UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

MIDDLEWARE PARA FORNECIMENTO DE SERVIÇO DE SEGURANÇA EM CONFORMIDADE COM A ISO/IEC 15.408

BRUNO CASTELLANI GUCOWSKI

BLUMENAU
2011

2011/1-08

BRUNO CASTELLANI GUCOWSKI

MIDDLEWARE PARA FORNECIMENTO DE SERVIÇO DE SEGURANÇA EM CONFORMIDADE COM A ISO/IEC 15.408

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

Prof. Paulo Fernando da Silva, Ms. - Orientador

BLUMENAU
2011

2011/1-08

MIDDLEWARE PARA FORNECIMENTO DE SERVIÇO DE SEGURANÇA EM CONFORMIDADE COM A ISO/IEC 15.408

Por

BRUNO CASTELLANI GUCOWSKI

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

Presidente:	Prof. Paulo Fernando da Silva, Ms. - Orientador, FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB
Membro:	Prof. Francisco Adell Péricas, M.Sc – FURB

Blumenau,07 de julho de 2011

Dedico este trabalho aos meus pais, Luciano e Jane, a minha irmã Nathália e a minha esposa Lolari e a todos os amigos que colaboraram para sua realização.

AGRADECIMENTOS

À minha família, que mesmo longe, sempre esteve presente.

Aos meus amigos, pelas cobranças, incentivos e momentos de alegria vividos.

A empresa Wheb Sistemas, por disponibilizar espaço, tempo para realização deste trabalho.

Ao meu orientador, Paulo Fernando da Silva, por ter acreditado no meu potencial e na minha proposta.

A todos professores que contribuíram para minha formação acadêmica.

Julgue seu sucesso pelas coisas que você teve que renunciar para conseguir.

Dalai Lama

RESUMO

Este trabalho descreve o desenvolvimento de um *middleware* para auxiliar no desenvolvimento de softwares seguros em Java. Baseando-se na norma *International Organization for Standardization / International Electrotechnical Commission* (ISO/IEC) 15.408, conhecida pelo nome *Common Criteria* onde são descritos os aspectos de segurança da informação. Com objetivo de facilitar a criação de softwares e abstrair os conceitos de segurança, o *middleware* implementa algumas classes da norma ISO/IEC 15.408, como é o caso das classes autenticação, proteção de dados do usuário, criptografia, auditoria, acesso ao sistema e gerenciamento de segurança. Para comunicação com o *middleware* foi utilizada a interface Remote *Method Invocation* (RMI), permitindo a chamada remota das rotinas implementadas. Para criptografia foi utilizado o algoritmo criado por Ron **R**ivest, Adi **S**hamir e Leonard **A**dleman (RSA), que através de chaves simétricas e fundamentos em teorias clássicas de números garante a ilegibilidade dos dados.

Palavras-chave: ISO/IEC 15.408. RMI. RSA. Software seguro.

ABSTRACT

This paper describes the development of a middleware to help in developing secure software in Java. Based on standard International Organization for Standardization / International Electrotechnical Commission (ISO / IEC) 15408, known as the Common Criteria are described aspects of information security. Aiming to simplify the creation of software and abstract concepts of security, middleware implements some classes of ISO / IEC 15408, such as class authentication, user data protection, encryption, auditing, access and security management. For communication with the middleware was used to interface Remote Method Invocation (RMI), allowing remote call routines carried out. For encryption was used the algorithm designed by Ron Rivest, Adi Shamir and Leonard Adleman (RSA), that through symmetric keys and grounds in classical theories of numbers ensures the illegibility of the data.

Keywords: ISO / IEC 15408. RMI. RSA. Secure software.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AES – *Advanced Encryption System*

API – *Application Programming Interface*

CC – *Common Criteria*

EUA – Estados Unidos da América

ISO/IEC – *International Organization for Standardization / International Electrotechnical Commission*

JDBC – *Java Database Connectivity*

JDK – *Java Development Kit*

JVM – *Java Virtual Machine*

PP – *Protection Profile*

RMI – *Remote Method Invocation*

RSA – *Ron **R**ivest, Adi **S**hamir e Leonard **A**dleman*

SSE-CMM – *Engineering Capability Maturity Model*

ST – *Security Target*

TOE – *Target of Evaluation*

UC – *Use Case*

SUMÁRIO

1 INTRODUÇÃO

Segundo Cruz (2007), é necessário se preocupar com a segurança desde o desenvolvimento de software, com o objetivo de garantir integridade e confiabilidade da informação. A cada dia está sendo mais necessário o desenvolvimento de softwares com menor probabilidade a falhas e com um nível maior de segurança.

Segundo Paula (2005, p. 10), existem muitos motivos que levam uma organização a proteger as suas informações. Um destes motivos leva em consideração que para criar, encontrar ou armazenar informações custa dinheiro e, portanto sua perda resulta em prejuízo.

A segurança nunca foi hábito dos desenvolvedores de aplicativos, pois os próprios consumidores não se importavam em agregar segurança. Isso significa investir para incluir recursos que não ajudavam nas vendas. Hoje em dia, vários clientes para comprarem um software exigem que o mesmo possua um alto nível segurança (BURNERTT; PAINE, 2002, p. XVII).

Segundo Albuquerque e Ribeiro (2002, p. XI), o desenvolvimento de software é um dos pontos mais críticos para a garantia de segurança da informação em uma empresa. Um processo de desenvolvimento bem fundamentado e estruturado, com atenção específica aos aspectos de segurança, aliado ao uso de ferramentas adequadas e seguras, é a melhor arma que se tem contra os prejuízos decorrentes de falhas de segurança em softwares.

Baseado na *International Organization for Standardization / International Electrotechnical Commission* (ISO/IEC), estudos foram realizados sobre a ISO/IEC 15.408 conhecida pelo nome *Common Criteria for Information Technology Security Evaluation* (COMMON CRITERIA, 2009) e com os resultados deste trabalho foi desenvolvido um *middleware*[1] na área de segurança da informação para auxiliar os desenvolvedores na fabricação do *software*.

[1] *Middleware* – é definido como uma camada de software que possibilita a comunicação entre aplicações distribuídas, tendo por objetivo diminuir a complexidade e heterogeneidade dos diversos sistemas existentes (MACIEL, 2004, p. 3).

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um *middleware* para garantir a segurança da informação e abstração dos aspectos e conceitos dos requisitos da ISO/IEC 15.408 para os desenvolvedores de softwares.

Os objetivos específicos do trabalho são:

a) garantir a auditoria de segurança: tratar a gravação e manutenção das ações realizadas no sistema e, posteriormente, a análise ou visualização destas;
b) garantir a proteção dos dados do usuário: tratar a confiabilidade e a disponibilidade das informações armazenadas no sistema;
c) garantir a identificação e autenticação do usuário;
d) garantir o acesso ao sistema: controlar e gerenciar o acesso e o encerramento de sessões entre o usuário e o sistema;
e) garantir a criptografia dos dados;
f) agilizar o desenvolvimento de softwares seguros.

1.2 ESTRUTURA DO TRABALHO

Apresentado a introdução e os objetivos, no capítulo dois é descrita a fundamentação teórica e as tecnologias utilizadas para a realização do trabalho. Nele são destacados tópicos relacionados aos aspectos de segurança da informação, a norma ISO/IEC 15.408, a chamada de métodos remotos, criptografia de dados e trabalhos correlatos.

No capítulo três é abordado o desenvolvimento do presente trabalho, detalhando a especificação e implementação.

No capítulo quatro discorre sobre as conclusões provenientes do desenvolvimento desse trabalho, bem como as possíveis extensões do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir são descritos os conteúdos pesquisados e as tecnologias utilizadas na comunicação com os métodos remotos e na criptografia dos dados. Na seção 2.1 são apresentados os aspectos de um desenvolvimento seguro. Na seção 2.2 encontram-se alguns conceitos a respeito da norma ISO/IEC 15.408 e explicações sobre suas classes. Na seção 2.3 e 2.4 encontram-se informações sobre RMI e RSA respectivamente. Por fim, a seção 2.5 apresenta os trabalhos correlatos.

2.1 ASPECTOS DE UM DESENVOLVIMENTO SEGURO

Segundo Batista (2007, p. 13), a construção de um software fidedigno é um dos grandes desafios a ser alcançado por todos aqueles que desenvolvem software. Entre as várias questões relacionadas à fidedignidade do software, o desenvolvimento de software seguro apresenta-se como uma área desafiadora e de interesse cada vez maior por parte das empresas e dos pesquisadores.

Ao melhorar a segurança em um software, quer-se que o mesmo resista a ataques deliberados ou acidentais, impedindo a divulgação de dados confidenciais a quem não tem direito, registrando as tentativas de agressão e acidentes e que disponibilize informações que facilitem localizar as brechas de segurança e as faltas na sua implementação. Em suma, um software mais fidedigno quanto aos seus requisitos de segurança.

Para Albuquerque e Ribeiro (2002, p. 1), três aspectos de segurança são considerados principais, os quais são:

a) confidencialidade: garantia de que a informação é acessível somente por usuários autorizados a terem acesso;
b) integridade de dados: visa garantir que a informação não teve seu conteúdo alterado ou, se foi, o foi de uma forma autorizada. O software tem que ser capaz de identificar se a informação foi alterada e impedir a alteração não autorizada;
c) disponibilidade: indica a quantidade de vezes que o software realizou uma determinada tarefa sem falhas sobre o número de vezes que a mesma foi executada.

De acordo com Albuquerque e Ribeiro (2002, p. 2), existem outros aspectos de

segurança que devem ser discutidos, os quais são:

a) autenticação: busca verificar a identidade digital do usuário de um sistema no momento em que ele requisita um acesso em um programa de computador;
b) não repúdio: capacidade do sistema de provar que um usuário executou determinada ação no software;
c) auditoria: capacidade do sistema de auditar tudo o que foi realizado pelos usuários, detectando acessos indevidos ou tentativa dos mesmos.

Maioria dos problemas de segurança ocorre devido à operação incorreta, ou seja, erro do usuário ou do administrador do sistema. Esse tipo de problema é normalmente considerado um "ataque ao sistema", já que dificilmente pode-se distingui-los na prática (ALBUQUERQUE; RIBEIRO, 2002, p. 3).

2.1.1 Segurança da aplicação desenvolvida

Segundo Albuquerque e Ribeiro (2002, p. 6), há um grande número de recomendações em comum entre segurança da aplicação e normas de boa programação. Seguindo as normas de boa programação, elimina-se muitos erros e falhas de segurança de uma aplicação. São exemplos de normas de boa programação e também práticas para melhorar a segurança do sistema:

a) funções intrinsecamente seguras: tratar todas as variáveis de entrada como não confiáveis, verificando sua autenticidade e validade antes de usá-las;
b) verificação de erros: sempre verificar os erros retornados por uma função, principalmente nas chamadas a *Application Programming Interface* (API) do sistema operacional;
c) documentar o código fonte: se for trabalho em equipe, é indispensável que não ocorra falha de comunicação entre a equipe;
d) testar o retorno das funções executadas: quando uma função for executada, seu retorno precisa ser validado. Existe uma tendência de ignorar determinados erros por serem muito raros. Mesmo em códigos com poucas chance de conflito, isso torna-se um grava erro de segurança (ALBUQUERQUE; RIBEIRO, 2002, p. 241);
e) versões consistentes do sistema: deve-se marcar de uma forma não ambígua o número de versões nos códigos fontes e na versão gerada, de forma a permitir com facilidade a identificação de problemas (SILVA, 2005, p. 67);

f) verificar caracteres especiais: tratar adequadamente os dados, seja pelo usuário ou outro sistema (ALBUQUERQUE; RIBEIRO, 2004, p. 243);
g) não armazenar senhas e chaves de criptografia no código: uma chave ou uma senha de criptografia incluída no código fonte da aplicação pode ser facilmente obtida através da engenharia reversa (ALBUQUERQUE; RIBEIRO, 2002, p. 243).

2.1.2 Garantia de segurança

Segundo Novak (2006), existe um elemento essencial para garantir a segurança da informação, que tem sido esquecido pelos desenvolvedores e administradores de sistema. Refere-se ao reconhecimento de que todo recurso de segurança e cada parte de sua implementação, criada para assegurar um produto mais seguro, deve se agregar a um recurso explícito ou a uma garantia de segurança implícita feita pelo produto. Isso faz sentido, uma vez que se reconhece que tudo que se diz respeito ao software se baseia em torno de seus requisitos funcionais.

Segundo Albuquerque e Ribeiro (2002, p.7), a melhor garantia de segurança para o desenvolvimento de software é através de teste comprovado pelo cliente. Nessa modalidade de garantia, o cliente realiza testes independentes para verificar se a segurança esta adequada.

Para desenvolver sistemas com poucos defeitos e mais confiáveis, as práticas atuais de desenvolvimento devem mudar. Isso requer que desenvolvedores usem métodos de implementações mais adequadas a segurança. Sistema com garantia de segurança são aqueles que conseguem se manter funcionais mesmo sob ação de algum defeito, não comprometendo a qualidade e segurança da informação (DAVIS, 2004, p. iii).

2.2 NORMA ISO/IEC 15.408

O *Common Criteria* (CC) é uma norma ou padrão de indústria, elaborado a partir da união dos diversos padrões anteriores, com a meta de gerar uma norma internacional no assunto. Em janeiro de 1996 foi lançada a primeira versão do CC. Uma grande revisão foi liberada em maio de 1998, denominada CC 2.0. Em dezembro de 1999 a versão 2.1 do CC foi homologada como a norma internacional ISO/IEC 15.408 (ALBUQUERQUE; RIBEIRO,

2002, p. 8).

A norma ISO/IEC 15.408 tem o objetivo de "fornecer um conjunto de critérios fixos que permitem especificar a segurança de uma aplicação de forma não ambígua a partir de características do ambiente da aplicação e definir formas de garantir a segurança de aplicação para o cliente final" (ALBUQUERQUE; RIBEIRO, 2002, p. 7).

O CC estabelece que qualquer sistema, para ser considerado seguro, precisa ter seu *Security Target* (ST) elaborado. O ST é a especificação de segurança para um determinado tipo de software, ou seja, indica quais aspectos de segurança da norma são obrigatórios para o desenvolvimento deste software. Existe também o *Protection Profile* (PP), que indica aspectos de segurança da norma sem especificar uma única aplicação, podendo ser aplicado a toda uma classe de sistema (COMMON CRITERIA, 2009).

O CC oferece uma avaliação independente sobre a capacidade de um produto em atender os padrões de segurança. Os clientes preocupados com segurança, como o governo federal dos Estados Unidos da América (EUA), estão exigindo a certificação ISO/IEC 15.408 como fator determinante nas decisões de compra (APPLE, 2009).

A ISO/IEC 15.408 e o CC definem e mantêm uma rede de laboratórios credenciados a avaliar a segurança das aplicações em determinado nível de garantia. O CC emprega o termo *Target of Evaluation* (TOE), que se refere ao sistema que está sendo desenvolvido ou avaliado. Esse sistema tem funções de segurança, que são o conjunto de rotinas responsáveis por garantir a política de segurança do TOE (ALBUQUERQUE; RIBEIRO, 2002, p. 8).

No Quadro 1 são apresentadas as classes da norma ISO/IEC 15.408.

denominação	**descrição da classe**	**tradução da classe**
FAU	*Function security AUdit*	auditoria de segurança
FDP	*Function user Data Protection*	proteção de dados do usuário
FCS	*Function Cryptographic Support*	criptografia
FPT	*Function Protection of the Total system failure*	autoproteção
FTP	*Function Trusted Path/channels*	caminhos ou canais confiáveis
FIA	*Function Identification and Authentication*	identificação e autenticação
FTA	*Function Targe of evaluation Access*	acesso ao sistema
FMT	*Function security ManagemenT*	gerenciamento de segurança
FRU	*Function Resource Utilization*	utilização de recursos
FPR	*Function PRivacy*	privacidade

Fonte: Common Criteria (2009).

Quadro 1 – Classes da norma ISO/IEC 15.408

2.2.1 Auditoria de segurança – classe FAU

A auditoria de segurança serve para reconhecer, gravar, armazenar e analisar as informações relativas às atividades realizadas pelos usuários nos sistemas. As informações de auditoria gravadas podem ser analisadas para saber como o processo se sucedeu e quem o realizou (COMMON CRITERIA, 2009, p. 29).

A auditoria de software significa uma parte da aplicação, ou conjunto de funções do sistema, que viabiliza uma auditoria. Isso ocorre pela gravação e manutenção de uma trilha de ações realizadas no sistema, ou seja, o sistema mantem os registros de tudo que foi feito nele de forma que, em caso de problema de segurança, alguém possa identificar o que ou quem o acessou (ALBUQUERQUE; RIBEIRO, 2002, p. 109).

Segundo Silva (2005, p. 42), para cada evento auditado devem-se registrar ao menos as seguintes informações:

a) data e hora do evento, tipo do evento, identidade do sujeito (usuário ou sistema) e resultado final (sucesso ou falha);
b) para cada tipo de evento, baseado na definição do evento na especificação de segurança.

De acordo com Albuquerque e Ribeiro (2002, p.110), os objetivos da auditoria podem ser:

a) segunda linha de proteção: ser capaz de responsabilizar o usuário em caso de falha das funções de segurança, devido a uma fraude no sistema;
b) melhoria do sistema: medir o funcionamento dos mecanismos de proteção e identificar falhas de proteção, de forma a definir possíveis pontos de melhoria no sistema;
c) prevenção: avisar sobre as tentativas de invasão ou ameaças que tentem repetidamente fraudar os mecanismos de segurança do sistema.

2.2.2 Proteção de dados do usuário – classe FDP

Segundo Alves e Alves (2009, p. 12), a proteção de dados do usuário está ligada diretamente ao controle de acesso e também à importação e exportação de dados. Pois estes definem diretamente como acessar e disponibilizar as informações de algum usuário para

outros usuários ou para outros sistemas.

O CC contém requisitos que especificam funções de segurança e políticas para proteger dados dos usuários. Os requisitos são divididos em quatro grupos (SILVA, 2005, p. 49):

a) políticas de funções de segurança para proteção dos dados do usuário:
    a. política de controle de acesso,
    b. política de controle de fluxo de informações;
b) forma de proteção dos dados do usuário:
    a. funções de controle de acesso,
    b. funções de controle de fluxo de informações,
    c. transferências internas,
    d. proteção de informação residual,
    e. retorno (*rollback*),
    f. integridade de dados armazenados;
c) armazenamento *off-line*, importação e exportação:
    a. autenticação de dados,
    b. exportação de dados para fora do controle do sistema,
    c. importação de dados de fora do controle do sistema;
d) comunicação interna:
    a. proteção de confidencialidade de dados do usuário,
    b. integridade na transferência de dados;

2.2.3 Criptografia – classe FCS

A criptografia consiste em um método que modifica o texto original de uma mensagem, gerando um texto criptografado na origem, através de um processo de codificação definido por um método de criptografia. A mensagem criptografada pode então ser transferida e, no destino, o processo inverso ocorre, ou seja, o método de criptografia é aplicado desta vez para decodificar o texto criptografo, transformando-o no texto normal original (ALVES; ALVES, 2009, p. 12).

Há dois tipos de criptografia a simétrica e a assimétrica. A simétrica utiliza uma chave para criptografar os dados e a mesma chave serve para decriptografá-los. Na criptografia assimétrica, as chaves são sempre produzidas em par. À garantia de confidencialidade na

criptografia deve ser baseada na segurança da chave, sendo ideal usar um algoritmo público conhecido (ALBUQUERQUE; RIBEIRO, 2002, p. 155).

Segundo Alves e Alves (2009, p. 13), o segredo tem que estar na chave e não no algoritmo, pois os algoritmos públicos mais conhecidos já foram desenvolvidos e aperfeiçoados para evitar a criptoanálise e decifração do texto.

2.2.4 Autoproteção – classe FPT

Segundo Albuquerque e Ribeiro (2002, p. 177), em uma aplicação existem atributos, funções e informações utilizadas para seus objetivos finais e existem outros atributos, funções e informações que definem os parâmetros de segurança desta. A autoproteção é destinada ao sistema de segurança do aplicativo. Existem três pontos que podem ser atacados nas funções de segurança do sistema:

a) dados e atributos de segurança: definições de controle de acesso, autenticação, configuração de autenticação e outros. Verifica-se que, perdido o controle sobre esses dados, as funções de segurança do sistema perdem sua utilidade;
b) implementação das funções de segurança: sendo possível alterar as funções de segurança, também atinge-se indiretamente os dados confidencias do sistema;
c) camada subjacente: o sistema não tem como manter a segurança sobre uma plataforma comprometida.

De acordo com Albuquerque e Ribeiro (2002, p. 178), existe um número mínimo de mecanismos de segurança que são obrigatórios:

a) controle de acesso a dados de segurança: todo acesso a nível de atributos de segurança, deve ser controlado pelo sistema, garantindo sua integridade;
b) autoteste: todo sistema de segurança precisa verificar sua integridade física no início da execução;
c) proteção física: todo software executado sobre um *hardware*, precisa de um nível mínimo de segurança física;
d) teste da camada subjacente: visa identificar se a plataforma ou o sistema operacional sobre o qual o sistema esta executando é seguro e não comprometido;
e) falha segura: erros no próprio sistema ou na camada subjacente que devem ser tratados de forma segura pelo sistema.

Todos os dados de segurança devem ser protegidos de forma análoga aos dados do

usuário, ou seja, as medidas (controle de acesso, integridade, disponibilidade, informação residual e outros) que protegem os dados do usuário devem também proteger os dados de segurança (ALVES; ALVES, 2009, p. 14).

