Sistema de Gerência Centralizada de Fechaduras Microcontroladas

Tatiane Martins Pacheco, Marco Antônio Pereira Araújo

Núcleo de Informática – Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais – Campus Juiz de Fora (IF Sudeste MG)
Rua Bernardo Mascarenhas, 1283 - Fábrica – 36.080-001 – Juiz de Fora – MG – Brazil

tatiiane.martiins@gmail.com, marco.araujo@ifsudestemg.edu.br

Abstract. *This paper deals with the specification and implementation of a microcontrolled locks management system , which aims to restrict and control access to certain sectors of an organization. This system is based on network technology , Web service , Internet and Internet of Things, applying security techniques, monitoring and encryption as well as software and design patterns development techniques. The work was implemented at IF Southeast MG, integrated into the work of the student Yuri Marx Guedes , which dealt specifically with the embedded system of the lock control, not neglecting the current technologies of locks currently adopted in the institution.*

Resumo. *Este trabalho trata da especificação e implementação de um sistema de gerenciamento de fechaduras microcontroladas, que visa restringir e controlar o acesso a determinados setores de uma organização. Esse sistema é baseado em tecnologias de Rede, Web Service, Internet e Internet das Coisas, aplicando técnias de segurança, monitoramento e criptografia, além de técnicas de desenvolvimento de software e de padrões de projeto. O trabalho foi implementado no IF Sudeste MG, integrado ao trabalho do aluno Yuri Marx Guedes, que tratou especificamente do sistema embarcado de controle da fechadura, não desprezando as tecnologias atuais de fechaduras atualmente adotadas na instituição.*

1. Introdução

1.1. Contextualização

A insegurança é um problema social muito discutido, de forma que a garantia da segurança das informações é bastante complexa. O acesso às informações de determinado setor organizacional, que deve ser restrito, é um dos grandes desafios dos sistemas informatizados, especialmente quando não se refere só ao acesso em um computador, por exemplo, mas em um setor tal que contenha informações sigilosas em seu domínio. É essencial garantir a segurança de pessoas e informações nas instalações, reduzindo a possibilidade de danos. Neste sentido, saber quem entra e sai dos ambientes de uma organização, bem como acompanhar esse fluxo, constitui uma poderosa alternativa de segurança. Em geral, o acesso a setores nessas organizações possui mecanismos pouco funcionais de controle, que incluem fechaduras tradicionais ou a presença de “porteiros”, por exemplo,

Dentre as várias possibilidades trazidas pelos sistemas de informação, a utilização de sistemas integrados de segurança e controle de acesso é de bastante interessante, pois implementam procedimentos de limitação de acesso e garantem características fundamentais da segurança da informação, como a confidencialidade e a integridade. Confidencialidade é a garantia de que a informação seja acessível apenas às pessoas autorizadas [Marinho 2004]. Integridade, por sua vez, significa garantir que a informação armazenada ou transferida está correta e é apresentada corretamente para quem a consulta [Benetti 2015]. Além disso, pode prover estatísticas como: fluxo em determinados setores, utilização por determinadas pessoas e outras, que possuem inúmeras aplicações nos contextos organizacionais. Com isso, é reduzida a possibilidade de acessos não autorizado, possibilitados por falha humana e mesmo falha nas trancas de portas tradicionais (com o uso de uma chave mestra, por exemplo). No caso de sistemas informatizados, várias são as tecnologias que auxiliam no aspecto de *hardware* da fechadura eletrônica, desde fechaduras comuns adaptadas a fechaduras com leitor biométrico, ou mesmo a utilização de dispositivos de leitura RFID (Radio-Frequency IDentification).

Para isso, um sistema desse porte deve utilizar mecanismos de autenticação de usuário. Dentre as formas mais tradicionais, estão o par login-senhas, cartões magnéticos e leitura biométrica.

A implantação de uma solução de controle de acesso provê maior segurança aos ambientes em que estão instalados, restringindo o acesso as instalações por pessoas mal-intencionadas. Dentre as atribuições de um software para controle se destacam a autenticação que identifica “quem”; a autorização, que diz “onde” e; a auditoria, que diz “quem fez o que, onde e quando”.

1.2. Visão Geral

O objetivo deste trabalho é a implementação do Sistema de Gerência Centralizada de Fechaduras Microcontroladas (ServidorMGC), cuja principal responsabilidade é o gerenciamento de fechaduras eletrônicas produzidas pelo aluno Yuri Marx Guedes em seu trabalho de conclusão de curso, sob a orientação do professor Marcelo C. P. Santos. Faz-se necessário delimitar as responsabilidades designadas a cada aluno, deixando claro que o trabalho do Yuri abrange a concepção da “fechadura” e tudo que envolve o sistema embarcado, enquanto este trabalho, compreende o desenvolvimento do SistemaWeb responsável por gerenciar esses dispositivos, promovendo toda a comunicação entre eles, disponibilizando as funcionalidades para cadastro e manutenção de usuários, ambientes, grupos de acesso, níveis de acesso, fechaduras e auditoria através das consultas ao *log*.

Embora estejam relacionados, esses trabalhos são independentes: o sistema controla a fechadura e depende das informações coletadas e enviadas por ela para então realizar o processamento dos dados e aplicar as regras de negócio estipuladas. Já a fechadura depende do sistema para funcionar e deixa toda a complexidade deste processamento de informações como responsabilidade do ServidorMGC. Essa transferência de responsabilidades é particularmente interessante, pois diminui o impacto gerado por esse processamento, que pode ser incompatível com o hardware limitado de um sistema embarcado, facilitando mesmo processos de atualização e

reparo. A figura abaixo representa todos os elementos envolvidos no funcionamento do sistema e na comunicação com a fechadura.

