UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

César Barone Marques Costa

MELHORIAS DE USABILIDADE E SEGURANÇA PARA O SGCI

Florianópolis(SC)

2012

César Barone Marques Costa

MELHORIAS DE USABILIDADE E SEGURANÇA PARA O SGCI

Trabalho de conclusão de curso para a obtenção
do Grau de Bacharel em Ciências da Computação.
Orientador: Jonathan Gehard Kohler
Coorientador: Ricardo Felipe Custódio

Florianópolis(SC)

2012

César Barone Marques Costa

MELHORIAS DE USABILIDADE E SEGURANÇA PARA O SGCI

Este Trabalho de conclusão de curso foi julgado aprovado para a obtenção do Título de "Bacharel em Ciências da Computação", e aprovado em sua forma final pelo Ciências da computação.

Florianópolis(SC), 15 de maio 2012.

Vitório Bruno Mazzola
Coordenador

Jonathan Gehard Kohler
Orientador

Ricardo Felipe Custódio
Coorientador

Banca Examinadora:

Ricardo Morais

Felipe Carlos Werlang

Lucas Gonçalves Martins

RESUMO

Softwares de gestão de infraestrutura de chaves públicas visam gerir o ciclo de vida de um certificado digital, criar e dar manutenção a uma ICP.
A proposta desse trabalho é apresentar melhorias nos quesitos de usabilidade e segurança dos dados do SGCI. Também visa propor ideias de futuras funcionalidades para o sistema baseado no estudo e pesquisa de outros SGCs.
Todas essas propostas juntas trarão resultados satisfatórios na a interação entre o usuário e o sistema, robustez e confiabilidade na segurança dos dados e uma reunião de funcionalidades atrativas disponíveis.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

No passado, o relacionamento do usuário com a interface de uma determinada aplicação não tinha prioridade alta na lista de requisitos de um software. Mas, felizmente, isso mudou. Os usuários estão mais exigentes quanto a facilidade de uso de um software e, embasado nisso, este trabalho visa como um dos objetivos, apresentar soluções efetivas para aumentar os níveis de usabilidade do SGCI.

O SGCI, sigla para Sistema de gestão de certificados ICPEDU, é um software web que foi concebido para gerir o ciclo de vida de um certificado digital na infraestrutura de chaves públicas educacional, ou, ICPEDU. Com ele pode-se criar e dar manutenção a elementos de uma ICP, tais como: certificados digitais, autoridades certificadoras, autoridades de registro e lista de certificados revogados.

O SGCI encontra-se atualmente em aprimoramento do código fonte. Aproveitando desse momento, este trabalho tem como intuito propor melhorias nos quesitos de segurança, para tornar o software mais seguro e menos vulnerável a ataques e também melhorias de usabilidade, proporcionando ao usuário final uma experiência mais agradável com o sistema.

1.1 OBJETIVOS

A seguir serão apresentados os objetivos a serem atingidos ao final deste trabalho, bem como as metas parciais.

1.1.1 Objetivos Gerais

Apresentar e implementar soluções efetivas para aumentar os níveis de usabilidade do SGCI, proporcionando ao usuário final uma experiência mais agradável com o sistema.

Em termos de melhorias de segurança, deseja-se apresentar e implementar soluções que aumentem ainda mais a segurança relacionada as operações no SGCI.

1.1.2 Objetivos Específicos

- Estudar conceitos de usabilidade em aplicações web e com o conhecimento adquirido, analisar o SGCI, propor e implementar melhorias.
- Estudar tipos de instalações de aplicativos, propor e implementar o tipo escolhido. Este deve possuir as seguintes características:

  Fácil utilização para o usuário final: Não tenha passos complexos de configuração.

  Atenda a maioria dos casos: Grande parte das instalações de softwares, não é feita de maneira customizada, onde são trocados os valores das opções *default* por outra qualquer. Busca-se então instalar o SGCI com as configurações padrões usada pela maioria dos seus usuários.
- Formalizar, propor melhorias de segurança para o protocolo de uso da chave privada de uma entidade e implementá-las.
- Implementar uma área administrativa para auxiliar o administrador do SGCI
- Propor e implementar melhorias no processo de desenvolvimento

1.2 JUSTIFICATIVA E MOTIVAÇÃO

O SGCI é desenvolvido para ser o software gestor da infraestrutura de chaves publicas educacional. Com essa grande responsabilidade atribuída a esse sistema, também vem o seu amplo uso em instituições de ensino de todo Brasil. Isso torna mais que necessário que esse sistema seja de qualidade, e não podemos ter um software de qualidade sem que se tenha conceitos e soluções de usabilidade aplicados nesse sistema. Ainda se tratando de um software na área de segurança, é crucial o conhecimento e aplicação de conceitos de segurança dos dados.

1.3 METODOLOGIA

Para a realização do trabalho, será efetuada uma pesquisa minuciosa em livros, na web e em normas a respeito do conteúdo a ser abordado. Para as funcionalidades, pretende-se realizar um estudo detalhado e um comparativo dos atuais SGCs mais populares existentes no mercado. Como resultado

desse estudo teremos um rico material contendo as funcionalidades mais interessantes de cada SGC pesquisado.

1.4 ORGANIZAÇÃO DO TRABALHO

O capítulo 2 aborda conceitos de segurança em computação, como: certificação digital, infraestrutura de chaves públicas, autoridades certificadoras e de registro, padrões e criptografia. As tecnologias utilizadas para desenvolvimento do trabalho são apresentadas também nesse capítulo.

O capítulo 3 engloba todas as melhorias de usabilidade no SGCI. Aborda melhorias na instalação, na interface gráfica e a criação de uma área administrativa.

As melhorias de segurança relacionadas a chave privada de uma entidade, são detalhadas no capítulo 4.

O capítulo 5 é o ultimo capítulo de desenvolvimento do trabalho. Apresenta o processo de desenvolvimento adotado no SGCI. Aborda aspectos relacionados a qualidade do código fonte e documentação.

Por último, no capítulo 6, as conclusões e os trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 CRIPTOGRAFIA

A palavra Criptografia tem sua origem no idioma grego, onde *kryptós* significa escondido e *gráphein* escrita. Da etimologia da palavra, conclui-se então que criptografia nada mais é que o ato de travestir a escrita através de códigos, ou simplesmente, escrever em códigos.

A história dos códigos e de suas chaves é a história de uma batalha secular entre os criadores de códigos e os decifradores uma corrida armamentista intelectual que teve um forte impacto no curso da história humana (SINGH, 2004). Reis, rainhas e generais tinham a necessidade de se comunicar de forma segura e eficiente, de modo a manter em sigilo as informações caso caíssem em mão erradas. A cifras criptográficas surgiam então e junto com elas grupos especializados em decifrar esses códigos.

Existiram milhares de cifras ao longo da história e a mais famosa é a cifra de César. Esta necessita de dois alfabetos. O normal e o cifrado. O cifrado corresponde ao normal mais um deslocamento. Usando um deslocamento de 3 posições, teríamos os seguintes alfabetos:

Normal: ABCDEFGHIJKLMNOPQRSTUVXZ
Cifrado:DEFGHIJKLMNOPQRSTUVXZABC

No exemplo apresentado, o cifrado é o alfabeto normal deslocado de três posições. Dessa forma, a cifragem de uma palavra ocorre da seguinte maneira: Troca-se cada letra da palavra a ser cifrada pela sua correspondente do alfabeto cifrado.

Palavra original: UFSC
Palavra cifrada: ZIVF

Como resultado, tem-se a palavra cifrada e sem significado algum. Se essa mensagem fosse interceptada por um atacante, ele não teria acesso ao conteúdo original, a não ser, é claro, se descobrisse a cifra utilizada.

O processo de decifragem é necessário apenas fazer o inverso. Para cada letra da palavra cifrada, busca-se no alfabeto cifrado trocando-a pela sua correspondente do alfabeto normal.

Palavra cifrada: ZIVF
Palavra original: UFSC

Um texto cifrado usando a cifra de César, pode ser decifrado através de um estudo de análise de frequência das letras mais comuns do idioma, trocando-as pelas mais comuns no texto cifrado, até que seja o texto faça sentido. Com o tempo esse método e seus similares deram lugar a métodos mais avançados de cifragem, surgindo então a criptografia simétrica e posteriormente a assimétrica.

2.1.1 Criptografia simétrica

Na criptografia simétrica é necessário um segredo pré-combinado entre as partes. O algoritmo pode ser divulgado. Esse segredo pode ser entendido como uma senha que juntamente com um algoritmo e o texto a ser cifrado, gera como saída um texto criptografado. A obtenção do texto original só será possível por quem tiver acesso ao segredo.

Para viabilizar esse processo, imagina-se duas pessoas, Alice e Beto. Alice deseja enviar uma mensagem sigilosa para Beto, então, os passos para viabilizar e enviar a mensagem serão os seguintes:

1. Alice e Beto combinam um segredo entre si através de um canal seguro

2. Alice cifra a mensagem com o segredo combinado anteriormente

3. Alice envia para Beto o texto cifrado

4. Beto decifra a mensagem utilizando o segredo combinado com Alice

A figura 2.1.1 ilustra os passos anteriores:

Os algoritmos mais populares de cifragem simétrica são: *Data Encryption Standard*(DES), e o *Advanced Encryption Standard*(AES) selecionado pelo governo dos EUA para substituir o DES.

Como pode-se ver na figura, a criptografia simétrica apresenta uma limitação, a combinação da chave entre as duas partes. Se a chave for interceptada por uma terceira parte, a mesma poderá abrir todos os documentos cifrados.

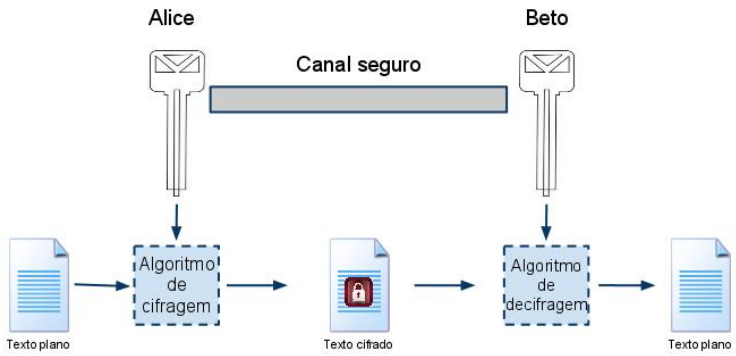

Figura 1: Cifragem simétrica

2.1.2 Criptografia assimétrica

A criptografia assimétrica, também chamada de criptografia de chave pública, possui esse nome porque tem-se duas chaves distintas, uma que deve ser mantida privada e outra que pode ser publicada(HOUSLEY; POLK, 2001). A criação destas chaves tem fundamento na aritmética modular e não é possível extrair a chave privada da chave pública. A cifragem e decifragem ocorre de forma assimétrica, onde um documento cifrado com uma chave pública, só pode ser decifrado com a respectiva chave privada

A criptografia assimétrica foi muito bem aceita pois não se tem mais a necessidade de um meio seguro visto que não há mais combinação de chaves. Outra característica importante é a redução na quantidade de chaves que um usuário deveria guardar para se comunicar com todos os possíveis outros usuários(MARTINA, 2005).

Se Alice deseja enviar um documento sigiloso para o Beto os passos são os seguintes:

1. Alice cifra o documento com a chave pública de Beto.
2. Alice envia para Beto o texto cifrado
3. Beto decifro o texto cifrado com a sua chave privada

A figura 2.1.2 ilustra o procedimento.