2.2.5 Caminhos ou canais confiáveis – classe FTP

Um canal seguro ou confiável é uma camada de comunicação entre o usuário e o sistema ou entre o sistema e outros sistemas e que oferece uma serie de características (DATASUS, 2008, p. 66):

a) os dados de segurança são isolados dos dados do usuário;
b) os canais devem ser iniciados de forma especifica, seja pelo sistema, seja pelo usuário;
c) o canal prove não repudio de origem e recebimento, garantindo para cada uma das partes que a outra é quem afirma ser.

Segundo Albuquerque e Ribeiro (2002, p. 173), existe uma distinção entre canal confiável e caminho confiável. No caso de canais confiáveis, a comunicação pode ser estabelecida por qualquer ponto, mas existe a autenticação de ambas as partes. Os canais confiáveis são criados através de uma distribuição de chaves públicas e privadas para os sistemas e usuários envolvidos. As partes se identificam trocando uma chave de sessão criptografada pela chave privada de cada um, assim não só garantem a autenticação como a integridade e confidencialidade da comunicação daquele canal.

O caminho confiável, foca em garantir que o usuário está efetivamente acessando o sistema desejado, por exemplo, fazendo sua autenticação.

2.2.6 Identificação e autenticação – classe FTP

Segundo Albuquerque e Ribeiro (2002, p. 129), o objetivo da autenticação é garantir que o usuário é de fato quem ele diz ser. Não faz sentido dispor de mecanismos fortes de controle de acesso se um usuário pode se fazer passar por outro, com mais direitos. Da mesma forma, não temos como responsabilizar alguém através da auditoria, pois não há garantia da identificação dos usuários.

Existem três maneiras de se garantir que um usuário é quem ele diz ser:

a) perguntar algo que só aquele usuário é quem ele diz ser;
b) solicitar a apresentação de algo que só aquele usuário teria;
c) identificar o usuário por características pessoais.

Das três alternativas descritas, a primeira é a mais fácil de ser implementada e a mais comum. A informação que somente o usuário sabe, é geralmente uma senha individual de acesso. A segunda alternativa é muito usada no sistema bancário, com os cartões magnéticos. A terceira opção trata de verificar a existência de características pessoais através de dispositivos biométricos, tais como, digitais, íris, formato do rosto, voz e outros.

2.2.7 Acesso ao sistema – classe FTA

Segundo Silva (2005, p. 59), o acesso ao sistema ou controle de sessões envolve desde questões como notificar o usuário sobre quais foram seus últimos acessos até o cancelamento da sessão após um período de inatividade do sistema. As sessões também podem ser restringidas a determinados horários e dias, como horário comercial. Se uma sessão ficar sem uso durante muito tempo, regras, como o seu cancelamento automático, podem ser estabelecidas.

O acesso ao sistema pode ser usado para ajudar na estratégia de segurança (ALBUQUERQUE; RIBEIRO, 2002, p. 145):

a) limitação do acesso ao sistema: conforme o tipo de acesso (local, via rede, via internet) e a hora de acesso (hora de trabalho normal ou fora do expediente), o usuário pode ser impedido de executar o sistema;
b) limitação do escopo do sistema: conforme o tipo de acesso (local, via rede, via internet) e a hora de acesso (hora de trabalho normal ou fora do expediente), o usuário pode ser limitado a alguma funções do sistema;
c) limitações do número de acesso: restringir o número máximo de sessões de um usuário. É importante considerar que um usuário pode querer acessar, por exemplo, simultaneamente do seu *desktop* e notebook;
d) mensagem de acesso: solicitar que o usuário confirme a leitura de uma mensagem antes de liberar o acesso ao sistema.

2.2.8 Gerenciamento de segurança – classe FMT

A CC define uma classe para gerenciamento de segurança que envolve atributos, informações e funções de segurança. Segundo Albuquerque e Ribeiro (2002, p. 217), todo gerenciamento de segurança passa por pontos importantes como:

a) definição e gerência de papéis de segurança, ou seja, a definição de quem são os administradores, ou quais usuários possuem acesso a determinadas funções ou informações de segurança;
b) capacidade de revogação e expiração de atributos de segurança, ou seja, até que ponto o sistema terá capacidade de revogar imediatamente um direito ou estabelecer um prazo para tal.

De acordo com Albuquerque e Ribeiro (2002, p. 218), para desenvolvimento de um software seguro, necessita de três funções:

a) gerenciamento das funções de segurança: descreve o acesso e os atributos das funções de segurança;
b) gerenciamento dos atributos de segurança: descreve como será feito o acesso aos atributos de segurança de outros objetos do sistema;
c) gerenciamento de dados de segurança: controle dos dados das funções de segurança do sistema. Deve ser restrita ao administrador a atribuição de direitos no acesso aos dados de segurança do sistema (DATASUS, 2008, p. 44).

2.2.9 Utilização de recursos – classe FRU

Esta classe visa garantir que memória, tempo do processador, espaço em disco e qualquer outro recurso que precise ser utilizado pelas funções de segurança, estejam disponíveis para garantir a estratégia da segurança da aplicação (DATASUS, 2008, p. 56).

Conforme Silva (2005, p. 58) “Sistemas que envolvem segurança precisam de uma política de utilização de recursos que garanta espaço e prioridade para suas rotinas de segurança”.

A CC define três atributos, cobrindo cada um dos aspectos apontados (DATASUS, 2008, p.56):

a) alocação de recursos: define que a memória e disco devem ser alocados de forma a

sempre permitirem uma margem mínima para o sistema de segurança (DATASUS, 2008, p. 56);

b) prioridade de serviço: define que o tempo do processador deve ser divido entre os diversos processos. Entretanto o sistema de segurança deve ter sempre a prioridade de execução quando a sua ação é exigida (DATASUS, 2008, p. 56);
c) tolerância a falhas: no caso de falha nos dois itens anteriores, o sistema de segurança deve ser capaz de manter um estado seguro, ou seja, apresentar certa tolerância a falhas (ALBUQUERQUE; RIBEIRO, 2002, p.234).

2.2.10 Privacidade – classe FPR

Privacidade é capacidade de um usuário realizar ações em um sistema sem que seja identificado. É completamente diferente de confidencialidade, que define que apenas usuários autorizados podem ter acesso à determinada informação (ALBUQUERQUE; RIBEIRO, 2002, p. 207).

De acordo com Albuquerque e Ribeiro (2002, p. 207), a privacidade pode existir de diversas formas, conforme a necessidade da política de segurança ou de privacidade interna do sistema:

a) invisibilidade: o sistema garante a um usuário que os demais não tem como saber quem está usando qual recurso do sistema (ALBUQUERQUE; RIBEIRO, 2002, p 207);
b) não rastreamento: ações realizadas pela usuário não interligadas entre si e nem a identidade do mesmo (ALBUQUERQUE; RIBEIRO, 2002, p 208);
c) pseudônimo: segundo Albuquerque e Ribeiro (2002, p 208), esse recurso é muito usado quando há necessidade de responsabilização do usuário e privacidade ao mesmo tempo. Para isto, o usuário é autenticado e identificado no sistema, sendo que, em todas as suas ações, aparece apenas seu pseudônimo;
d) anonimato: corresponde a não identificação efetiva do usuário. Um exemplo desse caso são as páginas web (ALBUQUERQUE; RIBEIRO, 2002, p 208).

2.3 JAVA RMI

Segundo Castro, Raeder e Nunes (2007, p. 3), JAVA RMI é uma solução *Java* para implementar comunicação entre aplicações distribuídas orientadas a objeto. A composição básica de uma aplicação que utiliza JAVA RMI é constituída por dois componentes principais: cliente e servidor. O servidor disponibiliza os serviços para serem acessados remotamente. O cliente, por sua vez, utiliza tais serviços de acordo com sua necessidade. Desta forma, é possível que objetos em diferentes *Java Virtual Machines* (JVM) interajam entre si, independentemente da localização física dessas máquinas.

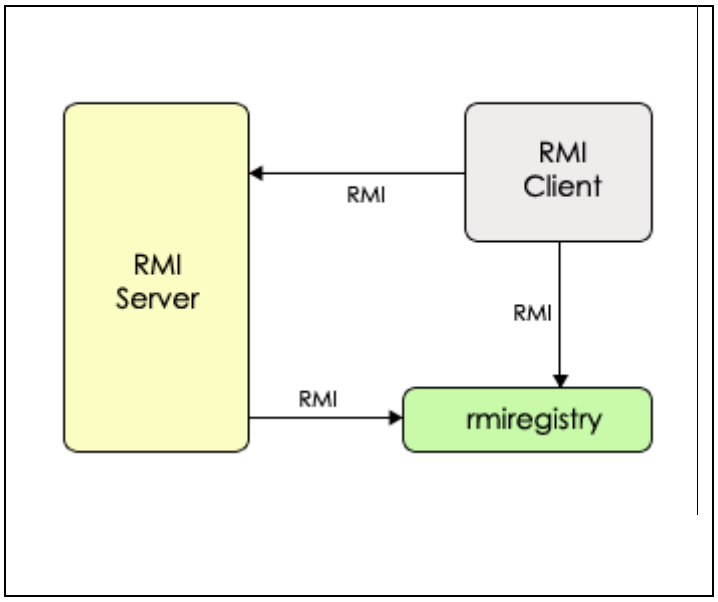
A tecnologia JAVA RMI foi disponibilizada a partir da versão 1.1 do kit de desenvolvimento *Java* (JDK) e sua *Application* P*rogramming Interface* (API) é especificada no pacote java.rmi (CASTRO; RAEDER; NUNES, 2007, p.3).

Segundo Castro, Raeder e Nunes (2007, p. 3), aplicações distribuídas orientadas a objetos, apresentam três características principais: localização dos objetos remoto, comunicação entre objetos remotos e definição de classes.

2.3.1 Localização dos objetos remotos

Segundo Castro, Raeder e Nunes (2007, p. 3), o cliente necessita de uma maneira de descobrir quais os objetos remotos providos pelo servidor. Para tanto, uma referência a estes objetos é fornecida ao cliente. Este mecanismo é denominado Serviço de Nomes RMI, implementado através do aplicativo *RMI Registry*.

O processo de localização de objetos no JAVA RMI é demonstrado na Figura 1. O servidor JAVA RMI, que esta sendo executado em uma JVM envia uma mensagem ao *RMI Registry*, associando um objeto (que contém os serviços disponibilizados pelo servidor) a um nome. O cliente, por sua vez, procura o objeto remoto através do seu nome, consultando o *RMI Registry*. Desta forma, o cliente invoca o método desejado.



Fonte: Castro, Raeder e Nunes (2007, p. 4).

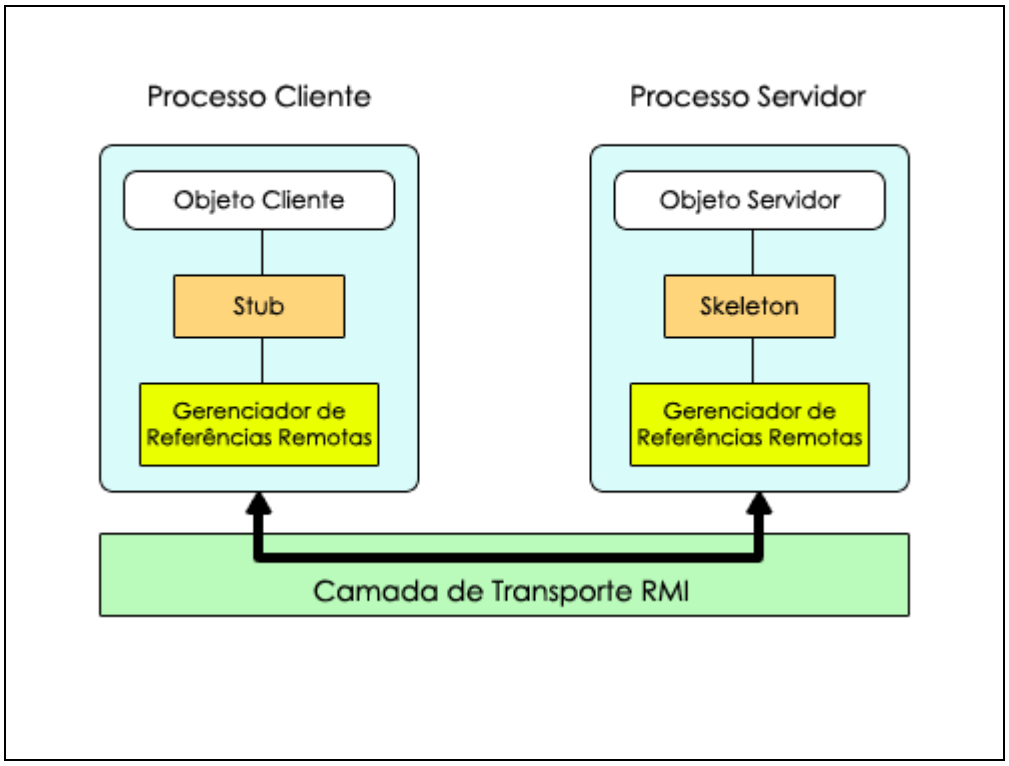
Figura 1 – Localizador de objetos remotos

2.3.2 Comunicação entre objetos remotos

Segundo Castro, Raeder e Nunes (2007, p. 3), os objetos devem ser capazes de comunicaram-se. Com a utilização de JAVA RMI, esta comunicação é realizada de forma transparente ao programador, dando a impressão de estar invocando métodos de objetos locais. A arquitetura JAVA RMI oferece a transparência de localização através da organização de três camadas entre os objetos e o servidor:

a) *stub/skeleton* : responsável por tratar os aspectos das redes, deixando transparentes todas as questões relacionadas a comunicação (CASTRO; RAEDER; NUNES, 2007, p. 4);
b) gerenciador de referência remota: é o *middleware* entre o *Stub* e o protocolo de transporte. Quando ocorre uma consulta ao *RMI Registry*, os objetos resultantes desta requisição são gerenciados por esta camada (CASTRO; RAEDER; NUNES, 2007, p. 4);
c) camada de transporte: oferece o meio necessário de comunicação entre os objetos cliente e servidor através da rede (CASTRO; RAEDER; NUNES, 2007, p. 4).

Na Figura 2, é ilustrada a organização destas três camadas em uma aplicação RMI.



Fonte: Castro, Raeder e Nunes (2007, p. 5).

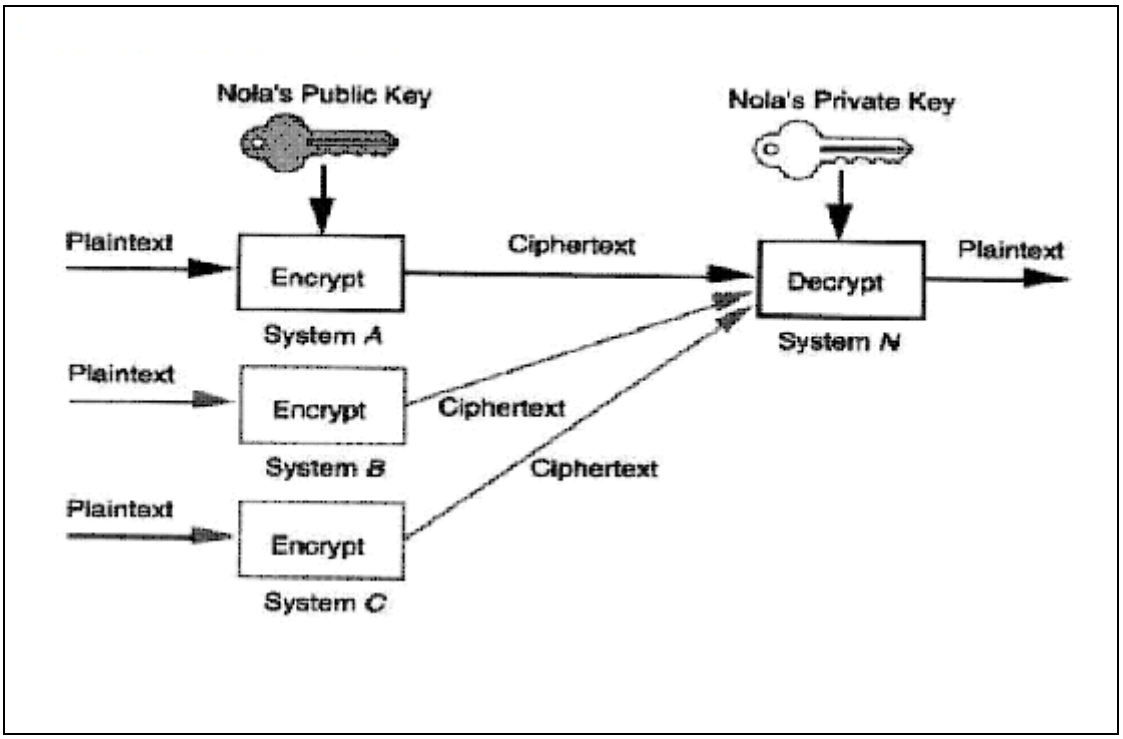
Figura 2 – Arquitetura básica de uma aplicação JAVA RMI

2.4 RSA

Segundo Pereira (2006, p. 20), dentre todos os algoritmos criptográficos de chave publica, o RSA é o mais conhecido e, atualmente, o mais usado, sobre tudo em aplicações comercias. Este algoritmo permite que qualquer usuário codifique mensagens e textos, mas como a chave de decodificação é secreta, só o destinatário legítimo poderá decodificá-la.

Segundo Barbosa e Braghetto (2003, p. 16), o RSA é o resultado de dois cálculos matemáticos. Um para cifrar e outro para decifrar. O RSA usa duas chaves criptográficas, uma chave pública e outra privada. No caso deste algoritmo usa-se a criptografia assimétrica que, a chave pública é usada para criptografar a mensagem e a chave privada para decriptografá-la.

Na figura 3 é ilustrado o funcionamento da chave pública e da chave privada em uma mensagem.



Fonte: adaptado de Schimitt (2001, p. 8).

Figura 3 – Criptografia assimétrica

2.5 TRABALHOS CORRELATOS

Podem-se encontrar exemplos de trabalhos que implementaram normas de segurança em sistemas, nas universidades, internet e livros. Dentre eles, foram escolhidos três trabalhos: “Implementação de Requisitos de Segurança para o Projeto de Rastreabilidade Bovina conforme a ISO 15.408” (CARMISINI, 2010), “Segurança no Desenvolvimento de Aplicação Web” (GOMES; SANTOS, 2006) e “Processo de Apoio à Segurança de Software” (NUNES, 2007).

2.5.1 Implementação de Requisitos de Segurança para o Projeto de Rastreabilidade Bovina conforme a ISO 15.408

O objetivo deste trabalho é o desenvolvimento de um [2]*framework* que implemente os atributos de segurança da informação, segundo a norma ISO/IEC 15.408, necessário neste trabalho (CARMISINI, 2010).

Segundo Carmisini (2010, p. 14), o *framework* é composto de métodos para garantir a proteção de dados do usuário, a auditoria de segurança e realizar cópia de segurança e restauração. Além de possuir um atualizador de versão para gerenciar cada nova versão disponibilizada, verificando a consistência dos dados.

O projeto de rastreabilidade bovina (PRB) tem por objetivo o desenvolvimento de uma solução de software e hardware voltado para o gerenciamento do gado de corte. Onde o público alvo não é da área de computação e, sabendo-se que os usuários leigos geralmente comentem erros de operação que desdobram em falhas no sistema, foi estabelecido o desenvolvimento deste trabalho (CARMISINI, 2010).

Como a norma ISO/IEC 15.408 possui muitos itens de segurança, foram estudados e implementados os itens mais adequados ao contexto do PRB que são as classes criptografia, auditoria, proteção de dados do usuário, autoproteção e canal seguro.

Ao final do trabalho Carmisini (2010, p.61) concluiu que, o desenvolvimento do *framework* é viável, porém torna-se muito complexo em se tratando de implementar todos os requisitos de segurança presentes na norma ISO/EIEC 15.408.

2.5.2 Segurança no Desenvolvimento de Aplicações Web

O principal objetivo deste trabalho foi fornecer conceitos tecnológicos voltados à segurança da informação, apresentando os principais requisitos funcionais de segurança dispostos na norma ISO/IEC 15.408. Também foram apresentadas as principais ameaças à segurança da informação, como os principais meios de desenvolvimento de sistema menos vulneráveis (GOMES; SANTOS; 2006; p. 13).

Segundo Gomes e Santos (2006, p. 14), o trabalho deu ênfase na análise das principais

[2] *Framework* – é um conjunto de classes que realizam determinadas soluções para a aplicação (JOHNSON; FOOTE, 1998).

vulnerabilidades existentes no sistema *on-line* da Universidade da Amazônia, o Aprendiz. Os métodos para desenvolvimento deste trabalho foram da seguinte forma:

a) foram identificados na literatura as técnicas e procedimentos considerados fundamentais para a segurança da informação;
b) foi apresentado uma visão da segurança da informação, avaliação de ambiente e definição de estratégias de segurança;
c) foi apresentado uma relação de requisitos funcionais de segurança, que são considerados durante o desenvolvimento de software;
d) foi realizada uma abordagem da necessidade de analisar os riscos que uma aplicação está sujeita, relacionando as principais presentes no ambiente web.

Foi feito uma analise deste projeto e verificaram-se algumas vulnerabilidades no ambiente de aprendizado o Aprendiz, em que alguns requisitos, com o acesso ao sistema, estão em desacordo com a norma ISO/IEC 15.408, ou com as boas práticas de programação.

