p. 164) que, com base nas informações disponibilizadas pelo serviço unificado, concede ou revoga o acesso ao sistema. A utilização de *Web Services* garante a interoperabilidade entre o serviço unificado e as aplicações de forma independente de plataforma.

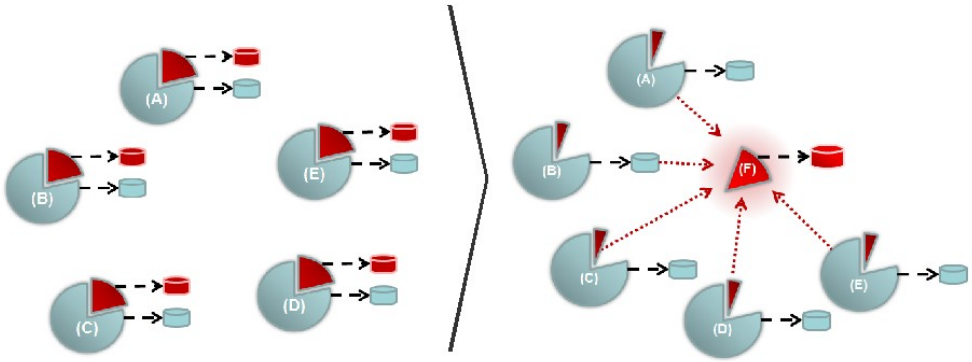

Figura 1.2 – Descentralização / centralização do controle de acesso

1.2. Objetivo

Este trabalho tem como principal objetivo prover uma solução orientada a serviços que suporte o controle de acesso unificado de usuários às aplicações, fundamentando-se em conceitos e tecnologias existentes para garantir a interoperabilidade em ambientes heterogêneos. Adicionalmente, pretende-se demonstrar que os conceitos e ferramentas aqui apresentados podem ser aplicados tanto internamente ao SERPRO quanto em clientes e demais órgãos do governo, proporcionando o reuso, a diminuição do retrabalho, a redução de prazos e custos e o aumento da segurança e confiabilidade das soluções construídas

2. Fundamentos

O problema gerado pela descentralização no controle de acesso às aplicações comerciais motivou a proposta de unificação para garantir a independência de plataforma. Este capítulo trata dos padrões, modelos, conceitos e paradigmas relacionados com a solução proposta, tais como: autenticação, autorização, controle de acesso e RBAC.

2.1. Autenticação, Autorização e Controle de Acesso

É comum que pessoas não habituadas com o assunto confundam os termos 'autenticação' e 'autorização'. Resumidamente, a autenticação enfoca na identificação enquanto a autori-

zação define o que é possível fazer. De acordo com Silva (2004, p. 66), “autenticação é a capacidade de garantir que alguém, ou alguma coisa, é de fato quem diz ser, dentro de um contexto definido”. O enfoque deste trabalho está no processo de autorização e controle de acesso e, por este motivo, questões relativas à autenticação não serão detalhadas.

Garantir a confidencialidade da informação é uma necessidade latente nas aplicações corporativas. Assegurar a autorização significa que ninguém poderá executar ações sobre um recurso caso não tenha permissão (JAAS, 2001). Segundo Barker (2008, p. 144) uma permissão é composta por uma ação e um recurso. Quando uma permissão é concedida a um usuário, tem-se uma autorização. Alguns autores, tais como Sandhu, Ferraiolo e Kuhn (2000), utilizam o termo 'operações' para referenciar as 'ações' e o termo 'objeto' para referenciar os 'recursos'.

De acordo com a norma internacional ISO/IEC 10181-3 (ISO, 1996, p. 1, tradução nossa), “o processo de determinar quais utilizações dos recursos [...] são permitidas, [...] prevenindo o acesso não autorizado, é chamado de controle de acesso”. Para garantir o tratamento e o controle efetivo das requisições de acesso ao sistema, este trabalho considera que os usuários estejam devidamente autenticados ao executarem operações sobre os recursos das aplicações (BARKER, 2008).

A solução proposta segue a premissa de que tudo é proibido, a menos que explicitamente permitido, conforme a norma ISO/IEC 17799 (ABNT, 2005, p. 66). Desta forma, o usuário somente terá acesso ao recurso caso esteja explicitamente autorizado. As autorizações podem ser concedidas ou negadas. Esta característica é conhecida como autorizações positivas ou negativas (BERTINO, 2001, p. 42).

2.2. RBAC

A sigla RBAC, do inglês *Role-Based Access Control*, significa controle de acesso baseado em papéis. Um papel representa um grupo de usuários de uma organização, que reúne pessoas do mesmo cargo, função, responsabilidade, atividade ou outra categorização. A abordagem orientada aos papéis é o diferencial do RBAC em relação aos seus antecessores *Mandatory Access Control* (MAC) e *Discretionary Access Control* (DAC).

Devido à diversidade de implementações do RBAC, o *National Institute of Standards and Technology* (NIST) consolidou as estratégias em um único modelo (SANDHU; FERRAIOLO; KUHN, 2000, p. 2), dando origem ao padrão ANSI INCITS 359-2004 (ANSI, 2004). A arquitetura do projeto RASEA tem como base o padrão ANSI INCITS 359-2004, doravante referenciado apenas como RBAC.

O padrão RBAC é composto por componentes abordados pelo Modelo de Referência e pela Especificação Funcional. O Modelo de Referência estabelece rigorosamente as entidades RBAC, seus relacionamentos e as nomenclaturas utilizadas pela Especificação Funcional. Esta última define as operações responsáveis por criar, excluir e manter os elementos do Modelo de Referência.

O Modelo de Referência e a Especificação Funcional são definidos em três segmentos, conhecidos como Componentes RBAC (FERRAIOLO, 2001, p. 228), que são: *Core* RBAC, *Hierarchical* RBAC e *Constrained* RBAC. Este trabalho se concentrará no componente *Core* RBAC por tratar com mais aprofundamento os elementos fundamentais. Serão apresentados o *Hierarchical* RBAC e o *Constrained* RBAC.

O Modelo de Referência do *Core* RBAC – ilustrado na Figura 2.1 – estabelece a relação entre USERS, ROLES e os demais elementos. A entidade SESSIONS e suas relações (*user_sessions* e *session_roles*) representam sessões de usuários autenticados. As entidades OPS, OBS e PRMS representam os elementos *operations*, *objects* e *permissions*, respectivamente.

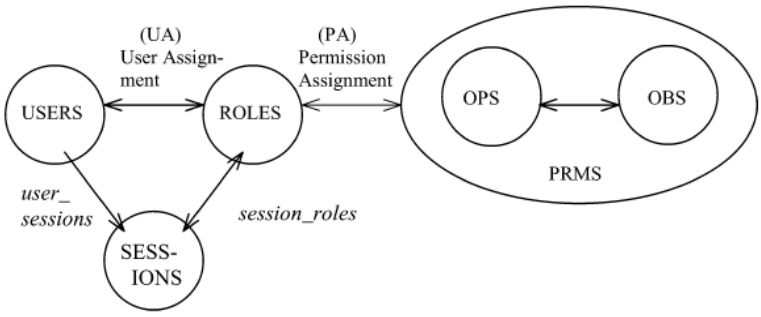

Figura 2.1 – Modelo de Referência do Core RBAC
Fonte: ANSI (2004).

A Figura 2.2 representa as operações suportadas pelo componente *Core* RBAC agrupadas em três categorias, que são: comandos administrativos, responsáveis pela criação e manutenção dos elementos RBAC e suas relações; funções de apoio ao sistema, as quais manipulam informações das sessões de usuários e executam o controle de acesso ao sistema e; funções de consulta, que recuperam informações referentes aos elementos RBAC e suas relações (ANSI, 2004, p. 11–17).

O *Hierarchical* RBAC acrescenta ao *Core* RBAC o conceito de hierarquia de papéis, onde um papel pode herdar as autorizações de outro papel (SANDHU; FERRAIOLO; KUHN, 2000, p. 3). Este componente contribui com a relação *role hierarchy* para o Modelo de Referência e com quatro operações administrativas para a Especificação Funcional, responsáveis pela

manipulação da relação *role hierarchy*, que são: *AddInheritance*; *DeleteInheritance*; *AddAscendant* e; *AddDescendant* (FERRAIOLO, 2001, p. 244; ANSI, 2004, p. 17–18).