Enquanto a chave privada estiver segura, a confidencialidade do documento é garantida, caso a chave seja comprometida, não é possível garantir o sigilo do documento.

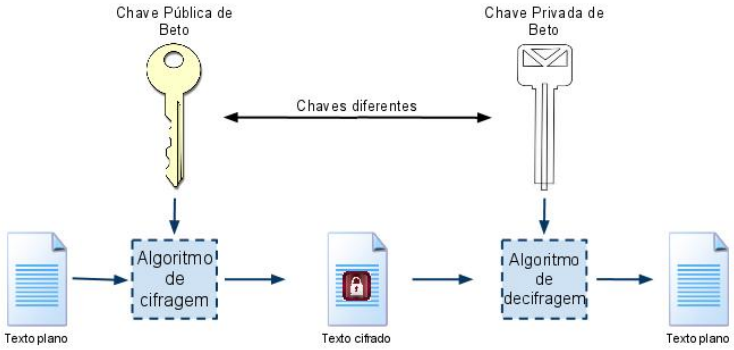

Figura 2: Cifragem assimétrica

Os algoritmos de geração do par de chaves e cifragem mais usados atualmente, são: *Elliptic Curve DSA*(ECDSA) e *o* RSA(MARTINA, 2005). A seguir será apresentado um dos usos da criptografia de chave pública, a assinatura digital.

2.1.3 Assinatura digital

Assinaturas digitais compartilham algumas funcionalidades com assinaturas manuscritas. Em particular, eles fornecem um método para assegurar que uma mensagem é autêntica de um usuário, ou seja, que ela origina da pessoa que alega ter gerado a mensagem (PAAR; PELZL, 2010). Se um documento é assinado digitalmente, as seguintes características são garantidas.

- Autenticidade: O receptor da mensagem pode validar se a assinatura do emissor é autêntica.
- Não-repúdio: O signatário não pode negar que assinou o documento.
- Integridade: Têm-se a garantia que a mensagem não foi adulterada.

Para assinar um documento digitalmente, primeiramente aplica-se uma função resumo no documento a ser assinado. O resultado gerado é assinado com a chave privada do signatário e anexado ao documento, como demonstra a figura 2.1.3.

Decifra-se a assinatura do documento com a chave pública do signatário, resultando em um resumo. Este resumo é comparado com o resumo

Figura 3: Assinatura digital

anexado ao documento. Se os resumos forem iguais, a assinatura é válida, como demonstra a figura 2.1.3.

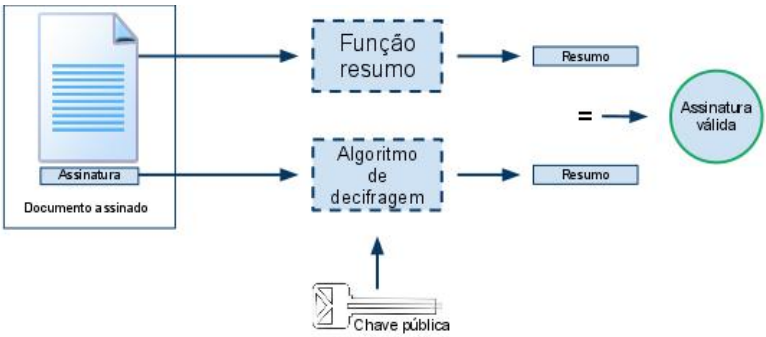

Figura 4: Verificar assinatura digital

2.2 INFRAESTRUTURA DE CHAVES PÚBLICAS

Documentos emitidos para pessoas, tal como um RG só são válidos pois foram emitidos por órgãos confiáveis ligados ao governo. Com a assinatura digital não é diferente, uma infraestrutura é necessária para que as assinaturas digitais sejam reconhecidas e tenham validade legal. Essa infraestrutura, denominada infraestrutura de chaves públicas (ICP) conta com alguns

elementos chave que serão apresentados nas próximas sessões.

2.2.1 Certificado digital

Com o advento da criptografia assimétrica, surge o desafio de comprovar quem é o detentor de uma chave privada correspondente a uma chave pública. O certificado digital surge então como resposta a esse desafio.

Um certificado ideal deve satisfazer as seguintes propriedades:(HOUSLEY; POLK, 2001)

1. deve ser um objeto puramente digital, para que possamos distribuí-lo através da internet e processá-lo automaticamente;

2. deve conter o nome do usuário que detém a chave privada, bem como informações de contato e da organização a qual pertence;

3. deve ser possível determinar se o certificado foi emitido recentemente ;

4. deve ser criado por um terceira parte confiável, ao invés do próprio usuário detentor da chave privada;

5. a parte confiável pode criar vários certificados, inclusive para o mesmo usuário e deve ser fácil diferenciá-los;

6. deve ser fácil determinar se o certificado é genuíno ou forjado;

7. deve ser a prova de alterações em seu conteúdo;

8. deve ser possível determinar de forma imediata se alguma informação no certificado deixou de ser válida;

9. deve-se poder determinar as aplicações para as quais o certificado é válido.

O padrão mais utilizado atualmente é o padrão x509 que se encontra atualmente na versão 3. Esse padrão especifica os campos que um certificado pode conter. Além do conteúdo de um certificado, o padrão x509 prevê extensões que dão mais flexibilidade a um certificado. Essas extensões estão descritas na RFC 5280 (COOPER et al., 2008).

Deve-se ter uma ferramenta que de suporte a oitava propriedade citada. Esse ferramenta é a Lista de certificados revogados (LCR) que será apresentada na próxima sessão.

2.2.2 Lista de certificados revogados

Um certificado digital pode deixar de ser válido, por comprometimento da chave da AC, da própria chave ou ainda pode necessitar de atualização dos seus dados. Por ser um objeto digital, pode facilmente ser copiado e redistribuído tornando impossível de se rastrear todas as cópias e destruí-las. A solução então é manter uma lista de certificados que não são mais válidos.

A ferramenta na ICP para disponibilizar informações a respeito do status de um certificado é a LCR. Esta contém uma lista de números serias de certificados revogados que não são mais confiáveis(HOUSLEY; POLK, 2001).

Se Alice deseja verificar se o certificado de Beto está válido, ela deve obter e consultar a LCR da entidade que emitiu o certificado de Beto.

Os campos que uma LCR pode conter estão especificados na RFC 5280(COOPER et al., 2008).

2.2.3 Autoridade Certificadora

A Autoridade Certificadora(AC), é a base de uma ICP. A AC é um coleção de hardware, software e pessoas que a operam. (HOUSLEY; POLK, 2001).

Uma AC emite certificados para entidades, finais ou não, afirmando que o conteúdo do certificado é válido bem como o requisitante do certificado detém a chave privada correspondente a chave pública contida no certificado. A AC pode adicionar informações no certificado que considerar válidas.

A AC ainda é responsável pela emissão de LCRs, podendo conter nessa o motivo da revogação dos certificados. Uma AC emite uma LCR e à assina com sua chave privada, atestando assim que o seu conteúdo é válido.

De acordo com, a AC é conhecida por dois atributos: seu nome e sua chave pública e tem as seguintes responsabilidades no escopo de uma ICP(HOUSLEY; POLK, 2001):

- Assinar certificados digitais
- Manter informações sobre o estado dos certificados e emitir LCR's
- Publicar certificados (não expirados) e LCR's, para que os usuários possam obter a informação que precisam para implementarem serviços de segurança
- Manter arquivos de informação de status de certificados expirados ou revogados

Uma AC inserida em uma ICP usada em larga escala, pode ser sobrecarregada realizando todas as tarefas pertinentes a ela. Então, pode-se delegar parte das tarefas a outra entidade, denominada, Autoridade de Registro.

2.2.4 Autoridade de Registro

Segundo Ignaczak (IGNACZAK, 2002), Uma Autoridade de Registro (AR) provê uma interface entre um usuário e uma AC. Ela e responsável por conferir as informações do usuário e enviar a requisição do certificado para a AC.

Uma AR pode estar comprometida a verificar os dados para várias ACs, assim como uma AC pode receber requisições de diversas ARs. As requisições aceitas e encaminhadas para a AC devem estar assinadas com a chave da AR.

2.2.5 Padrão de sintaxe de mensagens criptográficas PKCS #7

Os padrões de criptografia de chaves públicas (PKCS) são disponibilizados pelo Laboratório RSA para o desenvolvimento de aplicações seguras, e outros padrões baseados na criptografia de chaves públicas. (NADA, b).

O PKCS #7 faz parte dessa família de padrões e define a sintaxe de tipos de mensagens protegidas por criptografia, incluindo mensagens cifradas e assinadas digitalmente (NADA, b).

2.3 TECNOLOGIAS UTILIZADAS

Nas próximas subseções, será apresentado algumas das tecnologias envolvidas no desenvolvimento deste trabalho.

2.3.1 SQLite

SQLite é uma biblioteca(SQLITE, 2011a) que implementa base de dados SQL que:

- auto-contido: Necessita de pouco suporte de bibliotecas externas. Isso tem a grande vantagem de facilitar a criação de aplicações embarcadas(SQLITE, 2011c),

- Não possui um servidor de processos: Com o SQLite, o processo que deseja acessar o banco de dados lê e escreve diretamente a partir dos arquivos de banco de dados no disco. Não existe nenhum processo servidor intermediário.(SQLITE, 2011d).

- Não é necessário configurar: Para instalar o SQLite, basta instalar a biblioteca deste. Feito isso, nenhuma configuração adicional é necessária(SQLITE 2011b).

2.3.2 Apache

O projeto Servidor HTTTP Apache é um esforço para desenvolver e manter um servidor HTTP de código aberto para sistemas operacionais modernos como UNIX e Microsoft Windows NT (APACHE, 2011). O Apache é o servidor HTTP mais popular da internet desde 1996 (NETCRAFT, 2011).

2.3.3 OpenSSL

O OpenSSL (OPENSSL, 2011) é uma biblioteca, de código aberto e gratuita, sobre a licença estilo Apache que permite seu uso em aplicações comerciais ou não. É mantida por esforço de colaboradores do mundo inteiro e tem como propósito disponibilizar funções criptográficas através da implementação dos protocolos para camadas de sockets seguros(SSL v2/v3) e de transporte seguro (TLS v1) (DIERKS; RESCORLA, 2008).

É a biblioteca criptográfica mais utilizadas em softwares na área de segurança. O SGCI utiliza essa biblioteca indiretamente através do módulo php5-libcryptosec que por sua vez utiliza a libcryptosec.

2.3.4 Libcryptosec

A libcryptosec (LIBCRYPTOSEC, 2011) é uma biblioteca criptográfica, orientada a objetos, escrita na linguagem de programação C++. Ela é implementada utilizando o OpenSSL e abstrai diversas funções. Tem como objetivo facilitar e tornar mais flexível o desenvolvimento de aplicativos e soluções de criptografia.

2.3.5 PHP5-Libcryptosec

O php5-libcryptosec (PHP5LIBCRYPTOSEC, 2011) é um módulo para PHP que permite usar funções da libcryptosec em uma aplicação PHP. Ele surgiu da necessidade da realização de operações criptográficas em aplicações PHP e da carência de uma biblioteca para tal objetivo.

2.3.6 PHP

PHP (PHP, 2011) é uma linguagem de programação alto-nível,interpretada, não tipada e de domínio específico. É possível integrar um código escrito em PHP a um código em HTML.

