

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CAMPUS DE PALMAS
CURSO TECNÓLOGO EM SISTEMAS PARA INTERNET

**MOISES DA SILVA LIMEIRA COELHO
URIAS CRUZ DA CUNHA**

**PROCESSO ADMINISTRATIVO ELETRÔNICO COM USO DE ASSINATURA
DIGITAL**

PALMAS-TO
2011

**MOISES DA SILVA LIMEIRA COELHO
URIAS CRUZ DA CUNHA**

**PROCESSO ADMINISTRATIVO ELETRÔNICO COM USO DE ASSINATURA
DIGITAL**

Trabalho de Conclusão do Curso de
Sistemas para Internet para obtenção do
título de tecnólogo em sistemas para
internet do Instituto Federal de Educação,
Ciência e Tecnologia do Tocantins.
Orientador: Wilson Wolf Costa.

PALMAS-TO
2011

C672p COELHO, Moises da Silva Limeira,
Processo Administrativo Eletrônico com uso de Assinatura Digital /
Moises Limeira, Urias Cruz – – Palmas, 2011.
108f. : il.

Trabalho de Conclusão de Curso (Graduação Tecnológica em
Sistemas para Internet) - Instituto Federal de Educação, Ciência e
Tecnologia do Tocantins – Campus Palmas, 2011.
Orientador(a): Prof. Wilson Wolf Costa

1. Gerenciamento – Eletrônico de documentos. 2. Criptografia. 3.
Certificação Digital. I. Instituto Federal de Educação, Ciência e
Tecnologia do Tocantins – Campus Palmas. II. Moises da Silva
Limeira Coelho. III. Urias Cruz da Cunha. IV. Título.

CDD 004

**MOISES DA SILVA LIMEIRA COELHO
URIAS CRUZ DA CUNHA**

**PROCESSO ADMINISTRATIVO ELETRÔNICO COM USO DE ASSINATURA
DIGITAL**

Trabalho de Conclusão de Curso foi julgado e aprovado como cumprimento às exigências legais do currículo do Curso Superior de Tecnologia em Sistemas para Internet pela Coordenação da Área de Informática do Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – Campus de Palmas.

Palmas, 30 de junho de 2011.

Banca Examinadora:

Wilson Wolf Costa
IFTO

Fernando Jorge Ebrahim Lima e Silva
IFTO

Alysson Martins Bruno
IEPO

DEDICATÓRIA

*Dedicamos este trabalho a todos aqueles
que, de alguma forma, contribuíram para sua
realização.*

AGRADECIMENTOS

A Deus, Regente do universo.

Aos nossos pais, que nos apoiaram em mais essa jornada.

Ao nosso orientador, que soube compreender as dificuldades que passamos e que nos deu sustentação para mais este desafio.

Agradecemos aos professores e colaboradores do Instituto Federal de Educação Ciência e Tecnologia do Tocantins, com as quais aprendemos a conviver e a crescer.

Muita estima e respeito ao Tribunal Regional Eleitoral do Tocantins, onde atualmente trabalhamos.

Aos amigos e colegas de turma, mesmo aqueles que não se fazem presentes neste momento de conclusão, mas que em algum momento estiveram presentes em nossa vida acadêmica.

RESUMO

Este trabalho aborda de uma maneira simples, porém clara, os princípios da tecnologia de Gerenciamento Eletrônico de Documentos (GED) e tem como objetivo proporcionar uma compreensão dos conceitos básicos e das principais questões relativas ao gerenciamento de documentos/processos eletrônicos. A primeira parte do trabalho aborda um resumo sobre a descrição e tecnologias empregadas. A segunda parte trata sobre o desenvolvimento do sistema utilizando tecnologias com ênfase à segurança da informação, como: técnicas de criptografia, assinatura digital e certificação digital que tornam a adoção de processo administrativo eletrônico e documento eletrônico um processo seguro e confiável. Por fim, o trabalho fornece informações básicas para aqueles que gostariam de exercer atividades nesta área, através do desenvolvimento de um protótipo de sistema, ora denominado ePad .

Palavras-chave: gerenciamento eletrônico de documentos, segurança da informação, *hash*, *hashing*, resumo de mensagem, criptografia, criptografia assimétrica, chave pública, chave privada, assinatura digital, certificado digital, certificação digital, protótipo, ePad.

ABSTRACT

This dissertation discusses the principles of Electronic Document Management (EDM) technology in a simple and clear way and aims to provide an understanding about basic concepts and major issues concerning electronic document/process management. The first part of this dissertation deals with a description and overview of the technologies used. The second part deals with the development of the system using technologies with emphasis on information security, as encryption techniques, digital signature and digital certification to make the adoption of electronic administrative process and electronic document a secure and reliable process. At last, the dissertation provides basic information for those who wish to perform activities in this area by developing a prototype system, now named EPAD.

Keywords: electronic document management, information security, hash, hashing, message digest, cryptography, asymmetric cryptography, public key, private key, digital signature, digital certificate, prototype, ePad.

LISTA DE ABREVIATURAS

AC	Autoridade Certificadora
AR	Autoridade de Registro
AKID	<i>Authority Key Identifier</i>
API	<i>Application Programming Interface</i>
CMS	Cryptographic Message Syntax
CPU	<i>Central Processing Unit</i>
CSR	Requisição para Assinatura de Certificado
MER	Modelo Entidade-Relacionamento
DN	<i>Distinguished Name</i>
ePad	Sistema Eletrônico de Processos Administrativos
GED	Gerenciamento Eletrônico de Documentos
HTML	<i>HyperText Markup Language</i>
IDE	<i>Integrated Development Environment</i>
JCA	<i>Java Cryptography Archicture</i>
JDK	Kit de Desenvolvimento Java
JPA	<i>Java Persistence API</i>
JSF	<i>JavaServer Faces</i>
JVM	Máquina Virtual Java
LCR	Lista Certificados Revogados
MVC	<i>Model View Controller</i>
PIN	Personal Identification Number
RFC	<i>Request for Comments</i>
SSL	<i>Secure Sockets Layer</i>
STF	Supremo Tribunal Federal
TRE-TO	Tribunal Regional Eleitoral do Tocantins
TI	Tecnologia da Informação
UML	<i>Unified Modeling Language</i>
URL	<i>Uniform Resource Locator</i>

LISTA DE FIGURAS

Figura 1 – Diagrama Caso de Uso – Modelo	20
Figura 2 – Diagrama de Classe – Modelo	21
Figura 3 – Modelo Lógico do DER	22
Figura 4 – Java Persistence API – Java Noroeste (2007)	25
Figura 5 – Arquitetura em alto nível do <i>Hibernate</i> (SAUVÉ, s.d)	26
Figura 6 – ePad – Tela inicial.....	28
Figura 7 – ePad – Classe de Validação de Login.....	29
Figura 8 – ePad – Tela e mensagem de retorno de login.....	29
Figura 9 – ePad – Administração do Sistema –Tipo de Processo	30
Figura 10 – ePad – Administração do Sistema –Tipo de Processo	30
Figura 11 – ePad – Administração do Sistema –Tipo de Processo – Resultado da busca...	31
Figura 12 – ePad – Administração do Sistema –Tipo de Processo - Lista	31
Figura 13 – ePad – Administração do Sistema –Tipo de Processo - Visualização	32
Figura 14 – ePad - Administração do Sistema – Tipo de Processo - Edição.....	32
Figura 15 – ePad - Administração do Sistema – Tabela: Tipo de Processo - Edição	32
Figura 16 – ePad – Lista de Certificados	33
Figura 17 – ePad – Cadastro de Certificado - Upload	34
Figura 18 – ePad – Cadastro de Certificado – Dados do Certificado	34
Figura 19 – ePad - Cadastro de Certificado – Falha em Upload	35
Figura 20 – ePad – Cadastro de Processo – Etapa: Processo.....	36
Figura 21 – ePad – Cadastro de Processo – Etapa: Processo (complemento)	37
Figura 22 – ePad – Cadastro de Processo – Etapa: Documento	37
Figura 23 – ePad – Cadastro de Processo – Etapa: Documento (complemento).....	38
Figura 24 – ePad – Cadastro de Processo – Etapa: Tramitação.....	39
Figura 25 – ePad – Cadastro de Processo – Etapa: Tramitação (complemento)	39
Figura 26 – ePad – Cadastro de Processo Sigiloso	40
Figura 27 – ePad – Caixa de Processo.....	41
Figura 28 – ePad – Caixa de Processo – Visualizar Processo.....	42
Figura 29 – ePad - Guarda de Processo.....	43
Figura 30 – ePad – Arquivamento de Processo	44
Figura 31 – ePad – Arquivamento de Processo - Confirmação.....	44
Figura 32 – ePad – Envio de Email – Classe HistoricoMovimentacaoController	45
Figura 33 – ePad – Classe Email.....	46

Figura 34 – Representação pictórica de um documento de GED (SILVA, 2001).....	47
Figura 35 – Processo de assinatura digital sem certificação (CertSign).....	52
Figura 36 - Bastão de Licurgo (CertSign).....	55
Figura 37 – Processo de criptografia com chave simétrica (CertSign)	57
Figura 38 – Criptografia com uso de chaves assimétricas (CertSign)	58
Figura 39 – Ataque de personificação na assinatura (CertSign)	60
Figura 40 – Man-in-the-middle no sigilo (CertSign)	61
Figura 41 – Conteúdo Básico do Certificado (CertSign).....	62
Figura 42 – Modelo de Certificado de Usuário Final (CertSign)	62
Figura 43 – Distribuição de Chaves Públicas (CertSign).....	63
Figura 44 – Cadeia de Certificação (CertSign).....	65
Figura 45 – Certificado Auto-assinado da AC Raiz	67
Figura 46 – Parâmetro <i>basicConstraints</i> presente no arquivo <i>/etc/ssl/openssl.cnf</i>	68
Figura 47 – Certificado da AC Intermediária emitido pela AC Raiz	68
Figura 48 – Certificado de Assinatura de Código	70
Figura 49 – Modelo de LCR (SwissSign)	73
Figura 50 – Autenticando o usuário no token SafeNet iKey 2032	77
Figura 51 – Importação de certificado digital no SafeNet iKey 2032	78
Figura 52 – Informando a senha de descriptografia do arquivo .pfx e o nome do certificado alojado no token	78
Figura 53 – Token criptográfico com um certificado digital.....	79
Figura 54 – Requisição de um cliente e resposta com o envio de um applet (Adaptado: COULOURIS, DOLLIMORE, KINDBERG).....	81
Figura 55 – Execução de código applet no ambiente do cliente.....	81
Figura 56 – Arquivo MANIFEST.MF original	82
Figura 57 – Ilustração do modelo de segurança com o uso do <i>sandbox</i> (Oracle)	83
Figura 58 – Modelo de segurança a partir do JDK 1.1 com a adoção de código assinado (Oracle).....	83
Figura 59 – Arquivo MANIFEST.MF alterado	84
Figura 60 – Arquivo MANIFEST.SF	85
Figura 61 – ePad – Trecho do arquivo formProcessoDocumento.xhtml.....	86
Figura 63 – Carregando o documento	86
Figura 63 – ePad – Trecho do método jButtonCarregarActionPerformed.....	87
Figura 64 – Cards.java - Trecho de código que armazena os dados dos tokens	90
Figura 65 – ePad – Seleccionando o token e inserindo a senha.....	90
Figura 66 – Instalação do provider SunPKCS#11	90
Figura 67 – Recuperando uma instância do KeyStore	91

Figura 68 – Recuperando a chave privada	91
Figura 69 – Recuperando o certificado digital do usuário.....	91
Figura 70 – Recuperando a cadeia de certificados	91
Figura 71 – Caminhos de Certificação (Binder, J., C.)	92
Figura 72 – ePad – Assinando o Documento	92
Figura 73 – Mozilla Firefox –Autoridades Certificadoras Confiáveis.....	93
Figura 74 – ePad - Método <i>loadRemoteTrustedAnchor</i> da classe <i>TrustedAnchorParser</i>	95
Figura 76 – ePad - Criação dos objetos CMSProcessable e CertStore.....	98
Figura 76 – ePad - Assinando o documento	99

SUMÁRIO

1. INTRODUÇÃO	13
2. OBJETIVOS.....	14
2.1. Objetivos Gerais.....	14
2.2. Objetivos Específicos	14
3. JUSTIFICATIVA.....	15
4. DESCRIÇÃO DO SISTEMA.....	17
4.1. Levantamento de Requisitos	18
4.1.1. Análise dos Requisitos.....	18
4.1.2. Premissas e Restrições	19
4.2. Modelagem do sistema	20
5. TECNOLOGIAS E PADRÕES	23
5.1. Java	23
5.2. JSF.....	23
5.3. PrimeFaces	24
5.4. API JPA.....	24
5.5. PostgreSQL.....	26
5.6. Método de Pesquisa.....	26
5.7. Modelo de Desenvolvimento	26
5.7.1 Modelo Incremental	27
6. DESENVOLVIMENTO DO SISTEMA.....	28
6.1. Administração do Sistema.....	28
6.1.1. Tipos de Processo	30
6.1.2. Certificado.....	33
6.2. Processo	35
6.2.1. Cadastro	36
6.2.2. Sigilo.....	39
6.2.3. Caixa de Processo.....	41
6.2.4. Envio de Email.....	44
6.3. Documento.....	46
6.3.1 Gerenciamento Eletrônico de Documentos - GED	47
7. ASSINATURA DIGITAL.....	50
7.1. Conceitos e definições de assinatura digital	50

7.2. Resumo da Mensagem (Message Digest) - Hash	52
7.3. Criptografia.....	54
7.3.1. Criptografia de Chave Simétrica	56
7.3.2. Criptografia de Chave Pública e Algoritmos de Assinatura	57
7.4. Certificação Digital e Infraestrutura de Chave Pública	59
7.4.1. Certificado Digital.....	61
7.4.2. Autoridade Certificadora	63
7.4.3. Hierarquia de Certificados Digitais	64
7.4.4. Geração de certificado para Autoridade Certificadora Raiz e Intermediária	66
7.4.5. Geração de certificado para assinatura de códigos.....	69
7.4.6. Geração de certificado para Assinatura de Documentos.....	70
7.4.7. Geração da Lista de Certificados Revogados	71
7.5. Aspectos Legais para Utilização de Assinatura Eletrônica	73
7.6. Dispositivos Criptográficos	75
7.6.1. O padrão PKCS#11	76
7.6.2. Copiando o certificado para dentro do token	77
7.7. Processo de Assinatura de Documentos	79
7.7.1. Código móvel.....	80
7.7.2. Carregando o arquivo para assinatura	86
7.7.3. Acessando tokens criptográficos.....	87
7.7.4. Validação do caminho do certificado.....	92
7.7.5. Assinando o documento.....	97
7.7.6. Enviando o documento e a assinatura	99
7.7.7. Validação do documento e assinatura no servidor	100
8. CONCLUSÃO	101
8.1. Considerações Finais	101
REFERÊNCIAS BIBLIOGRÁFICAS	103
APÊNDICE	108

1. INTRODUÇÃO

Desde a antiguidade o homem registra informações em suas mais variadas formas, tais como os da pré-história que deixaram registros por meio de pinturas em rochas e os egípcios com seus sinais de escrita talhados em pedra. Esse processo tem se repetido e evoluído no decorrer do tempo, levando-nos a adotar modelos para padronização e classificação da informação.

Com modelos de documentos bem definidos e estruturados, criaram-se processos para geração e manuseio das informações que vêm sendo persistidas em papel. Até anos atrás, estes modelos eram capazes de atender aos requisitos de negócio, entretanto, com o crescente volume de informação gerada, novos modelos precisam ser definidos para suportar as demandas atuais e vindouras. Além disso, a informação passou a agregar maior valor às organizações. Neste novo cenário, a informação deixou de ser um produto meio para ser o produto final, tornando-se vital para as instituições.

Em organizações que manipulam processos, sejam eles jurídicos ou administrativos, o volume de impressos gerados é bastante elevado. A gerência dessas informações apresenta um alto custo financeiro e uma baixa eficiência nas operações de manuseio. É atribuído a esses fatores o fato das informações serem criadas, reproduzidas e armazenadas em papel.

Atualmente existe uma forte tendência para utilização de ferramentas de gerenciamento eletrônico de documentos. Isso pode ser visto em organizações privadas e públicas. O setor público tem buscado melhorias para os processos de gerência de documentos, impulsionado tanto pelos requisitos de negócio quanto pela necessidade de conformidade com leis e regulamentos vigentes.

2. OBJETIVOS

2.1. Objetivos Gerais

Desenvolver um sistema de gerenciamento eletrônico de documentos que tenha como foco processos administrativos assinados digitalmente. O sistema deve atender aos requisitos propostos, considerando as tecnologias previamente determinadas e o tempo de desenvolvimento estabelecido em cronograma.

2.2. Objetivos Específicos

- Descrever as principais características do processo eletrônico e analisar os requisitos necessários para sua aplicação no âmbito organizacional;
- Estudar sobre as tecnologias envolvidas que serão utilizadas no desenvolvimento;
- Desenvolver o sistema de processo administrativo eletrônico com suporte a assinatura digital;

3. JUSTIFICATIVA

Na tentativa de otimizar o processo de gerenciamento de documentos, instituições públicas e privadas têm adotado as mais variadas soluções, partindo desde a conversão de documentos impressos em micro-filmes até a digitalização por meio de scanners e, em seguida, armazenamento em mídias digitais. (TADANO 2002)

As instituições, em suas atividades ordinárias, geram um grande volume de documentos, arquivos e processos. Esse material é manuseado constantemente por meio de consultas, inclusões, ratificações, retificações e arquivamento. O fluxo de controle pode ser manual ou eletrônico, contudo, é comum encontrarmos a informação precípua (documento, arquivos, processos, etc.) existente somente em sua forma física, apostada em papel.

A utilização do papel como forma de registro apresenta um custo significativo ao longo de sua existência. A busca e a recuperação de documentos e processos são lentas quando comparadas ao tempo de resposta obtido em um sistema de gerenciamento eletrônico de documentos - GED. Além do mais, as informações existentes em documentos impressos apresentam menor flexibilidade de acesso, bem como estão sujeitas a degradação, adulteração e extravio. (SILVA, 2001)

Seguindo uma tendência de mercado, adotaremos uma solução que vise a redução de custos e que seja economicamente viável. A proposta deve suprir as principais demandas existentes no modelo tradicional de gerenciamento de documentos, como a redução de custos com suprimento para impressão, diminuição no tempo de busca por processos, prevenção de extravio e adulteração de documentos. Juntamente com as respostas às demandas, obtêm-se as vantagens que um sistema computacional pode oferecer.

A adoção de um sistema computacional que permita o acesso a informação a partir de pontos remotos é essencial para o sucesso do projeto, pois, além de ir ao encontro das tendências de mercado, oferece uma maior disponibilidade da informação aos usuários. É preciso considerar também os controles necessários da segurança da informação.

A área de Tecnologia da Informação – TI – tem proposto soluções para as mais diversas áreas do conhecimento, proporcionando maior celeridade e

produtividade nos processos executados por meio de computadores. Uma das soluções de controle administrativo criada pela TI para o é o GED, sistema computacional que tem a finalidade de gerenciar e controlar documentos eletrônicos visando a otimização das tarefas e atividades desempenhadas por aqueles que manipulam processos administrativos e arquivos em papel. Segundo TADANO (2006, p.20) o GED é capaz de reduzir os custos com impressão, uma vez que os documentos são gerados e persistidos digitalmente, garantir a alta disponibilidade de acesso à informação, reduzir gastos com armazenamento e conservação, pois o documento eletrônico não necessita de espaço físico e sim lógico, garantir a segurança da informação prevenindo contra o acesso indevido, o extravio dos dados e a adulteração com a adoção da assinatura digital.

O uso de um sistema de gerenciamento eletrônico de documentos e processos em uma entidade governamental poderá proporcionar um ganho efetivo no que diz respeito ao gerenciamento de seus processos. Somando-se a isso, o uso da ferramenta está em conformidade com a Lei nº 11.419/06 que dispõe sobre a informatização dos processos.

Com foco nas características citadas, o projeto propõe o desenvolvimento de um sistema de gerenciamento eletrônico de documentos capaz de atender os requisitos de negócio levantados durante sua análise.

4. DESCRIÇÃO DO SISTEMA

O sistema atenderá ao controle de processos administrativos com uso das tecnologias de gerenciamento eletrônico de documentos e uso de assinatura digital. No que diz respeito ao acesso e utilização do sistema, possuirá dois grupos de perfis: administrativo e de usuário, possuindo este sub-níveis, conforme o setor que pertence.

- **Nível Administrativo**
 - Deverá ser composto por usuários com privilégios administrativos, sendo responsável por gerir os cadastros básicos do sistema, alimentar e controlar as tabelas de apoio ao sistema, tais como: tipo de processo, tipo de documento, setor, autoridade certificadora, etc: Os cadastros de novos usuários serão feitos na área administrativa do sistema, assim como a redefinição das senhas desses usuários.
- **Nível de Setor**
 - O usuário de setor é o usuário do tipo genérico, ou seja, são as pessoas que utilizarão o sistema para movimentar processo, juntar e assinar documento. O usuário deverá possuir um diretório lógico contendo a lista de processos que estão em seu setor ou que fora enviado diretamente a um usuário de seu setor, sendo esse local denominado caixa de processos.
 - Deverá ser possível realizar consulta de processos baseada em alguns atributos, tais como número e/ou ano do processo, tipo de processo, assunto, palavra chave, data de criação. Nessa consulta, o usuário poderá visualizar e imprimir todos os documentos do processo, exceto os que tramitam em sigilo e que não lhe foi dada permissão nem ao seu setor.

Para que um usuário possa manipular um processo é necessário a obtenção da guarda do processo. A obtenção da guarda de um processo está condicionada a localização do processo, ou seja, o processo deverá se encontrar no setor a que o usuário pertence e não estar sob guarda de outro usuário.

Nos processos em que foi realizada a obtenção da guarda, o usuário poderá juntar documento ou despacho e também tramitá-lo. Quando da inclusão de um documento, o usuário deverá ter aplicado a assinatura digital para que o documento possa ser inserido no sistema. Na tramitação, o usuário deverá informar o local /

destino do processo, escolhendo entre um setor ou um usuário de um determinado setor.

Quando o processo estiver sob guarda e não for do tipo sigiloso, o usuário terá permissão somente para visualizar e imprimir documentos desse processo.

Os níveis de acesso do sistema variam de acordo com o setor a que esse usuário pertence:

- Biblioteca
 - Esse usuário possui algumas funções privativas, como é o caso do arquivamento / desarquivamento de processo.

A biblioteca é o local de arquivamento dos processos da instituição, sendo que o seu arquivamento deverá ser feito com a tramitação do processo juntamente com um despacho solicitando o arquivamento. Do mesmo modo, o desarquivamento deverá ser solicitado à biblioteca.

- Protocolo
 - O usuário de protocolo possui as mesmas permissões que um usuário de setor, com a adição da permissão para cadastro de processos que é exclusiva a este.

É no protocolo da instituição que um processo tem sua origem, onde o cadastramento de um processo é realizado por meio de uma demanda inicial. Para a realização do cadastro é necessário incluir o documento de abertura, demandado por um usuário ou setor da instituição, sendo obrigatório o uso de assinatura digital do requerente ou do usuário de protocolo.

4.1. Levantamento de Requisitos

4.1.1. Análise dos Requisitos

A etapa inicial do sistema foi baseada na tendência de mercado e no uso da ferramenta SADP¹ que atende à Justiça Eleitoral. A utilização do SADP no acompanhamento de processos tornou possível a obtenção de uma visão geral sobre a movimentação de processos e também da inclusão de despachos.

¹ Sistema de Acompanhamento de Documentos e Processos

O embasamento do sistema desenvolvido deu-se em sua grande maioria no PAE², desenvolvido e utilizado no Tribunal Regional Eleitoral do Paraná (TRE-PR), sendo esse um sistema de processos administrativos que utiliza assinatura digital nos documentos de seus processos.

Outra fonte utilizada foi o conhecimento e a demonstração do uso da ferramenta de processos judiciais utilizada pelo Tribunal de Justiça do Tocantins (TJ-TO) foi o Projudi, desenvolvido pelo Conselho Nacional de Justiça (CNJ) e utilizado em todo país.

Cabe aqui ressaltar que o conhecimento prévio sobre sistemas de protocolo, processos judiciais e administrativos contribuíram e também foram colocados em prática por parte dos desenvolvedores no projeto do sistema.

4.1.2. Premissas e Restrições

O projeto do sistema foi desenvolvido para fins acadêmicos, não havendo finalidade lucrativa. O desenvolvimento do sistema abrange recursos humanos, de sistema de software e de hardware, limitados sob as seguintes condições:

- O sistema será de controle de processos, categoria administrativa;
- Linguagem de codificação *Java*;
- Base de Dados compatível com o *framework Hibernate*;
- A modelagem do sistema será feita utilizando a tecnologia *Unified Modeling Language* (UML);
- A assinatura digital utilizada será a CMS/PKCS#7;
- Os *tokens* suportados pelo sistema são limitados a dois modelos: iKey 2032 e eToken Alading ;
- Nenhum usuário, setor ou administrador poderá ser excluído do sistema, para que não se perca a auditoria e as restrições de integridade do banco de dados, podendo, entretanto, serem desativados;
- O processo sigiloso será exibido na caixa do usuário, porém os documentos desse processo não estarão disponíveis para visualização caso o usuário não possua autorização de acesso, limitando-o à visualização das informações do cabeçalho do processo e ao seu histórico / movimentação;

² Processo Administrativo Eletrônico

- O ato de obter guarda não pode ser feito para um processo sigiloso que não tenha sido tramitado diretamente para o usuário ao seu setor;
- O usuário deverá dispor de conexão direta à internet sem a necessidade do uso de proxy para possibilitar o *download* da Lista Certificados Revogados (LCR).

4.2. Modelagem do sistema

4.2.1 Caso de Uso

“Um caso de uso é uma descrição de um conjunto de sequências de ações, inclusive variantes que um sistema executa para reproduzir um resultado de valor observável por um ator”. (BOOCH, 2005, p. 40). As ações que são executadas no diagrama abaixo representado são executadas por um ator. Já um caso de uso é representado por uma elipse.

A interação do usuário com o sistema ePad pode ser vista nos diagramas de caso de uso localizados no apêndice deste trabalho.

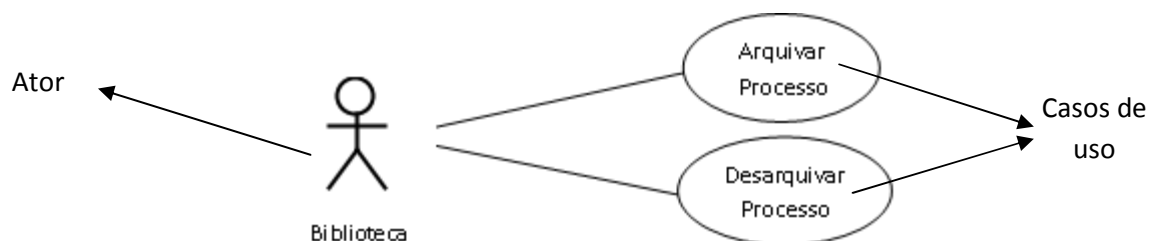


Figura 1 – Diagrama Caso de Uso – Modelo

4.2.2. Diagrama de classe

Quando se inicia o desenvolvimento de um sistema dividimos a problemática da programação em objetos, termo utilizado para representar uma entidade do mundo real, abstraindo-o. Os objetos assim como no mundo real possuem propriedades e métodos.

Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações e relacionamentos e semântica. Em UML as classes

são representadas por um retângulo dividido em três compartimentos: (BOOCH, 2005)

- Nome: identificação da classe modelada;
- Atributos: características do objeto que a classe representa;
- Operações: que serão os métodos de manipulação de dados e de comunicação de uma classe com outras do sistema.

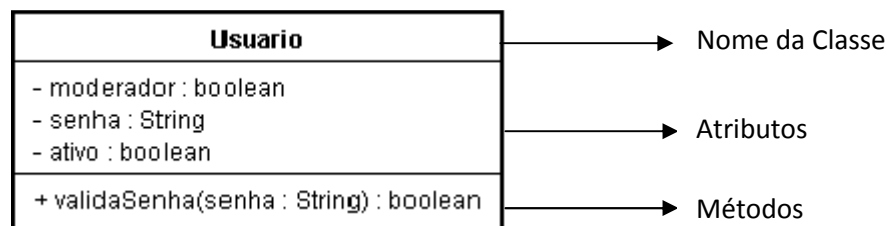


Figura 2 – Diagrama de Classe – Modelo

Além das classes modeladas, existe um tipo chamado *enum*. Um tipo *enum* é um tipo cujos campos consistem de um conjunto fixo de constantes (ORACLE, s.d.), ou seja, quando utilizamos uma quantidade já conhecida de valores constantes, como, por exemplo, os dias da semana.

4.2.2. Modelo Entidade-Relacionamento (MER)

O Modelo Entidade-Relacionamento (MER) tem o objetivo de representar as estruturas de dados da forma mais próxima do mundo real dos negócios. O Diagrama Entidade Relacionamento (DER) pode ser construído através de um MER. Esse modelo foi concebido por Edgar Frank Codd, nos anos 70, e é um modelo de dados baseado na lógica de predicados e na teoria dos conjuntos. O DER mostra graficamente os relacionamentos entre as entidades do banco de dados, o que justifica a nomenclatura atribuída a ele. (RUMBAUGH, JACOBSON, 2005)

Ainda segundo Codd, o modelo lógico relacional é uma “ferramenta para descrição de estruturas de dados em uma forma passível de ser processada por um SGB”.