O componente *Constrained* RBAC acrescenta ao *Hierarchical* RBAC o tratamento de conflitos de interesses. Com este componente é possível estabelecer regras para impossibilitar que um usuário seja associado ao papel solicitante e aprovador de orçamentos simultaneamente, por exemplo. O *Constrained* RBAC contribui com o Modelo de Referência com os seguintes elementos: *Static Separation of Duty Relations*, aplicado às relações *user assignment* e *role hierarchy* e; *Dynamic Separation of Duty Relations*, aplicado à relação *session_roles*.

Core RBAC

		nome	descrição
comandos e funções	administrativos	*AddUser*	Cria um novo usuário.
		DeleteUser	Exclui um usuário existente.
		AddRole	Cria um novo papel.
		DeleteRole	Exclui um papel existente.
		AssignUser	Associa um usuário a um papel.
		DeassignUser	Desassocia um usuário a um papel.
		GrantPermission	Concede permissão para um papel.
		RevokePermission	Revoga permissão para um papel.
	apoio ao sistema	*CreateSession*	Cria uma nova sessão.
		DeleteSession	Deleta uma sessão existente.
		AddActiveRole	Adiciona um papel a uma sessão existente.
		DropActiveRole	Remove um papel de uma sessão existente.
		CheckAccess	Indica se o usuário da sessão possui autorização para executar uma determinada operação em um recurso do sistema.
	consulta	*AssignedUsers*	Lista os usuáros associados a um papel.
		AssignedRoles	Lista os papéis associados a um usuário.
		RolePermissions	Lista as permissões concedidas a uma papel.
		UserPermisssions	Lista as permissões concedidas a um usuário.
		SessionRoles	Lista os papéis associados a uma sessão.
		SessionPermissions	Lista todas as permissões associadas a uma sessão.
		RoleOperationsOnObject	Lista as operações que um papel pode executar em um determinado recurso.
		UserOperationsOnObject	Lista as operações que um usuário pode executar em um determinado recurso.

Figura 2.2 – Especificação Funcional do Core RBAC

O padrão RBAC não menciona o controle de acesso a múltiplos sistemas. A arquitetura do projeto RASEA utiliza o padrão RBAC ANSI INCITS 359-2004 (ANSI, 2004), acrescentando-se a entidade *application* com o propósito de controlar o acesso às diversas aplicações corporativas simultaneamente.

3. Histórico do projeto

As ideias apresentadas neste trabalho foram validadas com aplicações práticas antes da concepção do projeto. O nome RASEA é um acrônimo do inglês *cRoss-plAtform accesS control for Enterprise Applications*, que pode ser traduzido como Controle de Acesso Multiplata-

forma para Aplicações. Regido pelas licenças GPL (2007) e LGPL (2007), o RASEA está disponível no maior repositório de Software Livre reconhecido internacionalmente (http://sourceforge.net/projects/rasea/).

3.1. Cronologia

Durante o período de mais de 30 (trinta) meses de pesquisas e implementações, participaram ativamente do processo usuários responsáveis pela administração de acesso nas organizações envolvidas. O projeto-piloto AvanSEG surgiu em fevereiro de 2007 (Figura 3.1). A execução do projeto foi bem sucedida, cumprindo o cronograma de 18 (dezoito) meses, finalizando em julho de 2008. Em junho de 2008, nasceu o S4A.

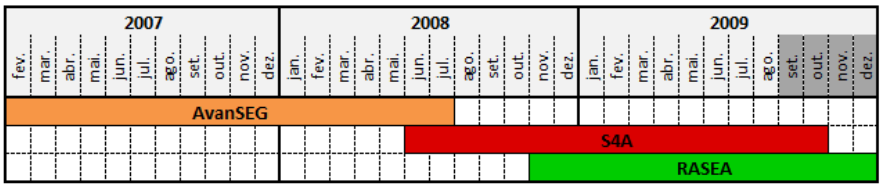

Figura 3.1 – Macro-cronograma dos projetos: AvanSEG, S4A e RASEA

Em novembro de 2008, surgiu o RASEA, que foi implantado em janeiro de 2009 em 4 (quatro) projetos em uma Fábrica de Software e em 2 (dois) projetos de uso interno de uma determinada empresa. Em maio de 2009, o RASEA foi instituído como sistema padrão para controle de acesso às aplicações da Fábrica de Software e aos sistemas internos da organização.

	Acesso à base única de usuários	Controle de acesso unificado	Independência de plataforma	Escalonamento do servidor	Código-fonte escrito na língua ing	Suporte à internacionalização	Código aberto / livre	Aderência ao padrão RBAC	Arquivo de configuração
AvanSEG	✓	✓	✓	✗	✗	✗	✗	✗	✗
S4A	✓	✓	✓	✓	✓	✗	✗	✗	✗
rasea	✓	✓	✓	✓	✓	✓	✓	✓	✓

Legenda
✓ Atende
✗ Não atende

Figura 3.2 – Comparativo entre os projetos: AvanSEG, S4A e RASEA

Com base nos relatos e sugestões dos usuários, as funcionalidades foram evoluindo a cada

projeto, conforme apresentado na Figura 3.2. Durante o processo de desenvolvimento, os usuários envolvidos elogiaram a evolução do trabalho e evidenciaram a simplicidade proporcionada nas atividades de controle de acesso e gerenciamento de permissões.

3.2. Cenário A – Sistemas Internos

Em meados de 2007, deu-se início ao projeto de unificação da base de usuários de uma organização. Até então, cada sistema possuía uma base própria com credenciais de usuários, dificultando o gerenciamento da política de segurança da empresa. Diversos estudos foram feitos para identificar as informações relevantes, descartando os dados redundantes e desnecessários. Após a criação da base unificada surgiu outro problema: o controle descentralizado de acesso às aplicações internas da empresa.

O projeto envolveu diretamente 20 (vinte) sistemas críticos, acessados por aproximadamente 450 (quatrocentos e cinquenta) usuários. Durante sua execução, concebeu-se uma solução para unificação do controle de acesso, tendo como premissa básica a integração com as aplicações internas de forma independente de tecnologia. Surgiu em meados de 2007 o projeto-piloto AvanSEG. Durante os experimentos, observou-se a necessidade de implementar algoritmos mais eficientes.

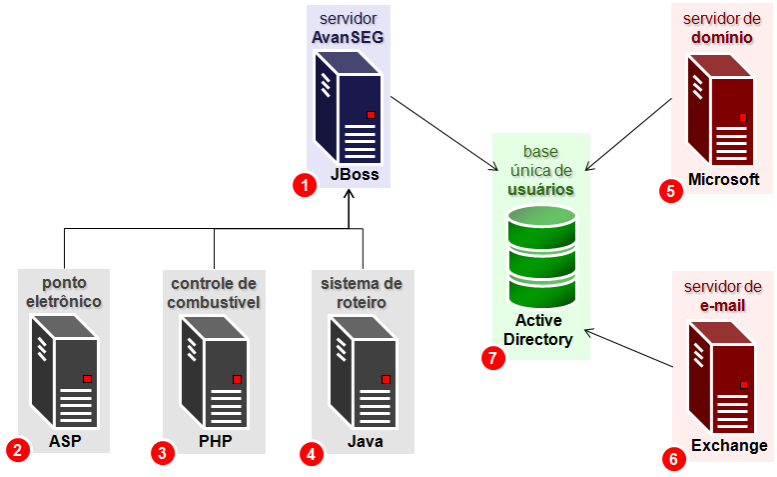

Figura 3.3 – Arquitetura dos Sistemas Internos

Na Figura 3.3, o elemento 1 representa o servidor, que provê os serviços de controle de acesso para as aplicações parceiras e integra com a base única de usuários 7. As aplicações 2, 3 e 4 acessam os serviços disponibilizados por 1. Apesar de não fazerem parte do escopo do projeto-piloto, os elementos 5 e 6 evidenciam a harmonia das ferramentas de infra-estrutura com a solução proposta.