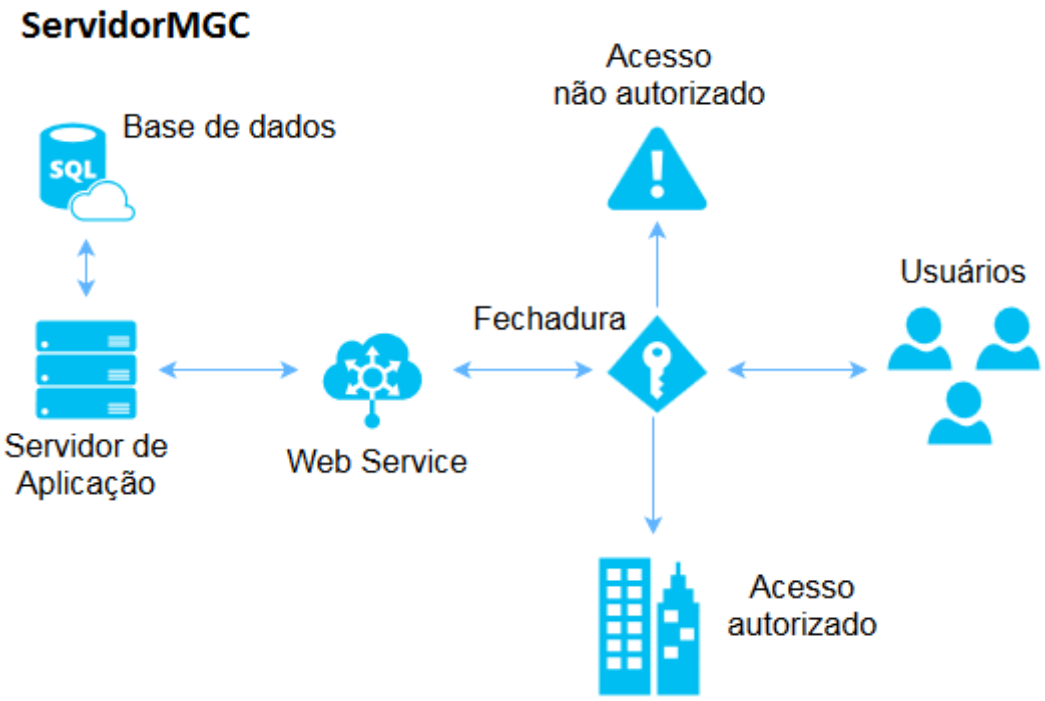

Figura 1. Visão geral do ServidorMGC e da comunicação com a fechadura.

O sistema proposto provê serviços à fechadura, permitindo facilitar a gerência das fechaduras instaladas nos ambientes assim como as permissões de acesso de usuários permitindo um controle total, simplificado e centralizado dos elementos envolvidos nesse processo.

Essa aplicação foi desenvolvida no intuito de fornecer um sistema de controle de acesso para o IF Sudeste MG, mas a sua abordagem genérica não impede que seja utilizado para qualquer ambiente em qualquer organização.

2. Revisão Bibliográfica

É importante diferenciar internet e World Wide Web (Web) para entender o funcionamento do ServidorMGC. A Internet é a camada ou rede física composta por switches, roteadores e outros equipamentos [Evans 2011]. Sua função primária é transportar informações de um ponto a outro de forma rápida, confiável e segura. A Web é uma camada de aplicativos que opera sobre a Internet Sua função primária é oferecer uma interface que transforme as informações que fluem pela Internet em algo utilizável [Evans 2011].

O ServidorMGC é uma aplicação Web ao qual se aplica o conceito de internet das coisas, pois recebe as informações advindas da fechadura que é um microcontrolador ou a “coisa” responsável por coletar os dados e transmitir para o Sistema Web que realiza um processamento tornando inteligente este processo.

A Internet das coisas é um conceito no qual dispositivos do dia a dia são equipados com sensores capazes de captar aspectos do mundo real (*Internet Of Things* –

IoT) [Nascimento 2015]. IoT é composta por 4 conceitos básicos: A "coisa" precisa ser algo físico que capture as informações; a "conectividade", a existência de um meio responsável por efetuar a comunicação em si; os "dados", que foram coletados pelas "coisas" e transmitidos, e; as "Análises", processos de transformação dos dados em informação útil [Luz e Humenhuk 2015].

2.1.1. Microcontrolador

É possível se definir o microcontrolador como sendo um pequeno componente eletrônico, que possui uma "inteligência" que pode ser programável [Souza 2000]. É um sistema computacional completo no qual estão incluídos uma CPU (Central Processor Unit), memória de dados e programa, um sistema de clock, portas de I/O (Input/Output), além de outros possíveis periféricos, tais como, módulos de temporização e conversores A/D entre outros, integrados em um mesmo componente [Denardim 2015].

Os sistemas embarcados, como o que será usado na fechadura eletrônica, estão relacionados ao uso de hardware (eletrônica) e software (instruções) incorporados em um dispositivo com um objetivo pré-definido [Delai 2013]. Um sistema embarcado é projetado para executar funções dedicadas [Bastos 2015].

2.1.2. XML

XML (*Extensible Markup Language*) é o formato universal para partilha de dados entre aplicações [Heitlinger 2001].Um dos objetivos por trás dessa linguagem é possibilitar a transferência e manipulação de dados através da Internet de modo fácil e consistente, de tal forma que qualquer tipo de aplicação, independentemente da plataforma, sistema operacional, ou linguagem em que foi construída consiga manuseá-los [Pereira 2011].

Dessa forma por "linguagem de marcação", entende-se um conjunto de convenções utilizadas para a codificação de textos. Uma linguagem de marcação deve especificar que marcas são permitidas, quais são exigidas, como se deve fazer distinção entre as marcas e o texto e qual o significado da marcação [Almeida 2002].

2.1.3. *Web Service*

Web Service é uma solução para interação entre diferentes sistemas possibilitando a troca de dados entre aplicações diferentes. Cada aplicação pode ter a sua própria "linguagem", e os dados recebidos e enviados são trocados em formatos variados como, por exemplo, JSON, XML e até texto puro.

Web Services fornecem um meio padrão de interoperabilidade entre diferentes aplicações de software, rodando em uma variedade de plataformas e / ou frameworks. [*W3C a* 2004].

Conceitos importantes:

O WSDL (*Web Services Description Language*) é uma linguagem que funciona como um descritor de *Web Service*s responsável por dizer quais os métodos existentes, tipo de dados recebidos, tipo de dados retornados etc. Através destas especificações que uma aplicação que deseja consumir dados de um *Web Service* toma conhecimento das informações necessárias para o cliente invocar um serviço.

SOAP (*Simple Object Access Protocol*) é um protocolo projetado para invocar aplicações remotas através de RPC(*Remote Procedure Call)* ou trocas de mensagens, em um ambiente independente de plataforma e linguagem de programação. SOAP é, portanto, um padrão normalmente aceito para utilizar-se com *Web Service*s. Desta forma, pretende-se garantir a interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem (XML) e mecanismo de transporte (*HTTP- Hypertext Transfer Protocol)* padrões [Dantas 2007].