2.3.7 Zend Framework

Zend Framework (FRAMEWORK, 2011) é uma framework concebido para ser de fácil utilização, escrito em PHP, orientado a objetos e carregado de boas práticas. Conta testes automatizados e é amplamente documentado. Adotou-se o uso do Zend Framework porque esse propicia ganho em produtividade, uma vez que abstrai diversas funções que antes teriam que ser implementadas e também em flexibilidade para, por exemplo, trocar a base de dados utilizada, uma vez que as operações em BD estão todas abstraídas pelo framework.

2.3.8 Servidor de integração Hudson

Hudson (CI, 2012) é um servidor de integração contínua escrito em Java e que roda em um contêiner como Tomcat ou Glassfish. Ele é utilizado para automatizar verificações de métricas de código, bem como rodar testes e gerar relatórios. O Hudson pode ser integrado com ferramentas de controle de versão e scripts em bash. Suas funcionalidades são bastante extensíveis e configuráveis com o uso de plugins, instaláveis através da sua interface web.

2.3.9 HTML

HTML abreviação para HyperText Markup Language, em português Linguagem de Marcação de Hipertexto é a linguagem padrão para criação

de paginas web. É uma linguagem interpretada e não é caracterizada como linguagem de programação.

2.3.10 CSS

Abreviação do inglês para *Cascading Style Sheets*, significa em português, Folhas de estilos em cascata. São utilizadas para definir estilo e formatação para linguagens de marcação, como: HTML e XML. Com a evolução do desenvolvimento de páginas web, surgiu a demanda por uma tecnologia, que concentrasse o estilo das páginas e facilitasse a manutenção. Atualmente, na versão 3, com efeitos para criação de páginas mais elaboradas e dinâmicas.

2.3.11 Framework Bootstrap

O framework Bootstrap é um framework CSS, cujo objetivo é fornecer uma gama de estilos para os mais variados elementos HTML. Oferece, também, um sistema de posicionamento para os elementos e para a criação de layouts. Ele facilita a personalização de páginas fornecendo componentes prontos para criação de barras de navegação, tabelas, mensagens informativas etc.

2.3.12 Phing

O Phing (PHING, 2012) é um sistema de construção de projetos em PHP, similar ao GNU Make. A configuração do phing, é feita através de um arquivo XML, que pode conter alvos para chamar ferramentas externas. Essas ferramentas podem ser de controle de versão, de teste automatizados, de geração de documentação, chamadas de sistemas etc.

2.3.13 Subversion

Subversion é um software de controle de versão, utilizado para manter a versão corrente e um histórico do projeto. Este disponibiliza um série de funcionalidades interessantes, como: fundir arquivos; suporte a logs a cada nova versão; visualização de diferenças entre diferentes versões de arquivos etc.

2.3.14 PHP Unit

Para a criação de testes automatizados para aplicações em PHP, é necessário uma ferramenta de suporte. O PHPUnit atende a essa demanda. Este disponibiliza um conjunto de implementações, para a escrita de testes automatizados e também, um aplicativo de execução dos testes. O PHPUnit (PHPUNIT, 2012) é a framework e o aplicativo mais difundido para esse tipo de tarefa em PHP.

2.3.15 PHP Documentor

PHP Documentor (PHPDOCUMENTOR, 2012) é uma ferramenta que possibilita a geração de documentação do código fonte. Para a geração, é necessário que o código fonte esteja documentado com anotações que o PHP Documentor reconhece.

2.3.16 PHP Code and Paste Detector

PHPCPD (PHPCPD, 2012) é uma ferramenta com o objetivo detectar códigos duplicados. Possui algumas configurações básicas, como o número de linhas e palavras, mínimas para identificar como uma duplicação.

2.3.17 PHP Mess Detector

O PHP Mess Detector (DETECTOR, 2012) é uma ferramenta versátil e completa para análise de códigos em PHP. Através dela é possível detectar problemas em potencial como:

- Bugs
- Códigos que podem ser melhorados
- Expressões complicadas
- Parâmetros, métodos e propriedades não utilizadas

3 MELHORIAS DE USABILIDADE

3.1 INTRODUÇÃO

Softwares como um todo, necessitam estar instalados e configurados para o seu correto funcionamento. Tratando-se de softwares *web*, tal qual o SGCI, a instalação pode não ser simples por envolver a configuração de ferramentas adicionais. Felizmente, esse esforço pode ser reduzido. As próximas sessões apresentarão soluções de instalações que podem vir a serem usadas no SGCI.

3.2 CARACTERIZAÇÃO DO PROBLEMA

O funcionamento do SGCI depende da utilização de 5 elementos: PHP, módulo PHP5-Libcryptosec, biblioteca Libcryptosec e o OpenSSL. Esses elementos se comunicam entre si para a realização de operações criptográficas. A figura 3.2 representa as camadas de dependências desde o SGCI até o sistema operacional.

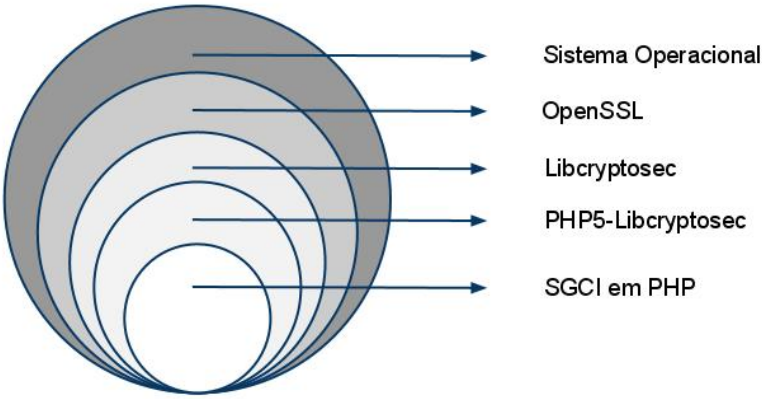

Figura 5: Camadas de dependências

Nem todas as versões das bibliotecas são compatíveis entre si. No caso do OpenSSL, a versão necessária para o SGCI, está disponível em poucas distribuições Linux. Era necessário que o usuário baixasse os elementos

citados, nas versões compatíveis entre si, compila-se cada um deles seguindo uma ordem específica e instalasse.

Esse tipo de tarefa, elevava o grau de complexidade de instalação do SGCI. Acarreta em perda de tempo com suporte técnico, tanto da parte dos desenvolvedores como do usuário que está tentando instalar o sistema.

As seções seguintes apresentarão soluções testadas para facilitar a instalação.

3.3 TIPOS DE INSTALAÇÕES

Há diversas abordagens possíveis de instaladores para serviços web, cada qual com suas vantagens e desvantagens.

3.3.1 Sistema auto-contido

Nesta modalidade de instalação, disponibiliza-se um pacote compactado, não instalável, contendo o código fonte do próprio software e todas as dependências necessárias. As bibliotecas são compiladas com as dependências estaticamente, gerando ao final da compilação, um arquivo binário com todas as funções agrupadas.

O pacote pode ser extraído em um diretório qualquer do sistema. Juntamente com esse pacote, disponibiliza-se um script que inicializa os serviços pertinentes, deixando o SGCI pronto para o uso. É necessário que este script seja executado manualmente.

3.3.2 Assistente de instalação

Neste modelo de instalação, tem-se um pacote que contém o software a ser instalado. As dependências a serem utilizadas são do próprio sistema operacional e caso não sejam satisfeitas, ou seja, não estejam instaladas no sistema, será impossível utilizar o software. A instalação consiste na execução de um script contido dentro do pacote disponibilizado. Este script é o assistente de instalação, que irá interagir com o usuário, através de telas para coletar informações de configurações e instalação.

3.3.3 Pacotes .deb

Algumas famílias de distribuições Linux contam com pacotes compactados instaláveis. Esses pacotes podem levar consigo uma lista das suas dependências de bibliotecas e outros aplicativos. São instalados e desinstalados por um software que acompanha o sistema operacional, comumente chamado de gerenciador de pacotes (no Debian, por exemplo, o dpkg). No ato da instalação do pacote, o gerenciador verifica se as dependências do pacote estão presentes no sistema operacional e caso não estejam, o pacote não é instalado.

Existe uma segunda ferramenta que agrega o gerenciador de pacotes. O gerenciador avançado de pacotes. Ela pode ser entendida como um automatizador de instalação de pacotes (no Debian leva o nome de *Advanced Packaging Tool*, ou simplesmente, APT). Com ele é possível no ato da instalação do pacote, baixar e instalar as dependências automaticamente, quando ainda não instaladas, facilitando a manipulação de pacotes. Na desinstalação de um pacote, as suas dependências são novamente verificadas e se não forem utilizadas por nenhum outro pacote instalado no sistema, são desinstaladas.

3.4 ANÁLISE DAS SOLUÇÕES PROPOSTAS

Os tipos de instalações podem ser analisadas sob diferentes óticas, cada uma possuindo vantagens e desvantagens em relação a determinado aspecto. Para a análise, criou-se uma tabela (ver 3.4) com as características que cada tipo de instalação possui. Essas características estão detalhadas a seguir.

Tabela 1: Características dos tipos de instaladores

Característica/Instalador	Auto-contido	Assistente	Pacotes .deb
Facilidade de instalação	X		X
Multiplataforma no Linux	X		
Disponibilização de novas versões	X		
Versatilidade		X	
Uso de repositórios			X
Checagem de dependências			X
Tamanho dos pacotes			X
Levantamento de dependências	X	X	X
Dificuldade de criação	X	X	
Dificuldade de realização de testes		X	
Necessidade de *workarounds*		X	

A facilidade de instalação, está atrelada a quantidade de decisões que o usuário terá que tomar. Quanto menos decisões, maior a chance do usuário instalar o software com sucesso.

Já multiplataforma no escopo Linux pois é possível que o sistema instalado seja isolado do sistema operacional, sendo assim, pode rodar em qualquer distribuição Linux. Para isso é necessário que todas as bibliotecas que são dependências estejam no diretório cujo software estará instalado.

No quesito de disponibilização de novas versões, recai na facilidade de atualizar uma instalação do software.

Possuir a característica de versatilidade está ligada a fazer uma instalação personalizada do software, como por exemplo, escolher diretório de logs do sistema, local de instalação do software, local da base de dados, se está vai estar remota ou não. Essa característica vai de encontro a facilidade de instalação. Quando a instalação é muito versátil, o grau de dificuldade de instalação aumenta também, pois é disponibilizado muitas opções para o usuário.

A possibilidade de usar um repositório para a disponibilização do software também é de grande valia, merecendo sem considerada como uma característica das abordagens de instaladores.

Podemos ainda citar a checagem de dependências automática, não ficando a cargo do usuário fazê-lo de maneira manual. Essa característica garante a integridade do sistema depois de instalado, uma vez que isso só acontece se as dependências forem satisfeitas.

O levantamento das dependências é necessária pois sem elas o software não funciona plenamente.

É possível ainda julgar um instalador pela dificuldade da sua criação. Por exemplo, pode se ter uma ferramenta que cria um pacote .deb instalável a partir do script de instalação daquela biblioteca.

Independente do tipo de instalação escolhida, é necessário que se tenha a certeza que esta funciona. Algumas abordagens de instaladores são mais fáceis de testar que outras. Um script em shell pode ser muito mais complexo de testar, por permitir diversas configurações para um sistema, que a simples instalação de um software através do gerenciador de pacotes.

Como nem todas as distribuições trabalham com a mesma estrutura de diretórios, pode ser que um arquivo de configuração, por exemplo, do banco de dados esteja em locais diferentes no sistema, necessitando que ambos os locais seja tratado no momento da instalação do software. Essa característica eleva o grau de dificuldade da criação do instalador.

