UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

IMPLEMENTAÇÃO DE REQUISITOS DE SEGURANÇA PARA O PROJETO DE RASTREABILIDADE BOVINA CONFORME A ISO 15.408

ANDREY CARMISINI

BLUMENAU
2010

2010/1-04

ANDREY CARMISINI

IMPLEMENTAÇÃO DE REQUISITOS DE SEGURANÇA PARA O PROJETO DE RASTREABILIDADE BOVINA CONFORME A ISO 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
2010

2010/1-04

IMPLEMENTAÇÃO DE REQUISITOS DE SEGURANÇA PARA O PROJETO DE RASTREABILIDADE BOVINA CONFORME A ISO 15.408

Por

ANDREY CARMISINI

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. Mauro Marcelo Mattos, Dr. – FURB
	_____
Membro:	Prof. Adilson Vahldick, Ms. – FURB

Blumenau, 01 de julho de 2010

Dedico este trabalho aos meus pais, Antonio e Marlene, e a todos os amigos que colaboraram para a sua realização.

AGRADECIMENTOS

A Deus, pela saúde a mim concedida.

À minha família, pelo amor e força que me deram durante a realização deste trabalho.

Aos meus amigos, pela amizade sempre constante e pelos momentos de diversão e apoio.

Ao meu orientador, Paulo Fernando da Silva, por ter acreditado na minha capacidade, me orientando para a conclusão deste trabalho.

Ao Prof. Dr. Mauro Marcelo Mattos, coordenador do Laboratório de Desenvolvimento e Transferência de Tecnologia, que concedeu a oportunidade da realização deste trabalho.

A todos que contribuíram, indiretamente, para a realização deste trabalho.

RESUMO

O Laboratório de Desenvolvimento e Transferência de Tecnologia (LDTT) - da Universidade Regional de Blumenau (FURB) - desenvolve o Projeto de Rastreabilidade Bovina (PRB) e um dos requisitos estabelecidos envolve os aspectos de segurança da informação. Assim, o presente trabalho descreve as técnicas envolvidas no desenvolvimento de um *framework* responsável pela segurança no PRB. Baseando-se na norma *International Organization for Standardization* / *International Electrotechnical Commission* (ISO/IEC) 15.408, conhecida pelo nome *Common Criteria for Information Technology Security Evaluation*, são descritos os aspectos de segurança da informação e o *Common Criteria* responsável pela elaboração da norma ISO/IEC 15.408. Para o desenvolvimento do *framework* utilizou-se o ambiente de programação Eclipse 3.4 juntamente com a API JDBC e a biblioteca XStream. Foram implementados alguns itens das respectivas classes/famílias da norma ISO/IEC 15.408, como é o caso das classes criptografia, proteção de dados do usuário, auditoria, autoproteção e canais seguros.

Palavras-chave: ISO/IEC 15.408. Segurança da informação.

ABSTRACT

The Laboratory of Development and Technology Transfer (LDTT) - at Universidade Regional de Blumenau (FURB) - develops Project Bovine Traceability (PRB) and the requirements surrounding the aspects of information security. Thus, this work describes the techniques involved in developing a framework responsible for security in PRB. Based on standard International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) 15.408 known by name Common Criteria for Information Technology Security Evaluation, describes the aspects of Information Security and Common Criteria responsible the preparation of ISO/IEC 15.408. For development of the framework used the environment Programming with Eclipse 3.4 and the JDBC API and XStream library. Were implemented some of the items respective classes/families of ISO/IEC 15.408, as is the case of class encryption, data protection user, audit, self-protection and secure channels.

Key-words: ISO/IEC 15.408. Information security.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API - *Application Programming Interface*

CC - *Common Criteria*

EAL - *Evaluation Assurance Level*

FURB - Universidade Regional de Blumenau

HTTPS - *HyperText Transfer Protocol Secure*

IP - Internet *Protocol*

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

JDBC - Java DataBase Connectivity

LDTT - Laboratório de Desenvolvimento e Transferência de Tecnologia

MAC - *Media Access Control*

PASS - Processo de Apoio à Segurança de Software

PRB - Projeto de Rastreabilidade Bovina

RF - Requisito Funcional

RNF - Requisito Não Funcional

ST - *Security Target*

TOE - *Target Of Evaluation*

UML - *Unified Modeling Language*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Na sociedade da inform0ação, segundo Dias (2000, p. 40), "[...] ao mesmo tempo em que as informações são consideradas o principal patrimônio de uma instituição, estão também sob constante risco, como nunca estiveram antes." Com isso, a segurança tornou-se um ponto de suma importância para a sobrevivência das organizações.

Na época em que as informações eram armazenadas apenas em papel, a segurança era considerada simples. Era necessário apenas trancar os documentos em algum lugar e restringir o acesso físico. Com a chegada dos computadores pessoais e das redes de computadores os aspectos de segurança tornaram-se complexos, sendo necessário o desenvolvimento de equipes cada vez mais especializadas (DIAS, 2000, p. 40).

A segurança nunca foi hábito dos desenvolvedores de aplicativos, pois os próprios consumidores não se importavam em agregar segurança. Isso significava investir para incluir recursos que não ajudavam nas vendas. Hoje em dia, vários clientes exigem segurança em seus produtos (BURNERTT; PAINE, 2002, p. XVII).

Segundo Paula (2005, p. 10), existem muitos motivos que levam uma organização a proteger as suas informações. Um destes motivos leva em consideração que criar, encontrar ou armazenar informações custa dinheiro e, portanto, a sua perda resulta em prejuízo. "Logo a informação é importante para a organização e para seus negócios, assim como também é importante para os seus concorrentes, o que a torna alvo preferido para sabotadores, espiões industriais e vários outros tipos de golpistas."

Uma destas organizações é o Laboratório de Desenvolvimento e Transferência de Tecnologia (LDTT) da Universidade Regional de Blumenau (FURB) que está desenvolvendo o Projeto de Rastreabilidade Bovina (PRB) e neste projeto um dos requisitos estabelecidos envolve os aspectos de segurança da informação.

Baseado na *International Organization for Standardization / International Electrotechnical Commission* (ISO/IEC), estudos foram realizados sobre a norma 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 *framework*[1] na área de segurança da informação para suprir os requisitos no referido projeto.

[1] *Framework* – é um conjunto de classes que realizam determinadas soluções para a aplicação (JOHNSON; FOOTE, 1988).

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de um *framework* que implemente os atributos de segurança da informação, segundo a norma ISO/IEC 15.408, necessários ao PRB.

Os objetivos específicos do trabalho são:

a) identificar os principais requisitos funcionais de segurança da informação dispostos na norma ISO/IEC 15.408 aplicáveis ao PRB;
b) garantir a proteção dos dados do usuário: tratar a proteção dos dados que são armazenados e transmitidos e controlar o acesso da informação;
c) garantir a auditoria de segurança: tratar as ações realizadas pelos usuários, detectando fraudes ou tentativas de ataques;
d) realizar cópia de segurança e restauração: salvar os dados da base e restaurar os mesmos;
e) criar um atualizador de versão para gerenciar cada nova versão disponibilizada, verificando a consistência dos dados.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos. O segundo capítulo contém a fundamentação teórica necessária para o entendimento do trabalho. Nele são discutidos tópicos relacionados aos aspectos de segurança da informação, a norma ISO/IEC 15.408 e apresentam-se dois trabalhos correlatos.

O terceiro capítulo trata sobre o desenvolvimento do *framework* e das ferramentas, onde são explanados os principais requisitos do problema trabalhado, a especificação contendo diagramas de caso de uso, classe e seqüência. Também são feitos comentários sobre a implementação abrangendo as técnicas, ferramentas utilizadas, operacionalidade e por fim são comentados os resultados e discussão.

O quarto capítulo refere-se às conclusões do presente trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir são descritos os conteúdos pesquisados. Na seção 2.1 são apresentados os aspectos de segurança da informação, na seção 2.2 se comenta a norma ISO/IEC 15.408. Por fim na seção 2.3 são apresentados dois trabalhos correlatos ao tema em questão.

2.1 ASPECTOS DE SEGURANÇA DA INFORMAÇÃO

A segurança da informação ratifica, cada vez mais, sua participação nas empresas, instituições de pesquisa e universidades (NUNES, 2007, p. 5).

> A segurança da informação também busca avaliar a forma como as aplicações são criadas por causa da necessidade de garantir a confidencialidade, integridade e disponibilidade das informações processadas. Isto se dá através da verificação da inclusão de controles no processo de desenvolvimento de software. Controles são análises executadas com a finalidade de garantir que determinadas atividades e especificações sejam eficazmente cumpridas e que não se desviem de padrões pré-estabelecidos. (NUNES, 2007, p. 5).

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

a) confidencialidade: capacidade de um sistema permitir que somente usuários autorizados acessem determinada informação, ao mesmo tempo em que usuários não autorizados tentam acessá-la;
b) integridade de dados: atributo de segurança que indica se uma informação pode ser alterada somente de forma autorizada. Capacidade de um sistema de impedir que uma informação seja alterada sem autorização ou, ao menos de detectar se isso ocorreu;
c) disponibilidade: indica a quantidade de vezes que o sistema cumpriu uma tarefa solicitada sem falhas internas, sobre o número de vezes em que foi solicitado a fazer uma tarefa.

De acordo com Albuquerque e Ribeiro (2002, p. 2), além dos principais aspectos de segurança, existem diversos outros como:

a) autenticação: capacidade de garantir que um usuário, sistema ou informação é mesmo quem alega ser;

b) não repúdio: capacidade do sistema de provar que um usuário executou determinada ação no sistema;
c) legalidade: aderência de um sistema à legislação;
d) privacidade: capacidade de manter incógnito um usuário, impossibilitando a ligação direta da identidade do usuário com as ações por este realizadas;
e) auditoria: capacidade do sistema de auditar tudo o que foi realizado pelos usuários, detectando fraudes ou tentativas de ataque.

Apesar de todos os aspectos citados serem importantes, dependendo da organização, alguns são mais importantes do que outros. Em sistemas bancários, por exemplo, integridade e auditoria são os aspectos mais importantes, seguidos de privacidade e disponibilidade. Na maioria dos sistemas é dada maior ênfase a disponibilidade e integridade. Em resumo, cada sistema tem necessidades de segurança diferentes (DIAS, 2000, p. 44).

2.1.1 Garantias de segurança

McGraw e Viega (1999) comentam que existem poucos recursos disponíveis para ensinar os desenvolvedores a construírem sistemas seguros. Além disso, a pressão do mercado exige que muitos aplicativos sejam desenvolvidos de maneira muito rápida, dificultando o aprendizado sobre segurança e sobre os processos de desenvolvimento, além do envolvimento de equipes de segurança. A conseqüência desta pressão é a falta de qualidade do sistema, incluindo segurança, que na maioria das vezes é deixada em segundo plano.

Segundo Albuquerque e Ribeiro (2002, p. 239), uma forma de reverter a falta de recursos disponíveis para os desenvolvedores é tomar alguns cuidados básicos como:

a) segurança do ambiente de desenvolvimento: não é possível desenvolver uma aplicação segura em um ambiente inseguro;
b) especificação da segurança: nenhuma aplicação é segura contra tudo. A base para a avaliação da segurança é a correta especificação da mesma;
c) desenvolvimento de aplicação com boas práticas de programação: quanto maior a qualidade do código, mais fáceis são os testes e a avaliação final da aplicação;
d) avaliação através de testes e auditoria antes da entrega da aplicação.

Como visto, existem vários cuidados básicos que precisam ser levados em conta para garantir a segurança de uma aplicação. De acordo com Albuquerque e Ribeiro (2002, p. 240), “[...] um código seguro implica em uma certa perda de desempenho, isso é inevitável devido

ao maior controle dentro do sistema, mas pode ser facilmente compensado com um hardware mais rápido, já para um código inseguro, não há hardware que dê jeito.” O desenvolvimento de aplicação com boas práticas de programação consiste em:

a) criar funções intrinsecamente seguras: todas as funções devem verificar os dados de entrada para impedir perda de controle do sistema, falhas gerais de proteção ou outros tipos de falhas catastróficas, como estouro de *buffer* (ALBUQUERQUE; RIBEIRO, 2002, p. 240);
b) usar funções intrinsecamente seguras: exemplo de função não intrinsecamente segura é a *strcpy*, na linguagem de programação C e C++, pois trata-se de uma função para copiar uma *string*. Esta função recebe dois ponteiros como parâmetros e copia todo o conteúdo de um para o outro, até encontrar um valor zero. A *strcpy*, se chamada com parâmetros inválidos, pode facilmente gerar resultados imprevisíveis, por exemplo, sobrescrevendo uma área imprópria da memória. Uma função melhor seria a *strncpy*, que permite definir o máximo de bytes a copiar, impedindo que a memória de destino seja extrapolada. Em linguagens de programação excessivamente tipadas, como *Visual Basic* e *Delphi*, este tipo de problema não é tão freqüente (ALBUQUERQUE; RIBEIRO, 2002, p. 241);
c) testar o retorno de funções chamadas: quando uma função for chamada, seu retorno precisa ser verificado. Existe uma tendência de ignorar determinados erros por serem muito raros. Isto não deveria ocorrer. Mesmo em códigos com pouca chance de conflito, isso torna-se um grave erro de segurança (ALBUQUERQUE; RIBEIRO, 2002, p. 241);
d) documentar funções corretamente: uma boa documentação da função evita o mal entendimento da mesma (SILVA, 2005, p. 66);
e) verificar caracteres especiais: tratar adequadamente os dados, seja pelo usuário ou outro sistema (ALBUQUERQUE; RIBEIRO, 2002, p. 243);
f) manter uma política de versões consistentes: deve-se marcar de forma não ambígua o número da versão nos códigos fontes e na aplicação gerada, de forma a permitir com facilidade a identificação de problemas (SILVA, 2005, p. 67);
g) usar componentes e bibliotecas confiáveis: o uso de bibliotecas ou sistemas auxiliares não confiáveis, podem comprometer toda a segurança do sistema. Deve-se assegurar que bibliotecas não comprometem a segurança do sistema (SILVA, 2005, p. 67);
h) evitar informações sensíveis em arquivos temporários: deve-se evitar que

informações sigilosas tornem-se vulneráveis (ALBUQUERQUE; RIBEIRO, 2002, p. 243);

i) 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. O armazenamento deve ser alvo de análise criteriosa (ALBUQUERQUE; RIBEIRO, 2002, p. 243);

j) tratar todas as entradas do sistema como não confiáveis: deve-se tratar de forma adequada os controles de direitos de acesso e tipo de armazenagem interna, sempre que dados estiverem sendo inseridos (ALBUQUERQUE; RIBEIRO, 2002, p. 244).

Segundo Davis et al. (2004, p. iii), para desenvolver sistemas com poucos defeitos e mais confiáveis, as práticas atuais de desenvolvimento devem mudar. Isso requer que desenvolvedores usem métodos de implementação mais adequados a segurança. Sistemas seguros são aqueles que conseguem se manter funcionais mesmo sob ação de algum defeito, não comprometendo a qualidade e segurança das informações. Tais métodos necessitam da utilização de processos que produzam sistemas seguros. É exigido também que a organização utilize sua experiência e seus processos de segurança quando produzir, melhorar e refazer o sistema.

2.2 NORMA ISO/IEC 15.408

Segundo Albuquerque e Ribeiro (2002, p. 7), 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 da aplicação para o cliente final."

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. Finalmente, 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 idéia do CC é a de que pode-se desenvolver um sistema, seguindo a norma, testá-lo em um laboratório credenciado e obter um selo de certificação. O CC estabelece que qualquer

sistema, para ser considerado seguro, precisa ter um *Security Target* (ST) elaborado. O ST é a especificação de segurança, ou seja, indica quais aspectos de segurança foram considerados importantes e por que foram para aquele sistema em particular (ALBUQUERQUE; RIBEIRO, 2002, p. 8).

Existem sete níveis de garantia de segurança definidas pelo CC. A cada nível há um número maior de testes e, portanto, maior garantia que o sistema atenda aos requisitos de segurança. Estes níveis são denominados *Evaluation Assurance Level* (EAL), e variam de EAL-1 a EAL-7, conforme o CC. Apenas os níveis EAL-1 a EAL4 são reconhecidos pelo ISO/IEC 15.408, pois os níveis EAL-5 a EAL-7 foram considerados extremamente rígidos e, na prática, inviáveis (ALBUQUERQUE; RIBEIRO, 2002, p. 8).