Desta forma, o modelo lógico dos relacionamentos entre as entidades propostos para o sistema ePad estão representados na figura abaixo:

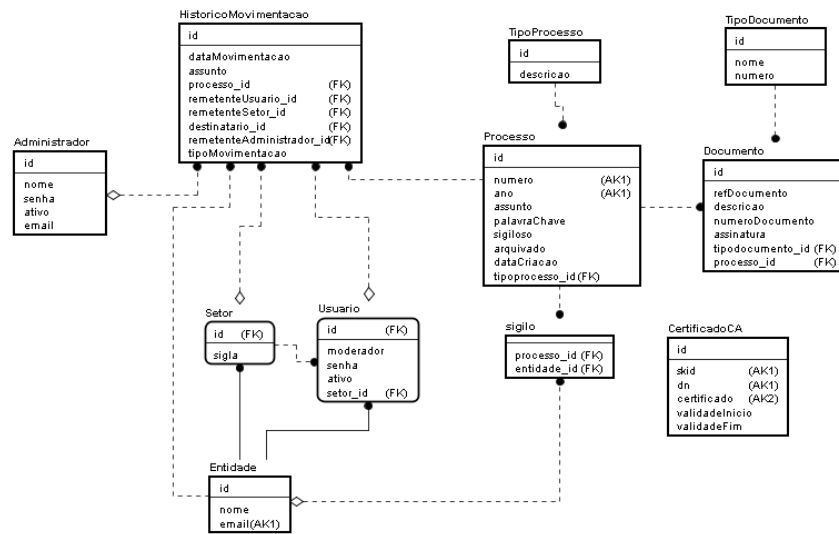


Figura 3 – Modelo Lógico do DER

5. TECNOLOGIAS E PADRÕES

5.1. Java

A linguagem de programação utilizada no desenvolvimento do ePad foi a tecnologia *Java*, que é uma linguagem orientada a objetos desenvolvida pela *Sun Microsystems*. Ela foi escolhida dentre as demais pela sua independência de plataforma, popularidade e nível de conhecimento entre os integrantes da equipe de desenvolvimento.

Como ferramenta de apoio ao desenvolvimento de software, utilizamos a interface de desenvolvimento - *Integrated Development Environment* (IDE) - *Netbeans* versão 6.9, juntamente com o Kit de Desenvolvimento *Java* (JDK), que contém o ambiente necessário para a criação e execução de aplicações *java*, o que inclui a Máquina Virtual *Java* (JVM), compilador e *APIs*.

5.2. JSF

JavaServer Faces – JSF é um *framework* de desenvolvimento web que implementa o modelo *Model View Controller* (MVC)³. As aplicações de múltiplas camadas em *Java* costumam ser implementadas com a utilização de recursos do *Java Enterprise Edition*. (DEITEL; 2009)

Dentre as vantagens da utilização desse *framework*, destacamos a implementação do modelo MVC, em que a manutenção da aplicação web tornou-se menos complexa e mais escalável, permitindo a divisão dos papéis e extensibilidade, o que possibilita a agilidade no desenvolvimento.

Atualmente o JSF encontra-se na versão 2, sob a especificação sob JSR⁴ - 314, sendo adotada a versão 2.0.3 no sistema ePad.

³ Arquitetura para plataforma de aplicação *Java TM 2 Enterprise Edition* (J2EE), que separa o modelo de negócio da lógica de apresentação. (Oracle, 2011, tradução nossa)

⁴ Proposta de Especificação *Java*

5.3. PrimeFaces

O JSF nativamente traz consigo componentes visuais prontos. Como destacado anteriormente, um dos seus pontos fortes é possibilidade de extensão e criação de componentes. Existem hoje no mercado diversos componentes visuais para o JSF, como *ICEFaces*, *RichFaces*, *PrimeFaces*, dentre outros.

O *PrimeFaces* foi uma das bibliotecas que primeiramente ofereceu suporte à nova versão do JSF. Embora os demais componentes ofereçam suporte à versão 2.0 do JSF, mesmo que em forma de compatibilidade, foram verificadas incompatibilidades com a versão atual do JSF.

Um dos destaques que contribuiu para a escolha desse componente, além do suporte ao JSF 2.0, foi a facilidade de integração com o projeto web. Para que o *PrimeFaces* funcione no projeto, devemos incluir o arquivo *.jar* nas bibliotecas do projeto, não sendo necessária nenhuma configuração adicional em qualquer arquivo de configuração do projeto. Para começarmos a utilizá-lo, basta o declararmos na página em que desejamos utilizá-lo, conforme pode-se verificar no arquivo */ePad/seções/content.xhtml*, linha 4:

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://primefaces.prime.com.tr/ui">
```

5.4. API JPA

O termo API, do inglês, "*Application Program Interface*", é definida como uma interface de programação de aplicativos. O dicionário *Tech Terms Computer Dictionary* (TECHTERMS.COM, s.d.) define API como:

Uma *API* é um conjunto de comandos, funções e protocolos que os programadores podem usar na construção de um software, em um determinado sistema operacional. Uma *API* permite aos programadores usarem funções pré-definidas para interagir com o sistema operacional, em vez de escrevê-los do zero.

Desse modo, o sistema ePad utilizou uma API de persistência em *java* – *Java Persistence API* (JPA) – em seu desenvolvimento. Essa API lida com a forma como os dados relacionais são mapeados para objetos *Java* e a forma com que esses objetos são armazenados em um banco de dados relacional, possibilitando que

esses objetos possam ser acessados posteriormente e permitindo a existência de um estado da entidade, mesmo após o término do aplicativo. Além de simplificar o modelo de persistência da entidade, o JPA padroniza o mapeamento objeto-relacional. (BISWAS, 2006) Alguns *frameworks*, como o *Hibernate*, implementam a JPA.

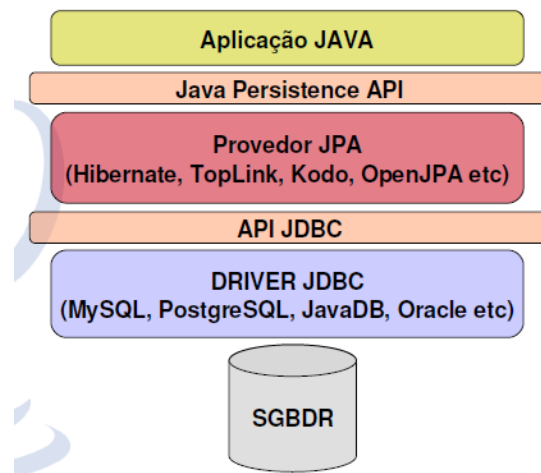


Figura 4 – Java Persistence API – Java Noroeste (2007)

Hibernate é um *framework* de mapeamento objeto/relacional escrito em *java*. O termo Mapeamento Objeto/Relacional, do inglês *ORM*, refere-se à técnica de mapeamento de dados, onde as tabelas são representadas pelas classes e os registros uma instância dessas classes, ou seja, transformando-se os dados tabulares em um grafo de objetos. (LINHARES, s.d.)

O *Hibernate* possui linguagem própria de *Structured Query Language (SQL)*, denominada *HQL*⁵, que é convertida para *SQLs* específicas de cada banco de dados, um dos recursos que faz desse *framework* bastante conhecido e utilizado para independência de banco de dados.

A técnica de ORM com *Hibernate* também está disponível para a linguagem *.Net*. Na configuração dos mapeamentos em *java* podemos utilizar arquivos *XML*⁶ ou o sistema de anotações que a linguagem oferece. No sistema desenvolvido, optamos pela utilização de anotação.

O “*Hibernate* objetiva reduzir em cerca de 95% do tempo de desenvolvimento de tarefas relacionadas à persistência!” (SAUVÉ, s.d)

⁵ Hibernate Query Language

⁶ Extensible Markup Language

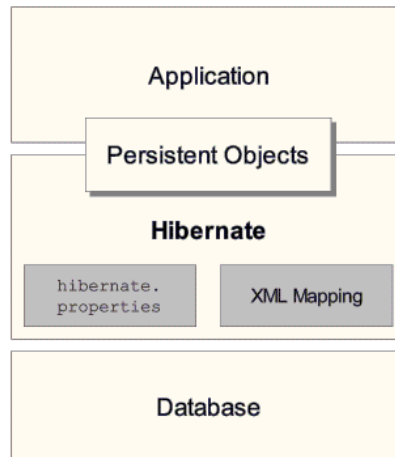


Figura 5 – Arquitetura em alto nível do *Hibernate* (SAUVÉ, s.d)

5.5. PostgreSQL

O Sistema de Gerenciamento de Banco de Dados (SGBD) utilizado no sistema é o PostgreSQL. Dentre suas principais características, o que definiu sua escolha foi o fato de ser *opensource*⁷, compatível com os sistemas operacionais mais conhecidos e por haver afinidade entre membros da equipe com esse SGBD.

5.6. Método de Pesquisa

O método científico de pesquisa adotado nesse projeto foi o método experimental.

5.7. Modelo de Desenvolvimento

O software, como todo sistema complexo, evolui com o passar do tempo [GIL88]. Os requisitos do negócio e do produto mudam frequentemente à medida que o desenvolvimento prossegue, dificultando um caminho direto para um produto final; prazos reduzidos de mercado tornam impossível completar um produto abrangente, mas uma versão reduzida pode ser elaborada para fazer face à competitividade ou às pressões do negócio; um conjunto de requisitos básicos de um produto ou sistema é bem entendido, mas os detalhes das extensões do produto ou sistema ainda precisam ser definidos. Nesse caso e em situações semelhantes,

⁷ Código aberto (tradução)

os engenheiros de software precisam de um modelo de processo que tenha sido explicitamente projetado para acomodar um produto que evolui com o tempo. (PRESSMAN, 2006)

5.7.1 Modelo Incremental

O modelo de desenvolvimento utilizado no ePad foi o incremental. Esse modelo é definido por Pressman (2006) como:

O modelo de processo incremental é iterativo igual à prototipagem, mais diferente da prototipagem o incremental tem como objetivo apresentar um produto operacional a cada incremento realizado. Esse modelo é muito útil quando a empresa não possui mão de obra disponível no momento para uma implementação completa, dentro do prazo estipulado.

6. DESENVOLVIMENTO DO SISTEMA

6.1. Administração do Sistema

A parte administrativa do sistema é o local onde um administrador previamente cadastrado no sistema com esse perfil e que não figura entre as entidades do sistema, como os setores e os usuários de setor, gerencia as tabelas auxiliares do sistema. A interface de *login* no sistema é única, sendo necessário para um administrador no momento do *login* marcar o *checkbox* “Administrador”, de modo que o sistema possa autenticá-lo como um usuário pertencente ao grupo de administradores.

Figura 6 – ePad – Tela inicial

A distinção de validação de *login* entre um usuário de setor e um administrador é dada por meio do método *isAdmin()*, que retorna um valor do tipo booleano. O método *isAdmin()* é implementado na classe *LoginController*.

Em Java, o nome dos métodos começando com *get* ou *set* não são obrigatórios, mas sim uma convenção de atribuição de valores. A adoção é altamente recomendada, pois seu uso é requerido para componentes de software *java* denominados *JavaBeans*, o que simplifica a programação na maioria dos IDEs. Os métodos que começam com *set* são utilizados como modificadores de acesso e os que começam com o *get* são para retorno do valor da variável (DEITEL, 2009).

No caso do método `isAdmin()`, que começa com “is” ao invés de “get”, o prefixo foi alterado em decorrência do seu tipo de retorno, neste caso um booleano.

Dessa forma, quando o método que verifica o tipo de usuário (Figura 7 - linha 113) retorna `TRUE`, os métodos necessários são testados, exibindo ao final uma mensagem (linhas 109 a 111) através de um balão, de acordo com o retorno desses testes.

```

107 public String validaLogin() throws Exception {
108     final String erroInvalido = "Usuário e/ou senha inválida!";
109     final String erroInativo = "Usuário inativo!";
110     final String sucesso = "Bem-vindo!";
111
112     if (isAdmin()) {
113         AdministradorDAO dao = new AdministradorDAO();
114         Administrador administrador = new Administrador();
115         administrador.setEmail(email);
116         administrador = dao.loadObjectByEmail(administrador);
117         if (administrador == null || !administrador.validaSenha(
118             this.senha)) {
119             geraMensagemErro(erroInvalido);
120             limparCampos();
121             return null;
122         } else if (!administrador.isAtivo()) {
123             geraMensagemErro(erroInativo);
124             limparCampos();
125             return null;
126         } else {
127             geraMensagemSucesso(sucesso);
128             limparCampos();
129             setAdministrador(administrador);
130             return "index";
131         }
132     } else {

```

Figura 7 – ePad – Classe de Validação de Login



Figura 8 – ePad – Tela e mensagem de retorno de login

Após efetuar login e receber permissão de acesso, o responsável pela gerência do sistema terá controle sobre as tabelas tipo de processo, tipo de

documento, setor, certificado, usuário e administrador, conforme nota-se na figura acima que exibe o menu superior.

Nos cadastros ora habilitados no menu será possível a manipulação dos dados que compõem o sistema. A manipulação desses dados compreende: a inclusão, alteração e exclusão das informações, com exceção de alguns cadastros, como, por exemplo, de usuários e administradores, que por questões de integridade do sistema não são passíveis de exclusão, podendo somente ser ativados ou desativados.

6.1.1. Tipos de Processo

A seguir demonstraremos a manutenção do cadastro dos Tipos de Processos, ressaltando que as ações existentes nos demais cadastros seguem o mesmo padrão no ePad.

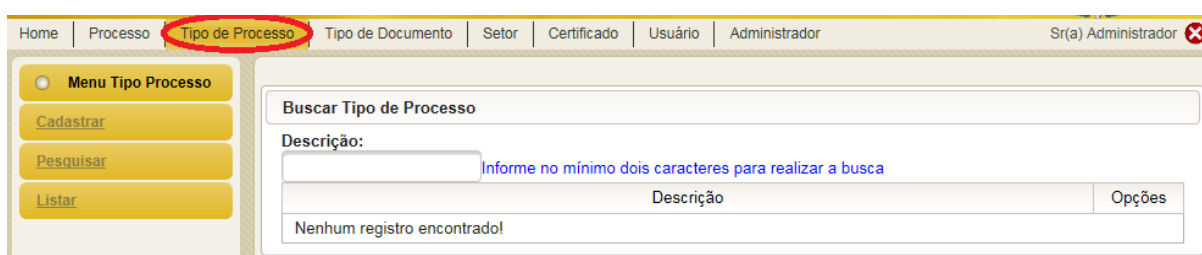


Figura 9 – ePad – Administração do Sistema –Tipo de Processo

Ao se clicar sobre o menu superior Tipo de Processo, esse menu será destacado com a cor amarela, para que o usuário saiba qual área do sistema está manipulando, tanto por questões de usabilidade como também para distinção entre os cadastros. As telas de cadastro são semelhantes, a exemplo do menu Tipo de Documento e Tipo de Processo que possui mesma estrutura e disposição em seu menu esquerdo: Cadastrar, Pesquisar e Listar.

Para a inserção de um registro, é necessário clicar no menu esquerdo sobre o botão “Cadastrar”. Será então exibida uma tela de cadastro com os campos a serem preenchidos.

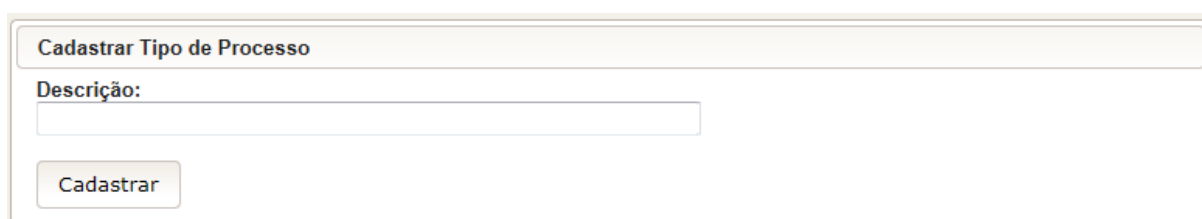


Figura 10 – ePad – Administração do Sistema –Tipo de Processo

A tela que será exibida ao usuário quando um item do menu superior for clicado é a de busca. Nela é possível realizar a localização rápida de um ou mais registros com base em um atributo que compõe a o cadastro. A pesquisa será feita pela descrição do tipo de processo, através do recurso autocompletar, onde resultados serão exibidos ao usuário a partir da inserção do segundo caractere no campo descrição. Caso a busca não retorne um resultado, será exibida a informação na tabela: “nenhum registro encontrado”.

Descrição	Opções
Pagamento de Diária	  
Pagamento de Fornecedor	  

Figura 11 – ePad – Administração do Sistema –Tipo de Processo – Resultado da busca

A tela que exibe os registros retornados na consulta assemelha-se a opção Listar, presente no menu esquerdo. Esta opção exibe todos os registros presentes na tabela, agrupando-os a cada dez registros por página para que a visualização desses dados não se estenda sobre a tela quando houver na base de dados uma grande quantidade de registros.

Descrição	Opções
Pagamento de Diária	  
Pagamento de Fornecedor	  
Aquisição de Material	  
Contratação de Serviço	  
Processo Administrativo Disciplinar	  
Sindicância	  

Figura 12 – ePad – Administração do Sistema –Tipo de Processo - Lista

A partir da exibição dos registros, pode-se ativar a editá-los por meio de um *click* sobre os botões exibidos na coluna Opções. São os botões, a saber: Visualizar, Editar e Excluir. A seguir demonstraremos as interfaces de cada uma dessas ações.

- Visualizar:



Figura 13 – ePad – Administração do Sistema –Tipo de Processo - Visualização

A ação de clicar sobre o botão visualizar será exibida uma caixa de diálogo ao usuário que contendo os detalhes do registro, ou seja, as informações que foram omitidas por questão de espaço na tela de registro serão exibidas nessa caixa. Também é possível proceder a edição do registro clicando no botão “Editar” na caixa de visualização (Figura 13).

- Editar:

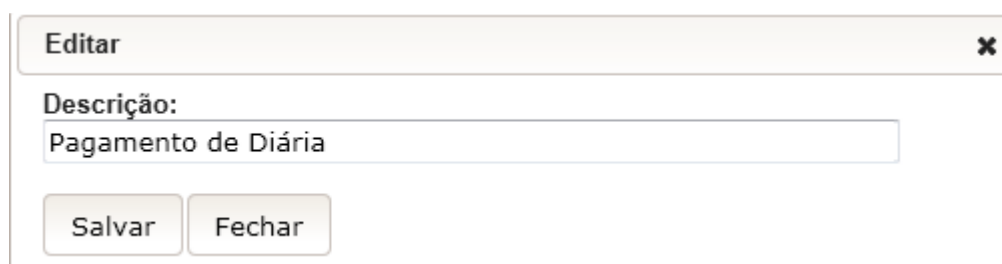


Figura 14 – ePad - Administração do Sistema – Tipo de Processo - Edição

Após a edição do registro, o usuário deverá confirmar a alteração clicando sobre o botão “Salvar”. Caso deseje cancelar a edição, poderá fazê-lo clicando sobre o botão “Fechar” ou pressionando a tecla *Esc* do teclado.

- Excluir:

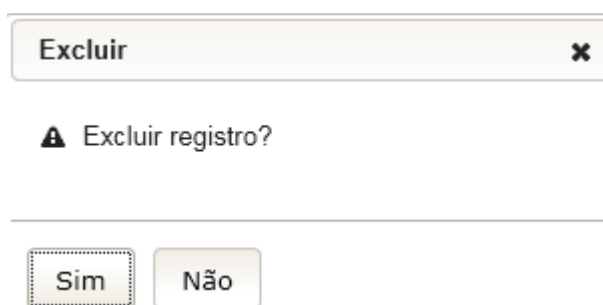


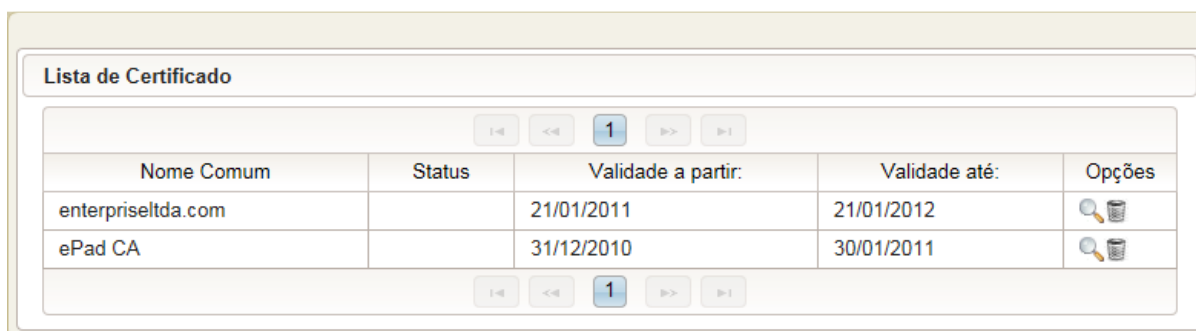
Figura 15 – ePad - Administração do Sistema – Tabela: Tipo de Processo - Edição

Ao se clicar sobre a opção Excluir, será exibida uma caixa de diálogo, conforme figura acima, solicitando ao usuário a confirmação da operação solicitada. Ao clicar sobre o botão “Sim”, o registro é excluído do sistema, não sendo possível reverter a ação. Caso seja clicado o botão “Não”, da mesma forma como na edição,

ou seja pressionando a tecla *Esc*, a caixa de diálogo de confirmação é fechada sem a exclusão do registro

6.1.2. Certificado

O repositório de certificados digital de Autoridade Certificadora (AC) Confiáveis é gerida pelo administrador do sistema. A lista de certificados cadastrados é exibida quando se clica no menu superior na opção “Certificado”.



Nome Comum	Status	Validade a partir:	Validade até:	Opções
enterpriseltda.com		21/01/2011	21/01/2012	 
ePad CA		31/12/2010	30/01/2011	 

Figura 16 – ePad – Lista de Certificados

A figura acima ilustra os certificados cadastrados no sistema, assim como os seus atributos. Os botões à direita, na coluna Opções, são: “Visualizar detalhes do cadastro” e “Excluir”, respectivamente.

O processo de cadastramento de certificado é realizado através do envio do arquivo que contém o certificado. Clicando-se no botão “Selecionar”, é exibida uma caixa de diálogo que permite ao usuário selecionar o certificado que se deseja enviar, como pode ser visto na figura abaixo:

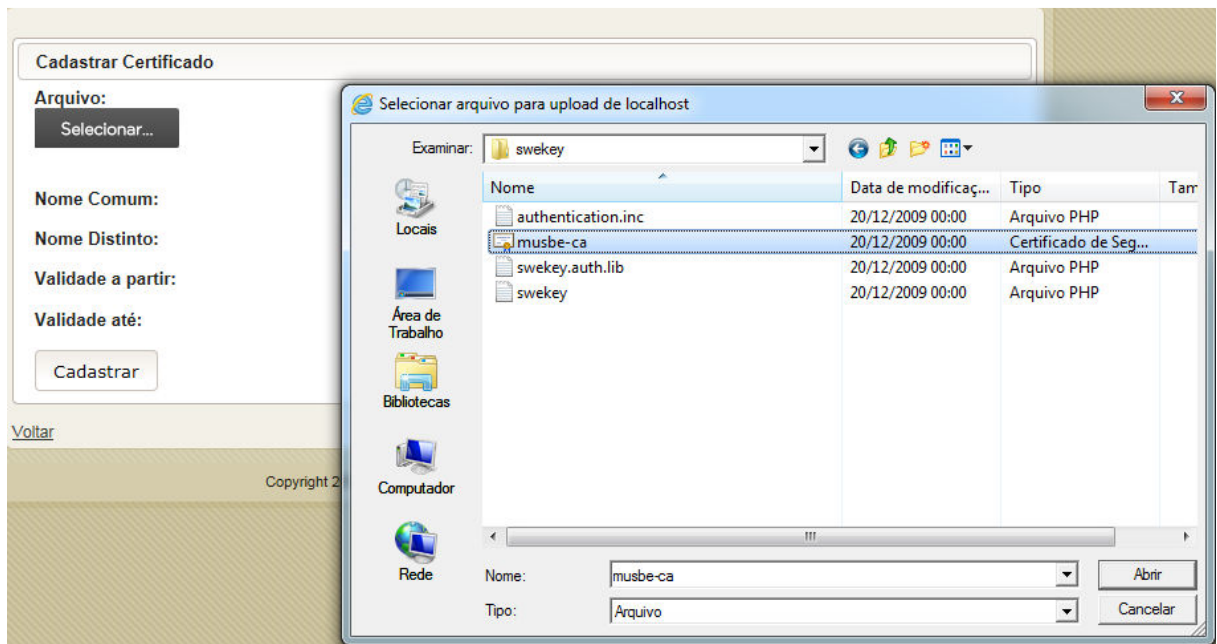


Figura 17 – ePad – Cadastro de Certificado - Upload

Nesse instante, ao clicar na opção “Abrir” presente na caixa de diálogo aberta, o navegador submeterá o certificado ao sistema e exibirá ao usuário os dados relativos ao certificado.

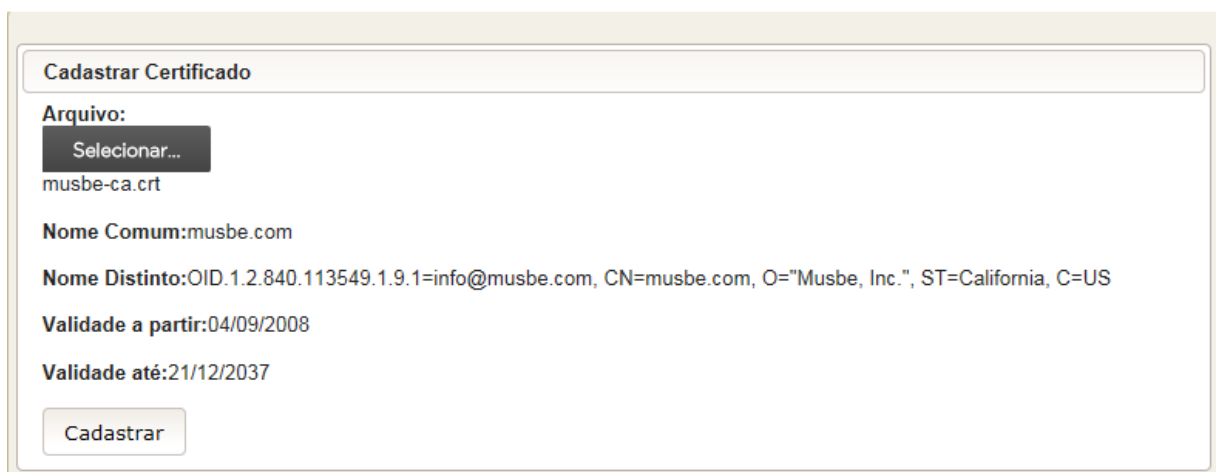


Figura 18 – ePad – Cadastro de Certificado – Dados do Certificado

Após conferir dos dados, o usuário administrador clicará sobre o botão Cadastrar para efetivação do cadastro. Caso ocorra algum erro na verificação e validação dos dados desse certificado, será apresentada uma tela de erro impedindo a continuação do processo de cadastro.

Figura 19 – ePad - Cadastro de Certificado – Falha em Upload

6.2. Processo

As instituições utilizam-se de processos em suas atividades para diversas finalidades.

Um processo é um grupo de tarefas interligadas logicamente, que utilizam os recursos da organização para gerar os resultados definidos, de forma a apoiar os seus objetivos. (Harrington, 1993)

A definição de processo para o judiciário pode ser entendida como um conflito entre partes, onde um interessado levanta uma questão para ser resolvida pelo poder judiciário, este por sua vez determinará a melhor solução para a lide. Na esfera administrativa é a maneira que o estado toma suas decisões, seja por iniciativa própria ou particular.

Neste sentido, Pietro (1999, p. 343) ratifica:

Assim, pode-se falar em processo num sentido muito amplo, de modo a abranger os instrumentos de que se utilizam os três Poderes do Estado – Judiciário, Legislativo e Executivo, para a consecução de seus fins. Cada qual, desempenhando funções diversas, se utiliza de processo próprio, cuja fonte criadora é a própria Constituição; ela estabelece regras fundamentais de competência e de forma, institui os órgãos, define suas atribuições, tudo com o objetivo de assegurar a independência e o equilíbrio no exercício de suas funções institucionais e, ao mesmo tempo, garantir que esse exercício se faça com respeito aos direitos individuais, também assegurados pela Constituição.

O sistema ePad, em sua proposta de desenvolvimento visa automatizar a criação, consulta e tramitação de processos administrativos, eliminando o uso de papel e tendo o processo como início, no Setor de Protocolo.

6.2.1. Cadastro

Para acesso ao cadastramento de processo, o usuário deverá pertencer ao Setor de Protocolo e preencher os campos solicitados no formulário. O cadastro do processo é dividido em etapas, o que possibilita uma visão compacta da tela de cadastro, agrupada por funcionalidade, sendo elas: detalhes do processo, documento, tramitação e confirmação.

A imagem mostra a interface de usuário para o cadastro de um processo no sistema ePad. No topo, há uma barra de navegação com quatro abas: 'Processo' (destacada), 'Documento', 'Tramitação' e 'Confirmação'. Abaixo, o formulário 'Detalhes do Processo' contém os seguintes campos: 'Assunto:' com uma caixa de texto; 'Palavras chave:' com uma caixa de texto e o exemplo 'Ex: palavra1; palavra2'; 'Sigiloso:' com uma caixa de seleção desativada; e 'Tipo de Processo:' com uma caixa de texto e um ícone de lupa. No canto inferior direito, há um botão '→ Próximo'.

Figura 20 – ePad – Cadastro de Processo – Etapa: Processo

Durante o preenchimento dessa etapa o usuário deverá informar o assunto, palavras-chave, se o processo é sigiloso (ver subitem 6.2.2) e o tipo do processo. Quando o usuário clicar na lupa, ao lado da caixa de texto do Tipo de Processo, será exibida uma caixa de diálogo, em formato *modal* para que o usuário realize a pesquisa do tipo de processo e selecione-o. Em seguida deverá clicar em Próximo para avançar à próxima etapa.

Processo Documento Tramitação Confirmação

Detalhes do Processo

Assunto:
Pagamento de Diárias ref. ao XI Congresso Eleitoral

Palavras chave:
diária, congresso eleitoral Ex: palavra1; palavra2

Sigiloso:
☐

Tipo de Processo:

Pesquisar Tipo de Processo ✕

Nome:

Ok

→ Próximo

Figura 21 – ePad – Cadastro de Processo – Etapa: Processo (complemento)

A etapa seguinte é o cadastro do documento que comporá inicialmente o processo. Nesse cadastro o usuário informará os dados: tipo de documento, descrição assim como os demais dados para o documento, conforme figura a seguir:

Processo Documento Tramitação Confirmação

Documento

Tipo de Documento: 🔍

Descrição:

Token:
--Selecione-- ▼

Carregar Arquivo Assinar Arquivo Coassinar Arquivo

Arquivo carregado:
(vazio)

Dados do(s) assinate(s):

← Anterior → Próximo

Figura 22 – ePad – Cadastro de Processo – Etapa: Documento

Da mesma maneira como no tipo de processo (etapa Processo) o usuário deverá clicar na lupa, ao lado da caixa de texto tipo de documento em que será aberta uma caixa de diálogo para consulta e seleção do tipo do documento a ser

cadastrado e em seguida realizar o envio do documento (ver 7.7.3). A figura a seguir ilustra o cadastro preenchido do documento.

Processo Documento **Tramitação** Confirmação

Documento

Tipo de Documento:
Memorando

Número:
12

Descrição:
Solicitação de Pagamento

Token:
--Selecione--

Carregar Arquivo Assinar Arquivo Coassinar Arquivo

Arquivo carregado:
localprova.pdf

Dados do(s) assinate(s):

Nome: Moises Limeira
E-mail: moiseslimeira@gmail.com
Emissor: ePad CA

← Anterior → Próximo

Figura 23 – ePad – Cadastro de Processo – Etapa: Documento (complemento)

A etapa seguinte é a tramitação do processo, que conforme requisito do sistema ficou estabelecido que ao se cadastrar um processo deve ser realizada uma tramitação, denominada tramitação inicial. Durante o procedimento de tramitação, o usuário informará o assunto da tramitação e também o destinatário do processo, que poderá ser um usuário do sistema ou um setor, do mesmo modo como nos demais cadastros, o usuário deverá clicar no botão para abrir a caixa de diálogo de pesquisa, realizar a consulta e selecionar o destinatário.

Figura 24 – ePad – Cadastro de Processo – Etapa: Tramitação

A figura abaixo ilustra o formulário já preenchido dessa etapa.

Figura 25 – ePad – Cadastro de Processo – Etapa: Tramitação (complemento)

Por fim, o usuário ao clicar em próximo, poderá conferir os dados informados nas etapas do formulário para que seja possível a sua confirmação ou correção, sendo essa, a última etapa. Ao clicar no botão Anterior, retornará à etapas anterior para as devidas alterações. Caso os dados estejam em conformidade ou não haja alteração, o usuário clicará em Cadastrar para que seja feita a inserção dos dados no sistema e concluído o cadastro do processo, sendo exibida uma caixa, em formado de balão, informado que o cadastro foi realizado com sucesso.

6.2.2. Sigilo

O Dicionário Aurélio (2009), define sigilo simplesmente como segredo. A palavra sigilo está relacionada à idéia de segredo, ou ainda, com algo que precisa ser guardado frente a uma verdade.

A Constituição Federal no artigo 5º, inciso XII, preceitua que:

é inviolável o sigilo da correspondência e das comunicações telegráficas, de dados e das comunicações telefônicas, salvo, no último caso, por ordem judicial, nas hipóteses e na forma que a lei estabelecer para fins de investigação criminal ou instrução processual penal.

Embora a lei defina que os atos da administração sejam público assim como os processos, ainda que judiciais e/ou administrativos, a legislação assegura o acesso restrito às informações às partes, tais como: processo de divórcio, pensão alimentícia, sindicância, sendo esse último restrito a alta direção da instituição e da comissão envolvida, ou até mesmo às partes previstas em lei ou regulamento.

Para o ministro do Supremo Tribunal Federal (STF) Celso de Mello: “a vontade do Estado de manter em segredo suas práticas não pode culminar em ofensa aos direitos da pessoa que é submetida a processo”. O sigilo refere-se tão somente ao conteúdo de um processo, quando garantido em lei e deve abranger a informação relativa à existência do processo, de forma que a instituição deve normatizar a classificação desse tipo de processo, como fizera o STF em 2007, através da Resolução 338, regulando sobre a classificação, acesso, manuseio, reprodução, transporte e guarda de documentos e processos de natureza sigilosa.

Dessa forma, um processo definido como sigiloso tem o acesso restrito a um grupo ou usuário, para que se mantenha a segurança das partes envolvidas. Durante o cadastro do processo, o usuário, ao classificá-lo como sigiloso, informará ao sistema quais usuários poderão acessar seus documentos, como pode ser visto na figura abaixo.

Processo Documento Tramitação Confirmação

Detalhes do Processo

Assunto:
Abertura de Sindicância

Palavras chave:
sindicância Ex: palavra1; palavra2

Sigiloso:
☒

Maria José Autorizar

Autorizada(s)	Opção
Procuradoria	

Tipo de Processo:

→ Próximo

Figura 26 – ePad – Cadastro de Processo Sigiloso

Nesse cadastro, que possui as mesmas informações de um processo comum, quando o usuário marcá-lo como sigiloso, será exibida na mesma etapa a tela

solicitando os nomes das pessoas que terão acesso aos documentos desse processo. Da mesma forma, é possível remover os usuários da lista de permissão de processos, clicando-se sobre a opção “Excluir”.

6.2.3. Caixa de Processo

A caixa de processos, conforme definida durante a análise do sistema, é a interface do sistema que será exibida ao usuário onde estarão os processos que estão sob sua guarda e também os processos enviados ao setor pertencente a esse usuário. Semelhantemente como uma caixa de correios em um condomínio, onde as correspondências de um bloco estarão juntas na mesma caixa.

No entanto, ao contrário das correspondências que não podem ser abertas, somente visualizar os dados externos à mesma, os processos que estão em sua caixa poderão ser visualizados, ou seja, estarão disponíveis: o resumo, o histórico da movimentação, assim como os documentos, desde que esses não estejam sob sigilo (vide item 6.2.2).

A figura a seguir ilustra a de processo de um usuário:

Numero	Tipo	Assunto	Opção
9/2011	Pagamento de Diária	Teste de Cadastro Normal	
12/2011	Pagamento de Diária	asdfasdf	
13/2011	Sindicância	Sindicancia	
14/2011	Pagamento de Diária	Processo Sigiloso	

Processo Sigiloso

Figura 27 – ePad – Caixa de Processo

Ao se clicar sobre o botão visualizar, serão exibidos os detalhes do processo selecionado:

The screenshot displays the 'Caixa de Processo' (Process Box) interface in ePad. It is organized into three main sections: 'Processo', 'Documento', and 'Movimentação'.

Processo Section: Contains summary information:

- Numero:** 9/2011
- Palavra-chave:** normal
- Tipo:** Pagamento de Diária
- Assunto:** Teste de Cadastro Normal
- Data Criação:** 06/03/2011 03:00
- Arquivado:** Não

Documento Section: Displays a table of documents. The first document is:

Núm	Tipo	Descrição	Opções
	Declaração	Doc Inicial	[Visualizar] [Baixar] [Baixar Assinado]

Movimentação Section: Displays a table of process movements.

Data	Tipo	Enviado por	Setor	Enviado para	Ver
06/03/2011	TRAMITACAO			Moisés Limeira	[Ver]
06/03/2011	GUARDA			Moisés Limeira	[Ver]
25/03/2011	TRAMITACAO			Setor de Tecnologia da Informação	[Ver]

At the bottom, there are three buttons: 'Incluir Documento', 'Tramitar Processo', and 'Obter Guarda'.

Figura 28 – ePad – Caixa de Processo – Visualizar Processo

A visualização do processo, para uma melhor usabilidade pelo operador do sistema, está agrupada por tipo, em painéis, sendo exibido uma *label* em cada painel, por assunto:

- **Processo:** exibição do resumo dos dados cadastrados no processo.
- **Documento:** exibição dos documentos que compõem o processo, com os seguintes botões:

- Visualizar documento;
- Baixar documento;
- Baixar documento assinado.

- **Movimentação:** exibição do histórico da movimentação do processo.

Os itens a seguir versarão sobre as opções de manipulação de um processo: guarda, tramitação e inclusão de documento.

6.2.3.1. Guarda

Para obtenção da guarda de um processo o usuário deverá clicar sobre o botão Obter Guarda, na parte inferior da visualização, para obtenção da guarda desse processo. Ao clicar sobre esse botão será exibida uma caixa de diálogo solicitando ao usuário a confirmação da operação solicitada.

The screenshot displays the ePad interface with three main sections: 'Processo', 'Documento', and 'Movimentação'. Below these is a confirmation dialog box.

Processo

Numero: 14/2011 Palavra-chave: sigiloso Tipo: Pagamento de Diária
 Assunto: Processo Sigiloso Data Criação: 22/03/2011 03:00 Arquivado: Não

Documento

Núm	Tipo	Descrição	Opções
3	Memorando	Memo	[Icons]

Movimentação

Data	Tipo	Enviado por	Setor	Enviado para	Ver
22/03/2011	TRAMITACAO			Setor de Tecnologia da Informação	[Icon]

Buttons at the bottom: Incluir Documento, Tramitar Processo, Obter Guarda.

Confirma a guarda do processo?

Buttons: Sim, Fechar.

Figura 29 – ePad - Guarda de Processo

Após clicar em Sim, a partir desse instante, nenhum outro usuário poderá obter guarda desse processo, sendo criada automaticamente uma movimentação nesse processo, do tipo GUARDA, para que os demais usuários saibam ao visualizar o histórico, que o processo está sob responsabilidade naquele momento de um usuário.

6.2.3.2. Tramitação

O processo de tramitação, analogicamente, assemelha-se a um funcionário levando um processo a um determinado local para entrega, etapa conhecida como protocolização.

No ePad a forma de envio é eletrônica, para tanto, o remetente, ou seja, o usuário que esteja sob guarda de um processo, deverá selecionar um destinatário no sistema para realizar a tramitação.

A tramitação é realizada da mesma forma como demonstrado no processo de cadastro de processo (item 6.2.1), em que o remetente seleciona um setor ou usuário, que será o destinatário.

6.2.3.3. Arquivamento

Para que um processo seja arquivado, o usuário habilitado para essa operação deverá pertencer ao setor Biblioteca.

A figura abaixo ilustra um usuário com permissões para arquivamento e desarquivamento de um processo:

Processo			
Numero:	7/2011	Palavra-chave:	equipamento
Tipo:	Aquisição de Material		
Assunto:	Aquisição de Equipamentos	Data Criação:	13/02/2011 03:00
Arquivado:	Não		

Documento
+

Movimentação
+

Figura 30 – ePad – Arquivamento de Processo

Para o arquivamento do processo o usuário deverá clicar no botão “Arquivar Processo”.

Confirma o arquivamento do processo? ✕

Figura 31 – ePad – Arquivamento de Processo - Confirmação

Em seguida, será exibida a caixa de diálogo para confirmação da operação, ao clicar em “Sim”, o processo será arquivado na Biblioteca. O processo de desarquivamento assemelha-se ao de arquivamento, onde o botão passará a chamar-se Desarquivar Processo e ao clicar sobre esse botão também será exibida um caixa de confirmação.

6.2.4. Envio de Email

O sistema ePad notificará o destinatário da tramitação sobre a tramitação de um processo, através de uma rotina de envio automático de email utilizando a *API commons-email*, versão 1.2, da *Apache Software Foundation*⁸. Essa *API* foi desenvolvida sob outra *API*, a *JavaMail (J2EE-API)* o que a tornou mais simples e robusta (*CommonsEmail*, Apache). Através da *commons-email*, é possível enviar e-mail simples, com textos puros, em formato HyperText Markup Language (HTML) e também e-mail com anexos.

A seguir demonstraremos a função de envio presente na classe *HistoricoMovimentacao.java*:

```

119 private void adicionarHistoricoMovimentacao(TipoMovimentacao
    tipoMovimentacao) {
120     HistoricoMovimentacaoDAO dao = new HistoricoMovimentacaoDAO();
121     try {
122         Date dataMovimentacao = new Date();
123         getHistoricoMovimentacao().setProcesso(
124             getProcessoController().getProcesso() );
125         getHistoricoMovimentacao().setDataMovimentacao( dataMovimentacao
126         );
127         getHistoricoMovimentacao().setTipoMovimentacao( tipoMovimentacao
128         );
129         dao.saveObject(getHistoricoMovimentacao());
130
131         Email email = new Email();
132         String emailCompleto =
133             getHistoricoMovimentacao().getDestinatario().getEmail() + ""
134             + UtilitarioConfiguracao.DOMINIO;
135         if(email.enviaEmailInformativo( emailCompleto,
136             getHistoricoMovimentacao()) == false){
137             geraMensagemErro("Erro ao enviar email!");
138         }
139
140         geraMensagemSucesso("Tramitado com sucesso!");
141     } catch (Exception ex) {
142         geraMensagemErro("Erro ao tramitar!");
143         ex.printStackTrace();
144     }
145 }

```

Figura 32 – ePad – Envio de Email – Classe HistoricoMovimentacaoController

Ao se realizar uma movimentação de qualquer natureza no sistema, é chamado o método *adicionarHistoricoMovimentacao* (figura 32, linha 119) passando o tipo da movimentação, que poderá ser: tramitação, guarda, arquivamento, desarquivamento. Em seguida, os valores da movimentação são setados e a função de envio de email, da classe *Email*, *email.enviaEmailInformativo* (figura 32, linha 132), dispara um email ao destinatário informando sobre a movimentação do

⁸ Apache Software Foundation – Organização sem fins lucrativos, mantenedora de projetos, incluindo o servidor web Apache HTTP Server

processo. Caso o destinatário seja um Setor, todos os usuários desse setor são notificados.

A figura abaixo demonstra o uso da classe Email.java utilizando a referida API para envio de email no formato *HTML*:

```

18 public boolean enviaEmailInformativo(String emailDest,
                                     HistoricoMovimentacao hm) {
19     try {
20         // cria o anexo.
21         EmailAttachment anexo = new EmailAttachment();
22         anexo.setDescription("Movimentação de Processo");
23         // configura o email
24         HtmlEmail email = new HtmlEmail();
25         // configura a mensagem para o formato HTML
26         email.setHtmlMsg("<html></html>");
27         // conf mensagem alternativa caso o servidor não suporte HTML
28         email.setTextMsg("Seu servidor de e-mail não suporta mensagem
HTML");
29         email.setAuthentication(USUARIO, SENHA);
30         // o servidor SMTP para envio do e-mail
31         email.setHostName(HOST);
32         //destinatário
33         email.addTo(emailDest, hm.getDestinatario().getNome());
34         // remetente
35         email.setFrom("cro-to@cro-to.org.br", "ePAD - CRO-TO");
36         // assunto do e-mail
37         email.setSubject("ePad - Movimentação de Processo");
38         //conteudo do e-mail
39         email.setMsg(CorpoEmail.msgCorpoEmail(hm));
40         email.setSmtpport(25);
41         //email.setSmtpport(465);
42         //email.setSSL(true);
43         //email.setTLS(true);
44         //email.setDebug(true);
45         // envia o email
46         email.send();
47         return true;
48     } catch (EmailException ex) {

```

Figura 33 – ePad – Classe Email

Conforme pode-se verificar através dos comentários no código acima, o uso dessa API é de fácil manutenção e usabilidade para envio de email, e, assim como nas demais APIs com essa função, também é possível configurar a porta, certificado *Secure Sockets Layer* (SSL) e o uso do protocolo *Transport Layer Security* (TLS).

6.3. Documento

“Os documentos eletrônicos nada mais são do que informações manipuladas e armazenadas com o uso do computador sendo, portanto, compostos unicamente por bits.” TADANO (2002, p. 16).

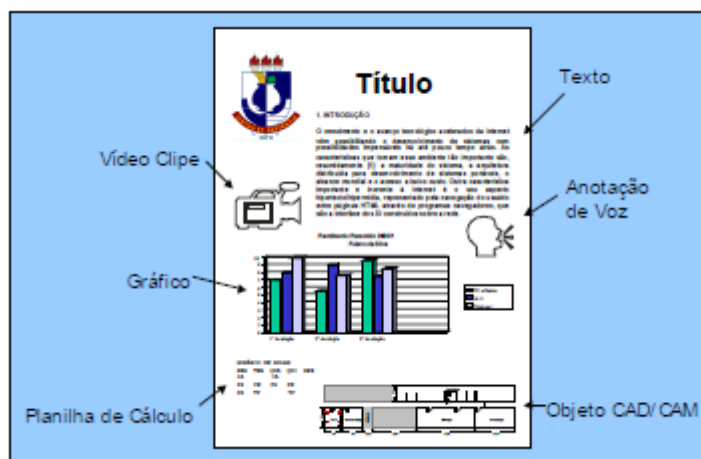


Figura 34 – Representação pictórica de um documento de GED (SILVA, 2001)

6.3.1 Gerenciamento Eletrônico de Documentos - GED

“O Gerenciamento Eletrônico de Documentos é uma solução que faz uso de novas tecnologias para agilizar o fluxo de documentos representados na forma de papel; aumentar a velocidade na troca de informações e tornar os processos de negócios mais rápidos”. (SPRAGUE JR., 1995). Em outras palavras, o GED fornece soluções práticas para o controle documental da organização. A forma mais comum de utilização é através de acesso web, ou por intranet, de forma a acessar versões digitais de documentos relevantes.

O principal destaque de um sistema de GED é a organização e controle de arquivos e documentos em que, através de uma metodologia diferenciada, é feito um trabalho de organização e controle de acesso para os documentos, maximizando assim os processos de busca, tanto física, quanto digital e obtendo eficiência em acesso.

Explicitando de uma forma mais concreta, podemos dizer que o GED é um sistema capaz de armazenar, gerenciar, controlar e exibir documentos em formato digital.

Um sistema de GED é composto pelo uso de equipamentos de captura/edição de imagens, tecnologias de armazenamento, que variam de acordo com a quantidade de documentos a serem convertidos, e de dispositivos de consulta/impressão. Para gerenciar e manter os documentos faz-se necessário o uso de sistemas operacionais e de banco de dados cliente/servidor, softwares de

workflow, softwares de tratamento e recuperação de imagens, além de implementações de segurança no acesso e backups (cópias de segurança).

Os documentos possuem duas maneiras de serem incorporados ao acervo digital de um sistema de GED, através da conversão de documentos do meio analógico para o digital (captura), ou simplesmente associar documentos já criados no meio digital ao sistema. A captura das imagens é realizada pelos scanners de alta precisão, ou seja, com resolução mínima de 400 dpi. (SILVA, 2001)

As organizações possuem maneiras diferenciadas de gerenciar suas informações. Em um fluxo de trabalho em sistemas de imagem eletrônica, os documentos não são simplesmente armazenados, localizados e manipulados, mas usados para conduzir etapas de negócios.

Os documentos eletrônicos podem ser armazenados em diversas mídias existentes no mercado e os meios de armazenamento mais comuns nos sistemas de GED são os ópticos, podendo também ser utilizados os discos magneto-óticos (CD, DVD).

A visualização dos documentos é realizada em terminais de consulta, nos monitores de vídeo, que devem possuir alta resolução para conseguir reproduzir os documentos com fidelidade, assim como as impressoras, essas por sua vez vão desde as impressoras a laser monocromáticas, dependendo do documento, laser coloridas e até plotter. (SILVA, 2001)

Quanto a sua validade jurídica, nesse contexto, o Projeto de Lei 2644/96, em seu artigo 1º, define documento eletrônico como:

Considera-se documento eletrônico, para efeito dessa lei, todo documento, público ou particular, originado por processamento eletrônico de dados e armazenamento em meio magnético, optomagnético, eletrônico ou similar.

Existe outro projeto que conceitua documento eletrônico, é o PL 4906/01 e o faz desta forma:

Art. 2º Para os efeitos desta lei considera-se:

I – documento eletrônico: a informação gerada, enviada, recebida, armazenada ou comunicada por meios eletrônicos, ópticos, optoeletrônicos ou similares.

Nesse contexto Tadano (2002) afirma que pelo fato da sociedade se basear em parâmetros para nortear suas relações com outrem, como por exemplo, o

amparo legal, as mudanças sociais, novas tecnologias e conseqüentemente novas relações ou fatos jurídicos, devem ser seguidas por regulamentação das leis.

Hoje no Brasil, inúmeras empresas, inclusive o setor público tem adotado sistemas de GED, desde os tribunais de primeira até a instância máxima do judiciário brasileiro, o STF.

6.3.1.1. Fluxo de Trabalho

Os fluxos de trabalho, também conhecidos como *workflow*, segundo a *Workflow Management Coalition*, são: "A automação de um processo de negócios, por inteiro ou em parte, durante o qual documentos, informações e tarefas são passadas de um participante para outro por ação respeitando um conjunto de regras procedurais".

Para construir workflow é preciso analisar o processo do trabalho, assim, a implantação contribui não apenas na organização e na geração dos documentos, mas também na otimização dos processos. Os processos devidamente estruturados, aumentam a confiança na realização das tarefas e eliminam a incertezas como, por exemplo, de quais pessoas devem receber um comunicado ou documento. Ao melhorar um workflow, os sistemas GED: reduzem o espaço de armazenamento de informações, reduzem o tempo de processamento, melhoram a eficácia em processos, agilizam o atendimento ao cliente, garantem o controle de documentos e dão suporte à tomada de decisões. (SCHEER, s.d.)

Uma organização pode ter seu fluxo de trabalho definido e não possuir uma ferramenta de GED, assim como possuir um sistema de gestão de documentos e não ter um *workflow*, pois ambos podem existir independentemente um do outro. A instituição objeto de estudo desse trabalho, o TRE-TO, não possui um fluxo de trabalho definido e o foco do trabalho não possibilitou a implementação de um *workflow* no sistema desenvolvido.

7. ASSINATURA DIGITAL

Hoje, para que um documento emitido por alguma entidade tenha valor no mundo jurídico, é necessário que se possa identificar sua autoria. Dessa forma, pode-se atribuir a responsabilidade pela composição, e, então, verificar a legitimidade da competência do autor, pois, por exemplo, não se espera de alguém que não seja um magistrado proferir uma sentença.

Em termos de práticas cartorárias, caso não houvesse a possibilidade de se comprovar a autoria e a integridade de documentos que tenham sido devidamente assinados por seus autores, certamente não encontraríamos muita utilidade para tais documentos. Seriam como qualquer outra informação em que a legitimidade é definida a critério do leitor, sem que haja qualquer instrumento ou entidade que possa encarregar-se de estabelecer sua autoria e integridade e, portanto, sem aplicabilidade na tomada de decisões críticas.

Da mesma forma, os documentos gerados e mantidos eletronicamente que apresentam valor jurídico devem ser assinados de forma a possibilitar a comprovação de sua autoria. Além da assinatura, deve-se também estabelecer mecanismos que impeçam a modificação indesejada de tais informações ou que ao menos seja possível identificá-las por meios e técnicas indubitáveis.

É nesse contexto, onde há necessidade de garantir a segurança da informação eletrônica, que surge a assinatura digital. Por meio dela, podemos gerar dados e informações e aplicar-lhes técnicas que produzam uma assinatura, algo semelhante ao que conhecemos no mundo físico, algo susceptível de verificação e capaz de determinar se aquela informação foi realmente assinada pelo indivíduo tido como autor ou partícipe.

7.1. Conceitos e definições de assinatura digital

A assinatura digital é um atributo que compõe uma mensagem, representando a assinatura desta. Sua finalidade é similar ao de uma assinatura no mundo real – é utilizada como dispositivo para comprovação de autenticidade de uma informação, documento ou correspondência. É formada por bits derivados da mensagem original

após a aplicação de algoritmos próprios para assinatura. (PETERS; BATHIA; KALIDINDI; MUKHERJEE, 2008)

Para se obter uma assinatura digital, executa-se um algoritmo de geração de *hash*⁹ – resumo – tendo como entrada a mensagem que se deseja assinar. Em seguida, cifra-se o *hash* por meio de um algoritmo de criptografia por chave pública ou assimétrica¹⁰ ou aplica-se um algoritmo próprio para assinatura digital, também baseado em chaves assimétricas, com o uso da chave privada do emissor do documento. Ao final, teremos uma sequência de *bits* de tamanho pré-definido e invariável que representará a assinatura do documento.

A fim de que uma assinatura digital seja usual, agregue valor e possa ser considerada segura, é necessário que se atenda a alguns requisitos. São eles, a saber: deve ser de fácil geração, sem que requeira muito recurso computacional; deve prover autenticidade; deve prover integridade; e deve ser irrevogável a ponto de garantir o não-repúdio¹¹.

Uma vez assinado o documento, para que possamos verificar se a assinatura é válida, ou seja, se ela realmente derivou do documento em questão e se foi gerada com o uso da chave privada do usuário emissor, é preciso fazer o processo inverso ao da assinatura. Primeiramente, deciframos o resumo criptografado resultante do processo de assinatura. Assim como no processo de assinatura, para decifrar o resumo é preciso executar um algoritmo de criptografia por chave pública, só que utilizando a chave pública do usuário signatário. Em seguida, gera-se um novo resumo a partir do documento original e, então, compara-se o novo resumo com o resumo obtido após o processo de decriptografia¹². Caso os resumos sejam iguais, a assinatura é válida, caso contrário, a assinatura é inválida e provavelmente houve uma mudança no documento original ou no resumo criptografado, ou mesmo a chave pública utilizada na decriptografia não é correspondente à chave privada utilizada no momento da assinatura.

A figura abaixo demonstra a assinatura de um documento e, posteriormente, sua verificação:

⁹ Hashing representa a criação de uma versão miniatura (minuta) de uma mensagem de forma a ser usada no lugar da mensagem original, para alguns aspectos de segurança.” (FOROUZAN, 2006)

¹⁰ “Chaves assimétricas são pares (...), comumente denominadas *chave pública* e *chave privada*, que são matematicamente relacionadas uma a outra.” (PETERS; BATHIA; KALIDINDI; MUKHERJEE, 2008).

¹¹ “Não-Repúdio (Não-recusa): garantia que o emissor ou pessoa que executou determinada transação de forma eletrônica, não poderá, posteriormente negar sua autoria.” (URFJ, 2007)

¹² Transformação do texto cifrado em texto legível.

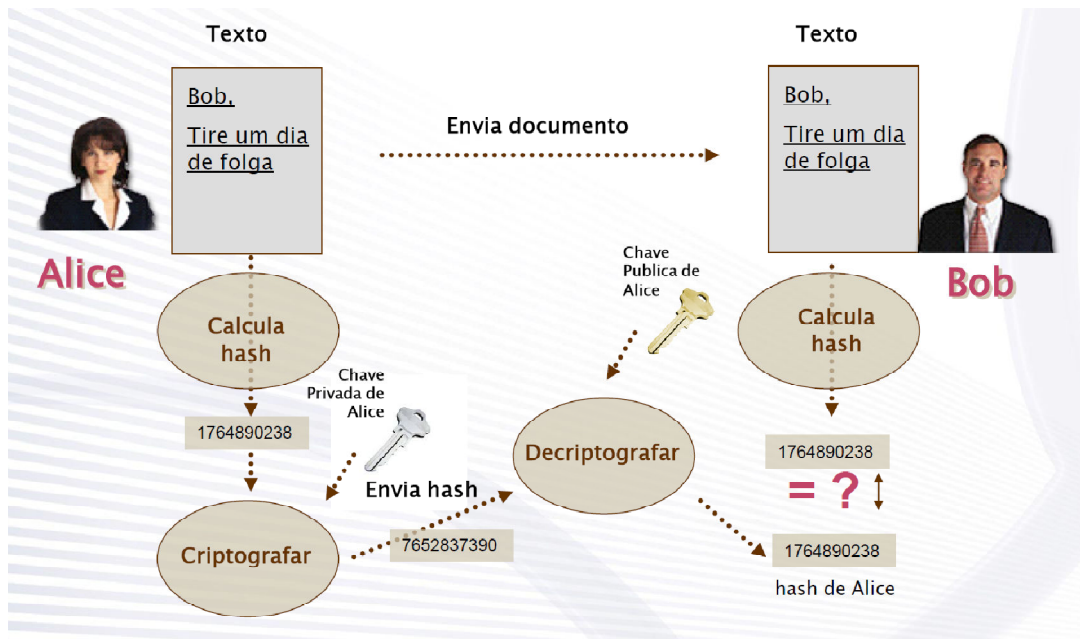


Figura 35 – Processo de assinatura digital sem certificação (CertSign)

7.2. Resumo da Mensagem (Message Digest) - Hash

No meio computacional, para assegurar que uma mensagem foi ou não alterada durante uma transferência, é utilizado um método matemático conhecido por *hash*. É por meio dele que podemos verificar se, por exemplo, um determinado pacote na rede não foi corrompido durante o seu envio ou se um determinado arquivo que foi baixado da Internet é igual ao disponibilizado no sítio de publicação.

O *Hash* é uma função matemática unidirecional (pode ser feita num sentido e não no outro – como a relação entre as chaves em um sistema de criptografia assimétrica) para escrever uma quantidade definida de bytes em relação a uma mensagem de qualquer tamanho. Em suma: não se pode obter o arquivo original a partir do hash (é impossível, ou, no mínimo improvável). (CARVALHO J, 2009, p. 493)

Cabe ressaltar que existem diferenças entre algoritmos de *checksum* e algoritmos de *hash* criptográfico. O primeiro tem como objetivo verificar se houve alterações não intencionais na mensagem, o que comprometeria o correto funcionamento do sistema/aplicativo que a utiliza ou invalidaria a informação recebida. Em geral, esse tipo de algoritmo apresenta um baixo custo de processamento e tem como saída um conjunto de bits menor que o de uma função de *hash* criptográfica. Além do mais, não apresenta preocupação concernente a problemas de colisão. Por tudo isso, é muito comum sua utilização por protocolos de rede que realizam validação de integridade em seus Protocol Data Unit (PDUs). Um

exemplo desse tipo de *hash* o é CRC-32¹³, utilizado na camada de enlace da pilha TCP/IP, que tem uma saída de 32 bits.

Contudo, quando nos referimos a *hash* criptográfico, os objetivos mudam. Ele deve possibilitar a detecção de alguma mudança intencional ou não intencional em uma mensagem, de modo que seja sensível a qualquer alteração, por menor que seja. Deve ser resistente à colisão, tornando improvável encontrar duas mensagens distintas que apresentem o mesmo *hash*. Outra característica importante é que deva ser impraticável a descoberta da mensagem original a partir do código *hash*, o que o torna bastante útil em sistemas de segurança onde são armazenadas senhas de usuários. Consequentemente, a função de *hash* criptográfico apresenta um maior custo de processamento para realizar suas operações quando comparados a uma função *checksum*. Atualmente, encontramos funções que geram saídas com tamanho de 128 a 512 bits, como o MD5¹⁴ e o SHA-512¹⁵ respectivamente.

Para tornar mais claro o funcionamento de uma função de *hash* criptográfico, mostraremos abaixo alguns exemplos. Quando aplicamos a função *hash* SHA-1 sobre a sequência de caracteres “**abcdef**”, obtemos a seguinte saída:

```
1f8ac10f23c5b5bc1167bda84b833e5c057a77d2
```

Se alterarmos a segunda letra da sequência, substituindo-a pela letra “a”, resultando em: “**aacdef**”; e então aplicarmos novamente a função SHA-1, obteremos a seguinte saída:

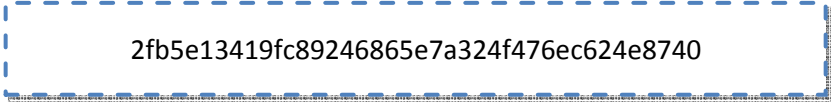
```
bb2e99824273e198ae2338f73d07641916a4327f
```

Novamente, se alterarmos o texto original por meio do acréscimo da letra “g” ao final da sequência de caracteres, resultando em: “**abcdefg**”; e, então, aplicarmos a função SHA-1, obteremos a seguinte saída:

¹³ “Cyclic Redundancy Check (CRC) – Teste de Redundância Cíclica. Algoritmo de detecção e correção de erros em teleprocessamento.” (SAWAYA, Marcia Regina. Dicionário de Informática e Internet: inglês/português, 2009, p. 110)

¹⁴ “MD5 (“Message Digest algorithm number 5”) is a cryptographically strong, 128-bit hash hash algorithm designed by Ron Rivest in 1991, one of a series (...).” (Daniel J. Barrett; Richard E. Silverman; Robert G. Byrnes. SSH the secure shell: the definitive guide, 2005, p. 90)

¹⁵ “SHA-1 (Secure Hash Algorithm) was designed by the NSA and the NIST for use with the U.S. government Digital Signature Standard. (...) It produces 160-bit hashes.” Id. *ibid*.



2fb5e13419fc89246865e7a324f476ec624e8740

Como podemos observar, por menor que seja a alteração feita na mensagem, os resultados obtidos serão completamente diferentes.

Uma função hash seria ideal se não gerasse colisão para qualquer que fosse sua entrada. No entanto, dada as características, isso não é possível, pois se há um conjunto de entrada de tamanho infinito e temos um número finito de possibilidades na saída, então, em algum momento haverá mais de uma entrada para uma única saída. Isso é facilmente confirmado ao estudarmos o Princípio de Pombal¹⁶. Por isso, a escolha do melhor algoritmo de *hash* recai sobre aquele que tende a gerar o menor número de colisões.

Atualmente, existem diversas funções *hash* sendo utilizadas no meio eletrônico. Segue, abaixo, lista com as que apresentam maior destaque:

- MD5 (Message-Digest algorithm 5) – 128 bits;
- SHA-1 (Secure Hash Algorithm) – 160 bits;
- SHA-256 – 256 bits;
- SHA-512 – 512 bits;

7.3. Criptografia

Desde os primórdios da civilização, a necessidade de proteger as mensagens que eram transmitidas entre as partes sempre foi relevante. Segundo a história, os Espartanos, 400 a.C., já teriam desenvolvido uma técnica para ocultar mensagens. Conhecida como Bastão de Licurgo, essa técnica consiste em enrolar uma tira de couro em um bastão e, em seguida, escreve-se a mensagem na tira de couro. Ao final, desenrola-se a tira e, então, a mensagem está cifrada. Para que seja possível lê-la novamente, é preciso voltar a enrolar a tira em um bastão de mesmo diâmetro. (QUARESMA; LOPES, 2007)

¹⁶ "The pigeonhole principle states that if there are more pigeons than pigeonholes, then there must be at least one pigeonhole with at least two pigeons roosting." (BISWAL, Purna Chandra. Discrete Mathematics and Graph Theory, 2nd ed; 2008, p. 46)

A força da criptografia¹⁹ depende de três critérios básicos: do tamanho e aleatoriedade da chave; da qualidade do algoritmo; e da correta implementação. Por exemplo, caso possuíssemos um texto cifrado²⁰ por um algoritmo de ótima qualidade e que fosse implementado corretamente, mas tivéssemos utilizado no processo de criptografia uma chave pequena, com tamanho de 4 *bytes*, obteríamos o texto original com apenas 2^n , onde n é o tamanho da chave em bits, tentativas, ou seja, bastaria 2^{32} combinações para que encontrássemos a chave utilizada no processo de criptografia.

Da mesma forma, quando não há a correta implementação do algoritmo ou o mesmo não possui a qualidade necessária, a força da criptografia fica sujeita à vulnerabilidade encontrada. Um exemplo disso é o algoritmo de chave simétrica Double-DES que é comprovadamente vulnerável ao ataque *meet-in-the-middle*²¹.

7.3.1. Criptografia de Chave Simétrica

Quando falamos em criptografia, logo nos vem a mente a ideia de existência de uma mensagem que precisa ser mantida em segredo até que encontre o seu destinatário, de um algoritmo a ser utilizado para criptografar essa mensagem e de uma chave que só pode ser conhecida pelo emissor e destinatário da mensagem. Nesse contexto, a chave passa ser um elemento essencial na segurança do processo de confidencialidade da informação.

Se utilizarmos uma única chave para criptografar e decriptografar a mensagem, então estamos nos referindo a criptografia de chave simétrica. Segundo Alfredo Santos (2008, p. 57):

Os algoritmos de chave-simétrica (também chamados de Sistemas de Chaves Simétricas, criptografia de chave única, ou criptográfica de chave secreta) são uma classe de algoritmos para a criptografia que usam chaves criptográficas relacionadas para a decifração e a encriptação. A chave de encriptação é relacionada insignificativamente à chave de decifração, que podem ser idênticas ou ter uma simples transformação entre as duas chaves. As chaves, na prática, representam um segredo compartilhado entre dois ou mais partidos que podem ser usados para manter uma ligação confidencial da informação. Há uma única chave, usada por ambos os interlocutores, e na premissa de que esta é conhecida apenas por eles.

¹⁹ Dificuldade de quebra do código.” Ibid.

²⁰ Criptografado.

²¹ Meet-in-the-middle é um tipo de ataque a sistemas criptográficos capaz de reduzir a força da criptografia em algoritmos que são aplicados duas vezes na tentativa de aumentar a segurança.

Apesar do grande uso, existem algumas considerações que devem ser feitas acerca de algoritmos de chave simétrica. Primeiro: o sistema é tão seguro quanto sua chave e a forma como foi compartilhada; segundo: dada a necessidade de uma chave secreta compartilhada, com uma cópia em cada ponta, as chaves ficam sujeitas à exposição, o que facilita a descoberta indesejada por um adversário criptográfico, fazendo necessário a troca contínua da chave; e por último, para garantir a troca de informações segura em uma população de n pessoas, seria necessário um total $n(n - 1)/2$ chaves. Todas essas características tornam o uso de algoritmos de chave simétrica muito complexo e oneroso quando utilizados com frequência e entre muitos indivíduos. (SANTOS, 2008)

Abaixo, a figura 37 ilustra o processo de assinatura utilizando algoritmo de chave simétrica:

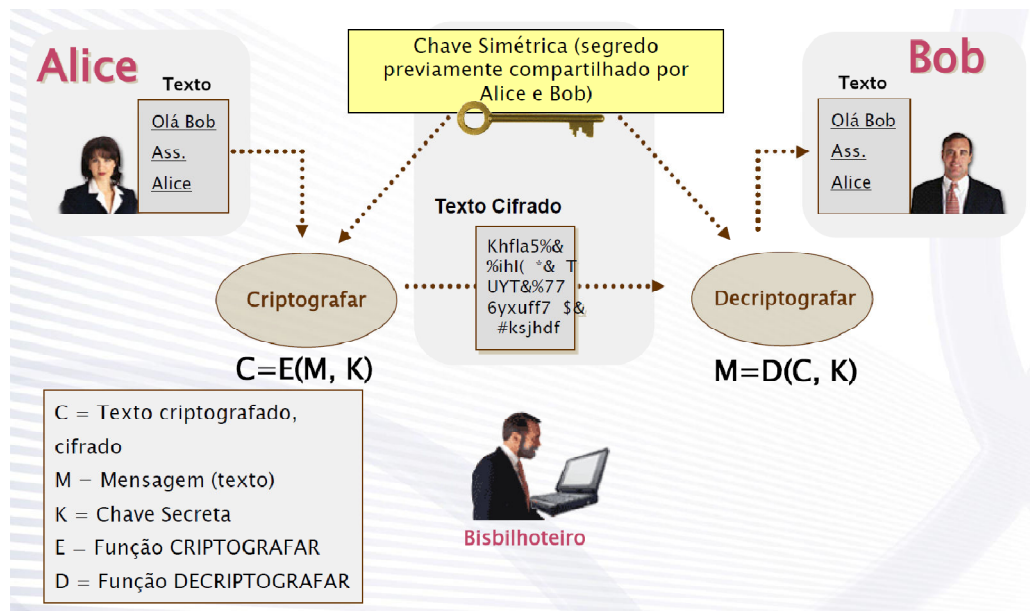


Figura 37 – Processo de criptografia com chave simétrica (CertSign)

7.3.2. Criptografia de Chave Pública e Algoritmos de Assinatura

Concebida por Whitfield Diffie, a criptografia de chave pública ou criptografia assimétrica é um método que utiliza um par de chaves: uma chave pública e uma chave privada; diferente da criptografia simétrica que utiliza uma única chave. Neste modelo, quando uma mensagem é cifrada com a chave pública, ela só poderá ser decifrada com chave privada; da mesma forma, quando a mensagem é cifrada com

a chave privada, ela só poderá ser decifrada com a chave pública. Essa característica permite que o algoritmo seja utilizado tanto para confidencialidade quanto para autenticidade. (SANTOS, 2008)

No sistema de criptografia assimétrica, a chave pública é distribuída entre todos os destinatários, enquanto mantém-se o sigilo da chave privada, conhecida apenas pelo proprietário. Quando o objetivo é prover confidencialidade, cifra-se a mensagem com a chave pública do destinatário, pois somente o destinatário poderá decifrá-la utilizando-se de sua chave privada. Semelhantemente, quando se deseja prover autenticidade, cifra-se a mensagem com a chave privada do emissor. Se o destinatário, de posse da chave pública do emissor, decifrar a mensagem, então estará confirmada a autenticidade da informação, pois somente quem possuísse a chave privada poderia cifrar um texto que fosse decifrável com aquela chave pública.

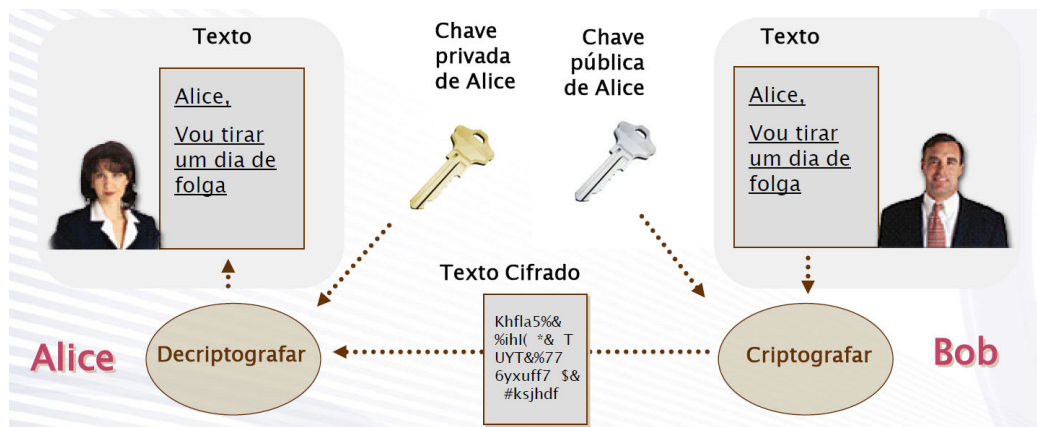


Figura 38 – Criptografia com uso de chaves assimétricas (CertSign)

Uma das vantagens da criptografia de chave pública em relação à criptografia simétrica é que, ao passo que o número de indivíduos envolvidos aumenta, a gerência das chaves torna-se proporcionalmente mais simples no sistema assimétrico. Por exemplo, no sistema assimétrico, para garantir a troca de informações segura em uma população de n indivíduos, seria necessário um total de $(n * 2)$, enquanto que no sistema simétrico seria necessário um total de $n(n - 1)/2$ chaves.

É necessário destacar que existem alguns fatores que pesam contra os algoritmos de chave assimétrica. Um deles é fato de serem muito mais lentos que um algoritmo de chave simétrica de qualidade. Outra questão é que o sistema de criptografia simétrica é considerado mais forte que o de chave pública, por exemplo: uma chave de 1024 bits no método assimétrico provê um nível de segurança

equivalente a uma chave de 80 bits no método simétrico; já uma chave de 2048 *bits* no método assimétrico provê um nível de segurança correspondente a uma chave de 112 bits no método simétrico. (RSA, 2003).

Entre os algoritmos de criptografia de chave pública mais comuns, podemos destacar o RSA²² e o *Elliptic Curve Cryptography*²³ (ECC). O RSA, quando combinado com algoritmos de *hash*, passa a ser utilizado como algoritmo de assinatura digital. Por exemplo, o algoritmo **SHA1withRSA** é um dos algoritmos de assinatura mais utilizados atualmente e é composto pelo algoritmo de *hash* SHA-1 mais o algoritmo de criptografia assimétrica RSA.

Desenvolvido pela Agência Nacional de Segurança dos Estados Unidos e adotado como o padrão federal de processamento de informações (FIPS), o *Digital Signature Algorithm*²⁴ (DSA) tem como único objetivo a geração e verificação de assinatura digital. Em uma de suas etapas de execução, ele realiza a operação de *hashing* sobre a mensagem, o que, entre outras coisas, impede sua utilização para fins de criptografia. Algumas pessoas costumam pensar que o algoritmo de assinatura DSA seja útil também para criptografia, porém, ele serve apenas para assinatura, apesar de basear-se em um sistema de chaves assimétricas.

7.4. Certificação Digital e Infraestrutura de Chave Pública

A criptografia assimétrica proporcionou um grande avanço para a segurança da informação. Além de permitir a confidencialidade, por meio dela também é possível realizar a troca segura de chaves, bem como autenticar o remetente da mensagem com o uso da assinatura digital. Porém, existem algumas características que a torna vulnerável quando empregada isoladamente.

Quando ocorre a troca de chaves públicas em um sistema de criptografia assimétrica que atua isoladamente, a chave pública não contém qualquer informação acerca do remetente. Para que o sistema funcionasse em larga escala, seria necessário que cada usuário envolvido dispusesse de uma base de dados associando cada chave pública ao seu proprietário. Isso tornaria o uso do sistema

²² Criado por um grupo de três pessoas, seu nome é composto pelas iniciais de seus nomes, a saber: Ronald Rivest, Adi Shamir e Leonard Adleman.

²³ Criptografia de Curvas Elípticas.

²⁴ Algoritmo de Assinatura Digital.

de chave pública mais oneroso a cada novo indivíduo com quem o usuário se relacionasse. Outra questão é o fato de o sistema de criptografia de chave pública não garantir se o remetente é quem realmente diz ser. Esse tipo de vulnerabilidade pode ser facilmente explorada por meio de um ataque de personificação ou intermediação fraudulenta. (CERTSIGN, s.d).

O ataque de personificação ocorre quando alguém tenta se passar por uma pessoa que não ela mesma, na tentativa de obter privilégios restritos a outrem. O ataque consiste em fazer com que a vítima acredite que a pessoa ou o sistema com quem ela está se correspondendo seja uma alguém de sua confiança, quando na verdade é o atacante se passando por outrem.

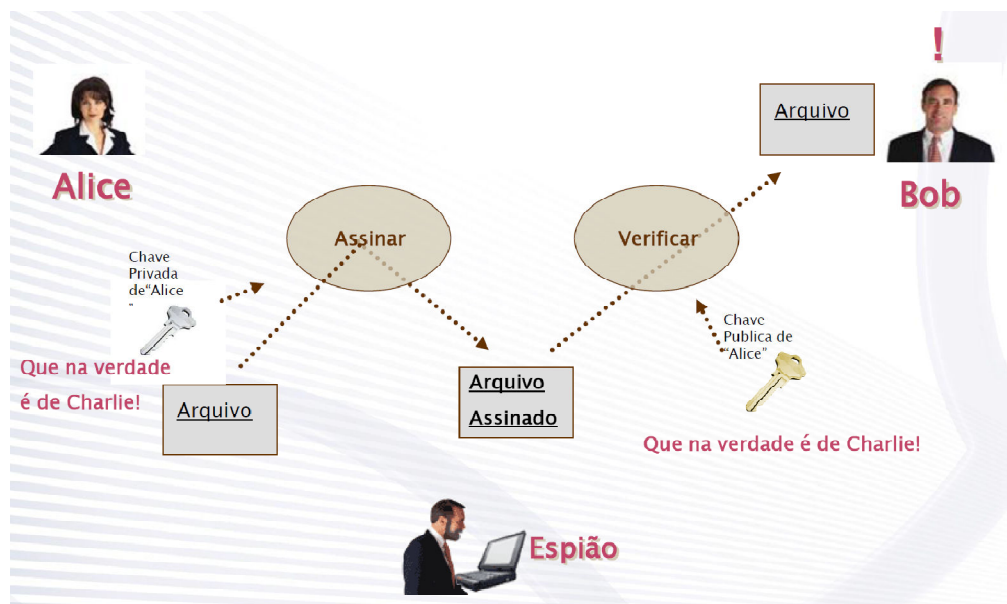


Figura 39 – Ataque de personificação na assinatura (CertSign)

Considerado um ataque de intermediação fraudulenta, o *man-in-the-middle*²⁵ (MITM) atua contra dois indivíduos que estão se correspondendo. O atacante passa a representar as duas partes ao mesmo tempo, fazendo com que uma pense que está se comunicando com a outra, quando na verdade, ambas estão comunicando-se com o atacante.

²⁵ Homem no meio.

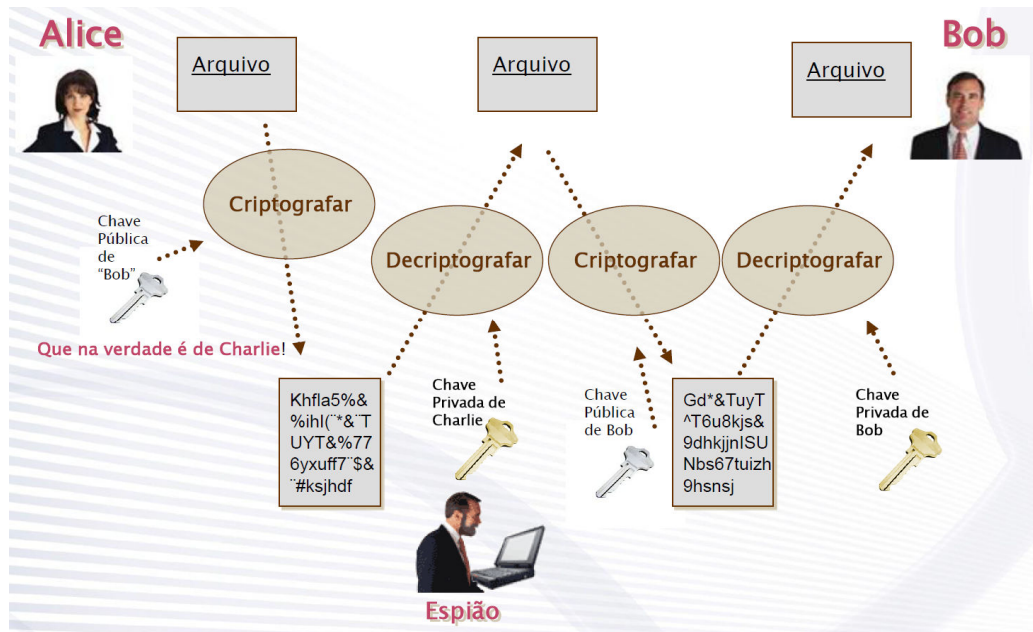


Figura 40 – Man-in-the-middle no sigilo (CertSign)

7.4.1. Certificado Digital

Com a necessidade de se estabelecer previamente o relacionamento entre as chaves públicas e os respectivos titulares, em 1976, Diffie-Hellman sugeriu a ideia de relacionar nomes com chaves públicas a partir do uso de um repositório público. (CertSign).

Já em 1978, Loren Kohnfelder, em sua monografia de graduação pelo *Massachusetts Institute of Technology* (MIT), criou o conceito de 'certificado'. A proposta consistia em incluir tanto a chave pública quanto o nome do usuário em um único arquivo e, em seguida, assiná-lo. Desta forma, a necessidade de se vincular a chave pública a seu titular seria atendida. (CERTSIGN, s.d.).

Atualmente, além da chave pública e do nome do titular, os certificados digitais são compostos por mais informações como e-mail; data de início e fim de validade do certificado; país, estado, organização e departamento do titular; etc.. O número de informações que comporão um certificado dependerá da versão em que for emitido, sendo a versão 3 a mais atual.

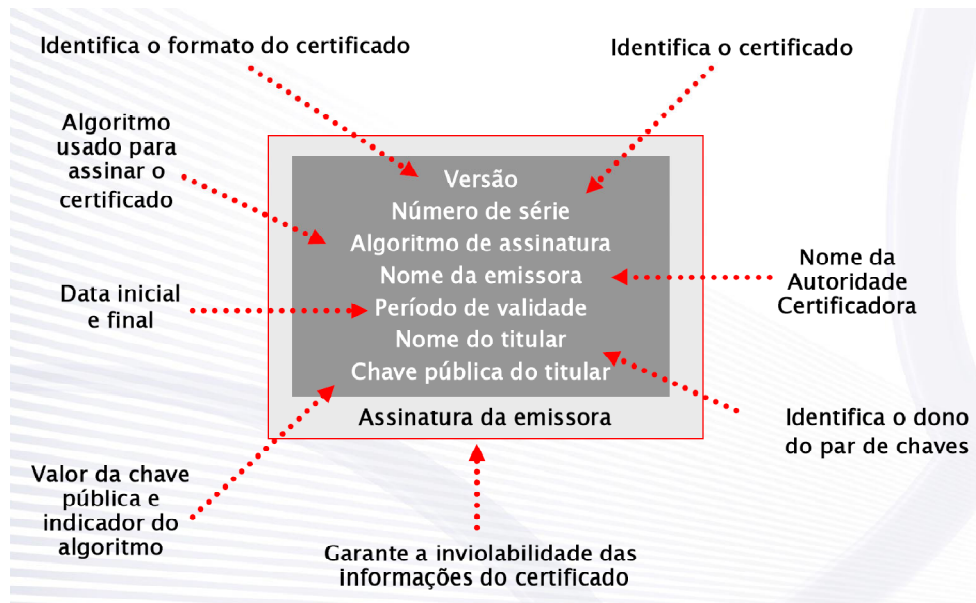


Figura 41 – Conteúdo Básico do Certificado (CertSign)

Ainda acerca de Certificado Digital, o Serasa (s.d.) afirma que:

O certificado digital é um documento eletrônico que possibilita comprovar a identidade de uma pessoa, uma empresa ou um site, para assegurar as transações online e a troca eletrônica de documentos, mensagens e dados, com presunção de validade jurídica.

Os certificados digitais apresentam finalidades e propósitos distintos, sendo, portanto, utilizados em diversos tipos de aplicações e sistemas computacionais. Entre os tipos de certificados digitais, podemos encontrar certificados para assinatura de mensagens, assinatura de código, assinatura de certificados, sigilo, etc..

		Titular	
Número de série	Issued To	Common Name (CN)	Joao da Silva
		Organization (O)	CertiSign Certificadora Digital S.A.
		Organizational Unit (OU)	Autenticado por CertiSign RH
		Serial Number	39:12:84:21:BF:64:AD:36:B7:7B:E1:85:CA:E9:14:4C
Emissor	Issued By	Common Name (CN)	CertiSign Autoridade Certificadora Classe 2
		Organization (O)	CertiSign Certificadora Digital S.A.
		Organizational Unit (OU)	VeriSign Trust Network
	Validity	Issued On	7/6/2005
Validade		Expires On	8/6/2006
	Fingerprints	SHA1 Fingerprint	58:80:9C:AD:44:EB:7D:41:CE:B6:96:72:17:DD:E0:B4:4B:8A:78:6C
Hash		MD5 Fingerprint	D7:B3:A7:8D:CC:BA:25:38:D7:7A:C3:6E:4F:39:B4:D0

Figura 42 – Modelo de Certificado de Usuário Final (CertSign)

7.4.2. Autoridade Certificadora

Como mostrado anteriormente, o problema da comprovação da titularidade da chave pública foi resolvido com a adoção de certificado digital, uma vez que o próprio certificado já traz as informações relativas ao seu titular juntamente com a chave pública. Porém, ainda é necessário garantir a legitimidade dessas informações. Surge, então, a figura da Autoridade Certificadora.

Ela é a entidade responsável pelo registro e assinatura das chaves públicas, devendo as partes envolvidas reconhecerem-na como autoridade confiável para emissão de certificados digitais. Como a assinatura é realizada com o uso da chave privada da AC, qualquer um que dispuser de sua chave pública, que está contida dentro do certificado da AC, poderá validar a assinatura e consequentemente as informações contidas no certificado do titular.

A AC detém um conjunto de atribuições que devem ser desempenhadas dentro de uma Infraestrutura de Chave Pública²⁶ (PKI). Essas atribuições compreendem a emissão, renovação, suspensão e revogação de certificados, bem como a expedição de LCR.

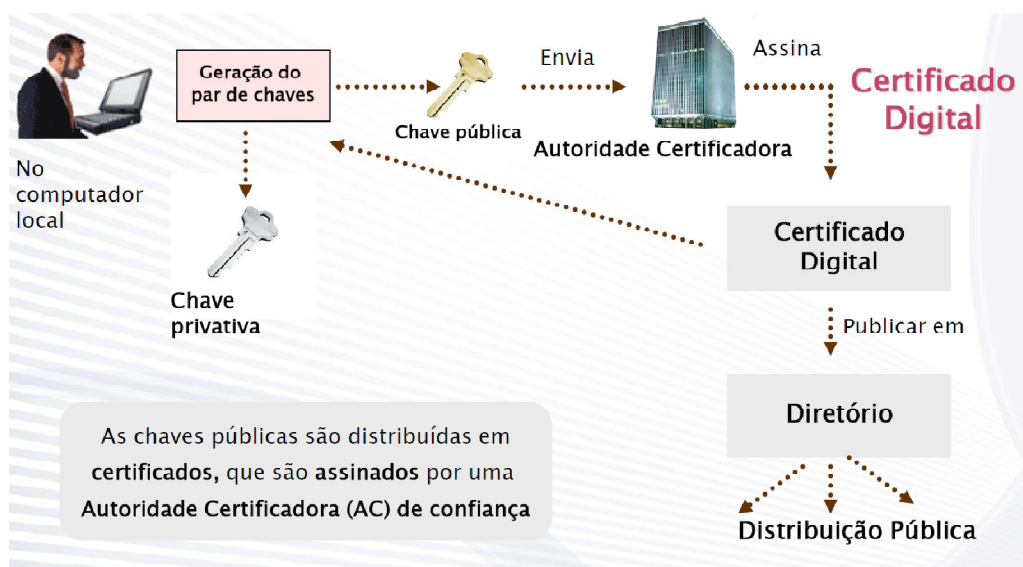


Figura 43 – Distribuição de Chaves Públicas (CertSign)

²⁶ Public Key Infrastructure.

No Brasil, a Medida Provisória nº 2.200-2, de 24 de agosto de 2001, institui a Infraestrutura de Chave Pública (ICP) – ICP-Brasil – como entidade responsável por garantir a autenticidade, a integridade e a validade jurídica dos documentos emitidos de forma eletrônica. A Medida Provisória cria, também, o Comitê Gestor da ICP-Brasil e a AC Raiz Brasileira, além de definir as demais entidades que compõem a estrutura. Segue, abaixo, parte do texto da Medida Provisória que institui a ICP-Brasil:

Art. 1º Fica instituída a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, para garantir a autenticidade, a integridade e a validade jurídica de documentos em forma eletrônica, das aplicações de suporte e das aplicações habilitadas que utilizem certificados digitais, bem como a realização de transações eletrônicas seguras.

Art. 2º A ICP-Brasil, cuja organização será definida em regulamento, será composta por uma autoridade gestora de políticas e pela cadeia de autoridades certificadoras composta pela Autoridade Certificadora Raiz - AC Raiz, pelas Autoridades Certificadoras - AC e pelas Autoridades de Registro - AR.

(...)

Art. 13. O ITI é a Autoridade Certificadora Raiz da Infra-Estrutura de Chaves Públicas Brasileira.

7.4.3. Hierarquia de Certificados Digitais

Assim como em um sistema de diretório ou em uma organização tradicional, um sistema de ICP também é organizado de forma hierárquica. Deste modo, as responsabilidades e competências são descentralizadas e distribuídas de forma a facilitar a manutenção dos certificados digitais.

Quando tratamos de hierarquia, sempre haverá três coisas a serem consideradas: um nó raiz, nós internos e folhas. No caso da hierarquia em uma ICP, a entidade que se encontra no topo é chamada de AC Raiz. É a partir dela que são emitidos os certificados das demais Autoridades Certificadoras Intermediárias e ela que representa o nó raiz de uma ICP. As ACs Intermediárias representam os nós internos, pois estão ligadas diretamente a uma AC Raiz ou a outras ACs Intermediárias. E por último as entidades finais, associadas aos certificados que não são próprios para assinatura de outros certificados, como certificados para assinatura de código ou objeto, certificados de sigilo, dentre outros; esses representam as folhas.

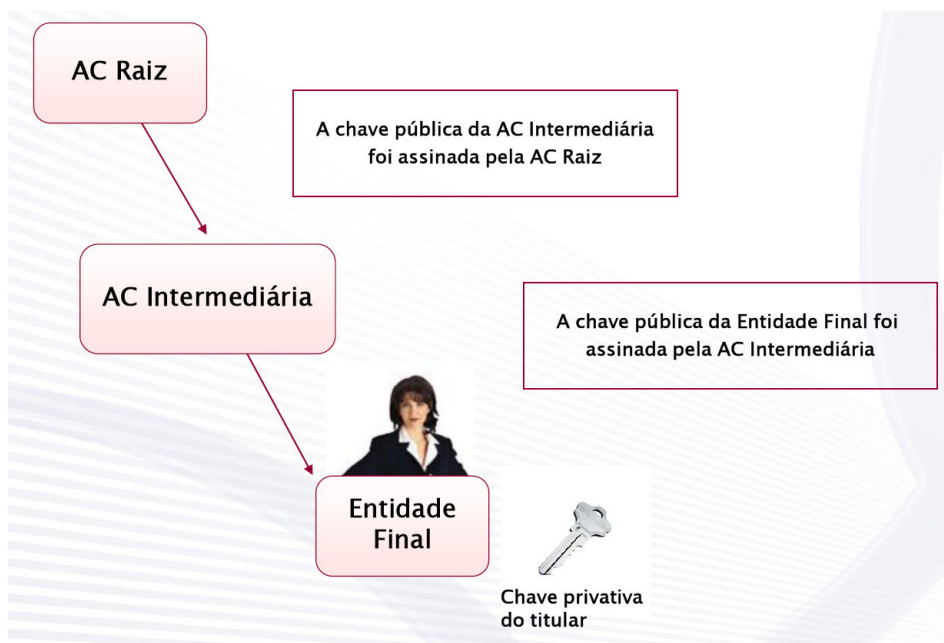


Figura 44 – Cadeia de Certificação (CertSign)

Um certificado é gerado a partir de uma Requisição de Assinatura de Certificado (CSR) feita por alguma entidade ou usuário a uma AC por intermédio de uma Autoridade de Registro (AR). Essa última, por sua vez, é a responsável por realizar a validação das informações contidas na CSR e enviá-las a AC. Ao receber a solicitação encaminhada pela AR, a AC emitirá o certificado mediante a assinatura da chave pública e das informações contidas no próprio pedido. Quando a CSR parte da própria AC Raiz, o certificado emitido será auto-assinado, de modo que o emissor do certificado será também o seu titular. Portanto, em uma ICP, o primeiro certificado a ser emitido é o certificado auto-assinado da AC Raiz. (MARTINS, 2001)

Na ICP-Brasil, a AC Raiz Brasileira é a AC Raiz e tem o seu certificado emitido pelo Instituto Nacional de Tecnologia da Informação. Segundo a Medida Provisória nº 2.200-2, de 24 de agosto de 2001, a AC Raiz Brasileira fica impedida de emitir certificados digitais para usuário final, conforme descreve o Parágrafo Único do Art. 5º da referida MP:

Art. 5º (...)

Parágrafo único. É vedado à AC Raiz emitir certificados para o usuário final.

7.4.4. Geração de certificado para Autoridade Certificadora Raiz e Intermediária

O sistema proposto neste projeto contempla a autenticação de documentos eletrônicos mediante assinatura digital. Para que pudéssemos testar essa funcionalidade, foi necessário criar uma pequena ICP privada em nosso ambiente. Fica óbvio que essa ICP não atende a todos os requisitos impostos a uma ICP comercial, pois tem único intuito de possibilitar os testes necessários à validação dos requisitos do sistema.

Para a criação da ICP privada foi utilizada a biblioteca *OpenSSL* a partir do Sistema Operacional *Linux Ubuntu*, sendo adotada a versão 0.9.8g. Na geração dos certificados digitais, foi adotada a versão 3 do formato de certificado padrão ITU-T X.509. Foram mantidas as configurações padrão e todos os comandos utilizados foram executados a partir do *bash*²⁷.

Inicialmente, é criada a AC Raiz que atuará no mais alto nível de nossa ICP. Para isso, primeiro é gerada uma chave privada do tipo RSA com tamanho de 1024 bits que será utilizada pela AC Raiz. Segue, abaixo, o comando utilizado para geração da chave:

```
root@ePad:~# openssl genrsa -out ca_raiz-key.pem 1024
```

Em seguida, é gerada uma CSR com base nos dados fornecidos no momento da geração e na chave pública do titular da chave privada. A solicitação deve ser assinada com a chave privada do solicitante. Segue, abaixo, o comando utilizado para geração da solicitação:

```
root@ePad:~/Certificado# openssl req -new -key ca_raiz-key.pem -out ca_raiz-csr.pem
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:Tocantins
Locality Name (eg, city) []:Palmas
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Fabrica de Software Ltda.
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:ePad.com
Email Address []:
root@ePad:~#
```

²⁷ Interpretador de comandos, também conhecido como *shell*, presente em diversas distribuições Linux.

Por último, é gerado o certificado da AC raiz de nossa ICP. Por se tratar de uma AC Raiz, o certificado deve ser auto-assinado, pois não há nenhuma outra AC hierarquicamente superior. O certificado será composto com as informações contidas na CSR, como o *common name*, unidade organizacional, o país e a chave pública, além das informações pré-estabelecidas no arquivo de configuração utilizado pelo *OpenSSL*. Segue, abaixo, o comando utilizado para geração do certificado auto-assinado da AC Raiz:

```
root@ePad:~/Certificado# openssl x509 -req -in ca_raiz-csr.pem -signkey ca_raiz-
```

Nesse momento, a AC Raiz já está criada e apta a emitir certificados. Segue figura que exibe o certificado da AC Raiz:

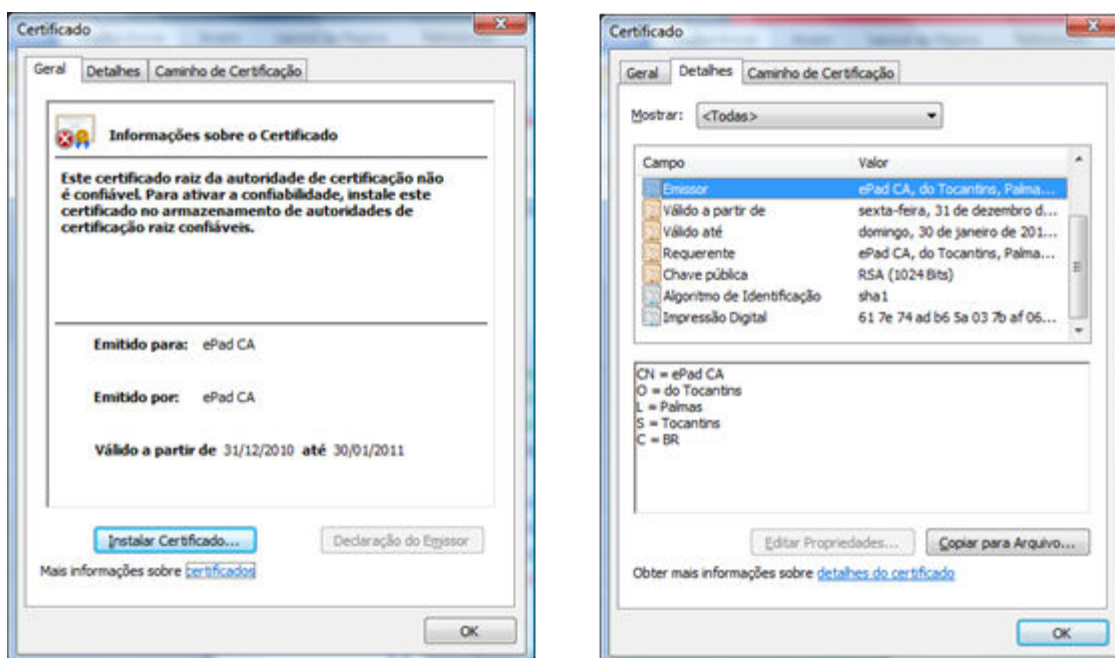


Figura 45 – Certificado Auto-assinado da AC Raiz

Após a criação da AC Raiz, são criadas, então, as autoridades certificadoras intermediárias. Em nossa ICP, criaremos apenas uma AC Intermediária, pois ela será suficiente para demonstrarmos o funcionamento do módulo de assinatura digital.

O processo de criação da AC intermediária é similar ao processo de criação da AC Raiz, com algumas poucas diferenças. Na AC Raiz, o certificado é assinado com sua própria chave privada, já na AC Intermediária o certificado é assinado com a chave privada da AC Raiz, além de ser composto, também, por alguns parâmetros

obtidos a partir da AC Raiz, como o *Distinguished Name* (DN) do emissor, o *Authority Key Identifier* (AKID) (vide 7.7.4.1), entre outros. A outra diferença é que o arquivo responsável por definir as extensões e restrições do certificado dever ser alterado, modificando-se o atributo CA de *false* para *true*. Desta forma, o certificado será reconhecido como um certificado do tipo CA durante a validação da cadeia de certificados e poderá ser utilizado para emissão de outros certificados digitais.

```
# This is what PKIX recommends but some broken software chokes on critical
# extensions.
#basicConstraints = critical,CA:true
# So we do this instead.
basicConstraints = CA:true
```

Figura 46 – Parâmetro *basicConstraints* presente no arquivo */etc/ssl/openssl.cnf*

Segue, abaixo, a sequência de comandos que foram utilizados na geração do certificado digital da AC Intermediária:

```
root@ePad:~# openssl genrsa -out ca_intermediaria-key.pem 1024
root@ePad:~# openssl req -new -key ca_intermediaria-key.pem -out ca_intermediaria-
csr.pem
root@ePad:~# openssl ca -cert ./ca_raiz-cert.pem -keyfile ./ca_raiz-key.pem -in
ca_intermediaria-csr.pem -out ca_intermediaria.crt
```

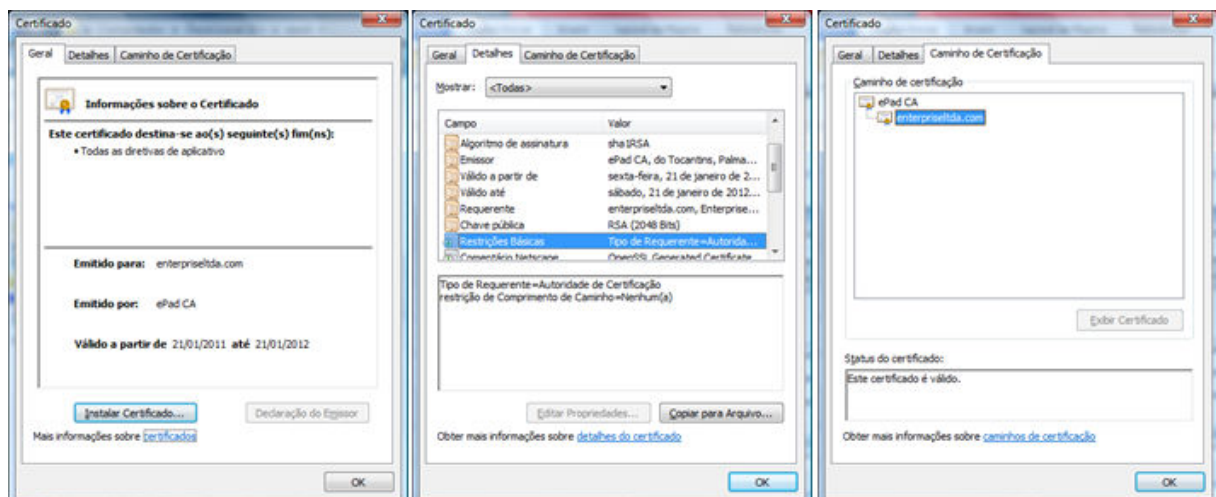


Figura 47 – Certificado da AC Intermediária emitido pela AC Raiz

7.4.5. Geração de certificado para assinatura de códigos

Para que o sistema ePad funcione corretamente, é necessário assinar os códigos móveis (ver 7.7.1 e 7.7.1.3) envolvidos no processo de assinatura dos documentos eletrônicos. Com isso, faz-se necessária a utilização de um certificado digital próprio para assinatura de código que será utilizado para assinar o módulo de assinatura do sistema.

Antes de dar início ao processo de criação do certificado de assinatura de código, é preciso alterar o arquivo `/etc/ssl/openssl.cnf` e redefinir os parâmetros que serão utilizados durante a emissão do certificado. Primeiramente, redefinimos o atributo CA de *true* para *false*, fazendo o caminho contrário ao da seção anterior (ver 7.4.4). Em seguida, definimos o atributo *extendedKeyUsage*, responsável por determinar as extensões de uso da chave, com o valor *codeSigning*, ficando da seguinte forma:

```
extendedKeyUsage = codeSigning
```

A partir de então, repete-se a sequência de comandos para criação da chave privada, geração da solicitação de assinatura e, por último, emissão do certificado digital. Segue, abaixo, a sequência de comandos utilizados na geração do certificado:

```
root@ePad:~/Certificado/signer# openssl genrsa -out singer.key
root@ePad:~/Certificado/signer# openssl req -new -key singer.key -out singer.csr
root@ePad:~/Certificado/signer# openssl ca -days 365 -cert ../ca_intermediaria-
cert.pem -keyfile ../ca_intermediaria-key.pem -in singer.csr -out singer.crt
```

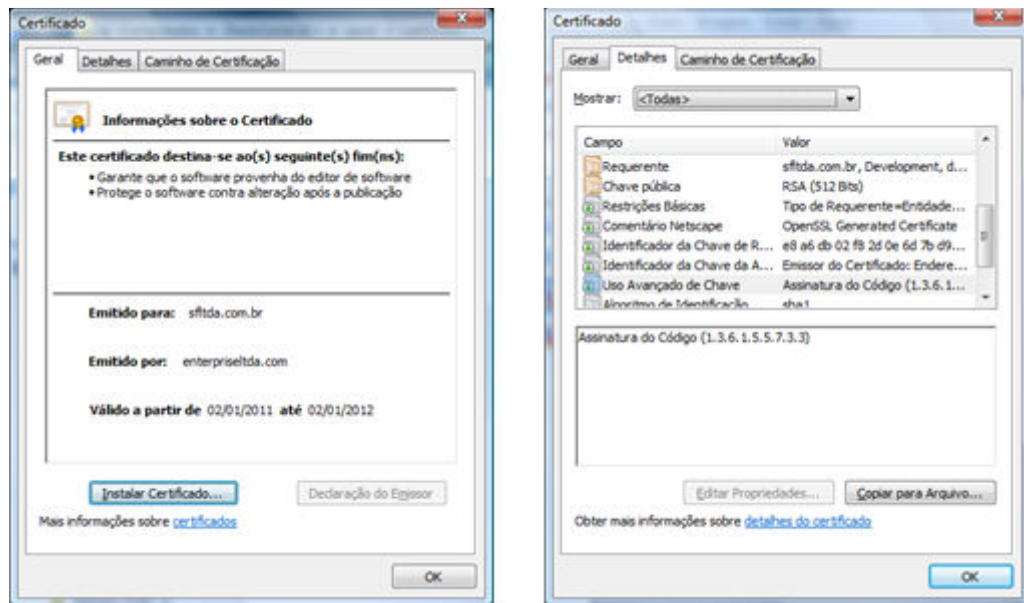


Figura 48 – Certificado de Assinatura de Código

7.4.6. Geração de certificado para Assinatura de Documentos

Para que se possa demonstrar o uso da ferramenta de assinatura digital presente no sistema, é necessária a utilização de certificados digitais próprios para assinatura de documento. Uma vez criadas as autoridade certificadores que compõem a ICP, pode-se, então, emitir os certificados que serão usados para assinatura de documentos, sendo esses, por sua vez, assinados pela AC intermediária.

Semelhantemente aos demais certificados citados anteriormente, os certificados digitais para assinatura de documentos necessitam do parâmetro *keyUsage* definido com um conjunto de valores específicos para o seu propósito. Os valores atribuídos são: *nonRepudiation* e *digitalSignature*.

Os valores *digitalSignature* e *nonRepudiation* tratam da assinatura digital e do não-repúdio, respectivamente. Quando esses valores estão definidos, o signatário da mensagem não poderá contestar sua assinatura, caso seja verificado que ele foi o autor. Da mesma forma que, se porventura alguém assinar um documento que não contenha tais valores no atributo *keyUsage* e no futuro vier a contestar essa assinatura, não será possível à outra parte reivindicar o não-repúdio, bem como a sua legitimidade, pois o certificado não fora emitido para tal fim.

keyUsage = nonRepudiation, digitalSignature

Depois de alterado o arquivo de configurações do *OpenSSL*, executam-se os comandos de geração de certificado digital. Seguem, abaixo, os comandos utilizados na geração do certificado digital de usuário final:

```
root@ePad:~/Certificado/moises# openssl genrsa -out moises.key 2048
root@ePad:~/Certificado/moises# openssl req -new -key moises.key -out moises.csr
root@ePad:~/Certificado/moises# openssl ca -cert ../ca_intermediaria-cert.pem -keyfile
../ca_intermediaria-key.pem -in moises.csr -out moises.crt
```

Em seguida, gera-se um arquivo no formato PKCS#12 que conterá a chave privada, o certificado digital do usuário e a cadeia de certificados digitais. O PKCS#12 é um padrão estabelecido pelos laboratórios da RSA e é especificado para ser um formato portátil para o armazenamento e transporte de chaves privadas de usuários, certificados, dados secretos, etc.. (RSA; 2010). Os comandos utilizados na criação do arquivo PKCS#12, de extensão *.p12*, são descritos abaixo:

```
root@ePad:~/Certificado/moises# cat ca_intermediaria-cert.pem ca_raiz-cert.pem >
cadeia.pem
root@ePad:~/Certificado/moises# openssl pkcs12 -export -chain -in cadeia.pem -inkey
moises.key -out moises.p12 -name "moises" -password 12345678
```

O arquivo *.p12* será utilizado posteriormente no momento da importação da chave privada e do certificado para dentro do *token* (ver 7.6.2).

Durante a geração dos certificados que compõem a ICP, todos os parâmetros que não foram fornecidos por meio de comando foram obtidos a partir do arquivo */etc/ssl/openssl.cnf*.

7.4.7. Geração da Lista de Certificados Revogados

Ao longo do período de validade de um certificado digital, alguns fatos que motivem a revogação de seu uso podem surgir, tornando-o inválido antes do término do prazo de validade. Os motivos podem ser diversos, indo desde o comprometimento da chave privada chegando até a cessação de operação.

Quando um certificado é revogado, a AC que o emitiu tem a responsabilidade de publicar a informação de revogação através de um sítio web. A publicação é feita com o lançamento do serial do certificado no arquivo que contém o registro de todos os certificados revogados emitidos pela AC. Esse conjunto de registros é denominado LCR.

Da mesma forma que os certificados digitais, uma LCR também apresenta um prazo de validade, sendo esse prazo definido pela AC que a emitiu. Antes do final do prazo de validade, a AC deverá reemitir a LCR, mantendo-a sempre atualizada. E é, portanto, por meio de uma LCR que se pode estabelecer a validade de um certificado digital durante o seu tempo de vigência.

7.4.7.1. Revogando um certificado digital e gerando uma LCR

O processo de revogação de certificado digital consiste de pelo menos dois passos: primeiro, deve-se informar os certificados que estão revogados e os motivos da revogação; em seguida, gera-se a lista de certificados revogados que conterá os seriais dos certificados revogados juntamente com a data da revogação de cada certificado, dentre outros valores. Ao final da execução dos passos, publica-se o arquivo em local previamente definido pela AC e informado no próprio certificado digital do usuário.

As informações que compõem uma LCR podem variar conforme a versão do formato que estiver sendo adotada. Atualmente, a LCR encontra-se em sua versão 2 e é definida pela RFC 5280, o qual especifica os atributos que podem ou devem compor a lista.

Geraremos as LCRs vinculadas a nossa ICP para posteriormente demonstrarmos o seu uso no sistema (ver 7.7.4.2 e 7.7.4.3). Segue, abaixo, a sequência de comandos adotados na revogação de um certificado digital:

```
root@ePad:~/Certificado/urias# openssl ca -revoke ../urias.crt -crl_reason  
cessationOfOperation  
root@ePad:~/Certificado/urias# openssl ca -gencrl -cert ../ca_intermediaria-cert.pem -  
keyfile ../ca_intermediaria-key.pem -out ../crl/ca-crl.pem
```

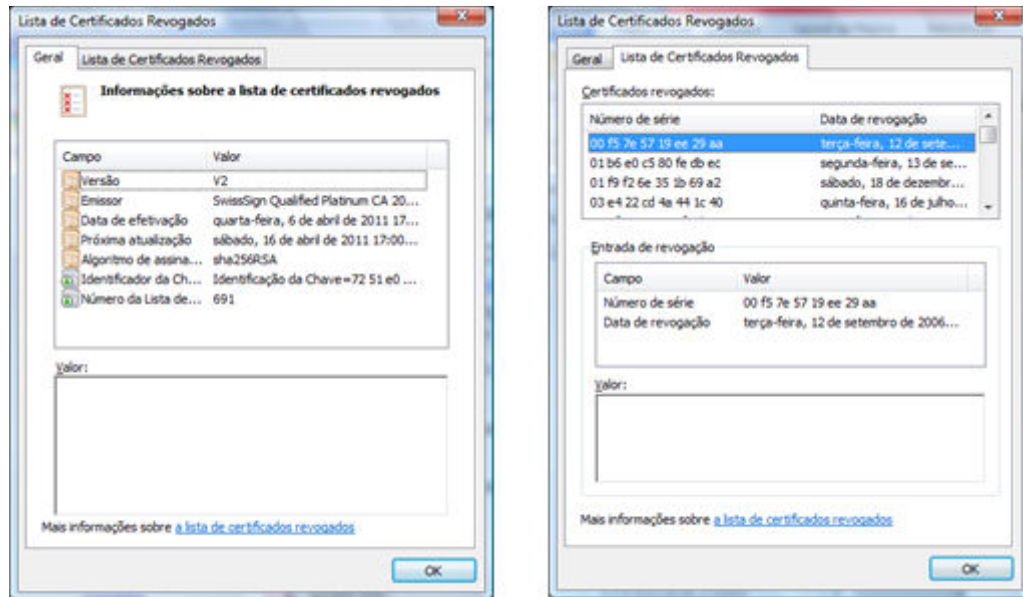


Figura 49 – Modelo de LCR (SwissSign)

7.5. Aspectos Legais para Utilização de Assinatura Eletrônica

A lei nº 11.419/2009, de 19 de dezembro de 2006, que dispõe sobre a informatização do processo judicial, determina que a prática de atos processuais, como o envio de petições e recursos, será aceita mediante uso de assinatura eletrônica.

Conforme o art. 1º e 2º da Lei nº 11.419:

Art. 1º O uso de meio eletrônico na tramitação de processos judiciais, comunicação de atos e transmissão de peças processuais será admitido nos termos desta Lei.

(...)

§ 2º Para o disposto nesta Lei, considera-se:

(...)

III - assinatura eletrônica as seguintes formas de identificação inequívoca do signatário:

- a) assinatura digital baseada em certificado digital emitido por Autoridade Certificadora credenciada, na forma de lei específica;
- b) mediante cadastro de usuário no Poder Judiciário, conforme disciplinado pelos órgãos respectivos.

Art. 2º O envio de petições, de recursos e a prática de atos processuais em geral por meio eletrônico serão admitidos mediante

uso de assinatura eletrônica, na forma do art. 1º desta Lei, sendo obrigatório o credenciamento prévio no Poder Judiciário, conforme disciplinado pelos órgãos respectivos.

Segundo Atheniense (2009):

Ainda em 2001, foi promulgada a medida provisória nº 2.200-2, que instituiu a Infra-estrutura de Chaves Públicas Brasileira (ICP-Brasil), conferindo presunção de validade jurídica aos documentos eletrônicos assinados digitalmente com certificados digitais emitidos por Autoridades Certificadoras credenciadas pela ICP-Brasil.

Posteriormente, a Lei 11.419/06 tornou obrigatória a assinatura digital nos atos processuais praticados por meio eletrônico, prevendo duas alternativas de assinatura eletrônica, seja baseada em certificado digital ou mediante cadastro por login e senha de usuário cadastrado no Poder Judiciário conforme disciplinado pelos respectivos órgãos (art. 1º, §2º, inc. III, “a” e “b”).

Ao nosso ver, a manutenção no texto da lei 11.419/06, da modalidade de protocolo de petições valendo-se da autenticação do requerente apenas em senhas e sem o uso da certificação digital para garantir a integridade na transmissão dos dados, irá representar a possibilidade de sérias vulnerabilidades, pois não haverá meios capazes de aferição se os dados que foram enviados por meio eletrônico tenham sido interceptados e alterados sem deixar algum indício de fraude.

Porém, segundo Advogados (2008):

A Lei nº. 11.419, de 19 de dezembro de 2006 prevê o uso de meio eletrônico para a tramitação de processos judiciais, a comunicação de atos e a transmissão de peças processuais. A utilização está prevista para os processos civis, penais e trabalhistas, além daqueles da competência dos juizados especiais.

(...)

O acesso ao sistema pode ser feito por meio da digitação do nome de usuário e senha, entregues ao usuário do sistema mediante identificação pessoal do mesmo. Os atos processuais devem ser assinados digitalmente, por meio de certificado digital (obrigatório para o envio de peças).

A obtenção dos meios de acesso dependerá de identificação presencial do interessado junto ao Poder Judiciário. A lei faculta aos órgãos do judiciário a criação de um cadastro único para o credenciamento previsto no artigo 2º., parágrafo 1º.

Desta forma, a Lei prevê um sistema misto de assinatura eletrônica: a digitação do usuário e senha para acesso ao sistema e a assinatura digital baseada em certificado digital, emitido por autoridade certificadora, que deverá constar nas peças enviadas ao sistema.

Isso se dá principalmente pela fragilidade do sistema de usuário e senha, quando empregado isoladamente. Além da necessidade de cadastro em todos os juízos e tribunais em que os profissionais do direito desejarem trabalhar, além do fato que a senha deve ficar armazenada no sistema, ao qual funcionários do departamento de informática podem ter acesso.

Como vimos, a legislação em vigor determina que a assinatura eletrônica deve ser utilizada por meio de cadastro de usuário no Poder Judiciário e assinatura digital como forma de autenticação de autos de processos. Contudo, há divergências

se ambos devem ser empregados juntos ou se podem ser empregados isoladamente. O que fica claro é que, quando utilizado o sistema de usuário e senha apenas, a assinatura eletrônica torna-se frágil, sujeitando-a a possíveis contestações de autoria.

Para livrar-nos de possíveis problemas quanto à vulnerabilidade no sistema, no tocante à contestação de autoria, adotaremos a interpretação que diz que as duas técnicas devem ser empregadas em conjunto.

7.6. Dispositivos Criptográficos

A adoção de certificados digitais no processo de autenticação elevou consideravelmente os padrões de segurança, tornando-a, em muitas situações, um pré-requisito dentro de um sistema seguro. Contudo, algumas questões relacionadas à segurança da informação não são resolvidas apenas com o uso da certificação. Podemos destacar o fato de que quando se transporta um arquivo .p12 em um ambiente sem maiores controles, sujeita-se a chave privada do usuário a uma possível exposição indesejada.

É sabido que um processo ou sistema é tão seguro quanto o seu elo mais fraco na cadeia de segurança da informação. O mesmo conceito se aplica à certificação digital: se a forma de transportar e utilizar a chave privada não é segura, então todo o processo torna-se inseguro. É neste contexto que se encaixa a adoção de dispositivos criptográficos capazes de prevenir contra a exposição ou utilização indesejada da chave privada do usuário.

Conhecidos como *smartcard*, o autor Cheswick (2003) o descreve da seguinte maneira:

Um cartão inteligente (*smartcard*) é um dispositivo portátil com uma CPU, algumas portas de entrada e saída e alguns mil *bytes* de memória não-volátil acessível somente por meio da CPU do cartão. (...) Os cartões inteligentes são “algo que você tem”, apesar de frequentemente serem aumentados por “algo que você conhece”, um PIN.

Quando importamos a chave privada ou a geramos dentro de um *smartcard*, o dispositivo grava a chave em uma área de memória que só pode ser lida pelo próprio cartão. O dispositivo por sua vez apresenta um conjunto de funções criptográficas que podem utilizar a chave privada como parâmetro, por exemplo, funções de criptografia e assinatura. Desta forma, a chave privada nunca precisará

ser lida por qualquer agente externo, ficando todo o acesso à chave restrito à *Central Processing Unit* (CPU) do *smartcard*.

No momento que o usuário deseja realizar uma operação de assinatura, ele informa o *Personal Identification Number* (PIN)²⁸ ao cartão que deverá autenticá-lo. Caso o PIN esteja correto, o cartão habilitará as funções criptográficas presente no dispositivo, caso contrário, negará o acesso e informará o usuário do ocorrido. Em muitos dispositivos, a ocorrência de tentativas de acesso sequenciais com o uso de um PIN incorreto dispara uma rotina de limpeza da área de memória do cartão. Esse mecanismo previne contra ataques do tipo força-bruta, assegurando que a utilização do *smartcard* fica obrigatoriamente condicionada ao conhecimento prévio do PIN, o que impede o uso do cartão por um terceiro que detenha a posse não autorizada.

Como o uso de *smartcards* e *tokens*²⁹ para armazenamento da chave privada e certificado do usuário, bem como a cadeia de certificados, já é amplamente aceito e difundido no meio corporativo, adotaremos a utilização de dispositivos criptográficos como o único meio disponível no sistema para assinatura de documentos.

7.6.1. O padrão PKCS#11

A fim de possibilitar a utilização de diversos tipos de dispositivos criptográficos por múltiplos sistemas de informação, foi estabelecido pelos Laboratórios RSA o padrão PKCS#11.

O padrão PKCS#11 especifica uma API, chamada de *Cryptoki*, para dispositivos que desempenham funções de criptografia de chave pública. A API *Cryptoki* propicia a independência de tecnologia (uso de qualquer dispositivo) e o compartilhamento de recursos, fornecendo às aplicações uma interface lógica comum dos *tokens* criptográficos. (RSA; 2010) (ver mais em 7.7.3)

²⁸ Personal Identity Number. Código de acesso definido pelo usuário e requerido pelo dispositivo criptográfico.

²⁹ "Token: geralmente um dispositivo criptográfico que obedece aos mesmos padrões do smart card."

(MARTINI, Renato da Silveira. Tecnologia e cidadania digital: tecnologia, sociedade e segurança; 2008, p. 189).

7.6.2. Copiando o certificado para dentro do token

Como a utilização do módulo de assinatura do sistema ePad está condicionado ao uso de *tokens* criptográficos, utilizaremos o *token* iKey2032, da marca *SafeNet*, para fins de demonstração da funcionalidade do sistema.

Depois de instalado o *driver* do *token* e obtido o certificado digital do usuário no formato PKCS#12, importa-se o certificado por meio do aplicativo do próprio *token*. Para isso, primeiro realiza-se o *login* no *token* informando o PIN previamente cadastrado:

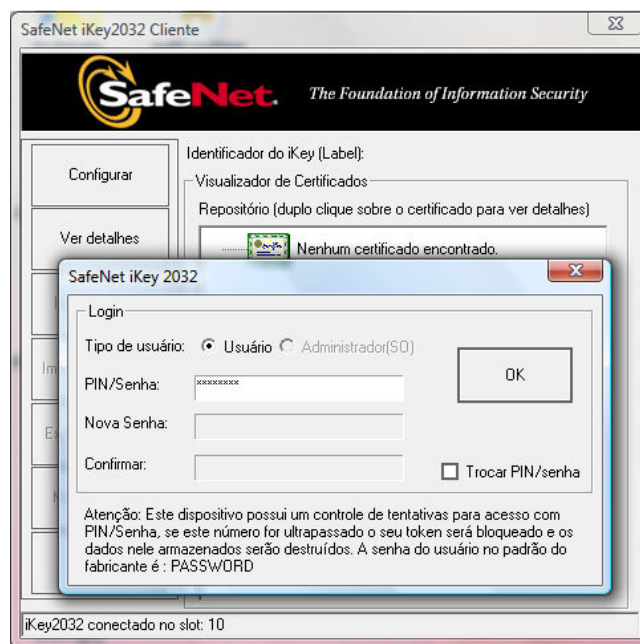


Figura 50 – Autenticando o usuário no token SafeNet iKey 2032

Em seguida, informa-se o arquivo .pfx que contém o certificado digital, a chave privada e a cadeia de certificados para importação:

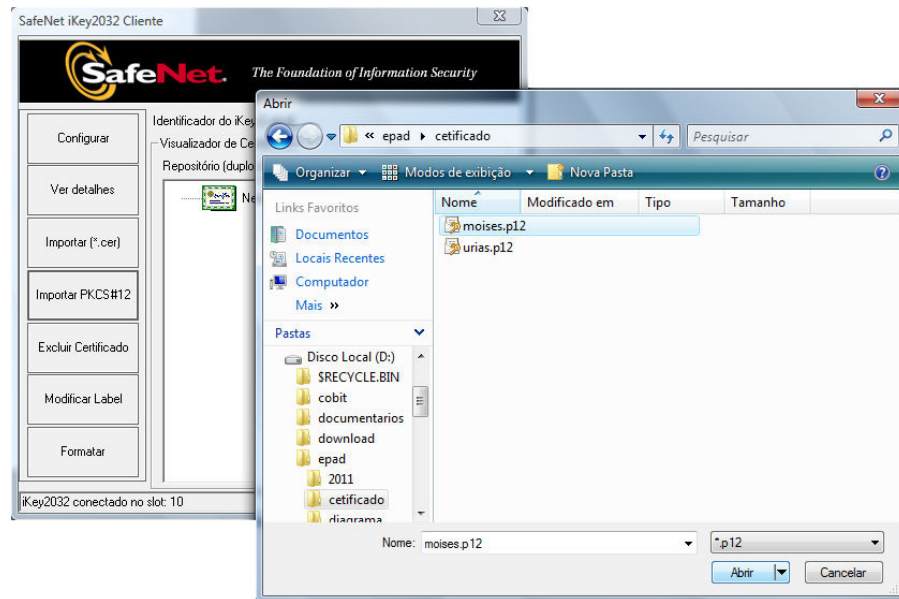


Figura 51 – Importação de certificado digital no SafeNet iKey 2032

Insere-se, então, a senha para decriptografar o arquivo .pfx e o *label* que identificará o certificado no *token* criptográfico:

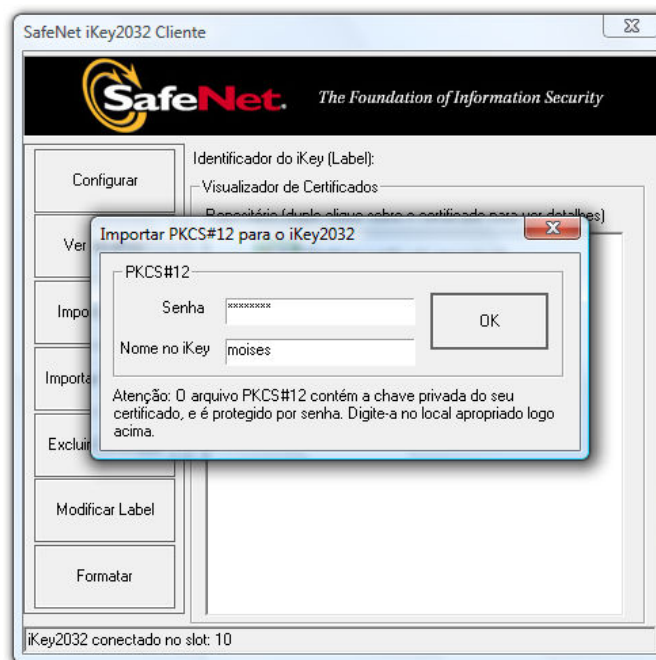


Figura 52 – Informando a senha de decriptografia do arquivo .pfx e o nome do certificado alojado no token

Ao final do processo, o *token* conterá o certificado digital e a chave privada associada:

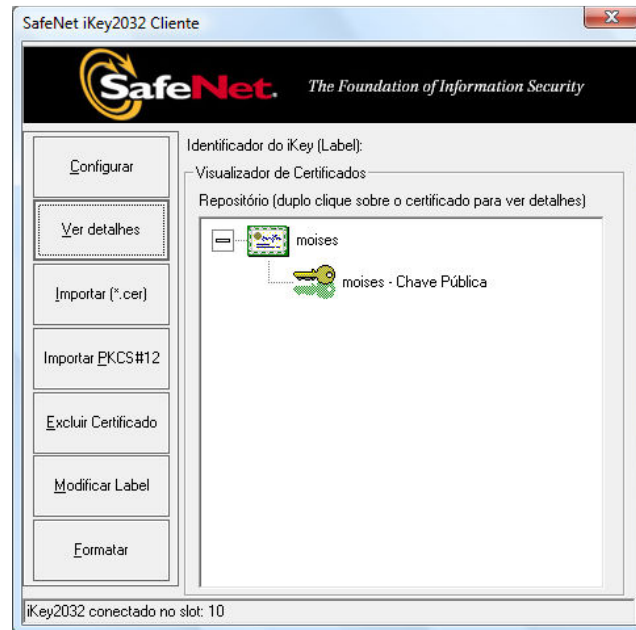


Figura 53 – Token criptográfico com um certificado digital

7.7. Processo de Assinatura de Documentos

Um dos pré-requisitos do sistema ePad é a necessidade de se assinar os documentos eletrônicos antes do seu envio. Desta forma, surge a necessidade de se desenvolver um aplicativo capaz de gerar a assinatura a partir do documento que se deseja assinar e com a utilização certificado digital do usuário signatário do documento.

A inserção de documentos em um processo deve seguir uma sequência lógica de passos. A sequência começa pelo carregamento do arquivo que se deseja inserir no processo. Em seguida, insere-se o *token* que contém a chave privada e o certificado do usuário utilizados na assinatura na máquina do usuário para que seja reconhecido pelo aplicativo. Por último, gera-se a assinatura no formato CMS/PKCS#7 juntamente com as validações necessárias e se envia o bloco de dados de assinatura juntamente com o arquivo para o servidor que os receberá e armazenará.

O diagrama de sequência do processo de assinatura (ver apêndice) demonstra o processo de assinatura. Esse aplicativo deverá ser executado no lado do cliente, ou seja, ele deverá rodar por meio do *browser* do usuário. Isso é

justificado pelo fato do *token* utilizado na assinatura ser acessível somente a partir da máquina onde estiver plugado e instalado para uso.

7.7.1. Código móvel

Também conhecido como código sob demanda, o código móvel é uma denominação a um conjunto de tecnologias de linguagem e plataforma de sistemas distribuídos disponibilizados em servidores remotos e transferidos sob demanda para os computadores clientes. Após o recebimento do código, os computadores clientes iniciam sua execução automaticamente ou submetem ao usuário questionando-o se deve ou não executá-lo. (UNB, 2011)

Uma das vantagens da utilização do código móvel sobre as aplicações executadas remotamente é a descentralização da carga de processamento localizada no servidor entre os computadores clientes, liberando recursos do computador remoto. Outra vantagem é a possibilidade de acessar os dispositivos localizados na máquina do cliente, o que proporciona a capacidade de extensão das funcionalidades da aplicação.

Entre os tipos de código móvel mais comuns, podemos destacar os seguintes:

- JavaScript;
- Applet;
- Flash;
- Middleware;
- Alua;
- Vírus.

7.7.1.1. *Uso de Applet*

Para realizar as atividades necessárias à assinatura, é preciso adotar uma tecnologia de código móvel que seja capaz de ler arquivos presentes em discos e que seja capaz de acessar dispositivos criptográficos, chamando suas funções e obtendo os resultados. Entre as tecnologias disponíveis o *applet* é um dos que oferecem o suporte necessário e com grande aceitação no mercado.

Desenvolvido pela Sun Microsystems, o *applet* é um programa escrito na linguagem de programação Java. Ele pode ser incluído em páginas HTML de forma semelhante à inclusão de um objeto, como uma imagem. Quando o usuário acessa

uma página que contenha um *applet* e o navegador oferece suporte à tecnologia, o código é transferido do servidor web para o sistema do usuário e é, então, executado por meio de uma JVM. (ORACLE, 2011)

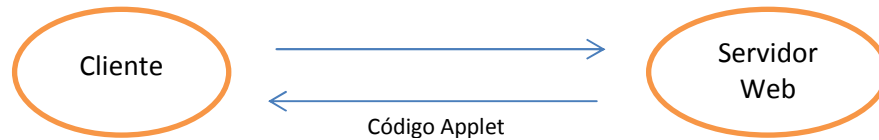


Figura 54 – Requisição de um cliente e resposta com o envio de um applet (Adaptado: COULOURIS, DOLLIMORE, KINDBERG)



Figura 55 – Execução de código applet no ambiente do cliente

Em geral, o suporte ao *applet* por parte dos navegadores se dá através da instalação de plug-ins específicos. A Oracle, por sua vez, disponibiliza gratuitamente o *plug-in* do *java* em seu sítio web.

7.7.1.2. Empacotando o aplicativo dentro de um JAR

Ao criar um aplicativo gráfico em *java* utilizando código *applet*, é comum o IDE gerar um grande conjunto de arquivos necessários à execução do programa. Esses arquivos devem ser publicados em conjunto, devendo ser mantidos na mesma estrutura de diretório em que foram gerados. A manipulação de um grande grupo de arquivos acarreta um custo operacional elevado ao desenvolvedor, ao mesmo tempo em que aumenta o risco de erro.

A fim de facilitar a manipulação de arquivos “.class” e auxiliares, a Oracle disponibiliza uma ferramenta capaz de gerar arquivos do tipo *Java Archive* (JAR). O formato de arquivo JAR provê um conjunto de benefícios que podem ser adotados durante a implantação de um código *applet*, como: segurança, por meio de assinatura digital, onde você pode reconhecer a autoria do código e estender privilégios de segurança; redução de tempo de *download*, uma vez que será necessária uma única transação HTTP para baixar o pacote com todos os arquivos, ao invés uma transação para cada arquivo; compressão, reduzindo o tamanho do pacote por meio de compressão sem perda; versionamento de pacote, pois um

pacote JAR pode conter a versão de seus arquivos; portabilidade, já que o mecanismo de manipulação de arquivos JAR é uma parte da API da plataforma Java. (ORACLE, 2011)

Para criar o pacote contendo o aplicativo necessário à geração da assinatura digital, foi utilizado o programa *jar.exe* localizado na área de instalação do JDK 1.6.0_24 para plataforma *Microsoft Windows*. Segue, abaixo, a linha de comando utilizada:

```
D:\projetos_cvs\ePad\build\web\WEB-INF\classes>jar -cf SmartCard.jar ./signer/
```

Após a execução da linha de comando, é criado o pacote *SmarCard.jar* que contém todos os arquivos necessários para a execução do applet de assinatura de documentos. Dentro desse pacote também é criado o arquivo *MANIFEST.MF*, responsável por armazenar as “meta” informações relacionadas aos arquivos contidos no pacote como controle de versão, extensões, etc.. (Oracle, 2011).

```
Manifest-Version: 1.0
Created-By: 1.6.0_24 (Sun Microsystems Inc.)
```

Figura 56 – Arquivo MANIFEST.MF original

7.7.1.3. Assinando o aplicativo

A despeito do grande número de funcionalidades disponíveis e vantagens oferecidas, o modelo de código móvel *applet* apresenta alguns problemas relacionados à segurança. Isso ocorre porque o código executado na máquina do cliente é obtido de um servidor remoto em que o usuário nem sempre pode confiar, sujeitando o usuário ao risco de execução de códigos maliciosos. (SANTOS, 2008).

A fim de prevenir ataques, a arquitetura de segurança do Java estabeleceu um modelo de execução em que os programas executados a partir de códigos remotos, como o *Applet*, têm o acesso às funções do sistema operacional da máquina limitado. Algumas limitações impedem, por exemplo, o acesso à dispositivos da máquina, leitura ou escrita de arquivos, abertura de *sockets* de rede para *hosts* distintos, etc.. (ORACLE, 2011)

Esse modelo que é conhecido por *sandbox* limita o acesso dos programas aos recursos do sistema e é controlado pelo componente *security manager*, sendo este a classe responsável por permitir a aplicação de política de segurança. (Oracle,

2011). Nesse sentido, os *applets* sempre serão executados sob a supervisão do *security manager*, o que limita sua atuação dentro do sistema operacional e reduz sua utilidade em aplicações comerciais. (SANTOS, 2008)

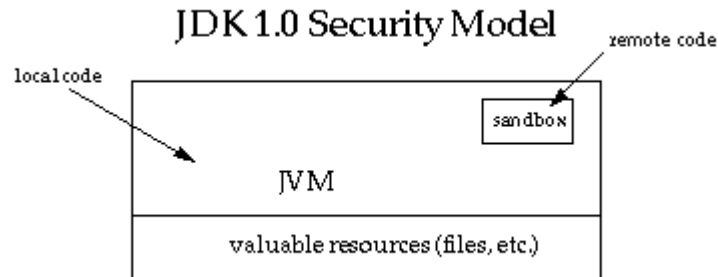


Figura 57 – Ilustração do modelo de segurança com o uso do *sandbox* (Oracle)

Para que os *applets* pudessem fazer uso dos mesmos privilégios de acesso ao sistema operacional que as aplicações locais possuem, a plataforma *Java*, a partir do JDK 1.1, introduziu o conceito de “*applet* assinado” ou “*signed applet*”. Nesse conceito, os *applets* que tiverem o seu código assinado terão os mesmos privilégios de segurança que as aplicações que possuem os códigos armazenados localmente, bastando somente que a assinatura seja reconhecida como confiável pelo *browser* do usuário. (ORACLE, 2011)

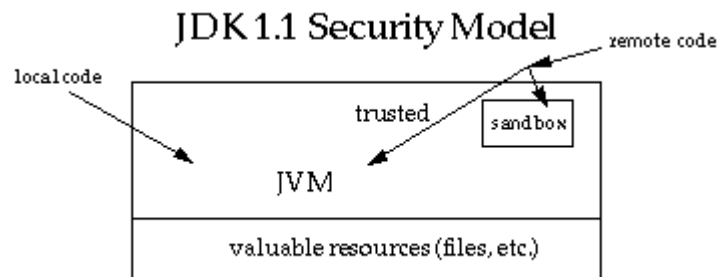


Figura 58 – Modelo de segurança a partir do JDK 1.1 com a adoção de código assinado (Oracle)

O processo de assinatura de código no Java consiste em, primeiramente, gerar o pacote JAR e, em seguida, assiná-lo com o certificado e a chave privada do autor utilizando o aplicativo *jarsigner* do Java. Segue, abaixo, o comando utilizado na assinatura do código *applet*:

```
root@ePad:~/Certificado/signer# $JAVA_HOME/bin/jarsigner -keystore ./signer.p12 -
storetype pkcs12 SmartCard.jar codesigner
```

Ao final do processo, o aplicativo *jarsigner* altera o pacote original adicionando mais dois arquivos e modificando o arquivo MANIFEST.MF. No arquivo MANIFEST.MF, são adicionados os *hash* de cada arquivo contido no pacote:

```
Manifest-Version: 1.0
Created-By: 1.6.0_24 (Sun Microsystems Inc.)

Name: signer/TrustedAnchorParser.class
SHA1-Digest: vfqlIbC25Z7ibpraYQFmqulVxJ8=

Name: signer/PKCS12Parser.class
SHA1-Digest: FaX0no8zT8gzS0YuOZtIhwa8TcA=

Name: signer/SmartCardSignerForEditor.class
SHA1-Digest: m39ZbEgER7hEHVa9Js5x5tCLeQQ=

Name: signer/X509ChainValidator.class
SHA1-Digest: 7kMwaNvyz4M0Jx324xu6g0welFc=

Name: signer/SmartCardSignerForUpload$2.class
SHA1-Digest: 7ZBRu8J/coLVMfJWO++1GW0VEq4=

Name: signer/SmartCardSignerForUpload.class
SHA1-Digest: 9K94/HD2B8qclneBLI0HlhvAx8g=

Name: signer/SmartCardSignerForEditor$1.class
SHA1-Digest: 0tz8J6F1HGFLssJ1lQwO5dFL1tw=

Name: signer/SmartCardSignerForEditor$3.class
SHA1-Digest: Fly66jLb0LcZDbJoQLnVfcNaAsY=

Name: signer/CRLLocator.class
SHA1-Digest: SBYl0E7sMSPL0lfoMxzGoRWIXmY=

Name: signer/SmartCardSignerForUpload$4.class
SHA1-Digest: 0HXaxN3flrQmc9ePWtZl0jcLP14=

Name: signer/SmartCardSignerForUpload$3.class
SHA1-Digest: 47kN7bLTPQfJzgV6JVlQaLDVVyg=
```

Figura 59 – Arquivo MANIFEST.MF alterado

Além da modificação do arquivo MANIFEST.MF, são criados dois arquivos que mantêm os dados da assinatura. O primeiro possui a extensão “.SF” e contém a assinatura de cada arquivo do pacote. O segundo, por sua vez, é um arquivo de assinatura de bloco e pode possuir a extensão “.RSA”, “.DSA” ou “.PGP”, pois a extensão varia conforme o algoritmo utilizado na assinatura. Este último representa a assinatura digital do arquivo “.SF”. (Oracle, 2011).

```

Manifest-Version: 1.0
Created-By: 1.6.0_24 (Sun Microsystems Inc.)

Name: signer/TrustedAnchorParser.class
SHA1-Digest: vfq1IbC25Z7ibpraYQFmqulVxJ8=

Name: signer/PKCS12Parser.class
SHA1-Digest: FaX0no8zT8gzS0YuOZtIhwa8TcA=

Name: signer/SmartCardSignerForEditor.class
SHA1-Digest: m39ZbEgER7hEHVa9Js5x5tCLeQQ=

Name: signer/X509ChainValidator.class
SHA1-Digest: 7kMwaNvyz4M0Jx324xu6g0welFc=

Name: signer/SmartCardSignerForUpload$2.class
SHA1-Digest: 7ZBRu8J/coLVMfJWO++1GW0VEq4=

Name: signer/SmartCardSignerForUpload.class
SHA1-Digest: 9K94/HD2B8qclneBLI0HlhvAx8g=

Name: signer/SmartCardSignerForEditor$1.class
SHA1-Digest: 0tz8J6F1HGFLssJ11QwO5dFLltw=

Name: signer/SmartCardSignerForEditor$3.class
SHA1-Digest: Fly66jLb0LcZDbJoQLnVfcNaAsY=

Name: signer/CRLLocator.class
SHA1-Digest: SBY10E7sMSPL0lfoMxzGoRWIXmY=

Name: signer/SmartCardSignerForUpload$4.class
SHA1-Digest: 0HXaxN3flrQmc9ePWtZl0jcLP14=

Name: signer/SmartCardSignerForUpload$3.class
...

```

Figura 60 – Arquivo MANIFEST.SF

7.7.1.4. Publicando o aplicativo

A publicação do código móvel é feita por meio da adição de *tags* HTML na página da qual se deseja carregar o aplicativo. Para isso, utiliza-se as *tags applet*, juntamente com o seus principais atributos: *id*, *codebase*, *archive* e *param*. O atributo *param* é o responsável por passar parâmetros de entrada no início do aplicativo.

Segue o código de publicação do arquivo SmartCard.jar utilizado neste sistema:

```

<applet id="appSigner"
  alt="Módulo de Assinatura Digital" code="signer/SmartCardSignerForUpload.class"
  codebase="#{facesContext.externalContext.request.contextPath}/signer"
  archive="SmartCard.jar,bcmail-jdk16-145.jar,bcprov-jdk16-145.jar,bctsp-jdk16-
  145.jar"
  width="750"
  height="310">
  <param name="charset" value="#{utilitarioConfiguracao.CHARSET}"/>
  Seu navegador provavelmente não está habilitado para rodar Applets.
</applet>

```

Figura 61 – ePad – Trecho do arquivo formProcessoDocumento.xhtml

7.7.2. Carregando o arquivo para assinatura

Para que se possa assinar um documento, é necessário primeiro carregá-lo em memória. No formulário de assinatura, foi criado um botão que ao ser clicado executa o *jButtonCarregarActionPerformed()*. O método, por sua vez, abre uma caixa de diálogo que permite escolher o arquivo que se deseja carregar.

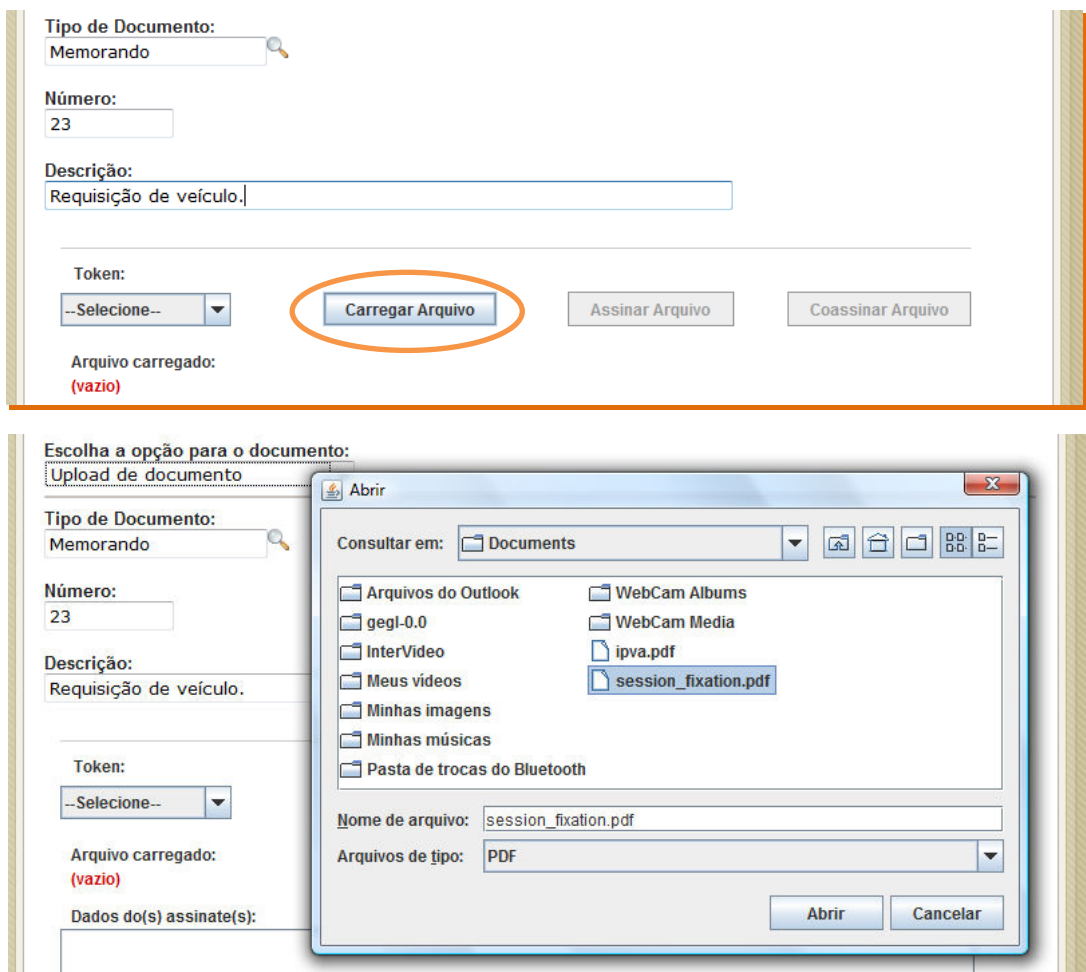


Figura 62 – Carregando o documento

Quando o arquivo é escolhido, o aplicativo cria uma referência que é atribuída a uma instância da classe *java.io.File* como o nome *file*. Em seguida, cria-se uma instância da classe *java.io.InputStream* a partir do objeto *file*. Com a instância criada, lê-se os blocos de dados para dentro do objeto de nome *documento* do tipo *byte[]*. A leitura é feita através do método *read()* da classe *InputStream*. É este último objeto que conterá os bytes referentes ao documento em todo o processo de assinatura.

Segue o trecho de código do método *jButtonCarregarActionPerformed()* responsável pela carga do documento em memória:

```

238 private void jButtonCarregarActionPerformed(
|         java.awt.event.ActionEvent evt) {
|
239     try {
240         JFileChooser chooser = new JFileChooser();
241         chooser.setFileFilter(new FileNameExtensionFilter("PDF", "pdf"));
242         if (chooser.showOpenDialog(null) == JFileChooser.CANCEL_OPTION) {
243             return;
244         }
245
246         File file = chooser.getSelectedFile();
247         InputStream inputFile = new FileInputStream(file);
248
249         setDocumento(new byte[inputFile.available()]);
250         inputFile.read(getDocumento());
251
252         jLabelArquivo.setText(file.getName());
253     } catch (Exception e) {

```

Figura 63 – ePad – Trecho do método *jButtonCarregarActionPerformed*

7.7.3. Acessando tokens criptográficos

A plataforma Java define um conjunto de interfaces de programação para execução de operações criptográficas. Estas interfaces são coletivamente conhecidas como a *Java Cryptography Architecture* (JCA). (ORACLE, 2011, tradução nossa)

As interfaces criptográficas são baseadas em providers. Especificamente, as aplicações conversam com as APIs e as operações criptográficas são na verdade executadas por meio de *providers* configurados, os quais aderem a um conjunto de SPIs³⁰. Esta arquitetura suporta implementações de diferentes *providers*. Alguns *providers* podem executar operações criptográficas em software; outros, por sua vez, podem executar as operações por meio de um *token* criptográfico, por exemplo,

³⁰ Service Provider Interfaces.

um smartcard ou um acelerador de hardware criptográfico. (ORACLE, 2011, tradução nossa)

O Cryptographic Token Interface Standard, PKCS#11, é produzido pela RSA Security e define interfaces nativas de programação para *tokens* criptográficos, tais como aceleradores de hardware criptográfico e *smartcards*. A fim de facilitar a integração de *tokens* PKCS#11 nativos dentro da plataforma Java, um novo *provider* criptográfico, o Sun PKCS#11, tem sido introduzido dentro da publicação do J2SE 5.0. Este novo *provider* habilita o acesso a *tokens* PKCS#11 nativos por parte das aplicações escritas para as APIs JCA e JCE. Não é requerida qualquer modificação nas aplicações. A única exigência é a configuração apropriada do *provider* dentro do *Java Runtime*. (ORACLE, 2011, tradução nossa)

O *provider* Sun PKCS#11, em contraste com outros *providers*, não implementa algoritmos de criptografia propriamente. Ao invés disso, ele age como uma ponte entre a API JCA e JCE do Java e a API criptográfica PKCS#11, traduzindo as chamadas e convenções entre ambos. Isso significa dizer que as aplicações Java que utilizam as APIs JCA e JCE podem, sem modificação, tomar vantagens dos algoritmos oferecidos pela camada de implementação do PKCS#11, por exemplo: (ORACLE, 2011, tradução nossa).

- *Smartcards* criptográficos;
- Aceleradores de hardware criptográfico; e
- Implementações de software de alta performance. (Oracle, 2011, tradução nossa).

No sistema ePad, a obtenção de uma assinatura pressupõe o uso de um dispositivo criptográfico. E para que seja possível utilizá-lo, primeiro devem ser mapeados os seus *drivers*, para que, em seguida, sejam feitas as chamadas às funções apresentada por eles.

Apesar das funções criptográficas presentes em um *token* serem, em geral, semelhantes a todos dos demais fabricantes, o modo de acessá-las varia de produto a produto, pois não há uma padronização que possibilite a adoção de um único *driver* genérico para todos os modelos de *token*, o que obriga a adoção de *drivers* específicos para cada modelo fabricado. Os *drivers* fornecem, por sua vez, uma interface baseada na API *Cryptoki* e é por meio dessa interface que uma determinada aplicação realiza as chamadas às funções de um *token*.

Quando falamos que não há uma padronização que possibilite a adoção de um único *driver* genérico para todos os modelos de *token*, fazemos referência ao padrão PKCS#11. Contudo, já existe um novo padrão que tem o objetivo de unificar a forma de acesso e possibilitar a utilização de um *driver* genérico para acesso às funções básicas de qualquer *token* que tenha sido desenvolvido em conformidade ao padrão. Este padrão é conhecido por PKCS#15³¹. Como a adoção a esse padrão ainda não está tão difundida quanto a do padrão PKCS#11, o que o torna menos comum entre os modelos de dispositivos conhecidos, não o consideraremos neste trabalho.

Já que é preciso mapear os *drivers* de cada modelo de *token* que se deseja utilizar, restringiremos, então, os modelos que serão fornecidos suporte em nosso sistema, lembrando que não há qualquer impedimento para que se possa adicionar outros modelos ao longo do ciclo de vida do sistema. Os modelos utilizados serão o *iKey 2032* e o *eToken Alading*.

Foi criada uma classe que armazena todas as informações necessárias ao acesso aos *token* que são suportados pelo sistema. Cada *token* terá os seus dados registrados em um elemento de uma lista do tipo *ArrayList<String>*. Os dados serão compostos pelo nome do *token*, uma referência à sua biblioteca que atua como *provider* da API *Cryptoki*, sendo um arquivo “.dll” no *Windows* e um “.so” no *Linux* e eventualmente um ou mais parâmetros de configuração específica do token. A figura abaixo é um trecho da classe *Cards* responsável por armazenar os dados:

```

15  public class Cards {
16
17      private List<String> listCards;
18
19      public Cards() {
20          listCards = new ArrayList<String>();
21          listCards.add("");
22          listCards.add( "name = iKey2032\n" +
23                        "library = c:/windows/system32/dkck201.dll\n" +
24                        "disabledMechanisms = { CKM_MD5_RSA_PKCS
25                                                CKM_SHA1_RSA_PKCS }");
26          listCards.add( "name = eTokenAlading\n" +
27                        "library = c:/WINDOWS/system32/eTPKCS11.dll");
28      }
29
30      public List<String> getListCardsWindows() {
31          return listCards;

```

³¹ "PKCS #15 establishes a standard that enables users in to use cryptographic tokens to identify themselves to multiple, standards-aware applications, regardless of the application's cryptoki (or other token interface) provider." (RSA, Laboratories. Cryptographic Token Information Format Standard. 2011)

32

Figura 64 – Cards.java - Trecho de código que armazena os dados dos tokens

O usuário, após inserir o *token* no computador, deverá então escolher qual dos dispositivos cadastrados ele deseja utilizar. Ao selecionar um dos itens listados, surgirá uma caixa de mensagem em que ele informará o PIN de acesso ao *token*.

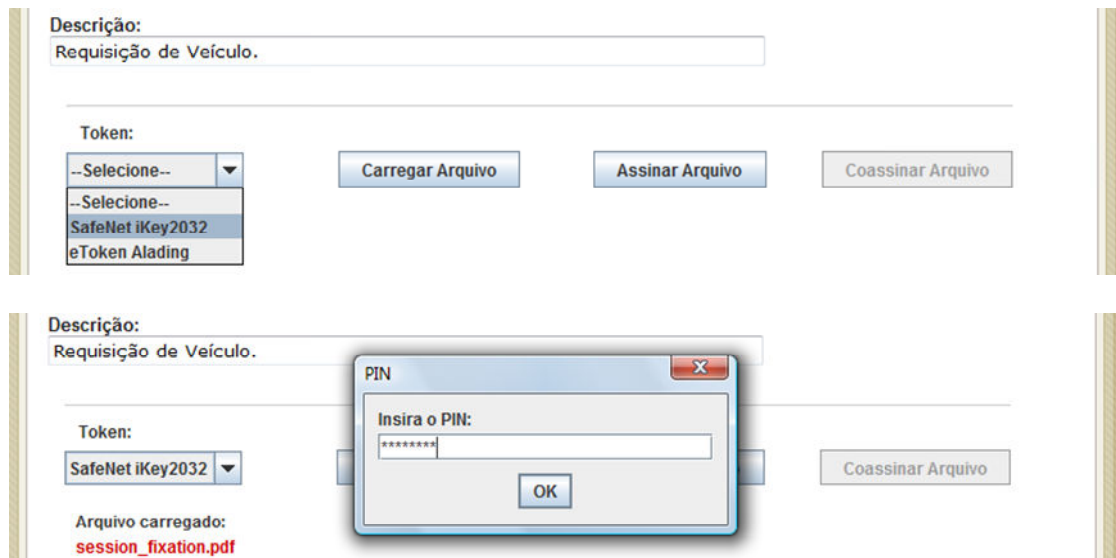


Figura 65 – ePad – Selecionando o token e inserindo a senha

Uma vez informado o PIN, o *applet* cria uma instância do *provider* *sun.security.pkcs11.SunPKCS11* com configurações específicas advindas da seleção do *token*. Em seguida, ele instala dinamicamente o *provider* por meio do método estático *addProvider()* da classe *Security*. Ao final, com a instância do *provider* criada e instalada, o *applet* poderá recuperar a chave privada, o certificado digital e a cadeia de certificados digitais presentes no *token*.

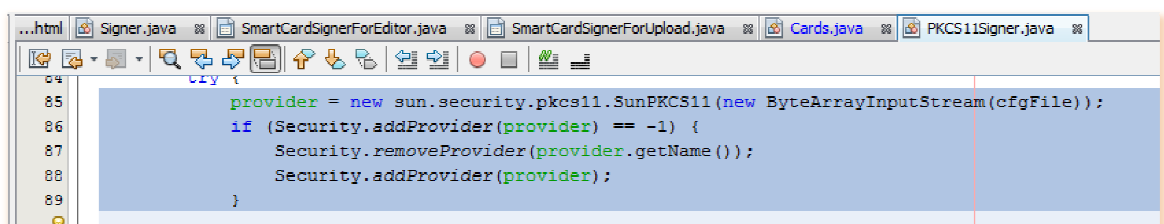


Figura 66 – Instalação do provider SunPKCS#11

7.7.3.1. Recuperando a chave privada

Após a instalação do provider *SunPKCS#11*, o *applet* executa o método *getInstance* para retornar um instância da classe *KeyStore* passando o objeto que representa o provider como parâmetro. A classe *KeyStore* foi desenvolvida para facilitar o armazenamento de chave criptográficas e certificados. (ORACLE, 2011)

```

85     provider = new sun.security.pkcs11.SunPKCS11( new
        ByteArrayInputStream(cfgFile) );
86     if (Security.addProvider(provider) == -1) {
87         Security.removeProvider(provider.getName());
88         Security.addProvider(provider);
89     }

```

Figura 67 – Recuperando uma instância do KeyStore

Com a instância da classe *KeyStore* devidamente atribuída, é possível realizar a recuperação da chave privada. Contudo, quando o *provider* informado na obtenção da instância da *KeyStore* for do tipo *SunPKCS#11*, o método *getKey*, que originalmente deveria retornar a própria chave privada, retorna apenas uma referência a ela. Isso porque a chave privada contida no *token* criptográfico não pode ser acessada externamente (ver 7.6).

```

55     if (keyEntryFound) {
56         cert = (X509Certificate) ks.getCertificate(keyEntry);
57         privateKey = (PrivateKey) ks.getKey(keyEntry, pin.toCharArray());
58         certChain = ks.getCertificateChain(keyEntry);
59     }

```

Figura 68 – Recuperando a chave privada

7.7.3.2. Recuperando o certificado

A recuperação do certificado digital armazenado no *token* é semelhante à recuperação da chave privada, com a única diferença que, ao invés de chamar o método *getKey* da classe *KeyStore*, chama-se o método *getCertificate*.

```

55     if (keyEntryFound) {
56         cert = (X509Certificate) ks.getCertificate(keyEntry);
57         privateKey = (PrivateKey) ks.getKey(keyEntry, pin.toCharArray());
58         certChain = ks.getCertificateChain(keyEntry);
59     }

```

Figura 69 – Recuperando o certificado digital do usuário

7.7.3.3. Recuperando a cadeia de certificados

Da mesma forma, a cadeia de certificados obtida por meio do *KeyStore*, só que com chamada ao método *getCertificateChain*.

```

55     if (keyEntryFound) {
56         cert = (X509Certificate) ks.getCertificate(keyEntry);
57         privateKey = (PrivateKey) ks.getKey(keyEntry, pin.toCharArray());
58         certChain = ks.getCertificateChain(keyEntry);
59     }

```

Figura 70 – Recuperando a cadeia de certificados

7.7.4. Validação do caminho do certificado

O processo de validação garante que as informações contidas no certificado ainda sejam válidas, pois as informações podem ser alteradas ou invalidadas ao longo do tempo. Em um ambiente composto por um grande número ACs, as ACs pertencentes à cadeia precisam validar umas as outras, seguindo um caminho hierárquico, até chegar ao certificado final. A esse processo de validação em cadeia é dado o nome de validação do caminho do certificado. (CARVALHO H, 2011)

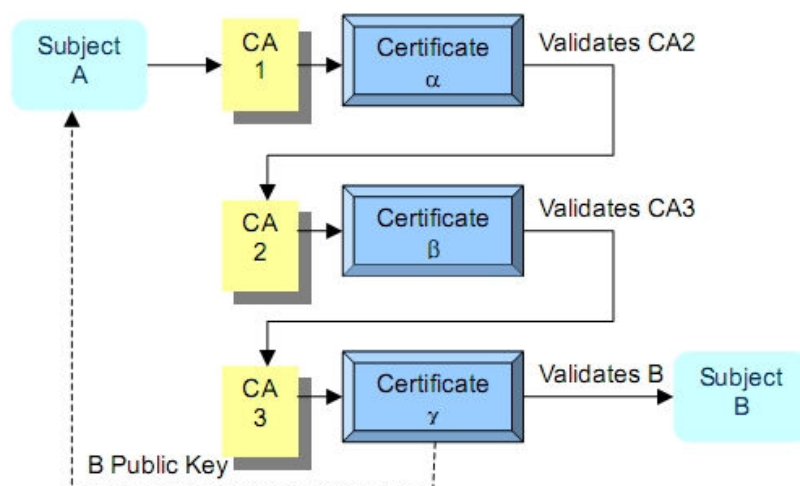


Figura 71 – Caminhos de Certificação (Binder, J., C.)

No sistema ePad, o processo de validação do caminho do certificado é disparado juntamente com a da assinatura. Isso ocorre quando o usuário clica no botão assinar presente no *applet* de assinatura.

Figura 72 – ePad – Assinando o Documento

7.7.4.1. Recuperando o Certificado Âncora

O processo de validação do caminho do certificado pressupõe a existência do certificado âncora da cadeia, pois a validação da cadeia de certificado inicia com os certificados de uma AC Âncora Confiável. A AC não precisa ser necessariamente

uma AC Raiz, bastando fazer parte de qualquer um dos níveis de AC da cadeia e ser reconhecida como confiável (RFC 5280, 2008)

Para que uma AC seja considerada uma âncora confiável, é preciso que ela tenha sido previamente cadastrada no sistema que realizará a validação da cadeia de certificados. O seu cadastro, por sua vez, é realizado por meio da importação do certificado digital da AC para dentro do sistema. Isso pode ser facilmente constatado em browsers como o Firefox e em sistemas operacionais como o *Microsoft Windows* que oferecem suporte à certificação digital.

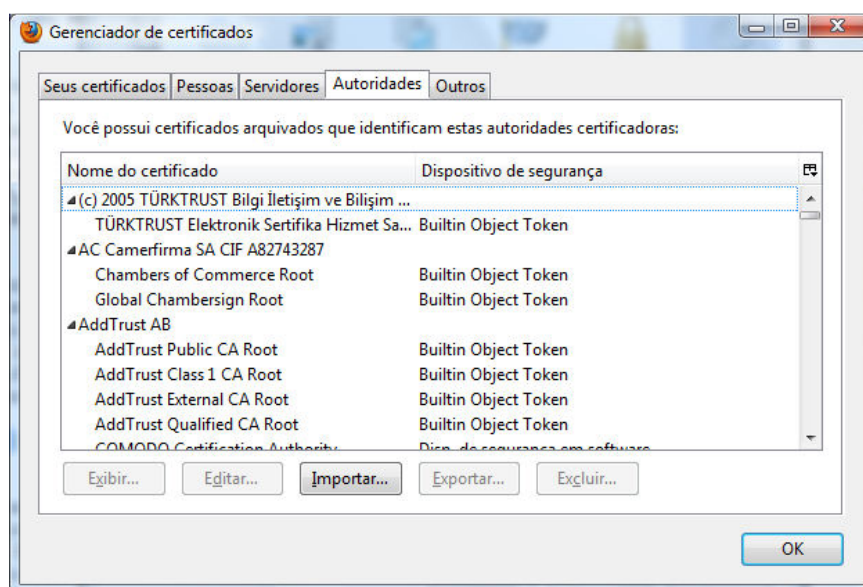


Figura 73 – Mozilla Firefox –Autoridades Certificadoras Confiáveis

No sistema ePad, existe um repositório de certificados digital de AC Confiáveis. Essa base é alimentada e gerenciada pelo Administrador do sistema (ver 6.1.2). Só que para podermos localizar o certificado da AC correta é preciso primeiro identificar qual é o primeiro certificado da cadeia para que possamos extrair seu AKID e DN para, então, localizarmos o certificado no repositório de certificados confiáveis. O primeiro certificado da cadeia pode ser um certificado auto-assinado de uma AC Raiz, um certificado pertencente a uma AC Intermediária ou mesmo o certificado final do usuário, neste último caso, quando o *token* criptográfico não suporta ou não foi configurado para armazenar a cadeia de certificados.

O DN é composto por um conjunto de atributos separados por vírgulas e estruturado de forma hierárquica. Ele é o nome que identifica unicamente uma entrada em um diretório sob o padrão X.500. Do mesmo modo, o DN deve identificar de forma única um certificado assinado por uma CA. Contudo, os certificados podem

ser reemitidos com o mesmo DN, mas com um novo par de chaves. Dessa forma, a única maneira de diferenciar os vários certificados com o mesmo DN é por meio do atributo *subject key identifier* (SKID). O SKID, por sua vez, é uma extensão que provê um meio de identificar certificados que contém determinada chave pública. O valor do SKID deriva, em geral, da própria chave pública contida no certificado do titular.

O AKID é uma extensão que provê um meio de identificação da chave pública associada à chave privada utilizada na assinatura do certificado. Esta extensão é utilizada quando um emissor (AC) tem múltiplas chaves de assinatura (também devido a múltiplos pares de chaves concorrentes ou a troca de chaves). A identificação PODE ser baseada no *keyIdentifier* (identificador de chave) ou número de série e nome do emissor. (IETF, RFC 5280, 2008, tradução nossa)

O campo *keyIdentifier* da extensão *authorityKeyIdentifier* DEVE ser incluído em todos os certificados gerados pelas ACs a fim de facilitar a construção do caminho de certificação. Contudo, existe uma exceção: quando a AC distribui sua chave pública em forma de certificado auto assinado, o AKID PODE ser omitido. A assinatura presente no certificado auto assinado é gerada com a chave privada associada ao campo *subject public key* do certificado (isto prova que o emissor possui ambas as chaves privada e pública). Neste caso, as extensões *subject key identifier* (SKID) e o *authority key identifier* seriam idênticas, mas somente o SKID seria necessário à construção do caminho da certificação. O valor do campo *keyIdentifier* DEVE ser derivado da chave pública utilizada para verificar a assinatura do certificado ou de um método que seja capaz de gerar um valor único. (IETF, RFC 5280, 2008, tradução nossa)

De posse do AKID e do DN, o *applet* realiza uma busca no repositório por meio de uma conexão HTTP. Para isso, utiliza-se uma instância com o nome “con” da classe *URLConnection* a qual provê as propriedades e métodos necessários para estabelecer uma conexão com o servidor web que mantém acesso ao repositório de ACs confiáveis.

```

156 private void loadRemoteTrustedAnchor(X509Certificate lastCertificate)
157     throws Exception {
158     try {
159         String dn = URLEncoder.encode(
160             lastCertificate.getIssuerX500Principal().getName("RFC1779"),
161             CHARSET);
162         String aki_encode;
163         if (getAuthorityKeyId(lastCertificate) == null) {

```

```

163     aki_encode = null;
164 } else {
165     aki_encode = URLEncoder.encode( new String(
166         Base64.encode(getAuthorityKeyId(
167             lastCertificate))), CHARSET);
168 }
169
170 HttpURLConnection con = null;
171 con = (HttpURLConnection) new URL(getURL()).openConnection();
172 con.setDoOutput(true);
173 con.setDefaultUseCaches(false);
174 OutputStreamWriter output = new OutputStreamWriter(
175     con.getOutputStream());
176 output.write("dn=" + dn);
177 if (aki_encode != null) {
178     output.write("&skid=" + aki_encode);
179 }
180
181 output.flush();

```

Figura 74 – ePad - Método *loadRemoteTrustedAnchor* da classe *TrustedAnchorParser*

Ao iniciar a conexão, o objeto “con” envia uma requisição com os parâmetros AKID e DN para o servidor web. Esse, por sua vez, lê os parâmetros recebidos por meio da requisição HTTP e executa a busca no repositório. Caso encontre algum certificado que possua os mesmos valores de AKID e DN, o servidor web envia-o na resposta à requisição e o *applet* processa-o criando um novo objeto do tipo *X509Certificate*, caso contrário, o servidor enviará uma resposta vazia (corpo da mensagem) e o *applet* interrompe o processo de assinatura informando ao usuário o motivo.

7.7.4.2. Recuperando as Listas de Certificados Revogados

Para validar a cadeia de certificados, é preciso primeiro possuir as listas de certificados revogados relativas aos ACs participantes da cadeia. Para cada AC, deverá haver uma lista informando quais certificados foram revogados por ela.

O processo de recuperação das LCRs é feito por meio de uma estrutura de iteração em que cada iteração está associada a um certificado da cadeia. Cada iteração tentará recuperar a LCR associada ao respectivo certificado e, para isso, deverá identificar no próprio certificado qual o caminho do repositório onde se encontra a LCR. O *applet* busca no certificado a *Uniform Resource Locator* (URL) que aponta para a lista de certificados revogados emitida pela AC que o emitiu. A URL deve ser informada no campo de extensão *cRLDistributionPoints*.

A extensão *cRLDistributionPoints* identifica onde a LCR pode ser obtida. Ela não é considerada crítica pela RFC 5280, mas a própria RFC 5280 recomenda o uso

desta extensão pelas ACs e aplicações. A extensão *cRLDistributionPoints* é uma sequência do tipo *DistributionPoint*, que por sua vez é formada por três campos: *distributionPoint*, *reasons* e *cRLIssuer*, sendo cada um desses opcional. O campo *distributionPoint* é o responsável por armazenar as informações relativas aos mecanismos que dão acesso ao local de armazenamento da LCR do certificado. (RFC, 5280).

A leitura do valor da extensão é feita por meio do método *getExtensionValue()* a partir de uma instância da classe *X509Certificate* que representa o certificado digital durante a execução da iteração. Ao chamar o método, é passado como parâmetro a *string* "2.5.29.31" a qual representa o OID da extensão *cRLDistributionPoints*.

Após a obtenção do valor, o applet inicia a recuperação da LCR, uma vez recuperada, converte os dados obtidos em um objeto do tipo *X509CRL* e, em seguida, armazena-o em um *array* de LCRs juntamente com as demais. Ao final, esse conjunto de LCRs será utilizado para realizar a validação do caminho de certificados. Caso não seja possível recuperar alguma LCR, o *applet* lança uma exceção e interrompe o processo de assinatura.

7.7.4.3. Validando o caminho do certificado

O objetivo primário da validação do caminho da certificação é verificar a relação entre o DN e a chave pública do titular, representadas no certificado alvo da validação, com o uso da chave pública contida no certificado âncora. (RFC 5280, 2008). O processo de validação implica também a utilização de um conjunto de políticas e restrições de certificados, que são válidas para o caminho da certificação, baseadas em políticas de extensão do certificado, mapeamentos da política de extensão e restrições de políticas de extensão. (RFC 5280, 2008).

Um exemplo de restrição é a sequência *BasicConstraints* a qual define se o titular do certificado é uma AC e a qual a profundidade máxima do caminho da certificação do certificado que está sendo validado. Com base nessa restrição, um certificado em que o titular não é uma AC não poderá assinar outros certificados atuando como uma AC intermediária. Da mesma forma, caso a profundidade máxima permitida seja 'X' e haja um nível 'X + Y', sendo 'Y > 0', no caminho da certificação, então as restrições básicas não foram respeitadas e o caminho é inválido.

De posse da cadeia de certificados e do conjunto de LCRs correspondente, o applet inicia o processo de validação do caminho do certificado. Ele verifica se as restrições e políticas foram respeitadas e se algum certificado está contido em uma LCR. Para a realização dessas atividades, o *applet* cria uma instância da classe *CertPathValidator* e insere nela todos os parâmetros necessários à validação, como: certificado âncora, cadeia de certificados e lista de certificados revogados; por meio do método *validate*, o qual realiza a validação do caminho da certificação. Caso algum certificado esteja em desacordo com as restrições, esteja expirado ou tenha sido revogado, o método *validate* lança uma exceção e o *applet* interrompe o processo de assinatura.

7.7.5. Assinando o documento

Após a validação do caminho da certificação, dá-se início à geração da assinatura digital. A sintaxe padrão de assinatura adotada é o *Cryptographic Message Syntax* (CMS) estabelecido pela RFC 3852. A CMS deriva do padrão PKCS#7 versão 1.5, que foi publicado originalmente pelos Laboratórios RSA em Novembro de 1993, e é uma sintaxe utilizada para assinar digitalmente, autenticar ou mesmo criptografar conteúdos arbitrários de mensagens. (RFC 3852, 2004)

Criada para descrever uma sintaxe de encapsulamento para proteção de dados, a CMS suporta tanto assinaturas digitais quanto criptografia. A sintaxe permite múltiplos encapsulamentos; um envelope de encapsulamento podendo ser aninhado dentro de outro. Ela permite também a adição de diversos atributos, tais como data e hora da assinatura, que devem ser assinados juntamente com o conteúdo da mensagem, além de prover outros atributos, tais como contadores de assinatura, para serem associados à própria assinatura digital. Os valores gerados com base na CMS seguem a notação *Abstract Syntax Notation One* (ASN.1), com o uso da codificação *BER-Encoding*, sendo tipicamente representados por cadeias de octetos (octet strings). (RFC 3852, 2004)

A sintaxe CMS suporta diversos tipos de conteúdo, contudo, a RFC 3852 define apenas seis tipos: *data*, *signed-data*, *enveloped-data*, *digest-data*, *encrypted-data* e *authenticated-data*. A RFC por sua vez não impõe restrições à criação de outros tipos, informando que outros tipos de conteúdo podem ser definidos fora do escopo da própria RFC 3852. (RFC 3852, 2004)

Neste aplicativo, o tipo de conteúdo que será utilizado é o *signed-data*, que é próprio para assinatura de dados. O *signed-data* consiste de um conteúdo de qualquer tipo e de zero ou mais assinaturas. Ele permite também que qualquer número de signatários assinem o conteúdo da mensagem em paralelo. (RFC 3852, 2004)

O processo no qual um tipo *signed-data* é construído envolve os seguintes passos:

- Para cada signatário, um resumo ou *hash* é computado com base no conteúdo do documento ou mensagem utilizando algoritmo de *hash* próprio para assinatura;
- Para cada signatário, o *hash* da mensagem é assinado utilizando a respectiva chave privada do signatário;
- Para cada signatário, o *hash* assinado e as outras informações do assinante são coletadas e inseridas dentro de um objeto do tipo *SignerInfo*³². Certificados e LCRs de cada signatário, bem como aqueles que não correspondem a qualquer signatário, são coletados neste passo;
- Os algoritmos de *hash* e os objetos *SignerInfo* de todos os signatários são coletados juntamente com o conteúdo do documento ou mensagem dentro de um *Signed-Data*.

Para gerar uma assinatura sob o padrão CMS, primeiramente o *applet* encapsula o conteúdo do documento dentro de um objeto do tipo *CMSProcessable*. Em seguida, o *applet* encapsula a cadeia de certificados juntamente com as LCRs dentro de um objeto do tipo *CertStore* com base no *provider Bouncy Castle*.

```

70 public byte[] signBuffer(byte[] buffer, boolean encapsulate, String
71     provider) throws Exception {
72     try {
73         ArrayList certList = new ArrayList();
74         CMSProcessable msg = new CMSProcessableByteArray(buffer);
75         CertStore certs = CertStore.getInstance("Collection", new
76             CollectionCertStoreParameters(certList), "BC");

```

Figura 75 – ePad - Criação dos objetos CMSProcessable e CertStore

Após o encapsulamento do documento, da cadeia de certificados e das LCRs em objetos gerenciáveis, o *applet* cria uma instância da classe *CMSSignedDataGenerator* que será a responsável pela geração de assinatura no

³² Tipo de sequência (ASN.1) que coleta informações acerca de um signatário a fim de construir uma assinatura digital sobre o padrão PKCS#7 Signed-Data. (Oracle, 2011).

formato PKCS#7. Por meio do método *addSigner*, adiciona-se as informações referentes aos signatários do documento, como certificado, chave privada e algoritmo de assinatura. Ressalta-se novamente que, quando utilizado um dispositivo criptográfico, na passagem da chave privada é informada somente uma referência e não a própria chave, outra informação relevante é que o algoritmo de assinatura adotado está diretamente relacionado ao algoritmo de geração da chave privada, não sendo possível escolher um outro algoritmo em seu lugar.

Com as informações dos signatários incluídas, adiciona-se as cadeias de certificados e listas de certificados revogados à instância da classe *CMSSignedDataGenerator* e gera-se a assinatura com a chamada do método *generate* passando como parâmetro o documento e o *provider* que será utilizado – neste caso o *Bouncy Castle*. É também na chamada deste método que se decide pela opção de gerar a assinatura contendo ou não o documento encapsulado. Para fins de otimização de armazenamento, decidiu-se por não adicionar o documento à assinatura. Ao final, o método retornará um objeto do tipo *CMSSignedData* contendo somente a assinatura.

Por último, recupera-se a assinatura por meio do método *getEncoded* e a armazena em um objeto do tipo *byte[]* para ser posteriormente enviada juntamente com o documento ao repositório do sistema ePad.