3.5 ESCOLHA DE UMA ABORDAGEM

Do estudo dos instaladores, concluiu-se que o Assistente de instalação dificultaria para o usuário para a instalação de dependências, que teriam que ser compiladas e instaladas manualmente. Adotando-se o Sistema auto-contido apenas, a dificuldade recairia sobre o mapeamento de todas as dependências e sua posterior compilação. A solução dos pacotes isolada, não era possível, pois as distribuições Linux não contavam oficialmente com a versão necessária do OpenSSL.

Foi então que percebeu-se que a era possível uma solução mista utilizando o Sistema autoconvido e os pacotes .deb da seguinte maneira: Disponibiliza-se um pacote .deb que é instalado em um diretório específico do Sistema Operacional. Este contém um OpenSSL da versão requerida compilado e ligado dinamicamente com suas dependências. Então compila-se a Libcryptosec ligando-a estaticamente com OpenSSL e dinamicamente com demais dependências. O módulo PHP5-Libcryptosec também ligado estaticamente com a Libcryptosec já compilada. Por último, compila-se o PHP com o OpenSSL usando novamente linkagem estática e dinâmica para o restante das dependências. Todo esse processo está representado pela figura 3.5. Ao final desse ciclo de compilações tem-se em um só binário o módulo PHP5Libcryptosec, a Libcryptosec e o OpenSSL e segundo binário com o PHP e OpenSSL.

Não obteve-se êxito na implementação da solução. Depois de alguns testes, percebeu-se que as funções da Libcryptosec não conseguiam achar as funções do OpenSSL. Enquanto tentava-se achar o problema, surgiu uma ideia de uma nova solução para o instalador que não fora cogitada antes, a de uma distribuição Linux.

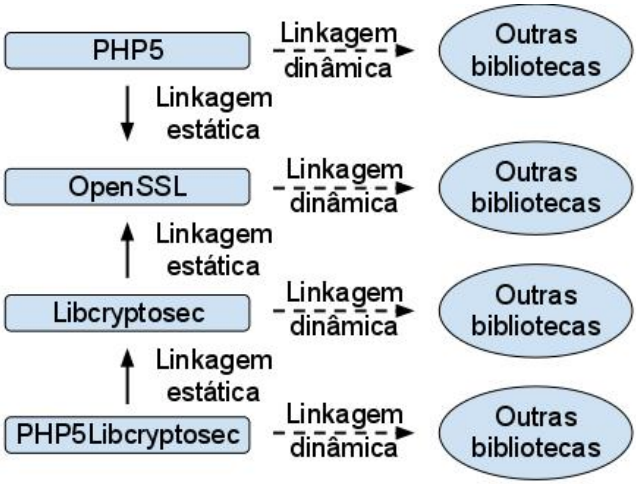

Figura 6: Esquema de linkagem de bibliotecas e aplicativos

3.6 DISTRIBUIÇÃO LINUX

O crescimento da virtualização nos dias atuais inspirou uma nova ideia para o instalador do SGCI durante o desenvolvimento do trabalho. Não se trata propriamente de um instalador, mas sim de uma distribuição Linux pronta para ser usada, na forma de *LiveCD* ou instalando o sistema operacional no disco rígido. Este LiveCD levava consigo uma instalação do SGCI e todos os seus requisitos (Libcryptosec, PHP, PHP5Libcryptosec e OpenSSL). Está solução apresenta os seguintes benefícios:

- **Possibilidade de testar o SGCI sem instalá-lo:** Através do LiveCD, é possível inicializar o sistema operacional para teste sem a necessidade de instalá-lo no disco rígido.
- **Instalação facilitada:** A instalação do SGCI se limitava a instalação do Sistema Operacional. Instalado, SGCI estaria pronto para uso.
- **Ambiente controlado:** Facilidade na correção de bugs e atualizações. Os usuários utilizam o mesmo sistema.

Para a implementação da solução, escolheu-se e modificou-se uma distribuição Linux. Pacotes desnecessários seriam retirados para reduzir o tamanho da imagem, objetivando um tamanho final de 200MB.

A distribuição Linux escolhida foi o Ubuntu 10.04 Desktop LTS por ser largamente difundida e ser a última versão LTS, cujo suporte é de 3 anos. A imagem da versão server e desktop, tem o mesmo tamanho, aproximadamente 700MB. Porém, a versão desktop conta com poucos aplicativos de interface gráfica, que ocupam muito espaço, como Gnome e OpenOffice, facilitando a redução de tamanho.

O desenvolvimento do LiveCD iniciou da instalação do Ubuntu em uma máquina virtual. Prossegui-se com a instalação dos pré-requisitos do SGCI e após essa etapa, gerou-se uma nova imagem através da ferramenta *Remastersys*. Ao final da geração, tinha-se uma imagem inicializável. Alguns testes foram efetuados na imagem inicializada como LiveCD. Atestou-se que o SGCI funcionava corretamente, porém, a imagem era muito grande, em torno de 900 *Megabytes(MB)*. Os próximos passos foram de redução do tamanho desta primeira imagem.

Iniciou-se um processo de desinstalação de pacotes desnecessários, começando com a interface gráfica e seus respectivos aplicativos. No ato da geração da nova imagem através do uso da ferramenta *Remastersys*, percebeu-se que esta baixava e instalava todos os pacotes anteriormente removidos, fazendo com que a distribuição voltasse ao seu tamanho original. Esse problema invalidava o uso da ferramenta a partir daquele ponto. Apelou-se para a customização manual. Primeiramente, montou-se a imagem em um diretório do sistema. Em seguida, utilizou-se o comando *chroot* para mudar o diretório raiz para o diretório montado. Desinstalou-se novamente os aplicativos de interface gráfica, dando origem uma imagem de 450 MB. Esta passou com sucesso por testes de inicialização, mas o tamanho da imagem ainda era elevado.

O próximo passo foi desinstalar bibliotecas e aplicativos de desenvolvimento e ainda pacotes de linguagens. O resultado foi uma imagem, com 250MB funcional. Este tamanho de imagem já serviria para o instalador do SGCI. Prosseguiu-se então com um teste de instalação dessa distribuição. Não obteve-se êxito, o sistema operacional inicializava corretamente, mas não havia um meio de instalá-lo. Pesquisando-se um pouco, descobriu-se que o instalador do Ubuntu necessitava das bibliotecas da interface gráfica o que traria um significativo aumento de tamanho para a imagem.

Surgiu então a ideia de customizar o Ubuntu 10.04 LTS para servidores. Examinando a imagem da distribuição, percebeu-se que esta levava consigo os pacotes .deb que eram instalados pelo gerenciador de pacotes no ato da instalação do sistema, impossibilitando a modificação manual.

Em um contexto geral, concluiu-se também que disponibilizar de uma distribuição acarretaria em problemas de atualização do PHP, OpenSSL, Libcryptosec e PHP5Libcryptosec na máquina do usuário. A cada nova atualização

desses componentes, haveria a necessidade de recompilá-los na máquina do usuário.

A principal dificuldade da instalação do SGCI era a versão do OpenSSL. Passou-se a procurar um distribuição Linux que contivesse essa versão. De acordo com site da Canonical, empresa responsável pelo Ubuntu, a versão 11.10 viria com atendia o requisito do SGCI quanto a versão do OpenSSL. Não necessitando a compilação e instalação manual dos pré-requisitos.

3.7 SOLUÇÃO IMPLEMENTADA

A abordagem escolhida foi gerar de um pacote .deb e disponibilizá-lo em um repositório compatível com a ferramenta *apt*. Essa solução apresenta as seguintes vantagens:

- Atualização do SGCI: Pode-se gerar novos pacotes do SGCI com correções de *bugs* e novas funcionalidades e disponibilizar no repositório. Os usuários atualizem as suas instalações através da própria interface do SGCI.
- Atualizações do OpenSSL e PHP: Usuário utiliza os repositórios oficiais para atualizar esses dois elementos.
- Instalações duplicadas: Elimina-se o risco do usuário instalar o PHP do repositório e ficar com duas versões do mesmo software em sua máquina.

Como a Libcryptosec e o módulo PHP5-Libcryptosec não contam com pacotes .deb, foi necessário criar os pacotes para essas duas bibliotecas.

3.7.1 Criação dos pacotes .deb

A geração de pacotes .deb é feita através da ferramenta *dpkg-deb*. Esta ferramenta gera um pacote dada uma estrutura de diretórios passada como parâmetro. A estrutura deve seguir um padrão especificado, conforme a figura:

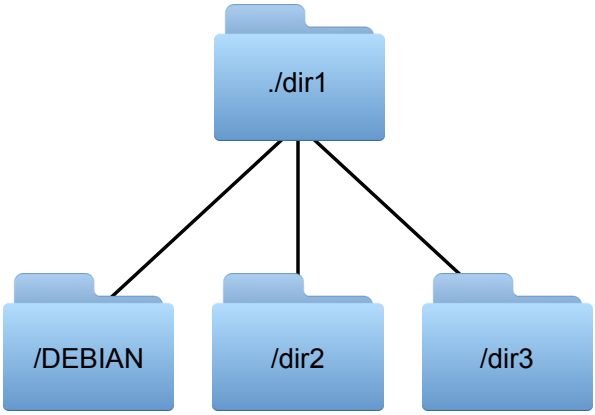

Figura 7: Estrutura de diretórios para criação de pacotes .deb

O diretório dir1 pode ter qualquer nome, e representa a raiz do sistema onde o pacote será instalado. Os diretórios dir2 e dir3 representam subdiretórios, que conterão arquivos a serem copiados para o sistema, seguindo a mesma estrutura do pacote. No exemplo, o diretórios dir2 e dir3 seriam copiados para a raiz do sistema com os mesmo nomes. O diretório DEBIAN é obrigatório e conterá scripts e arquivos configurações.

O primeiro passo na implementação da solução foi a instalação do Ubuntu 11.10 em uma máquina virtual. Em seguida, compilou-se o código fonte da Libcryptosec e obteve-se a um arquivo binário. O mesmo foi feito com o módulo PHP5Libcryptosec.

Com posse dos binários dessas duas dependências, prossegui-se com a criação da estrutura necessária. Essa estrutura contando com três diretórios: Um para geração do pacote do SGCI, um para a Libcryptosec e um para o módulo. A figura a seguir mostra o diretório de pacotes do SGCI.

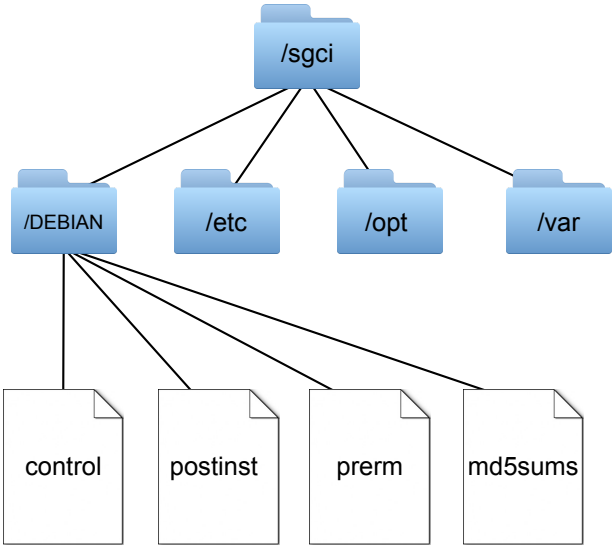

Figura 8: Estrutura de diretórios para criação de pacotes .deb

Os arquivos postinst e prerm são scripts executados após a instalação do pacote e antes da remoção. A seção 3.8 detalha esses scripts. O arquivo *md5sums* contém os *digest* de cada arquivo do pacote e são utilizados para comparar com os *digest* gerados na instalação. O arquivo *control* contém os informações gerais do pacote e uma lista de dependências.