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 (EALs 1 a 4 na ISO/IEC, EALs 1 a 7 no CC) (ALBUQUERQUE; RIBEIRO, 2002, p. 8).

Finalmente 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. 9).

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

Nome da classe	Sigla
Auditoria	FAU
Proteção de dados do usuário	FDP
Criptografia	FCS
Autoproteção	FPT
Canais seguros	FTP
Autenticação	FIA
Acesso ao sistema	FTA
Gerenciamento de segurança	FMT
Utilização de recursos	FRU
Privacidade	FPR

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

Quadro 1 - Classes da norma ISO/IEC 15.408

2.2.1 Auditoria – classe FAU

Auditoria em software baseia-se na gravação, armazenamento e análise das informações. O sistema mantém o registro de tudo que foi feito nele de forma que, em caso de problema de segurança, seja possível identificar o que ou quem o causou. As rotinas de auditoria devem determinar uma série de fatores como registrar as ações importantes, tratar a privacidade, analisar as trilhas e armazená-las com segurança (ALBUQUERQUE; RIBEIRO, 2002, p. 109).

Para cada evento auditado deve-se registrar ao menos as seguintes informações (SILVA, 2005, p. 42):

a) data e hora do evento, tipo de 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.

2.2.2 Proteção de dados do usuário – classe FDP

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:
   - política de controle de acesso,
   - política de controle de fluxo de informações;
b) formas de proteção dos dados do usuário:
   - funções de controle de acesso,
   - funções de controle de fluxo de informações,
   - transferências internas,
   - proteção de informação residual,
   - retorno (Rollback),
   - integridade de dados armazenados;
c) armazenamento *offline*, importação e exportação:
   - autenticação de dados,

- exportação de dados para fora do controle do sistema,
- importação de dados de fora do controle do sistema;

d) comunicação interna:

- proteção de confidencialidade de dados do usuário,
- integridade na transferência de dados.

A proteção de dados do usuário tem como objetivo principal garantir a confidencialidade e a disponibilidade das informações armazenadas (DATASUS, 2008, p. 28). Segundo Albuquerque e Ribeiro (2002, p. 61) descrevem que proteção de dados do usuário é basicamente controle de acesso. "A função de controle de acesso do sistema é utilizada para definir se o usuário tem direito a acessar ou alterar determinada informação e para garantir que apenas o usuário com esse direito pode ter acesso a essa informação."

2.2.3 Autenticação – classe FIA

Segundo Albuquerque e Ribeiro (2002, p. 129), autenticação é garantir que o usuário é de fato quem ele diz ser. Existem três maneiras de se garantir que um usuário é quem ele diz ser:

a) perguntar algo que só ele saberia responder corretamente;
b) solicitar algo que só ele teria;
c) identificá-lo por características pessoais.

A identificação difere-se de autenticação. A identificação trata de saber quem está operando o sistema, já a autenticação trata de garantir que o usuário é quem diz ele ser. Pode-se identificar um usuário sem autenticá-lo, mas a autenticação necessita sempre uma identificação (ALBUQUERQUE; RIBEIRO, 2002, p. 130).

2.2.4 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 o horário comercial. Se uma sessão ficar sem uso durante muito tempo, regras, como o seu cancelamento automático, podem ser

estabelecidas.

O acesso ao sistema envolve uma série de aspectos que podem ser usados 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 do 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 do acesso (hora de trabalho normal ou fora do expediente), o usuário pode ser limitado a algumas funções do sistema;
c) limitação do número de acessos – 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. Pode ocorrer também da sessão ficar congelada devido a alguma falha no processo de encerramento;
d) mensagem de acesso – solicitar que o usuário confirme a leitura de uma mensagem antes de liberar o acesso ao sistema. O objetivo é responsabilizar legalmente ou alertar para o caso de algum erro de operação;
e) histórico de acesso – informar ao usuário quando foi o último acesso com sucesso e eventuais falhas na autenticação ou no estabelecimento da sessão;
f) travamento de sessão – mecanismo que permite ao usuário bloquear o console sem encerrar a sessão.

2.2.5 Gerenciamento de segurança – classe FMT

O 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, determinar quem possui acesso a determinadas funções ou informações de segurança;
b) capacidade de revogação e expiração de atributos de segurança, ou seja, a capacidade do sistema revogar um direito imediatamente ou estabelecer um prazo para tal;
c) gerenciamento das funções de segurança, no qual se descreve o acesso e os atributos das funções de segurança;

d) gerenciamento dos atributos de segurança, no qual se descreve o acesso aos atributos de segurança de outros objetos do sistema, tais como quem define o proprietário de um arquivo;
e) gerenciamento de dados de segurança, o qual, ao contrário dos atributos, não se liga a um objeto particular.

2.2.6 Autoproteção – classe FPT

Segundo Albuquerque e Ribeiro (2002, p. 177), existem três pontos que podem ser destacados nas funções de segurança do sistema:

a) camada subjacente *(abstract machine*), que é a plataforma (sistema operacional ou hardware) sob a qual o sistema opera. Uma plataforma comprometida representa um risco para o sistema;
b) implementação das funções de segurança, que opera sob a camada subjacente e implementa os mecanismos que protegem o sistema;
c) dados e atributos de segurança, que são o banco de dados administrativo que orientam a proteção do sistema.

Segundo Silva (2005, p. 57), o CC define um grupo de famílias importantes para a proteção da segurança:

a) teste de camada subjacente;
b) falha segura;
c) disponibilidade de dados exportados pelo sistema;
d) confidencialidade dos dados exportados pelo sistema;
e) integridade dos dados exportados pelo sistema;
f) transferência interna de dados;
g) proteção física do sistema;
h) recuperação segura;
i) detecção de repetição;
j) monitor de referência;
k) separação de domínios;
l) protocolo de sincronização de estado;
m) registros de tempo;
n) consistência de dados entre funções de segurança;

o) consistência de dados replicados;

p) autoteste.

"Sistemas que não implementam proteção nas funções de segurança comprometem a segurança dos dados do usuário, pois controles de proteção tornam-se vulneráveis e, dessa forma, comprometem a segurança como um todo." (SILVA, 2005, p. 57).

2.2.7 Utilização de recursos – classe FRU

O CC define três atributos que são (ALBUQUERQUE; RIBEIRO, 2002, p. 233):

a) política de cotas para alocação de recursos, a memória e o disco devem ser alocados de forma a sempre permitirem uma margem mínima para o sistema;

b) política de prioridade de serviço, o tempo do processador deve ser dividido entre os diversos processos, ou seja, o sistema de segurança deve ter sempre a prioridade de execução quando sua ação é exigida;

c) tolerância a falhas, em caso de falha dos dois itens anteriores, os sistema de segurança deve ser capaz de manter um estado seguro, ou seja, apresentar certa tolerância a falhas.

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."

2.2.8 Privacidade – classe FPR

Segundo Albuquerque e Ribeiro (2002, p. 207) privacidade "é a 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."

O CC define quatro atributos para a privacidade (ALBURQUERQUE; RIBEIRO, 2002, p. 208):

a) invisibilidade - garante que um usuário possa usar um recurso ou serviço sem que outros, possam saber que o recurso ou serviços está sendo usado;

b) não-rastreamento - garante que um usuário possa fazer uso de vários recursos e

serviços sem que outros possam ligá-lo a esses usos;

c) pseudônimo - garante que um usuário possa usar um recurso ou serviço sem ter sua identidade revelada, porém suas ações são rastreadas e reveladas, geralmente, em situações especiais. Portanto, permite a responsabilização e privacidade ao mesmo tempo.
d) anonimato - garante que um usuário possa usar um recurso ou serviço sem ter sua identidade revelada.

2.2.9 Canais seguros – classe FTP

Segundo Albuquerque e Ribeiro (2002, p. 167), definem que 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 série de características:

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

> Canal e caminho confiável distinguem-se na medida em que canais confiáveis focam a comunicação entre sistemas, geralmente criados através da distribuição de chaves públicas e privadas para os sistemas e usuários envolvidos. O caminho confiável, por sua vez, foca na comunicação entre o usuário e o sistema, buscando garantir que o usuário está efetivamente acessando o sistema desejado, por exemplo, para fazer autenticação. Canais seguros são especialmente recomendados quando existe a necessidade de garantir não-repúdio para um grande número de ações. (SILVA, 2005, p. 61).

2.2.10 Criptografia – classe FCS

Segundo Albuquerque e Ribeiro (2002, p. 155), a criptografia serve de base a uma série de mecanismos de segurança. Muitas funcionalidades da segurança só são possíveis com o uso de criptografia.

> Criptografia é o processo pelo qual uma informação ou texto é embaralhado de forma que só seja possível a obtenção do texto original aplicando-se uma operação baseada em uma chave de acesso. Para obtermos o dado original, precisamos, portanto, saber qual a operação para decriptografia (o algoritmo) e a chave de acesso. (ALBUQUERQUE; RIBEIRO, 2002, p. 155).

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 (ALBUQUERQUE; RIBEIRO, 2002, p. 155).

2.2.10.1 Assinatura digital

Assinatura digital é um mecanismo eletrônico que faz uso de criptografia, mais precisamente, de chaves criptográficas. Estas são um conjunto de *bits* baseado em um determinado algoritmo capaz de cifrar e decifrar informações. Para isso, podem-se usar chaves simétricas ou chaves assimétricas (ALECRIM, 2009).

As chaves simétricas são mais simples, pois com elas o emissor e o receptor utilizam a mesma chave para, respectivamente, cifrar e decifrar uma informação. Já as chaves assimétricas, por sua vez, trabalham com duas chaves: a chave privada e a chave pública. Assim, uma pessoa ou uma organização deve utilizar uma chave de codificação e disponibilizá-la a quem for mandar informações a ela, esta é a chave pública. A outra chave deve ser usada para o processo de decodificação, esta é a chave privada, que é sigilosa e individual. Ambas as chaves são geradas de forma conjunta, portanto, uma está associada à outra (ALECRIM, 2009).

2.3 TRABALHOS CORRELATOS

Existem vários trabalhos semelhantes ao proposto. Dentre eles, foram escolhidos dois cujas características enquadram-se na mesma área de estudo. Foram selecionados os seguintes trabalhos: “Segurança no Desenvolvimento de Aplicações Web” (GOMES; SANTOS, 2006) e “Processo de Apoio à Segurança de Software” (NUNES, 2007).

2.3.1 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 sistemas menos vulneráveis (GOMES; SANTOS, 2006, p. 13).

O trabalho deu ênfase na análise das principais vulnerabilidades existentes no sistema *on-line* da Universidade da Amazônia, o Aprendiz. Assim foram propostas soluções para a diminuição dos riscos decorrentes das falhas detectadas (GOMES; SANTOS, 2006, p. 14).

Segundo Gomes e Santos (2006, p. 14), o trabalho foi elaborado da seguinte forma:

a) foram identificados na literatura as técnicas e procedimentos considerados fundamentais para a segurança da informação;
b) foi apresentada uma visão da segurança da informação, avaliação de ambiente e definição de estratégias de segurança;
c) foi apresentada 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 vulnerabilidades presentes em ambiente web;
e) foi apresentado um estudo de caso em uma aplicação web.

O trabalho apresentou os principais aspectos de segurança, identificou políticas e planos de segurança, levantou e analisou riscos, ameaças e os principais meios para que se desenvolvam sistemas mais seguros.

Foi realizado um estudo de caso onde se verificou algumas vulnerabilidades no ambiente de aprendizado *on-line* da Universidade da Amazônia, o Aprendiz, em que alguns requisitos estão em desacordo com as normas sugeridas pela ISO/IEC 15.408 ou com as boas praticas de programação.

2.3.2 Processo de Apoio à Segurança de Software

Segundo Nunes (2007, p. 2), o trabalho propôs um conjunto de atividades de segurança que formam o Processo de Apoio à Segurança de Software (PASS), que teve como uma das metas, auxiliar desenvolvedores para a construção de software seguro e, por conseguinte, software de maior qualidade.

Os requisitos funcionais de segurança, segundo a norma ISO/IEC 15.408 estudados no trabalho, são descritos no Quadro 2. Estes requisitos definem as características funcionais, no que diz respeito à segurança da informação do sistema (NUNES, 2007, p. 57).

Classe		Descrição
Nome	Sigla	
Auditoria de segurança	FAU	Lida com a funcionalidade de trilha de auditoria fornecida pelo produto, envolvendo as atividades que serão auditadas, o armazenamento seguro dessas trilhas, e a funcionalidade de analisar e revisar os registros dessas trilhas.
Comunicação	FCO	Trabalha tanto com o não repúdio da informação de origem quanto da informação de destino.
Apoio criptográfico	FCS	Gerencia a geração, a distribuição, o acesso, e a destruição de chaves. Trata do uso da chave na encriptação/decriptação, geração de *hash* ou *checksum*, assinatura digital, entre outras operações. Especifica também o algoritmo criptográfico a ser usado e o tamanho das chaves.
Proteção dos dados do usuário	FDP	Trata da proteção dos dados do usuário que são armazenados e transmitidos pelo TOE. Especifica a criação de uma política de controle de acesso e fluxo de informação ao redor da qual as funções de segurança devem ser projetadas e implementadas.
Identificação e autenticação	FIA	Determina como um usuário é identificado (por nome ou identificador de usuário) e autenticado (senha, *smartcard*, ou sistemas de biometria) pelo sistema. Faz também a atribuição dos atributos corretos de segurança ao usuário autenticado.
Gerência de segurança	FMT	Trabalha com as características internas do produto para a gerência de segurança, tais como listas de controle de acesso, parâmetros de segurança, definição e atribuição de papéis de segurança, etc.
Privacidade	FPR	Protege a identidade do usuário no sistema.
Proteção das funções de segurança do TOE	FPT	Cobre a proteção do TOE em termos de componentes que o constituem. Está relacionada a questões do tipo: • *O produto trabalha em modo de segurança*? • *O produto realiza testes de inicialização*? • *É possível recuperar o produto de falhas, protegendo-se ainda suas informações*? • *A checagem de controle de acesso é executada antes de permitir que sejam realizadas ações restritas*?
Utilização de recursos	FRU	Garante que recursos tais como capacidade de processamento e armazenamento sejam gerenciados para prevenir condições de falta de serviço. Prioriza também tarefas e limitação de recursos aos usuários.
Acesso ao TOE	FTA	Gerencia sessões de usuário, estando relacionada a questões do tipo: • *A aplicação previne sessões concorrentes, terminadas ou bloqueadas após intervalos pré-definidos*? • *A aplicação inclui janela de login, definindo o uso apropriado do TOE*? • *A aplicação mostra históricos anteriores de login após sucesso da conexão*?
Canais e caminhos confiáveis	FTP	Lida com aspectos de canais de comunicação entre usuários e o produto, e entre o produto e outros produtos confiáveis de TI.

Fonte: Nunes (2007, p. 56).

Quadro 2 – PASS requisitos funcionais de segurança

3 DESENVOLVIMENTO

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

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O presente trabalho possui como Requisitos Funcionais (RF) o Quadro 3 e como Requisitos Não Funcionais (RNF) o Quadro 4.

REQUISITOS FUNCIONAIS	CASO DE USO
RF01: deverá permitir ao PRB gerar nova versão do sistema	UC01
RF02: deverá permitir ao usuário realizar cópias de segurança	UC02
RF03: deverá permitir ao usuário realizar a restauração da base de dados	UC03
RF04: deverá permitir ao PRB gerar trilha de auditoria	UC04
RF05: deverá permitir ao PRB gerar criptografia dos dados	UC05
RF06: deverá permitir ao PRB gerar *hash*[2] dos dados	UC06

Quadro 3 – Requisitos funcionais

REQUISITOS NÃO FUNCIONAIS
RNF01: deverá ser implementado na linguagem de programação Java
RNF02: deverá ser implementado utilizando o ambiente de desenvolvimento Eclipse 3.4
RNF03: deverá utilizar a *Application Programming Interface* (API) Java DataBase Connectivity (JDBC) para realização das cópias de segurança e restaurações da base de dados

Quadro 4 – Requisitos não funcionais

[2] *Hash* - é uma sequência de caracteres (letras ou números) gerada por um algoritmo de dispersão que transforma uma grande quantidade de dados em uma pequena quantidade. Geralmente, é uma variável que serve para identificar grandes cadeias de dados (PEREIRA, 2009).