```

89 CMSSignedDataGenerator gen = new CMSSignedDataGenerator();
90
91 if (origPK.getAlgorithm().equals("RSA")) {
92     gen.addSigner(origPK, origCert, "1.2.840.113549.1.1.5");
93 } else if (origPK.getAlgorithm().equals("DSA")) {
94     gen.addSigner(origPK, origCert,
95 CMSSignedDataGenerator.ENCRYPTION_DSA );
96 } else {
97     throw new Exception("Algoritmo de assinatura não suportado!\n");
98 }
99
100
101 gen.addCertificatesAndCRLs(certs);
102
103 CMSSignedData s = gen.generate(msg, encapsulate, provider);
104
105 this.assinatura = s.getEncoded();

```

Figura 76 – ePad - Assinando o documento

7.7.6. Enviando o documento e a assinatura

Com o documento e a assinatura representados por vetores de *bytes*, o *applet* os copia para variáveis contidas em um formulário HTML. O usuário ao clicar

no botão “Salvar” submete o formulário enviando os dados para o repositório do Sistema ePad.

A cópia dos valores do documento e assinatura para o formulário é feita por meio de chamadas a funções *javascript* a partir do próprio *applet*. A chamada é realizada por meio do método *showDocument* da classe *AppletContext* onde é passada como parâmetro uma instância da classe *URL* contendo o nome da função *javascript* a ser chamada. A função *javascript* por sua vez acessa o *applet* e copia os valores para as variáveis presentes no formulário HTML.

Percebe-se que existe um alto acoplamento entre o *applet* e as funções *javascript* de cópia de documento e assinatura, o que não é tido como uma boa prática de programação. O motivo da implementação ter sido mantida desta forma é por não ter sido achado um meio mais “elegante” de copiar valores contidos em *applets* para formulários HTML. É certo que nas próximas versões uma melhor abordagem terá sido desenvolvida e aplicada.

7.7.7. Validação do documento e assinatura no servidor

O servidor, ao receber o documento e a assinatura, verifica se a assinatura corresponde ao documento apresentado. Este processo é executado com o intuito de prevenir que usuários mal intencionados submetam assinaturas inválidas ou que não correspondam ao documento apresentado.

A validação da assinatura é feita por meio da conversão do documento e assinatura em um objeto do tipo *CMSSignedData*, sendo que desta vez o documento é encapsulado juntamente com a assinatura em um mesmo objeto. Constrói-se então uma estrutura de iteração onde são extraídos os dados de cada signatário, armazenando-os em objetos do tipo *SignerInformation*. Para cada objeto criado é chamado o método *verify* que valida a assinatura do signatário. Caso a assinatura não seja válida, o método retorna *false* e o sistema lança uma exceção interrompendo o processo. Caso a assinatura seja válida, o método retorna *true* e o sistema continua sua execução normalmente.

8. CONCLUSÃO

O uso de uma ferramenta de gerenciamento eletrônico de documentos e processos tornou possível a manipulação informações contidas nas organizações que antes não tinham uma maneira organizada de serem armazenadas e consultadas, graças ao conceito complexo de documento e processo definido nesta nova tecnologia.

O sistema foi desenvolvido utilizando-se de boas práticas de desenvolvimento de software, de técnicas de engenharia de software padronizadas e de manutenção relativamente fácil, o que possibilita a adição de novos recursos e funcionalidades, como: novos modelos de tokens, utilização de certificados válidos, etc. Este trabalho teve seu mérito no que diz respeito à aquisição de conhecimentos com relação ao gerenciamento eletrônico de processos documentos com uso de assinatura digital e à disponibilização desses conhecimentos.

Ressaltamos ainda que este trabalho poderia ser expandido, englobando as etapas de um projeto de implantação de um sistema de GED com workflow, logo após a realização de um Estudo de Caso. Os referidos tópicos não foram abordados devido ao custo operacional, tempo disponível para sua realização e aos limites do escopo proposto no início do desenvolvimento do trabalho.

8.1. Considerações Finais

O método de assinatura digital com uso de certificação digital representa uma técnica de excelência que garante os requisitos básicos da validade jurídica dos documentos eletrônicos, pois a sociedade fundamenta-se em parâmetros para nortear suas relações, como, por exemplo, o amparo legal, as mudanças sociais, novas tecnologias ou fatos jurídicos, que devem ser seguidos pela regulamentação das leis.

A implantação de um sistema dessa natureza implica um processo de mudança em uma organização para atender às novas condições de trabalho sendo imprescindível o treinamento do quadro de pessoal no sentido de se adequarem às novas exigências. Com a automatização dos procedimentos da instituição e para

garantir uma efetiva melhora no desempenho surge a necessidade de se estabelecer rotinas e procedimentos de backups das informações contidas no sistema, uma vez que esse sistema passa a ser um serviço crítico para a instituição com o aumento da dependência das tecnologias de computação.

Outra demanda que atenderá é a possibilidade de acompanhamento dos processos a qualquer tempo, de forma a atender os requisitos de disponibilidade e publicidade dos atos e informações das entidades públicas e a otimização do fluxo de processos a partir da coleta de informações disponíveis na ferramenta.

REFERÊNCIAS BIBLIOGRÁFICAS

ADVOGADOS, Carneiro e Candido. **A informatização do processo judicial no Brasil**. Disponível em:

<http://www.carneiroecandido.adv.br/site_index.php?tipo_pag=7&id_menu=2301&id_artigo=2378>. Acesso em: 30 mar. 2011.

APACHE. **Commons Email**. Disponível em: <<http://commons.apache.org/email/>>. Acesso em: 12 abr. 2011.

ATHENIENSE, Alexandre Rodrigues. **As controvérsias do peticionamento eletrônico após a Lei 11419/2006**. Conteudo Juridico, Brasília-DF: 05 mar. 2009. Disponível em: <<http://www.conteudojuridico.com.br/?artigos&ver=2.23309>>. Acesso em: 30 mar. 2011.

BARRETT, Daniel J., SILVERMAN, Richard E.; BYRNES, Robert G.. **SSH the secure shell: the definitive guide**. O'Reilly Media, Inc., 2005.

BISWAL, Purna Chandra. **Discrete Mathematics and Graph Theory**, 2nd ed. Editora: PHI Learning Pvt. Ltd. 2008.

BISWAS, Rahul. ORT, Ed. **The Java Persistence API - A Simpler Programming Model for Entity Persistence**. Disponível em: <<http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html>>. Acesso em: 29 mar. 2011.

BITTENCOURT BRASIL, Angela. **Assinatura digital**. Disponível em: <<http://jus.uol.com.br/revista/texto/1782/assinatura-digital>>. Acesso em: 01 fev. 2011.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML - Guia do Usuário**. 5ª ed. Rio de Janeiro: Campus, 2005.

BRASIL. Conselho da Justiça Federal. **O que é Assinatura Digital**. Disponível em: <<http://www.jf.jus.br/cjf/tecnologia-da-informacao/identidade-digital/o-que-e-assinatura-digital>>. Acesso em: 01 fev. 2011.

BRASIL. Presidência da República. **Lei nº 11.419, de 19 de dezembro de 2006**. Dispõe sobre a informatização do processo judicial; altera a Lei no 5.869, de 11 de janeiro de 1973 – Código de Processo Civil; e dá outras providências.

_____. **Medida Provisória nº 2.200-2**. Institui a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, transforma o Instituto Nacional de Tecnologia da Informação em autarquia, e dá outras providências. Disponível em: <http://www.planalto.gov.br/ccivil_03/mpv/Antigas_2001/2200-2.htm>. Acesso em: 01 abr. 2011.

CALACHE, Matheus. **JSF**. Disponível em: <<http://matheuscalache.blogspot.com/2009/10/jsf-definicao.html>>. Acesso em: 26 mar. 2011.

CARVALHO, Hugo. **Certificação - PKI - Infra-estrutura de Chaves Públicas**. Disponível em: <https://www.gta.ufrj.br/ensino/eel879/Anos-anteriores/2008-2/trabalhos_vf/hugo/Certificacao.html>. Acesso em: 12 fev. 2011.

CARVALHO, Joao Antonio. **Informática Para Concursos**. 4ª ed. 2009. CAMPUS.

CHESWICK, William R.; BELLOVIN, Steven M.; RUBIN, Aviel D.. **Firewalls e Segurança na Internet: Repelindo O Hacker Ardiloso**. 2º Edição. Editora Bookman. 2003.

COULOURIS, George, DOLLIMORE, Jean, Kindberg, TIM. **Sistemas Distribuídos - Conceitos e Projeto**. Bookman - 784 páginas.

DEITEL, Harvey M.; DEITEL, Paul J. **Java: Como Programar**. 6ª Ed. São Paulo: Pearson Hall, 2005.

DEVMEDIA. **Java**. Disponível em: <<http://javafree.uol.com.br/wiki/Java>>. Acesso em: 25 mar. 2011.

FERREIRA, Aurélio B. de Holanda. **Novo Dicionário da Língua Portuguesa**. 7ª ed. Rio de Janeiro: Positivo, 2009.

FOROUZAN, Behrouz A. **Comunicação de dados e redes de computadores**. 3. ed. Porto Alegre: Bookman, 2006.

FRANCIS, Eder. **Aprendendo sobre MER**. Disponível em: <<http://www.eia.com.br/wp/banco-de-dados/aprendendo-sobre-mer-modelo-de-entidade-e-relacionamento-e-der-diagrama-de-entidade-e-relacionamento/>>. Acesso em: 20/04/2011.

GALLO, André F.; GOMES, Flávio A.; LUIZ, Nelson B.; CUNH, Wagner. **Princípios da Engenharia de Software**. Universidade do Sul de Santa Catarina. - UNISUL. Santa Catarina. 2011.

GLOBALCODE. **PrimeFaces com JSF2 no ScrumToys**. Disponível em: <<http://blog.globalcode.com.br/2010/03/primefaces-com-jsf2-no-scrumtoys.html>>. Acesso em: 27/03/2011.

IETF - Internet Engineering Task Force. **Cryptographic Message Syntax (CMS)**. Disponível em: <<http://www.ietf.org/rfc/rfc3852.txt>>. Acesso em: 12 mar. 2011.

_____. **Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. Disponível em: <<http://www.ietf.org/rfc/rfc5280.txt>>. Acesso em: 14 fev. 2011.

IRIB – Instituto de Registro Imobiliário do Brasil. **O que é Certificação Digital**. Disponível em: <<https://www.oficioeletronico.com.br/Downloads/CartilhaCertificacaoDigital.pdf?>>. Acesso em: 02 fev. 2011.

JAMES PETERS, MANOJ BATHIA, SATISH KALIDINDI, SUDIPTO MUKHERJEE. **FUNDAMENTOS DE VOIP - 2º Edição**. Editora Bookman. Ano 2008. ISBN 8577801136, 9788577801138. 389 páginas.

JBOSS. **HIBERNATE - Persistência Relacional para Java Idiomático**. Disponível em: <http://docs.jboss.org/hibernate/core/3.5/reference/pt-BR/html_single/>. Acesso em: 04 abr. 2011.

LINHARES, Maurício. **Introdução ao Hibernate 3**. Disponível em: <http://www.guj.com.br/content/articles/hibernate/intruducacao_hibernate3_guj.pdf>. Acesso em: 04 abr. 2011.

MACEDO, José. **Utilizando a biblioteca Primefaces**. Disponível em: <<http://jamacedo.wordpress.com/category/java/java-server-faces/prime-faces/>>. Acesso em: 27 mar. 2011.

MARTINI, Renato da Silveira. **Tecnologia e Cidadania Digital: Tecnologia, Sociedade e Segurança**. 1ª Edição. Editora Brasport. 2008.

MARTINS, Alessandro. **Autoridade Certificadora para Acesso Seguro**. Rio de Janeiro, 2001, 17 p.. Artigo. UFRJ. Disponível em: <<http://www.mat.uc.pt/~pedro/lectivos/CodigosCriptografia/artigo-gazeta08.pdf>>. Acesso em: 29 mar. 2011.

NUNES, Délio. **PKI – Public key infrastructure**. Disponível em: <http://www.gta.ufrj.br/grad/07_2/delio/>. Acesso em: 10 fev. 2011.

ORACLE. **Enum Types**. Disponível em: <<http://download.oracle.com/javase/tutorial/java/javaOO/enum.html>>. Acesso em: 25 mar. 2011.

_____. **Java Security Architecture**. Disponível em: <<http://download.oracle.com/javase/1.4.2/docs/guide/security/spec/security-spec.doc1.html>>. Acesso em: 04 mar. 2011.

_____. **Java™ PKCS#11 Reference Guide**. Disponível em: <<http://download.oracle.com/javase/6/docs/technotes/guides/security/p11guide.html>>. Acesso em: 26 fev. 2011.

_____. **Lesson: Packaging Programs in JAR Files**. Disponível em: <<http://download.oracle.com/javase/tutorial/deployment/jar/>>. Acesso em: 03 mar. 2011.

_____. **Signature File**. Disponível em: <<http://download.oracle.com/javase/1.4.2/docs/guide/jar/jar.html#Signature%20File>>. Acesso em: 03 mar. 2011.

_____. **Signing JAR Files**. Disponível em: <<http://java.sun.com/developer/Books/javaprogramming/JAR/sign/signing.html>>. Acesso em: 03 mar. 2011.

_____. **Understanding the Manifest**. Disponível em:
<<http://java.sun.com/developer/Books/javaprogramming/JAR/basics/manifest.html>>.
Acesso em: 02 mar. 2011.

PAIVA, Marcelo. **Português Jurídico**. 7ª ed. Brasília: Edurece, 2010.

PIETRO, Maria Sylvia Zanella Di. **Direito Administrativo**. 3ª ed, São Paulo: Atlas, 1999.

POSTGRESQLBR. **Sobre o PostgreSQL**. Disponível em:
<<http://www.postgresql.org.br/sobre>>. Acesso em: 07 abr. 2011.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª ed. São Paulo: MCGRAW-Hill, 2006.

PRIMEFACES. **PrimeFaces**. Disponível em: <<http://www.primefaces.org/>>. Acesso em: 27 mar. 2011.

QUARESMA, Pedro; LOPES, Elsa. **Criptografia**. Coimbra, 2007, 8 p. Artigo. F.C.T.U.C; Univerisdade de Coimbra. Disponível em:
<<http://www.mat.uc.pt/~pedro/lectivos/CodigosCriptografia/artigo-gazeta08.pdf>>.
Acesso em: 29 mar. 2011.

REZENDE, Paulo Eduardo. **Documentação de Projetos Web**. Disponível em:
<<http://imasters.com.br/artigo/8568/banco-de-dados/documentacao-de-projetos-web-der>>. Acesso em: 20 mar. 2011.

RSA, Laboratories. **PKCS#11: Cryptographic Token Interface Standard**. Disponível em: <<http://www.rsa.com/rsalabs/node.asp?id=2133>>. Acesso em: 9 abr. 2011.

_____. **PKCS#12: Personal Information Exchange Syntax Standard**. Disponível em: <<http://www.rsa.com/rsalabs/node.asp?id=2138>>. Acesso em: 6 mar. 2011.

_____. **PKCS #15: Cryptographic Token Information Format Standard**. Disponível em: <<http://www.rsa.com/rsalabs/node.asp?id=2141>>. Acesso em: 26 fev. 2011.

_____. **TWIRL and RSA Key Size**. Disponível em: <<http://www.rsa.com/rsalabs/node.asp?id=2004>>. Acesso em: 15 fev. 2011.

SANTOS, Alfredo. **Quem Mexeu No Meu Sistema?**. 1º edição. Editora Brasport, 2008.

SAUVÉ, Jacques. **Java Server Faces**. Disponível em:
<<http://www.dsc.ufcg.edu.br/~jacques/cursos/daca/html/jsf/jsf.htm>>. Acesso em: 26 mar. 2011.

SAWAYA, Marcia Regina. **Dicionário de Informática e Internet: Inglês/Português**. 3ª ed. Editora Nobel. 2009.

SCHEER, Sérgio. **Gerenciamento Eletrônico de Documentos no Processo de Projetos de Edifícios**. Disponível em: <http://www.eesc.usp.br/sap/workshop/anais/GERENCIAMENTO_ELETRONICO_DE_DOCUMENTOS.pdf>. Acesso em: 12 jan. 2011.

SERASA. **O que é um certificado digital?** Disponível em: <<http://loja.certificadodigital.com.br/Serasa/O-que-e-um-certificado-Digital>>. Acesso em: 31 mar. 2011.

SILVA, Flávio Luiz D'Oliveira. **Gerenciamento Eletrônico de Documentos (GED): Natureza, Princípios e Aplicações**. Cuiabá: Universidade Federal de Mato Grosso – UFMT, 2001.

TADANO, Katiucia. **GED – Assinatura digital e validade jurídica de documentos eletrônicos**. Cuiabá: Universidade Federal de Mato Grosso – UFMT, 2002.

TECHTERMS.COM. **API**. Disponível em: <<http://www.techterms.com/definition/api>>. Acesso em: 29 mar. 2011.

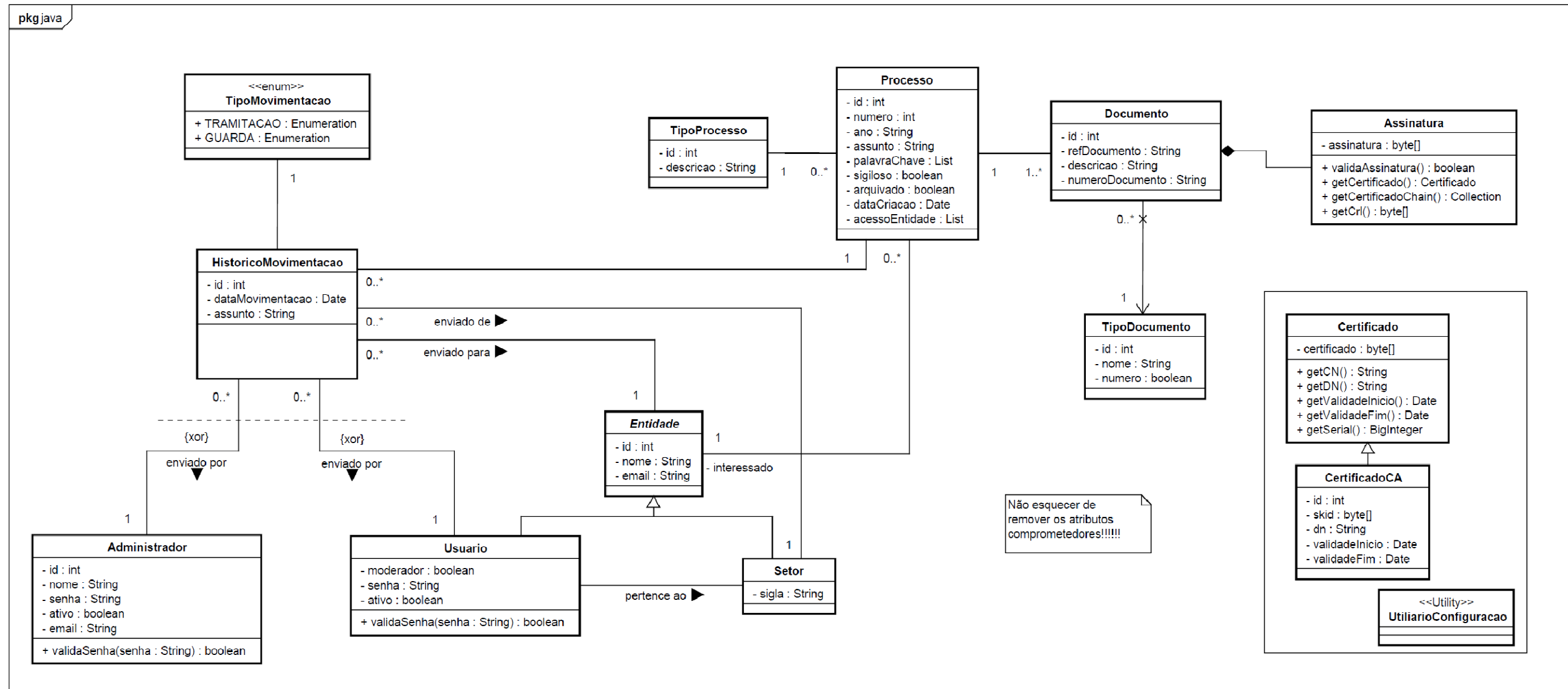
TSCHALÄR. Ronald. **How the Applet Network Security Policy works**. Disponível em: <<http://www.innovation.ch/java/security.html>>. Acesso em: 15 fev. 2011.

UNB. **Código Móvel**. Disponível em: <<http://www.cic.unb.br/~jhcf/MyBooks/ciber/doc-ppt-html/CodigoMovel.html>>. Acesso em: 15 fev. 2011.

WORKFLOW MANAGEMENT COALITION. **The Workflow Reference Model**. Disponível em: <<http://www.wfmc.org>>. Acesso em: 12 jan. 2011.

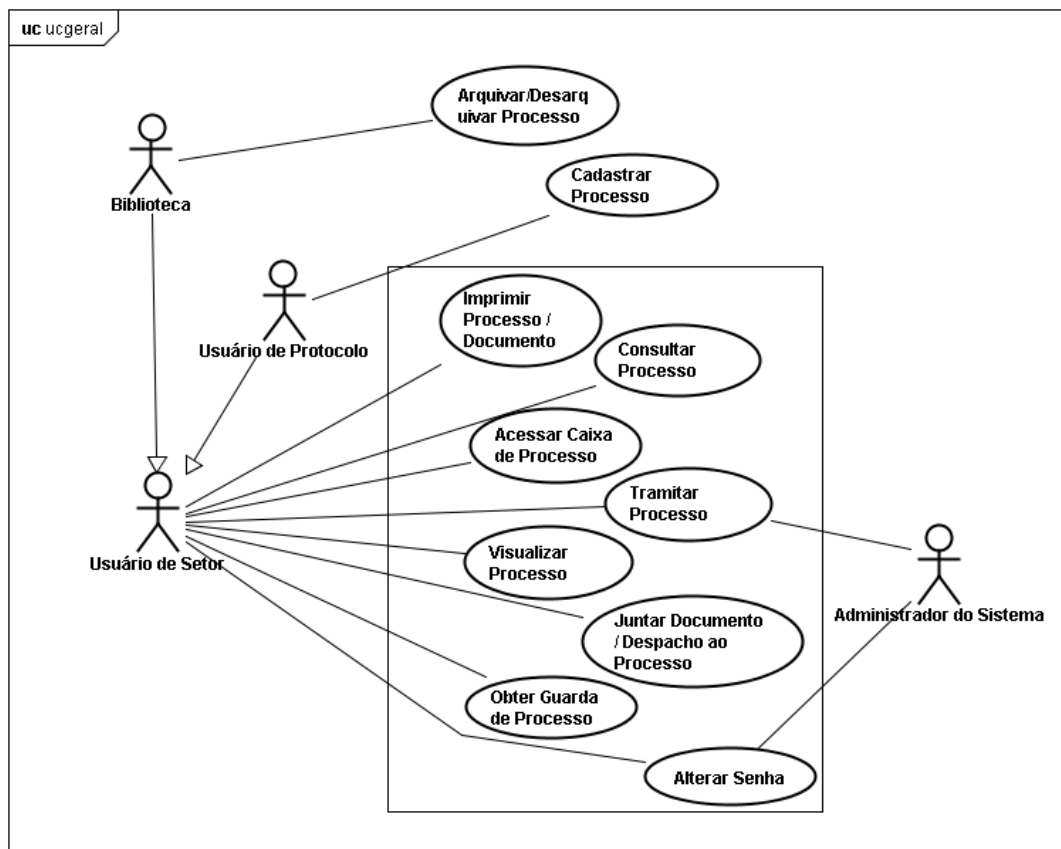
APÊNDICE

Diagramas de Classes

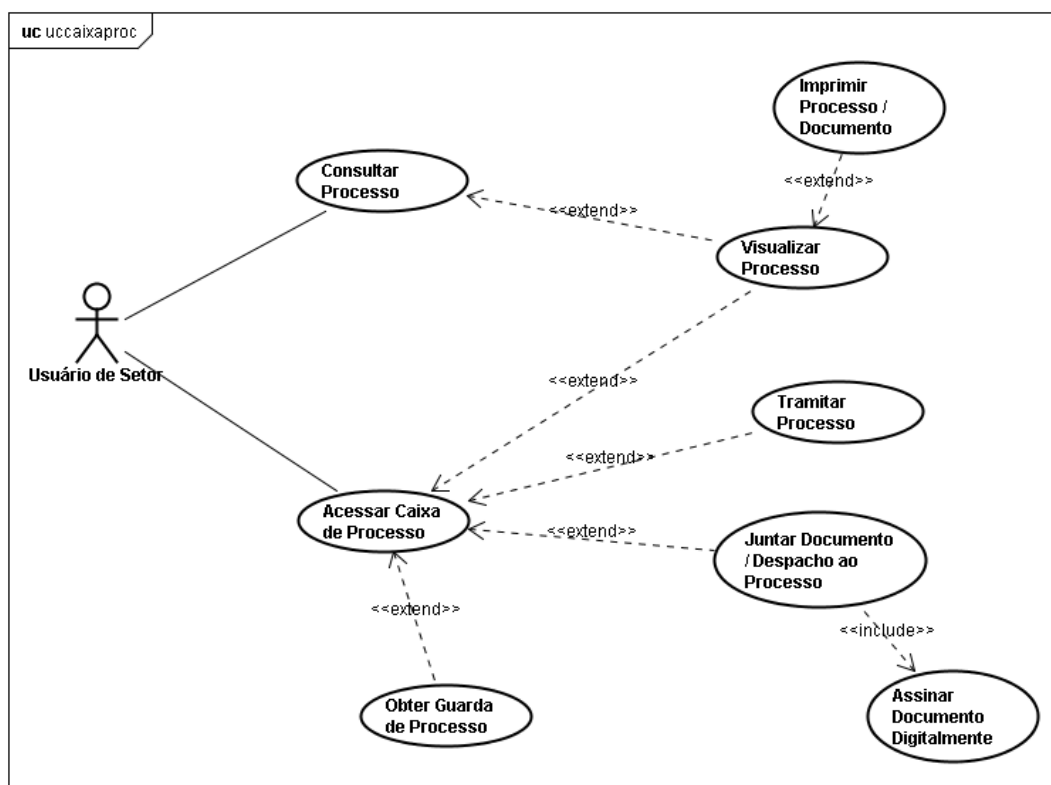


Diagramas de Caso de Uso

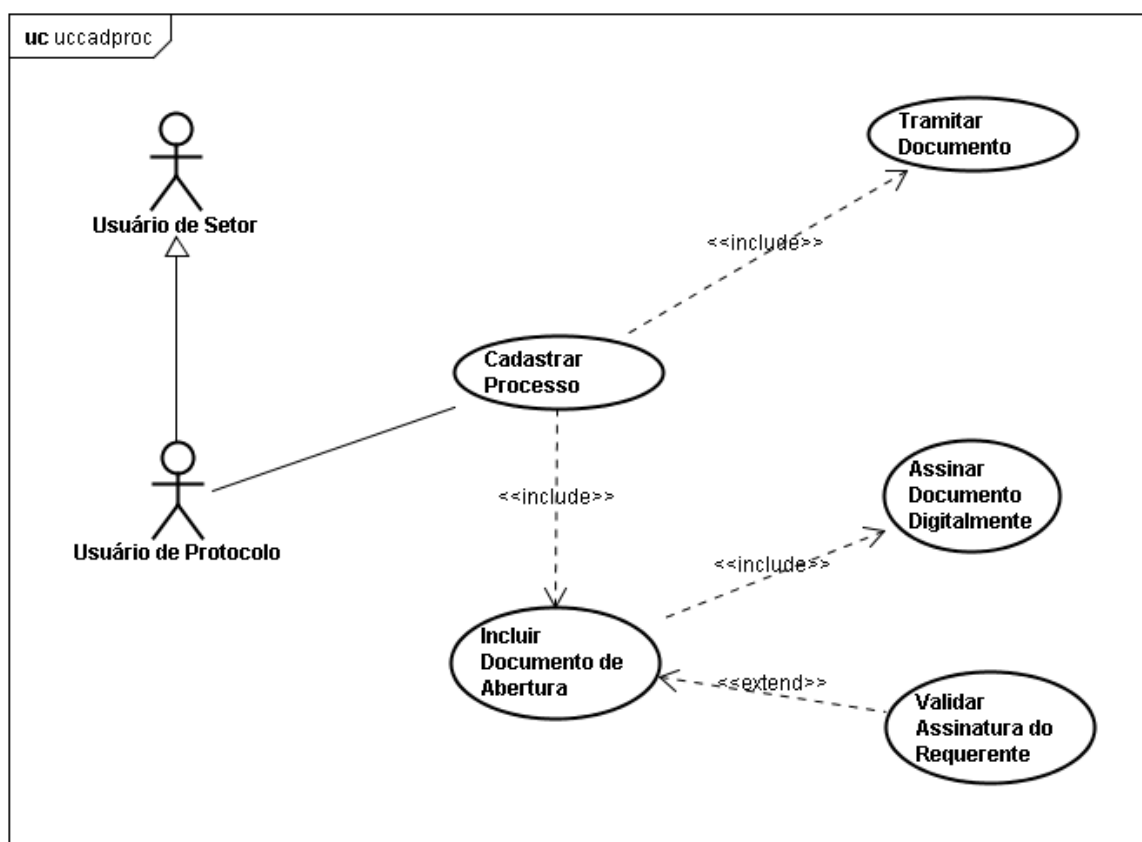
Caso de Uso - Geral



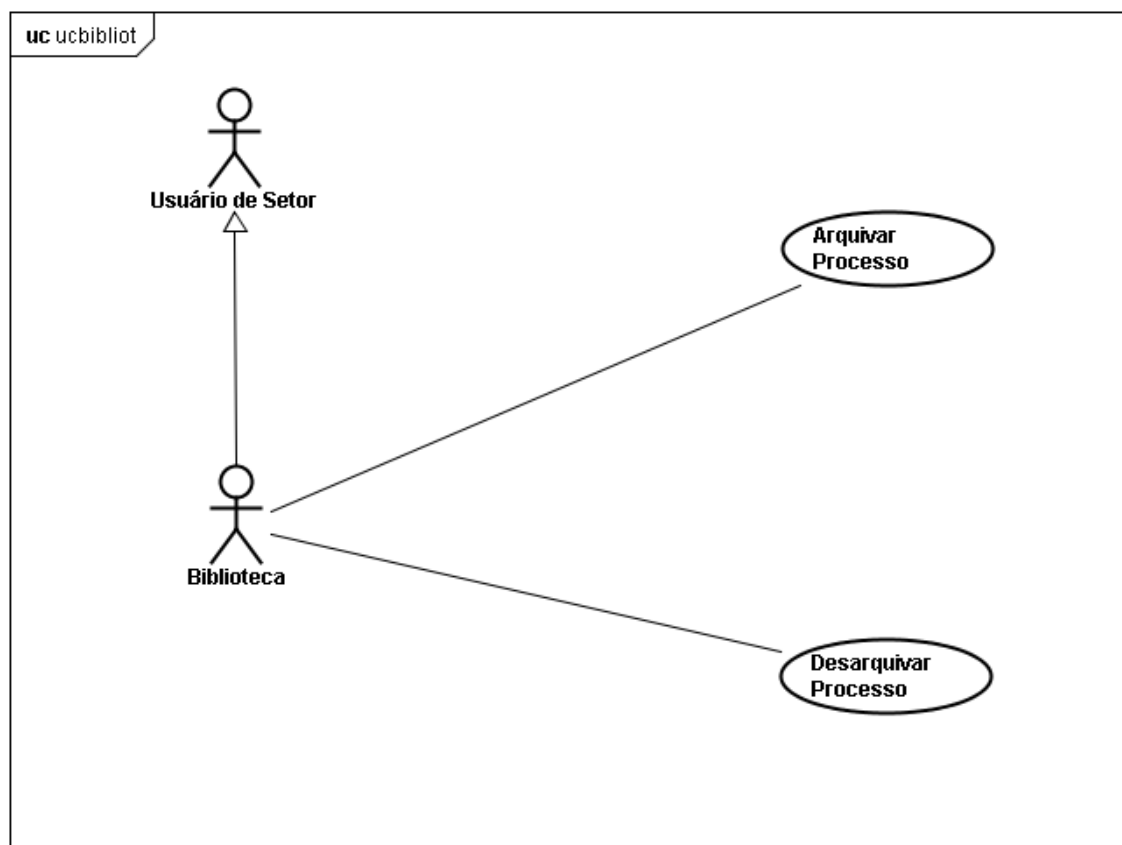
Caso de Uso – Caixa de Processo



Caso de Uso – Cadastro de Processo



Caso de Uso - Biblioteca



Caso de Uso - Administrador

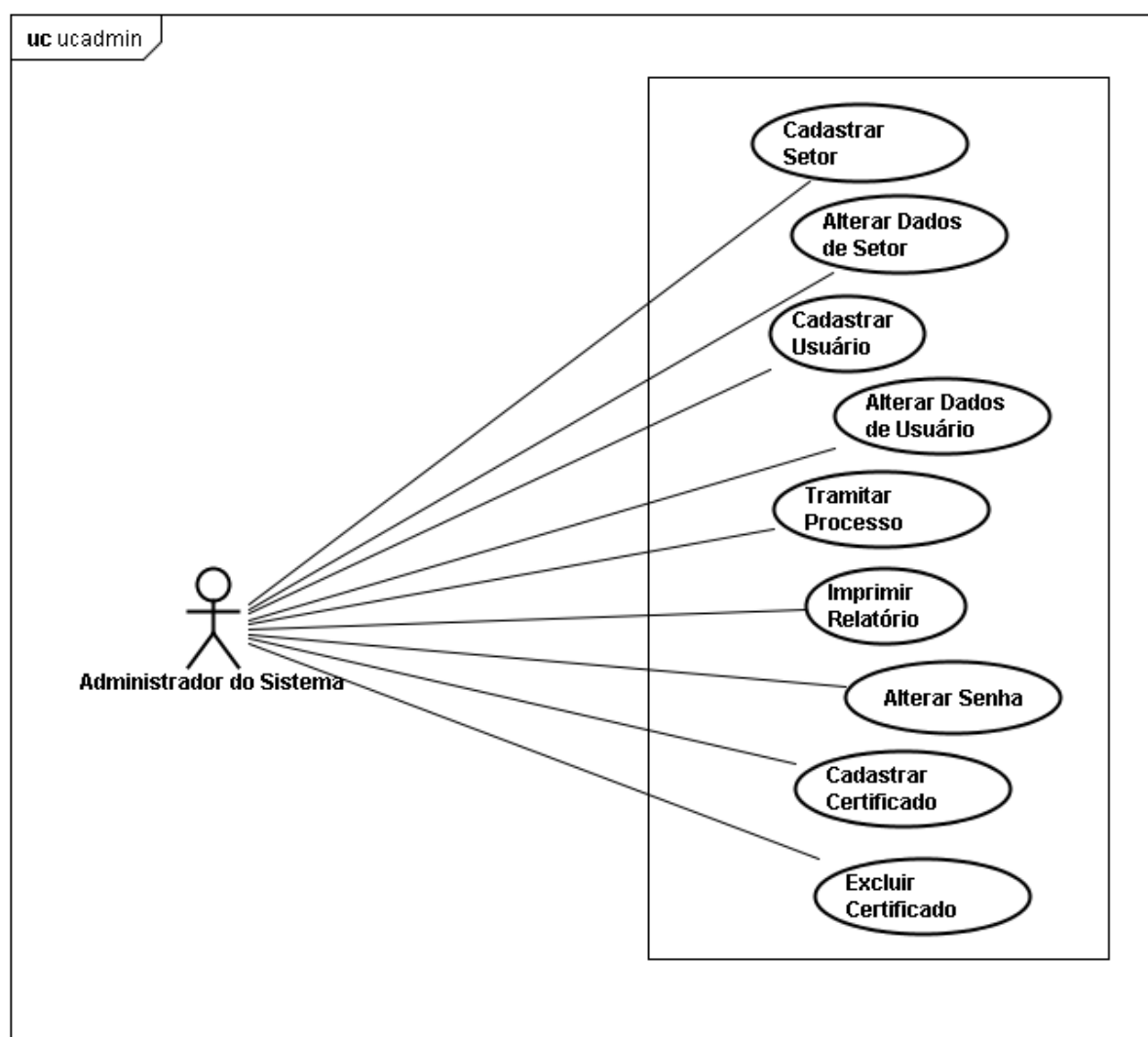


Diagrama de Sequência do Processo de Assinatura