3.3. Cenário B – Sistema de Matrícula

Após testes exaustivos e refatorações do código-fonte do AvanSEG, surgiu o *Security for Applications* (S4A) como uma solução para ambiente de produção. Em princípio, pretendia-se tornar o projeto S4A *open source*, porém, devido às demandas internas da empresa, o plano foi suspenso.

Em 2008, deu-se início ao Sistema de Matrícula da rede pública de ensino, projetado para suportar 720 (setecentos e vinte) usuários com acesso simultâneo em horários de pico pré-determinados, geograficamente distribuídos por mais de 600 (seiscentas) unidades conectadas à Internet. Os acessos à aplicação foram controlados pelo S4A. Os requisitos não-funcionais do sistema focaram em dois aspectos: performance e disponibilidade.

Durante o período de 60 (sessenta) dias, aproximadamente 1.000.000 (um milhão) de alunos foram cadastrados no sistema, gerando picos de acesso durante o horário comercial. Apesar de utilizar apenas dois servidores, o sistema foi projetado para *cluster* com alta escalabilidade, possibilitando a substituição de servidores sem interrupção do serviço.

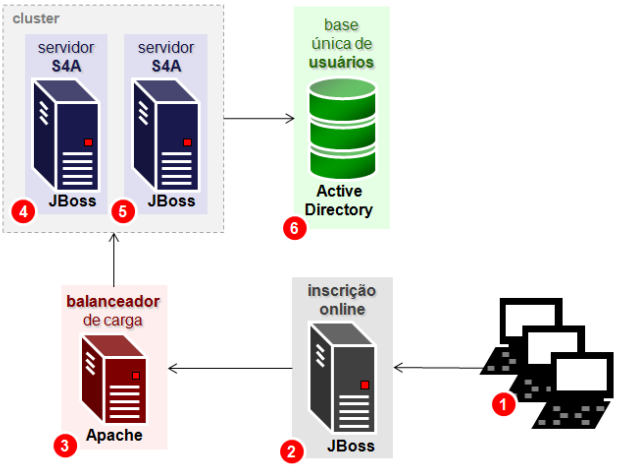

Figura 3.4 – Arquitetura do projeto Inscrição Online

Na Figura 3.4, o elemento 1 representa os usuários da aplicação, que acessam o sistema através da intranet ou Internet. O item 2 representa o servidor, que hospeda o Sistema de Matrícula. O controle de acesso é executado pelo agente S4A implantado na aplicação, que acessa o servidor 4 ou 5 através do balanceador de carga 3. A base unificada de usuários da organização é ilustrada pelo elemento 6. Nenhuma configuração especial foi feita no S4A para suportar *cluster*, e sim no servidor de aplicação, conforme a documentação oficial da ferramenta (STANSBERRY; ZAMARRENO, 2009).

3.4. Cenário C – Fábrica de Software

Em muitas empresas de desenvolvimento de sistemas, o estabelecimento dos custos e prazos dos projetos é feito com base na estimativa do tamanho funcional, utilizando a técnica de Análise de Pontos de Função (APF) (VAZQUEZ; SIMÕES; ALBERT, 2003).

Para cada nova aplicação, fazia-se necessária a construção de um novo módulo para controle de acesso que, com base na estimativa da organização, representava 190 PF (cento e noventa pontos de função). Supondo um sistema de 1.000 PF (mil pontos de função), o controle de acesso representaria 19% (dezenove por cento) do custo de desenvolvimento.

Empregando o S4A como componente reutilizável (DAN; JOHNSON; CARRATO, 2008, p. 25), a organização economizou 190 PF (cento e noventa pontos de função) a cada aplicação desenvolvida, com investimento mínimo na integração do *software*. Os benefícios foram percebidos principalmente na redução do custo e prazo do projeto. Por questões de confidencialidade, não serão apresentados o custo do PF nem a produtividade da organização.

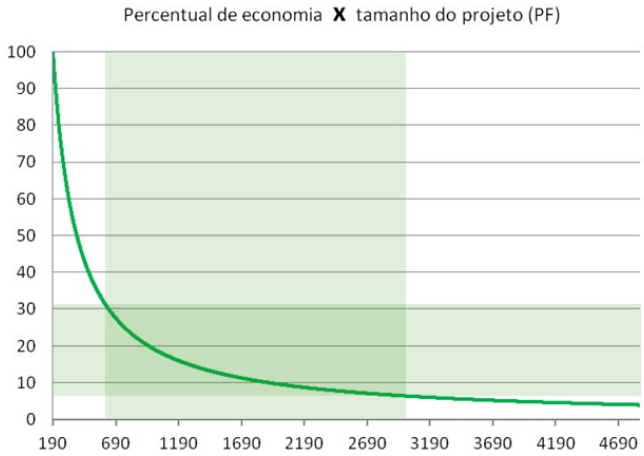

Figura 3.5 – Economia na utilização do S4A

Na Figura 3.5, o eixo horizontal representa o tamanho funcional da aplicação e o eixo vertical indica o percentual de economia, considerando que a aplicação possua funcionalidades para controle de acesso. De acordo com a base histórica da empresa (2004 à 2009), os sistemas construídos pela Fábrica de Software variam entre 600 PF (seiscentos pontos de função) e 3.000 PF (três mil pontos de função), gerando uma economia entre 6% (seis por cento) e 31% (trinta e um por cento) nos custos do projeto.

4. Projeto RASEA

Projetado como uma solução para controle de acesso unificado, o RASEA implementa os

conceitos RBAC, SOA e *Web Services* para garantir a maturidade da solução e a independência de plataforma. Este capítulo apresenta a arquitetura da solução proposta. Os detalhes de codificação não serão abordados, haja vista que o código-fonte está disponível publicamente em http://svn.rasea.org/svnroot/rasea/.

4.1. Visão Geral

A Arquitetura de Software auxilia na identificação dos componentes e seus relacionamentos de alto nível (SHAW; GARLAN, 1994), facilitando o entendimento da solução como um todo. O projeto RASEA fundamenta sua arquitetura sobre dois elementos principais tratados sob duas perspectivas, que são respectivamente: servidor, agente, independência de plataforma e integração.

Enquanto o elemento 'servidor' preocupa-se com a disponibilidade dos serviços, o 'agente' aborda aspectos específicos das plataformas das aplicações. A perspectiva 'independência de plataforma' agrega padrões e tecnologias, possibilitando que diferentes plataformas beneficiem-se da solução. A perspectiva 'integração' promove a interação entre o RASEA, as informações da organização e as aplicações clientes.

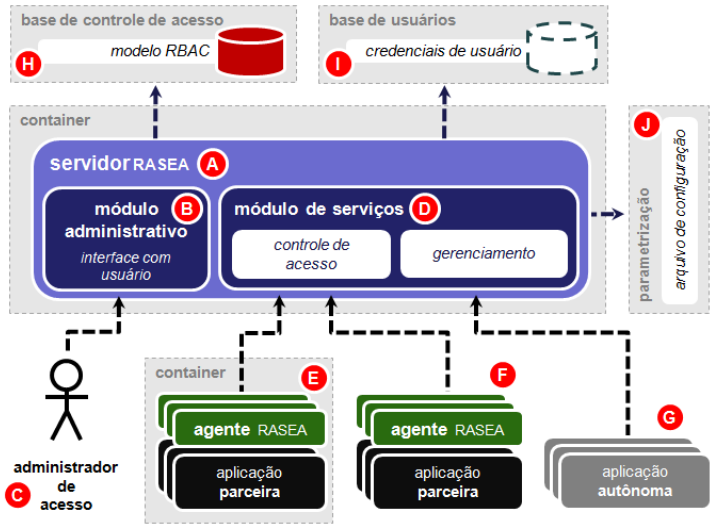

Figura 4.1 – Visão geral da arquitetura do projeto RASEA

Na Figura 4.1, o elemento A representa o servidor, o qual disponibiliza um módulo administrativo B (Anexo A – Módulo administrativo) para que o administrador do acesso C gerencie as permissões aos sistemas. O servidor provê o módulo de serviços D. Os agentes são conectados às aplicações parceiras E e F. As aplicações G, que não utilizam os serviços para controle de acesso, são chamadas de aplicações autônomas, que podem acessar os serviços de gerenciamento.