UDDI (*Universal Description, Discovery and Integration protocol*) cria uma plataforma padrão que possibilita que companhias e aplicações encontrem e usem *Web Service*s de forma rápida, fácil e dinâmica [Rolim 2005].

Um *Web Service* é um aplicativo projetado para suportar interoperabilidade entre máquinas através de uma rede. O *Web Service* utilizado neste trabalho possui uma interface apresentada em WSDL (*Web Service*s Description Language), que descreve como outros sistemas podem interagir com ele através da utilização de mensagens SOAP (Simple Object Access Protocol). Essas mensagens são tipicamente transportadas usando HTTP (Hypertext Transfer Protocol) com uma formatação XML (Extensible Markup Language) e em conjunto com outras convenções relacionadas à Web [*W3C b* 2004].

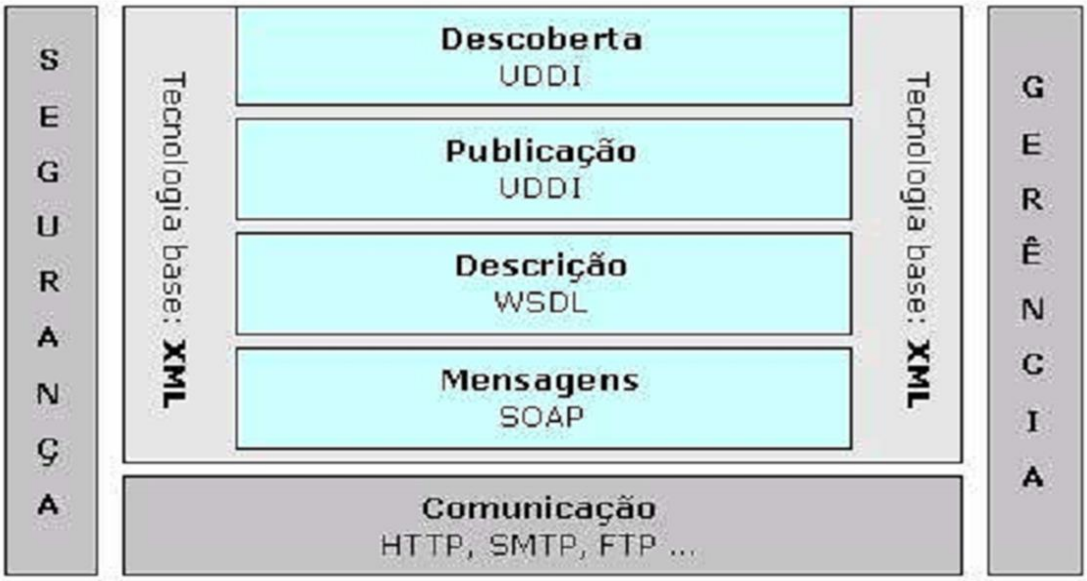

Figura 2. Arquitetura de Web service [Rolim **2005].**

2.1.4. Java e orientação a objetos

Java é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995 [ORACLE 2015]. A tecnologia Java é usada para desenvolver aplicativos para uma ampla variedade de ambientes, de dispositivos para o consumidor a sistemas corporativos heterogêneos. O paradigma de programação da linguagem Java e baseado no conceito de programação orientada a objetos (OOP) [Perry 2011].

A Orientação a Objetos modela o mundo real com classes e instâncias. Cada classe é a estrutura de uma variável, ou seja, um tipo de dado. Nela, são declarados atributos e

métodos que poderão ser executados ou acessados nas instâncias da mesma classe. As classes possuem uma função muito importante na modelagem orientada a objetos, elas dividem o problema, modularizam a aplicação e baixam o nível de acoplamento do software. Um Objeto é uma entidade que carrega informações das suas características até suas ações, ou seja, uma abstração de um domínio de um problema [Pamplona 2012].

2.1.5. Padrões de projeto

Os padrões de projeto solucionam muitos dos problemas que os projetistas enfrentam no dia-a-dia, e de muitas maneiras diferentes [Gamma 2005]. Os padrões de projeto permitem a reutilização de soluções encontradas que comprovadamente foram úteis na solução de questões análogas. Na implementação deste TCC foram utilizados alguns padrões buscando uma codificação limpa seguindo as boas práticas de programação. Entre eles:

Padrão *Model-View-Controller*: MVC é um padrão de arquitetura de *software* [Belem 2013] que separa a informação (e as suas regras de negócio) da interface com a qual o usuário interage. É uma forma de estruturar seu projeto/aplicação de forma que a interface de interação *(view*) esteja separada do controle da informação em si (*models*), separação essa que é intermediada por uma outra camada controladora *(controllers*) *software* [Belem 2013].

Padrão *Front Controller*: Trata-se de uma maneira de centralizar os acessos em um único ponto e então, a partir deste, rotear para uma ação (ou comando) [Marques 2015].

Padrão *Action* ou *Command*: Encapsular uma solicitação como objeto, desta forma permitindo parametrizar cliente com diferentes solicitações, enfileirar ou fazer o registro de solicitações e suportar operações que podem ser desfeitas. [Gamma 2005].

Padrão *DAO*: é capaz de isolar todo o acesso a banco em classes bem simples, cuja instância é um objeto responsável por acessar os dados. Da responsabilidade deste objeto surgiu o nome de *Data Access Object* ou simplesmente *DAO*, um dos mais famosos padrões de projeto.

2.1.6. *MySQL*

MySQL, o mais popular sistema de gerenciamento de banco de dados. *SQL Open Source*, é desenvolvido, distribuído e tem suporte da *MySQLAB*. O *MySQL* é um sistema de gerenciamento de bancos de dados relacional. Um banco de dados relacional armazena dados em tabelas separadas em vez de colocar todos os dados em um só local. Isso proporciona velocidade e flexibilidade [MySQL manual].

2.1.7. Criptografia

A criptografia é uma técnica utilizada há anos que com o passar do tempo evoluiu a ponto de oferecer soluções eficazes no que diz respeito à segurança da informação. Hoje, ela é uma ferramenta de segurança amplamente utilizada nos meios de comunicação e consiste basicamente na transformação de determinado dado ou informação a fim de ocultar seu real significado [Sampaio 2012]. A criptografia segue quatro princípios básicos: confidencialidade, autenticação, integridade da informação e

não repudiabilidade (ou seja, o remetente não pode negar o envio da informação) [Santiago 2012]. Criptografar é a ação de modificar uma mensagem para ocultar o seu real significado. Descriptografar é o processo inverso que envolve pegar um texto criptografado e transforma-lo em um texto normal.