Para a instalação dos pacotes via *apt*, é necessário a criação de um repositório. A seguir será apresentado a sua criação.

3.7.2 Repositório de pacotes

Um repositório é um espaço onde são guardados pacotes. Esse espaço pode ser um servidor remoto, um servidor local ou uma mídia. Para o SGCI, utilizou-se um servidor remoto. Os usuários que instalarão o SGCI estarão em outros locais físicos, diferentes do local do repositório. Também é inviável financeiramente, enviar uma mídia para cada usuário.

O repositório é utilizado pela ferramenta *apt*. No seu arquivo de configuração */etc/apt/sources.list* deve conter o endereço do repositório e o protocolo de transferência. O protocolo pode ser o *http*, *https*, *ssh* ou *ftp*. Para a implementação da solução, utilizou-se o https, por já ter-se o conhecimento necessário para a

configuração do servidor e garantir uma conexão segura entre os dois pontos.

É necessário também, um servidor *web* para estabelecer a conexão segura e redirecionar o cliente para o local correto. O servidor pode ser qualquer um e deve dar suporte ao protocolo escolhido. Escolheu-se o apache por experiência prévia de configuração.

O primeiro passo é a criação de um *virtual host* para o apache, que apontará para o diretório raiz do repositório. No virtual host deve estar especificado a utilização do protocolo *https*.

O diretório raiz do repositório pode ter qualquer nome. Dentro dele tem-se outro diretório cujo nome deve ser *binary*. Este segundo diretório conterá um arquivo nomeado de *Packages.gz* e os pacotes *.deb*. A figura 3.7.2 ilustra essa estrutura.

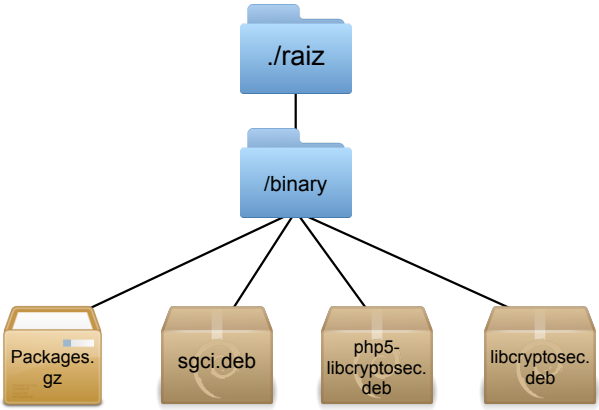

Figura 9: Estrutura de diretórios do repositório do SGCI

O arquivo Packages.gz é um arquivo compactado utilizado pelo *apt*. Ele contém o nome, a versão, o tamanho, a descrição e as dependências de cada pacote. O arquivo packages é gerado através de duas ferramentas: *dpkg-scanpackages* e *gzip*. A primeira faz uma varredura do diretório que contém os pacotes, coletando as informações de cada um deles. Essa informação é salva dentro de um arquivo que em seguida é compactado e gerando o arquivo final, Packages.gz.

O processo de geração de pacotes e publicação no repositório, pode ser automatizado. As seção seguinte abordará a automatização.

3.7.3 Automatização da geração de pacotes

A geração dos pacotes e publicação no repositório são tarefas repetitivas e podem ser automatizadas.

Implementou-se um script de automatização desse processo. Esse é responsável por baixar o código fonte do SGCI, da Libcryptosec e do módulo PHP5-Libcryptosec e compilar os dois últimos. Atualiza-se os arquivos *control* 3.7.1 de cada um dos pacotes com o tamanho do pacote, e a suas respectivas versões. Em seguida, atualiza-se os arquivos *md5sums* 3.7.1 de cada elemento. O passo final é a publicação no repositório, atendendo os pré-requisitos descritos em 3.7.2.

O próximo passo são as configurações de pós-instalação e pré-remoção.

3.8 CONFIGURAÇÃO DE PÓS-INSTALAÇÃO E PRÉ-REMOÇÃO

Pacotes *.deb* podem conter quatro *scripts* que são executados pelo *apt*. O *preinst* é executado antes da instalação do pacote, o *postinst* após a instalação, o *prerm* e *posrm* executados antes da remoção e após a remoção, respectivamente. O uso de todos os scripts não é obrigatório. Para o SGCI utiliza a pós-instalação e a pré-remoção. As seguintes tarefas são executadas pelo *script* de pós-instalação:

- Popular a base de dados
- Setar permissões para usuário do apache do diretório de *logs*
- Setar permissões para o usuário do apache na base de dados do SGCI
- Ativar módulo *rewrite* do apache
- Reiniciar apache
- Gerar par de chaves e certificado auto-assinado para o servidor SSL

A pré-remoção executa as seguintes tarefas:

- Desativar o *virtual host* do SGCI
- Apagar o par de chaves e o certificado SSL
- Remover os logs

3.9 MELHORIAS GERAIS DE INTERFACE GRÁFICA

A interação do usuário com um sistema é feita através de uma interface gráfica. Em termos de interface gráfica pode-se aplicar o conceito de usabilidade que é definida pela norma ISO9241 - Ergonomia de software de escritório ISO9241-11 como: A medida pela qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação.

O estudo de usabilidade realizado para o SGCI, resultou em propostas e implementações de uma de melhorias, que serão apresentadas a seguir. Como produto final, destaca-se a implementação de uma interface mais intuitiva, agradável e limpa.

O design de interfaces web, é feito através de folhas de estilos, ou CSS 2.3.10. Com o uso dessa tecnologia é possível posicionar os elementos HTML 2.3.9 na tela, fazer mudanças na tipografia e aplicar alguns efeitos dinâmicos mais básicos. Um dos maiores problemas no uso dessa tecnologia provém dos navegadores que não seguem com precisão a norma especificada pelo órgão W3C, para interpretação dessa linguagem. O descumprimento da norma afeta a visualização da página. O design da página sofre alterações de acordo com a interpretação de cada navegador. Muitas vezes, para garantir que a página web seja mostrada igualmente, é necessário *work around's*, também conhecidos como *hacks*. *Hacks* são implementações CSS para corrigir um determinado erro de interpretação de uma propriedade.

Para garantir que o SGCI seja visualizado igualmente, independente do navegador, utilizou-se o framework Bootstrap 2.3.11. Escolheu-se a utilização do Bootstrap por ser versátil, de fácil utilização, com um bom design e trabalhar com sistema de grades para posicionamento.

3.9.1 Barra de navegação

A barra de navegação do sistema é por onde o usuário do SGCI, tem acesso as funcionalidades disponíveis. Ela deve ser intuitiva e natural para o usuário e é garantida pelas seguintes características, que podem ser visualizadas na figuras 3.9.1. e 3.9.1.

- Alto contraste: Menus em alto contraste facilitam a diferenciação visual de categorias e subcategorias. A cor de fundo da barra de navegação é preta, contrastando fortemente com o nome das categorias e a cor de fundo das subcategorias.

- Identificação visual do papel: No menu do SGCI, do lado direito é possível visualizar o nome do usuário bem como o papel que ele está logado no sistema. A cor do botão depende do papel corrente. Para operador, o botão é mostrado em amarelo, para administrador, vermelho e azul para o criador, independendo se o usuário está logado em um AC ou uma AR.
- Identificação visual da entidade: Ao lado esquerdo da identificação do usuário é mostrada, quando houver, a entidade que o usuário está logado. Com um click no nome da entidade, o usuário é redirecionado para uma página que mostra mais detalhadamente os detalhes da entidade em questão.

Figura 10: Barra de navegação

Figura 11: Identificação visual do papel, da entidade e menus em alto contraste.

3.9.2 *Breadcrumbs*

Breadcrumbs, no português, migalhas de pão, são uma identificação visual para mostrar ao usuário o caminho da página inicial até a página corrente. Ficam no topo da página, logo abaixo da barra de navegação e em fonte pequena. Os *breadcrumbs* do SGCI podem ser vistos na figura 3.9.2.

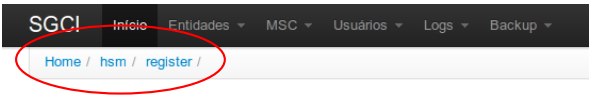

Figura 12: Breadcrumbs

3.9.3 Tarefas pendentes

Cada papel de usuário no SGCI desempenha tarefas que necessitam serem realizadas com frequência. Um operador de AC, por exemplo, deve aprovar pedidos de certificados e pedidos de revogação. Com intuito de facilitar a realização dessas tarefas, foi implementado no SGCI, na página inicial de cada usuário, uma listagem de tarefas pendentes. Clicando em uma tarefa pendente, o usuário é redirecionado para a página correspondente a tarefa.

Figura 13: Tarefas pendentes

3.9.4 Listagem e ícones

No SGCI, a apresentação de alguns dados é feita através de listagens. Tem-se listagem para entidades, para usuários, para certificados etc. As listagens conseguem apresentar uma quantidade significativa de conteúdo em pouco espaço. Porém, se não estiverem bem organizadas, podem poluir a

página e dificultar a visualização dos dados. Com intuito de organizar as listagens do SGCI, criou-se abas que as categorizam. Uma listagem de usuários, por exemplo, pode contar com duas abas, uma para listar os usuários operadores e outra os usuários administradores 3.9.4.

As listagens contam com ícones para as ações disponíveis relativas a uma entrada. Essas ações podem ser aprovação ou revogação de um certificado, deleção de uma entidade, deleção de um usuário etc. Cada ação está associada a um ícone na última coluna da listagem 3.9.4. Esse ícones devem ser intuitivos. Utilizou-se a biblioteca de ícones do framework Bootstrap 2.3.11.

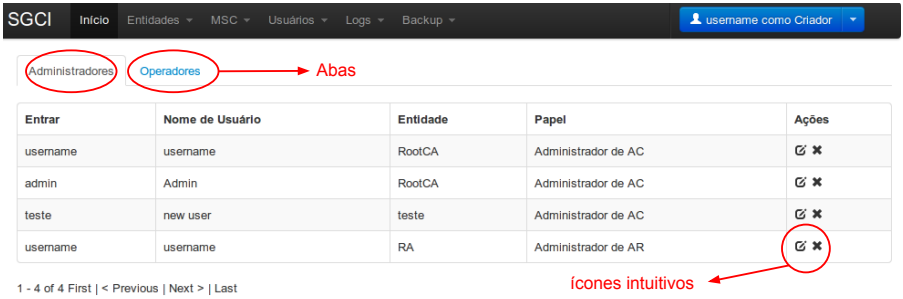

Figura 14: Listagem com abas e ícones intuitivos

3.10 ÁREA ADMINISTRATIVA

O funcionamento do SGCI, depende de configurações em arquivos no sistema. Esses arquivos podem ser alterados manualmente pelo administrador. Porém, essa maneira não é aconselhável, pois exige que o administrador tenha conhecimento da estrutura interna do sistema. Ainda, alterações manuais, se feitas de maneira incorreta, acarretam em um mau funcionamento do sistema.

Uma maneira de reduzir essa lacuna de erros, é a através da implementação de uma área administrativa com uma interface gráfica. As configurações antes feitas diretamente nos arquivos, passam a ser feitas por intermédio desta interface. Tais configurações são validadas, antes de serem salvas, evitando erros de sintaxe.