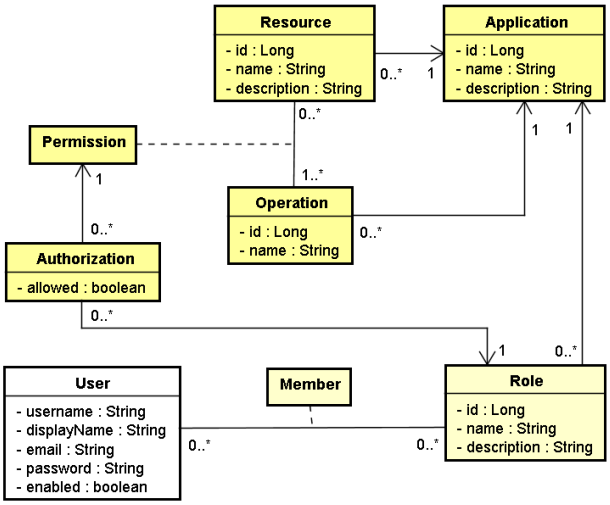

Figura 4.2 – Diagrama de Classes de Domínio do RASEA

Ainda na Figura 4.1, o servidor A mantém a base de dados H, que contém informações estruturadas conforme o Modelo de Referência do padrão RBAC. A base de dados I detém credenciais dos usuários existentes na organização. O servidor A acessa a base de dados de usuários I para executar tarefas de autenticação.

O Diagrama de Classes de Domínio (Figura 4.2) representa a visão orientada a objetos (OO) do padrão RBAC utilizando UML (RAY, 2004). Por se tratar de uma percepção OO, algumas entidades (descritas no capítulo 2.2.) sofreram adaptações na sua nomenclatura, aproximando-se mais da realidade das aplicações (GAEDKE; MEINECKE; NUSSBAUMER, 2005, p. 1156). A associação *permission_assignment* é representada pela classe *Authorization*. A entidade *object* ganhou um nome mais intuitivo: *Resource*, que pode ser traduzido como 'recurso'. A associação *user_assignment* é representada pela classe *Member*.

4.2. Independência de plataforma e Integração

Uma das premissas do RASEA é garantir a interoperabilidade com aplicações parceiras em diferentes plataformas. Para possibilitar esta característica, optou-se por duas estratégias: garantir a independência de plataforma e empregar *Web Services* no canal de comunicação. De acordo com Li e Karp (2007, p. 9), os padrões abertos, adotados pelos *Web Services*, possibilitam o desacoplamento das aplicações, facilitando o desenvolvimento de componentes independentes de tecnologias.

Esta decisão está totalmente aderente aos padrões de interoperabilidade de governo eletrônico elencados na arquitetura e-PING (2010, cap. 10.1.4): “Orienta-se o uso de *Web Services* para demandas de integração entre sistemas de informação de governo. De maneira que, independente das tecnologias em que foram implementados, possa-se adotar um pa-

drão de interoperabilidade que garanta escalabilidade, facilidade de uso, além de possibilitar atualização de forma simultânea e em tempo real".

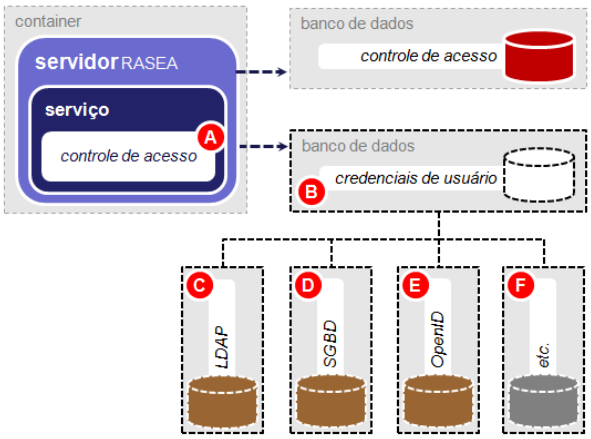

Figura 4.3 – Integração com base de usuários legada

Os serviços foram agrupados em dois *EndPoints*: controle de acesso e gerenciamento. O *EndPoint* de controle de acesso contém serviços consumidos pelos agentes e provê informações para o efetivo controle de acesso. O *EndPoint* de gerenciamento reúne os serviços de administração do servidor RASEA, acessados por aplicações (parceiras ou autônomas) que necessitam modificar informações do servidor.

Além da integração com seus serviços, o RASEA possibilita a conexão com repositórios de credenciais de usuários, tal como o LDAP. A Figura 4.3 ilustra a integração entre o servidor A e a base de usuários B. A estratégia de armazenamento das credenciais (C, D e E) é definida no arquivo de configuração do servidor. Para garantir a extensibilidade da solução, o RASEA permite que novas estratégias F sejam acopladas.

É fundamental que todos os níveis de integração (CUMMINS, 2002, p. 316) estabeleçam conexões seguras. A integração entre as aplicações e os serviços RASEA podem empregar o *HyperText Transfer Protocol Secure* (HTTPS) ou o *Secure LDAP* (LDAPS), por exemplo. Não faz parte do escopo desde trabalho detalhar protocolos e técnicas de segurança.

4.3. Servidor

O servidor é o ponto central da arquitetura RASEA, responsável por implementar o modelo RBAC, prover serviços por meio de *Web Services*, integrar com dados da organização e disponibilizar uma interface gráfica para gerenciamento. Todas as informações fornecidas pelos serviços do RASEA têm sempre como fonte duas bases de dados: a base de autorização, que implementa o modelo RBAC; e a base de autenticação, que contém as credenciais dos usuários.

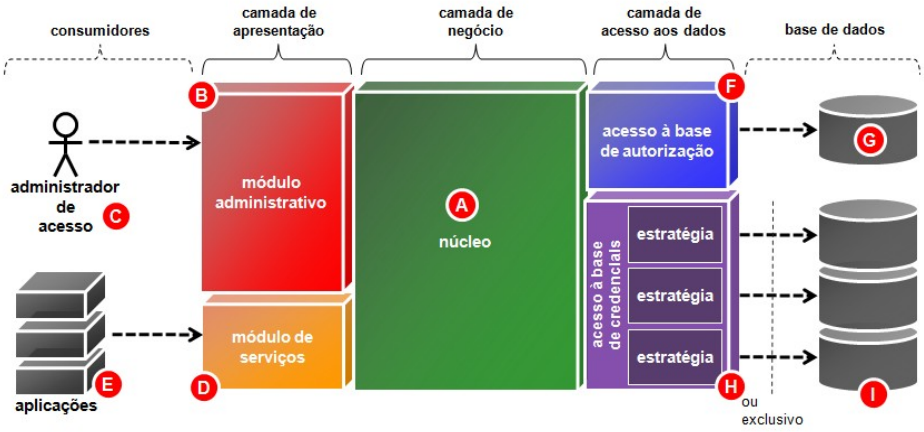

Figura 4.4 – Arquitetura em camadas do servidor RASEA

A base de autorização é mantida única e exclusivamente pelo servidor RASEA, sendo possível efetuar modificações nos dados através do módulo administrativo ou via serviços de gerenciamento. A interação com a base de autenticação implementa o padrão de projeto *Strategy* (GAMMA, 2000, p. 292), onde o servidor toma o arquivo de configuração como fonte de parâmetros.

O servidor RASEA tem a estrutura interna dividida em 3 (três) camadas: apresentação, negócio e acesso aos dados. A responsabilidade da camada de apresentação é possibilitar a interação entre os consumidores e as funcionalidades do servidor. A camada de negócio é o núcleo do servidor, composto pelas operações do RBAC. A camada de acesso aos dados mantém as informações armazenadas nas bases.