O RC4 foi a criptografia escolhida para utilização neste trabalho pois é uma técnica de fluxo o que permite criptografar/descriptografar qualquer tamanho de mensagem, uma vantagem sobre os algoritmos que usam a técnica de blocos onde há a necessidade de tamanho fixo das mensagens para o processo de cifragem/decifragem funcionar. O RC4 é uma abordagem muito usada, por funcionar em fluxo continuo e também por ser bastante rápida.

O *RC4* consiste em utilizar um *array* que a cada utilização tem os seus valores permutados, e misturados com a chave, o que provoca uma dependência muito forte com esta chave. Esta chave, utilizada na inicialização do *array*, pode ter até 256 bytes (2048 bits). Entretanto o algoritmo é mais eficiente quando temos uma chave menor devido a superior perturbação aleatória induzida no *array. É* um algoritmo que promove transformações lineares não sendo necessários cálculos complexos, visto que o sistema funciona basicamente por meio de permutações e somas de valores inteiros, o que o torna muito simples e rápido [Medeiros a, 2012].

2.1.8. JSP

JSP é o acrônimo para *Java Server Pages*, uma linguagem criada pela SUN. JSP é uma linguagem de *script* com especificação aberta que tem como objetivo primário a geração de conteúdo dinâmico para páginas da Internet. Podemos ao invés de utilizar HTML (*HyperText Markup Languag)* para desenvolver páginas *Web* estáticas e sem funcionalidade, utilizar o JSP para criar dinamismo. É possível escrever HTML com códigos JSP embutidos. Como o HTML é uma linguagem estática, o JSP será o responsável por criar dinamismo [Rocha 2014].

Em JSP geramos arquivos com extensão .jsp que incluem, dentro da estrutura de etiquetas HTML, as sentenças Java a executar no servidor. Antes que os arquivos sejam funcionais, o motor JSP realiza uma fase de tradução dessa página em um *servlet*, implementado em um arquivo class (*Byte codes* de Java). Esta fase de tradução se realiza habitualmente quando se recebe a primeira solicitação da página .jsp, embora exista a opção de pré-compilar em código para evitar esse tempo de espera na primeira vez que um cliente solicita a página[Alvarez 2014]

3. Materiais e Metodologia

O protótipo implementado é uma aplicação *Web*, que possui uma interface com o usuário a fim de permitir a alimentação do sistema pelos administradores. O sistema ainda oferece uma camada de interface, com a fechadura, para prover serviços necessários para o funcionamento da mesma, através de um *Web Service.* A persistência dos dados é garantida através do *Mysql*.

3.1. Funcionamento geral do Sistema

O ServidorMGC foi desenvolvido em linguagem JAVA que é uma linguagem robusta, orientada a objetos e muito poderosa. A orientação a objetos nos permite modelar

problemas reais. Interpretando estes problemas através de classes e objetos. Nesse contexto surgiu à ideia da implementação do sistema. Para explicar seu funcionamento, por completo a figura a seguir retrata o diagrama de caso de uso do sistema.

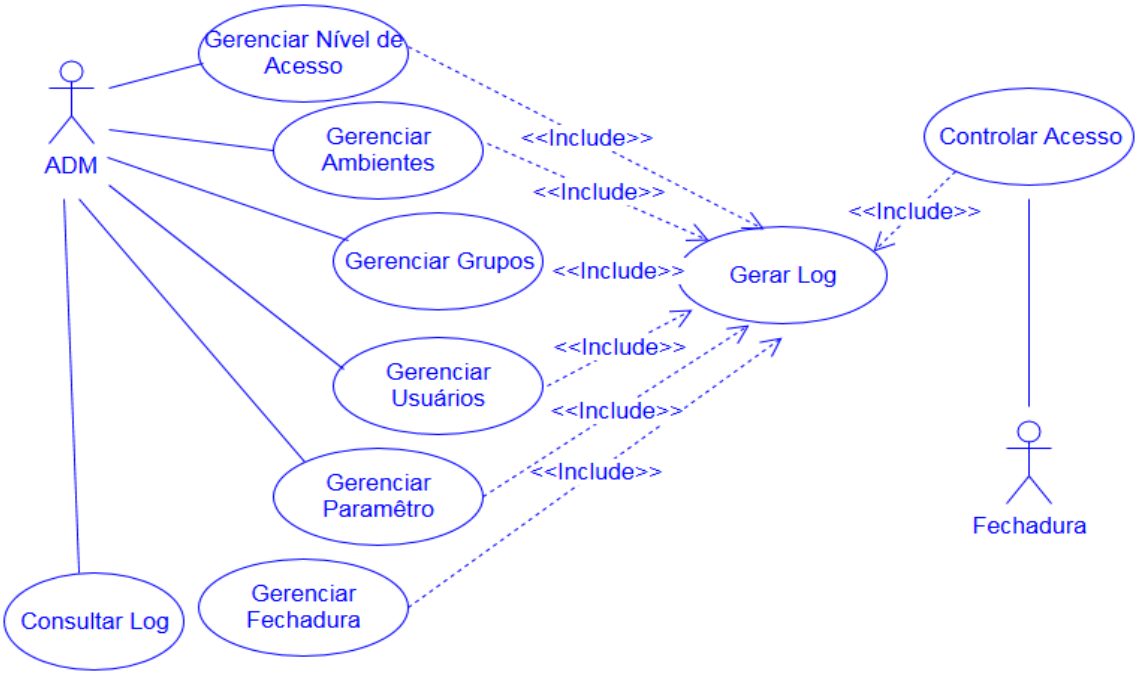

Figura 3. Diagrama de caso de Uso do ServidorMGC.

Para melhor ilustrar e facilitar o entendimento segue a tela principal do sistema onde é possível acessar todas as funcionalidades.

Figura 4. Tela principal ServidorMGC.

Inicialmente o administrador (ADM) precisa logar no sistema, para isso deve digitar seu *login* e senha, após a validação o mesmo será direcionado para o menu principal onde pode acessar todas as opções gerenciais citadas no caso de uso. Vale ressaltar que os casos de uso "Gerenciar" são uma forma simplificada de representar o *CRUD*. *CRUD* é o acrônimo da expressão do idioma Inglês, *Create* (Criação), *Retrieve* (Consulta), *Update* (Atualização) e *Delete* (Destruição). Este acrônimo é comumente utilizado definir as quatro operações básicas usadas em Banco de Dados Relacionais [Araújo 2010].