2.5.3 Processo de Apoio a Segurança de Software

Segundo Nunes (2007, p. 3), uma das metas do trabalho é auxiliar desenvolvedores de software a evitarem a abordagem de segurança "aprofundar e ajustar", em que falhas são corrigidas, quando estes aprendem ou tomam conhecimento sobre elas, atestando que a segurança não foi adequadamente considerada.

Este trabalho tem como objetivo contribuir para a construção de software seguro e, por conseguinte, software de maior qualidade, baseando-se no *System Security Engineering Capability Maturity Model* (SSE-CMM), no processo de avaliação OCTAVE, na ISO/IEC 15.408 e na ISO/IEC 27002 (NUNES, 2007, p. 3).

O SSE-CMM é um modelo que determina o processo de maturidade de uma organização que trabalha com engenharia de segurança.

O OCTAVE é um processo de auto orientado de avaliação de riscos para segurança por meio de análise dos ativos, ameaças, vulnerabilidade, e impacto organizacional.

A ISO/IEC 27002 orienta para a preservação da confidencialidade, da integridade e da disponibilidade das informações, por meio da implementação de controles.

Segundo Nunes (2007, p. 175), podem-se destacar como principais contribuições deste trabalho, a proposta de um processo de desenvolvimento de software formado por atividades de segurança, permitindo uma visão mais detalhada de segurança da informação. Além de

ressaltar a importância de se aplicar boas práticas de segurança no ciclo de vida do sistema.

3 DESENVOLVIMENTO

Este capítulo detalha as etapas do desenvolvimento do *middleware*. São apresentados os requisitos, a especificação e a implementação do mesmo, mencionando as técnicas e ferramentas utilizadas. Também são comentadas questões sobre a operacionalidade e os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O presente trabalho possui como Requisitos Funcionais (RF) o Quadro 2 e como Requisitos Não Funcionais (RNF) o Quadro 3.

REQUISITOS FUNCIONAIS	**CASO DE USO**
RF01: o *middleware* deve permitir realizar a validação do login	UC001
RF02: o *middleware* deve permitir criptografar dados de acesso	UC002
RF03: o *middleware* deve permitir validar acesso às opções do sistema	UC003
RF04: o *middleware* deve permitir alterar a senha do usuário	UC004
RF05: o *middleware* deve permitir realizar a verificação de tempo inativo	UC005
RF06: o *middleware* deve permitir gerar senha padrão	UC006
RF07: o *middleware* deve permitir gravar log de alteração	UC007
RF08: o *middleware* deve permitir gravar log de exclusão	UC007
RF09: o *middleware* deve permitir gravar log de inclusão	UC007
RF10: o *middleware* deve permitir gravar log de acesso ao sistema	UC007
RF11: o *middleware* deve permitir gravar log de acesso a opções do sistema	UC007
RF12: o *middleware* deve permitir gerar chave pública	UC008
RF13: o *middleware* deve permitir gerar chave privada	UC008
RF14: o *middleware* deve permitir criptografar um arquivo	UC002
RF15: o *middleware* deve permitir decriptografar um arquivo	UC002
RF16: o *middleware* deve permitir gravar arquivos criptografados	UC009
RF17: o *middleware* deve permitir cadastrar os usuários do sistema	UC010
RF18: o *middleware* deve permitir cadastrar os grupos do sistema	UC011
RF19: o *middleware* deve permitir cadastrar os softwares que irão utilizá-lo	UC012

RF20: o *middleware* deve permitir cadastrar os parâmetros do sistema	UC013
RF21: o *middleware* deve permitir cadastrar as telas do sistema	UC014
RF22: o *middleware* deve permitir consultar os logs gerados	UC015
RF23: o *middleware* deve permitir gerar relatórios gerenciais	UC016
RF24: o *middleware* deve permitir consultar os arquivos criptografados	UC009

Quadro 2 – Requisitos funcionais

REQUISITOS NÃO FUNCIONAIS
RNF01: o *middleware* deverá ser implementado na linguagem Java
RNF02: deverá ser implementado utilizado o ambiente de desenvolvimento Netbeans 6.9.1
RNF03: deverá utilizar a *Application Programming Interface* (API) *Java DataBase Connectivity* (JDBC), para gravação dos dados
RNF04: o sistema gerencial deverá ser implementado na linguagem Pascal
RNF05: o sistema gerencial deverá ser implementado no ambiente Delphi 7.0
RNF06: deverá utilizar o Sistema de Gerenciamento de Bando de Dados (SGDB) Oracle 10g
RNF07: deverá realizar a comunicação entre o cliente e o servidor
RNF08: deverá utilizar o algoritmo de criptografia *Message Digest algorithm* (MD5)

Quadro 3 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Na seqüência é apresentada a especificação do *middleware*, que foi modelado na ferramenta *Enterprise Architect* (EA) (SPARXSYSTEMS, 2000). Para o desenvolvimento dos diagramas de casos de uso, atividades, classes e de seqüência, foi utilizado conceitos de Orientação Objeto (OO) e de *Unified Modeling Language* (UML) (OMG, 2005).

O Quadro 4 apresenta dados referente ao desenvolvimento do *middleware*, sob o aspecto das classes da norma ISO/IEC 15.408 (descritas no anexo A). É indicado na coluna ‘Atendido’ se na implementação um ou mais itens da classe foram implementados.

Classe	Sigla	Atendido	Motivo
Auditoria	FAU	Sim	Implementado
Proteção de dados do usuário	FDP	Sim	Implementado
Criptografia	FCS	Sim	Implementado

Autoproteção	FPT	Não	Não estava nos requisitos
Canais seguros	FTP	Não	Pouco utilizado por aplicações comerciais
Autenticação	FIA	Sim	Implementado
Acesso ao sistema	FTA	Sim	Implementado
Gerenciamento de segurança	FMT	Sim	Implementado
Utilização de recursos	FRU	Não	Não estava nos requisitos
Privacidade	FPR	Não	Não estava nos requisitos

Quadro 4 – Classes atendidas no desenvolvimento

3.2.1 Diagrama de Caso de Uso

Na especificação do *middleware* existem dois cenários. O primeiro trata das opções existentes para o ator software (Figura 4) e o segundo trata das opções existentes para o ator administrador de segurança (Figura 5).

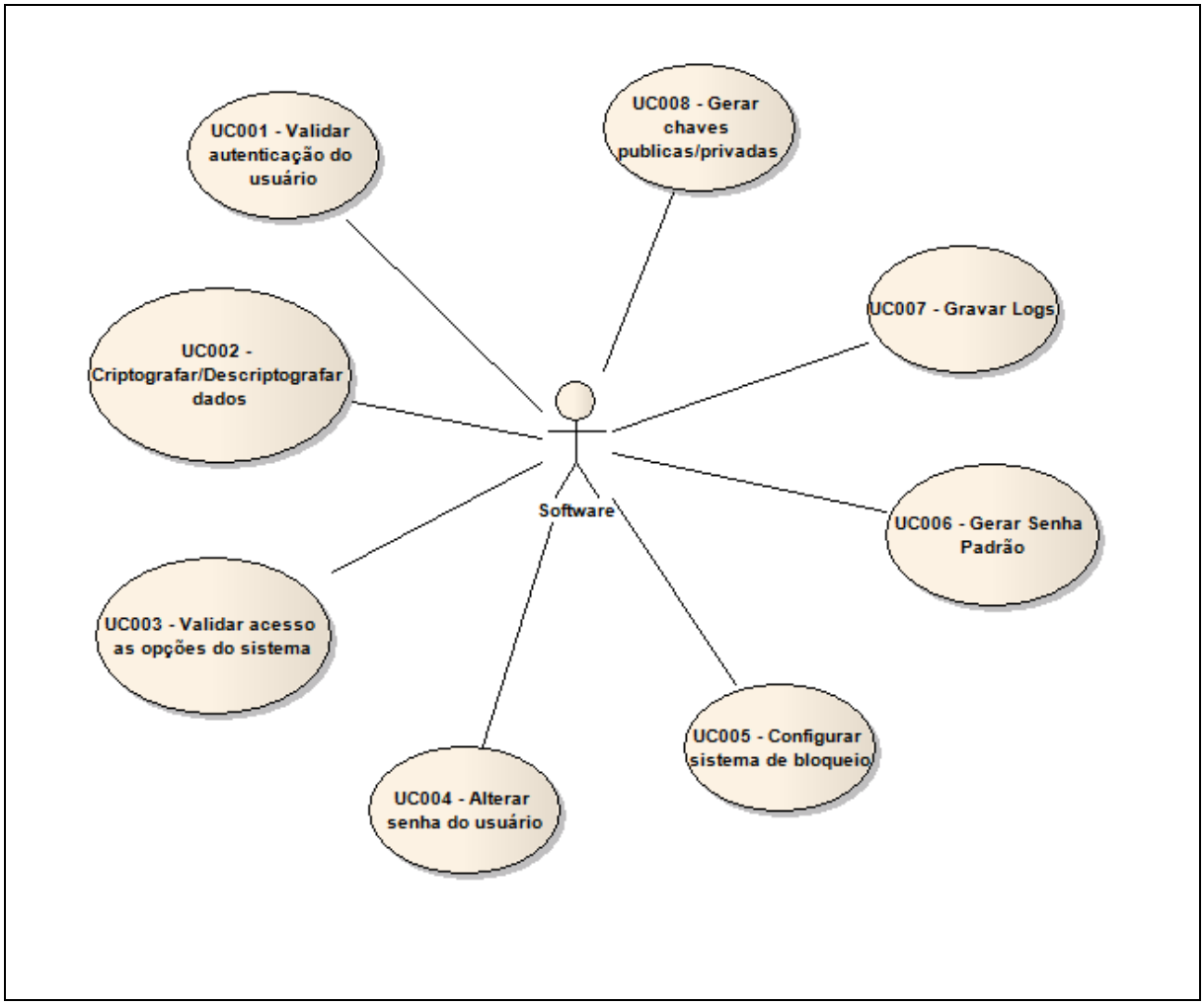


Figura 4 – Diagrama de caso de uso executado pelo software

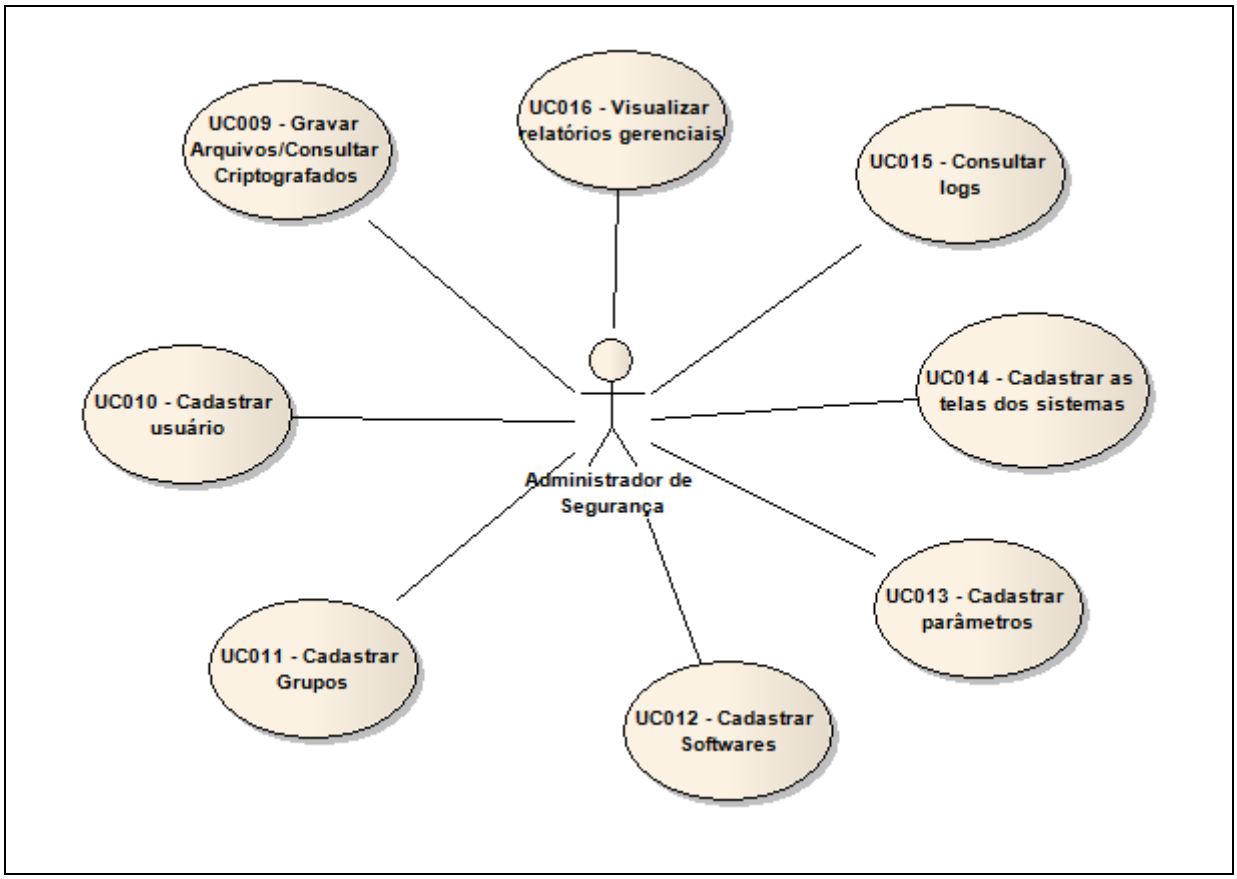


Figura 5 – Diagrama de caso de uso executado pelo administrador de segurança

3.2.2 Especificações a classe FDP – proteção de dados do usuário

Os itens atendidos da classe FDP (descrito no anexo A) são:

a) FDP_ACF.1 – controle de acesso com base de atributos de segurança;

b) FDP_ACC.1 – controle de acesso de subconjuntos;

c) FDP_DAU.2 – autenticidade dos dados com identidade do gerador;

d) FDP_ROL.1 – Retorno básico.

O caso de uso UC002 – Criptografia/Decriptografia de dados (Quadro 5) é responsável pela criptografia e decriptografia de dados do software como de arquivos externos ao mesmo. Para realizar este procedimento é necessário ter pelo menos um dado a ser utilizado. Este caso de uso possui um cenário principal, um alternativo e duas exceções.

UC002 – Criptografar/Descriptografar dados: criptografia dados do software ou de arquivos externos, o mesmo realiza a decriptografia.	
Requisitos atendidos	RF14 e RF15
Pré-condições	Estar conectado no *middleware* e possui um dado a ser criptografado ou decriptografado.

Cenário principal	1. O software executa a função criptografarArquivo. Vai para o passo 3. 2. O software executa a função decriptografarArquivo. Vai para o passo 6. 3. O *middleware* executa a rotina de criptografia. 4. O *middleware* retorna o arquivo criptografado. 5. O software executa a opção de gravar o arquivo em banco de dados. Vai para o passo 8. 6. O *middleware* executa a rotina de decriptografia. 7. O *middleware* retorna o arquivo criptografado. 8. O software finaliza a execução.
Exceção 1	No passo 1,2 e 5 caso o software não consiga conectar com o *middleware*, o RMI retorna uma exceção de *RemoteConnection.*
Exceção 2	No passo 3 e 4, caso não exista arquivo selecionado o *middleware*, retorna uma exceção e o valor booleano *false*.
Pós-condições	Retorno da função booleana *true*.

Quadro 5 – Detalhamento do caso de uso UC002 - Criptografia/Decriptografia de dados

O caso de uso UC003 – Validar acesso às opções do sistema (Quadro 6) é responsável em fazer a validação de acesso aos menus ou opções do sistema. Para realizar este procedimento é necessário o administrador de segurança cadastrar os parâmetros e o programador fazer a validação sobre o resultado do parâmetro. Este caso de uso possui um cenário principal, um alternativo e uma exceção.

UC003 – Validar acesso as opções do sistema: valida através de parâmetros cadastrados para cada funcionalidade.	
Requisitos atendidos	RF03 e RF05
Pré-condições	Estar conectado com o *middleware.*
Cenário principal	1. O software faz a chama da função obterValorParametro. 2. O *middleware* executa a verificação do parâmetro. 3. Software captura o valor retornado pelo *middleware.* 4. Software realiza a validação do parâmetro.
Alternativo 1	No passo 2, caso o *middleware* não consiga validar o valor do parâmetro. 2.1 o *middleware* retorna ‘E’.
Alternativo 2	No passo 2, caso o *middleware* não encontre valor para o parâmetro. 2.1 o *middleware* retorna ‘X’.
Exceção 1	No passo 1, caso o software não consiga conectar com o *middleware*, o RMI retorna uma exceção de *RemoteConnection.*
Pós-condições	Retorno do parâmetro com sucesso.

Quadro 6 – Detalhamento do caso de uso UC003 – Validar acesso às opções do sistema

O caso de uso UC008 – Gerar chave públicas/privadas (Quadro 7) é responsável por gerar um par de chaves onde existirá uma chave pública e uma privada. Para geração das chaves é necessário apenas configurar na parametrização do *middleware* o destino onde às mesmas serão salvas. Este caso de uso possui um cenário principal e dois cenários de exceção

informando possíveis erros durante a geração das chaves.

UC008 – Gerar chaves públicas/privadas: gera chaves publicas no local definido na parametrização do *middleware*.	
Requisitos atendidos	RF12 e RF13
Pré-condições	Estar conectado no *middleware* e possui caminho de destino parametrizado.
Cenário principal	1. O software executa a função criarChavePublicaPrivada. 2. O *middleware* executa as rotinas de criação das chaves 3. O *middleware* grava os arquivos no caminho especificado na parametrização.
Exceção 1	No passo 1, caso o software não consiga conectar com o *middleware*, o RMI retorna uma exceção de *RemoteConnection.*
Exceção 2	No passo 3, caso não existe parametrização configurada o *middleware*, retorna uma exceção e o valor booleano *false*.
Pós-condições	Retorno da função booleana *true*.

Quadro 7 – Detalhamento do caso de uso UC008 - Gerar chave púplicas/privadas

O diagrama de classe (Figura 6) dos casos de uso UC002 - Criptografia/Decriptografia de dados, UC003 – Validar acesso às opções do sistema e UC008 - Gerar chave púplicas/privadas, apresenta as classes responsáveis pela validação de acesso aos menus do software, pela geração de chaves públicas e privadas e pela criptografia dos dados, as quais são:

a) ParametroSistemaCtr: classe que é responsável pela validação e controle de acesso ao software e em seus menus;
b) GeradorParChaves: classe responsável pela geração do par de chaves (chave pública e chave privada), que é utilizado na assinatura digital dos dados. A mesma encontra-se dentro do pacote Criptografia;
c) Cifrador: classe responsável por criptografar os dados ou arquivos do software. A mesma encontra-se dentro do pacote Criptografia;
d) Decifrador: classe responsável por decriptografar os dados ou arquivos do software;
e) CarregadorChavePrivada: classe responsável pela busca e conversão da chave privada. A mesma encontra-se dentro do pacote Criptografia;
f) CarregadorChavePublica: classe responsável pela busca e conversão da chave pública. A mesma encontra-se dentro do pacote Criptografia;
g) Usuário: classe que encapsula os dados do usuário autenticado no momento no software;
h) FIA_AutenticacaoImpl: classe responsável por tratar todas as chamadas feita através da comunicação RMI. Por ela que é feita a troca de informação entre o *middleware* e o software;

i) FIA_Autenticacao: interface obrigatória na implementação de métodos de acesso remoto.

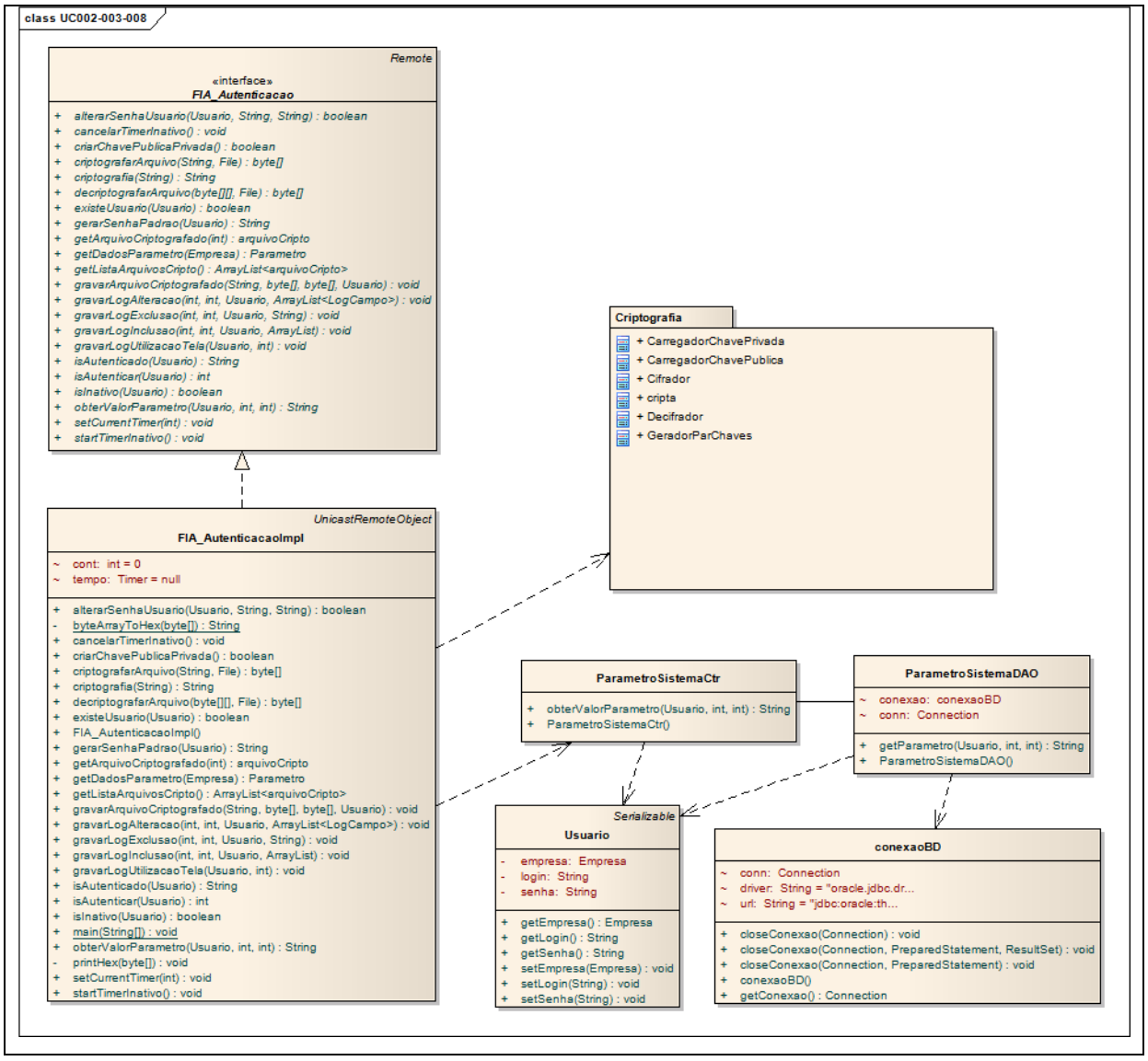


Figura 6 – Diagrama de classe dos casos de uso UC002, UC003 e UC008

A Figura 7 apresenta o diagrama de atividades correspondente ao cenário do caso de uso UC002 – Criptografia/Decriptografia de dados. O software realiza a conexão com o *middleware* através do RMI, seleciona o arquivo, solicita a criptografia de dados e envia o arquivo ou dado a ser criptografado. Por fim o *middleware* valida os dados recebidos, gera a criptografia e devolve ao software o arquivo ou dado criptografado.