A Figura 4.4 apresenta a arquitetura em camadas do servidor RASEA, tendo como principal elemento o seu núcleo A, envolto pelas camadas de apresentação e acesso aos dados. Os elementos B e D fazem parte da camada de apresentação, que expõem as funcionalidades do servidor para os consumidores C e E. A camada de acesso manipula a base de autorização G (por meio de F) e a base de credenciais I (por meio de H).

4.4. Agente

Todas aplicações parceiras devem estar conectadas a um agente. O agente RASEA atua como um conversor do modelo independente de plataforma para a tecnologia específica de cada aplicação (IBM, 2009). Além disso, o agente executa efetivamente as tarefas de controle de acesso às aplicações.

Os agentes simplificam o desenvolvimento de aplicações parceiras, que passam a dispor de uma API de controle de acesso local e não precisam se preocupar com detalhes de acesso ao serviço *Web*. Além disto, os agentes ampliam a atuação do RASEA dentro das aplicações parceiras, oferecendo suporte completo à interceptação das tentativas de acesso.

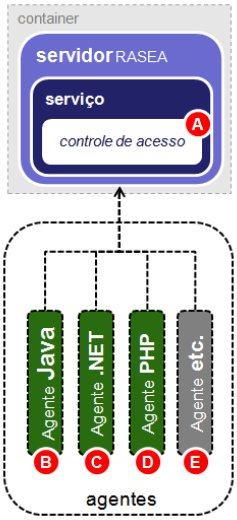

Figura 4.5 – Tipos de agente

A Figura 4.5 destaca a especialização do agente RASEA em diferentes tecnologias. O item A representa os serviços providos pelo servidor, que são acessados pelos agentes. Os itens B, C e D representam especializações de agentes, construídos em tecnologias específicas, tais como: Java, .NET e PHP. O item E representa agentes desenvolvidos em qualquer tecnologia que suporte o consumo de Web Services, proporcionando flexibilidade à solução.

Os agentes devem se especializar ao máximo, garantindo o encaixe perfeito com a tecnologia da aplicação parceira. Como exemplo, pode-se citar a utilização de Java na construção de aplicações. Existem diversas ferramentas, paradigmas, *frameworks*, bibliotecas e tecnologias que podem ser utilizadas na construção de aplicações, tais como: *JavaServer Faces* (JSF), *Apache Struts*; *Google Web Toolkit* (GWT), *Abstract Window Toolkit* (AWT), dentre outras. É fundamental que exista um agente para cada tecnologia específica.

O servidor RASEA é também uma aplicação que requer o controle de acesso às suas funcionalidades. Este requisito possui um caráter recursivo (DAVISON; HINKLEY, 1997), visto que o servidor acessa os seus próprios serviços para garantir o controle de acesso a si mesmo, caracterizando-o também como uma aplicação parceira.

O agente conectado ao servidor RASEA foi implementado na tecnologia *Seam Security* (SEAM, 2009), visto que o servidor é compatível com o *JBoss Seam Framework* para platafor-

ma Java. O processo de interceptação das requisições à aplicação está de acordo com a especificação oficial da ferramenta (SEAM, 2009), materializada na classe *RaseaPermissionResolver*.

A premissa para todo agente é atuar no ponto central, onde todas as requisições aos recursos da aplicação sejam interceptadas. Para conseguir isto, existem soluções simplórias e soluções arrojadas. Um exemplo de solução para agentes *Web* é a inserção da *tag* <*include*> em todas as páginas. Esta é uma solução simples, porém pouco elegante. Como exemplo de solução robusta para a plataforma Java, tem-se o *Java Authentication and Authorization Service* (JAAS), que atua diretamente no núcleo da máquina virtual.

5. Considerações finais

Neste capítulo, serão apresentados os resultados obtidos a partir de análises elaboradas durante o desenvolvimento deste trabalho. Foram detectadas oportunidades de melhoria da solução proposta, motivando a sua evolução em trabalhos futuros. Por fim, conclui-se que o projeto RASEA é capaz de promover a redução de custos no desenvolvimento de aplicações comerciais e a simplificação na administração do controle de acesso de aplicações, inclusive no SERPRO.

5.1. Resultados obtidos

Os cenários apresentados na evolução do projeto foram estrategicamente selecionados para experimentar e validar o objetivo geral deste trabalho, que é "prover uma solução orientada a serviços que suporte o controle de acesso unificado de usuários às aplicações, fundamentando-se em conceitos e tecnologias existentes para garantir a interoperabilidade em ambientes heterogêneos" (capítulo 1.2.).

O projeto está em constante evolução e pode ser acompanhado pela rede social Twitter (http://twitter.rasea.org). No decorrer deste trabalho, programadores autônomos ofereceram contribuições e algumas empresas demonstraram interesse na implantação do RASEA em suas instalações.

O *design* de uma solução que concentra os elementos essenciais de controle de acesso baseado nas experiências adquiridas durante a evolução do projeto é a principal contribuição deste trabalho. No decorrer do desenvolvimento do AvanSEG, S4A e RASEA os usuários elogiaram a simplicidade no controle de acesso e gerenciamento das permissões.

A simplicidade no uso (Anexo B – Exemplo de uso) foi uma decisão de projeto que permitiu

não apenas a atuação de profissionais não especialistas em segurança na administração de acesso a aplicações, como também melhorou o controle na segurança do sistema, dado que o administrador não se perde em demasiados detalhes, concentrando-se nos elementos essenciais do controle de acesso.

5.2. Trabalhos futuros

Durante o desenvolvimento deste trabalho foram identificadas diversas oportunidades de melhoria e evolução do projeto ainda não implementadas, enumeradas a seguir: implementação do *Hierarchical* e *Constrained* RBAC; criação de agentes em diferentes linguagens de programação; seleção de outros métodos de autenticação e; avaliar e integrar com mais tecnologias de mercado.

Como contribuição prática deste trabalho, construiu-se o agente para a plataforma Java integrado com a ferramenta *Seam Security*. É fundamental a criação de novos agentes em outras plataformas e tecnologias, tais como: ASP, .NET e PHP, além de oferecer integrações com métodos de autenticação utilizando certificados digitais e ferramentas de *Single Sign-On* (SSO).

Está em andamento a criação de um agente para o Demoiselle[1], que permitirá a integração do RASEA com qualquer aplicação que utiliza o *framework* de governo. Este agente pode ser empregado para viabilizar a unificação do controle de acesso de todas as aplicações corporativas construídas no SERPRO, promovendo uma solução padronizada para as questões relacionadas a autorização, mantendo os mecanismos atuais de autenticação baseados nos diretórios de usuários existentes (notadamente o SENHA-REDE e a Rede Local de Software Livre). Clientes e demais órgãos de governos poderiam fazer o mesmo, potencializando o uso e evolução do RASEA em conjunto com o Demoiselle enquanto projetos *open source* de governo.

Indo além do raciocínio acima, os sistemas unificados de permissões poderiam ser oferecidos como serviços em nuvem. Pode-se imaginar um cenário onde, em poucos minutos, o cliente teria à sua disposição uma instância confiável e segura do RASEA com o Demoiselle, permitindo o imediato controle de acesso das suas aplicações (que também poderiam estar na nuvem). Pensar em *Cloud Computing* amplia as possibilidades e favorece a incorporação de tecnologias como REST e NoSQL, podendo aumentar ainda mais a escalabilidade e perfomance da solução atual.

1 O Framework Demoiselle é a integração de várias tecnologias de *software* e uma arquitetura de referência, focados no desenvolvimento de aplicações Java/Web, provendo independência por meio de padronização. http://www.frameworkdemoiselle.gov.br

Reuso, padronização, economia, confiabilidade, segurança, interoperabilidade, serviços, evolução e, principalmente, inovação. Além de conceitos, este trabalho apresenta uma solução comprovada – com altíssimo potencial de evolução – para um problema recorrente não só no SERPRO. Ao unificar esforços em torno das necessidades e objetivos comuns, todos ganham. Para mais informações técnicas sobre o RASEA, acesse o site oficial do projeto: http://www.rasea.org

Anexo A – Módulo administrativo

A interface gráfica foi projetada para atender as solicitações dos usuários, almejando a facilitação e simplificação no gerenciamento de permissões.