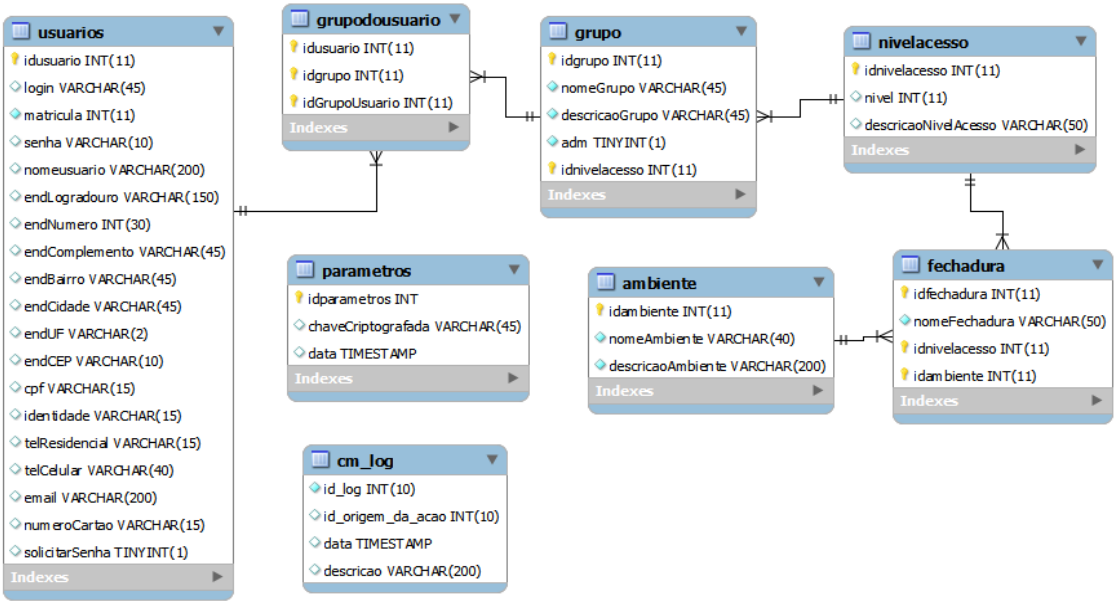

Figura 5. Diagrama entidade relacionamento ServidorMGC.

Através deste menu o ADM pode incluir, editar, excluir, listar e consultar *Log* do sistema. É possível realizar estas operações com usuários, ambientes, grupos, nível de acesso, fechaduras e parâmetros. Explicando detalhadamente um usuário é uma pessoa que possui um cadastro no sistema que é utilizado para autorizar ou não a abertura de uma fechadura.

Um grupo foi criado para facilitar a gerência dos usuários sem que se tivesse a necessidade de gerenciar as permissões individualmente, ou seja, cada grupo é um conjunto de usuários. Nível de acesso e uma classificação dada ao grupo e a fechadura a fim de limitar acessos através deste parâmetro.

Uma fechadura é o sistema embarcado responsável por interagir com o usuário, captando os dados fornecidos e transmitindo para o ServidorMGC. Ambiente pode ser uma sala, um prédio. Um grupo possui apenas um nível de acesso, mas um usuário pode estar em vários grupos. Por exemplo, um aluno pode ser do grupo alunos, mas também pode ser do grupo bolsistas e sendo assim possuir as permissões pertinentes aos dois grupos. Nível de acesso torna possível que um ambiente possua várias fechaduras com níveis de acesso diferentes, por exemplo, existe um ambiente X nesse ambiente há laboratórios de informática onde os usuários do grupo alunos precisam acessar, entretanto no mesmo ambiente existe a sala dos professores. E não é desejável que um aluno tenha permissão para aceder a este local cuja entrada somente é permitida aos usuários do grupo professores, portanto basta que o nível de acesso da sala dos

professores, seja diferente do nível de acesso do grupo de alunos. E assim podemos ter em um ambiente várias fechaduras, com níveis de acesso diferentes e gerenciar tudo isto mais facilmente através de grupos.

Cada ação efetuada no sistema chama o caso de uso GerarLOG que é responsável por gravar todas as modificações ou acessos efetuados. O ADM possui acesso a estes dados através do caso de uso ConsultarLOG. O Gerenciar Parâmetros permite alterar alguns valores do sistema utilizados para configuração inicial da fechadura tais como lista de acesso *off-line*, tempo entre a troca da chave de criptografia, data e hora do servidor.

O caso de uso que representa a comunicação da fechadura com o ServidorMGC chama-se “Controlar Acesso” que ocorre quando a fechadura faz uma requisição ao *WebService*. O Web Service de comunicação entre fechadura e servidor possui disponíveis quatro métodos são eles: AutorizaAberturaCartão() onde a fechadura envia deu ID e o número do cartão RFID lido e como retorno recebe a autorização ou não da entrada no ambiente. O AutorizaAberturaCartãoSenha() onde a fechadura envia deu ID e o número do cartão RFID e a senha do usuário que foi digitada e como retorno recebe a autorização ou não. AutorizaAberturaMatricula() onde usuário digita sua matricula e senha e a fechadura envia seu ID, matricula e senha do usuário e recebe o retorno, para autorizar ou não a abertura da porta. E por fim o método ConfiguracaoInicial() que está configurado para ser solicitado pela fechadura na primeira comunicação com o servidor neste método são enviados todos os parâmetros iniciais necessários para o correto funcionamento das fechaduras.

Das responsabilidades deste método estão: enviar data e hora do servidor e da próxima comunicação agendada entre servidor e fechadura. Lista com nomes e senhas de usuários do grupo ADM para acesso offline em caso de falta de comunicação entre servidor e fechadura facilitando assim a manutenção da fechadura. No retorno deste método também é enviado o tipo de acesso daquela fechadura se ela autoriza as aberturas utilizando o par matricula e senha, cartão e senha ou apenas cartão isso e determinado na classe parâmetros.

Toda a comunicação é criptografada, portanto a requisição da fechadura chega criptografada no *WebService* que faz a descriptografia, identifica o método solicitado e efetua a operação. Antes de enviar o envelope soap de retorno os dados são criptografados novamente. Na comunicação entre fechadura e ServidorMGC a fechadura sempre envia seu ID que é seu código identificador, caso o sistema não encontre aquele ID no banco, entende que esta fechadura ainda não está cadastrada e retorna uma mensagem de alerta caso contrario efetua o procedimento e como retorno lhe envia os dados solicitados.