A criação dessa área permite ainda, adicionar novas flexibilidades no sistema. Atualmente, foram adicionados duas funcionalidades: manipulação de backups e configurações do servidor SSL. A manipulação de backups in-

clui a geração e restauração, feitas anteriormente por outro usuário do sistema, passou a ser responsabilidade do administrador. As configurações do servidor SSL anteriormente, eram feitas substituindo-se os arquivos do certificado digital e a chave privada, geradas no ato da instalação.

A seguir, será apresentado mais afundo, as interface administrativa e as funcionalidades de manutenção e configuração.

3.10.1 Configurações de idioma

No atual mundo globalizado, é necessário que os softwares estejam prontos para serem usados em diferentes idiomas. Para solucionar esse problema é necessário que o software esteja internacionalizado. A internacionalização de softwares, é definida como (SAVOUREL, 2001): "O processo de desenvolvimento de um produto de forma a funcionar com dados em diversos idiomas e que possa ser adaptado a diversos mercados sem alterações de engenharia"(SAVOUREL, 2001). Um software internacionalizado garante que sua visualização será idêntica em qualquer idioma escolhido.

As configurações de um software internacionalizado são feitas através da definição de uma localidade. Esta é responsável por informar um idioma e um local físico, que pode ser entendido como um país.

O SGCI através do Zend Framework está preparado para trabalhar com novas localizações. Porém,apenas duas localizações são disponibilizadas por padrão. São elas: pt_BR, que indica o Brasil e o idioma português e en_US, cujo país é os Estados Unidos da América e o idioma inglês.

A configuração de novos idiomas consiste em dois passos: Primeiro, salva-se no diretório de idiomas do SGCI, um novo arquivo de localização contendo as traduções para o idioma requerido. O arquivo deve seguir o padrão idioma_pais. O idioma deve estar de acordo com a norma ISO6391 que especifica a nomenclatura para cada idioma. Para o país, a abreviação deve estar de acordo com a norma ISO3166 . O segundo passo é configurar o SGCI para trabalhar com esse novo idioma. Esta configuração consiste em mudar uma diretiva em um arquivo. Os passos citados anteriormente foram automatizados e serão detalhados a seguir.

3.10.1.1 Cadastrar novos idiomas

Implementou-se um formulário, contendo 3 campos para o cadastro de novos idiomas:

- **Arquivo**

- **Idioma**
- **País**

Após o upload, o arquivo passa por uma validação para verificar se é um arquivo válido de idioma. No caso de sucesso, é salvo no formato idioma_pais em um diretório do SGCI e está pronto para uso. A figura 3.10.1.1 abaixo mostra o formulário de cadastro de idiomas.

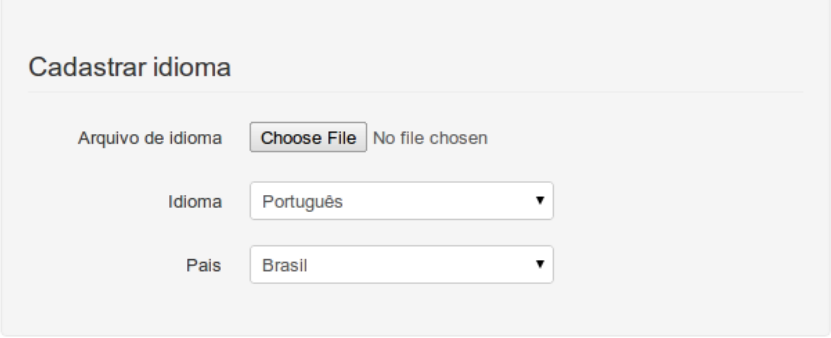

Figura 15: Cadastro de um novo idioma

3.10.1.2 Administração de idiomas existentes

A administração de idiomas existentes é feita através da listagem dos idiomas cadastrados. Através da listagem pode-se deletar idiomas ou setar um idioma como padrão. A figura 3.10.1.2 mostra a listagem.

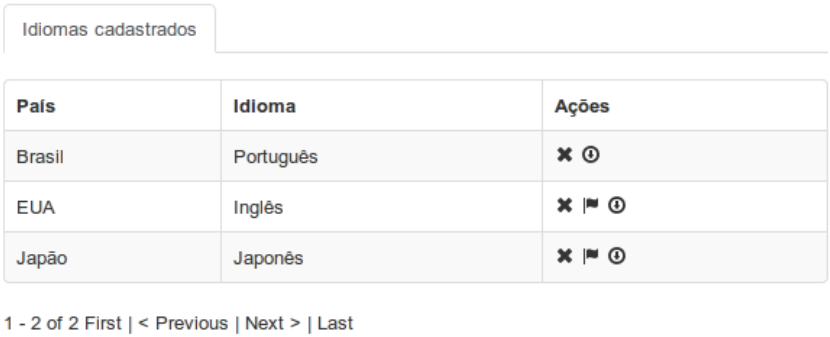

Figura 16: Listagem de idiomas

3.10.2 Backup

O backup é de crucial importância em todas as instâncias. Softwares devem implementar funcionalidades de backup. Pessoas devem ter backup do seus arquivos. Fazer backups constantemente, pode evitar transtornos em caso de perdas de dados. No escopo do SGCI, o prejuízo seria grande caso houvesse um comprometimento do sistema. Dados de certificados emitidos, logs de operações, vínculos entre entidades e mais uma série de outros dados e configurações seriam perdidos. É inviável que uma aplicação do porte e responsabilidade do SGCI esteja em produção, sem que se tenha um sistema eficiente e seguro de backup.

O SGCI concentra os seus dados em um banco de dados, facilitando a criação e restauração do backup. A seguir serão apresentadas as funcionalidades relativas a essa tarefa.

3.10.2.1 Exportação

A exportação do backup no SGCI, através da interface web, consiste na criação de um arquivo compactado contendo uma cópia da base de dados. Este arquivo é cifrado no ato da exportação, com uma senha escolhida pelo administrador através do formulário. A figura 3.10.2.1 mostra o formulário de exportação do backup.

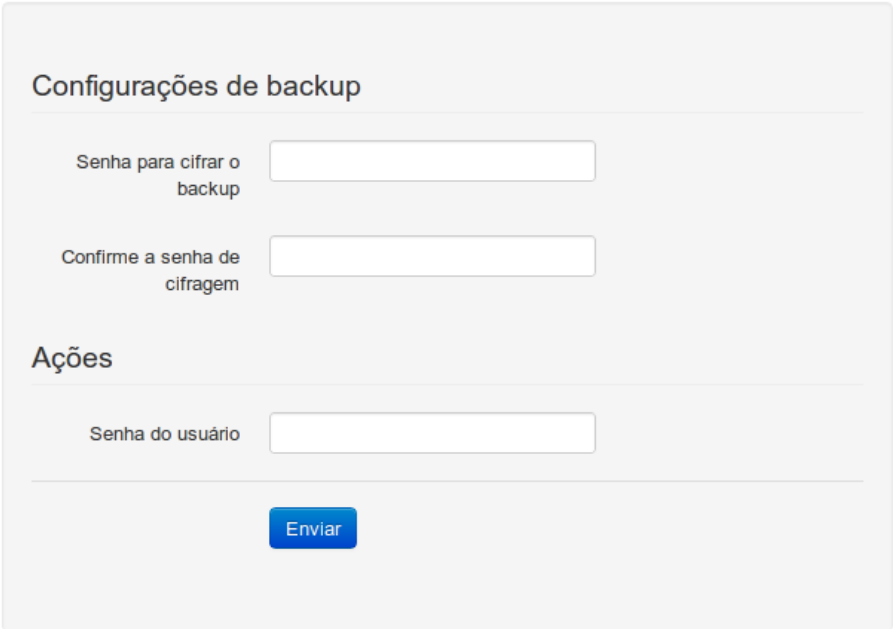

Figura 17: Exportação do backup

3.10.2.2 Restauração

A restauração do backup consiste no *upload* de um arquivo gerado através da exportação. O administrador escolhe o arquivo para upload e coloca a senha que foi utilizada para cifrá-lo. O SGCI decifrará o arquivo seguido o descompactará. A base de dados atual é substituída pela do backup. A figura 3.10.2.2 mostra o formulário de restauração do backup.

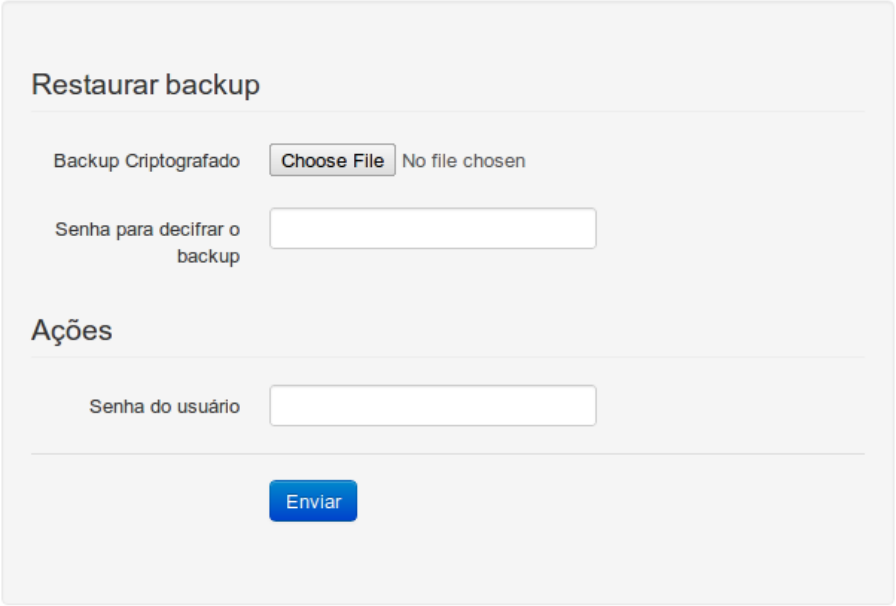

Figura 18: Restauração do backup

3.10.3 Servidor SMTP

Algumas funcionalidades futuras do SGCI exigirão a configuração de um servidor de e-mail SMTP. Não há necessidade que este servidor seja para cada entidade, podendo assim, ser configurado globalmente para uma instalação do SGCI. Algumas dessas funcionalidades são:

- **Notificação de erros:** Envio de e-mail para o administrador, em caso de anomalia no sistema.
- **Recuperação de senha:** Usuário poderá recuperar sua senha.
- **Expiração do certificado de entidades:** Envio de e-mails quando o certificado de uma entidade está próximo de ser expirado.
- **Relatório de uso:** E-mail informando a quantidade de acessos ao SGCI, quantos certificados foram emitidos etc.

É necessário que o servidor SSL esteja cadastrado. O SGCI só trabalha com um servidor que é configurado através de um formulário **??**.

3.10.4 Servidor SSL

Com o intuito de proporcionar mais segurança na comunicação entre o cliente (navegador) e o servidor com o SGCI em um ambiente de produção, é recomendável que este seja instalado utilizando-se um certificado para servidor SSL. Com a utilização desse certificado, a comunicação entre o cliente e o servidor será cifrada. Um atacante que tenha acesso as mensagens trafegadas, não conseguirá ver o seu conteúdo.

A instalação do SGCI, através do repositório de pacotes, se responsabiliza para a criação de um certificado SSL auto-assinado. Porém, o certificado emitido no ato da instalação não é assinado por uma Entidade confiável, devendo ser substituído por um certificado válido.