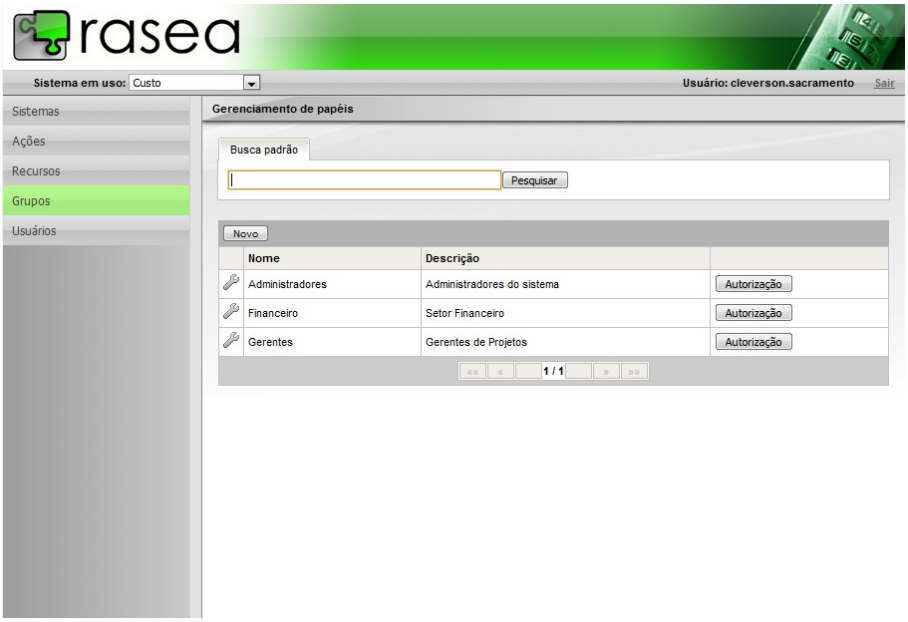

Ilustração 1 – Foram incluídos filtros nas listagens para facilitar a busca, conforme solicitações dos usuários.

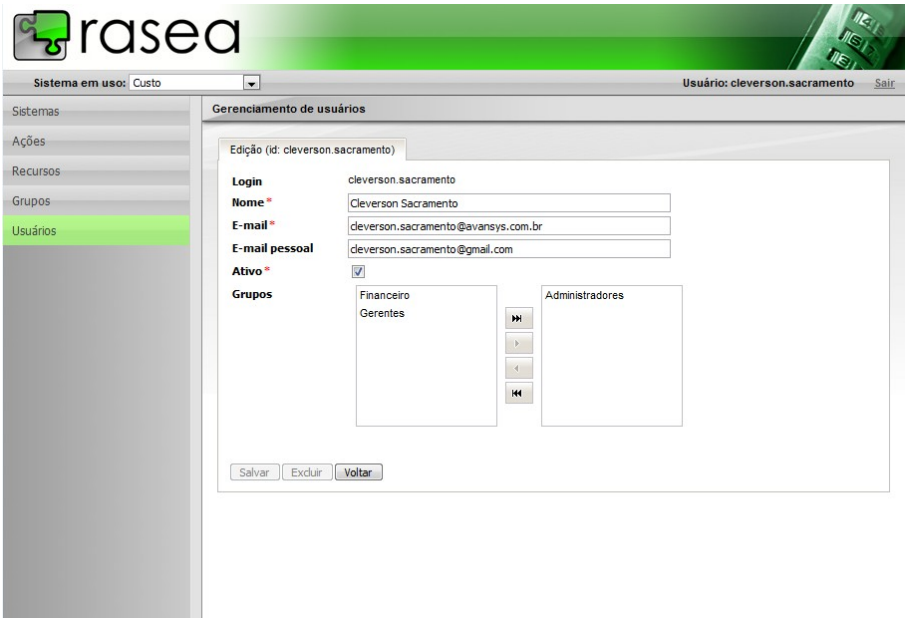

Ilustração 2 – Cada listagem antecede a tela de cadastro ou alteração de registros.

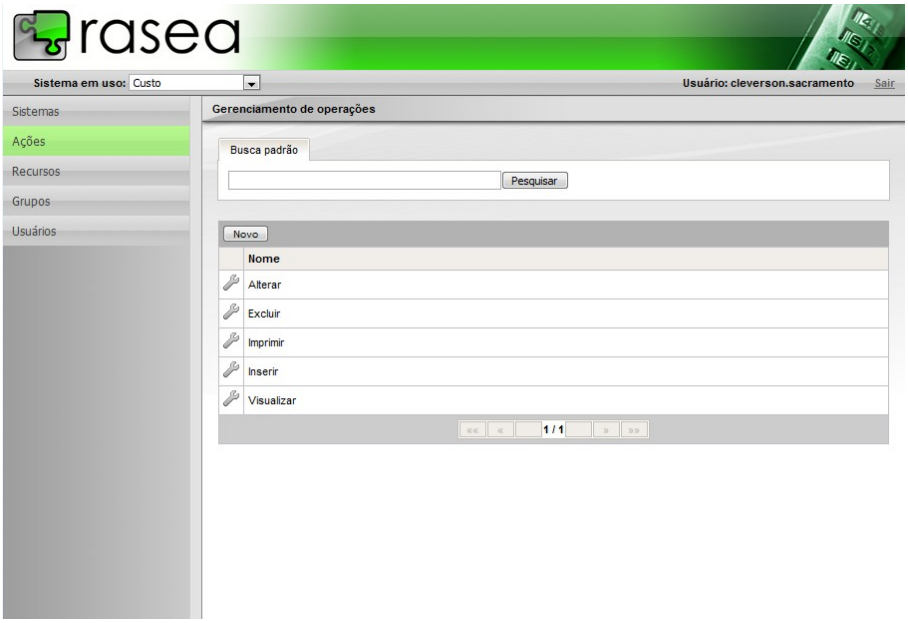

Ilustração 3 – Listagem de todas as ações associadas a uma aplicação.

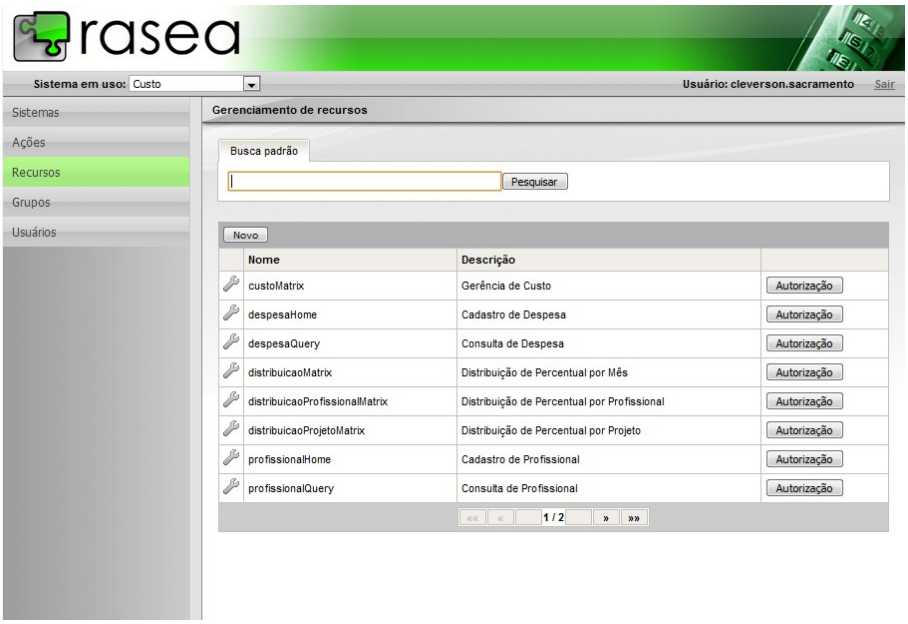

Ilustração 4 – Listagem de todos os recursos do sistema de custos.