3.2. Especificações de segurança e criptografia

Além do controle das fechaduras em si, a segurança da informação abarca o tráfego dos dados no sistema, uma vez que deve-se presumir que, na pior das hipóteses, alguém pode tentar burlar o acesso ao sistema caputando informações desprotegidas nesse tráfego.

Assim, uma política de segurança adequada deve considerar o uso de criptografia. Para esse sistema, buscou-se a utilização de uma criptografia reversível,

criptografada na emissão e descriptografada no destinatário. Na primeira comunicação com o servidor, a fechadura deve solicitar a configuração inicial, para isso, envia seu ID criptografado com uma chave pré-definida, o Web Service recebe esta requisição, realiza a descriptografia, efetua o processamento e envia como resposta, os parâmetros iniciais que são : uma lista de pessoas que possuem permissão para acessar a fechadura localmente no caso de falta de comunicação com o servidor. Permitindo assim um funcionamento mínimo perante uma adversidade. Data e hora do servidor para sincronização de tarefas agendadas. Tempo em HH:MM:SS para a próxima atualização, onde o ServidorMGC envia essa configuração inicial novamente para atualizar as fechaduras caso haja alguma alteração.

O algoritmo de criptografia escolhido é do tipo simétrico, define uma única chave para cifrar e decifrar uma mensagem.Cifras de fluxo processam as mensagens bit a bit ou byte a byte [Stallings 2008]. Neste trabalho será utilizada cifragem de fluxo onde não é necessário um tamanho fixo para as mensagens. O algoritmo escolhido para a criptografia é o RC4.

Outro campo para segurança dos dados é o banco de dados, em que dados como cpf, identidade, cartões de crédito também devem estar seguros. O ideal é que informações importantes sejam guardadas criptografadas unidirecionalmente. Desta forma, será utilizado o algoritmo SHA256. Assim, quando o usuário informar os dados considerados críticos, estes serão criptografados aplicando-se o algoritmo SHA256 e armazenando-se o *Hash* gerado ao invés da informação em si. Na necessidade de verificar se o dado digitado está idêntico ao armazenado no banco aplica-se novamente o algoritmo SHA256 e confere-se o *hash* gerado com o que se encontra armazenado no banco. Se os valores forem iguais, significa que a dado informado está correto.

3.3. Especificações de desenvolvimento

Para facilitar o entendimento da estuturação do projeto e como ocorre a interação entre as classes, abaixo estão as imagens da estrutura do projeto e o diagrama de classes do sistema.

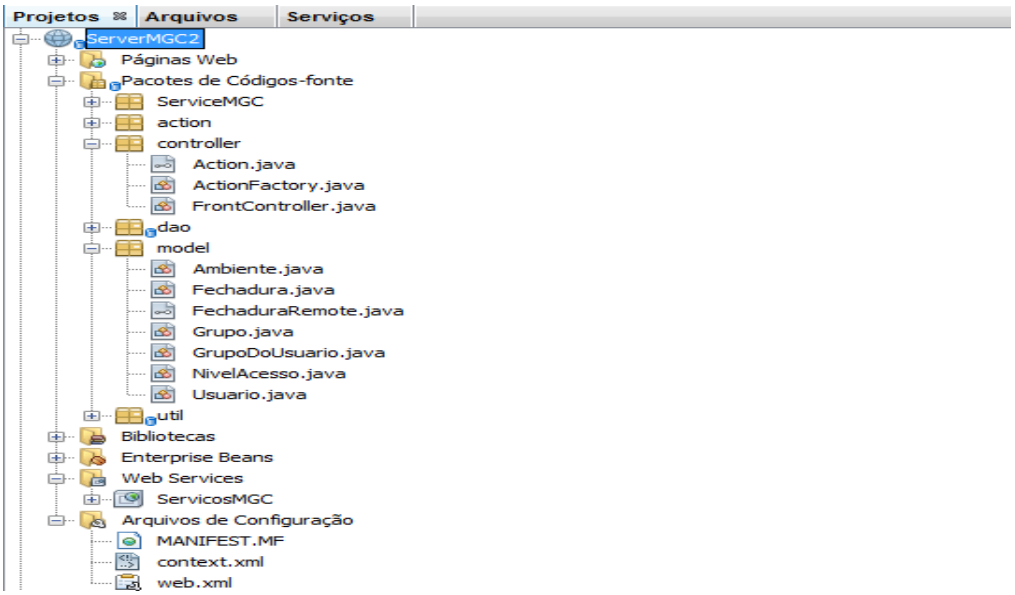

Figura 6. Estrutura do projeto ServidorMGC.

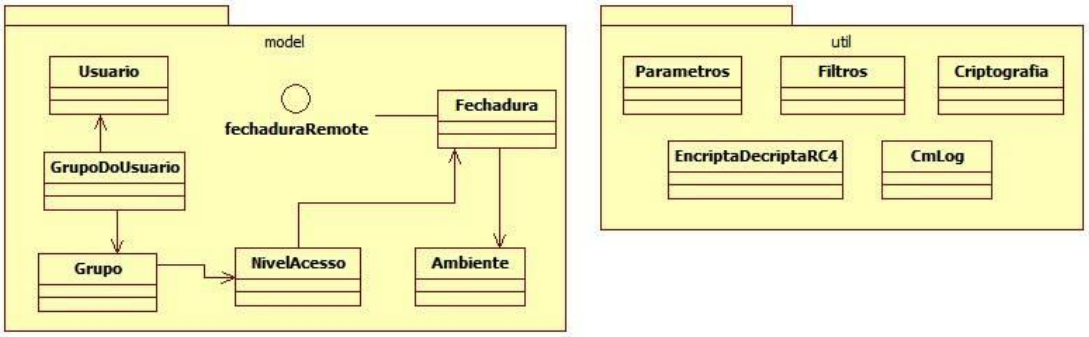

Figura 7. Diagrama de classes ServidorMGC.

Inicialmente a aplicação foi dividida em camadas seguindo o padrão MVC com o intuito de organizar melhor o código e facilitar manutenções posteriores. Separando a camada de visão, que neste projeto estão no pacote páginas Web e se encontram todos os arquivos .JSP, que são responsáveis pela interação com o usuário. As regras de negócio estão no pacote model. E o controle é feito pelas classes do pacote controller.