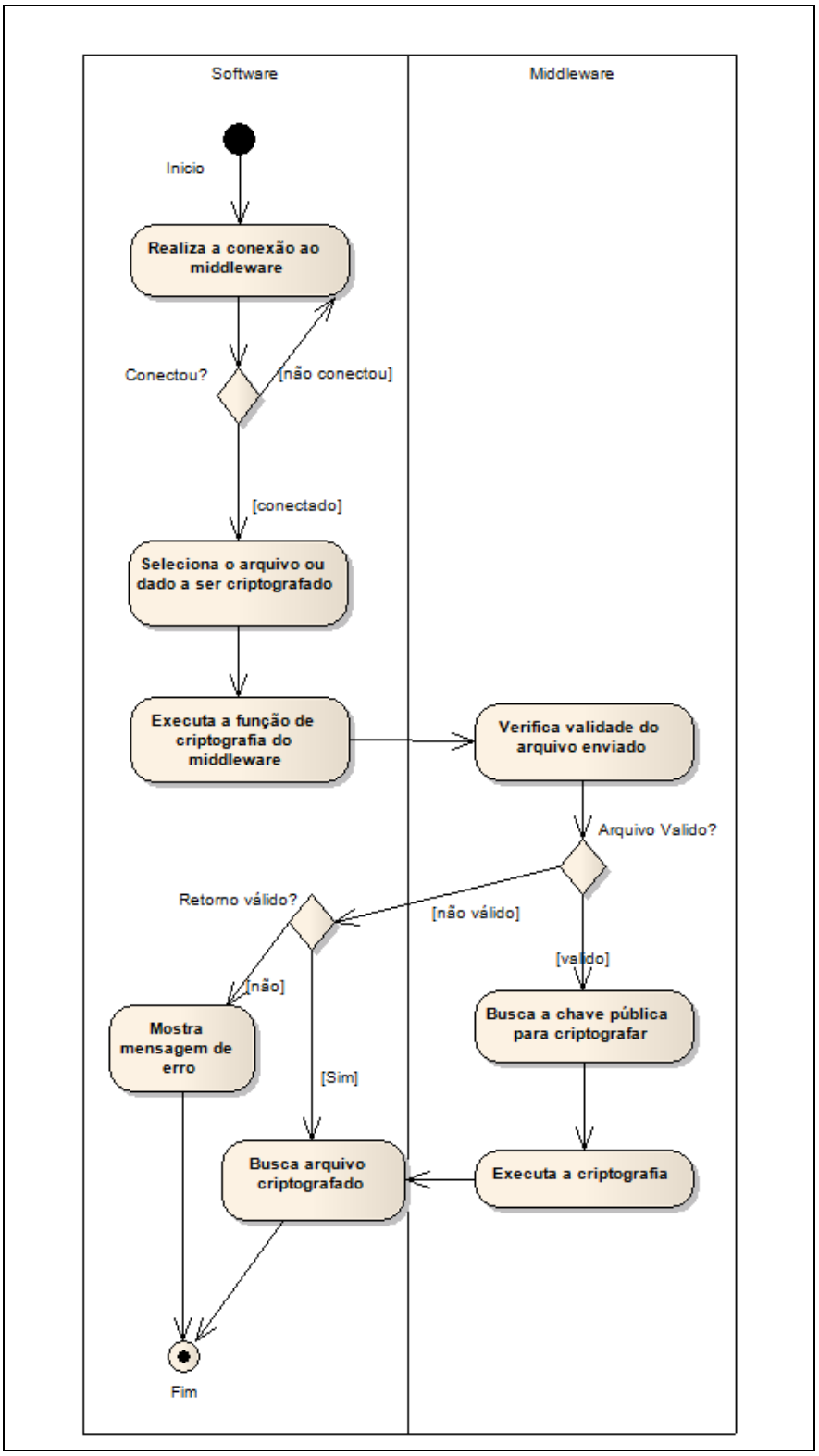


Figura 7 – Diagrama de atividades do caso de uso UC002

A Figura 8 apresenta o diagrama de seqüência correspondente ao cenário do caso de uso UC002 - Criptografia/Decriptografia de dados. Através do software é selecionado o arquivo para criptografia, o mesmo realiza a chamada da rotina de criptografia no *middleware*. O *middleware* captura a chave pública, processa o arquivo e realiza a criptografia

do mesmo.

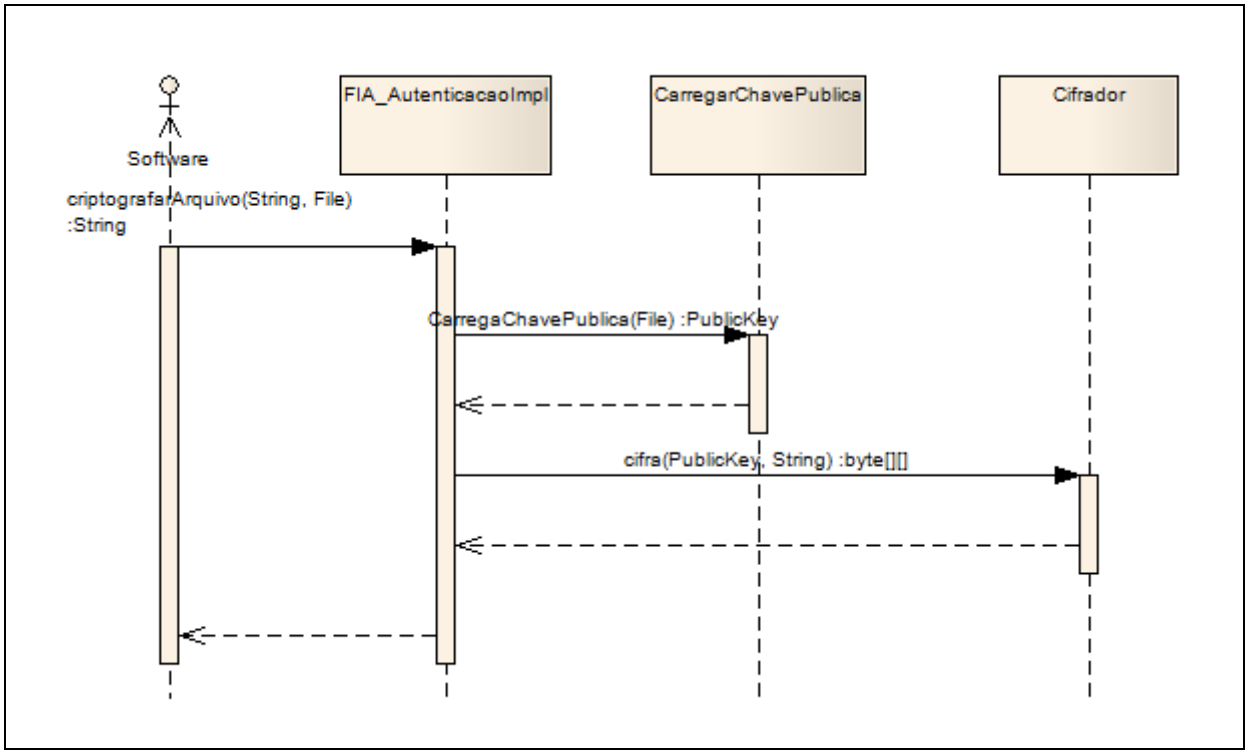


Figura 8 – Diagrama de seqüência do cenário criptografia do caso de uso UC002

3.2.3 Especificação da classe FAU – Auditoria

Os itens atendidos da classe FAU (descrito no anexo A) são:

a) FAU_GEN.1 – geração de dados para auditoria;

b) FAU_GEN.2 – associação do usuário ao evento da auditoria;

c) FAU_SAA.1 – análise de violação potencial;

d) FAU_SAR.1 – revisão da auditoria;

e) FAU_SEL.1 – auditoria seletiva;

f) FAU_STG.1 – armazenamento protegido da trilha de auditoria;

g) FAU_STG.2 – garantia da disponibilidade dos dados para auditoria.

O caso de uso UC007 – Gravar *logs* (Quadro 8) apresenta a gravação de *logs* para auditoria. Ao realizar a gravação o software pode definir qual tipo de ação será o *log*, como por exemplo, *logs* de alteração de registro, logs de acesso ao sistema ou *logs* de inclusão de registros. Este caso de uso possui um cenário principal e duas exceções.

UC007 – Gravar logs: grava *logs* de acesso ao software, alteração, inclusão, exclusão de registros e também de acesso a tela ou funcionalidades do software.	
Requisitos atendidos	RF07, RF08, RF09, RF10 e RF11

Pré-condições	Estar conectado no *middleware*.
Cenário principal	1. O software executa a função de geração de *log* específica para o caso. 2. O *middleware* executa a validação do *log*. 3. O *middleware* conecta com a base de dados. 4. O *middleware* grava os dados do *log*.
Exceção 1	No passo 1 caso o software não consiga conectar com o middleware, o RMI retorna uma exceção de *RemoteConnection*.
Exceção 2	No passo 3, caso o banco de dados esteja *off-line*, o middleware retorna uma exceção.

Quadro 8 – Detalhamento do caso de uso UC007 – Gravar *logs*

O diagrama de classe do caso de uso UC007 – Gravar logs é apresentado na Figura 9 e possui as seguintes classes:

a) LogCtr: classe responsável pela validação, controle e consistência dos logs ;
b) LogDAO: classe responsável pelo armazenamento das trilhas de auditoria, ou seja, gravação dos log em banco de dados;
c) FIA_Autenticacao: interface obrigatória na implementação de métodos de acesso remoto;
d) UsuarioDAO: classe responsável por fazer as conexões com o banco de dados;
e) ConexaoBD: classe responsável por parametrizar os dados de acesso ao banco de dados;
f) FIA_AutenticacaoImpl: classe responsável por tratar todas as chamadas feita através da comunicação RMI. Por ela que é feita a troca de informação entre o *middleware* e o software;
g) Usuário: classe responsável por gravar os dados do usuário conectado no momento no software.

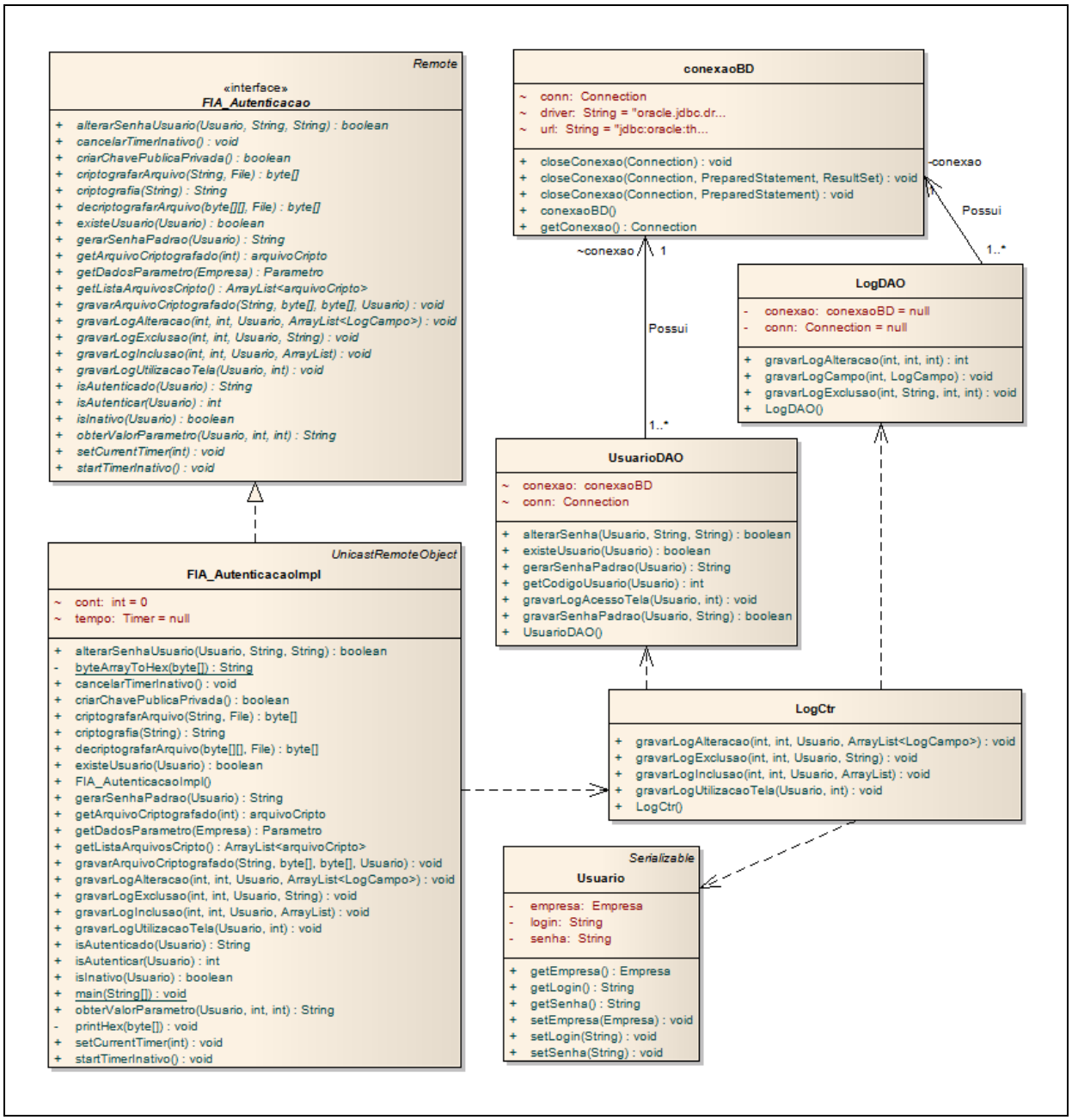


Figura 9 – Diagrama de classe do caso de uso UC007

O caso de uso UC015 – Consulta de logs (Quadro 9) apresenta a opção de poder consultar as trilhas de auditoria. Este caso de uso abrange as etapas FAU_SAA.1 – análise de violação potencial, FAU_SAR.1 – revisão da auditoria, FAU_SEL.1 – auditoria seletiva, FAU_STG.1 – armazenamento protegido da trilha de auditoria e FAU_STG.2 – garantia da disponibilidade dos dados para auditoria. Este caso de uso possui um cenário principal e duas exceções.

UC015 – Consulta logs: verificação das trilhas de auditoria, conforme regras de permissões cadastradas e filtros utilizados.	
Requisitos atendidos	RF22, RF23 e RF24
Pré-condições	1. Ter permissão de acesso ao sistema de auditoria.

	2. Estar conectado ao banco de dados
Cenário principal	1. Administrador de segurança acessar o sistema de auditoria. 2. Administrador de segurança realizar a pesquisa dos *logs*. 3. Administrador de segurança realizar a validação dos dados. 4. Administrador de segurança emite os relatórios gerenciais.
Exceção 1	No passo 1, o acesso pode ser negado e o sistema irá apresentar a mensagem de ‘Acesso negado’
Exceção 2	No passo 1, caso o usuário ou senha seja inválido, o sistema de auditoria irá apresentar a mensagem de ‘Usuário ou senha invalido’.

Quadro 9 – Detalhamento do caso de uso UC015 – Consulta de *log*

3.2.4 Especificação da classe FCS – Criptografia

O item atendido da classe FCS (descrito no anexo A) foi o FCS_COP.1 - operação de criptografia. Este processo foi apresentado no Quadro 5. O diagrama de seqüência apresentado na Figura 10, mostra o funcionamento do item FCS_COP.1 em uma operação de criptografia de senha do usuário.

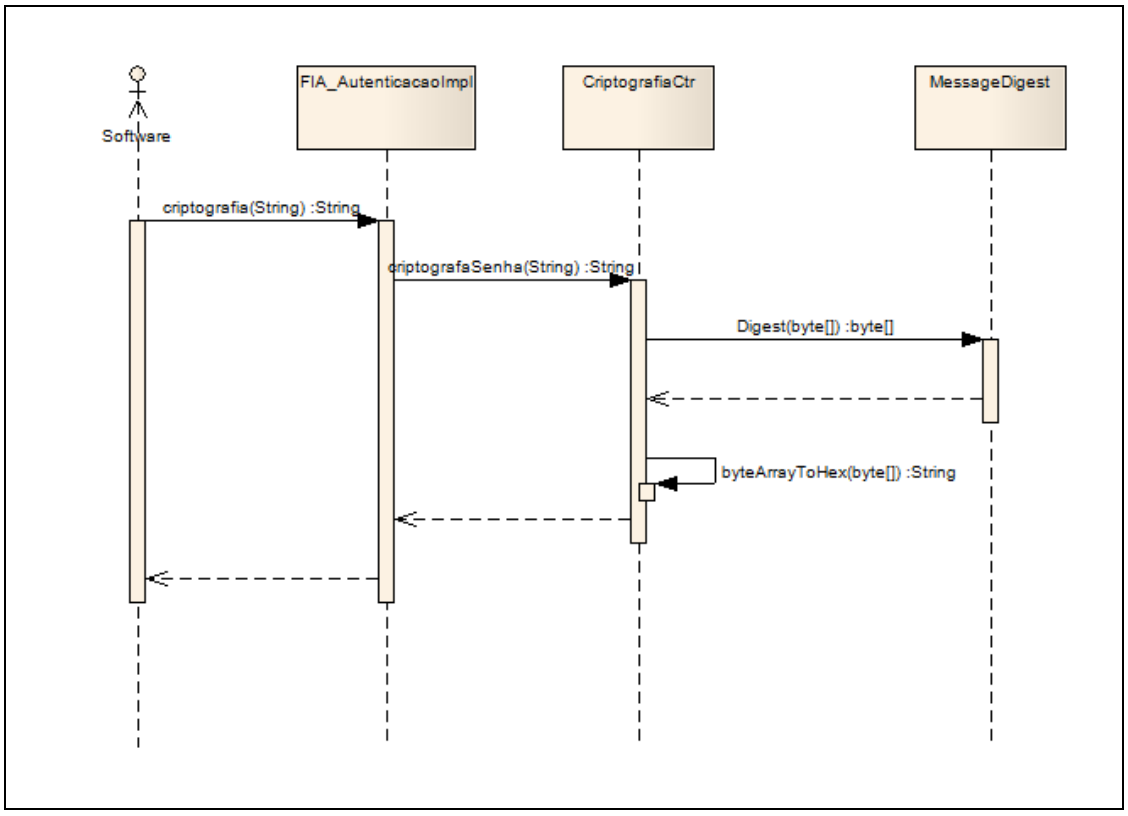


Figura 10 – Diagrama de sequência do item FCS_COP.1 - operação de criptografia

3.2.5 Especificação da classe FIA – autenticação

Os itens atendidos da classe FIA (descrito no anexo A) são:

a) FIA_AFL.1 – tratamentos de falhas de autenticação;
b) FIA_ATD.1 – definição de atributos do usuário para autenticação;
c) FIA_SOS.1 – métrica mínima das senhas;
d) FIA_SOS.2 – capacidade de gerar senhas;
e) FIA_UAU.1 – ações anteriores a autenticação;
f) FIA_UAU.2 – autenticação do usuário antes de qualquer coisa;
g) FIA_UAU.6 – re-autenticação;
h) FIA_UAU.7 – resposta restrita da autenticação;
i) FIA_USB.1 – ligação do usuário com o sistema.

O caso de uso UC005 - Configuração de bloqueio do sistema (Quadro 10) apresenta o bloqueio do sistema através de tempo de inatividade. Sendo este tempo cadastrado pelo administrador de segurança. Também é cadastra a quantidade de tentativas falhas de login que possam ocorrer consecutivamente. Este caso de uso possui um cenário principal, um cenário

alternativo e duas exceções.