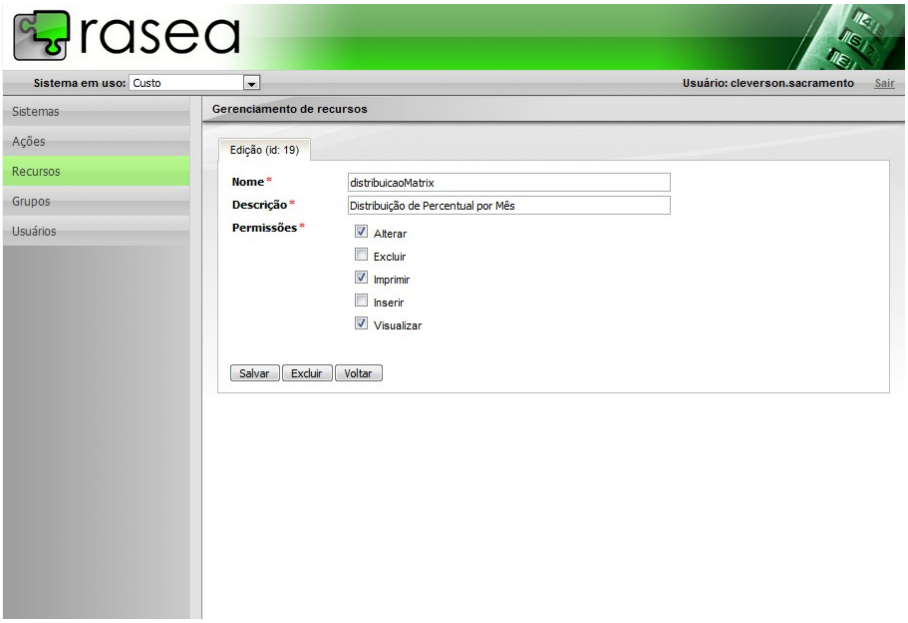

Ilustração 5 – Alteração dos dados de um recurso e associação das possíveis operações, também conhecida como permissões.

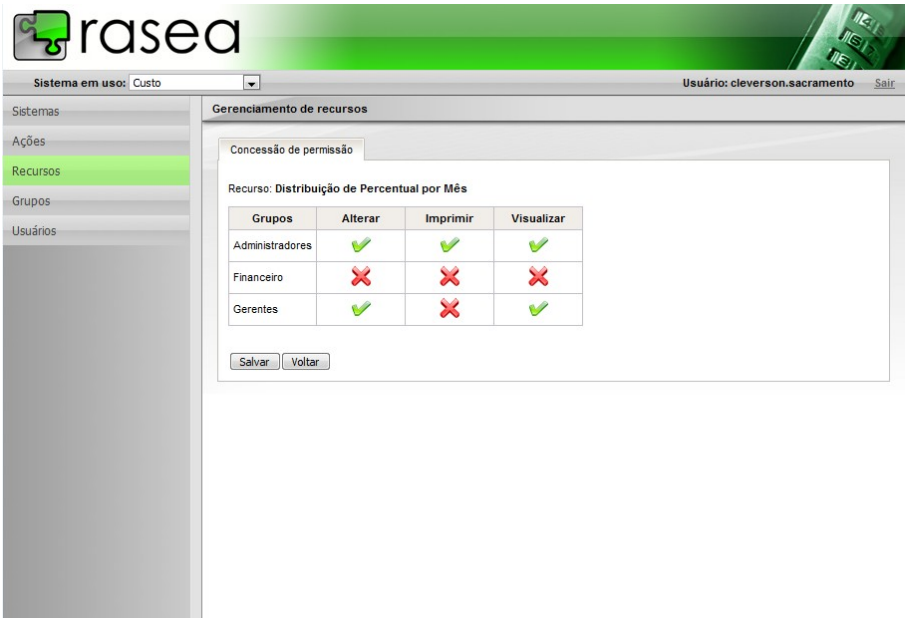

Ilustração 6 – Concessão das permissões de um determinado recurso para os papéis do sistema.

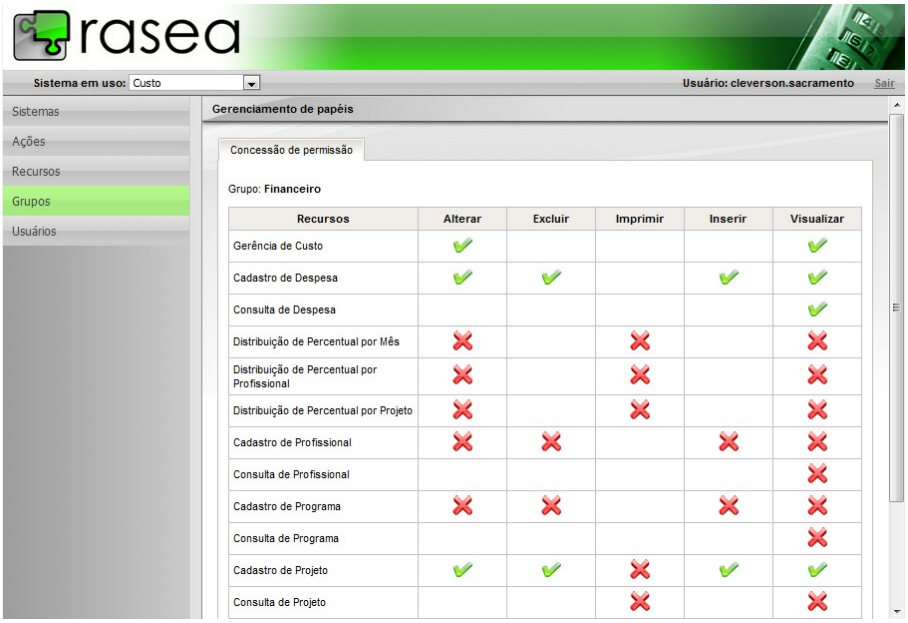

Ilustração 7 – A mesma funcionalidade da tela anterior, porém possibilitando o gerenciamento das permissões sob a perspectiva de um determinado papel.

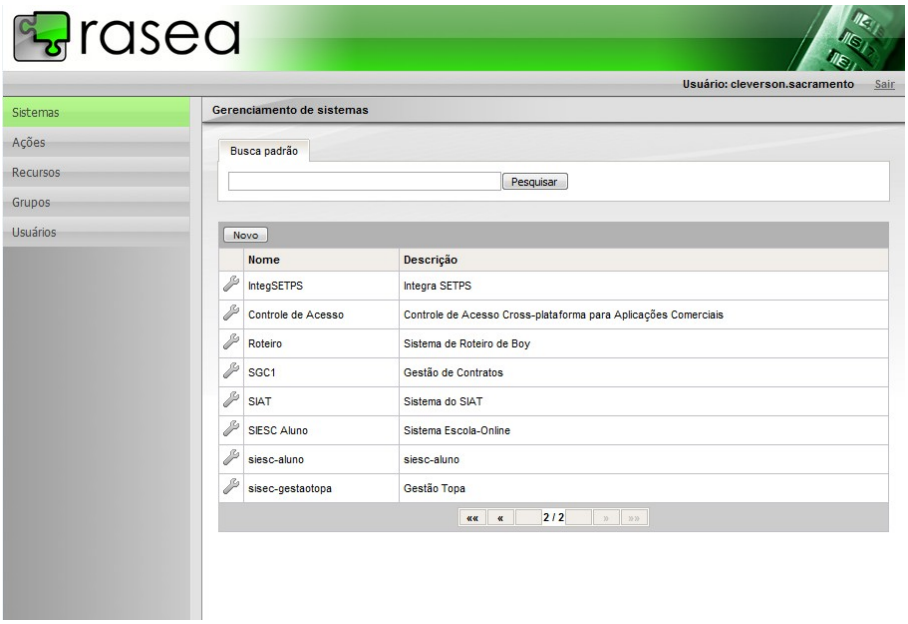

Ilustração 8 – Listagem das aplicações cadastrados que possuem o seu controle de acesso gerenciado pelo RASEA.