Para centralizar as requisições foi utilizado o padrão *Front Controller* juntamente com o Command. Basicamente o Front Controller trata todas as chamadas vindas de um site web e é organizado em duas partes: através de um Manipulador Web e uma hierarquia de Comandos. O Manipulador Web é o objeto que efetivamente recebe as solicitações HTTP do tipo POST ou GET do servidor Web. Ele extrai as informações necessárias da URL e das solicitações e então decide que tipo de ação iniciar e por fim delega a um objeto Comando para executar a ação. Vale salientar que tanto o Manipulador Web quanto o objeto Comando são partes do Front Controller. Dessa forma, o Comando escolhe qual Visão (ou página) usar para a resposta. O Manipulador Web tem como única responsabilidade escolher qual Comando executar[Medeiros b, 2015]. Como segue nas imagens a seguir o relacionamento entre *controllers* e a*ctions.*

```
public class FrontController extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String action = request.getParameter("action");
        Action actionObject = null;
        if (action == null || action.equals("")) {
            response.sendRedirect("erro.jsp");
        }
        actionObject = ActionFactory.create(action);
        if (actionObject != null) {
            actionObject.execute(request, response);
        }
    }
```

Figura 8. Classe FrontController.

```java
public class ActionFactory{

    public static Action create(String action) {
        Action actionObject = null;
        String nomeClasse = "action." + action + "Action";
        Class classe = null;
        Object objeto = null;
        try {
            classe = Class.forName(nomeClasse);
            objeto = classe.newInstance();

        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException e) {
            return null;
        }
        if (!(objeto instanceof Action)) {
            return null;
        }
        actionObject = (Action) objeto;
        return actionObject;
    }
}
```

Figura 9. Classe ActionFactory.

Neste ponto a aplicação possui recursos de autenticação, assim como o armazenamento de senhas criptografadas no banco de dados. Porém ainda é possível sabendo sua URL(Uniform Resource Locator) acessar páginas sem nem mesmo ter efetuado login no sistema. Isto é uma falha grave de segurança e para evitar que isso aconteça, foi utilizado controle de sessão implementado com filtros. Os filtros da API(*Application Programming Interface*) *servlet* são responsáveis por interceptar a chamada a certo recurso, fazer algum processamento e liberá-la ou não [Souza 2014]. Abaixo uma parte do código do arquivo web.xml.

```xml
<filter>
    <filter-name>Filtro</filter-name>
    <filter-class>util.Filtro</filter-class>
</filter>
<filter-mapping>
    <filter-name>Filtro</filter-name>
    <url-pattern>*.FrontController</url-pattern>
</filter-mapping>
```

Figura 10. Parte do código do arquivo web.xml.

Desta forma ao fazer uma requisição à aplicação a configuração realizada no arquivo web.xml faz com que seja acionado *o filter-mapping*, chamando a classe filtro e no método doFilter() é verificada a existência do objeto "usuario" na sessão, se verdadeiro é porque o usuário está logado, então o método ChamaAction() libera a execução do action, caso o objeto seja nulo, o usuário é redirecionado para a pagina de login, e nenhuma action é chamada segue figura da classe Fitro responsável por fazer a verificação mencionada.

```
public void doFilter(ServletRequest req, ServletResponse res,FilterChain chain) throws
        IOException, ServletException {

      HttpSession session = ((HttpServletRequest)req).getSession();
      Usuario usuario = (Usuario)session.getAttribute("usuario");

      if(usuario==null){
            session.setAttribute("msg","Você não está logado no sistema!");

            ((HttpServletResponse)res).sendRedirect("../erroLogin.jsp");

      }else{
            chain.doFilter(req, res);
      }
}
```

Figura 11.Método doFilter da classe Filtro.

Todo o acesso a banco é isolado e efetuado pelo padrão DAO. Que trata todas as operações de inclusão, exclusão, edição e consulta a dados este padrão é muito utilizado para desacoplar a persistência dos dados da lógica da aplicação, deixando o código fácil de manter e entender.

4. Conclusão

Este trabalho buscou a especificação e a utilização de um sistema de controle de acesso baseado em Web Services e fechaduras eletrônicas, ao qual foi testado e utilizado em fechaduras reais, desenvolvidas de forma a se integrar ao sistema apresentado, assim visando a melhora no controle de acesso de indivíduos em locais específicos dentro de uma organização, aumentando a segurança e o conforto desses usuários, já que o acesso é dado apenas a pessoas autorizadas e devidamente cadastradas no sistema desenvolvido, tornando a identificação rápida e eficaz.

4.1. Trabalhos Futuros

Como foi bem-sucedido a realização do que o sistema se propõe em si, que é o controle de acesso através de autenticação, alguns pontos podem ser abordados em trabalhos futuros.

É importante trabalhar na segurança do servidor de aplicação em si, com aplicação de políticas de segurança que previnam ataques ao servidor, como a utilização de *firewalls* e outras ferramentas de proteção.

A integração com outros sistemas da organização também é uma possibilidade. No caso de uma organização acadêmica como o IF Sudeste MG, esse serviço pode ser integrado ao SIGA(Sistema Integrado de Gestão Acadêmica), vinculando o acesso de alunos a salas onde são ministradas disciplinas tais que estejam de fato autorizados a cursar. Também constituiria, nesse caso, uma forma de se fazer a “chamada” dos alunos, em que a frequência poderia ser levantada da própria fechadura.

Outro ponto é a restrição de acesso acumulada, baseada não só na autenticação mas em outros critérios, como horários e padrões de acesso. Para isso, pode ser interessante a aplicação de métodos de inteligência computacional nos dados gerados pelo sistema, buscando pela existência de padrões e comportamentos relevantes na

utilização do sistema. Para isso, ferramentas de *Business Intelligence* podem ser adequadas.

Por fim, a própria política de acesso aos logs pode ser incrementada, com a adoção de filtros que otimizem as buscas e, desta forma, a obtenção de informações mais específicas.

Referências Bibliográficas

Almeida, Maurício Barcellos(2002) "Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares" Disponível em: <http://www.scielo.br/pdf/ci/v31n2/12903>. Ci. Inf., Brasília, v. 31, n. 2, p. 5-13, maio/ago. 2002. Acesso em: 30 Dez. 2015.

Alvarez, Miguel Angel (2014) "O que é JSP" Disponível em: <http://www.criarweb.com/artigos/227.php>. Acesso em 01 Jan 2016.

Araújo, Everton Coimbra (2010) de "CRUD com JSP" Disponível em: http://www.linhadecodigo.com.br/artigo/2997/crud-com-jsp.aspx. Acesso em 02 Jan. 2016.