Com propósito de oferecer maior facilidade para a troca do certificado SSL, incluiu-se na área administrativa uma funcionalidade para upload de um arquivo no formato PCKS #12. Após o upload, o antigo certificado do SGCI é substituído pelo contido no arquivo, bem como a chave privada. O administrador então é orientado a reiniciar o servidor web. Uma vez reiniciado, o SGCI passa a trabalhar com o novo certificado. A figura 3.10.4 mostra o formulário de upload do PKCS #12.

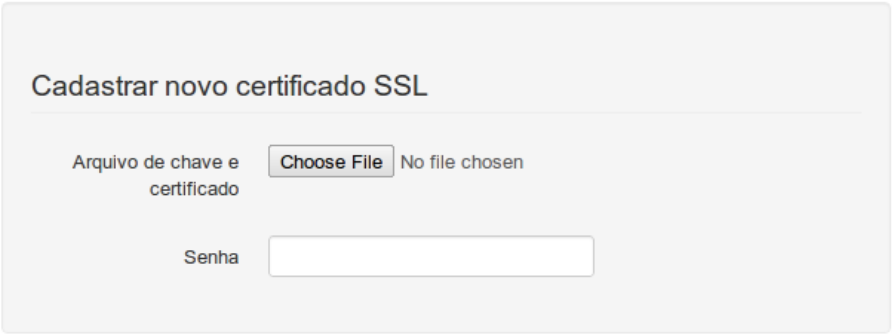

Figura 19: Cadastro de um novo par de chaves e certificado SSL

4 MELHORIAS DE SEGURANÇA

4.1 PROTOCOLO DE ACESSO A CHAVE PRIVADA DE ENTIDADES

4.1.1 Introdução

A chave privada de uma entidade deve ser guardada em segurança. Se um atacante tem acesso chave, ele pode emitir certificados falsos se passando pela entidade, fazendo os usuários acreditarem nesses certificados. Para proteger essa chave privada, a entidade comumente utiliza um hardware criptográfico, podendo ser um Módulo de segurança criptográfico, um smartcard ou ainda um PCMCIA. A chave da entidade pode estar armazenada também, em mídia comum. O SGCI pode armazenar a chave no disco, porém, é necessário que se tenha um mecanismo de segurança que garanta o acesso restrito e seguro a chave privada.

4.1.2 Caracterização do problema

Os hardwares citados anteriormente, garantem níveis de segurança físico e lógico. Um MSC, não permite a exportação da chave privada por software e na tentativa de acesso a mídia de armazenamento manualmente, ela se autodestrói. Uma mídia de armazenamento comum, como um disco rígido ou um pendrive, não possui mecanismos de proteção física, necessitando que a proteção seja feita via software. A solução para guardar a chave nesse tipo de mídia, em segurança, é armazená-la cifrada.

No SGCI 2.0.0, é possível configurar uma AC ou uma AR armazenando sua chave privada em disco ou MSC. Quando a chave está em disco, deve estar cifrada. A necessidade da cifragem origina o seguinte problema: Uma entidade no SGCI pode ser administrada por mais de um usuário e todos esses usuários realizam operações utilizando a chave. Portanto, é necessário uma implementação que suporte a utilização da chave por todos os usuários. A estrutura que dá suporte a esse tipo de implementação é o PKCS #7 (ver subseção 2.2.5). O PKCS #7 necessita de uma ou mais chaves públicas, que no escopo do SGCI, seriam dos usuário. Para emitir essas chaves, foi criada uma ICP interna ao SGCI que permite que cada usuário de autenticação cadastrado no sistema possuísse um certificado digital.

4.1.3 ICP interna do SGCI

A ICP interna do SGCI é composta por duas entidades internas, a AC Raiz e a AC usuários, cujo certificado foi emitido pela AC Raiz. Ambas as AC's são criadas na instalação do software e tem como único propósito dar suporte ao protocolo de uso da chave privada de uma entidade. Essas AC's são necessárias pois o protocolo prevê o uso de chave públicas no cifragem da chave da entidade. Os usuários não tem conhecimento da ICP interna e nem dos seus próprios certificados. A emissão do certificado interno de um usuário é realizada da atribuição de um papel que será explicado na próxima sessão.

4.1.4 Papéis de usuários

O SGCI conta com 3 papéis distintos de usuário. Criador, administrador e operador, sendo que os dois últimos, estão necessariamente ligados a uma Autoridade. Pode-se expandi-los para administrador de uma AR, administrador de uma AC, operador de uma AR e operador de uma AC. O usuário criador é único no sistema e sua existência não está atrelada a nenhuma uma autoridade. Ele é o usuário responsável por criar a primeira entidade no SGCI.

4.1.5 Tipos de usuários no SGCI

O SGCI conta com dois tipos de usuários. O primeiro tipo representa uma pessoa cadastrada no sistema com dados como: telefone, e-mail e nome, e não exerce função nenhuma e não se autentica. O segundo tipo de usuário é chamado de usuário de autenticação e necessariamente está vinculado a um usuário pessoa e a uma entidade por meio de um papel. Um usuário pessoa pode ter vários usuários de autenticação vinculados a diferentes entidades possibilitando que uma pessoa exerça diferentes papéis no sistema. A figura 4.1.5 ilustra esse relacionamento.

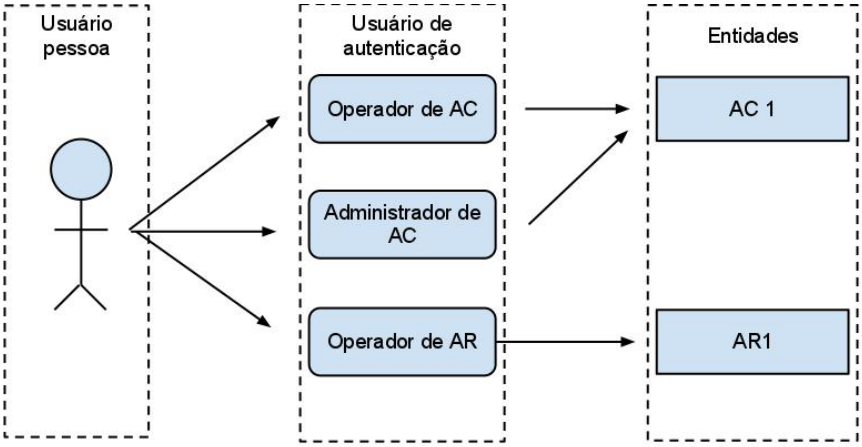

Figura 20: Tipos de usuários no SGCI

4.1.6 Cifragem da chave privada da entidade

Como dito anteriormente, para o armazenamento de uma chave privada em disco, utiliza-se uma estrutura PKCS #7 (ver 2.2.5. Esta estrutura pode armazenar conteúdos de 6 diferentes tipos: *Data*, *signed data*, *enveloped data*, *signed-and-enveloped data*, *digest data* e *encryted data*. Cada tipo de conteúdo tem um propósito específico. Para o SGCI, é necessário que vários usuários tenham acesso a chave da entidade sem ter acesso a chave dos outros usuários. O tipo que atende essa necessidade é o *enveloped data*.

O tipo *enveloped data* permite a criação de um envelope com chaves simétricas cifradas que, isoladamente decifram a chave da entidade. A criação do envelope acorre da seguinte maneira:

1. Gera-se uma chave simétrica aleatoriamente.
2. A chave simétrica é cifrada com a chave pública de um usuário.
3. A chave privada da entidade é cifrada com a chave simétrica.
4. A chave simétrica cifrada é guardada em um envelope.

Neste ponto, a chave da entidade está cifrada. Porém, não está segura. Um atacante que tivesse acesso ao banco de dados, poderia usar a chave pri-

vada de um usuário para decifrar a chave da entidade. Para resolver esse impasse, a chave privada do usuário deve estar cifrada.

4.1.7 Cifragem da chave privada do usuário no SGCI

Um usuário de autenticação é cadastro no SGCI através de um nome de usuário, a entidade que está relacionado e uma senha. A senha não é armazenada no banco de dados. Armazena-se também, uma string aleatória, chamada salt. Essa string é concatenada com a senha do usuário e em seguida, aplicada uma função resumo, obtendo-se um *digest*, utilizado para cifrar a chave privada do usuário.

Toda operação que utiliza a chave privada da entidade é considerada crítica e solicita ao usuário que digite sua senha pessoal. A seguir será explicado como decifragem da chave da entidade.

4.1.8 Uso da chave da entidade

Para o uso da chave privada da entidade para uma operação criptográfica, é necessária que esta seja decifrada. O ponto de partida da decifragem é a senha pessoal do usuário logado no sistema e é feito da seguinte maneira:

- O usuário submete sua senha para o SGCI para realizar alguma operação como por exemplo, a revogação de um certificado
- A função de revogação concatena a senha pessoal com o salt e em seguida extrai o *digest* da concatenação
- O *digest* é usado para decifrar a chave privada do usuário em questão
- Decifra-se a chave simétrica contida no envelope para aquele usuário com a chave privada.
- Decifra-se a chave privada da entidade com a chave simétrica.

4.1.9 Conclusão

Como podemos notar, a origem do uso de uma chave privada de uma entidade é a senha pessoal do usuário, garantindo que mesmo que um invasor ou um usuário mal intencionado tenha acesso ao banco de dados do SGCI,

não poderá usar a chave privada da entidade para nenhum fim pois ainda sim é necessário a senha pessoal do usuário.

5 PROCESSO DE DESENVOLVIMENTO

5.1 PROCESSO DE DESENVOLVIMENTO

A qualidade de um software está ligada a definição e padronização do processo de desenvolvimento. Para tal, existem uma série de boas práticas e técnicas que, se aplicadas ao desenvolvimento, visam garantir um produto final que seja de fácil manutenibilidade e entendimento. Estas boas práticas podem ser verificadas de maneira automatizada através de ferramentas de análise de código. Atualmente existe uma gama dessas ferramentas, que variam de linguagem para linguagem. A seguir apresentaremos as análises e métricas de código utilizadas no SGCI juntamente com algumas ferramentas.

5.1.1 Ferramentas de análise

As ferramentas de análise de código são utilizadas para avaliar o software de acordos com métricas. Elas podem estar integradas a uma plataforma de desenvolvimento (*Eclipse, Netbeans* etc), podem ser executadas via linha de comando ou de maneira automatizada, através de um servidor de integração.

5.1.2 Servidor de integração

O servidor de integração utilizado no desenvolvimento do SGCI é o Hudson 2.3.8 que é responsável por integrar todas as ferramentas de verificação de código e publicar os dados gerados por estas em forma de documentos, números e gráficos que podem ser visualizados através da sua interface web. Os dados gerados são coletados através de plugins específicos para cada tipo ou grupo de verificações. O Hudson pode ser configurado para escutar um repositório Subversion 2.3.13 ou executar uma nova verificação a cada novo commit. Ele ainda suporta vários projetos que podem ser interelacionados entre si ou não. Possui controle de acesso através de autenticação e permissões diferenciadas para cada usuário ou grupo de usuários.

Porém, o servidor de integração não faz todo o trabalho sozinho, sua integração com os aplicativos de análise bem como suas configurações são feitas através de uma terceira ferramenta, no SGCI o *Phing*2.3.12. Através do arquivo de configuração desta, cria-se alvos que farão o trabalho de chamar

as ferramentas de análise que publicam seus resultados em diretórios específicos. A figura a seguir mostra toda a infraestrutura.