3.2 ESPECIFICAÇÃO

Na sequência é apresentada a especificação do *framework*, que foi modelado na ferramenta Entreprise Architect (SPARXSYSTEMS, 2000). Foram utilizados conceitos da orientação a objetos e a *Unified Modeling Language* (UML) (OMG, 2005) para a criação dos diagramas de casos de uso, atividades, classe e de sequência.

O Quadro 5 apresenta os nomes das classes, conforme a norma ISO/IEC 15.408 (descritas no anexo A), a sigla e é indicado quais classes possuem pelo menos um ou mais itens atendidos no desenvolvimento do *framework*.

Nome da classe	Sigla	Possui itens atendidos
Auditoria	FAU	Sim
Proteção de dados do usuário	FDP	Sim
Criptografia	FCS	Sim
Autoproteção	FPT	Sim
Canais seguros	FTP	Sim
Autenticação	FIA	Não
Acesso ao sistema	FTA	Não
Gerenciamento de segurança	FMT	Não
Utilização de recursos	FRU	Não
Privacidade	FPR	Não

Quadro 5 – Classes que possuem itens atendidos

Na especificação do *framework* existem dois cenários. O primeiro (Figura 1) tem como ator o PRB e o segundo (Figura 2) tem como ator o usuário.

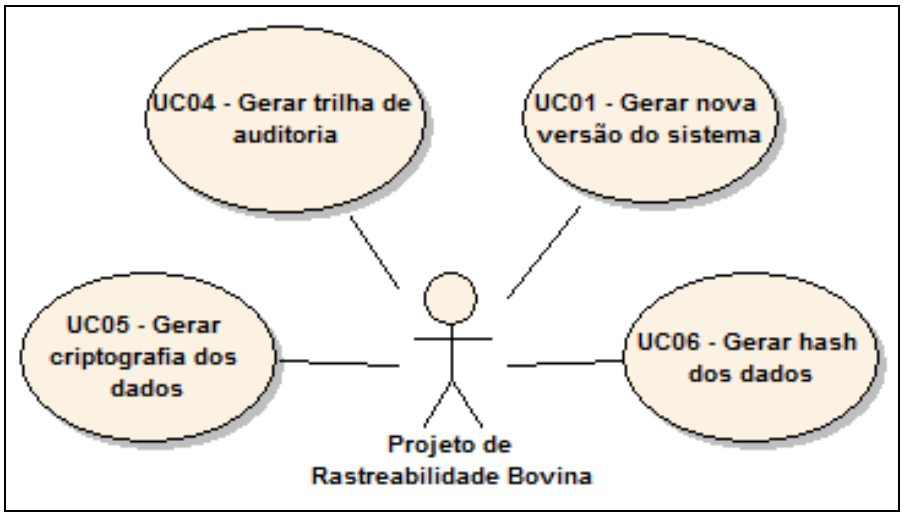

Figura 1 – Diagrama de casos de uso executados pelo PRB

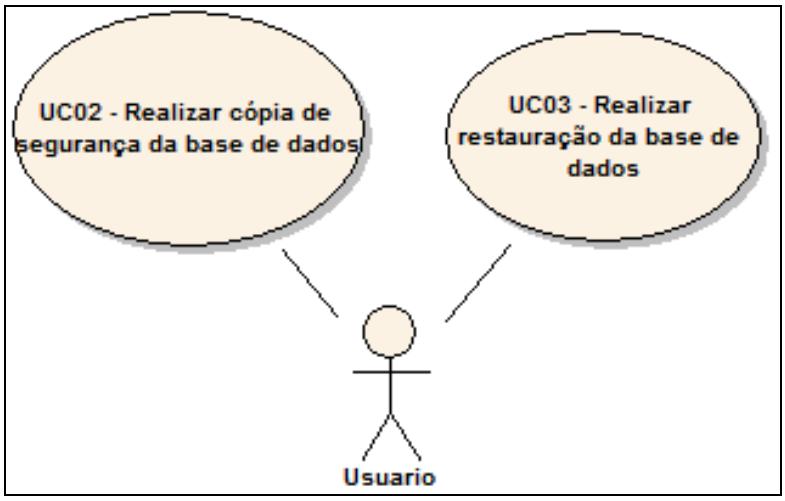

Figura 2 – Diagrama de casos de uso executados pelo usuário

3.2.1 Especificação da classe FDP – proteção de dados do usuário

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

a) `FDP_DAU.1` – autenticação básica dos dados;

b) `FDP_DAU.2` – autenticação dos dados com identidade do gerador;

c) `FDP_ETC.1` – exportação de dados com atributo de segurança;

d) `FDP_IFC.1` – controle de fluxo de informação;

e) `FDP_ITC.2` – importação de dados com segurança.

O caso de uso `UC01 - Gerar nova versão do sistema` (Quadro 6) é responsável pela geração de nova versão onde o PRB informa o número da nova versão e a relação de arquivos, os quais serão assinados digitalmente. Este caso de uso possui, além de um cenário principal, um alternativo e quatro cenários de exceção informando possíveis erros durante a geração da nova versão.

UC01 - Gerar nova versão do sistema: assina digitalmente os arquivos fontes selecionados e gera a nova versão do sistema.	
Requisitos atendidos	RF01.
Pré-condições	Existir uma base de dados MySQL.
Cenário principal	1. O PRB abre a tela para gerar a nova versão. 2. O framework solicita os dados para conexão com a base de dados 3. O framework solicita o(s) arquivo(s) a serem assinados. 4. O PRB informa o(s) arquivo(s). 5. O framework solicita o número da versão do sistema 6. O PRB informa o número da versão do sistema. 7. O framework solicita o número da versão da base 8. O PRB informa o número da versão da base. 9. O PRB solicita a geração da nova versão. 10. O framework gera a nova versão do sistema.
Alternativo 1	No passo 2, caso o PRB não esteja conectado em uma base de dados. 2.1 O framework apresentará uma tela para a realização da conexão. 2.2. O PRB informa os dados para conexão. 2.3 O framework conecta a base de dados.
Exceção 1	No passo 2.2, caso seja informado algum dado incorreto o framework apresenta a mensagem “Não foi possível conectar ao Banco de Dados, verifique os campos e tente novamente!”.
Exceção 2	No passo 4, caso não seja informado (s) arquivo(s) que será(ão) assinado(s), o framework apresenta a mensagem “Selecione um diretório ou arquivo!”.
Exceção 3	No passo 6, caso seja informado algum dado incorreto o framework apresenta a mensagem “Número da versão inválido!
Exceção 4	No passo 8, caso seja informado algum dado incorreto o framework apresenta a mensagem “Número da versão inválido!”.
Pós-condições	Versão do sistema gerada com sucesso.

Quadro 6 – Detalhamento do caso de uso `UC01 - Gerar nova versao do sistema`

O caso de uso `UC06 - Gerar hash dos dados` (Quadro 7) é responsável por codificar determinado conteúdo informado pelo PRB. Este caso de uso possui um cenário principal e um cenário de exceção informando possível erro durante a geração do *hash*.

UC06 – Gerar hash dos dados: codifica os dados.	
Requisitos atendidos	RF06.
Pré-condições	Existir um dado para ser codificado.
Cenário principal	1. O framework solicita o dado a ser codificado. 2. O PRB informa o dado. 3. O framework codificado o(s) dado(s).
Exceção 1	No passo 2, caso o PRB não informe o dado o framework apresenta a mensagem “Erro ao codificar os dados.”.
Pós-condições	Hash gerado com sucesso.

Quadro 7 – Detalhamento do caso de uso `UC06 - Gerar hash dos dados`

O diagrama de classe (Figura 3) dos casos de uso `UC01 – Gerar nova versão do sistema` e `UC06 – Gerar hash dos dados` apresenta as classes responsáveis pela geração de novas versões do sistema e *hash*, as quais são:

a) `Assinatura`: responsável por garantir a integridade dos dados através da assinatura digital dos arquivos;
b) `MainNovaVersao`: classe que define interface da geração de versão;
c) `ParChaves`: classe responsável pela geração do par de chaves (chave pública e chave privada), que é utilizado para a assinatura digital dos arquivos;
d) `VerificaAssinatura`: classe responsável por verificar a integridade dos arquivos assinados;
e) `GravaXML`: contém todos os atributos e métodos para a gravação no arquivo *eXtensible Markup Language* (XML);
f) `Versao`: classe responsável por controlar as versões existentes e pelo armazenamento e leitura do arquivo de versão;
g) `Hash`: classe responsável por codificar o conteúdo.

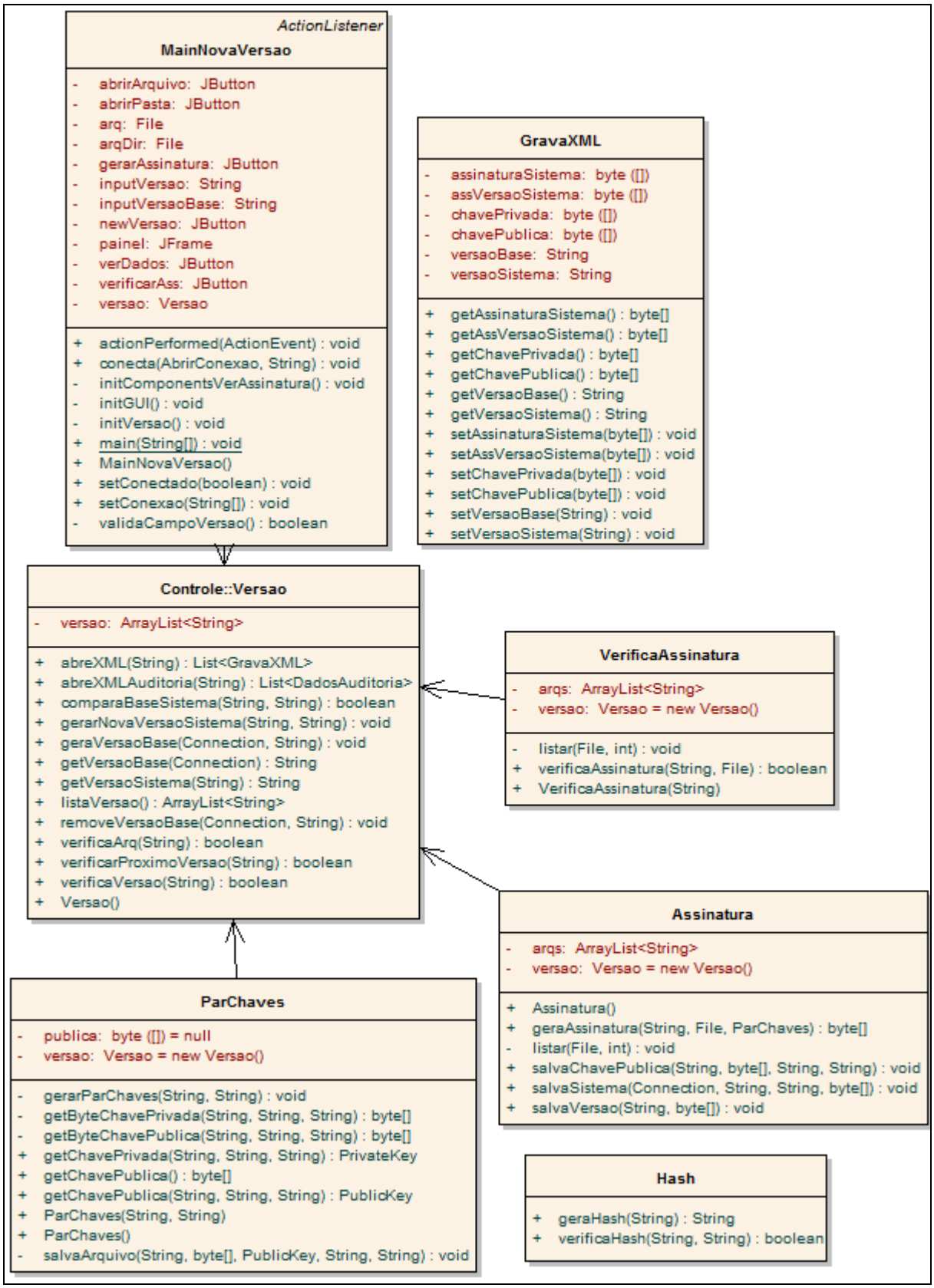

Figura 3 – Diagrama de classe dos casos de uso `UC01 - Gerar nova versão do sistema` e `UC06 - Gerar hash dos dados`

A Figura 4 apresenta o diagrama de atividades correspondente ao caso de uso `UC01 - Gerar nova versão do sistema`. O PRB conecta-se a base de dados, informa o conteúdo a

ser assinado digitalmente, a nova versão do sistema e a nova versão da base de dados. Por fim o *framework* gera o par de chaves (chave pública e chave privada) e assina digitalmente o conteúdo informado.

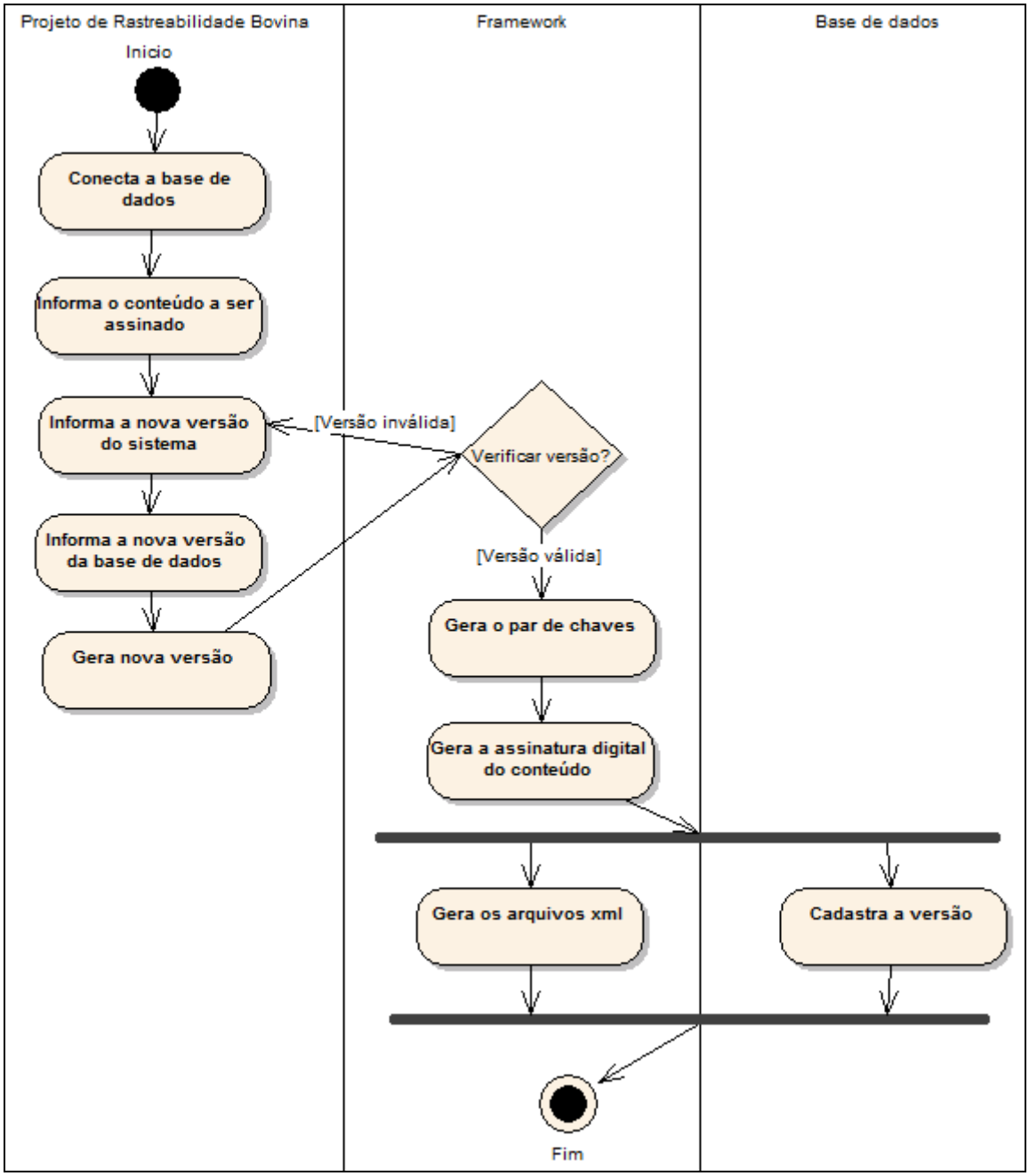

Figura 4 - Diagrama de atividades do caso de uso `UC01 – Gerar nova versão do sistema`

A Figura 5 apresenta o diagrama de sequência correspondente ao caso de uso `UC01 – Gerar nova versão do sistema`. O PRB informa o número da nova versão do sistema e os arquivos correspondentes. O *framework* gera o par de chaves (chave pública e chave privada) o qual é utilizada na assinatura digital, em seguida é gerada a nova versão do sistema. Após assinado é informado o arquivo com a assinatura digital verificando assim a sua integridade.