Bastos , Alex Vidigal (2013) "Apresentação –Sistemas Embutidos" Disponível em: <http://www.decom.ufop.br/alex/arquivos/bcc425/slides/introducao_sistemas_embarcados.pdf>. Acesso em 02 Jan 2016.

Belem, Thiago(2013) "Mas afinal, o que é o MVC?"

Disponível em: http://blog.thiagobelem.net/o-que-e-o-mvc/. Acesso em 01 Jan. 2016.

Dantas,Daniel Chaves Toscano (2007) "Simple Object Access Protocol (SOAP)",

Disponível em: <http://www.gta.ufrj.br/grad/07_2/daniel/>. Acesso em: 29 Dez. 2015.

Delai, Andre Luiz(2013) "Sistemas Embarcados: a computação invisível"

Disponível em: <http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/conceito.html>. Acesso em: 28 Dez. 2015.

DENARDIN, Gustavo Weber "Microcontroladores", Disponível em: <http://www.joinville.udesc.br/portal/professores/eduardo_henrique/materiais/apostila_micro_do_Gustavo_Weber.pdf>. (Apostila). Acesso em: 02 Jan. 2016.

Evans, Dave(2011), "Internet das Coisas Como a próxima evolução da Internet está mudando tudo" Disponível em: <http://www.cisco.com/web/BR/assets/executives/pdf/internet_of_thin _0411final.pdf>. Acesso em: 10 Jan. 2016.

Luz, Marlon; Humenhuk, Lucas (2015) "Microsoft Virtual Academy: A Internet das Coisas – Fundamentos de IoT", Disponível em: <https://www.microsoftvirtualacademy.com/pt-br/training-courses/a-internet-das-coisas-fundamentos-de-iot-12622>. Acesso em: 28 Dez. 2015.

Gamma Erick; Helm, Richard; Johnson, Ralph; Vlissides, John (2005) "Padrões de projeto". Porto Alegre, RS Bookman.

Heitlinger , Paulo (2001) "Guia Prático da XML" Portugal, Centro Atlântico.

ORACLE , Java Documentação (2015) "O que é a Tecnologia Java e porque preciso dela?", Disponível em: <https://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em: 02 Jan. 2016.

Marques ,Gustavo(2015) "Reusabilidade com Front Controller e Padronização com Command", Disponível em: <http://programadorprofissional.blogspot.com.br/>. Acesso em: 07 Jan 2016.

Marinho, Sérgio (2004), "Segurança da Informação - Conceitos e Modelo de Gestão" Disponível em: <http://www2.dem.inpe.br/ijar/Auditoria%20de%20SI/Aulas/SegInform.pdf>. Acesso em: 17 Jan. 2016.

Medeiros a, Higor() "Utilizando Criptografia Simétrica em Java", Disponível em: <http://www.devmedia.com.br/utilizando-criptografia-simetrica-em-java/31170>.Acesso em: 15 Jan.2016.

Medeiros b, Higor (2015) "Padrões de Projetos: Introdução aos Padrões Front Controller e Command", Disponível em: < http://www.devmedia.com.br/padroes-de-projetos-introducao-aos-padroes-front-controller-e-command/30644>. Acesso em: 02 Jan. 2016.

MySQL, manual "Visão Geral do Sistema de Gerenciamento de Banco de Dados MySQL", Disponível em: <http://mysql.spd.co.il/doc/refman/4.1/pt/what-is.html>. Acesso em: 20 Dez. 2015.

Nascimento, Rodrigo (2015) "O que, de fato, é internet das coisas e que revolução ela pode trazer?", Disponível em: <http://computerworld.com.br/negocios/2015/03/12/o-que-de-fato-e-internet-das-coisas-e-que-revolucao-ela-pode-trazer>. Acesso em: 28 Dez. 2015.

SOUZA, D.J(2000) "Desbravando o PIC:Baseado no microcontrolador PIC" São Paulo, Érica.

Pamplona, Vitor Fernando (2012)"Tutorial Java 3: Orientação a Objetos" Disponível em: <http://javafree.uol.com.br/artigo/871497/Tutorial-Java-3-Orientacao-a-Objetos.html>. Acesso em: 27 Dez. 2015.

Pereira, Dani Edson(2011) "O que é esse tal de XML?" Disponível em: < http://programandofacil.com.br/artigos/xml/o-que-e-esse-tal-de-xml/ >. Acesso em: 06 Jan. 2016.

Perry,J Steven(2011) "Introdução à Programação em Java, Parte 1: Fundamentos da linguagem Java", Disponível em: <https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/: Acesso em: 02 Jan. 2016.

Rocha, Fabio Gomes(2014) "Introdução ao Java Server Pages – JSP", Disponível em: <http://www.devmedia.com.br/introducao-ao-java-server-pages-jsp/25602>. Acesso em: 30 Dez. 2015.

Rolim,Carlos Oberdan (2005) "Tópicos de Sistemas de Informação A", Disponível em: <http://www.san.uri.br/~ober/arquivos/disciplinas/tsia/aulas/tsi_uddi.ppt>. Acesso em: 05 Jan. 2016.

Sampaio,Ericksen Viana(2012) "Criptografia: Conceito e aplicações - Revista Easy Net Magazine 27", Disponível em: <http://www.devmedia.com.br/criptografia-conceito-e-aplicacoes-revista-easy-net-magazine-27/26761>. Acesso em: 10 Jan. 2016.

Santiago, Emerson(2012) "Criptografia" Disponível em: <http://www.infoescola.com/informatica/criptografia/>. Acesso em: 12 Jan. 2016.

Benetti, Ticiano(2015), Disponível em: <http://www.profissionaisti.com.br/author/ticiano/>. Acesso em 17 Jan. 2016.

Souza, Jair Elton Batista(2014) “Controle de acesso com filtros”, Disponível em: <http://www.devmedia.com.br/controle-de-acesso-com-filtros/1720>. Acesso em: 20 Dez. 2015.

Stallings, William (2008) “Criptografia e segurança de redes: Princípios e práticas”, 4 ed. São Paulo: Prentice Hall.

W3C a, ”Web Services Architecture” (2004), Disponível em: <http://www.w3.org/TR/ws-arch/#introduction>. Acesso em 30 Dez. 2015.

W3C b, (2001) “WSDL, Specification”, Disponível em: <http://www.w3.org/TR/wsdl>. Acesso em: 19 Dez. 2015.