Anexo B – Exemplo de uso

Para exemplificar o uso do RASEA, foi utilizado a aplicação de exemplo *contactlist*. Como esta aplicação foi desenvolvida pelo grupo JBoss utilizando a ferramenta *Seam Framework*, reutilizamos o agente RASEA para o *Seam Security*. O código-fonte da aplicação de exemplo original está disponível no seguinte repositório:

http://anonsvn.jboss.org/repos/seam/tags/JBoss_Seam_2_2_0_GA/examples/contactlist/

Como o código-fonte original não contém diretivas de segurança, a aplicação foi modificada. Incluiu-se a biblioteca *rasea-seam-agent.jar*, que é ativada a partir da configuração do arquivo *components.xml* do *JBoss Seam*. Segue o exemplo da configuração com destaque para a simplicidade:

```
<?xml version="1.0" encoding="UTF-8"?>
<components xmlns="http://jboss.com/products/seam/components" (...)
      xmlns:rasea="http://rasea.org/agent/seam"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="(...)">

      <!-- Configurações do Seam sem relação com o agente RASEA -->
      (...)

      <!-- Configurações do agente Seam Security com suporte ao SSO -->
      <rasea:partner-application app-name="custo" />
      <rasea:single-sign-on-identity />
</components>
```

Na aplicação é possível utilizar a diretiva do Seam Security no arquivo JSF:

```
<h:commandLink value="Insert" action="#{contactBean.insert}"
      disabled="#{!s:hasPermission('contact', 'insert')}">
```

Ou na classe Java diretamente:

```
@Restrict("#{s:hasPermission('contact', 'insert')}")
public String insert(){ ... }
```

O processo completo de modificação na aplicação *contactlist* está disponível publicamente no YouTube (http://www.youtube.com/watch?v=DV53pW14kso).

Bibliografia

ABNT: ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 17799**: tecnologia da informação: técnicas de segurança: código de prática para a gestão da segurança da informação. Rio de Janeiro, 2005.

_____. **NBR ISO/IEC 27001**: tecnologia da informação: técnicas de segurança: sistemas de gestão de segurança da informação: requisitos. Rio de Janeiro, 2006.

ANSI: American National Standard Institute. **ANSI INCITS 359-2004**: Information Technology: Role Based Access Control. New York, 2004.

ALUR, Deepak; CRUPI, John; MALKS, Dan. **Core J2EE Design Patterns**: Best practices and Design Strategies. 2 ed. Upper Saddle River: Prentice Hall PTR, 2003. ISBN: 0-13-142246-4.

BARKER, Steve. Access control by action control. In: Symposium on Access Control Models and Technologies, 2008, Estes Park. **Anais...** New York: ACM, 2008. p. 143–152. ISBN: 978-1-60558-129-3.

BERTINO, Elisa et al. A logical framework for reasoning about access control models. In: ACM Workshop on Role Based Access Control, 2001, Chantilly. **Anais...** New York: ACM, 2001. p. 41–52. ISBN: 1-58113-350-2.

BUEGE, Brian; LAYMAN, Randy; TAYLOR, Art. **Segurança contra Hackers J2EE e Java**: Desenvolvendo aplicações seguras com a tecnologia Java. São Paulo: Futura, 2003. ISBN: 85-7413-156-3.

CUMMINS, Fred A. **Integração de Sistemas**: EAI – Enterprise Application Integration. Rio de Janeiro: Campus, 2002. ISBN: 85-352-0975-1.

DAN, Asit; JOHNSON, Robert D.; CARRATO, Tony. SOA service reuse by design. In: International Conference on Software Engineering, 2008, Leipzig. **Anais...** New York: ACM, 2008. p. 25–28. ISBN: 978-1-60558-029-6.

DAVISON, Anthony C.; HINKLEY, D. V. **Bootstrap methods and their application**. [S.l.]: Cambridge University Press, 1997. ISBN: 0521574714.

E-PING: **Padrões de Interoperabilidade de Governo Eletrônico**. Versão 2010**.** Governo Brasileiro: Comitê Executivo de Governo Eletrônico. Disponível em: <http://www.governoeletronico.gov.br/anexos/e-ping-versao-2010>. Acesso em: 27 ago. 2010.

FERRAIOLO, David F. et al. Proposed NIST standard for role-based access control. **ACM Transactions on Information and System Security (TISSEC)**, v. 4, n. 3, p. 224–274, 2001. ISSN:1094-9224.

GAEDKE, Martin; MEINECKE, Johannes; NUSSBAUMER, Martin. A modeling approach to federated identity and access management. In: International World Wide Web Conference, 14, 2005, Chiba. **Anais...** New York: ACM, 2005. p. 1156–1157. ISBN: 1-59593-051-5.

GAMMA, Erich et al. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000. ISBN: 85-7307-610-0.

GPL: **GNU General Public License**. Version 3. [S.l.]: Free Software Foundation, 2007. Disponível em: <http://www.gnu.org/licenses/gpl.html>. Acesso em: 25 ago. 2010.

IBM. **The PIMA Project**: Platform-Independent Model for Applications. [S.l.]: IBM Research. Disponível em <http://www.research.ibm.com/PIMA/>. Acesso em: 25 de ago. 2010.

ISO: International Organization for Standardization. **ISO/IEC 10181-3**: information technology, open systems interconnection, security frameworks for open systems: access control framework. [S.l.], 1996.

JAAS, **Java Authentication and Authorization Service**: Reference Guide. Version 1.5.0. [S.l]: Sun Microsystems, 2001. Disponível em: <http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASRefGuide.html>. Acesso em: 25 ago. 2010.

KERN, Axel et al. A meta model for authorisations in application security systems and their integration into RBAC administration. In: Symposium on Access Control Models and Technologies, 9, 2004, Yorktown Heights. **Anais...** New York: ACM, 2004. p. 87–96. ISBN: 1-58113-872-5.

LI, Jun; KARP, Alan H. Access control for the services oriented architecture. In: Workshop On Secure Web Services, 2007, Fairfax. **Anais...** New York: ACM, 2007. p. 9–17. ISBN: 978-1-59593-892-3.

LGPL: **GNU Lesser General Public License**. Version 3. [S.l.]: Free Software Foundation, 2007. Disponível em: <http://www.gnu.org/licenses/lgpl.html>. Acesso em: 25 ago. 2010.

RAY, Indrakshi et al. Using uml to visualize role-based access control constraints. In: Symposium on Access Control Models and Technologies, 2004, New York. **Anais...** New York: ACM, 2004. p. 115–124. ISBN: 1-58113-872-5.

SANDHU, Ravi; FERRAIOLO, David; KUHN, Richard. **The NIST Model for Role-Based Access Control**: Toward A Unified Standard. [S.l]: National Institute of Standards and Technology, 2000. Disponível em: <http://cs-www.ncsl.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf>. Acesso em: 25 ago. 2010.

SEAM Security: **The Seam Security API**. Version 2.1.1.GA. [S.l], 2009. Disponível em: <http://docs.jboss.org/seam/2.1.1.GA/reference/en-US/html/security.html>. Acesso em: 25 ago. 2010.

SHAW, Mary; GARLAN, David. **An Introduction to Software Architecture**. School of Computer Science – Carnegie Mellon University: Pittsburgh, 1994.

SILVA, Lino S. **Public Key Infrastructure PKI**: Conheça a Infra-estrutura de Chaves Púbicas e a Certificação Digital. São Paulo: Novatec, 2004. ISBN: 85-7522-046-2.

STANSBERRY, Brian; ZAMARRENO, Galder. **JBoss Application Server Clustering Guide**. [S.l.]: JBoss Labs. Disponível em: <http://labs.jboss.com/file-access/default/members/jbossas/freezone/docs/Clustering_Guide/4/html/index.html>. Acesso em: 25 ago. 2010.

VAZQUEZ, Carlos E.; SIMÕES, Guilherme S.; ALBERT, Renato M. **Análise de Pontos de Função**: Medição, Estimativas e Gerenciamento de Projetos de Software. 4. ed. São Paulo: Érica, 2003. ISBN: 85-7194-899-2.