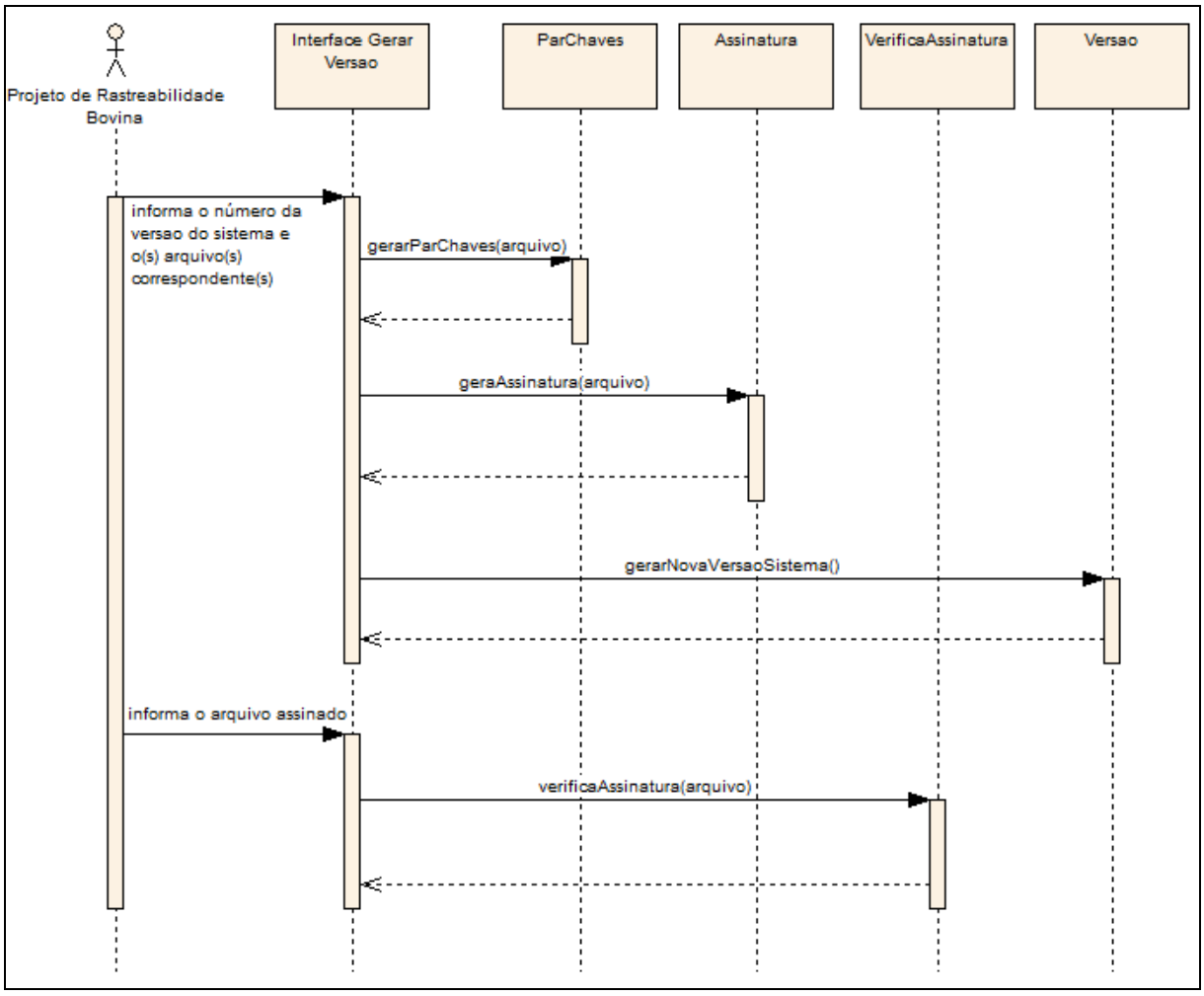

Figura 5 – Diagrama de sequência do caso de uso `UC01 - Gerar nova versão do sistema`

3.2.2 Especificação da classe FAU – auditoria

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

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

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

c) `FAU_SAR.1` – revisão de auditoria;

d) `FAU_SEL.1` – auditoria seletiva;

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

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

O caso de uso `UC04 - Gerar trilha de auditoria` (Quadro 8) apresenta a geração da trilha de auditoria das operações executadas pelo PRB. Este caso de uso possui um cenário principal e um cenário de exceção informando o possível erro durante a geração da trilha de

auditoria.

UC04 – Gerar trilha de auditoria: gera trilha de auditoria.	
Requisitos atendidos	RF04.
Pré-condições	Existir uma operação para auditar.
Cenário principal	1. O framework solicita a operação para auditar. 2. O PRB informa a operação. 3. O framework gera a trilha de auditoria.
Exceção 1	No passo 2, caso não seja informado a operação o framework apresenta a mensagem “Erro ao gerar a auditoria.”.
Pós-condições	Auditoria gerada com sucesso.

Quadro 8 – Detalhamento do caso de uso `UC04 – Gerar trilha de auditoria`

O diagrama de classe do caso de uso `UC04 – Gerar trilha de auditoria` é apresentado na Figura 6 e possui as seguintes classes:

a) `Auditoria`: classe responsável pelo controle e armazenamento da trilha de auditoria;

b) `DadosAuditoria`: classe que contém todos os atributos e métodos da trilha de auditoria.

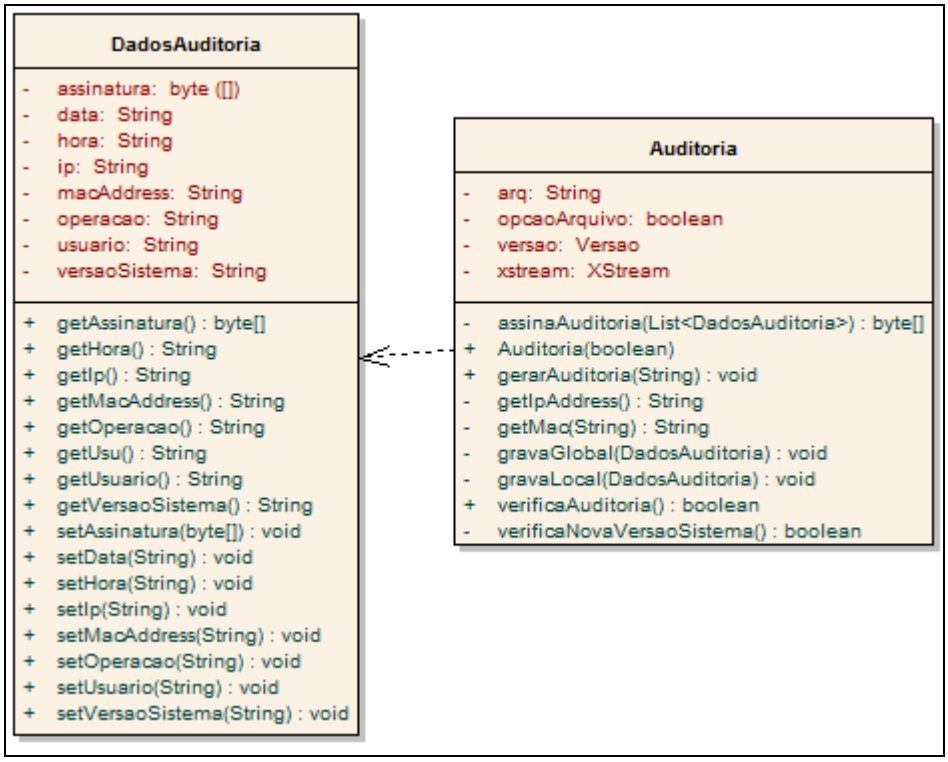

Figura 6 – Diagrama de classe do caso de uso `UC04 – Gerar trilha de auditoria`

3.2.3 Especificação da classe FCS – criptografia

O item atendido desta classe `FCS` (descrito no anexo A) foi o `FCS_COP.1` – operação

de criptografia. O caso de uso `UC05 – Gerar criptografia dos dados` (Quadro 9) apresenta a geração da criptografia dos dados pelo PRB. Este caso de uso possui um cenário principal e um cenário de exceção o qual informa o possível erro durante a criptografia dos dados.

UC05 – Gerar criptografia dos dados: criptografa os dados.	
Requisitos atendidos	RF05.
Pré-condições	Existir um dado para ser criptografado.
Cenário principal	1. O framework solicita o dado a ser criptografado. 2. O PRB informa o dado. 3. O framework criptografa o(s) dado(s).
Exceção 1	No passo 2, caso não seja informado o(s) dado(s) o framework apresenta a mensagem “Não foi possível criptografar os dados.”.
Pós-condições	Criptografia gerada com sucesso.

Quadro 9 – Detalhamento do caso de uso `UC05 – Gerar criptografia dos dados`

O diagrama de classe do caso de uso `UC05 – Gerar criptografia dos dados` é apresentado na Figura 7 e possui duas classes, as quais são:

a) `Criptografia`: classe responsável por criptografar os dados;

b) `Configuracao`: classe de controle que contém os algoritmos de criptografia.

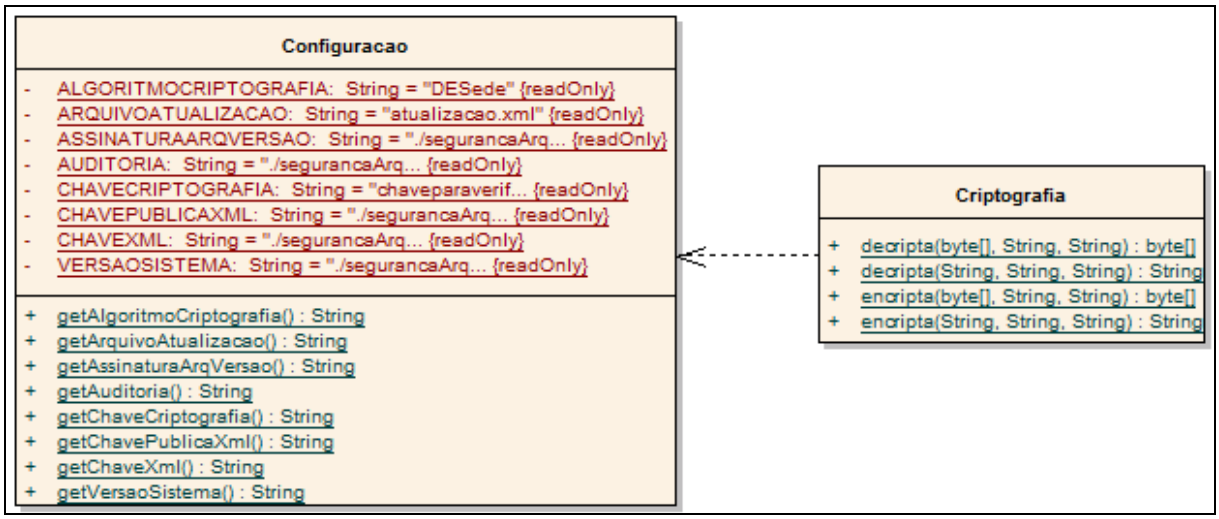

Figura 7 – Diagrama de classe do caso de uso `UC05 – Gerar criptografia dos dados`

3.2.4 Especificação da classe FPT – autoproteção

Os itens atendidos da classe `FPT` (descritos no anexo A) são:

a) `FPT_ITC.1` – confidencialidade dos dados exportados pela aplicação;

b) `FPT_ITI.1` – detecção de modificações;

c) FPT_ITT.1 – proteção básica de dados internos da aplicação.

O caso de uso UC02 – Realizar cópia de segurança da base de dados (Quadro 10) representa as ações do usuário no processo de geração da cópia de segurança. Este caso de uso possui, além de um cenário principal, um alternativo e um cenário de exceção informando o possível erro durante realização da cópia de segurança.

UC02 – Realizar cópia de segurança da base de dados: realiza um backup criptografado da base de dados correspondente.	
Requisitos atendidos	RF02.
Pré-condições	Existir uma base de dados MySQL.
Cenário principal	1. O usuário abre a tela para realização do backup. 2. O framework solicita os dados para conexão com a base de dados. 3. O usuário solicita a realização do backup. 4. O framework realiza o backup.
Alternativo 1	No passo 2, caso o usuário não esteja conectado em uma base de dados. 2.1 O framework apresentará uma tela para a realização da conexão. 2.2. O usuário informa os dados para conexão. 2.3 O framework conecta a base de dados.
Exceção 1	No passo 2.2, caso o usuário informe algum dado incorreto o framework apresenta a mensagem “Não foi possível conectar ao Banco de Dados, verifique os campos e tente novamente!”.
Pós-condições	Backup realizado com sucesso.

Quadro 10 – Detalhamento do caso de uso UC02 – Realizar cópia de segurança da base de dados

O caso de uso UC03 – Realizar restauração da base de dados (Quadro 11) representa as ações do usuário no processo de restauração da cópia de segurança. Este caso de uso possui, além de um cenário principal, um alternativo e dois cenários de exceção informando possíveis erros durante o processo de restauração da cópia de segurança.

UC03 – Realizar restauração da base de dados: realiza a restauração da base de dados.	
Requisitos atendidos	RF02.
Pré-condições	Existir uma base de dados MySQL. Uma cópia de segurança deve ter sido gerada.
Cenário principal	1. O usuário abre a tela para a realização da restauração. 2. O framework solicita os dados para conexão com a base de dados. 3. O usuário solicita a realização da restauração da base de dados. 4. O framework solicita o arquivo para realizar a restauração. 5. O usuário informa o arquivo de restauração. 6. O framework realiza a restauração da base de dados.
Alternativo 1	No passo 2, caso o usuário não esteja conectado em uma base de dados. 2.1 O framework apresentará uma tela para a realização da conexão. 2.2. O usuário informa os dados para conexão. 2.3 O framework conecta a base de dados.
Exceção 1	No passo 2.2, caso o usuário informe algum dado incorreto o framework apresenta a mensagem "Não foi possível conectar ao Banco de Dados, verifique os campos e tente novamente!".
Exceção 2	No passo 5, caso o arquivo esteja corrompido o framework apresenta a mensagem: "Erro na verificação da integridade não foi possível restaurar!".
Pós-condições	Restauração realizada com sucesso!

Quadro 11 – Detalhamento do caso de uso `UC03 – Realizar restauração da base de dados`

O diagrama de classe dos casos de uso `UC02 – Realizar cópia de segurança da base de dados` e `UC03 – Realizar restauração da base de dados` é apresentado na Figura 8, e possui as seguintes classes:

a) `Backup`: classe responsável por gravar no arquivo a cópia de segurança da base de dados criptografada;

b) `Restore`: classe responsável por decriptar o arquivo da cópia de segurança e restaurar a base de dados;

c) `MainBaseDados`: classe responsável pela interface de geração da cópia de seguranca e restauração da base de dados;

d) `ConexaoJDBC`: classe responsável pelo conexão a base de dados MySQL;

e) `AbrirConexao`: classe responsável pela interface para a conexão a base de dados;

f) `AbrirEsquema`: classe responsável pela visualização dos esquemas presentes na base de dados conectada.