UC005 – Configuração de bloqueios do sistema: configura e bloqueia o sistema contra tentativas falhas de login consecutivas e tempo de inatividade.	
Requisitos atendidos	RF05
Pré-condições	Estar conectado no *middleware*
Cenário principal	1. Usuário realiza login no software. 2. *Middleware* valida dados de login. 3. Software autentica usuário. 4. Usuário conectado no software.
Alternativo 1	No passo 4, o *middleware* registra o tempo de inatividade do software. 4.1 *Middleware* busca do parâmetro o valor de inatividade. 4.2 *Middleware* inicia contagem do tempo de inatividade. 4.3 *Middleware v*erifica se passou do tempo limite. Se sim passo 4.5. 4.4 Software verifica se usuário está operando o sistema. Se sim passo 4.1, se não 4.3. 4.5 *Middleware* retorna ao software para bloqueá-lo. 4.6 Software fecha sistema e retorna para tela de login. Retorna passo 1.
Exceção 1	No passo 2, o *middleware* não valida os dados de login 2.1 *Middleware* retorna para o software ‘N’, conexão não permitida. 2.2 O software apresenta a mensagem de ‘Usuário ou senha inválidos’. Retorna passo 1.
Exceção 2	No passo 2, o *middleware* bloqueia login por tentativas falhas 2.3 *Middleware* retorna para o software ‘N’, conexão não permitida. 2.4 O software apresenta a mensagem de ‘Limite de tentativas ultrapassados. Login bloqueado por n minutos’. Retorna passo 1.

Quadro 10 – Detalhamento do caso de uso UC005 – Configuração de bloqueio do sistema

O diagrama de classe do caso de uso UC005 – Configuração de bloqueio do sistema é apresentando na Figura 11 e possui as seguintes classes:

a) FIA_AutenticaçãoImpl: classe responsável por tratar todas as chamadas feitas através da comunicação RMI. Por ela que é feita a troca de informação entre o *middleware* e o software;
b) FIA_Autenticacao: interface obrigatória na implementação de métodos de acesso remoto;
c) ConexaoBD: classe responsável pela conexão com o banco de dados;
d) ParametroDAO: classe de ligação com as tabelas e registros referentes aos

parâmetros de bloqueio do banco de dados;

e) ParametroCtr: classe responsável pelo controle e validação dos parâmetros;
f) TempoInativo: classe responsável pelo controle e verificações de inatividades do software;
g) Inativida: classe que instancia os controles de inatividade para uma determinada tela;
h) PrincipalFrm: classe que contém os componentes visuais, onde é feito o controle de inatividade.

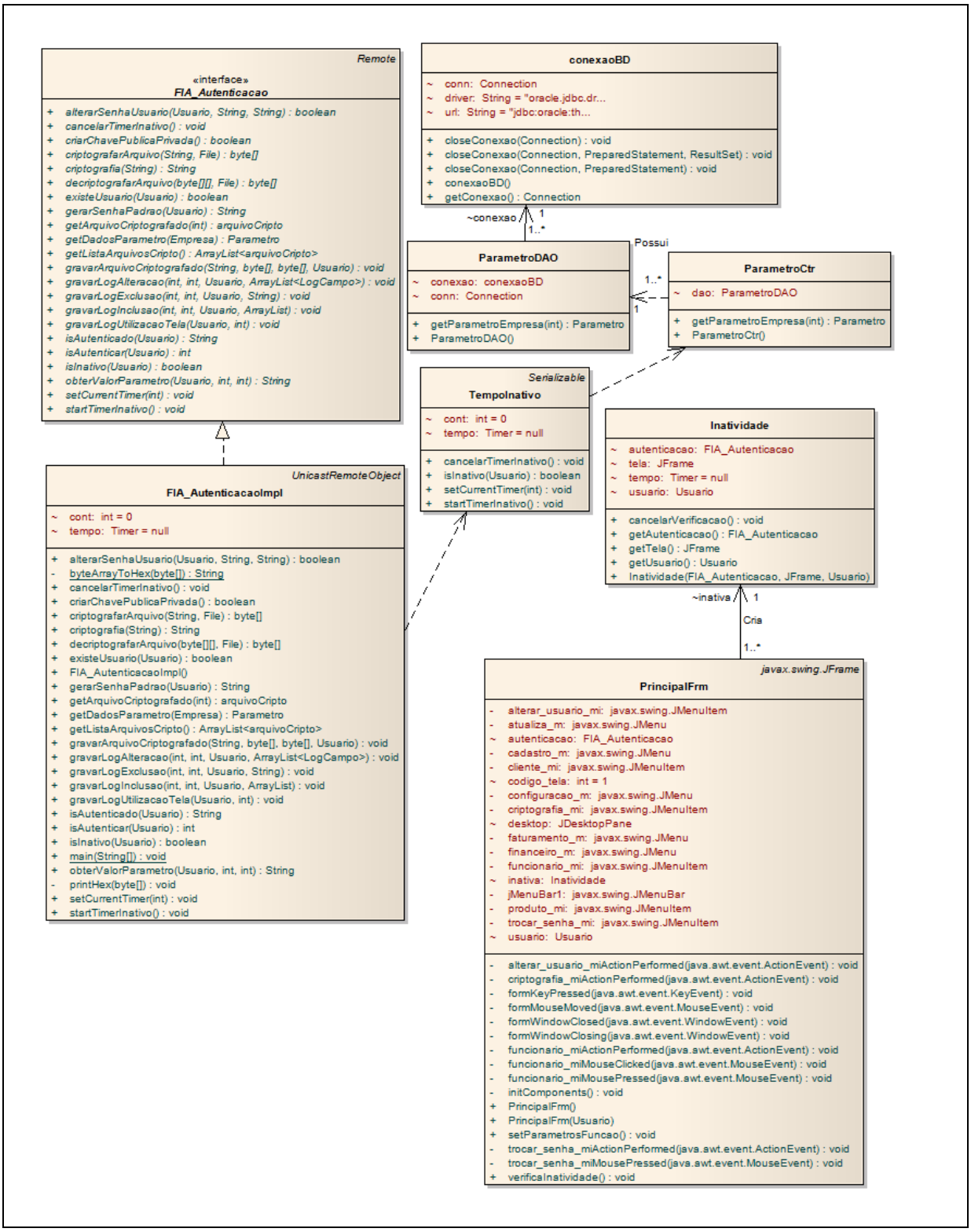


Figura 11 – Diagrama de classe do caso de uso UC005 – Configuração de bloqueio do sistema

A Figura 12 apresenta o diagrama de atividade do caso de uso UC005 – Configuração de bloqueio do sistema. Nesta está descrita o fluxo que o software faz quando realiza o login do sistema, mostrando as etapas de login inválido, login bloqueado e login com sucesso. Além de mostrar o fluxo do software, quando o mesmo fica por algum tempo inativo.

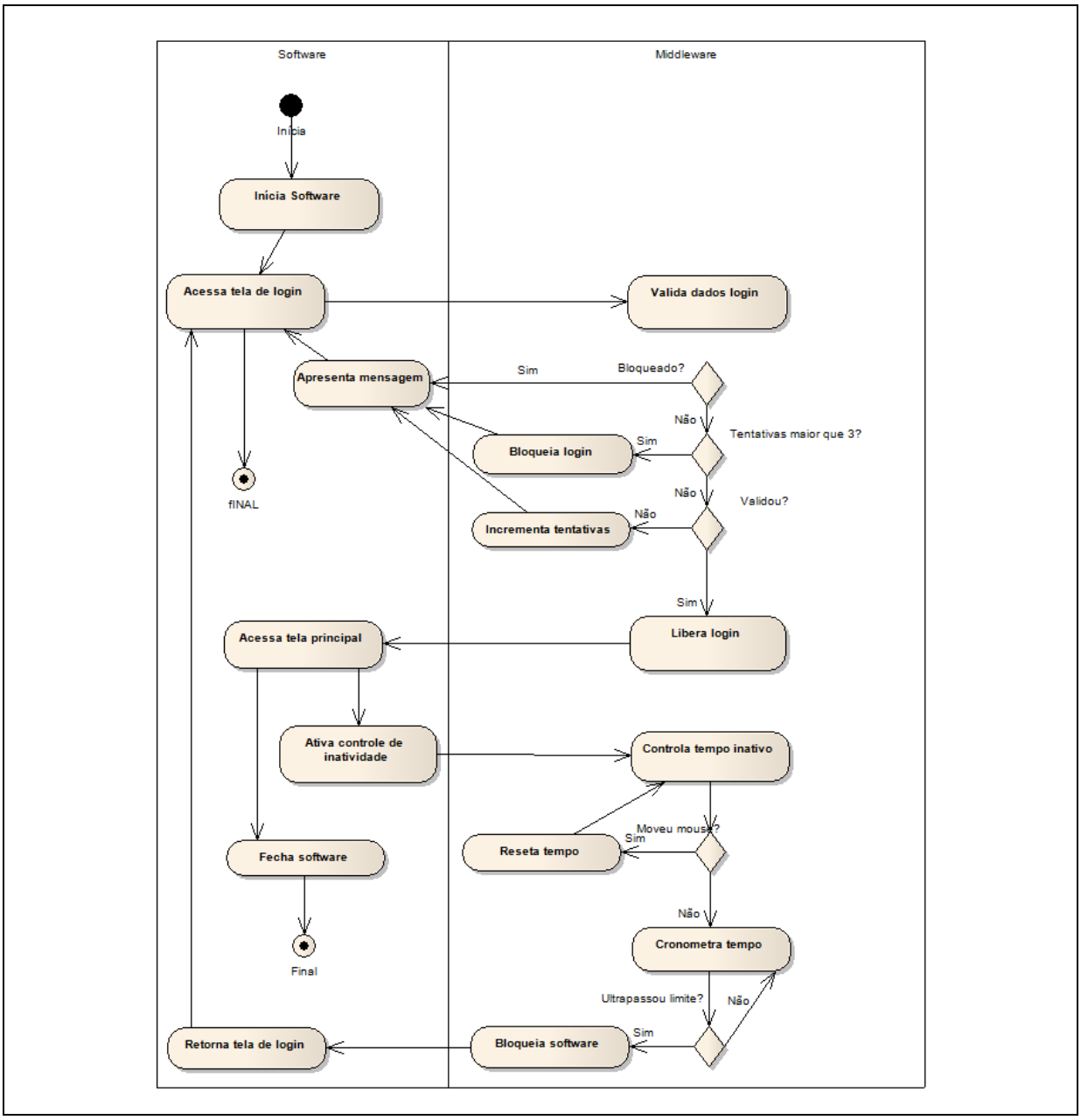


Figura 12 – Diagrama de atividade do caso de uso UC005 – Configuração de bloqueio do sistema

Os casos de uso UC004 – Alterar senha do usuário (Quadro 11) e UC013 – Cadastrar parâmetros refere-se aos itens FIA_SOS.2 – capacidade de gerar senhas e FIA_SOS.1 – métrica mínima das senhas. O administrador de segurança cadastra as métricas mínimas como tamanho e formação da palavra (número e caractere ou só caractere). Ao gerar uma senha ou alterá-la o *middleware* irá consistir se a mesma esta de acordo com as métricas.

UC004 – Alterar senha do usuário: altera a senha do usuário conforme métricas pré-estabelecidas pelo administrador de segurança.	
Requisitos atendidos	RF04
Pré-condições	Estar conectado no *middleware*
Cenário principal	1. Usuário realiza login no software.

	2. Usuário seleciona a opção ‘Trocar Senha’. 3. Software envia os dados para o *middleware.* 4. *Middleware* valida a senha. 5. *Middleware* grava nova senha no banco de dado.
Exceção 1	No passo 2, o usuário digite seus dados atuais errados. 2.5 *Middleware* retorna para o software *false*. 2.6 Software mostra a mensagem ‘Usuário ou senha inválido’.
Exceção 2	No passo 4, o *middleware* valida que a nova senha não confere com as métricas cadastradas. 4.1 *Middleware* retorna para o software *false.* 4.2 Software mostra a mensagem ‘Senha não está de acordo com as regras!’.

Quadro 11 - Detalhamento do caso de uso UC004 - Alterar senha do usuário

3.2.6 Especificação da classe FMT – Gerenciamento de segurança

Os itens atendidos da classe FMT (descrito no anexo A) são:

a) FMT_MOF.1 – gerenciamento de funções de segurança;

b) FMT_MSA.1 – gerenciamento de atributos de segurança;

c) FMT_SMR.1 – papéis de segurança.

Os casos de uso UC010 – Cadastro de usuário, UC011 – Cadastro de grupo, UC012 – Cadastro de software, UC013 – Cadastro de parâmetros (Quadro 12) e UC014 – Cadastro de telas, que são utilizados no controle da segurança. Por estes casos de uso, o administrador consegue cadastrar novos parâmetros para o software, consegue mapear as telas existentes no mesmo, consegue definir atributos de permissão aos usuários e de permissão de acesso ao software.

O caso de uso UC013 – Cadastro de parâmetros, só pode ser alterado pela empresa desenvolvedora do software, pois nesse cadastro fica toda configuração de acesso do sistema. Os parâmetros podem ser definidos por grupo, usuário, software e pelo padrão da empresa desenvolvedora do software.

UC013 – Cadastro de parâmetros: cadastra os parâmetros por usuário definindo a suas permissões	
Requisitos atendidos	RF20
Pré-condições	1. Estar conectado no sistema de gerenciamento de segurança.

	2. Ter permissão de acesso a tela de parâmetros.
Cenário principal	1. Administrador seleciona o usuário. 2. Administrador seleciona a tela. 3. Administrador seleciona o parâmetro. 4. Administrador define o valor do parâmetro. 5. Administrador salva o parâmetro.
Exceção 1	No passo 3, caso o usuário já possua o parâmetro configurado. 3.1 Sistema de gerenciamento mostrará a mensagem 'Usuário já possui o parâmetro informado'.

Quadro 12 – Detalhamento do caso de uso UC013 – Cadastro de parâmetros

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas informações sobre as técnicas e ferramentas utilizadas para implementação do *middleware*, bem como o processo de implementação.

3.3.1 Técnicas e ferramentas utilizadas

O *middleware* foi implementado na linguagem de programação Java, utilizando-se do ambiente de desenvolvimento Netbeans (NETBEANS, 2011). Para gravação dos dados foi utilizado o banco de dados Oracle 10g (ORACLE, 2011). Para desenvolvimento do sistema de gerenciamento de segurança, foi utilizado o ambiente de desenvolvimento Delphi 7.0 (DELPHI, 2010).

Nas seções seguintes são apresentados as técnicas utilizadas, pseudocódigos e trechos de códigos fontes, referentes às implementações do JAVA RMI e das classes da norma ISO/IEC 15.408 (Quadro 4).

3.3.1.1 Implementação do JAVA RMI

Toda funcionalidade do *middleware* depende exclusivamente do funcionamento do RMI. Pois através deste que o software faz a comunicação com o *middleware* e consegue

acessar os métodos implementados para garantir a segurança do mesmo.

Para o JAVA RMI funcionar é necessário existir uma *interface* implementada que estenda da classe *Remote*. Nesta *interface* são inseridos todos os métodos que são utilizados pelo software. No Quadro 13 é apresentado parte da *interface* FIA_Autenticacao.

```
 * @author Bruno Castellani Gucowski
 */
public interface FIA_Autenticacao extends Remote {

    String isAutenticado(Usuario usuario) throws RemoteException;

    int isAutenticar(Usuario usuario) throws RemoteException;

    String criptografia(String senha) throws RemoteException;

    Parametro getDadosParametro(Empresa empresa) throws RemoteException;

    String obterValorParametro(Usuario usuario, int tela, int parametro) throws RemoteException;

    boolean alterarSenhaUsuario(Usuario usuario, String senha,String cripto) throws RemoteException;

    boolean existeUsuario(Usuario usuario) throws RemoteException;

    void startTimerInativo() throws RemoteException;

    void setCurrentTimer(int valor) throws RemoteException;

    boolean isInativo(Usuario usuario) throws RemoteException;

    void cancelarTimerInativo() throws RemoteException;

    String gerarSenhaPadrao(Usuario usuario) throws RemoteException;

    void gravarLogUtilizacaoTela(Usuario usuario, int tela) throws RemoteException;
```

Quadro 13 - Parte do código da *interface* FIA_Autenticacao

Para desenvolver os métodos da *interface* é necessário criar uma classe que implementa a *interface* FIA_Autenticacao. No Quadro 14 é apresentado uma parte da classe FIA_AutenticacaoImpl.

```
    public String gerarSenhaPadrao(Usuario usuario) throws RemoteException {
        UsuarioDAO dao = new UsuarioDAO();
        String senha = dao.gerarSenhaPadrao(usuario);
        dao.gravarSenhaPadrao(usuario, criptografia(senha));
        return senha;
    }

    public void gravarLogUtilizacaoTela(Usuario usuario, int tela) throws RemoteException {
        LogCtr ctr = new LogCtr();
        ctr.gravarLogUtilizacaoTela(usuario, tela);
    }

    public void gravarLogAlteracao(int tipo_log, int codigo_tela, Usuario usuario, ArrayList<LogCampo> lista_campos) throw
        LogCtr ctr = new LogCtr();
        ctr.gravarLogAlteracao(tipo_log, codigo_tela, usuario, lista_campos);
    }

    public void gravarLogExclusao(int tipo_tela, int codigo_tela, Usuario usuario, String ds_campo) throws RemoteException
        LogCtr ctr = new LogCtr();
        ctr.gravarLogExclusao(tipo_tela, codigo_tela, usuario, ds_campo);
    }

    public void gravarLogInclusao(int tipo_log, int codigo_tela, Usuario usuario, ArrayList lista_campos) throws RemoteExc
        this.gravarLogAlteracao(tipo_log, codigo_tela, usuario, lista_campos);
    }
```

Quadro 14 – Parte do código da classe FIA_AutenticacaoImpl

Para o JAVA RMI executar no servidor é necessária a criação de uma nova classe, que fará a instância do JAVA RMI. Esta classe será composta por um objeto chamado *Naming* que indica onde o servidor estará executando. No Quadro 15 é apresentado uma parte da classe FIA_AutenticacaoServer.

```
package rmi;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Empresa;
import model.Usuario;

/**
 *
 * @author NoteBook-Bruno
 */
public class FIA_AutenticacaoServer {
    public FIA_AutenticacaoServer(){

        try {
            FIA_Autenticacao autenticacao = new FIA_AutenticacaoImpl();
            Naming.rebind("rmi://localhost:1099/AutenticacaoServer", autenticacao);
        } catch (MalformedURLException ex) {
            Logger.getLogger(FIA_AutenticacaoServer.class.getName()).log(Level.SEVERE, null, ex);
        } catch (RemoteException ex) {
            Logger.getLogger(FIA_AutenticacaoServer.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static void main (String args[]){
        new FIA_AutenticacaoServer();
    }
```

Quadro 15 – código da classe FIA_AutenticacaoServer

Após a instância da classe FIA_AutenticacaoServer, qualquer outro software

implementado em Java poderá chamar as rotinas criadas dentro do *middleware*.

3.3.1.2 Implementação da classe FIA – autenticação

Para garantir a autenticidade e a identificação do usuário que está conectando no software, foi criado o método isAutenticado na classe FIA_AutenticacaoImpl. Este método é responsável pela validação do login do usuário como, por exemplo, dados inválidos, tentativas falhas e bloqueio da máquina. Além de realizar a gravação de trilhas de auditoria com informações da tentativa de login. No Quadro 16 apresenta a implementação do método validaAutenticacao da classe LoginDAO, que é chamado dentro do método isAutenticado.

```
public String validaAutenticacao(Usuario usuario) {
    try {
        int tentativa = 0;
        String login = null;
        InetAddress inet = InetAddress.getLocalHost();
        String maquina = inet.getHostName();

        conexaoBD conexao = new conexaoBD();
        //Valida dados do usuário
        Connection conn = conexao.getConexao();
        CallableStatement cs = conn.prepareCall("{call VALIDAR_USUARIO_LOGIN(?,?,?,?,?,?)}");
        cs.setInt(1, usuario.getEmpresa().getCodigo());
        cs.setString(2, usuario.getLogin());
        cs.setString(3, usuario.getSenha());
        cs.setString(4, maquina);
        cs.registerOutParameter(5, java.sql.Types.INTEGER);
        cs.registerOutParameter(6, java.sql.Types.VARCHAR);
        cs.execute();
        tentativa = cs.getInt(5);
        login = cs.getString(6);
        cs.close();
        cs = conn.prepareCall("{call INSERIR_LOG_LOGIN(?,?,?,?,?)}");
        cs.setInt(1, usuario.getEmpresa().getCodigo());
        cs.setString(2, usuario.getLogin());
        cs.setString(3, maquina);
        cs.setInt(4, tentativa);
        cs.setString(5, login);
        cs.execute();
        conexao.closeConexao(conn);
```

Quadro 16 – Parte do código do método validaAutenticacao

No Quadro 17 apresenta a classe LoginFrm, que faz o tratamento final do retorno do método isAutenticado. Se o login foi realizado com sucesso é por que o retorno foi ‘S’, caso os dados fornecidos forem inválidos, o retorno será ‘N’ e caso as tentativas de login superem o máximo permitido nos parâmetros do *middleware*, o retorno será ‘B’. Caso o retorno seja ‘B’ o usuário estará bloqueado para realizar login na máquina por um determinado tempo, que é estipulado nos parâmetros do *middleware*.

```
        usuario.setEmpresa(empresa);
        usuario.setLogin(usuario_ed.getText());
        String senha = new String(senha_ed.getPassword());
        System.out.println("Senha:     " + autenticacao.criptografia(senha));
        usuario.setSenha(autenticacao.criptografia(senha));
        String login = autenticacao.isAutenticado(usuario);
        if (login.equals("S")) {
            String varVisualizarTela = autenticacao.obterValorParametro(usuario, 1, 1);
            if (varVisualizarTela.equals("S")) {
                new PrincipalFrm(usuario).setVisible(true);
                dispose();
                  JOptionPane.showMessageDialog(null, "Logado!");
            } else{
                JOptionPane.showMessageDialog(null, "Permissão negada");
            }

        } else if (login.equals("N")) {
            JOptionPane.showMessageDialog(null, "Usuário ou senha inválido!");
        } else if (login.equals("B")) {
            Parametro param = autenticacao.getDadosParametro(empresa);
            JOptionPane.showMessageDialog(null, "Limite de tentativa ultrapassado. Login bloqueado por " +
                                          param.getTempo_espera());
        }
    } catch (RemoteException ex) {
        Logger.getLogger(LoginFrm.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Quadro 17 – parte do código responsável pela validação do usuário na classe LoginFrm

Para maior segurança do software, foi implementado a rotina de geração automática de senha, caso o usuário venha a esquecer de sua senha, o administrador de segurança poderá gerar uma nova senha com base nas métricas cadastradas nos parâmetros do *middleware*. No Quadro 18 apresenta uma parte da implementação do método alterarSenha da classe UsuarioDAO. Este método possui três parâmetros:

a) usuário: parâmetro da classe Usuario que contém os dados referente ao login;
b) senha: parâmetro que contém a senha digita pela usuário. Utilizado para realizar a verificação da métrica estabelecida;
c) cripto: parâmetro que contém a senha já criptografada.