5.1.3 Execução automatizada após um commit

Como dito anteriormente, o servidor de integração pode ser configurado para disparar uma construção do projeto, ou seja, executar todas as tarefas que foram programadas para executar, após um commit em um repositório Subversion. O fluxo completo ocorre da seguinte maneira.

1. Desenvolvedor envia uma alteração para o repositório Subversion;
2. Repositório Subversion envia uma notificação para o Hudson avisando de um novo commit;
3. Hudson atualiza sua cópia de trabalho;
4. Hudson dispara as tarefas programadas chamando o Phing;
5. Phing chama as ferramentas de análise, teste e geração;
6. Ferramentas publicam seus resultados em um diretório específico;
7. Hudson publica busca resultados nos diretórios do passo anterior e através de plugins gera gráficos e documentos.

A figura a seguir ilustra o processo demonstrado anteriormente.

5.1.4 Código duplicado

Um dos maiores vilões no desenvolvimento de software é o código duplicado. A presença deste tipo de problema dificulta a correção a de bugs e implementação de novas funcionalidades. Uma aplicação com muito código duplicado pode vir a ter sérios problemas de manutenibilidade, podendo até ter seu desenvolvimento descontinuado. Felizmente existem ferramentas que fazem uma varredura no código e detectam códigos duplicados. No SGCI a ferramenta utilizada é a *PHP Code and Paste Detector*2.3.16.

5.1.5 Complexidade ciclomática

A complexidade é uma métrica de software, que visa medir a complexidade de um trecho de código ou um software. Ela se baseia nos caminhos

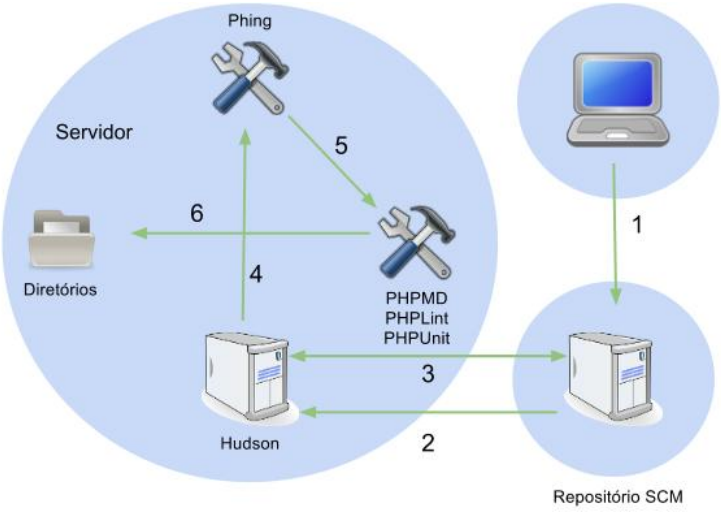

Figura 21: Processo de desenvolvimento

possíveis no código de acordo com as decisões ou estruturas de controle existentes. Pode-se afirmar que um código com complexidade ciclomática igual a um, não possui nenhuma decisão de controle, em outras palavras, existe apenas um caminho possível para aquele código. Para a medição da complexidade ciclomática no SGCI, é usado software de apoio, cujo nome é *PHP Mess Detector*2.3.17 e de acordo com a documentação deste, a gravidade das complexidades são dadas da seguinte maneira:

- Complexidade de 1 a 4: Baixa
- Complexidade de 5 a 7: Moderada
- Complexidade de 8 a 10: Alta
- Complexidade maior que 11: Muito alta

5.1.6 Boas práticas de codificação

Aqui serão apresentadas algumas boas práticas de codificação utilizadas no SGCI, que são de fácil entendimento e não precisam estar separadas em varias subseções. Geralmente o seu nome já é autoexplicativo, não

necessitando detalhamento. Todas elas são analisadas utilizando, novamente, a ferramenta *PHP Mess Detector*.

As seguintes regras são relativas ao tamanho de código de classes, métodos etc.

- Comprimento excessivo de métodos: A violação desta regra geralmente acusa que um método está fazendo funções demais e pode ser quebrado em mais métodos.

- Comprimento excessivo de classe: Classes com comprimento excessivo denunciam uma baixa coesão, devendo ser repensada e dividida em mais classes.

- Número excessivo de métodos: Uma lista longa de parâmetros indica que uma nova classe deve ser criada para conter esses parâmetros.

- Número excessivo de métodos públicos: Uma classe com muitos métodos públicos indica, novamente, baixa coesão, devendo ser quebrada em classes menores.

- Classes com muitos atributos: Uma classe com muitos atributos pode derivar novas classes com menos atributos.

- Classes com muitos métodos: Classes com número excessivo de métodos indicam erro de modelagem e podem dar origem a novas classes

Tem-se ainda regras relativas a nomenclatura, a maioria destas são abordadas pelo conceito de *Clean Code*, visando um código limpo e de fácil entendimento.

- Nomes de variáveis: Avalia o tamanho do nome de uma variável. Esta deve ter um nome autoexplicativo, nomes curtos de variáveis não remetem ao conteúdo que está sendo guardado, porém, por questão de estética, também não pode ser muito longo.

- Nome curtos para métodos: O nome de um método deve remeter a sua função. Essa regra detecta quando esse tipo de regra é violada.

- Nome do construtor igual ao nome da class: Um método construtor não pode ter o mesmo nome da classe.

- Método get com retorno booleano: Um método cujo nome tem o prefixo get não deve retornar um tipo booleano. Para tal, o nome do método deve ter o prefixo *is* ou *has*.

Trechos de códigos não utilizados não devem ser deixados no aplicação. Para tal, existem algumas regras rastreamento desse tipo de problema.

- Campo privado não utilizado: Detecta campos privados declarados e/ou atribuídos porém não são utilizados.
- Variável local não utilizada: Detecta variáveis locais que foram criadas e atribuídas ou não um valor, mas não são usadas.
- Métodos privados não utilizados: Acusa quando se tem um método privado não utilizado.
- Parâmetros não utilizados: Acusa quando se tem parâmetros que foram passados para métodos, porém não foram utilizados.

5.1.7 Outras funcionalidades implantadas

Além das funcionalidades e verificações mostradas anteriormente, implementou-se algumas outras funcionalidades de apoio ao desenvolvimento que serão mostradas a seguir.

5.1.7.1 Cobertura de código

É possível através da ferramenta de testes automatizados, *PHPUnit*2.3.14 gerar uma documentação em HTML que mostra a cobertura dos testes em relação ao código fonte. Este documento tem detalhes específicos tais como: Quantas vezes os testes passaram em cada linha do código, quais trechos de código não foram testados e porcentagem de cobertura da classe pelos testes. A figura abaixo foi extraída de uma verificação de cobertura do SGCI.

5.1.7.2 Documentação

No SGCI todo código deve estar documentado. Para garantir que isso seja respeitado, existem também uma verificação que notifica o desenvolvedor quando este implementa um código e não o documenta. Além disso, tem-se ainda a geração automática da documentação através da integração da ferramenta *PHPDocumentor*2.3.15 no servidor de integração. A última versão da documentação está sempre disponível no Hudson para acesso.

5.1.7.3 Automatização da geração de pacotes

O Servidor de integração Hudson (2.3.8) suporta a execução de scripts *shell*. Como visto na seção 3.7.3, a geração dos pacotes é feita de forma automatizada através da execução de um script em shell. Este script foi incorporado no servidor de integração para gerar um novo pacote .deb e publicá-lo automaticamente após cada commit.

6 CONSIDERAÇÕES FINAIS

6.1 CONSIDERAÇÕES FINAIS

O estudo e a implementação de um instalador contribui para facilitar a instalação do SGCI. Com o uso dele o processo de instalação se tornou trivial. A infraestrutura criada para gerar e disponibilizar os pacotes facilitou a geração de versões do SGCI.

Com as melhorias de usabilidade implementadas, o SGCI ficou com um visual mais limpo e com uma interface mais amigável para o usuário. Todos os objetivos propostos foram atingidos nesse quesito.

A área administrativa auxilia nas configurações gerais do software através de uma interface gráfica e evita que o administrador tenha que se conectar na máquina para fazer modificações.

O protocolo de ciframento é seguro e pode-se afirmar que a chave da entidade está segura no banco de dados. Além de ser versátil, possibilitando que diferentes usuários façam uso da mesma.

7 ANEXO

8 APÊNDICE

REFERÊNCIAS BIBLIOGRÁFICAS

YVES Savourel.

APACHE. *The Number One HTTP Server On The Internet*. September 2011. <http://httpd.apache.org/>.

CI, H. *What is Hudson?* 2012. <http://wiki.eclipse.org/Hudson-ci/>.

COOPER, D. et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF, maio 2008. RFC 5280 (Proposed Standard). (Request for Comments, 5280). <http://www.ietf.org/rfc/rfc5280.txt>.

DETECTOR, P. M. *PHP Mess Detector*. 2012. <http://phpmd.org/>.

DIERKS, T.; RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF, ago. 2008. RFC 5246 (Proposed Standard). (Request for Comments, 5246). Updated by RFCs 5746, 5878, 6176. <http://www.ietf.org/rfc/rfc5246.txt>.

FRAMEWORK, Z. *About Zend Framework*. 2011. <http://framework.zend.com/about/overview>.

HOUSLEY, R.; POLK, T. *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. 1st. ed. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 0471397024.

IGNACZAK, L. *Um Novo Modelo de Infra-estrutura de Chaves Publicas para Uso no Brasil Utilizando Aplicativos com o Código Fonte Aberto*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2002.

LIBCRYPTOSEC. *Projeto Libcryptosec*. 2011. <https://projetos.labsec.ufsc.br/libcryptosec/>.

MARTINA, J. E. *Projeto de um provedor de serviços criptográficos embarcado para infra-estrutura de chaves públicas e suas aplicações*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2005.

NETCRAFT. *September 2011 Web Server Survey*. September 2011. <http://news.netcraft.com/archives/2011/09/06/september-2011-web-server-survey.html>.

OPENSSL. *The OpenSSL Project*. 2011. <http://www.openssl.org/>.

PAAR, C.; PELZL, J. *Understanding Cryptography: A Textbook for Students and Practitioners*. [S.l.]: Springer-Verlag New York Inc, 2010.

PHING. *Phing Wiki?* 2012. <http://www.phing.info/trac/>.

PHP. *What is PHP?* 2011. <http://php.net/>.

PHP5LIBCRYPTOSEC. *Projeto PHP5 Libcryptosec*. 2011. <https://projetos.labsec.ufsc.br/php5-libcryptosec>.

PHPCPD. *PHPCPD*. 2012. <https://github.com/sebastianbergmann/phpcpd>.

PHPDOCUMENTOR. *What is phpDocumentor 2?* 2012. <http://www.phpdoc.org/>.

PHPUNIT. *PHPUnit Readme*. 2012. <https://github.com/sebastianbergmann/phpunit/>.

SAVOUREL, Y. *XML Internationalization and Localization*. [S.l.: s.n.], 2001. xi + 519 p. ISBN 0-672-32096-7.

SINGH, S. *O livro dos códigos*. 4. ed. [S.l.: s.n.], 2004.

SQLITE. *About SQLite*. 2011. <http://www.sqlite.org/about.html>.

SQLITE. *SQLite Is A Zero-Configuration Database*. 2011. <http://www.sqlite.org/zeroconf.html>.

SQLITE. *SQLite Is Self-Contained*. 2011. <http://www.sqlite.org/selfcontained.html>.

SQLITE. *SQLite Is Serverless*. 2011. <http://www.sqlite.org/serverless.html>.