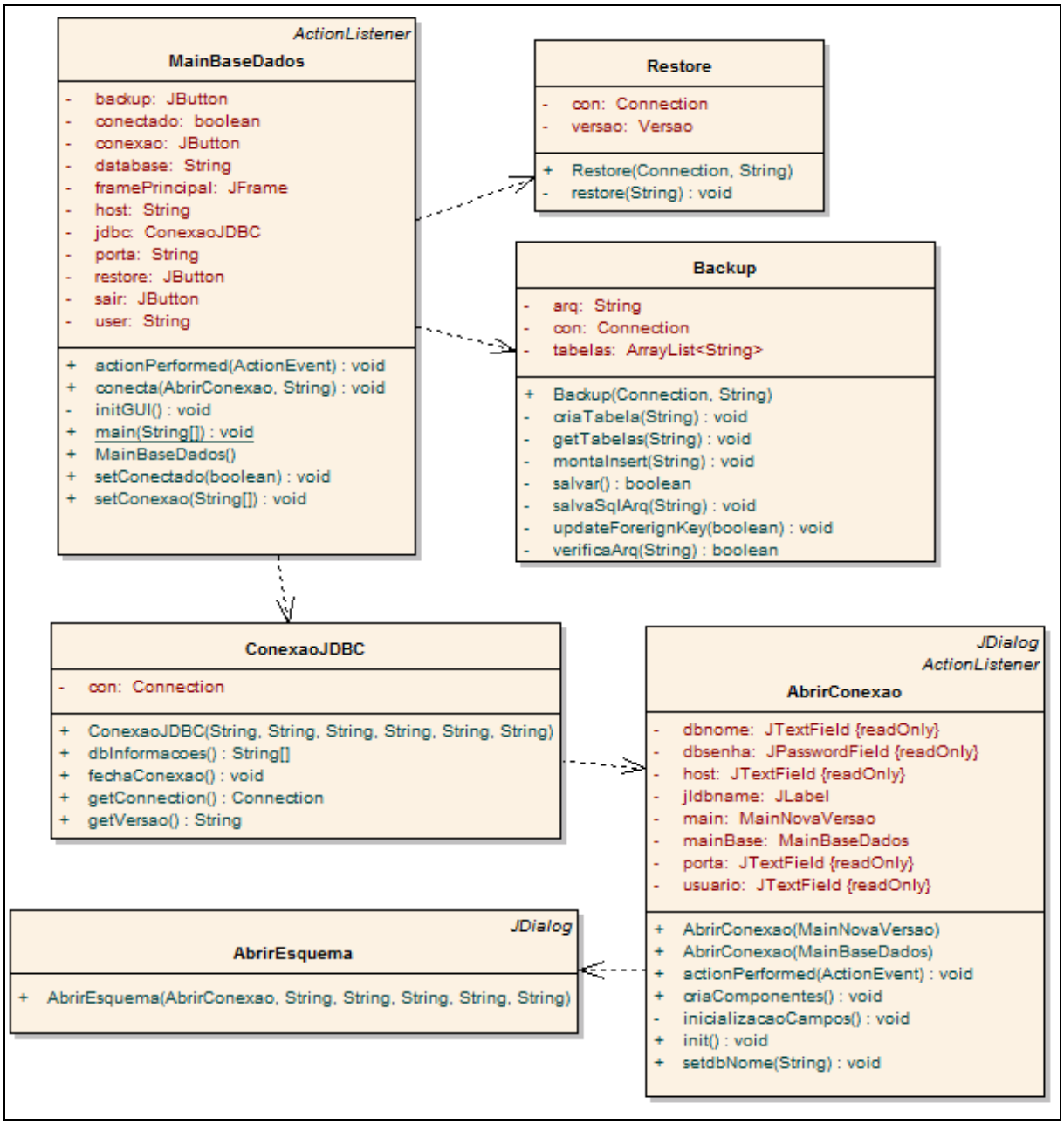

Figura 8 – Diagrama de classe dos casos de uso `UC02 – Realizar cópia de segurança da base de dados` e `UC03 – Realizar restauração da base de dados`

A Figura 9 apresenta o diagrama de atividades correspondente aos casos de uso `UC02 – Realizar cópia de segurança da base de dados` e `UC03 – Realizar restauração da base de dados`. O usuário conecta-se a base de dados, após conectado o usuário gera a cópia de segurança. O *framework* analisa os dados da base e criptografa os mesmo gravando no arquivo de cópia de segurança. Na restauração da base de dados o usuário informa o arquivo criptografado, o qual é decriptografado pelo *framework* e então os dados são gravados na base de dados.

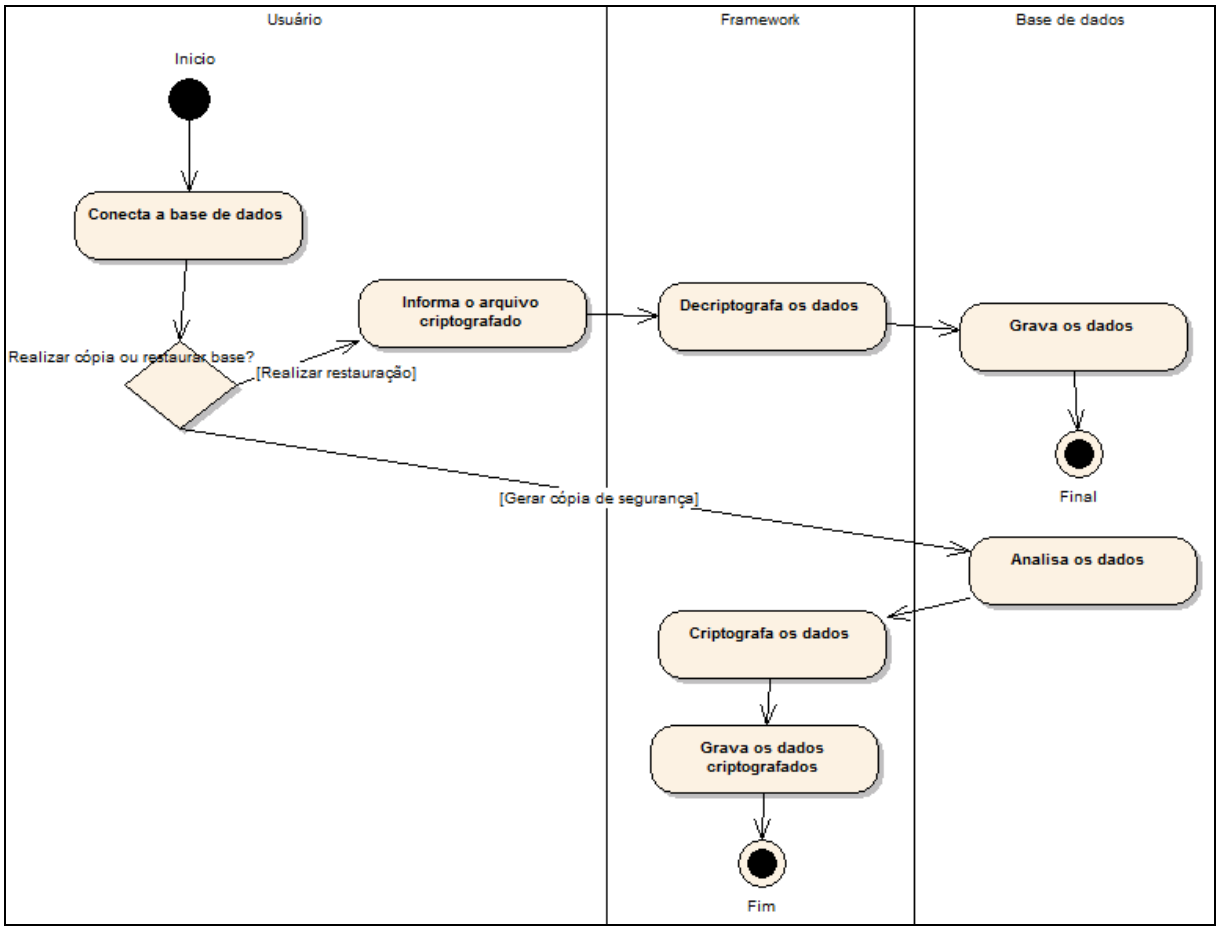

Figura 9 - Diagrama de atividades dos casos de uso `UC02 – Realizar cópia de segurança da base de dados` e `UC03 – Realizar restauração da base de dados`

A Figura 10 apresenta o diagrama de sequência correspondente aos casos de uso `UC02 – Realizar cópia de segurança da base de dados` e `UC03 – Realizar restauração da base de dados`. O *framework* solicita o nome da base de dados para efetuar a conexão, após conectado a cópia de segurança criptografada é gerada. Para restaurar é necessário que o usuário informe o arquivo de restauração.

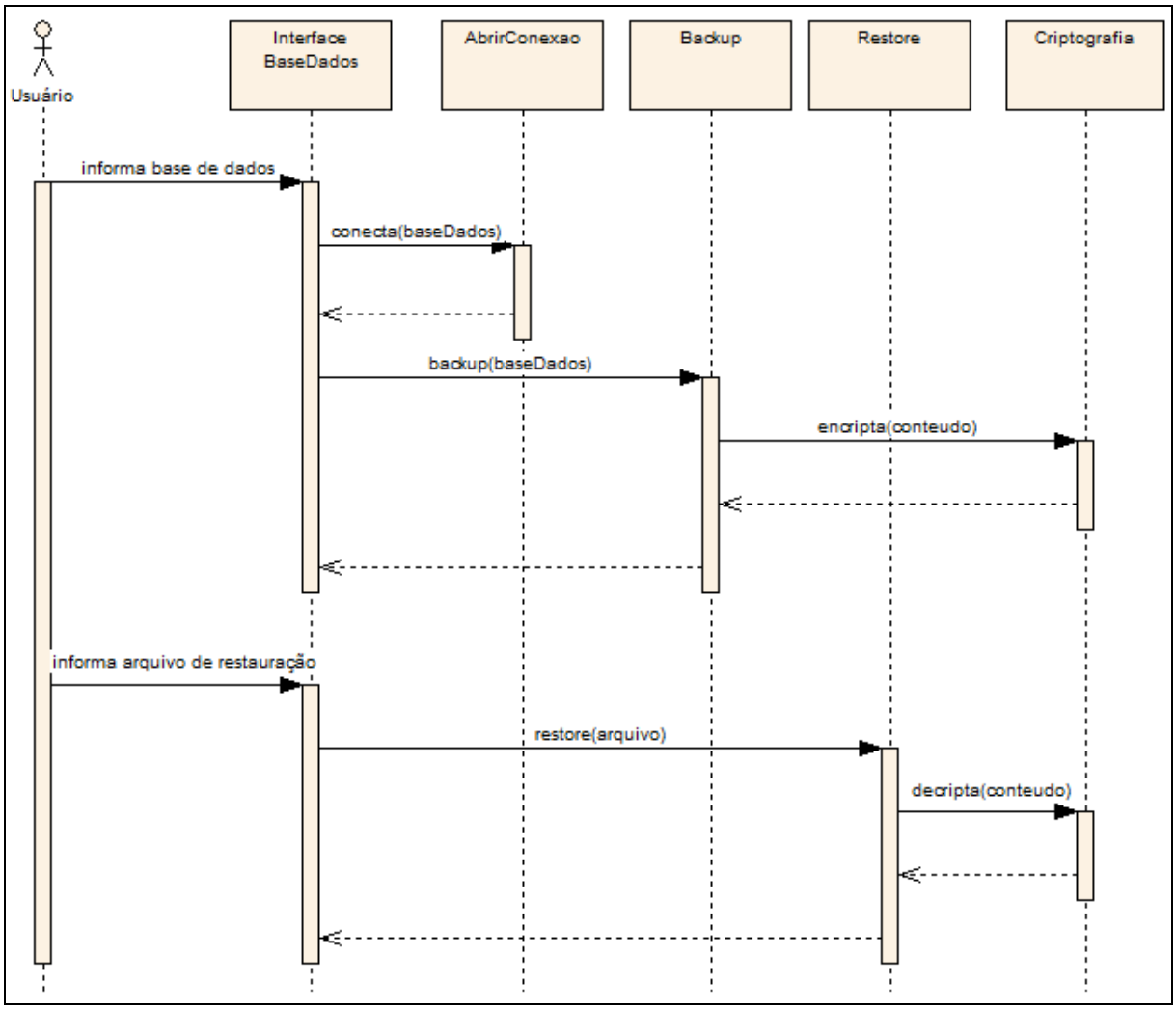

Figura 10 – Diagrama de sequência dos casos de uso `UC02 – Realizar cópia de segurança da base de dados` e `UC03 – Realizar restauração da base de dados`

3.3 IMPLEMENTAÇÃO

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

3.3.1 Técnicas e ferramentas utilizadas

O *framework* foi implementado na linguagem de programação Java, utilizando-se o ambiente de desenvolvimento Eclipse (ECLIPSE, 2008), juntamente com a API JDBC

(MYSQL, 2009) e para a geração de arquivos XML foi utilizada a biblioteca XStream (XSTREAM, 2008).

3.3.2 Técnicas e código fonte implementados

Nas seções seguintes são apresentadas as técnicas utilizadas, pseudocódigos e trechos de código fonte referentes às implementações das classes da norma ISO/IEC 15.408 (Quadro 5).

3.3.2.1 Implementação da classe FDP – proteção de dados do usuário

O controle das versões é implementado na classe `Versao` e possui o método `verificaVersao`, apresentado no Quadro 12, o qual verifica se a nova versão informada já existe ou não cadastrada no arquivo de controle de versões. O número da versão é formado por três números, por exemplo, (1.5.3) e possui o seguinte controle:

a) o primeiro número representa uma mudança estrutural profunda no projeto, e deve raramente ser alterado;
b) o segundo número representa uma mudança que implica na compatibilidade entre base de dados e sistema;
c) o terceiro número deve ser incrementado a qualquer liberação, de base de dados ou de sistema;
d) a igualdade nos dois primeiro números indica compatibilidade entre a base de dados e o sistema.

```
public boolean verificaVersao(String novaVersao) throws Exception {
    // o metodo listaVersao() retorna todas as versões
    // cadastradas no arquivo XML
    versao = listaVersao();
    // para cada versao existente verifica se é igual
    for (int i = 0; i < versao.size(); i++) {
        if (versao.contains(novaVersao))
            return false;
    }
    return true;
}
```

Quadro 12 – Método responsável pela verificação da nova versão

Cada versão é gravada no arquivo de controle contendo o par de chaves (chave pública

e chave privada) criptografadas juntamente com o número da versão.

Na geração de uma nova versão utiliza-se a assinatura digital onde foi necessário a implementação do método `gerarParChaves` que é responsável por gerar o par de chaves (chave pública e chave privada), apresentado no Quadro 13.

```
private void gerarParChaves(String versao, String versaoBase)
        throws Exception {
    // Gerando o par de chaves
    KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
    SecureRandom random = SecureRandom.getInstance("SHA1PRNG", "SUN");
    keyGen.initialize(1024, random);
    KeyPair pair = keyGen.generateKeyPair();
    PrivateKey keyPriv = pair.getPrivate();

    PublicKey KeyPub = pair.getPublic();
    publica = KeyPub.getEncoded();

    // criptografia na chave privada
    byte[] chaveCriptada = Criptografia.encripta(keyPriv.getEncoded(),
            Configuracao.getChaveCriptografia(), Configuracao
                    .getAlgoritmoCriptografia());
    // grava chaves no arquivo
    salvaArquivo(Configuracao.getChaveXml(), chaveCriptada, KeyPub, versao,
            versaoBase);

}
```

Quadro 13 – Método responsável por gerar o par de chaves

Após o par de chaves ser gerado o método `gerarAssinatura` (Quadro 14) da classe `Assinatura`, percorre todo o conteúdo a ser assinado, seja este um arquivo ou um diretório, gerando a assinatura de todos os arquivos a partir da chave privada.

```
public byte[] geraAssinatura(String versao, File diretorio, ParChaves chaves)
        throws Exception {
    File arq = new File(diretorio.toString());// seta o nome do arquivo ou diretorio
    listar(arq, 0);// percorre todo o diretorio guardando todos os arquivos
    Signature dsa = Signature.getInstance("SHA1withRSA");// inicializa o algoritmo RSA
    PrivateKey priv = chaves.getChavePrivada(Configuracao.getChaveXml(),
            versao, null);// carrega a chave privada criptografada
    dsa.initSign(priv);     // inicializa a assinatura com a chave privada
    priv = null;
    for (int i = 0; i < arqs.size(); i++) {// para cada arquivo carregado é assinado
        FileInputStream fis = new FileInputStream(arqs.get(i));
        BufferedInputStream bufin = new BufferedInputStream(fis);
        byte[] buffer = new byte[bufin.available()];
        int len;
        while (bufin.available() != 0) {
            len = bufin.read(buffer);
            dsa.update(buffer, 0, len);// para cada arquivo atualiza a assinatura
        }
        bufin.close();
        fis.close();
    }
    // depois que todos os arquivos/pastas foram lidas, gera a assinatura
    byte[] realSig = dsa.sign();
    arqs.clear();
    return realSig;
}
```

Quadro 14 – Método responsável por gerar a assinatura digital

Na verificação da assinatura digital implementou-se o método `verificarAssinatura` (Quadro 15), o qual possui os seguintes parâmetros:

a) `arqChavePublica` – arquivo que contém a chave pública, por padrão o arquivo é o `chavePublica.xml`;

b) `arqAssinatura` – arquivo que contém a assinatura digital dos arquivos.