```
public boolean alterarSenha(Usuario usuario, String senha,String cripto) {
    try {
        int cod_usuario = getCodigoUsuario(usuario);
        conn = conexao.getConexao();
        PreparedStatement stm = conn.prepareStatement("Select CONSISTIR_NOVA_SENHA(?,?,?) from dual");
        stm.setInt(1, cod_usuario);
        stm.setString(2, senha);
        stm.setInt(3, usuario.getEmpresa().getCodigo());
        ResultSet rs = stm.executeQuery();
        String valida = "";
        while (rs.next()) {
            valida = rs.getString(1);
        }

        if (valida.equals("S")) {
            CallableStatement cs = conn.prepareCall("{call GRAVAR_ALTERACAO_SENHA(?,?)}");
            cs.setInt(1, cod_usuario);
            cs.setString(2, cripto);
            cs.execute();
        }

        conexao.closeConexao(conn, stm, rs);
        return valida.equals("S");

    } catch (SQLException ex) {
        Logger.getLogger(UsuarioDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
    return false;
```

Quadro 18 – Parte do código do método alterarSenha

3.3.1.3 Implementação da classe FCS – criptografia

No *middleware* existem duas formas de realizar a criptografia a MD5 e a criptografia *Advanced Encryption System* (AES). O MD5 é utilizado para criptografar apenas pequenos dados, como por exemplo, as senhas do usuário. Este procedimento é realizado através do método criptografarSenha (Quadro 19) da classe CriptografiaCtr.

A criptografia através do AES é utilizada no *middlware* para criptografar arquivos (Quadro 20) e gravá-los em banco de dados. Para realizar este procedimento primeiro o *middleware* criptografa o arquivo com o AES e depois assina o arquivo com a chave pública gerada pelo RSA.

```
public String criptografaSenha(String senha) {
    String sen = "";
    MessageDigest md = null;
    try {
        md = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(CriptografiaCtr.class.getName()).log(Level.SEVERE, null, ex);
    }
    byte[] hash = md.digest(senha.getBytes());
    sen = byteArrayToHex(hash);
    return sen;
}
```

Quadro 19 – Código do método criptografaSenha

```
public class Cifrador {

    public byte[][] cifra(PublicKey pub, byte[] textoClaro) throws NoSuchAlgorithmException,
            NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException,
            BadPaddingException, InvalidAlgorithmParameterException {
        byte[] textoCifrado = null;
        byte[] chaveCifrada = null;

        //-- A) Gerando uma chave simétrica de 128 bits
        KeyGenerator kg = KeyGenerator.getInstance("AES");
        kg.init(128);
        SecretKey sk = kg.generateKey();
        byte[] chave = sk.getEncoded();
        //-- B) Cifrando o texto com a chave simétrica gerada
        Cipher aescf = Cipher.getInstance("AES/CBC/PKCS5Padding");
        IvParameterSpec ivspec = new IvParameterSpec(new byte[16]);
        aescf.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(chave, "AES"), ivspec);
        textoCifrado = aescf.doFinal(textoClaro);
        //-- C) Cifrando a chave com a chave pública
        Cipher rsacf = Cipher.getInstance("RSA");
        rsacf.init(Cipher.ENCRYPT_MODE, pub);
        chaveCifrada = rsacf.doFinal(chave);