O método `verificarAssinatura` percorre o conteúdo a ser verificado e compara a assiantura digital passada por parâmetro com a gerada neste método informando, assim se o conteúdo está ou não corrompido.

```
public boolean verificaAssinatura(String arqChavePublica, File arqAssinatura)
        throws Exception {
    // abre o arquivo xml que contem a chave publica
    List<GravaXML> dados = (ArrayList<GravaXML>) this.versao
            .abreXML(arqChavePublica);
    byte[] encKey = dados.get(0).getChavePublica();// bytes da chave publica
    KeyFactory rsaKeyFac = KeyFactory.getInstance("RSA");
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(encKey);
    RSAPublicKey pubKey = (RSAPublicKey) rsaKeyFac.generatePublic(keySpec);
    // carrega a assinatura do arquivo do sistema ou arquivo de nr de versao
    List<GravaXML> dados2 = (ArrayList<GravaXML>) this.versao
            .abreXML(arqAssinatura.getAbsolutePath());
    byte[] sigToVerify = dados2.get(0).getAssinaturaSistema();
    if (dados2.get(0).getAssinaturaSistema() == null)
        sigToVerify = dados2.get(0).getAssVersaoSistema();
    Signature sig = Signature.getInstance("SHA1withRSA");
    PublicKey pubkey = pubKey;
    sig.initVerify(pubkey);// carrega chave publica
    for (int i = 0; i < arqs.size(); i++) {// para cada arquivo é carregado
        FileInputStream datafis = new FileInputStream(arqs.get(i));
        BufferedInputStream bufin = new BufferedInputStream(datafis);
        byte[] buffer = new byte[bufin.available()];
        int len;
        while (bufin.available() != 0) {
            len = bufin.read(buffer);
            sig.update(buffer, 0, len);
        }
        bufin.close();
        datafis.close();
    }
    // retorna true se os arquivos nao estiverem corrompidos
    boolean verifies = sig.verify(sigToVerify);
    return verifies;
}
```

Quadro 15 – Método responsável pela verificação da integridade dos arquivos assinados

Para a implementação da classe `Hash` foi desenvolvido os seguintes métodos (Quadro 16):

a) `gerarHash` – responsável por codificar o conteúdo passado por parâmetro;

b) `verificaHash` – responsável por codificar o conteúdo e compará-lo com o `hash` retornando verdadeiro caso sejam iguais ou falso caso contrário.

```
public static String geraHash(String conteudo) {
    try {
        MessageDigest md = MessageDigest.getInstance("SHA-1");
        md.update(conteudo.getBytes());
        BigInteger hash = new BigInteger(1, md.digest());
        return hash.toString();
    } catch (NoSuchAlgorithmException ns) {
        ns.printStackTrace();
    }
    return null;
}
public static boolean verificaHash(String hash, String conteudo) {
    //return true (se o hash passado é igual ao hash do conteudo)
    return hash.equals(geraHash(conteudo));
}
```

Quadro 16 – Métodos da classe `Hash`

3.3.2.2 Implementação da classe FAU – auditoria

O Quadro 17 apresenta o método `gerarAuditoria` da classe `Auditoria`, responsável pela geração da trilha de auditoria, o qual possui os seguintes atributos de controle:

g) `operacao` – atributo responsável pela descrição da trilha de auditoria;

h) `usuario` – atributo de identificação de quem gerou a trilha de auditoria;

i) `data` – atributo que guarda a data da trilha de auditoria;

j) `hora` – atributo que guarda a hora da trilha de auditoria;

k) `ip` – atributo que guarda o endereço Internet *Protocol* (IP);

l) `macAddress` – atributo que guarda o endereço *Media Access Control* (MAC);

m) `versaoSistema` – atributo que guarda o número da versão atual do sistema;

n) `assinatura` – atributo que guarda para cada trilha de auditoria gerada a assinatura digital da mesma.

```
public void gerarAuditoria(String operacao, String usuario)
        throws Exception {
    // inicializa o objeto DadosAuditoria
    DadosAuditoria dados = new DadosAuditoria();
    dados.setOperacao(operacao);// seta a operação (descrição)
    dados.setUsuario(usuario);// seta o usuario
    Date data2 = new Date(System.currentTimeMillis());// seta data
    SimpleDateFormat fDate = new SimpleDateFormat("dd/MM/yyyy");
    dados.setData(fDate.format(data2));
    SimpleDateFormat fd = new SimpleDateFormat("HH:mm:ss");
    dados.setHora(fd.format(data2));// seta hora
    dados.setVersaoSistema(this.versao.getVersaoSistema(Configuracao
            .getVersaoSistema()));// seta o numero da versao sistema
    String ip = getIpAddress();
    dados.setIp(ip);// seta o endereço ip
    String mac = getMac(ip);
    dados.setMacAddress(mac);// seta o endereço mac
    if (this.opcaoArquivo) {
        // grava a auditoria no arquivo da pasta raiz do sistema
        gravaGlobal(dados);
    } else {
        // grava a auditoria no arquivo local
        gravaLocal(dados);
    }
}
```

Quadro 17 – Método responsável pela auditoria

O armazenamento da trilha de auditoria foi implementado de duas formas diferentes utilizando a biblioteca XStream (XSTREAM, 2008). Uma é o armazenamento global, onde as trilhas de auditoria são gravadas no arquivo padrão do *framework* chamado `auditoria.xml`, a outra é o armazenamento local onde as trilhas de auditoria são gravadas na pasta raíz do PRB.

3.3.2.3 Implementação da classe FCS – criptografia

O Quadro 18 apresenta os métodos `encripta` e `decripta` da classe `Criptografia`. Estes métodos possuem os seguintes parâmetros:

a) `valor`: conteúdo que será criptografado ou decriptografado;

b) `chave`: chave (senha) para a realização da criptografia, o *framework* é responsável pelo controle/armazenamento da mesma;

c) `algoritmo`: algoritmo de criptografia.

```
public static byte[] encripta(byte[] valor, String chave, String algortimo)
        throws Exception {
    //dados a serem criptografados
    byte[] encryptKey = chave.getBytes("UTF-8");
    //inicializa o algoritmo de criptografia
    Cipher cipher = Cipher.getInstance(algortimo);
    KeySpec keySpec = new DESedeKeySpec(encryptKey);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory
            .getInstance(algortimo);
    SecretKey secretKey = secretKeyFactory.generateSecret(keySpec);
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);//inicializa modo para criptografar
    byte[] cipherText = cipher.doFinal(valor);
    return cipherText;//returna conteúdo criptografado
}
public static byte[] decripta(byte[] valor, String chave, String algortimo)
        throws Exception {
    //dados a serem decriptados
    byte[] encryptKey = chave.getBytes("UTF-8");
    //inicializa o algoritmo de criptografia
    Cipher cipher = Cipher.getInstance(algortimo);
    KeySpec keySpec = new DESedeKeySpec(encryptKey);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory
            .getInstance(algortimo);
    SecretKey secretKey = secretKeyFactory.generateSecret(keySpec);
    cipher.init(Cipher.DECRYPT_MODE, secretKey);//inicializa modo de decriptação
    byte[] decipherText = cipher.doFinal(valor);
    return decipherText;//returna conteúdo decriptado
}
```

Quadro 18 – Métodos responsáveis pela criptografia e decriptografia

3.3.2.4 Implementação da classe FPT – autoproteção

Na geração da cópia de segurança foi implementada a classe `Backup` que utiliza a API JDBC para conectar-se a base de dados. A cópia de segurança é gravada no arquivo pelo método `salvaSqlArq` (Quadro 19), o qual utiliza a classe `Criptografia` para criptografar o mesmo.

```
private void salvaSqlArq(String conteudo) throws Exception {
    //cria ou abre o arquivo de bakcup
    FileWriter out = new FileWriter(arq, true);
    //criptografa o conteudo e salva no arquivo
    out.write(Criptografia.encripta(conteudo.toString(),
            Configuracao.getChaveCriptografia(),
            Configuracao.getAlgoritmoCriptografia()));
    out.write(";\n");
    out.close();
}
```

Quadro 19 – Método de criptografia da cópia de segurança

O Quadro 20 apresenta o método `restore`, este método tem como parâmetro o

arquivo da cópia de segurança criptografado. A partir deste arquivo é percorrida cada linha, a qual é decriptografada e então os dados são inseridos na base de dados.

```
private void restore(String arq) {

    Statement stmt = con.createStatement();
    String strLine = null;
    StringBuilder cript = new StringBuilder();
    // carrega o arquivo de restauração
    FileInputStream fstream = new FileInputStream(arq);
    DataInputStream in = new DataInputStream(fstream);
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
    // para cada quebra de linha "\n"
    while ((strLine = br.readLine()) != null) {
        cript.append(strLine);
        // se encontra um ";"
        if (strLine.contains(";")) {
            // decripta o conteúdo
            String[] palavras = Criptografia.decripta(cript.toString(),
                    Configuracao.getChaveCriptografia(),
                    Configuracao.getAlgoritmoCriptografia()).split(";");
            for (int i = 0; i < palavras.length; i++) {
                // remove carecteres especiais
                if (!palavras[i].equals("\n")
                        && !palavras[i].equals("")
                        && !palavras[i].equals("\n\n")) {
                    //executa comando SLQ decriptado do
                    //arquivo de restauração
                    stmt.executeUpdate(palavras[i]);
                }
            }
            cript = null;
            cript = new StringBuilder();
        }
    }
}
```

Quadro 20 – Método responsável por decriptografar e restaurar a base de dados

3.3.3 Operacionalidade da implementação

Esta seção tem como objetivo mostrar a operacionalidade do *framework*, abordando as classes atendidas (Quadro 5) pelo mesmo. Na seção 3.3.3.2 é apresentado um exemplo da operacionalidade da classe `FCS` – criptografia.

Por motivos de confidencialidade do PRB, a seção 3.3.3.3 foi desenvolvida com intuito de simular o sistema de autenticação de usuários do PRB, apresentando a operacionalidade das classes `FAU` – auditoria e `FTP` – canais seguros.

3.3.3.1 Operacionalidade da classe FDP – proteção de dados do usuário

Ao executar a classe `MainNovaVersao` é apresentada a interface (Figura 11), a qual permite a geração de uma nova versão e a verificação da assinatura digital.

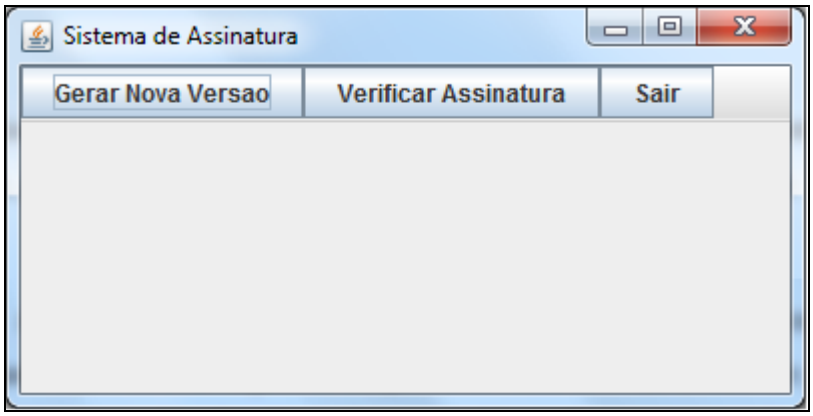

Figura 11 – Interface do sistema de assinatura digital

Na área superior estão localizados os botões que permitem respectivamente, gerar uma nova versão, verificar assinatura digital e fechar a interface. Ao clicar no botão `Gerar Nova Versão` é inicializada a sua respectiva interface (Figura 12) que possui os seguintes botões:

a) `Conectar Base` – responsável pela conexão com a base de dados;
b) `Testar` – após a conexão com a base de dados, este botão testa se a mesma foi efetuada com sucesso ou não;
c) `Selecionar` – abre a janela a qual é informado o diretório ou arquivo que será assinado digitalmente;
d) `Gerar Versão` – após informar o número da versão do sistema e da base de dados, o *framework* gera o par de chaves (chave pública e chave privada), assina digitalmente o conteúdo informado e armazena as versões informadas no arquivo XML, conforme o Quadro 21.

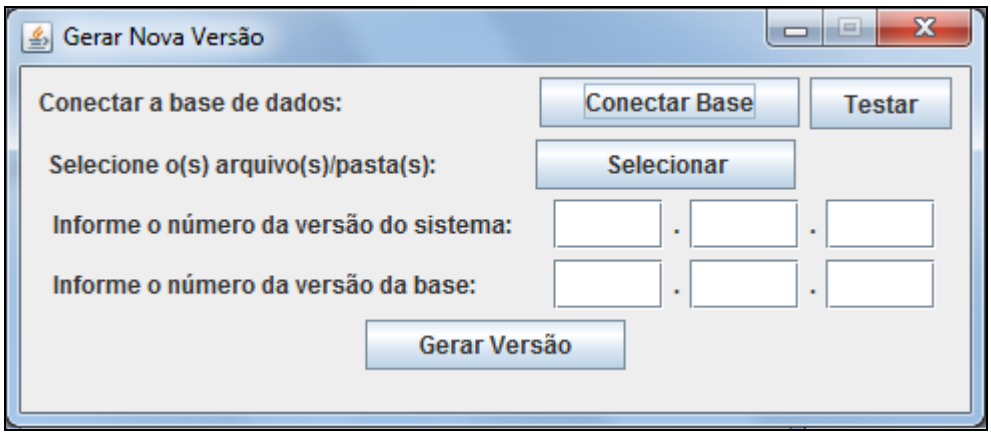

Figura 12 – Interface Gerar Nova Versão

Após a clicar no botão `Gerar Versão` os seguintes arquivos XML são criados (Figura 13):

a) `chaves.xml` – neste arquivo é armazenado a chave privada criptografada, a chave publica, o número da versão do sistema e o número da versão da base, conforme apresentado no Quadro 21;

b) `chavePublica.xml` - neste arquivo é armazenada uma cópia da chave pública;

c) `versaoSistema.xml` - neste arquivo é armazenado uma cópia do número da versão do sistema;

d) `assinaturaArqVersaoSis.xml` - neste arquivo é armazenado a assinatura digital do conteúdo do arquivo `versaoSistema.xml`;

e) `assinaturaUltimaVersaoSistema.xml` - neste arquivo é armazenado a assinatura digital do respectivo diretório ou arquivo informado na Figura 12.

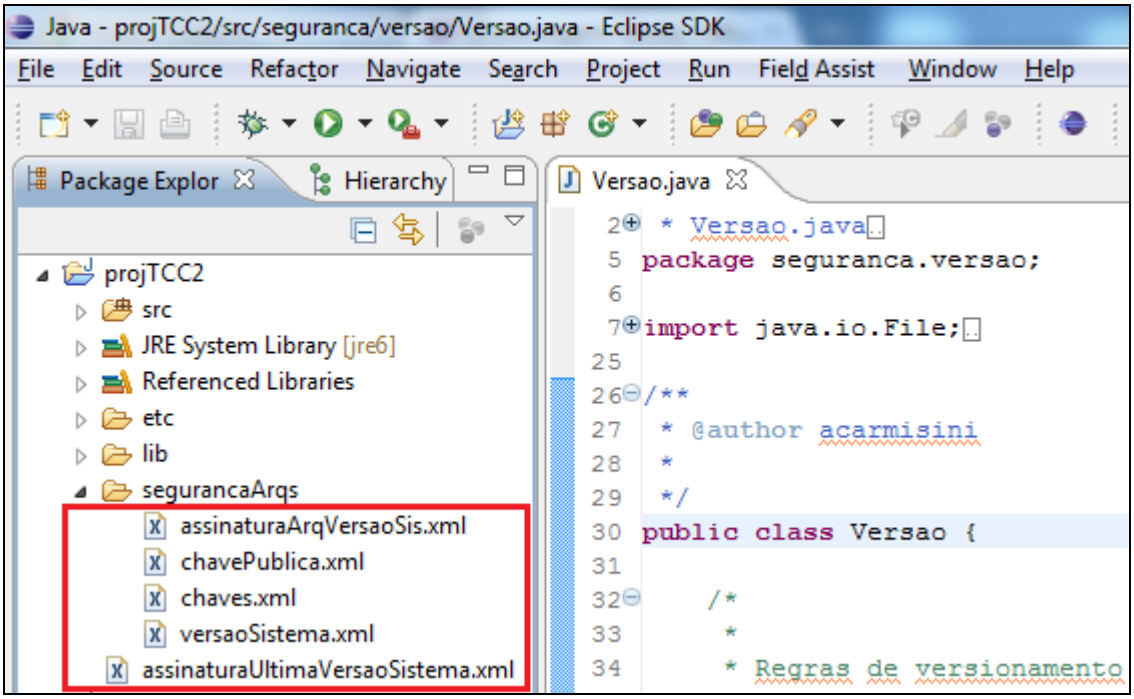

Figura 13 – Arquivos XML criados e o ambiente de desenvolvimento Eclipse

```
<list>
    <seguranca.assinatura.GravaXML>
        <versaoSistema>1.2.1</versaoSistema>
        <versaoBase>1.2.0</versaoBase>
        <chavePrivada>
            xtpraAPPnmdVe2qLUt8OmHraz/Uq+T615Er39oJnjS6XzdZOuagHte85aCTQuwtoCfCLxlF+TJCh
            8VI1o6kL8Uv7WvkuX19Cqbfpw5BDP2C3qHTB9D3A8N26IxUbRTSDBSlI1RoyqwWS41xr4hiFg/dz
            ITLhuFOhErOxrjiuPirGtjJaJ5bOtNbQk9EzUlsiecFVBaMzpq2DOF5HzMEOpDnW81HiZr+AjgbZ
            mpvgQwAu/kA1cLFQslNw7Rgf8NVsTiNp73j8tDqSrgWsuM9gEG9oAoa1wXEJZv9kETK1PubCwlDj
            RGlU51zRGXIAW+aPKwfuhfVSp/veCkuGEgIRMZXyIzSVRHFOD4U51QSfYO2Z9XBuGLjw/tJCw97N
            keqVYYOFhC4AAyZQO6kr6F4lTIYHrLwkY5z3P42wVUDn1bJ9qCbezGoovBh7Z1isvuI+Zm4Ur90l
            iyHeVBVB6S/A73zQZOOSGWsvTrMSaD4ZKWgdAV78NVB8r95hDKJPKE/ubFYENBT1SBZvcLrU1T3I
            +xaGf/sqIbcrMFsTlpdi43Hu+vBx5do67sp3DvJSv1MeagVF6rX4fzQpHGITmz3SI5B7iHXKQxJf
            P1VOsi+K93ONtWoJO3nkwPnFWhiMUNxY2ZdeXIXM95avjknrOlABChq5J9fV55dxyoNU6TKc8zlZ
            M2dS4EtUzMzzCn5+sNTfZsWo3YZGz90fF2aBcQfczZ5K1im/brgBBz7pnfMutCGOoLXMRxA/aP9a
            2AlkF3A/bOw6r8Cb2yleQRhMeVztBSurDnkGqXITryIJ113fewkjmA8VEKARObwZHKCXuKl7vid5
            IaTtJj5zjEm4mKT1HQ==</chavePrivada>
        <chavePublica>
            MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpVoSghMqhqa1eY2SV4jZdG+dOgrMo7IVgssKd
            BsW1Muf+GYStRj9REU7dm2nn6O2hSOxrhFhxNWdDOxf6ylsMbfe3kPaOoulyndx9NX8LdOcvviix
            jPSryNHd5Crs6Y2AedlaJF+Z44x/TdToGiISOSaG9MlincCaUyIisoicqwIDAQAB
        </chavePublica>
    </seguranca.assinatura.GravaXML>
</list>
```

Quadro 21 – Arquivo XML que armazena o par de chaves e a versão

Ao clicar no botão `Verificar Assinatura` é inicializada a interface (Figura 14), onde é informado o conteúdo (diretório ou arquivo(s)) e o arquivo que contém a assinatura digital (`assinaturaUltimaVersaoSistema.xml`).

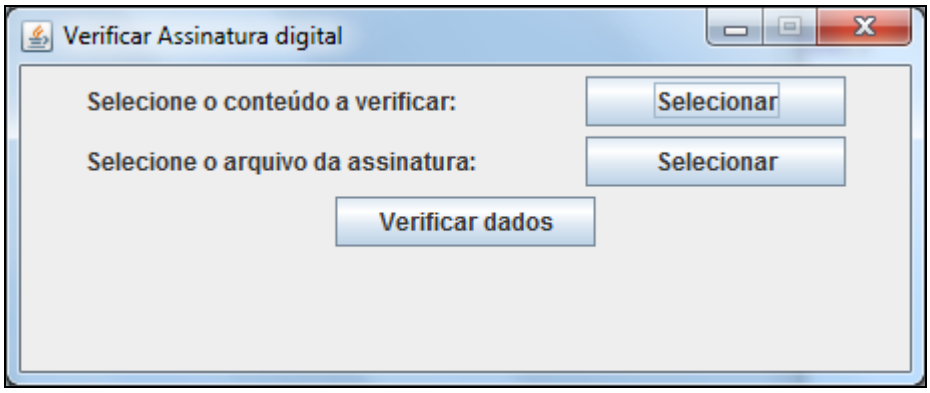

Figura 14 – Interface verificar assinatura digital

Após clicar no botão `Verificar dados` é verificada a integridade do conteúdo, onde é apresentada a mensagem “Verificação realizada com sucesso!” caso não haja alteração do mesmo, caso contrário é apresentada a mensagem “Os arquivos estão corrompidos”.

3.3.3.2 Operacionalidade da classe FPT – autoproteção e FCS – criptografia

Ao executar a classe `MainBaseDados`, é carregada a interface (Figura 15) responsável

pela geração da cópia de segurança e restauração da base de dados. Esta possui os botões `Conectar Base`, `Backup`, `Restore` e `Sair`.

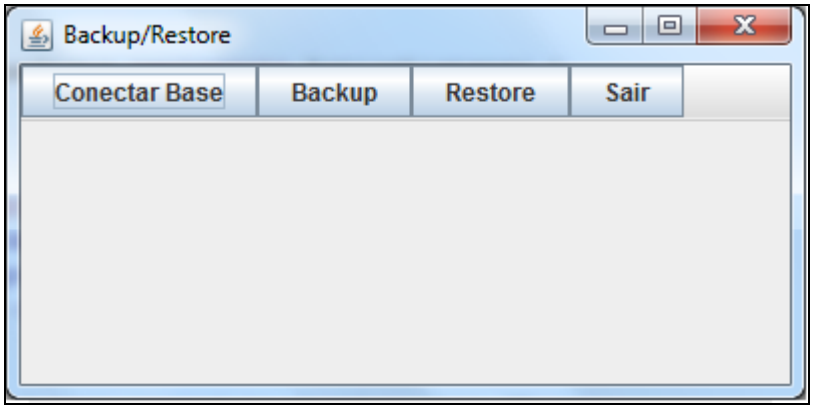

Figura 15 – Interface da cópia de segurança e restauração da base de dados

Ao clicar no botão `Conectar Base` é inicializada a interface (Figura 16) para conexão com a base de dados, a qual possui os seguintes campos:

a) `Host`: o endereço IP da base de dados;
b) `Porta`: o número da porta da base de dados;
c) `Usuário`: o nome do usuário da base de dados;
d) `Senha`: a senha do usuário;
e) `Base de dados`: o nome da base de dados.

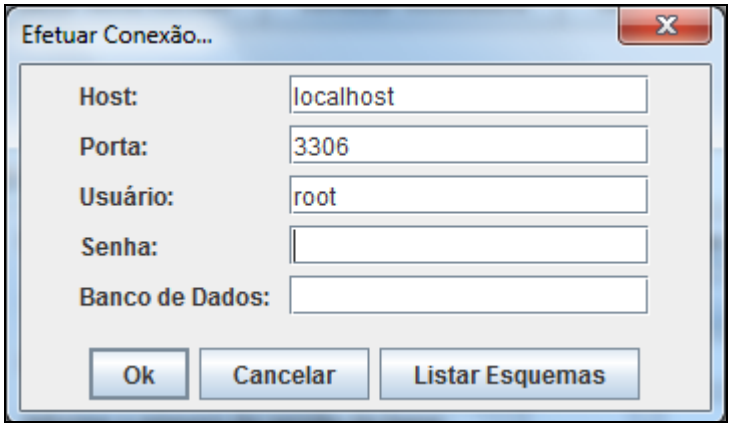

Figura 16 – Interface para efetuar conexão com a base de dados

Com os dados `Host`, `Porta`, `Usuário` e `Senha` informados é possível clicar no botão `Listar Esquemas`, para listar todos os esquemas cadastrados na base de dados. Ao clicar no botão `Ok` conecta-se na base de dados selecionada.

A cópia de segurança é gerada a partir do clique no botão `Backup`, onde o usuário informa o local onde será salvo o arquivo da cópia de segurança. O botão `Restore` é responsável pela restauração da base de dados, onde o usuário informa o arquivo da cópia de segurança a ser restaurado. O Quadro 22 apresenta um exemplo da cópia de segurança criptografada.

```
o3xMkj8JyRrHHmiUWmeexQcTghI4qiVbO8J69lriADlB+D6wXqhLhtOkwR1ZCwOOpkXH
8Ofby2R/tDk5eD37jJHlIUNK7qgvjOdWQ7xyfeP5eWzWzRdkyL+t+Pg=;
gTQ1w+rwOOmZLjL9AgbEtSJTjNnztgzNDkwT9zel4E4=;
BIZkBEw3utgkk6lb4G3AcmzLjCEUdigf2kh1WNwNqOnt9QX+GZeF4Wpul9jyNzXNv4pM
dHJhWDBHnTqmvcu55OaAC4vHMCR23chg588w2DvhTDrTlT7hTTo8jxjfscksVhguRiIn
```

Quadro 22 – Exemplo da cópia de segurança criptografada

3.3.3.3 Operacionalidade das classes FAU – auditoria e FTP – canais seguros

Os itens atendidos da classe `FTP` (descritos no anexo A) são:

a) `FTP_ITC.1` – canal confiável entre funções de segurança;

b) `FTP_TRP.1` – caminho confiável.

Para atender estes itens foi desenvolvido/configurado o sistema de autenticação de usuários, onde foi utilizado o servidor Apache TomCat o qual foi configurado localmente, para executar por meio do protocolo de comunicação *HyperText Transfer Protocol Secure* (HTTPS).

Também foi utilizada a biblioteca OpenSSL (YOUNG; HUDSON, 1999) responsável pela geração do certificado digital, este porém não é considerado válido, pois o objetivo é demonstrar a troca de mensagens entre cliente e servidor por meio de uma comunicação segura.

A Figura 17 apresenta a página inicial onde o usuário insere o seu nome de usuário e sua senha, após clicar no botão `Logar`, é verificado se os dados informados estão ou não corretos, estes dados utilizam a classe `Hash` onde é verificado o *hash* passado no campo da senha com o *hash* da senha armazenada. Por fim é gerada trilha de auditoria, conforme apresentado no exemplo do Quadro 23.

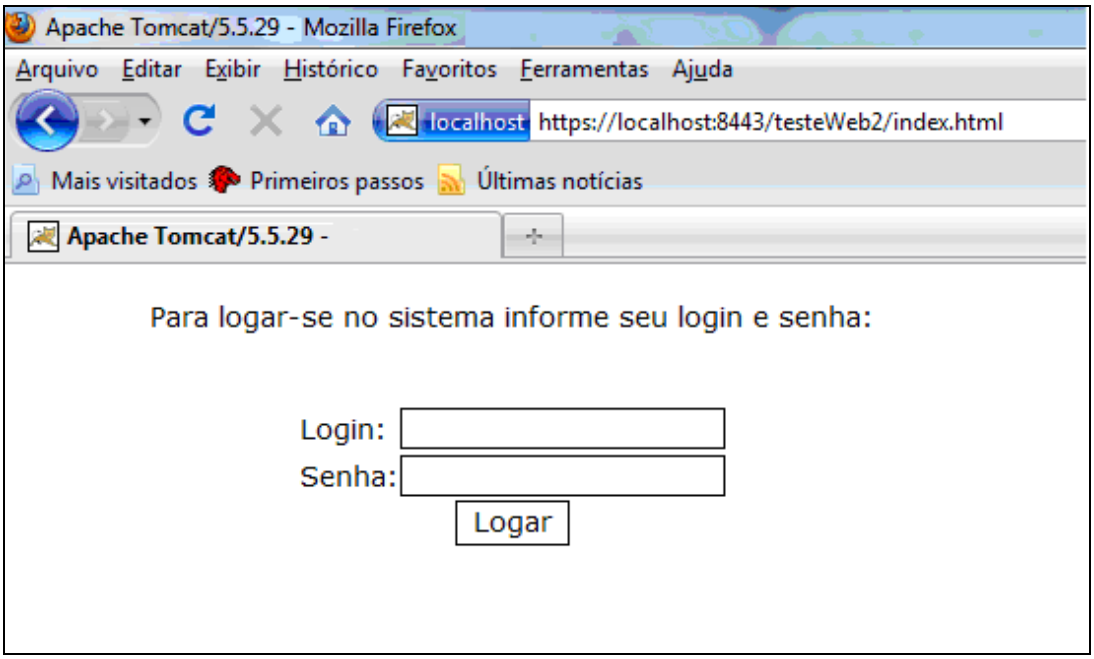

Figura 17– Tela de autenticação de usuários

```
<auditoria>
  <seguranca.auditoria.DadosAuditoria>
    <operacao>Login efetuado com sucesso!</operacao>
    <usuario>admin</usuario>
    <data>04/05/2010</data>
    <hora>09:57:06</hora>
    <versaoSistema>1.1.1</versaoSistema>
    <ip>201.54.196.81</ip>
    <macAddress>00-1E-68-20-3B-22</macAddress>
  </seguranca.auditoria.DadosAuditoria>
  <seguranca.auditoria.DadosAuditoria>
    <operacao>Logout efetuado com sucesso!</operacao>
    <usuario>admin</usuario>
    <data>04/05/2010</data>
    <hora>09:57:14</hora>
    <versaoSistema>1.1.1</versaoSistema>
    <ip>201.54.196.81</ip>
    <macAddress>00-1E-68-20-3B-22</macAddress>
    <assinatura>YocfdPd/DXjDECFSNUqr2qFs4bWj6dAW79IzTYn
    fJ3Ng6qpLXKh5SYkTcnYLOeFRH2698uuqVx20
lPiPzxHNQjqRG5d+15i7wQ9+KgK2OumzK2amQ9kCrFeMmVMDDFYUnL
Ni4dy1jOO67jPCmL8nhc86
00jWDRMmuh0WcMZqR/A=</assinatura>
  </seguranca.auditoria.DadosAuditoria>
</auditoria>
```

Quadro 23 – Exemplo de trilha de auditoria

3.4 RESULTADOS E DISCUSSÃO

Os resultados obtidos com o término do trabalho são satisfatórios, pois com a

utilização do *framework* no PRB foi possível suprir o requisito referente aos aspectos de segurança da informação, conforme a norma ISO/IEC 15.408, alcançando os objetivos iniciais.

No Quadro 24 apresentam-se as classes e seus itens de segurança atendidos com o desenvolvimento do *framework*.