        return new byte[][]{textoCifrado, chaveCifrada};
    }
}
```

Quadro 20 – Código do método cifra

3.3.2 Operacionalidade da implementação

Esta seção tem como objetivo mostrar a operacionalidade do *middleware*, abordando as classes da norma ISO/IEC 15.408 atendidas (Quadro 4) pelo mesmo. Para evidenciar os métodos desenvolvidos no *middleware*, foi desenvolvido um software auxiliar que irá apresentar as funcionalidades e requisitos implementados das classes da ISO/IEC 15.408.

A Figura 13 apresenta a operacionalidade da implementação, mostrando a relação

entre os softwares e o *middleware*. Esta figura contém quatro componentes:

a) sistema gerenciador de segurança: responsável pela auditoria dos dados informados pelo *middleware*, como por exemplo, logs de acesso e arquivos criptografados;
b) banco de dados: responsável por guardar todas as informações referenciadas pelo *middleware*;
c) middleware: responsável por garantir a segurança dos softwares, mantendo a conformidade com a norma ISO/IEC 15.408;
d) softwares utilitários: são aqueles que irão utilizar o *middleware* para garantir a sua segurança.

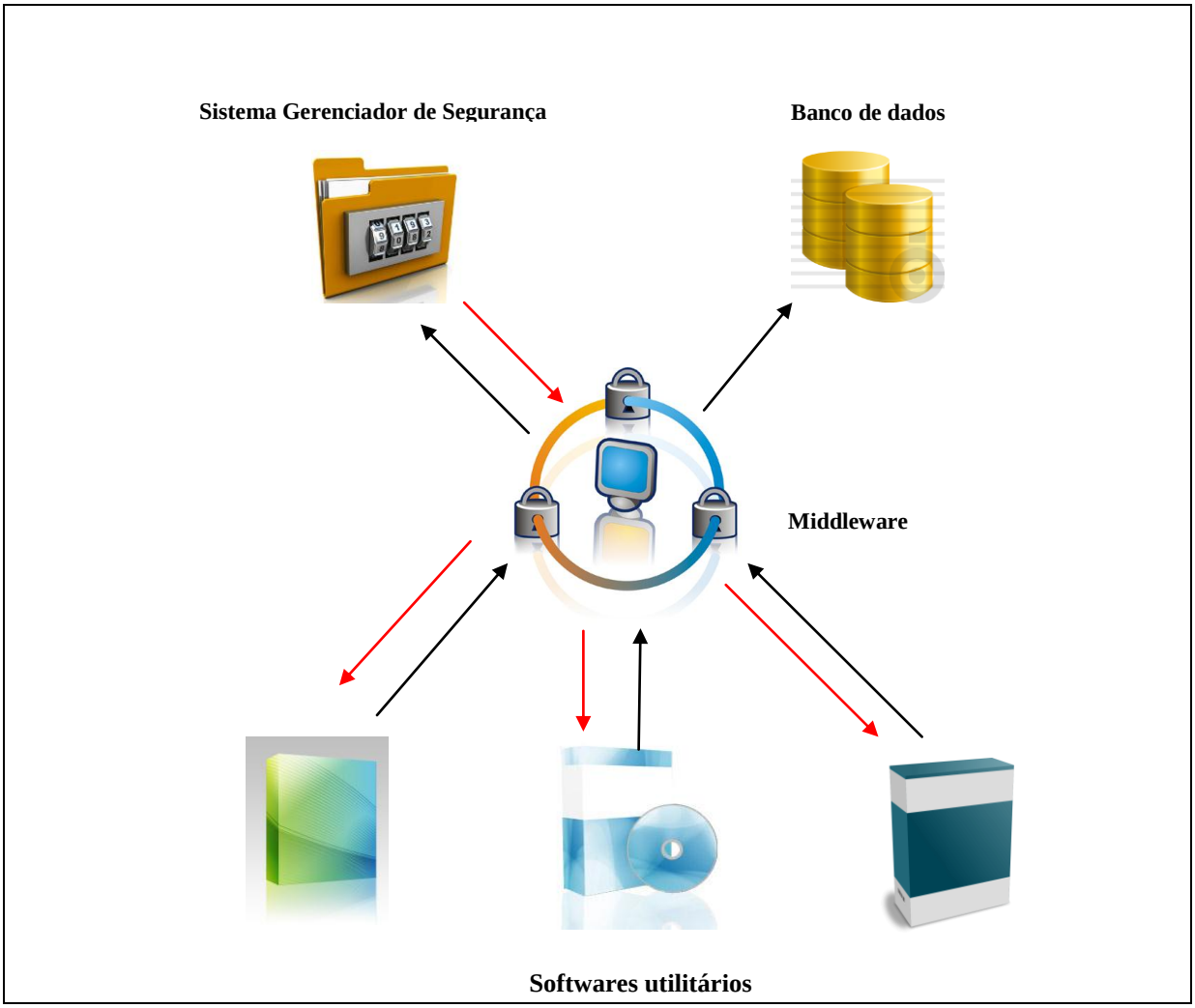


Figura 13 – Estrutura do desenvolvimento do *middleware*

3.3.2.1 Operacionalidade da classe FIA – autenticação

Ao executar a classe LoginFrm é apresentada a *interface* (Figura 14), a qual permite a autenticação do usuário no software.

Figura 14 – Tela Login - Sistema

São apresentados na classe LoginFrm três campos e um botão:

a) usuário: define o usuário que será identificado no software;
b) senha: define a senha do usuário que será identificado no software;
c) empresa: empresa pelo qual o usuário está se autenticando;
d) botão Login: botão que realiza a validação dos dados e acessa o sistema.

Após o clique no botão login o software valida os dados digitados em tela e pode retornar três tipos de avisos:

a) permissão negada: mensagem retornada quando o parâmetro 1 – “Permitir Visualizar Tela Principal” estiver configurado para ‘N’ (Figura 16). Deste modo mesmo que os dados estejam corretos o sistema irá bloquear o usuário. A Figura 15 apresenta a mensagem visualizada no software;

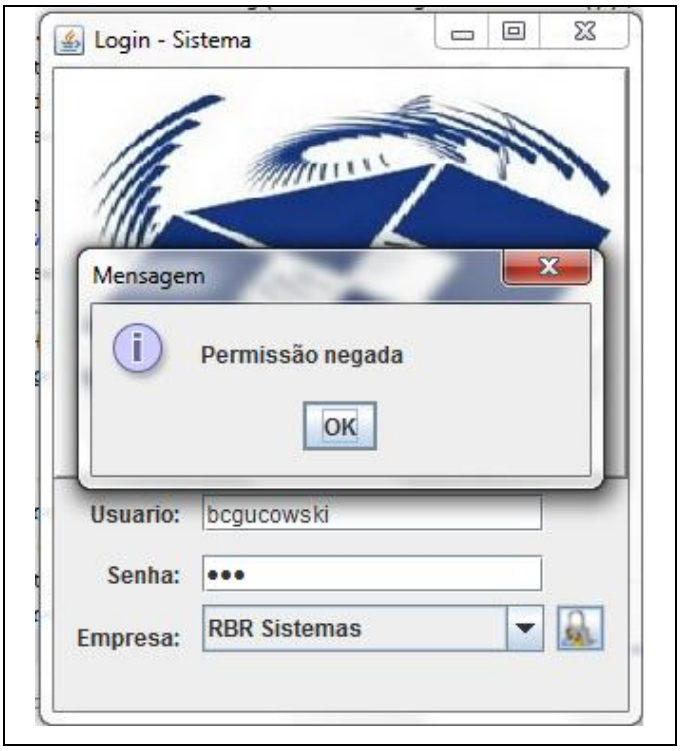


Figura 15 – Mensagem “Permissão negada”

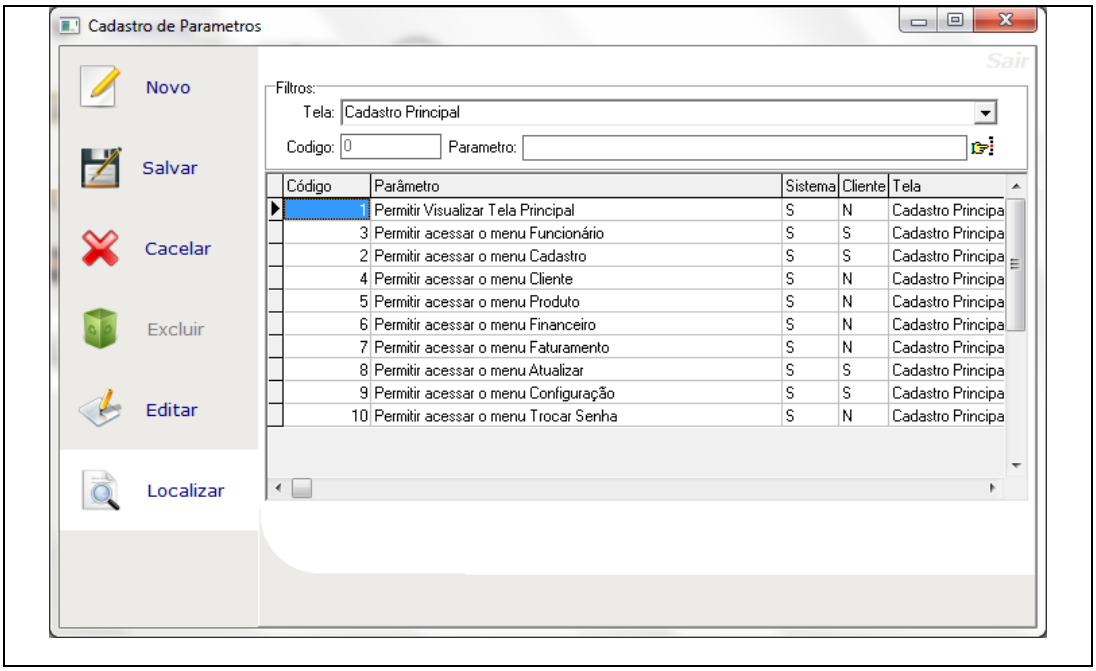


Figura 16 – Tela Cadastro de Parâmetros

b) usuário ou senha inválidos: mensagem retornada quando o *middleware* verifica que a autenticidade dos dados é invalida. A Figura 17 apresenta a mensagem visualizada no software;

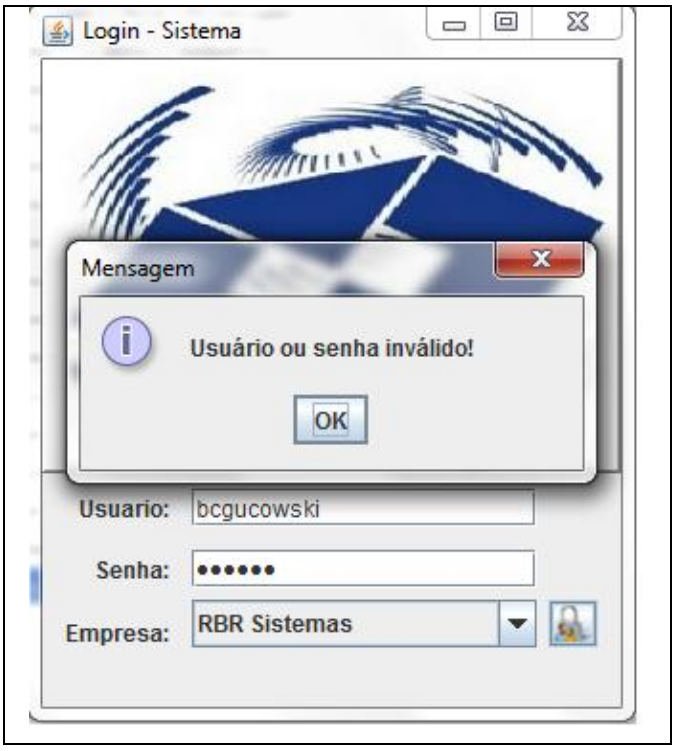


Figura 17 – Mensagem "Usuário ou senha inválido!"

c) limite de tentativas ultrapassado. Login bloqueado por "n" minutos: mensagem retornada pelo *middleware* quando a tentativa de login com um mesmo usuário ultrapasse o limite estipulado nos parâmetros do *middleware*. A Figura 18 apresenta a mensagem visualizada no software e a Figura 19 apresenta a configuração dos parâmetros do *middleware*.

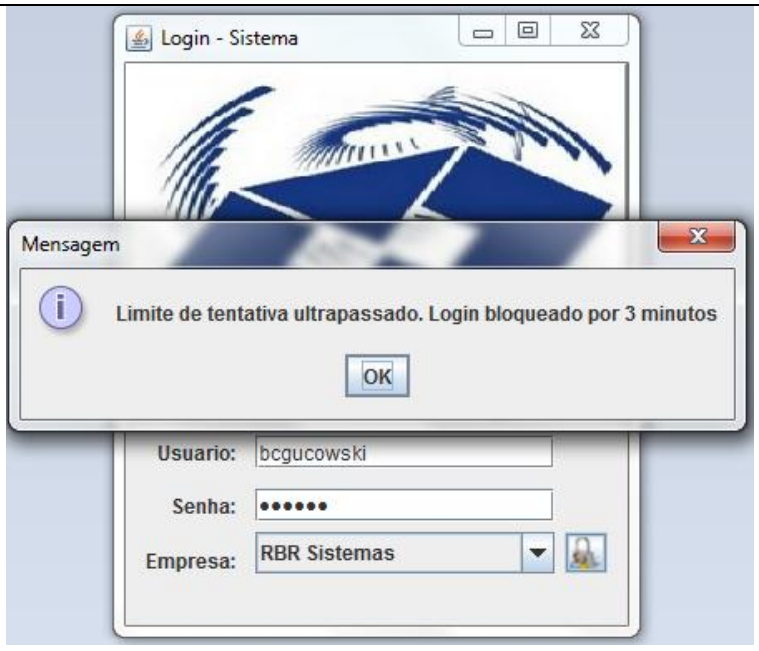


Figura 18 – Mensagem
"Limite de tentativas ultrapassado. Login bloqueado por 3 minutos"

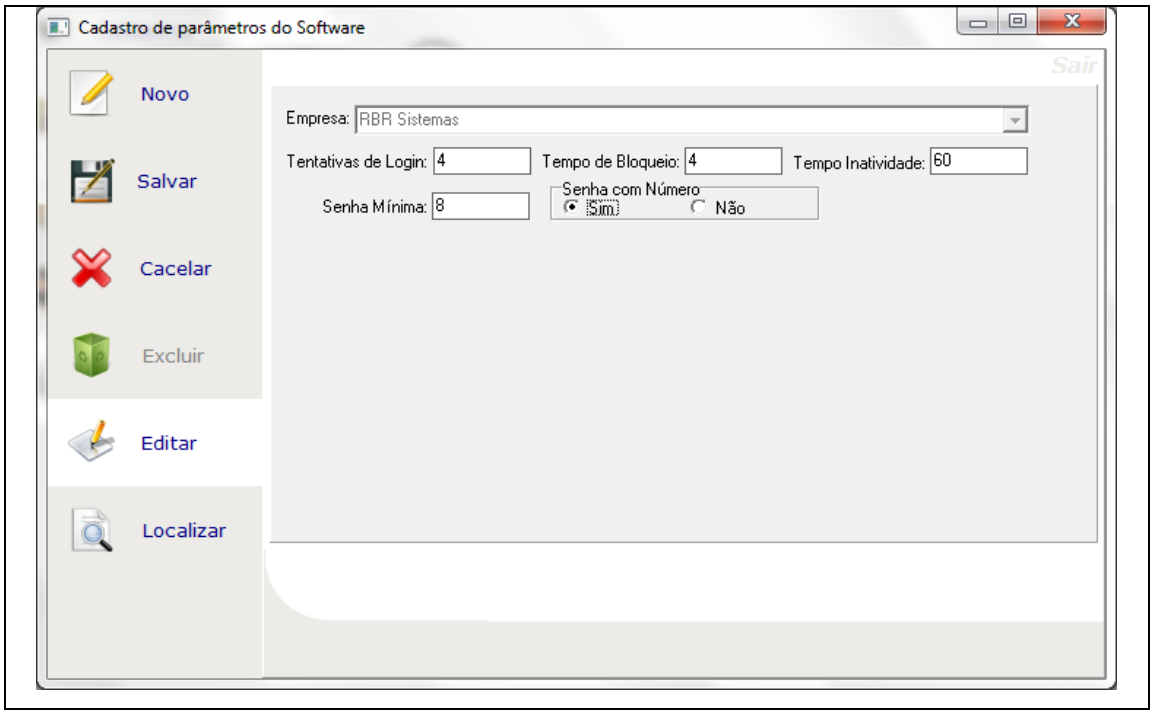


Figura 19 – Tela de cadastro dos parâmetros do Software

3.3.2.2 Operacionalidade da classe FCS – Criptografia

Para atender e exemplificar os itens implementados desta classe foi criado no software a opção de criptografar arquivos e salvá-los em banco de dados. Esta opção pode ser acessada na opção do sistema Configuração-Criptografia (Figura 20). Nesta tela existe a opção de gerar as chaves públicas e privadas através do RSA, que serão utilizadas na criptografia do arquivo com o algoritmo AES. Também é solicitado um nome para cada chave, pois as mesmas ficam gravadas em um arquivo no servidor (Figura 21).

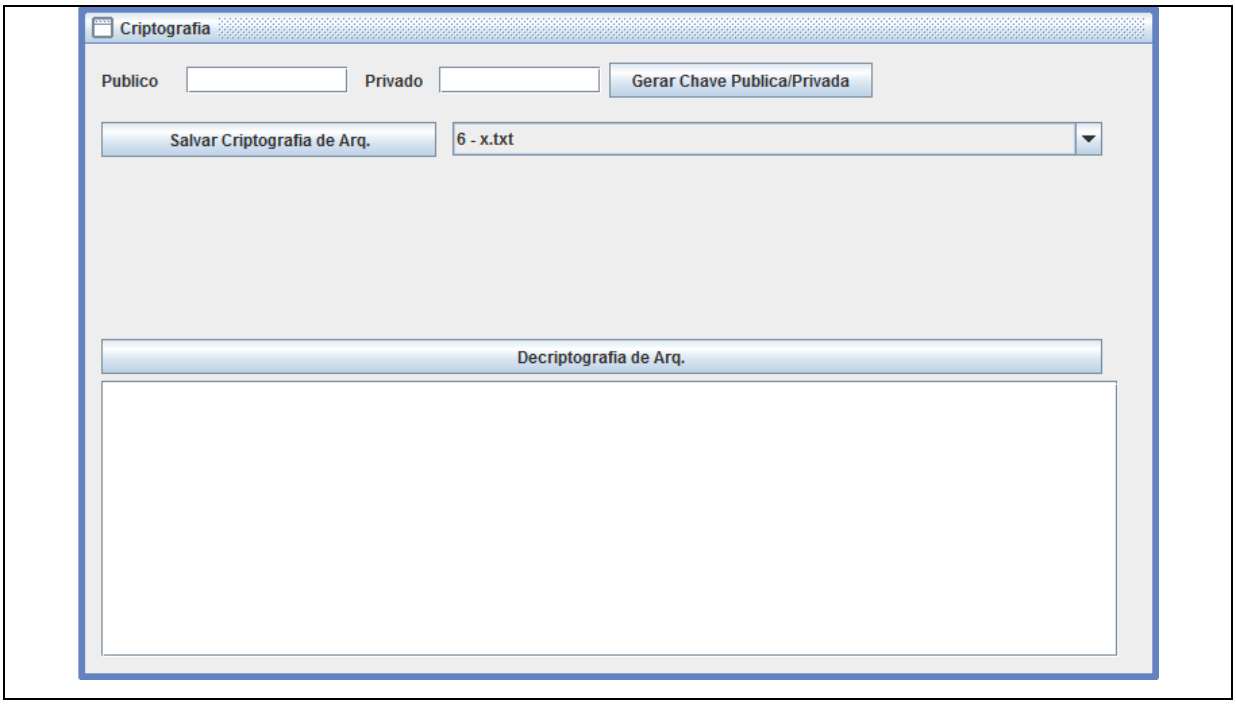


Figura 20 – Tela de Criptografia

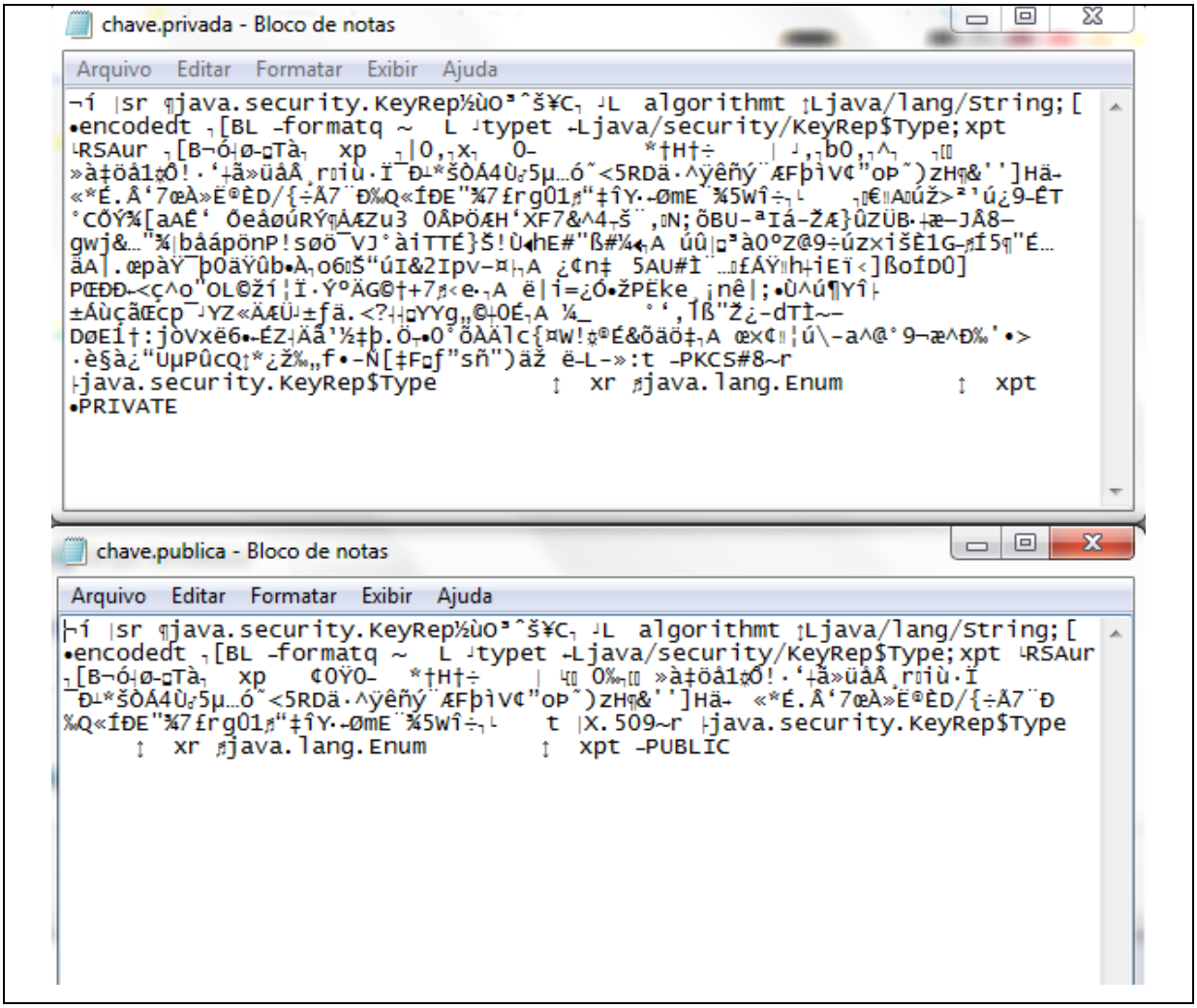


Figura 21 – Arquivos com a chave pública e com a chave privada

A Figura 22 apresenta o arquivo teste.txt, que será criptografado e gravado no banco de dados do *middleware* (Figura 23). Devido ao tamanho do arquivo criptografado, é necessário a utilização de um campo *blob* no banco de dados.

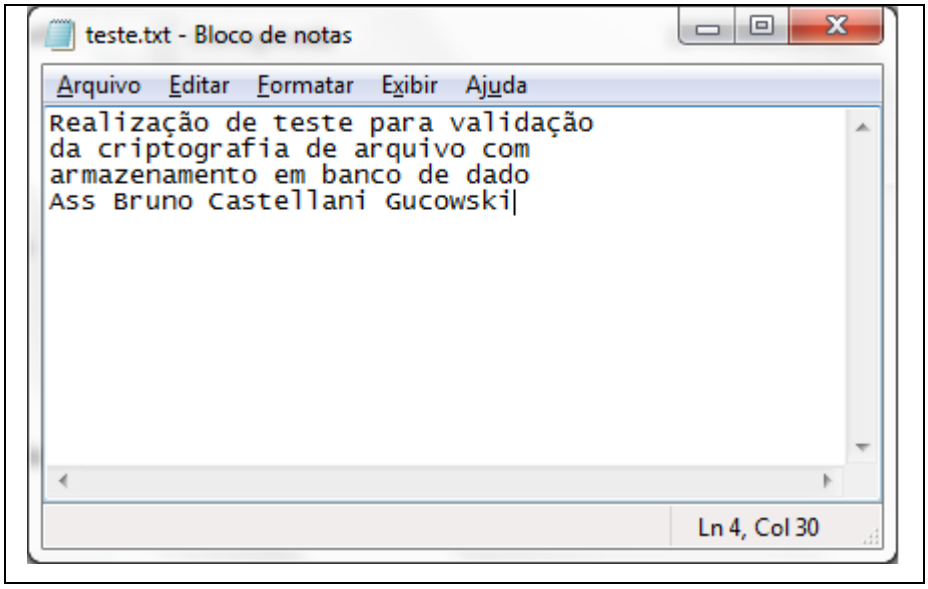


Figura 22 – Arquivo teste.txt

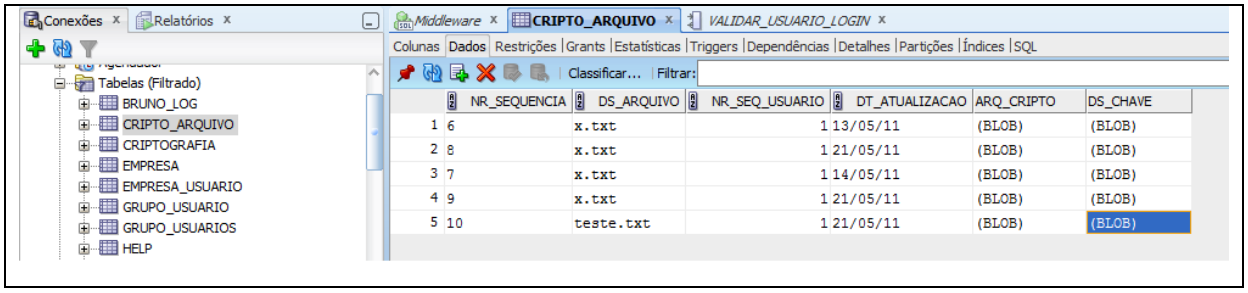


Figura 23 – Arquivo teste.txt gravado no banco de dado do *middleware*

Para decriptografar um arquivo, é necessário selecioná-lo e informar o nome da chave privada que será utilizado para decriptografá-lo. Ao clicar no botão ‘Decriptografia de Arq.’, o software irá capturar no banco de dados do *middleware* o arquivo criptografado e sua chave de criptografia, e irá decriptografá-lo, mostrando na tela o resultado do processo (Figura 24).

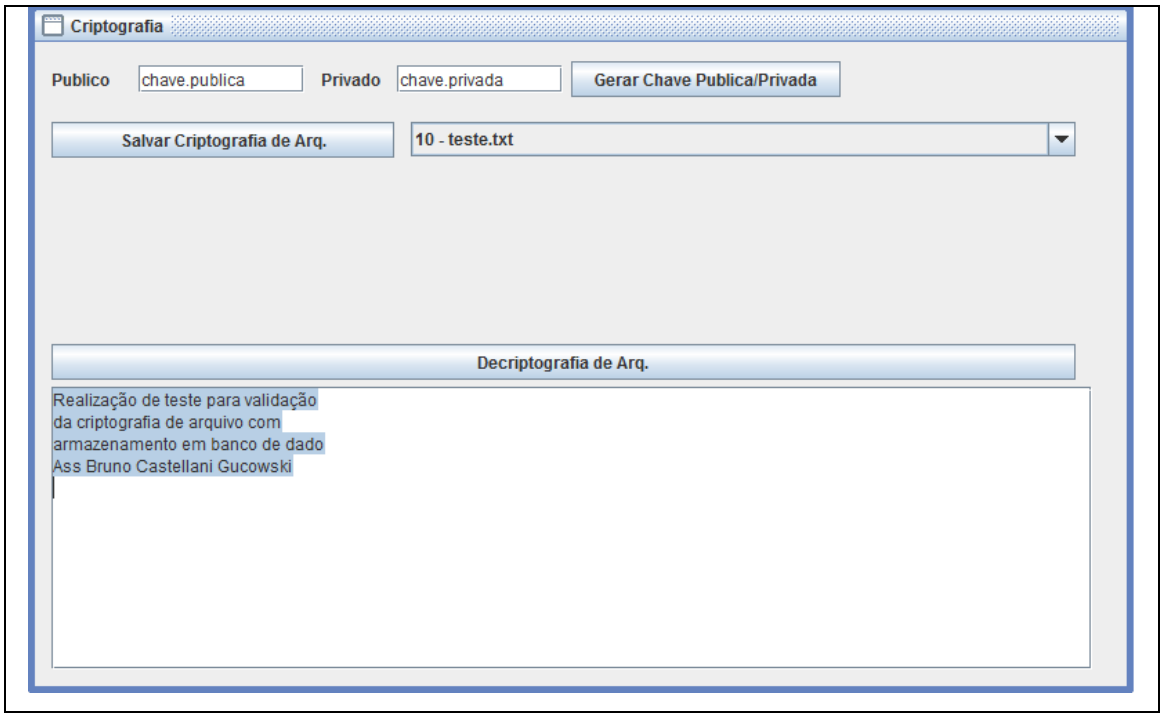


Figura 24 – Arquivo teste.txt decriptografado

3.3.2.3 Operacionalidade da classe FMT - Gerenciamento de segurança

Para atender os itens implementados desta classe, foi criado um sistema de gerenciamento das auditorias, gravadas pelos softwares que utilizam o *middleware*. Este sistema possui os seguintes itens:

a) cadastro de usuários: utilizado pelo administrador de segurança, para criar os usuários que poderão ter acesso aos softwares;
b) cadastro de grupos: utilizado pelo administrador de segurança, para criar os grupos de usuários;
c) cadastro de empresa: utilizado pelo administrador de segurança apenas para consulta, pois é cadastrado apenas pela empresa desenvolvedora do software;
d) cadastro de parâmetros: utilizado pelo administrador de segurança apenas para consulta, pois é cadastrado apenas pela empresa desenvolvedora do software;
e) parâmetros por usuário: utilizado pelo administrador de segurança para cadastrar as permissões de acesso aos usuários e grupo de usuário;
f) log acesso: utilizado pelo administrador de segurança para realizar a consulta das tentativas de acesso ao software;
g) log acesso às telas: utilizado pelo administrador de segurança para realizar as

consultas de acessos às telas do software;

h) log de alteração, inclusão e exclusão: utilizado pelo administrador de segurança para realizar as consultas de alterações de registros do software.

A Figura 25 apresenta a tela de consulta de logs de acesso do sistema. Esta tela tem as opções de poder filtrar os dados conforme necessidade da auditoria.

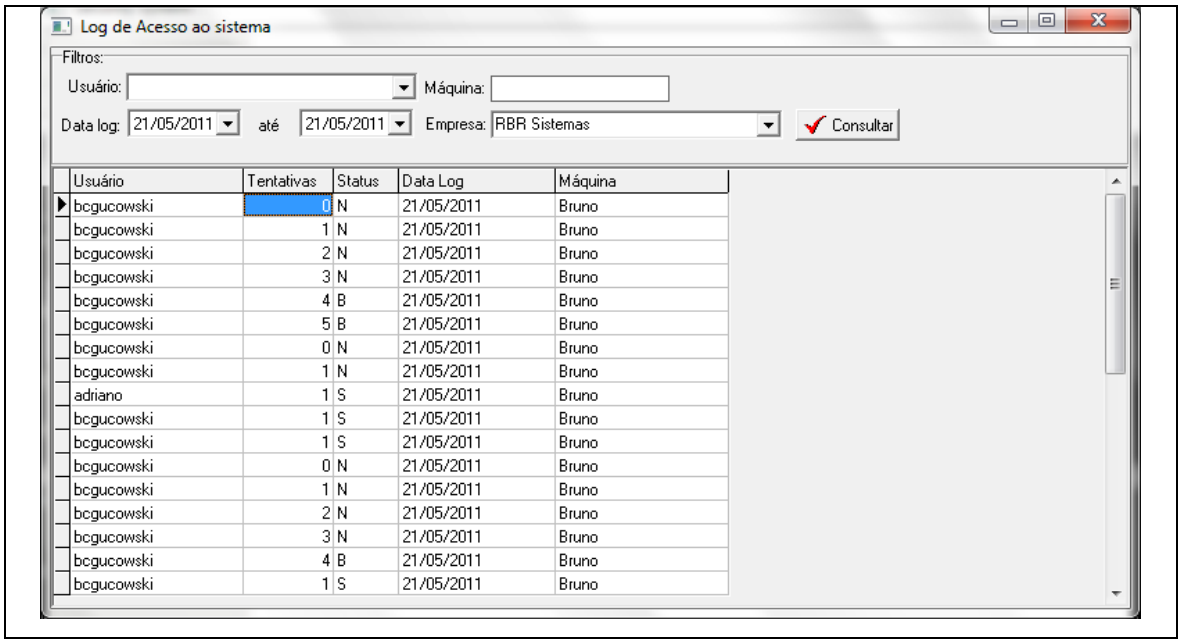


Figura 25 – Tela de log de acesso ao sistema

A Figura 26 apresenta a tela de consulta de acesso às telas do software. Esta tela tem a opção de poder filtrar os dados conforme necessidade da auditoria.

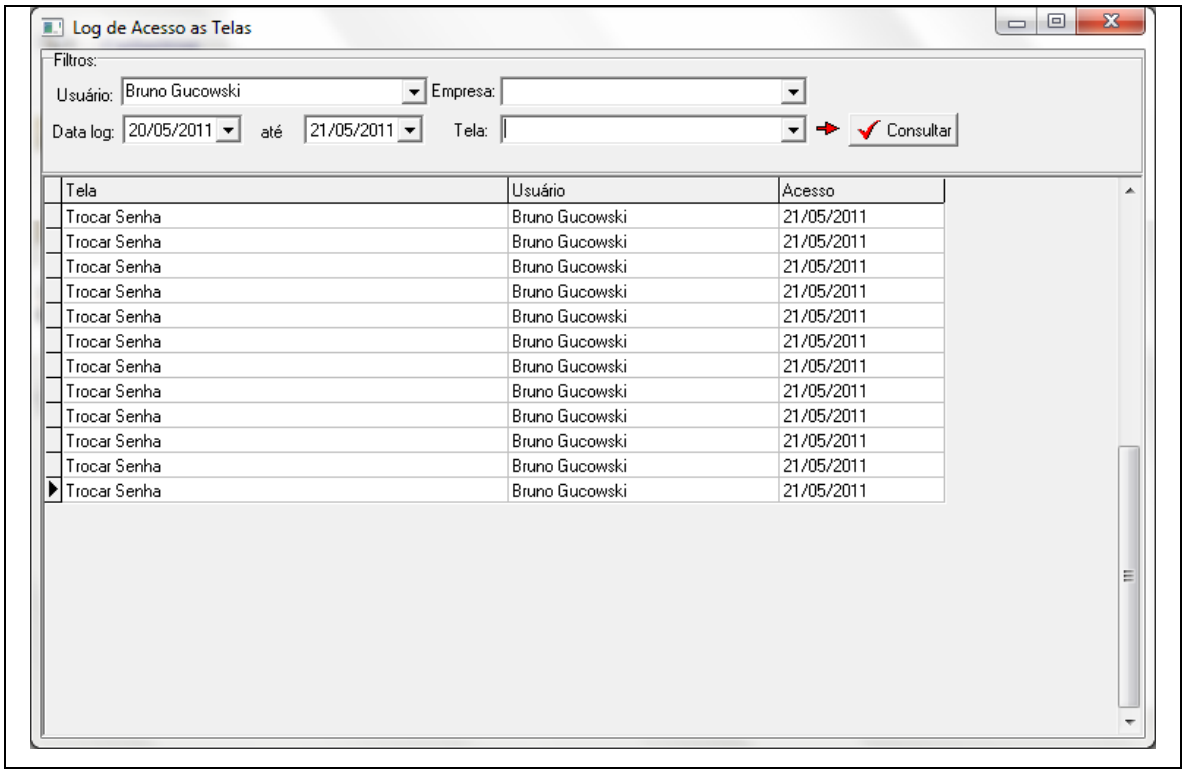


Figura 26 – Tela de log de acesso as telas

3.4 RESULTADOS E DISCUSSÃO

Com o término deste trabalho pode-se verificar que o resultado foi satisfatório, pois o *middleware* foi implementado utilizando os conceitos e aspectos de segurança conforme a ISO/IEC 15.408. Para as empresas desenvolvedoras de software, este *middleware* trará benefícios na produtividade e na qualidade de seus programas.

No Quadro 21 apresentam-se as classes e seus itens de segurança atendidos com o desenvolvimento do *middleware*.

Nome da Classe	Sigla	Itens Atendidos
Proteção de dados do usuário	FDP	FDP_ACF.1 – controle de acesso com base de atributos de segurança
		FDP_ACC.1 – controle de acesso de subconjuntos
		FDP_DAU.2 – autenticidade dos dados com identidade do gerador
		FDP_ROL.1 – Retorno básico
Auditoria	FAU	FAU_GEN.1 – geração de dados para auditoria

		FAU_GEN.2 – associação do usuário ao evento da auditoria
		FAU_SAA.1 – análise de violação potencial
		FAU_SAR.1 – revisão da auditoria
		FAU_SEL.1 – auditoria seletiva
		FAU_STG.1 – armazenamento protegido da trilha de auditoria
		FAU_STG.2 – garantia da disponibilidade dos dados para auditoria
Criptografia	FCS	FCS_COP.1 - operação de criptografia
Autenticação	FIA	FIA_AFL.1 – tratamentos de falhas de autenticação
		FIA_ATD.1 – definição de atributos do usuário para autenticação
		FIA_SOS.1 – métrica mínima das senhas
		FIA_SOS.2 – capacidade de gerar senhas
		FIA_UAU.1 – ações anteriores a autenticação
		FIA_UAU.2 – autenticação do usuário antes de qualquer coisa
		FIA_UAU.6 – re-autenticação
		FIA_UAU.7 – resposta restrita da autenticação
		FIA_USB.1 – ligação do usuário com o sistema
Acesso ao sistema	FTA	FTA_LSA.1 – Limitações de escopo no acesso ao sistema
		FTA_SSL.1 – Travamento automático de sessão
		FTA_SSL.2 – Travamento de sessão por requisição do usuário
Gerenciamento de segurança	FMT	FMT_MOF.1 – Gerenciamento de funções de segurança
		FMT_MSA.1 – Gerenciamento de atributos de segurança
		FMT_MTD.1 – Gerenciamento de dados de segurança

Quadro 21 – Requisitos de segurança implementados no *middleware*

O uso da linguagem Java facilitou a implementação do *middleware*, pois o mesmo possui bibliotecas referentes a RSA que ajudam na implementação da criptografia de dados. Na implementação do sistema de gerenciamento de segurança do *middleware*, foi utilizado o ambiente de desenvolvimento Delphi, por possuir melhores condições de implementação em ambiente *desktop*.

Para gravação das informações do *middleware*, foi utilizado o banco de dados Oracle 10g, que já possui em sua estrutura uma proteção e uma auditoria de todos os dados inseridos nele, garantindo assim, uma maior segurança para os dados gravados.

Em relação aos trabalhos correlatos, o de Carmisini (2010), teve como objetivo a implementação de um *framework* para atender a necessidades do projeto PRB em conformidade com a norma ISO/IEC 15.408. O *middleware* desenvolvido neste projeto em relação ao *framework* desenvolvido por Carmisini obteve como uma de suas diferenças principais o desacoplamento em relação aos softwares, tornando-o mais flexível.

Em relação ao trabalho correlato de Gomes e Santos (2006), verifica-se que foi focado apenas na análise de segurança do ambiente de aprendizado *on-line* da Universidade da Amazônia. Não houve implementação de uma ferramenta de apoio à segurança, apenas foi relatado na forma de relatório os atributos do sistema que não condizem com a norma ISO/IEC 15.408 ou com as boas práticas de programação. No *middleware* além da analise, também foi implementado as rotinas de segurança e o gerenciado de segurança.

O trabalho correlato PASS teve como objetivo identificar na norma ISO/IEC 15.408 os requisitos funcionais que ajudariam os desenvolvedores na construção de softwares mais seguros, e por consequência softwares com maior qualidade. Não houve implementação de uma ferramenta, mas obteve sucesso no modo que transcreveu a ISO/IEC 15.408 para os desenvolvedores.

O Quadro 22 apresenta as classes que possuem no mínimo um item implementado pelo *middleware* em relação aos trabalhos correlatos.

Nome da classe	**Itens Implementados?**			
	Middleware	**Trabalho Carmisini (2010)**	**Trabalho de Gomes e Santos (2006)**	**Trabalho de Nunes – PASS (2007)**
Auditoria	Sim	Sim	Não	Não
Criptografia	Sim	Sim	Não	Não
Autenticação	Sim	Não	Não	Não
Acesso ao sistema	Sim	Não	Não	Não
Gerenciamento de segurança	Sim	Não	Não	Não
Proteção de dados do usuário	Sim	Sim	Não	Não

Quadro 22 – Classes implementadas no *middleware* em relação aos trabalhos correlatos

.

4 CONCLUSÕES

Visto que cada vez mais as pessoas e as empresas dependem dos softwares e esses possuem informações sigilosas de extrema importância para ambos, e que na programação destes softwares ainda hoje não há uma preocupação ou um conhecimento tão grande com relação à segurança da informação, foi desenvolvido o *middleware*. Ele tem como propósito tornar mais abstratas as rotinas de segurança da informação para os desenvolvedores e unificar em apenas um lugar o controle sobre os dados de segurança dos softwares.

Durante este trabalho desenvolveu-se um estudo na área de segurança da informação, principalmente na norma ISO/IEC 15.408. Este estudo trouxe subsídios para a aplicação dos requisitos referentes à segurança da informação no *middleware*.

Como a norma ISO/IEC 15.408 possui muitas classes e cada uma possui muitos itens, foi feito um estudo sobre esses itens e escolhido as classes mais prioritárias e mais utilizadas nos softwares em geral, que são auditoria (FAU), criptografia (FCS), autenticação (FIA), acesso ao sistema (FTA), gerenciamento de segurança (FMT) e proteção de dados do usuário (FDP).

Todas as classes escolhidas foram implementadas no *middleware* de maneira a deixá-las com aspectos multiuso, ou seja, podendo ser executadas por mais de um sistema ao mesmo tempo, garantindo assim o conceito de Sistema Distribuído (SD). Para a classe gerenciamento de segurança foi desenvolvido o sistema de gerenciamento, que faz a integração com o middleware que busca os dados gravados e auditados pelos softwares.