Nome da Classe	Sigla	Itens atendidos
Proteção de dados do usuário	FDP	FDP_DAU.1 – autenticação básica dos dados
		FDP_DAU.2 – autenticação dos dados com identidade do gerador
		FDP_ETC.1 – exportação de dados com atributo de segurança
		FDP_IFC.1 – controle de fluxo de informação
		FDP_ITC.2 – importação de dados com segurança
Auditoria	FAU	FAU_GEN.1 – geração de dados para auditoria
		FAU_GEN.2 – associação do usuário ao evento de auditoria
		FAU_SAR.1 – revisão de 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
Autoproteção	FPT	FPT_ITC.1 – confidencialidade dos dados exportados pela aplicação
		FPT_ITI.1 – detecção de modificações;
		FPT_ITT.1 – proteção básica de dados internos da aplicação
Criptografia	FCS	FCS_COP.1 – operação de criptografia
Canais seguros	FTP	FTP_ITC.1 – canal confiável entre funções de segurança
		FTP_TRP.1 – caminho confiável

Quadro 24 – Requisitos de segurança implementados no PRB

O uso da versão 6.0 da linguagem Java facilitou a implementação dos itens da norma ISO/IEC 15.408, pois em se tratando de segurança faz-se necessário ter uma linguagem e um ambiente de desenvolvimento adequados.

No desenvolvimento, optou-se por uma padronização nos arquivos gerados e/ou lidos para o formato XML. Para isto foi utilizada a biblioteca XStream (XSTREAM, 2008), a qual facilitou o trabalho de manipulação dos mesmos. Também a utilização da API JDBC

(MYSQL, 2009) foi de suma importância para a implementação das classes `MainBaseDados` e `Restore`, facilitando a comunicação entre o *framework* e base de dados.

Para a realização dos testes do simulador de autenticação foi necessário configurar o servidor Apache TomCat para o uso do protocolo de comunicação HTTPS. Também utilizou-se a biblioteca OpenSSL (YOUNG; HUDSON, 1999), responsável pela geração do certificado digital, os quais foram utilizados para o desenvolvimento da operacionalidade do sistema de autenticação de usuários.

Em relação aos trabalhos correlatos o trabalho de Gomes e Santos (2006), foi realizado um estudo de caso onde se verificou algumas vulnerabilidades no ambiente de aprendizado *on-line* da Universidade da Amazônia, o Aprendiz, em que alguns requisitos estão em desacordo com as normas sugeridas pela ISO/IEC 15.408.

O trabalho correlato PASS teve como objetivo identificar os requisitos funcionais de segurança e a finalidade de auxiliar desenvolvedores na construção de software seguro, e por conseqüência software de maior qualidade.

O Quadro 25 apresenta as classes/famílias que possuem no mínimo um item implementado pelo *framework* em relação aos trabalhos correlatos.

Nome da classe/família	**Possui itens implementados?**		
	Este *Framework*	**Trabalho Gomes e Santos**	**Trabalho - PASS Nunes**
Auditoria	Sim	X	X
Proteção de dados do usuário	Sim	X	X
Criptografia	Sim	X	X
Autoproteção	Sim	X	X
Canais seguros	Sim	X	X
Autenticação	X	X	X
Acesso ao sistema	X	X	X
Gerenciamento de segurança	X	X	X
Utilização de recursos	X	X	X
Privacidade	X	X	X

Quadro 25 – Classes que possuem itens implementados pelo *framework* e trabalhos correlatos

4 CONCLUSÕES

O PRB tem por objetivo o desenvolvimento de uma solução de software e hardware voltada para o gerenciamento de gado de corte. Onde o público alvo não é da área de computação e, sabendo-se que os usuários leigos geralmente cometem erros de operação que desdobram em falhas no sistema, foi estabelecido o desenvolvimento deste *framework*.

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

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

Na criptografia foram implementados os métodos responsáveis por encriptar e decriptar o conteúdo, estes utilizaram dos algoritmos e chaves de criptografia. Na auditoria implementou-se as classes responsáveis por gravar a trilha de auditoria no arquivo XML, onde foi utilizada a biblioteca XStream.

Na classe/família proteção de dados do usuário foram implementados métodos responsáveis por gerar e verificar a assinatura digital dos arquivos. Também foi desenvolvido uma classe responsável pelo controle dos pares de chaves (chave pública e chave privada). Na autoproteção foi desenvolvida a classe `Backup` responsável por gerar a cópia de segurança e a classe `Restore` responsável pela restauração, estas classes utilizaram a API JDBC para a comunicação com base de dados.

O sistema de autenticação teve o objetivo de atender os itens da classe canais seguros e mostrar uma utilização das classes auditoria e *hash*, já que ambas foram implementadas para serem utilizadas em várias situações, assim também foi desenvolvida a classe de criptografia.

Com relação às técnicas e tecnologias utilizadas para o desenvolvimento do *framework*, a linguagem de programação Java em conjunto com a biblioteca XStream (XSTREAM, 2008) e a API JDBC (MYSQL, 2009) atenderam as expectativas permitindo que todos os requisitos elicitados fossem atendidos.

Com este trabalho pode-se concluir que o desenvolvimento de um *framework* que atenda os itens de segurança da informação é viável, porém torna-se muito complexo em se tratando de implementar todos os requisitos de segurança presentes na norma ISO/IEC 15.408.

4.1 EXTENSÕES

Como extensão para o presente trabalho propõe-se:

a) análise e implementação de outros itens de segurança segundo a norma ISO/IEC 15.408;
b) empacotamento do *framework* para utilização *stand-alone*.

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.

ALECRIM, Emerson. **Entendendo a certificação digital**. São Paulo, 2009. Disponível em: <http://www.infowester.com/assincertdigital.php>. Acesso em: 26 out. 2009.

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

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: 28 abr. 2010.

DATASUS. **Perfil de segurança**: requisitos funcionais de segurança. Sistemas financeiros/gerenciais. Rio de Janeiro, 2008. Disponível em: <http://pds.datasus.gov.br/PDS/downloads/PRS-PDS-002-PerfilSegFinanceiros.pdf>. Acesso em: 31 mar. 2010.

DAVIS, Noopur et al. **Processes to produce secure software**. [S.l.], 2004. Disponível em: <http://www.cigital.com/papers/download/secure_software_process.pdf>. Acesso em: 14 set. 2009.

DIAS, Cláudia. **Segurança e auditoria da tecnologia da informação**. Rio de Janeiro: Axcel, 2000.

ECLIPSE. [S.l.], 2008. Disponível em: <http://www.eclipse.org>. Acesso em: 10 mar. 2010.

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: 16 maio 2010.

MCGRAW, Gary; VIEGA, John. **Making software behave**. [S.l.], 1999. Disponível em: <http://www.ibm.com/developerworks/library/s-behave.html>. Acesso em: 11 set. 2009.

MYSQL. [S.l.], 2009. Disponível em: <http://dev.mysql.com/doc/>. Acesso em: 10 mar. 2010.

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.

OMG. **UML - Unified Modeling Language specification**. Version 2.0. [S.l.], 2005. Disponível em: <http://www.uml.org/>. Acesso em: 12 abr. 2010.

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, Ana P. S. S. **O que é hash e pra que serve**. [S.l], 2009. Disponível em: <http://www.baixaki.com.br/info/1663-o-que-e-hash-.htm>. Acesso em: 15 maio 2010.

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.

SPARXSYSTEMS. Creswick, 2000. Disponível em: <http:// www.sparxsystems.com/>. Acesso em: 15 mar. 2010.

XSTREAM. [S.l.], 2008. Disponível em: <http://xstream.codehaus.org/>. Acesso em: 10 mar. 2010.

YOUNG Eric A.; HUDSON Tim J. **The openssl project**. [S.l.], 1999. Disponível em: <http://www.openssl.org/>. Acesso em: 06 maio 2010.

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

No Quadro 26 é apresentando os requisitos funcionais de segurança segundo a norma ISO/IEC 15.408.

Classe	Atributo de segurança	Sigla	Item(s) do atributos de segurança
AUDITORIA - CLASSE FAU	FAU_ARP – Resposta automática a auditoria	FAU_ARP.1	Alarmes de segurança
	FAU_GEN – Geração de dados para auditoria	FAU_GEN.1	Geração de dados para auditoria
		FAU_GEN.2	Associação da identidade do usuário
	FAU_SAA – Análise da auditoria de segurança	FAU_SAA.1	Análise de violação potencial
		FAU_SAA.2	Detecção de anomalia baseada em perfil de uso
		FAU_SAA.3	Detecção de ataques por heurística simples
		FAU_SAA.4	Detecção de ataques por heurística complexa
	FAU_SAR – Revisão de dados da auditoria	FAU_SAR.1	Revisão de auditoria
		FAU_SAR.2	Revisão restrita de auditoria
		FAU_SAR.3	Revisão selecionável 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 de auditoria
		FAU_STG.2	Garantia da disponibilidade dos dados para auditoria
		FAU_STG.3	Ação em caso de possível perda de dados
		FAU_STG.4	Ação em caso de perda de dados
PROTEÇÃO DE DADOS DO USUÁRIO – CLASSE FDP	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 acessos de subconjuntos
		FDP_ACC.2	Controle de acesso completo
	FDP_DAU - Autenticação de dados	FDP_DAU.1	Autenticação básica dos dados
		FDP_DAU.2	Autenticação dos dados com identidade do gerador
	FDP_ETC - Exportação de dados para fora do controle do sistema	FDP_ETC.1	Exportação de dados sem atributos de segurança
		FDP_ETC.2	Exportação de dados com 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 informações	FDP_IFF.1	Atributos simples de segurança da informação
		FDP_IFF.2	Atributos hierárquicos de segurança da informação
		FDP_IFF.3	Fluxo ilícito limitado
		FDP_IFF.4	Fluxo ilícito parcialmente eliminado
		FDP_IFF.5	Monitoração do fluxo ilícito
	FDP_ITC - Importação de fora do controle das funções de segurança da aplicação	FDP_ITC.1	Importação de dados sem atributos de segurança
		FDP_ITC.2	Importação de dados com segurança

	FDP_ITT - Transferência interna	FDP_ITT.1	Proteção básica para transferência interna
		FDP_ITT.2	Proteção baseada em atributo para transferência interna
		FDP_ITT.3	Monitoração da integridade
		FDP_ITT.4	Monitoração da integridade baseada em atributos
	FDP_RIP - Proteção da informação residual	FDP_RIP.1	Proteção contra informação residual por subconjunto
		FDP_RIP.2	Proteção total contra informação residual
	FDP_ROL - Rollback (retorno)	FDP_ROL.1	Retorno básico
		FDP_ROL.2	Retorno avançado
	FDP_UCT - Confidencialidade de transferência de dados básica	FDP_UCT.1	Confidencialidade de transferência de dados básica
	FPT_SDI - Integridade de dados armazenados	FDP_SDI.1	Monitoração de integridade de dados
		FDP_SDI.2	Monitoração de integridade de dados com resposta
	FDP_UIT - Proteção de integridade de dados do usuário	FDP_UIT.1	Integridade na transferência de dados
		FDP_UIT.2	Recuperação com ajuda da origem
		FDP_UIT.3	Recuperação sem ajuda da origem
AUTENTICAÇÃO – CLASSE FIA	FIA_AFL - Falhas na autenticação	FIA_AFL.1	Tratamento de falha de autenticação
	FIA_ATD - Atributos do usuário para autenticação	FIA_ATD.1	Definição de atributos do usuário para autenticação
	FIA_SOS - Especificação de senhas	FIA_SOS.1	Métrica mínima das senhas
		FIA_SOS.2	Capacidade de gerar senhas
	FIA_UAU - Autenticação do usuário	FIA_UAU.1	Ações anteriores à autenticação
		FIA_UAU.2	Autenticação do usuário antes de qualquer ação
		FIA_UAU.3	Autenticação à prova 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 da autenticação
	FIA_UID - Identificação do usuário	FIA_UID.1	Ações anteriores à identificação
		FIA_UID.2	Identificação do usuário antes de qualquer ação
	FIA_USB - Ligação do usuário com o sistema	FIA_USB.1	Ligação do usuário com o sistema
GERENCIAMENTO DE SEGURANÇA – CLASSE FMT	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 de atributos de segurança
		FMT_MSA.3	Inicialização de atributos estáticos
		FMT_MTD.1	Gerenciamento de dados de segurança

	FMT_MTD - Gerenciamento de dados de segurança	FMT_MTD.2	Gerenciamento dos limites de dados de segurança
	FMT_SMF - Especificação do gerenciamento de funções	FMT_SMF.1	Especificação do gerenciamento de funções
	FMT_SMR - Papéis de gerenciamento de segurança	FMT_SMR.1	Papéis de segurança
		FMT_SMR.2	Restrição nos papéis de segurança
		FMT_SMR.3	Incorporação de papéis de segurança
AUTOPROTEÇÃO – CLASSE FPT	FPT_AMT - Teste da camada subjacente	FPT_AMT.1	Teste da camada subjacente
	FPT_ITA - Disponibilidade de dados exportados pela aplicação	FPT_ITA.1	Disponibilidade de dados exportados pela aplicação
	FPT_ITC - Confidencialidade dos dados exportados pela aplicação	FPT_ITC.1	Confidencialidade dos dados exportados pela aplicação
	FPT_ITI - Integridade dos dados exportados pela aplicação	FPT_ITI.1	Detecção de modificações
	FPT_ITT - Proteção na transferência interna de dados	FPT_ITT.1	Proteção básica de dados internos da aplicação
		FPT_ITT.2	Separação de dados de segurança
		FPT_ITT.3	Monitoração de integridade de dados
	FPT_PHP - Proteção física do sistema	FPT_PHP.1	Detecção passiva de ataque físico
		FPT_PHP.2	Notificação automática de ataque físico
		FPT_PHP.3	Resistência a ataque físico
	FPT_RPL - Detecção de repetição	FPT_RPL.1	Detecção de repetição
	FPT_RVM - Monitor de referência	FPT_RVM.1	Não-contorno da política de segurança
	FPT_RCV- Recuperação segura	FPT_RCV.1	Recuperação manual
		FPT_RCV.2	Recuperação automática
		FPT_RCV.3	Recuperação automática sem perdas
		FPT_RCV.4	Recuperação de função
	FPT_SSP - Protocolo de sincronismo de estado	FPT_SSP.1	Protocolo de sincronismo simples
		FPT_SSP.2	Protocolo de sincronismo mútuo
	FPT_STM - Registro de tempo	FPT_STM.1	Registros de tempo
	FPT_TDC - Consistência de dados entre funções de segurança	FPT_TDC.1	Consistência de dados entre funções de segurança
	FPT_TRC - Consistência de dados replicados	FPT_TRC.1	Consistência de dados replicados
	FPT_TST – Autoteste	FPT_TST.1	Autoteste
CRIPTOGRAFIA – CLASSE FCS	FCS_COP – Operação de criptografia	FCS_COP.1	Operação de 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 a chaves de criptografia

		FCD_CKM.4	Distribuição de chaves de criptografia
UTILIZAÇÃO DE RECURSOS – CLASSE FRU	FRU_PRS - Prioridade de serviços	FRU_PRS.1	Priorização de serviços limitada
	FRU_RSA - Alocação de recursos	FRU_RSA.1	Cota máxima de utilização
ACESSO AO SISTEMA – CLASSE FTA	FTA_LSA - Limitação de escopo ao sistema	FTA_LSA.1	Limitação de escopo no acesso ao sistema
	FTA_MCS - Limitação do número de seções simultâneas	FTA_MCS.1	Limitação básica do numero de seções
		FTA_MCS.2	Limitação básica do numero de sessões por usuário
	FTA_SSL - Travamento de sessão	FTA_SSL.1	Travamento automático de sessão
		FTA_SSL.2	Travamento de sessão por requisição do usuário
		FTA_SSL.3	Encerramento automático da sessão
	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
PRIVACIDADE – CLASSE FPR	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
	FPR_UNL – Não-rastreamento	FPR_UNL.1	Não-rastreamento
	FPR_UNO – Invisibilidade	FPR_UNO.1	Invisibilidade
		FPR_UNO.2	Alocação de informação com impacto na invisibilidade
		FPR_UNO.3	Invisibilidade sem solicitar informações
		FPR_UNO.4	Visibilidade autorizada
CANAIS SEGUROS – CLASSE FTP	FTP_ITC - Canal confiável entre funções de segurança	FTP_ITC.1	Canal confiável entre funções de segurança
	FTP_TRP - Caminho confiável	FTP_TRP.1	Caminho confiável

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

Quadro 26 - Requisitos funcionais de segurança segundo a norma ISO/IEC 15.408