Com relação às técnicas e tecnologias utilizadas para o desenvolvimento do *middleware*, pode-se dizer que foram satisfatórias, pois atenderam as expectativas permitindo que todos os requisitos elicitados fossem atendidos. Para uma melhor adequação da metodologia de criptografia foi utilizado também o algoritmo AES, pois agiliza a criptografia de arquivos maiores.

Com este trabalho pode-se concluir que os objetivos descritos na seção 1.1 foram atendidos. E que o desenvolvimento de um *middleware* que atenda os itens de segurança da informação e torne o desenvolvimento mais ágil e mais prático, é viável. Como a norma ISO/IEC 15.408 é muito extensa, o *middleware* implementado pode ser incrementado para que possa atender completamente os requisitos da norma.

4.1 EXTENSÕES

Como extensões para o presente trabalho propõem-se:

a) análise e implementação de outros itens de segurança segundo a norma ISO/IEC 15.408;
b) desenvolvimento do *middleware* em modo webservice;
c) disponibilizar o *middleware* para aplicações feitas em outras linguagens sem ser o Java.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, Ricardo; RIBEIRO, Bruno. **Segurança no desenvolvimento de software**: como desenvolver sistemas seguros e avaliar a segurança de aplicações desenvolvidas com base na ISO 15.408. Rio de Janeiro: Campus, 2002.

ALVES, Fabrício S.; ALVES, André L. **Aplicabilidade dos principais conceitos de segurança no ciclo de vida de sistemas em software *house***. 2009, 20 f. Monografia (Especialização em Qualidade e Gestão de Softwares) – Curso de Pós-graduação em Qualidade e Gestão de Softwares, Pontifícia Universidade Católica de Goiás, Goiás. Disponível em: <http://www.cpgls.ucg.br/ArquivosUpload/1/File/V%20MOSTRA%20DE%20PRODUO%20CIENTIFICA/EXATAS/8-.PDF> . Acesso em: 14 mar. 2011.

APPLE. **Common criteria**. [S.1.], 2009. Disponível em: <http://www.apple.com/br/support/security/commoncriteria/>. Acesso em: 10 mar. 2011

BARBOSA, Luiz A.; BRAGHETTO, Luiz F.. **RSA**: Criptografia assimétrica e assinatura digital [S.1], 2003. Disponível em: <http://www.braghetto.eti.br/files/Trabalho%20Oficial%20Final%20RSA.pdf>. Acesso em: 23 mar. 2011.

BATISTA, Carlos F. A. **Métrica de segurança de software.** 2007. 97 f. Dissertação (Mestrado em Informática) – Curso de Pós-graduação em Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

BURNETT, Steve; PAINE, Stephen. **Criptografia e segurança**: o guia oficial RSA. Tradução Edson Furmankiewic. Rio de Janeiro: Campus, 2002.

CARMISINI, Andrey. **Aplicação de requisitos de segurança no projeto de rastreabilidade bovina conforme a ISO 15.408**. 2010. 68 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CASTRO, Marcio; RAEDER, Mateus; NUNES, Tiago. **RMI**: uma visão conceitual. [S.1.], abr., 2007. Disponível em: <http://www.inf.pucrs.br/~gustavo/disciplinas/sd/material/Artigo_RMI_Conceitual.pdf>. Acesso em: 23 mar. 2011.

COMMON CRITERIA. **Common criteria for information technology security evaluation**. V. 3.1. [S.l.], 2009. Disponível em: <http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R3.pdf>. Acesso em: 10 mar. 2011.

CRUZ, Edimar F.. **TSDD**: teste de segurança durante o desenvolvimento. In: União Educacional Minas Gerais, 06, 2007, Minas Gerais: Uberlândia. Disponível em: <http://www.testexpert.com.br/files/TSDD%20-%20Teste%20de%20seguran%C3%A7a%20durante%20o%20desenvolvimento.pdf>. Acesso em: 27 fev. 2011.

DATASUS. **Perfil de segurança**: requisitos funcionais de segurança. Rio de Janeiro, 2008. Disponível em: <http://pds.datasus.gov.br/PDS/downloads/PRS-PDS-002-PerfilSegFinanceiros.pdf>. Acesso em: 15 mar. 2011.

DAVIS, Noopur; REDWINE, Samuel. **processes to produce secure software**: towards more secure software. [S.1.], mar. 2004. Disponível em: <www.cigital.com/papers/download/secure_software_process.pdf>. Acesso em: 08 mar. 2011.

DELPHI. [S.1.], 2010. Disponível em <http://www.borland.com>. Acesso em: 15 fev. 2011.

GOMES, Kleiton S.; SANTOS, Newton. **Segurança no desenvolvimento de aplicações web**. 2006. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Tecnológica, Universidade da Amazônia, Belém.

JOHNSON, Ralph E.; FOOTE, Brian. **Designing reusable classes**. Urbana, 1988. Disponível em: <http://www.laputan.org/drc/drc.html>. Acesso em: 10 maio. 2011.

MACIEL, Rita S. P. **Middleware:** uma solução para o desenvolvimento de aplicações distribuídas. CienteFico, Salvador, v. I, [n. 1], p. 1-16, jan./jun. 2004. Disponível em: <http://www.frb.br/ciente/Impressa/Info/I.8.Semiramis.Middleware.pdf>. Acesso em: 10 maio. 2011.

NETBEANS. [S.1.], 2011. Disponível em: <http://netbeans.org/>. Acesso em: 12 fev. 2011.

NOVAK, Mark. **Documentação e avaliação das garantias de segurança de seu aplicativo.** [S.1.], nov. 2006. Disponível em: <http://msdn.microsoft.com/pt-br/magazine/cc163522.aspx>. Acesso em: 09 mar. 2011.

NUNES, Francisco J. B. **PASS:** processo de apoio à segurança de software. 2007. 203 f. Dissertação (Mestrado em Informática Aplicada) – Curso de Pós-graduação em Informática Aplicada, Universidade de Fortaleza, Fortaleza. Disponível em: <http://busca.ibict.br/SearchBDTD/search.do?command=search&q=+assunto:%22INFORM%C3%81TICA%20-%20MEDIDAS%20DE%20SEGURAN%C3%87A%20-%20DISSERTA%C3%87%C3%95ES%22 >. Acesso em: 24 mar. 2011.

OMG. **UML – Unified Modeling Language specification**. Versão 2.0. [S.1.], 2005. Disponível em: <http://www.uml.org>. Acesso em: 16 maio. 2011.

ORACLE. [S.1.], 2011. Disponível em: <http://www.oracle.com/br/index.html>. Acesso em: 22 fev. 2011.

PAULA, Cirilo V. F. F. **Uma abordagem para avaliação da segurança em sistemas de TI**. 2005. 45 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Faculdade de Ciências Aplicadas de Minas, União Educacional Minas Gerais, Uberlândia.

PEREIRA, Ellen V. **Introdução a criptografia RSA**. [S.1], 2006. Disponível em: <http://www.impa.br/opencms/pt/eventos/downloads/jornadas_2006/trabalhos/jornadas_elen_pereira.pdf>. Acesso em: 23 mar. 2011.

SCHIMITT, Marcelo A. R. **Criptografia de chaves públicas**. 2001. Disponível em: <http://www.rnp.br/wrnp2/2001/palestras_middleware/pal_middl_02.pdf>. Acesso em: 23 mar. 2011.

SILVA, Renato P. **Engenharia de software seguro**: segurança em desenvolvimento de software. 2005. 100 f. Monografia (Especialização em Engenharia de Software) – Curso de Pós-graduação em Engenharia de Software, Universidade Candido Mendes, Rio de Janeiro. Disponível em: <http://www.scribd.com/doc/19287132/engenhariasoftwareseguro renatopessanha >. Acesso em: 09 mar. 2011.

SPARXSYSTEMS: Creswick, 2000. Disponível em: <http://www.sparxsystems.com>. Acesso em: 15 maio. 2011.

ANEXO A – Requisitos funcionais de segurança segundo a norma ISO/IEC 15.408

No Quadro 23 são apresentados os requisitos funcionais de segurança segundo a norma ISO/IEC 15.408.

Classe	Itens da classe	Sigla	Atributos dos itens
CLASSE FAU AUDITORIA	FAU_ARP – Respostas automáticas a auditoria	FAU_ARP.1	Alarmes de Segurança
	FAU_GEN – Geração de dados para a auditoria	FAU_GEN.1	Geração de dados para auditoria
		FAU_GEN.2	Associação da identidade do usuário
	FAU_SAA – Análise de auditoria de segurança	FAU_SAA.1	Análise de violação potencial
		FAU_SAA.2	Detecção de anomalia
		FAU_SAA.3	Detecção de ataques simples
		FAU_SAA.4	Detecção de ataques complexos
	FAU_SAR – Revisão de dados de auditoria	FAU_SAR.1	Revisão de auditoria
		FAU_SAR.2	Revisão restrita de auditoria
		FAU_SAR.3	Revisão da trilha de auditoria
	FAU_SEL – Auditoria seletiva	FAU_SEL.1	Auditoria seletiva
	FAU_STG – Armazenamento da trilha de auditoria	FAU_STG.1	Armazenamento protegido da trilha
		FAU_STG.2	Garantia da disponibilidade dos dados
		FAU_STG.3	Ação em caso de possível perda de dados
		FAU_STG.4	Ação em caso de total perda de dados
CLASSE FDP PROTEÇÃO DE DADOS DO USUÁRIO	FDP_ACF – Funções de segurança	FDP_ACF.1	Controle de acesso com base de atributos de segurança
	FDP_ACC – Política de controle de acesso	FDP_ACC.1	Controle de acesso de subconjunto
		FDP_ACC.2	Controle de acesso completo
		FDP_ACC.3	Autenticação básicas dos dados
	FDP_DAU – Autenticação de dados	FDP_DAU.1	Autenticação básica dos dados
		FDP_DAU.2	Autenticação com identidade do gerados
	FDP_ETC – Exportação de dados par fora do controle do sistema	FDP_ETC.1	Exportação de dados sem atributo de segurança
		FDP_ETC.2	Exportação de dados com atributo de segurança

	FDP_IFC – Funções de controle de fluxo de informações	FDP_IFC.1	Controle de fluxo de informação
	FDP_IFF – Política de controle de fluxo de informação	FDP_IFF.1	Atributos simples de segurança
		FDP_IFF.2	Atributos complexos de segurança
		FDP_IFF.3	Fluxo ilícito ilimitado
		FDP_IFF.4	Fluxo ilícito parcialmente eliminado
		FDP_IFF.5	Monitoração do fluxo
	FDP_ITC – Importação de fora do controle das funções de segurança	FDP_ITC.1	Importação de dados sem atributo de segurança
		FDP_ITC.2	Importação de dados com segurança
	FDP_ITT – Transferência interna	FDP_ITT.1	Proteção básica
		FDP_ITT.2	Proteção baseada em atributos
		FDP_ITT.3	Monitoração
		FDP_ITT.4	Monitoração baseada em atributos
	FDP_RIP – Proteção da informação residual	FDP_RIP.1	Proteção por subconjuntos
		FDP_RIP.2	Proteção total
	FDP_ROL - Rollback	FDP_ROL.1	Retorno básico
		FDP_ROL.2	Retorno avançado
	FDP_UIT – Proteção de integridade dos dados do usuário	FDP_UIT.1	Transferência de dados
		FDP_UIT.2	Recuperação com ajuda da origem
		FDP_UIT.3	Recuperação sem ajuda da origem
CLASSE FIA AUTENTICAÇÃO	FIA_AFL – Falhas na autenticação	FIA_AFL.1	Tratamento de falhas na autenticação
	FIA_ATD – Atributos do usuário para autenticação	FIA_ATD.1	Definição dos atributos dos usuários
	FIA_UAU – Autenticação do usuário	FIA_UAU.1	Ações anteriores a autenticação
		FIA_UAU.2	Autenticação do usuário antes de qualquer ação
		FIA_UAU.3	Autenticação aprova de fraude
		FIA_UAU.4	Autenticação de utilização única
		FIA_UAU.5	Múltiplos mecanismos de autenticação
		FIA_UAU.6	Re-autenticação
		FIA_UAU.7	Resposta restrita
	FIA_UID – Identificação do usuário	FIA_UID.1	Ações anteriores a autenticação
		FIA_UID.2	Identificação do usuário
	FIA_USB – Ligação do usuário com	FIA_USB.1	Ligação do usuário com o sistema

	o sistema		
CLASSE FMT GERENCIAMENTO DE SEGURANÇA	FMT_MOF – Gerenciamento de funções de segurança	FMT_MOF.1	Gerenciamento de funções de segurança
	FMT_MSA – Gerenciamento de atributos de segurança	FMT_MSA.1	Gerenciamento de atributos de segurança
		FMT_MSA.2	Segurança dos atributos
		FMT_MSA.3	Inicialização dos atributos
	FMT_MTD – Gerenciamento de dados de segurança	FMT_MTD.1	Gerenciamento de dados
		FMT_MTD.2	Gerenciamento do limite os dados
	FMT_SMF – Especificação do gerenciamento de funções	FMT_SMF.1	Especificação do gerenciamento
	FMT_SMR – Papéis de gerenciamento de segurança	FMT_SMR.1	Papéis de segurança
		FMT_SMR.2	Restrições nos papéis
		FMT_SMR.3	Incorporação de papéis
CLASSE FPT AUTOPROTEÇÃO	FPT_AMT – Teste da camada subjacente	FPT_AMT.1	Teste da camada subjacente
	FPT_ITA – Disponibilidade de dados exportados	FPT_ITA.1	Disponibilidade de dados exportados
	FPT_ITI – Integridade dos dados exportados	FPT_ITI.1	Detecção de modificações
	FPT_PHP – Proteção física do sistema	FPT_PHP.1	Detecção passiva
		FPT_PHP.2	Notificação automática
		FPT_PHP.3	Resistência a ataques físicos
	FPT_RCV – Recuperação segura	FPT_RCV.1	Recuperação manual
		FPT_RCV.2	Recuperação automática
		FPT_RCV.3	Recuperação sem perdas
		FPT_RCV.4	Recuperação de função
	FPT_TRC – Consistência de dados replicados	FPT_TRC.1	Consistências de dados replicados
	FPT_TST - Autoteste	FPT_TST.1	Autoteste
CLASSE FRU UTILIZAÇÃO DE RECURSOS	FRU_PRS – Prioridade de serviços	FRU_PRS.1	Cota máxima de utilização
CLASSE FCS CRIPTOGRAFIA	FCS_CKM – Gerenciamento de chaves de criptografia	FCS_CKM.1	Geração de chaves de criptografia
		FCS_CKM.2	Distribuição de criptografia
		FCS_CKM.3	Acesso as chaves
		FCS_CKM.4	Distribuição de chaves
CLASSE FTA ACESSO AO	FTA_LSA – Limitação de escopo ao sistema	FTA_LSA.1	Limitação de escopo ao sistema

SISTEMA	FTA_MCS – Limitações de número de seções simultâneas	FTA_MCS.1	Limitação básica
		FTA_MCS.2	Limitações por usuário
	FTA_SSL – Travamento sessão	FTA_SSL.1	Travamento automático
		FTA_SSL.2	Travamento pelo usuário
		FTA_SSL.3	Encerramento automático
	FTA_TAB – Mensagem de acesso	FTA_TAB.1	Mensagem de acesso
	FTA_TAH – Histórico de acesso	FTA_TAH.1	Histórico de acesso
	FTA_TSE – Limitação de acesso ao sistema	FTA_TSE.1	Limitação de acesso ao sistema
CLASSE FPR PRIVACIDADE	FPR_ANO - Anonimato	FPR_ANO.1	Anonimato
		FPR_ANO.2	Anonimato sem identificação pelo sistema
	FPR_PSE - Pseudônimo	FPR_PSE.1	Pseudônimo
		FPR_PSE.2	Pseudônimo reversível
		FPR_PSE.3	Formação do pseudônimo
CLASSE FTP CANAIS SEGUROS	FTP_ITC – Canais confiáveis entre funções de segurança	FTP_ITC.1	Canais confiáveis entre funções de segurança
	FTP_TRP – Caminho confiável	FTP_TRP.1	Caminho confiável

Fonte: adaptado de Albuquerque e Ribeiro (2002, p. 61-238)

Quadro 23 – Requisitos funcionais de segurança segundo a norma ISO/IEC 15.408